SMGroup

SM Journal of Biometrics & Biostatistics

Research Article

Interactive Big Data Analytics Platform for Healthcare and Clinical Services

Chrimes D[1]*, Kuo MH[2], Moa B[3] and Kushniruk AW[2]

[1]*Database Integration and Management, IMIT Quality Systems, Vancouver Island Health Authority, Canada*
[2]*School of Health Information Science, University of Victoria, Canada*
[3]*Compute Canada/WestGrid/University Systems, University of Victoria, Canada*

Article Information

Received date: May 31, 2018
Accepted date: Jun 11, 2018
Published date: Jun 15, 2018

***Corresponding author**

Chrimes D, Database Integration and Management, IMIT Quality Systems, Vancouver Island Health Authority, Canada, Email: dillon.chrimes@viha.ca

Keywords Adaptable Architectures; Big Data; Data Mining; Distributed Filing System; Distributed Data Structures; Healthcare Informatics; Hospital Systems; Metadata; Relational Database

Abstract

The study objective is to establish an interactive Big Data Platform Analytics (BDA) platform with Hadoop/MapReduce technologies distributed over HBase (key-value NoSQL database storage) and to generate hospitalization metadata on the platform. Performance tests retrieved results from simulated patient records with Apache tools in Hadoop's ecosystem. At optimized iteration, the Hadoop distributed file system (HDFS) ingestion with HBase exhibited sustained database integrity over hundreds of iterations; however, the platform required a month to complete its bulk loading via MapReduce to HBase and validate queries required a month. To generate HBase datafiles, the framework took a week for one billion (10TB) files and a month for three billion (30TB) files. Inconsistencies of MapReduce limited the capacity to generate/replicate data efficiently. Dependencies among the data elements system could be expressed via "family" primary keys set in code via Apache Phoenix as database generator. Modeling a hospital system based on a patient encounter-centric database was very difficult because data profiles were fully representative of complex relationships. Apache Spark and Apache Drill showed high performance. Recommendations regarding key-value storage should be considered when analyzing large volumes of healthcare data securely.

Introduction

The term BigData refers to the collection of large, complex, distributed data at an extremely fast rate; its shorthand formula is the "5Vs": volume, variety, velocity, veracity and value [1,2]. In each of its attributes, Big Data has high potential for unlocking new sources of economic value, providing fresh insights into scientific discoveries, and assisting in policy making [3]. However, Big Data is not practically useful until it can be aggregated and integrated in a manner that computer systems can use to generate knowledge. As data originally reside in individual silos, or across multiple domains that are not integrated, each source of knowledge serves a very specific localized purpose in the system or else is comprised of data entries for viewing only. However, the combination of data silos across data warehouses can provide new insights into important problems. Particularly, correlation of data from multiple datasets can result in the identification of individual events and phenomena, as well as in the creation of profiles that can be used to track activities proactively.

The application of Big Data to healthcare is different from its application in other areas, such as social networks or business, in that healthcare data includes structured Electronic Health Records (EHRs), coded data from such sources as the International Statistical Classification of Diseases (ICD) and the Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT), semi-structured data (e.g. HL7 messages), unstructured clinical notes, medical images (e.g. MRI, X-rays), genetic and laboratory data and other types of data (e.g. patient registration, public health and mental health data). Huge volumes of very heterogeneous raw data are generated by a variety of health information systems, such as EHR, Computerized Physician Order Entry (CPOE), Picture Archiving and Communication Systems (PACS), Clinical Decision Support Systems (CDSS) and Laboratory Information Systems (LIS). These system types are used in many distributed healthcare settings, such as hospitals, clinics, laboratories, and physician's offices. Several published studies have asserted that if Big Data in healthcare is managed well, it can drastically improve patient care and reduce costs [3-6]. A McKinsey Global Institute study suggests, "If US healthcare were to use Big Data creatively and effectively to drive efficiency and quality, the sector could create more than $300 billion in value every year" [7]. Shah and Tenenhaum [8] believe that Big Data-driven research will enable the discovery of new treatments for diseases. Garrison [9] also proclaims that Big Data will benefit population health and improve decision making.

While the potential value of Big Data in healthcare has been widely discussed, real-world applications and even simulated applications have rarely been attempted. Applications involving data from hospital systems have seldom been implemented. In our study, we designed and implemented a framework using Big Data technologies for supporting Big Data Analytics (BDA) in healthcare. In that framework, emulated patient data was distributed using the Hadoop Distributed

File System (HDFS) to simulate the interactive usability of extremely large volumes of data, representing billions of patient encounters across multiple hospitals. To achieve this scale, three to nine billion patient records were constructed and cross-referenced with data profiles of metadata in existing data warehouses at the Vancouver Island Health Authority (VIHA), headquartered in Victoria, British Columbia, Canada. The resulting data set was then validated via workflow to retrieve results from patient queries. In general terms, the simulation was a proof-of-concept implementation of the use of real patient data on a BDA platform.

