Comparative Study of Query Performance in a Remote Health Framework using Cassandra and Hadoop

Himadri Sekhar Ray[1], Kausik Naguri[1], Poly Sil Sen[2] and Nandini Mukherjee[1]
[1]*Department of Computer Science and Engineering, Jadavpur University, Kolkata, India*
[2]*Department of Information Technology, Techno India, Salt Lake, Kolkata, India*

Keywords: Electronic Health Record, Big Data, Sensors, Hadoop, Cassandra.

Abstract: With the recent advancements in distributed processing, sensor networks, cloud computing and similar technologies, big data has gained importance and a number of big data applications can now be envisaged which could not be conceptualised earlier. However, gradually as technologists focus on storing, processing and management of big data, a number of big data solutions have come up. The objective of this paper is to study two such solutions, namely Hadoop and Cassandra, in order to find their suitability for healthcare applications. The paper considers a data model for a remote health framework and demonstrates mappings of the data model using Hadoop and Cassandra. The data model follows popular national and international standards for Electronic Health Records. It is shown in the paper that in order to obtain an efficient mapping of a given data model onto a big data solution, like Cassandra, sample queries must be considered. In this paper, health data is stored in Hadoop using xml files considering the same set of queries. Next, the performances of these queries in Hadoop are observed and later, performances of executing these queries on the same experimental setup using Hadoop and Cassandra are compared. YCSB guidelines are followed to design the experiments. The study provides an insight for the applicability of big data solutions in healthcare domain.

1 INTRODUCTION

It is not always possible for a patient to get a doctor at the time and place of requirement. Moreover it is not always possible for a doctor to be physically present with all patients at the time and place of requirement. This situation particularly may occur in case of remote villages, hilly areas and in other places. The reason is that the ratio of patients to doctors is very poor in such areas. A remote health framework is proposed in (Mukherjee, 2014). The framework helps to monitor health parameters of a patient from a distance. Doctors can provide remote guidance to these patients. Sensors are used for continuous monitoring of patients' health on advice of doctors. This sensor data is to be stored.

Health data is huge in volume. If we include sensor observation data as part of health data, it becomes more voluminous.

Health data may consist of text (as in history of a patient), images (as in X-Ray or USG), audio (as in heart beat), video (as in ECG) or many other formats. Thus, different data structures may be needed to store different parts of health data and therefore, health data usually have variety.

Health data contains sensor observation data that is generated and sent very fast. It has to be processed fast as well. Thus velocity is another characteristics of sensor data.

Furthermore, sensor observation data may also have uncertainty and loss of accuracy is another feature of such type of data due to various obstacles and environmental situations. Thus, it is not trustworthy always. Hence, veracity is another property. Having all the above features of big data, it can be suggested that usual database schemas are not effective for health data and this data be stored in some big data solutions.

However, in the present scenario, choice can be made among a large number of big data solutions. It is suggested that big data be stored in a NOSQL database, like Cassandra or MongoDB. On the other hand, Hadoop provides a Map-Reduce framework that enables distributed processing of large data sets across clusters of commodity servers. A big data solution must be judged on the basis of its operational ease, cost effectiveness and performance

Ray, H., Naguri, K., Sen, P. and Mukherjee, N.
Comparative Study of Query Performance in a Remote Health Framework using Cassandra and Hadoop.
DOI: 10.5220/0005706803300337
In *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2016) - Volume 5: HEALTHINF*, pages 330-337
ISBN: 978-989-758-170-0

scalability.

The purpose of this paper is to study implementation of a data model in Cassandra and in Hadoop. The data model cannot be directly implemented in big data solutions. It is preferable that a set of queries are considered for deciding efficient storage of data in big data solutions. Thus, we present mappings from the proposed data model to data models suitable for Cassandra and Hadoop. Once this mapping is done, a comparative study is made for running queries in Hadoop and Cassandra.

The remaining part of the paper is organised as follows. Section 2 discusses some big data solutions. Section 3 describes a data model for our remote healthcare framework. Section 4 and 5 discuss the mappings of the data model in Cassandra and in Hadoop-linked xml files respectively. Section 6 presents the experimental setup and the results. Finally, Section 7 concludes with a direction of future work.