Literature Review

Challenges in Big Data Analytics

BDA is the process of extracting knowledge or data mining from sets of Big Data [10]. Wang et al. [11] describe the extraction of useful knowledge from Big Data in terms of a processing pipeline that transfers, stores, and analyses data for whole systems. According to Kuo et al. [12], the process of achieving full Big Data utilization involves the following five distinct configuration stages, each of which presents specific challenges:

Data aggregation: Copying/transferring data to a storage drive is the most commonly used method of aggregating and migrating large quantities of data, but its efficiency declines as volume increases. Big Data usually involves multiple organizations, geographic locations and multiple devices to aggregate over a system; typically, therefore, generating large datasets by replication from production should minimize ongoing consumption of network services and database resources, enabling the production system to render the system in operation. Further, exchange of data between groups and databases is very difficult to coordinate; hence, a secondary database, external to production systems, is usually created. Another approach to aggregation is to transfer data over a network. However, transferring, aggregating and indexing vast amounts of data require a significant bandwidth over a long duration. A third option is to replicate data from the sources and generate it iteratively across instances and multiple nodes, as Hadoop does when replicating file blocks and storing them via distributed batch processes [13-15].

Data maintenance: Since Big Data, by definition, consists of large volumes of information, it is very difficult to store and maintain for ongoing queries, especially with continuous batch processing of data. Moreover, the necessary investment of time and cost can prohibit small organizations or departments from managing large amounts of data. Another challenge in healthcare is that real patient data, metadata and data profiles need to be constantly updated; otherwise, the analytics is rendered useless. Many solutions to this problem are available, including cloud computing [16], grid computing [17], NoSQL/NewSQL and other storage systems (e.g., MongoDB, HBase, Voldemort DB, Cassandra and the Hadoop Distributed File System) and Google's BigTable [18-20]).

Legality and ethics are major issues in data maintenance. Security, confidentiality and privacy mandated by legislation and strict regulations are key components of data governance, which holds those responsible for data maintenance accountable. For example, the Health Insurance Portability and Accountability Act require the removal of 18 types of information that could identify patients. Privacy concerns can be addressed by applying appropriate software and database technologies, such as key-value storage services; however, doing so requires advanced configuration and technical knowledge. For example, Pattuk et al. [21] have proposed a framework for secure Big Data management involving an HBase database called Big Secret, which securely outsources and processes encrypted data over public key-value stores. Derbeko et al. [22] provide a comprehensive review of existing security and privacy protocols for distributed processing of large-scale data in a cloud computing environment. Most hospitals house their data in server racks in a highly secure building and vendors commonly are not allowed to use cloud services.

Data integration: Processes of data integration and interoperability involve combining (and perhaps transforming) data in an appropriate format for analysis. Since Big Data in healthcare is extremely large, distributed at different locations, unstructured and heterogeneous, data integration is very challenging [16,23]. Numerous solutions have been proposed for raw Big Data integration [24-28]. These methods are problem-oriented; in other words, they are applied to specific datasets or aggregates. Generic approaches to the integration of unstructured data are very few.

Data analysis: BDA involves the programming of analytic algorithms. However, with increasing complexity, computing time increases dramatically even with small increases in data volume. For example, in the case of Bayesian Networks, which are commonly used algorithms for modeling knowledge in computational biology and bioinformatics, the computing time required to find the best network increases exponentially as the number of records rises [29]. Several days may be required even for simple data analysis, and when databases are very large and SQL-like "joins" are executed, it may take months to obtain a result. Consequently, many studies suggest parallelizing computing models to enhance performance and reduce the computational intensity of analysis [30-37].

Pattern interpretation: Knowledge representation is an absolute necessity for BDA in healthcare. BDA is of little value if decision-makers do not understand the patterns of information that may yield a discovery. However, given the complex nature of Big Data, representations of trends and individualized results will not be immediately comprehensible to non-experts attempting to use a BDA platform. Many people intuitively believe that bigger data means better information. Agile data science can offer accurate data visualizations but cannot protect users from inaccuracies and faulty assumptions. Reporters are often fooled into thinking that correlations are significant with outbeing aware of nuances in the data, its quality, and its structure.

Big Data Technologies and Platform Services

Big Data technologies fall into four main categories: High-Performance Computing (HPC) technologies, storage technologies, resource/workflow allocators, and insertion or ingestion processes [36]. An HPC system is usually the backbone of the technology framework (e.g., IBM's Watson). HPC can consist of a distributed system, grid computing, and a Graphical Processing Unit (GPU). In a distributed system, several computers (computing nodes) can participate in processing large volumes and varieties of structured, semi-structured, and/or unstructured data. A grid computing system is a distributed system employing resources over multiple locations (e.g., CPUs, storage of computing systems across a network, etc.),

which enables processes and configurations to be applied to any task in a flexible, continuous and inexpensive manner. GPU computing is well-adapted for throughput-oriented workload problems, like batch fills at large volume. Parallel data processing can be handled by GPU clusters. However, "GPUs have difficulty communicating over a network, cannot handle virtualization of resources and using a cluster of GPUs to implement a commonly used programming model (namely, MapReduce) presents some challenges" [36].

In the quest for the most efficient platform, distributed systems appear to be the best choice of technologies for the near future. Therefore, it is important to understand the nature of a distributed system as compared to conventional grid computing, which can also be applied to supercomputing and high-performance computing, and this does not necessarily mean data mining or big data. Furthermore, a distributed computing system can manage hundreds to thousands of computers, each of which is limited in its processing resources (e.g., memory, CPU, storage, etc.). By contrast, a grid computing system makes efficient use of heterogeneous systems with optimal workload-management servers, networks, storage and so forth. Therefore, a grid computing system supports computation across a variety of administrative domains, unlike a traditional distributed system.

The most common computing devices, such as personal desktop computers, use a single processor with a main memory, a cache, and a local disk (or a computing node). In the past, applications used for parallel processing in large scientific statistical calculations employed special-purpose parallel computers with many processors and specialized hardware. However, the prevalence of large-scale web services has encouraged a turn toward distributed computing systems: that is, installations that employ thousands of computing nodes operating more or less independently with a given application (e.g.,[38,39]). Since these computing nodes are off-the-shelf hardware, the cost of distributed systems is lower than that of special-purpose parallel machines. Moreover, distributed systems allow for localized interactive access to BDA platforms. In response to the needs of distributed computing, a new generation of programming frameworks has emerged. These frameworks take advantage of the power of parallelism while avoiding pitfalls, such as the reliability problems posed by the use of hardware consisting of thousands of independent components, any of which can fail at any time. The framework used in the present study, the Hadoop cluster, with its distributed computing nodes and connecting Ethernet switch, runs jobs controlled by a master node (known as the NameNode); this master node is responsible for chunking data, cloning it, sending it to the distributed computing nodes (known as DataNodes), monitoring the cluster status and collecting or aggregating the results [38]. It thus becomes apparent that one must consider not only the architecture and computing system used in a BDA platform but also the extent to which the data processes used to produce results are inherent or customizable.

A number of high-level programming frameworks have been developed for use with HDFS. MapReduce, a programming framework for data-intensive applications introduced by Google, is the most popular component for use with Hadoop. Borrowing from functional programming, MapReduce enables programmers to define Map and reduce tasks in order to process large sets of distributed data in a specified sequence, thus allowing many of the most common calculations on large-scale data to be performed on computing clusters efficiently and in a way that is tolerant of hardware failures during compaction and computation. Therefore, distributed computing using the MapReduce-Hadoop framework represents a significant advance in the processing and utilization of Big Data, especially in the banking and transportation sectors but also in healthcare [36,40]. However, MapReduce is not suitable for online transactions or streaming; therefore, it does not lend itself to the accurate retrieval of data in real time.

The key strengths of the MapReduce programming framework are its high degree of parallelism combined with its simplicity and its applicability to a wide range of domains. The degree of parallelism depends on the size of the input data [20]. The Map function processes paired inputs (e.g.,key1 and value1) and returns intermediary pairs (e.g.,key2 and value2). Then the intermediary pairs are grouped together according to their keys. The Reduce function outputs new key-value pairs (e.g.,key3 and value3). High performance is achieved by dividing the processing into small tasks run in parallel in a cluster. Programs written in a functional style are automatically parallelized and executed by MapReduce. Employing MapReduce and Hadoop has two advantages: (1) reliable data processing with fault-tolerant storage methods that replicate computing tasks and clone data chunks on different computing nodes across the computing cluster and (2) high-throughput data processing via a batch processing framework and the Hadoop Distributed File System (HDFS) [38,40-43]. Thus, not only are Hadoop and MapReduce compatible but their inherent processes allow them to perform well in combination.

A large and growing range of technologies related to Hadoop is emerging. Open-source software for use with the HDFS is updated monthly and even weekly. Chang et al. [41] state that the HDFS is designed to build a distributed data center mostly from software rather than hardware. Software applications installed in various versions of HDFS use data duplicates to enhance reliability. Although data duplicates need extra storage space and hence require a larger investment in infrastructure, deduplication techniques can improve efficiency in the use of storage space [41]. Within the growing Hadoop ecosystem, a number of software applications and Big Data platforms have been developed for use in bioinformatics and genetics. However, very few Big Data platforms have been adopted for real-time utilization in hospitals globally. Relatively new applications like Apache Spark (2013) and Apache Drill (2015) have yet to be tested for use with clinical databases and systems. In general, very little is known about the usability of Big Data platforms or frameworks in healthcare.

Design the Analytics Framework

In considering the design of a framework for a BDA platform to use in the field of healthcare, the basic construct of processes over a platform requires diverse clinical data from multiple sources and querying large volumes. Also, the configuration must ensure patient data security, confidentiality and privacy. This section describes our approach of constructing the framework.

The Conceptual Analytics Framework

The BDA platform should harness the power of accessible front-end applications to analyze large quantities of data in an interactive manner, while enriching the user experience with data visualizations.

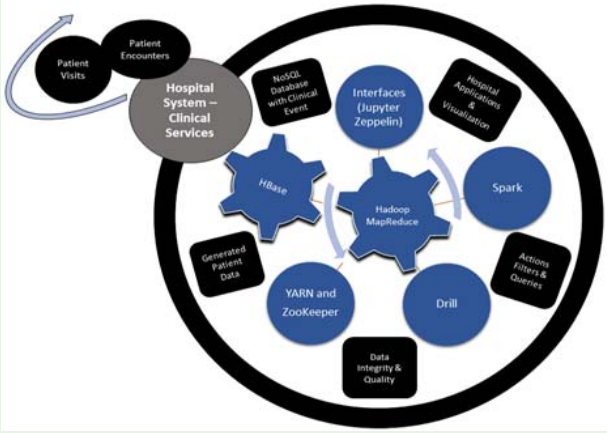

Figure 1: The proposed Big Data Analytics (BDA) of hospital system including Hadoop/MapReduce with YARN and ZooKeeper, HBase, Apache Spark, Apache Drill, and Interfaces (Jupyter and Zeppelin). The squares represent desired outputs of the platform to end users.