2 BIG DATA SOLUTIONS

In this section, two big data solutions, namely Cassandra and Hadoop, are described as they have been used in this research work.

Cassandra NoSQL Database: Cassandra uses column family to store distributed NOSQL data. It provides high read throughput. Search latency is low in it because of replication of data across multiple nodes with a replication factor. Column family is similar to tables of RDBMS. Column families have schema but that can be dynamic and every record may not contain all fields of data.

Row key of Cassandra can be formed using multiple keys. The first portion of such composite key is the partition key.

Hadoop Map-Reduce Framework: Hadoop is an open source framework to provide scalability, reliability to distributed computing jobs. Hadoop follows a master-slave architecture with a master or name node and many slave or data nodes in a cluster. Name node manages namespace of a file system and controls access of files by clients. Files are divided into blocks and these blocks are mapped to different data nodes by the name node. Name node maintains meta data. Data nodes create, replicate and delete blocks and also read and write data following instructions of name node. Data nodes have access to one or more local disks. Scale in and scale out are done by adding and removing data nodes. There is a secondary name node that does housekeeping work for the name node.

In Hadoop Map-Reduce framework, the Map function divides a task into subtasks that can be executed in parallel. Reduce function combines the output of the map phase into an output form that is the output of the complete task.

Job tracker runs on the name node that accepts jobs from clients, schedules them on data nodes and gets status report of the jobs running. It gets information of number of slots available in map and reduce slots. One job tracker runs on a name node. Task tracker runs on data or worker node. It accepts task from job tracker and reports job status to it. Communication among job tracker and task tracker uses RPC for communication.

If one of the parallel jobs fails due to storage or processing failure then it can be completed on some other node as Hadoop manages component failure by replicating data on different nodes.

Map-Reduce framework can execute on a file system which is considered as unstructured, though it can also work on structured databases. Map-Reduce ensures locality of data through processing data on local nodes. In this work, health data is stored in xml files for use in Hadoop framework.

3 A DATA MODEL FOR HEALTHCARE DOMAIN

In this section, a data model (Sil Sen, 2014) for healthcare delivery is presented. The data model is developed on the basis of guidelines provided by Indian Standard for Electronic Health Records approved by Ministry of Health and Family Welfare, Government of India (2013) and also other popular international standards of Electronic Health records and Electronic Medical Records. The member variables in each of the classes are added following the above mentioned standards. In contrast to many other health data models published in the literature, our data model (Sil Sen, 2014) is hierarchical. This data model can be improved further to address complexities of Electronic Health Record. The class diagram is shown in Figure 1.

A brief description of the important classes is provided in this section. However, the data members are not shown, because of space limitation. A *patient* may have one or more *complaints*. A *complaint* is treated by one or many *treatment_episodes*. A *treatment_episode* is handled by a *doctor*. A *doctor* may be engaged in one or more *treatment_episodes*. A *treatment_episode* contains one or more *visit*s. In a *visit*, *doctor* may prescribe one or more *investigations*. *Investigation* can be of two types –

continuous_monitoring and *discrete_monitoring*. *Discrete_monitoring* can be either *one_time_investigation* (such as X-Ray) or *periodic_investigation* (such as blood pressure monitoring). *Continuous_monitoring* is done using body sensor networks (not shown in the diagram). Diagnosis may be concluded in a *Visit*. *Treatment_plan* is prescribed in a *Visit*.

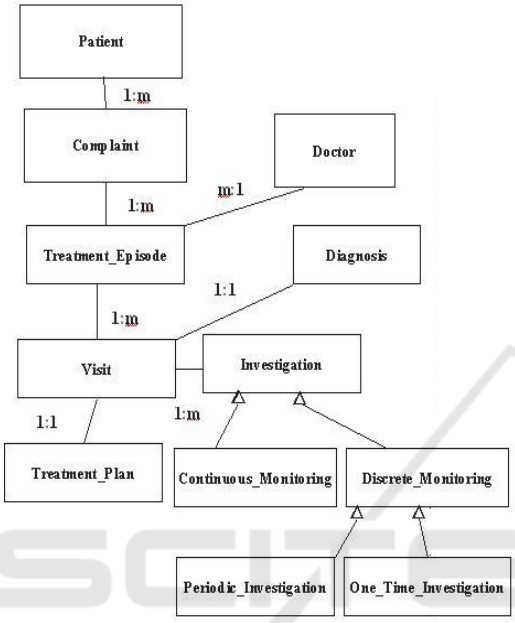

Figure 1: Proposed data model for Remote Health framework.