All this functionality must be accomplished at moderate expense. Based on these requirements, we construct an interactive dynamic framework (Figure 1). And to meet requirements of clinical users front-end and interfaced applications were tested (i.e. Apache Phoenix, Spark, and Drill) and integrated with the HDFS and back-end NoSQL database of HBase. Together, the framework and the applications allow users to query, visualize, interpret and modify outputs of the data. The overall purpose is to make Big Data capabilities accessible to stakeholders, including University of Victoria (UVic) researchers, VIHA physicians, healthcare practitioners, database administrators and web-based application developers.

The proposed analytics framework included the following nine components:

1. A high performance computer (HPC) clusters with a total of 72 cores and parallelized six nodes. Serial programs run on one CPU or core in one of the computing nodes in the cluster. Multiple copies are run over the cluster by submitting serial batch jobs by user; while parallel programs, on the other hand, had multiple processes or threads that can run at the same time. This study utilized the batch serial process to access and start jobs over the Hadoop with some parallel MapReduce processes in the top-down architecture from head node to worker or slave nodes.

2. A back-end NoSQL database of HBase (NoSQL version 0.98.11) was used. HBase is a NoSQL database composed of the main deployment master (DM) node and its fail-over master, the RegionServers holding HBase data, and ZooKeeper, which contained services to allocate data locality [44], of three nodes, that orchestrated that ensemble. The xml configuration files of HBase-site.xml and the HBase-env.sh were adjusted to improve the performance of HBase. HBase was chosen due to its NoSQL services, especially linear to modular scalability to document architecture. It uniquely indexed rows to columns and established key-stores to encrypt the data. In addition, it allowed for SQL-like layer of Apache Phoenix to be configured on top of HBase big-tables.

3. In the emulation of the database, each row represented encounter-based patient data as a Big Integer type with diagnoses, interventions and procedures specific to that patient. This patient encounter model coincided with the Health Authority's current Admission, Discharge and Transfer (ADT) system in its database schema linked to data warehouses, which also includes Discharge Abstract Database (DAD) (see Table 1). This patient-specific structure in the database allows for active updates to add to the data generation while maintaining accurate patient querying over the platform in the system. All necessary data fields were populated for 50 million records before replication to form three-nine billions of records. This data followed the workflow of VIHA staff. Rows across the columns according to the existing abstraction of patient visits to hospital. HBase established this row-column relationship(s) with a wide range of indexes for each unique row and each row contained a key value that was linked to the family of qualifiers and primary keys (columns). HBase operators were specific to family qualifiers at each iteration; therefore, the data was patient-centric combined with certain data (from different sources of metadata), such that summary of diagnosis or medical services as a whole could be queried.

4. The construction of the framework of HBase (NoSQL) across Hadoop (HDFSversion 2.6.0)\MapReduce established the BDA platform. This construct coincided with and was enforced by the existing architecture of the WestGrid clusters at UVic (secure login via lightweight directory access protocol or LDAP to deploy jobs under restricted accounts). It was initially running the architecture of the platform with five worker nodes and one master node (each with 12 cores) and planned on increasing the (dedicated) nodes to 11 and possibly to 101, as well as incorporating a non-dedicated set of virtual machines on WestGrid's openstack cloud.

5. A master Deployment Manager (DM) is the portable batch serial login that was configured as head node to worker nodes. Deployment of the Hadoop environment on the nodes carried out via a sequence of setup scripts that the user calls after loading the necessary modules and setup additional configuration to the head node with YARN and ZooKeeper as allocators of various deployments. Setup scripts created an initial configuration depending on the number of nodes chosen when launching the job. The user can adjust those configurations to match the needs of the job and its performance. The data were distributed in parallel on the nodes via a balanced allocation with every running part of the batch jobs in an initialized serial computing process.

6. Making the DBA platform InfiniBand-enabled was challenging, as most of the Hadoop environment services rely on the hostname to get the IP address of the machine. Since the hostnames on a cluster are usually assigned to their management network, the setup scripts and the configuration files required adjustment. The InfiniBand was used because it offers low latency and high bandwidth (~40Gb/s) connectivity between the nodes. YARN, Hadoop's resource job manager, unfortunately still partly uses the Gig-Ethernet interface when orchestrating processes among the nodes, but the data transfer was carried out on the InfiniBand. Yarn was the resource manager of Hadoop and services of scheduling incongruent to running the Hadoop jobs [38].

7. The queries via Apache Phoenix (version 4.3.0) resided as a thin SQL-like layer on HBase. This allowed ingested data to form structured schema-based data in the NoSQL database. Phoenix can run SQL-like queries against the HBase data. Similar to the HBase

Table 1: Use cases and patient encounter scenarios related to metadata of the patient visit and its placement in the database related to query output.