4 STORING HEALTH RECORD IN CASSANDRA

There is no formal method to migrate a data model from one form to another form suitable for a big data datastore. It has been felt that every big data solution has its own advantages and disadvantages and signature characteristics. Implementation of a data model in a big data solution requires trial and error depending on few factors. These factors are:

- Nature of the application and its requirements involving type of data;
- The queries that the system will execute;
- Performance requirement of these queries;
- Amount of data involved;
- Features of the target big data solution;
- Statistics related to the queries like the frequency of running a query.

In the remote health framework presented in (Mukherjee, 2014), the following characteristics of health data have been observed:

- data is stored in various formats, like text, image, audio, video and sensor observed data;
- some part of health data changes frequently, while some other part hardly changes, e.g. demographic information. Thus every part of health data is not read/ written with same frequency;
- huge volume of sensor data are generated at high velocity and with uncertainty and are stored as part of health data.

To implement the data model, initially all fields are kept together in denormalized form. Later, depending on different queries from the designed set of queries, the initial column family is broken into more than one column families. A data field may be replicated several times in different column families to improve query performance. It is to be noted that, in contrast with RDBMS which tries to avoid redundancy of data, in many big data solutions data redundancy is preferred to improve availability and performance. The entire procedure is repeated until the query performance reaches the desired level

Patel (2012) discussed some best practices of Cassandra. These are discussed below.

- Data is to be kept in denormalized form to improve the query performance. Cassandra has data compression facility embedded within it to manage redundancy of data.
- Join operation incurs overhead and therefore is avoided in Cassandra. The seek time also increases after joining.
- Read heavy data is to be kept separately from write heavy data. The query performance improves if a proper row key is chosen.
- Finally, if all the fields related to a query is kept in one partition, then Cassandra performs well. However, this leads to redundancy in data.

Dynamic column family, super column family and indexes are avoided as they are inappropriate here. Index ensures order on one field. In real life situation, all the required search order cannot be based on one index. Performance using indexes has shown poor performance (Naguri, 2015). Therefore indexes are avoided in this implementation.

Some sample queries are considered as follows: specialization wise doctor search, patient wise information (demographic, food habit etc.) search, patient wise complaint search, complaint wise treatment search, treatment wise investigation search.

Initially, in order to facilitate the patient-related

information search and doctor-related information search, two column families are designed, these are *Patient* and *Doctor*. It may be noted that these queries mostly search data that are less frequently updated. For rest of the queries, the *Treament_Episode* and the *Visit* column families along with one separate column family for sensor data are created. . Relevant medical information related to a specific complaint is kept in *treatment_Episode* and *Visit* files. *Visit* file also contains all generally observed medical parameters like pulse rate etc.

Initially sensor data has been stored in *Visit* column family. But this degrades the query performance and as size of file increases time out happens. Therefore, the above decision was made.

5 STORING HEALTH RECORD IN HADOOP

The data model discussed in the previous section is converted to a structural form for analysis of data through Hadoop. The classes are stored in xml as it has a standard interoperable format. The data are fetched from the database using CQL/SQL and connected to a java program to build large xml file to run analysis. The important attributes of patient column family are shown in Table 1.

Table 1: Patient File.

Patient
patient_id,
patient_name,
patient_dob,
alternate_patient_ids,
patient_gender,
patient_occupation,
patient_address_type,
patient_address,
patient_email_id,
patient_phone_ll,
patient_mob,
Emergencycontactperson _name,
Emergencycontactperson _relation,
Emergencycontactperson_address,
pt_foodhabbit,

XML File Description for Patients:

```
<?xml version="1.0" encoding="UTF-
8"?>
<properties>
 <pt_id> Patient Unique ID </pt_id>
 <pt_name> Patient's  name
</pt_name>
 <pt_dob> Patient's date of birth
</pt_dob>
  :   :   :   :   :
  :   :   :   :   :
<properties>
```

For the remaining files, their respective fields are given in Tables 2, 3, 4 and 5. It must be noted that the files do not exactly correspond to the classes shown in Figure 1. Based on the discussion in Section 4, data fields are repeated in different files, so that a file is able to produce answers to all the queries considered initially.

The large volume of health records is stored in Hadoop file system using these tables. Map-Reduce programs are written for retrieval and analysis of data that are run on Hadoop file system.

Table 2: Doctor File.

Doctor
doctor_id
doctor_name
doctor_type
doctor_specialization
doctor_organization
doctor_address_type
doctor_address
doctor_email_id
doctor_mob

Table 3: Treatment_Episode File.

Treatment_Episode
treatment _episode_id,
patient_id,
patient_name,
patient_address,
patient_gender,
patient_food_habbit,
patient_allergy,
blood_group,
doctor_id,
doctor_name,
doctor_specialization,
start_date,
is_active,
complaint,
diagnosis,
clinical_status

Table 4: Visit File.

Visit
visit_id,
patient_id,
patient_name,
patient_address,
patient_food_habbit,
allergy,
blood_group,
treatment_episode_id,
treatment_start_date,
initial_complaint,
visit_time,
reason,
systolic_bp,
diastolic_bp,
pulse_rate,
temp,
respiration_rate,
height_cms,
weight_kgs,
investigations,
investigation_reports,
medication,
diagnosis,
therapies
surgeries,
current_status

Table 5: Sensed_data File.

sensor_data
patient_id,
treatment_episode_id,
visit_id,
parameter,
data
....

6 EXPERIMENTAL RESULTS

(A) Experimental Setup – This paper presents two sets of experiments – (i) using Map-Reduce programs in Hadoop with varied amount of data on different number of Hadoop nodes, (ii) varying the amount of data, while keeping the number of Hadoop nodes fixed (16 here) and comparing the results with Cassandra implementation in the same environment.

All experiments are performed on a 16 node cluster. Each node has the configuration of 2.93 GHz Intel Core2 Duo Processor with 3GB RAM and 1 TB storage space.

From the initial set of queries (considered for designing the databases), a subset of queries is chosen with the support of YCSB guidelines (research.yahoo.com/files/ycsb.pdf). Eight simple queries and one statistical 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 a patient.
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.
6. Patient treatment case history for a patient.
7. Find all the doctor's names who are attached with an organization.
8. Find all the complaint details of a Patient.
9. Retrieving sensor data for a particular patient observed during one hour time.

The statistical query is

1. Find the number of babies with age is between 0 and 5 and suffering from malnutrition.

Each time number of data records in respective files is varied from 50 to 800,000 of records and number of Hadoop nodes is varied from 1 to 16. Every experiment is performed three times for each query and the average of the three results is taken.
The same queries are executed in Cassandra on 16 nodes. Data volume is varied as in the previous case.

(B) Results - (i) Experiments running on different number of Hadoop nodes: Based on the above experimental setup, a study is made on the data retrieval time with varied amount of data stored in Hadoop. All the nine queries and the statistical query are run. Due to space limitation, only the results of query 1 (Figure 2), query 9 (Figure 4) and the statistical query (Figure 3) are presented in this paper.

DISCUSSION

It is observed that while executing the queries is Hadoop, higher number of slaves does not signify better performance. Optimal performance is obtained when a particular load is given to a particular number of slaves. For example, the performance of the implementation with 8 nodes is poor, even worse than one or two slaves.

It is also observed that performance on 16 slaves improves and become better in comparison with others when number of records is high (above

500,000). Thus, with low volume of data, Hadoop Map-Reduce code on different number of slaves has almost no effect.

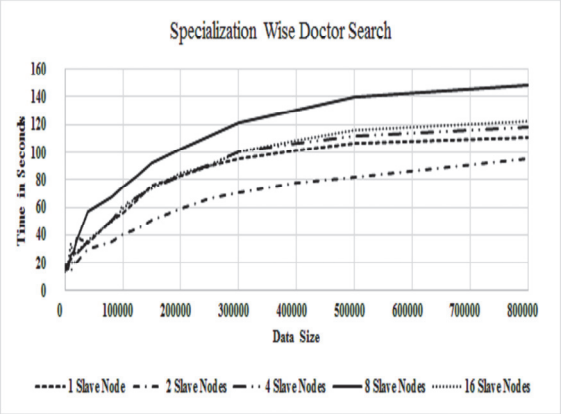

Figure 2: Query performance of *specialization-wise doctor search* in Hadoop.