Case No.	Case*	Column (Metadata) Used for Analysis	Database Build	Query Output
1	Uncontrolled Type 2 diabetes & Complex comorbidities	ICD10-CA, MRN, PHN and LOS, Discharge	DAD with Diagnosis Codes, patient IDs and Discharge in Columns	ICD10-CA codes with counts, frequencies or max values for patient encounters
2	TB of the lung & uncontrolled DM 2	ICD10-CA, MRN, PHN, Inpatient Encounter, Location, Unit Transfer	DAD and ADT columns	ICD10-CA and encounter type codes with counts, frequencies or max values for patient encounters
3	A on C Renal Failure, Fracture, Heart Failure to CCU and stable DM 2	ICD10-CA, MRN, PHN, Intervention (CCI), Episode, Unit Transfer, Bed Location, CCU codes, Discharge	DAD and ADT columns	ICD10-CA, CCI and encounter types and unit transfer and bed location codes with counts, frequencies or max values for patient encounters
4	Multi-location Cancer patient on Palliative	ICD10-CA, MRN, PHN, Intervention (CCI), Surgery, Episode, Bed Location, Transfer to ALC Unit, Medical Services and Patient Services, Discharge	DAD and ADT columns	ICD10-CA, CCI and encounter types and unit transfer and bed location and medical service codes with counts, frequencies or max values for patient encounters
5	1 cardiac with complications	ICD10-CA, MRN, PHN, Intervention (CCI), Surgery, Episode, Bed Location, Transfer, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, CCI and encounter types and transfer codes with counts, frequencies or max values for patient encounters
6	1 ER to surgical, Fracture, re-admit category 7 days and some complication after	ICD10-CA, MRN, PHN, Intervention (CCI), Surgery, Episode, Bed Location, Medical Services, Progress Notes, Discharge, Re-Admission	DAD and ADT columns	ICD10-CA, CCI and medical services and re-admit codes with counts, frequencies or max values for patient encounters
7	1 Simple Day-Surg. with complication, so got admitted to Inpatient (Allergy to medication)	ICD10-CA, MRN, PHN, Intervention (CCI), Surgery, Bed Location, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, CCI and medical services codes with counts, frequencies or max values for patient encounters
8	1 cardiac with complications and Death	ICD10-CA, MRN, PHN, Intervention (CCI), Episode, Bed Location, Transfer, Medical Services, Discharge Disposition	DAD and ADT columns	ICD10-CA, CCI and medical services, discharge disposition and transfer codes with counts, frequencies or max values for patient encounters
9	1 Normal birth with postpartum hemorrhage complication	ICD10-CA, MRN, PHN, Intervention (CCI), Surgery, Episode, Bed Location, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, CCI and medical services and discharge codes with counts, frequencies or max values for patient encounters
10	1 HIV/AIDS patient treats for underlying factor (an infection)	ICD10-CA, MRN, PHN, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, and medical services codes with counts, frequencies or max values for patient encounters
11	Strep A infection	ICD10-CA, MRN, PHN, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, and medical services codes with counts, frequencies or max values for patient encounters
12	Cold but Negative Strep A. Child with throat culture	ICD10-CA, MRN, PHN, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, and medical services codes with counts, frequencies or max values for patient encounters
13	Adult patient with Strep A. positive and physical exam	ICD10-CA, MRN, PHN, Medical Services, Patient Services, Discharge	DAD and ADT columns	ICD10-CA, patient and medical services codes with counts, frequencies or max values for patient encounters
14	Severe Pharyngitis with physical exam	ICD10-CA, MRN, PHN, Medical Services, Patient Services, Discharge	DAD and ADT columns	ICD10-CA, patient and medical services codes with counts, frequencies or max values for patient encounters
15	Child, moderate Pharyngitis, throat culture negative, physical exam	ICD10-CA, MRN, PHN, Medical Services, Discharge	DAD and ADT columns	ICD10-CA, and medical services codes with counts, frequencies or max values for patient encounters
16	Adult, history of heart disease, Positive culture for Strep A.	ICD10-CA, MRN, PHN, Medical Services, Patient Services, Discharge	DAD and ADT columns	ICD10-CA, patient and medical services codes with counts, frequencies or max values for patient encounters
17	Adult, physical exam, moderate pharyngitis, positive for strep A. culture and positive second time, re-admit	ICD10-CA, MRN, PHN, Medical Services, Patient Services, Discharge	DAD and ADT columns	ICD10-CA, patient and medical services codes with counts, frequencies or max values for patient, readmit encounters

***Use Case Descriptions:**

Case 1: Uncontrolled Type 2 Diabetes (DM) & Complex comorbidities: A retired 69-year-old man with a 5-year history of type 2 diabetes. Although he was diagnosed 5 years ago, he had symptoms indicating hyperglycemia for 2 years before diagnosis. He had fasting blood glucose records indicating values of 118-127 mg/dl, which were described to him as indicative of "borderline diabetes." Presented with uncontrolled type 2 diabetes and a complex set of comorbidities, all of which needed treatment.

Case 2: Tuberculousus (TB) & uncontrolled Diabetes (DM) Type 2: Tuberculin skin test was reactive and her chest x-ray revealed opacity in the right lung. A smear test was positive for acid-fast bacilli and culture and sensitivity results are pending. In addition, the patient had an uncontrolled DM episode which was caught late and she ended up with acute renal failure. She spent extra days in the hospital to take care her DM type2 complications. The culture test was eventually confirmed TB.

Case 3: A on C Renal Failure, Fracture, Heart Failure to CCU and stable DM Type 2: A patient was admitted from a long term care facility. The elderly patient is 80 years old. She was admitted for renal failure. She was diagnosed with acute on chronic renal failure. She has a history of diabetes type 2. During this episode of care she developed an infection that resulted in respiratory failure and severe sepsis, Group A Streptococcus, pneumonia. her health deteriorated, her attending physician also admitted her to the CCU for heart failure.

Case 4: Multi-location Cancer patient on Palliative: A 79 year old patient visited ER for cold that had lasted a month. He was tested for bacterial infection and was negative. After further examination a skin lesion was noted, which was growing. The patient subsequently was admitted to inpatient ward and went for the biopsy of the lesion which came back positive for melanoma. After further examination of prostate, the PSA test, it was noted that patient required biopsy of the prostate. After further examination, the patient was designated ALC and was discharged to palliative care unit. He was discharged to a hospice after 12 days of ALC stay in the hospital.

Case 5:1 cardiac with complications: Patient was admitted for pain in his chest. He has had a history heart disease and high blood pressure. The ECG showed a ST elevation. Subsequently, patient underwent a PCI and the left main coronary artery dilation was performed. We used laser dilator with stent to perform dilation. Patient was stable after the intervention. He had DM 2 and needed immediate access to Reno-dialysis. With patient consent, we implemented a hemodialysis device subcutaneously for better care. Patient discharged in stable conditions.

Case 6: 1 ER to surgical, Fracture, re-admit category 7 days and some complications: Patient admitted to ER for fracture of the right arm. After an x-ray, it showed a complex fracture of right radius, and right ulna, and needed immediate intervention to fix the fracture. Subsequently, patient was admitted to surgery and underwent an open surgery for internal fixation of the fractures of both radius and ulna. The ulna required the use of staple and wire. All bleedings were cauterized. The patient was given antibiotics interaoperatively. The patient will need the removal of the implemented devices on a later date. Patient left the OR in stable conditions. Patient stayed overnight and was discharged next morning.

Case 7: 1 Simple Day-Surg. with complication, so got admitted to Inpatient (Allergy to medication): Patient was scheduled for a laparoscopic removal of gallbladder (Cholecystectomy). In addition, surgeon did exploration of the bile ducts and found several stones that were also removed. At postop, patient got a severe reaction to an antibiotic (penicillin) and she was admitted to inpatient for monitoring. During her stay, she developed fever and eventually went to anaphylactic shock and had 8 seizures.

Case 8: 1 cardiac with complications and death: 90 year old patient was admitted from a nursing home. She had cardiac and renal failure. The patient was treated with ACE inhibitors and Digoxin. She was admitted to CCU, and for her renal failure she was put a dialysis device. Unfortunately, she developed a bacterial infection and had a severe reaction to the antibiotic medication which caused her to go into anaphylactic shock. She lost consciousness at 12 am and pronounced dead at 3:30 am.

Case 9:1 Normal birth with postpartum hemorrhage complication: An Anemic mother gave birth to a healthy newborn. However, after her normal birth, she developed increased blood loss and resulted in the postpartum hemorrhage. The surgeon was called in to do a partial hysterectomy. The patient undergone the hysterectomy and after a day of monitoring, she was discharged.

Case 10:1 HIV/AIDS patient treats for underlying factor (an infection): A known HIV patient was admitted for treatment of his current infection. He was diagnosed with Pneumocystis carinii pneumonia and admitted for care. He was treated with antibiotic and he was discharged with stable conditions.

Case 11: A 22-year old male college student presented with a two-day history of runny nose and slight cough, fever to 102 degrees Fahrenheit with occasional chills and muscle aches. He had taken two aspirin prior to seeing his physician. Physical examination was unremarkable accept for temperature of 101, marked pharyngitis and clear rhinitis. RADT was positive for group A streptococci. The patient was given a prescription for Penicillin V 500 mg and told to take one pill twice a day for 10 days. Patient was also told to call the office if his symptoms did not improve in 3 days. He did not call to report any continued problems.

Case 12: A 3-year old girl was seen in the hospital with a 24-hour history of acute throat and fever. On physical examination, her temperature was 103 Fahrenheit and her left tonsil acutely inflamed. A sample was taken for a throat culture and the patient's mother was given a prescription for Amoxicillin 250mg to give to the child three times per day for 10 days. The mother was told to call the office if the child's symptoms persisted. The throat culture report, filed in the patient's chart four days after the initial visit, was negative.

Case 13: A 42-year old man who works as a long-haul truck driver presented to the hospital with a stuffed nose, severe sore throat, and a cough. On physical examination, his temperature was 100 degrees Fahrenheit, his respiration rate was 18, and his lung had fine rales of the left side. RADT was positive for group A Streptococci. Patient was given a prescription for Penicillin V 250 mg to take three times a day for 10 days.

Case 14: An 18-year old moderately retarded female who lives in a foster care home was brought to the hospital because she was complaining of feeling sick and having a mild sore throat. On physical examination, her temperature was 100 Fahrenheit, she had severe pharyngitis with a few petechiae on her uvula, and erythematous tonsils, and cervical adenopathy. RADT was positive for group a streptococci. The patient's caregiver was given a prescription for 500 mg Amoxicillin (oral suspension) to give the girl twice a day for 10 days.

Case 15: A 5-year old boy was brought to the physician. His mother said he had been complaining of a sore throat and had exhibited a decreased appetite over the past 3-4 days. On physical examination, there was moderate pharyngitis with some exudates and marked bilateral swelling of the anterior cervical nodes. The patient's temperature was 99.8 degrees Fahrenheit. A throat culture was attempted but the patient resisted attempts to obtain an adequate specimen. The mother was given a prescription for Amoxicillin (oral suspension) 250mg to give to the child twice a day for 10 days.

Case 16: A 52-year old female with a history of heart disease secondary to childhood rheumatic fever was seen by her physician following a 2-day history of several sore throat, fever, and myalgia. On physical examination, the woman's throat was found to be inflamed and anterior cervical adenopathy was present. Temperature was 101.2 degrees Fahrenheit. RADT was positive for group a streptococci. A throat culture was done. Patient was given a prescription for Erythromycin 500 mg to take three times a day for 10 days (she is allergic to penicillin). The patient was scheduled to return in 10days for a repeat throat culture. Upon return to the hospital, the patient's symptoms had improved. The repeat throat culture was negative.