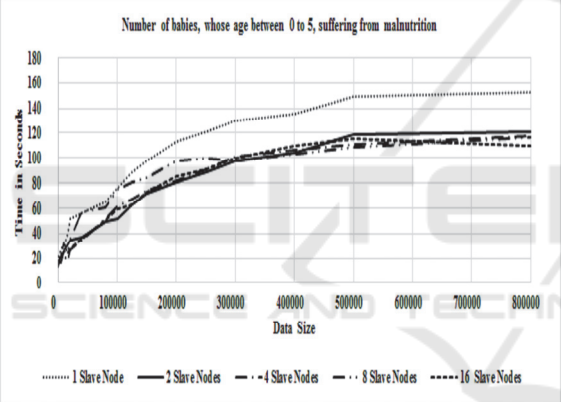

Figure 3: Query performance of *number of babies between age 0 and 5 suffering from malnutrition.*

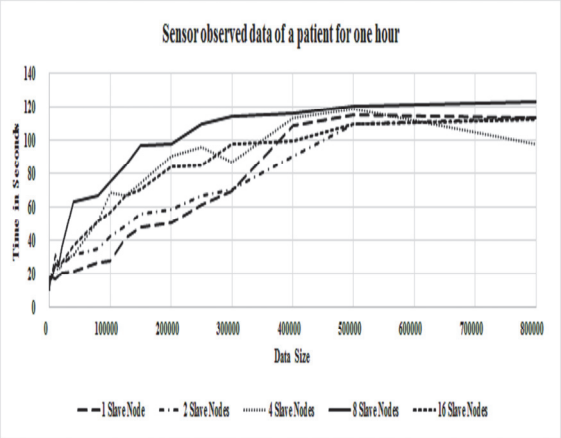

Figure 4: Query performance of Sensor observation data of a patient during an hour.

The above observations can be explained from the study presented in (Guo, 2012) (Bezerra, 2013). It has clearly been shown that Hadoop scheduling is non-optimal. The scheduling algorithm in Hadoop for the *Map* tasks considers data locality, but overlooks other factors such as system load and fairness.

(ii) Comparative study between Hadoop and Cassandra: Same set of queries as given in Section 6 are executed on the data stored in Cassandra with varying number of records in respective column families corresponding to each query. Number of nodes in the set up is taken as 16.

Figures 5, 6, 7 and 8 depict the results of the comparison for queries 1, 2, 3 and 4 respectively. Data retrieval time required by Hadoop is shown as **th** and time required by Cassandra is shown as **tc**. Number of records shown is in 100,000.

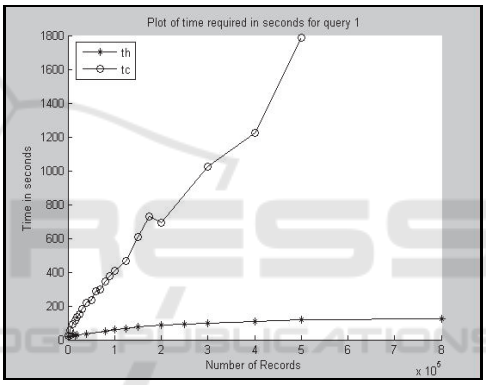

Figure 5: Execution of query1 in Cassandra and Hadoop.

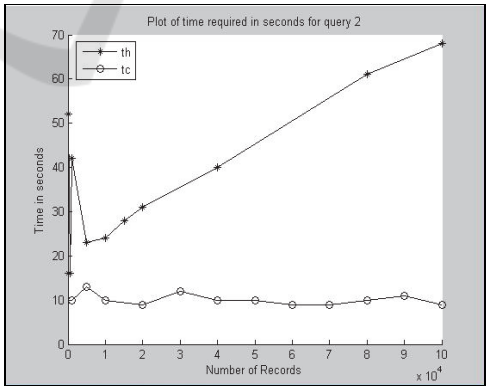

Figure 6: Execution of query2 in Cassandra and Hadoop.