Case 17: A 42-year old female who works in a day care center was seen by her physician for a 2-day history of fever, malaise, and sore throat. On physical examination, she was found to have anterior cervical adenopathy and moderate pharyngitis. Her temperature was 100.2 degrees Fahrenheit. RADT was negative. A throat culture was done and patient was given a prescription for Penicillin V 250 mg to take three times a day. The culture report, which was filed in the patient's records 2 days after her hospital visit, was positive for group a streptococci. The patient was notified of the culture results. She indicated to the physician's nurse, who phoned with the results, that her symptoms were starting to subside and her temperature had returned to normal. The muse told the patient to continue taking the antibiotics for the full 10 days and to call the hospital if she did not improve. The patient returned to the hospital in six weeks with symptoms similar to her previous visit. The RADT was positive at that time. The patient indicated that several of the children in the day care center were taking Amoxicillin for strep throat infections. The physician discussed the need for good hand-washing techniques for the patient and for the children in the day care center. The patient was given a prescription for Penicillin V 250 mg to take three times a day for 10 days and told to contact the hospital if her symptoms did not improve or if she had a recurrent infection. The patient did not contact the hospital again for this problem.

shell, Phoenix is equipped with a python interface to run SQL statements and it utilizes a .csv file bulkloader tool to ingest a large flat file using MapReduce. The load balancing between the RegionServers (e.g., "salt bucket" code) was set to the number of worker nodes that allowed ingested data to be distributed evenly.

8. Apache Spark (version 1.3.0) was also built from source and installed to use on HBase and the Hadoop cluster. Spark utilizes Yarn and HDFS architecture and is known to scale and perform well in the data space (over distributed files).

9. Inspired by Google's big query engine Dremel, Drill offers a distributed execution environment for large-scale ANSI-SQL: 2003 queries. It supports a wide range of data sources, including .csv, JSON, HBase, etc... [45]. By (re)compiling and optimizing each of the queries while it interacted with the distributed data sets via the so-called Drillbit service, Drill showed capacity of the query with performance at a low latency SQL query. Unlike the master/slave architecture of Spark, in which a driver handles the execution of the DAG on a given set of executors, the Drillbits were loosely coupled and each could accept a query from the client [46]. The receiving Drillbit becomes the driver for the query, parsing, and optimization over an efficient, distributed, and multiphase execution plan; it also gathers the results back when the scheduled execution is done [46,47]. To run Drill over a distributed mode, the user will need a ZooKeeper cluster continuously running. Drill 1.3.0 and ZooKeeper 3.4.6 were installed and configured on the framework of the platform over a port with a local host.

Computational Platform

In this study, we used high performance clusters of WestGrid. WestGrid is a Canada government-funded infrastructure program started July 2010 at UVic. In Western Canada, it provides access to high performance computing and distributed data storage, using a combination of grid, networking and collaboration tools [48].

The WestGrid computing facilities at UVic house the Hermes and Nestor system, which are the two super clusters that share infrastructure, such as resource management, job scheduling, networked storage, and service and interactive nodes. Also, Hermes and Nestor share login nodes: hermes.westgrid.ca and nestor.westgrid.ca (hermes.westgrid.ca and nestor.westgrid.ca are aliases for a common pool of head nodes named litaiNN.westgrid.ca). Nestor is a 2304-core capability cluster geared towards parallel jobs. It consists of 288 IBM iDataplex servers with eight 2.67 GHz Xeon x5550 cores and 24 GB of RAM. Data is shared between nodes and the GPFS file system using a high-speed InfiniBand interconnect. Hermes is a 2112-core capacity cluster geared towards serial jobs. It consists of 84 nodes having eight cores each and 120 nodes with 12 cores each.

The Data Process Pipeline

The proposed framework with the techniques and tools used in a data process pipeline were carried in six steps.

Step 1. Data preparation and privacy protection

Privacy protection is an important requirement in healthcare data. All patient data must be identifiable and cataloged across the hospital system. Typically this is carried out by business/security analysts based on regulations and data analyst for maintenance in data warehouse. The goal of this step (in our study) is to generate a comprehensive list of sensitive elements specific to the organization and discover the associated tables, columns and relationships across the data warehouse. After that is established, data masking or generating an encrypted data replication over its distribution will be used to conceal the patient's identity while preserving the information, relationships and context [49].

Step 2. Data Acquisition

There is a wide range of tools and technologies for bulk loading and accessing large datasets from different sources. In this study, we proposed to use Apache Sqoop to ease the transfer between VIHA data warehouse and the WestGrid. Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data repositories. For collecting and gathering unstructured data, such as logs, we used Apache Flume. Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. High-speed file transfer technologies, such as SCP and GridFTP, were used to transfer huge amounts of data into the parallel file system (GPFS).

Step 3. Data maintenance

The aggregated data was stored and maintained over HDFS in NoSQL HBase over Hermes nodes in WestGrid. To improve the ingestion of the 90 columns over generated three-nine billion rows, local disks of 40TB in total were physically installed on the worker nodes. After local disks were installed, a set of shell scripts was used to automate the generation and ingestion process of 50 million records as a batch at each of the iterations (via MapReduce). The goal of study was to run towards ten billion rows but due to operational barriers, workflow limitations and table space of key stores, which almost tripled during the iterations, nine billion achieved but only three billion could be queried. Some limitations of the Hadoop/Map Reduce framework occurred because local disks had to be reinstalled after the failover from WestGrid's 500GB disks to 2TB slots did not work.