Figure 5 depicts the retrieval time for doctors' names and their details with specialization string 'cardiologist'. It is clear that Hadoop performs much better as number of records increases.

Performance of query 2 is shown in Figure 6.

Here Patient_id is used as the row key, the time of retrieving patient information decreases with increase in number of records in Cassandra. For this query, Cassandra performs better than Hadoop.

Performance of query 3 is shown in Figure 7. Time of accessing ECG data of a patient increases slowly with increase in number of records. Hadoop performs much better than Cassandra.

Performance of query 4 is shown in Figure 8. Time to retrieve records of a diabetic patient increases with number of records.

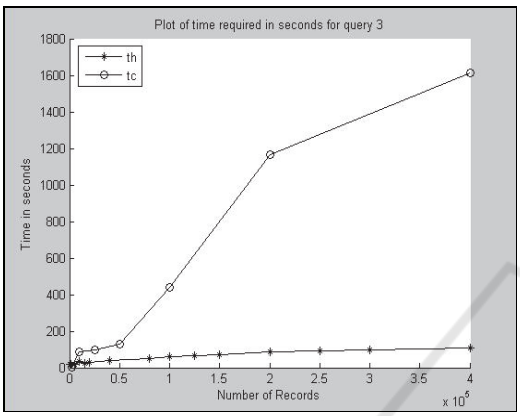

Figure 7: Performance of query 3 in Cassandra and Hadoop.

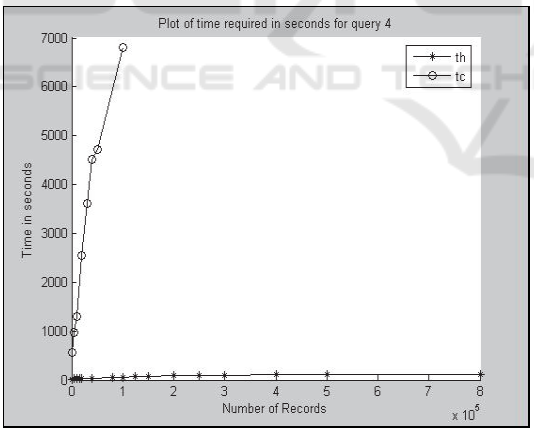

Figure 8: Performance of query 4 in Cassandra and Hadoop.

DISCUSSION

The above results show that Hadoop performs better in case of all queries leaving one. The performance of Cassandra improves significantly with increase in number of records for patient wise information search (Figure 6). Cassandra, when compared with Hadoop in this case, performs well in spite of the fact that Hadoop is well known for large volume of data processing. In case of query 2, the Patient_id is used as the row key (similar to primary key in RDBMS), and the query is made on the basis of the primary key. Therefore, data retrieval becomes fast. In other cases, queries do not involve the row key and therefore query performance is poor.

Cassandra particularly performs poorly in case of query 4. The data mapping strategy can be responsible for this. There are two conflicting goals in Cassandra data mapping – (i) data must be spread evenly around the cluster, and (ii) the number of partitions read must be minimized. On the basis on these two goals, a trade-off is to be made while implementing in Cassandra for a set of queries. Requirements for one query may be conflicting with another leading to performance degradation.

The study further demonstrates that Hadoop has the capacity to work with voluminous data set as expected.

7 RELATED WORK

With the recent advancements in sensor networks and cloud computing, scientists have focussed on application of big data concepts in healthcare domain. However, much of the contemporary research works actually consider big data analytics utilizing large volumes of medical data and combining multimodal data from disparate sources (Belle, 2015). Only few aim at studying various big data tools in order to develop an insight about suitability of these tools for healthcare data storage, processing and management. A comparative study of NOSQL databases is provided in (Manoj, 2014). However, the authors only provide a theoretical study of architecture and internal working of Cassandra, MongoDB and HBase and carry out an evaluation of Cassandra for performance and concurrency in comparison with relational databases within only a limited scope. Lourenço et al (Lourenço, 2015) provides a detailed study on various big data solutions including Cassandra, HBase etc. However, the study is based on literature review and no performance evaluation is presented. The architecture for a distributed sensor data collection, storage, and analysis system developed using Hadoop in a cloud environment is presented in (Aydin, 2015). Performance of this implementation is also observed. However, this work does not address the specific issues in healthcare domain. Performance of Hadoop is also studied in (Guo, 2012) and (Bezerra, 2013) for generic applications.

Our work is different from the above mentioned research works, because we consider the specific

issues related to healthcare applications and carry out experimental study regarding the performance of two big data solutions with particular consideration of healthcare applications.

8 CONCLUSIONS

This paper has studied the performance of two big data solutions in healthcare domain. We have considered a healthcare data model proposed on the basis of national and international EHR standards and demonstrate its mapping onto two big data solutions, namely Cassandra and Hadoop. The performance of a representative set of queries has been studied using Hadoop. Further, we have also carried out a comparative study of query performance in Cassandra and in Hadoop systems.

It is observed that the mapping strategy is an important issue in improving performance of any big data solution. However, this should be based on some queries which are considered to be frequent in the application domain. It is also observed that Hadoop performs better with large data sets in comparison with Cassandra. The observations from the experimental study clearly establish the fact that Hadoop is more focussed on data processing and therefore, scheduling strategy is important in case of Hadoop. On the other hand, because data storage and distribution is the main goal in Cassandra, implementation is Cassandra should consider the data mapping strategy as the primary issue.

In future, our goal is to develop algorithms for transformation of data models to big data solutions. In order to combine the strengths of Hadoop (in terms of data processing) and Cassandra (in terms of storage), it is planned to extend the work to use Hadoop integrated with Cassandra. Other big data solutions, like Mongodb and Hbase may also be considered and their performances can be compared. It is interesting to investigate the impact of other file formats used to store data on the performance of Hadoop Map-Reduce framework.

REFERENCES

Aydin. G., Hallac I.R., and Karakus B. (2015) Architecture and Implementation of a Scalable Sensor Data Storage and Analysis System Using Cloud Computing and Big Data Technologies. Journal of Sensors, Volume 2015, Article ID 834217, Hindwai Publishing Corporation.

Belle A., Thiagarajan R., Soroushmehr S.M.R., Navidi F., Beard D.A., and Najarian K. (2015) Big Data Analytics in Healthcare, BioMed research international. Volume 2015, Article ID 370194, Hindwai Publishing Corporation.

Bezerra A., Hernández P., Espinosa A., and Carlos J. (2013) Job scheduling for optimizing data locality in Hadoop clusters. Proceedings of the 20th European MPI Users' Group Meeting (EuroMPI'13). ACM, New York, NY, USA, pp 271-276.

Guo Z., Fox G., and Zhou M. (2012) Investigation of data locality and fairness in MapReduce, In Proceedings of third international workshop on MapReduce and its Applications Date, pp. 25-32. ACM.

Lourenço J.R., Cabral B., Carreiro P., Vieira M., and Bernardino J. (2015) Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data, 2 (1), pp 1-26.

Manoj V. (2014) Comparative study of NoSQL Document, Column Store Databases And Evaluation Of Cassandra. International Journal of Database Management Systems, 6 (4), pp11-26.

Ministry of Health and family Welfare, Government of India (2013) Approved “Electronic Health Record Standards for India”, August 2013.

Mukherjee, N., Bhunia, S. S., and Sil Sen, P. (2014) A Sensor-Cloud Framework for Provisioning Remote Health-Care Services. Proceedings of the Computing & Networking for Internet of Things (ComNet-IoT) workshop co-located with 15th International Conference on Distributed Computing and Networking.

Naguri, K., Sil Sen P., Mukherjee, N. (2015) Design of a Health-Data Model and a Query-driven Implementation in Cassandra, Proceedings of the 3rd International Workshop on Service Science for e-Health (SSH), co-located with IEEE HealthCom.

Patel J. (2012) (Online)
www.ebaytechblog.com/2012/07/16/cassandra-data-modeling-best-practices-part-1/ & -part-2/

Sil Sen, P., Mukherjee, N. (2014) Standards of EHR and their scope of implementation in a sensor-cloud environment, Proceedings of the international Conference on Medical Imaging, m-health and Emerging Communication System (MedCom), IEEE, pp241-246.