Step 4. Data integration

The data were modelled according to the critical data profiles from the clinical event tables. Each of the data elements were derived from naming conventions of the patient record tables of Admission, Discharge and Transfer (ADT), i.e., Figure 2, as well as data standard of Discharge Abstract Database (DAD), i.e., Figure 3, for clinical reporting. This was technically established by SQL-like code run over HBase via Apache Phoenix [50]. Additionally, the most important step in the data integration was that the data was transformed into appropriate formats via task scheduler in Map Reduce over

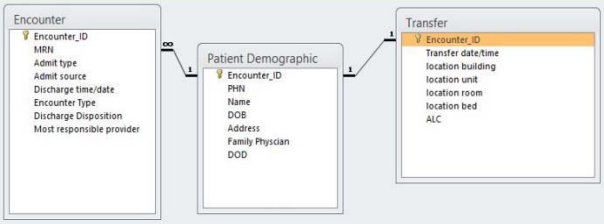

Figure 2: Conceptualized database of Admission, Discharge, and Transfer (ADT) from Vancouver Island Health Authority (VIHA) for simulation.

HDFS. The two components Map and Reduce placed files over the distributing file process but there were different performances. Map portion took only three minutes, but the Reduce part of bulkload and indexing took three to twelve hours. Also, Reduce did not run at a constant percentage of Map and ranged from 20-60%. To improve the Reducer, several manual steps were taken including compression of the H Files in HBase. However, there was still on average 400-500 minutes to complete the upload and integration, and the compression failed periodically, which resulted in a delay to clean out failed files and re-process the data from Map Reduce. This issue was investigated and deemed caused in large part by HBase's Region Server hot-spotting or over-utilization that stalled the releasing process. Thus, the data process of the platform had a major limitation with the Map Reduce that persisted. Additionally, as part of the process, ZooKeeper and Yarn connectivity and Region Servers connecting to network only persistently connected to UVic's network instead of WestGrid's InfiniBand, which drastically slowed performance. Some configuration to ZooKeeper and Yarn minimized this occurrence but no concrete resolution was found. Thus, this step to integrate the data over the HDFS to HBase required manually configuration and monitoring at each and every batch submitted.

Step 5. Data analysis

To handle intensive computation, we used the MapReduce programming model that distributed analytic/data mining tasks. The MapReduce templates were tailored to use with applications of Apache Phoenix, Spark and Drill on the HBase database. To give the structured data the same semantics, as its relational database counterpart, we opted to use Phoenix on HBase because it is still a relational database layer that runs on top of HBase and offers a low-latency SQL-like access to HBase by running multiple parallel HBase scans on different region servers [14]. For developing some of the applications for end users, we also engineered notebooks on Jupyter and Zeppelin interfaces over Web browsers to connect to the database and interactively apply queries simultaneously to generate results.

Step 6.Pattern representation and visualization

Pattern presentation was found thru interface tools of Apache Spark (version 1.3.0) and Drill over the platform. Apache Spark was also built from source and installed to use on HBase and the Hadoop cluster. The intent was to compare different query tools like Apache Spark and Drill against Apache Phoenix using similar SQL-like queries. The results showed that the ingestion time of one billion records took circa two hours via Apache Spark [51]. Apache Drill outperformed Spark/Zeppelin and Spark/Jupyter. However, Drill was restricted to running more simplified queries, and was very limited in its visualizations. Therefore, it exhibited poor usability for healthcare. Zeppelin, running on Spark, showed ease-of-use interactions for health applications, but it lacked the flexibility of its interface tools and required extra setup time with a 30-minute delay before running queries. Jupyter on Spark offered high performance stacks not only over our platform but also in unison to run all queries simultaneously for a variety of reporting for providers and health professionals.

Simulation

Computing Platform and Configurations

In this section, we describe our steps and experiences to test the technical framework of the BDA platform. To accomplish this, we installed and configured a Hadoop environment from source on the WestGrid cluster, and a dynamic Hadoop job was launched. It was configured by hdfs-site.xml and a MapReduce frame, configured via MapRed-site.xml that ran under the Hadoop resource manager Yarn (with configuration file yarn-site.xml). The number of replicas was set to three. To interact with HDFS, command scripts were run to automate the ingestion step (generate data replication in the exact format specified by SQL script to the nodes).

The BDA platform was built on top of the available open source software called HBase. HBase comprised a main Deployment Master (DM) and fail-over master. Its region servers held HBase data and a ZooKeeper of three nodes orchestrated the ensemble. The xml configuration file HBase-site.xml and the HBase-env.sh were adjusted to configure and fine tune HBase. HBase was chosen due to its real-time read/write access, and linear and modular scalability. HBase consisted of unique rows and each row contains a key value. A key value entry has five parts: row-key (row), family (fam), qualifier (qua), timestamp (ts) and value (val) denoted as:

KEY:=row||fam||qua||ts.

Additionally, to establish the HBase key value entries there are four operations:

1. Put - inserts data;
2. Get - retrieves data of a specific row;
3. Delete - removes a data row;
4. Scan - retrieves a range of indexed rows.

The last three operations can be limited to specific family, qualifiers, or over a certain time range. In addition, it allows for SQL-like layers via Apache Phoenix to be configured on top to bulkload to HBase [50]:

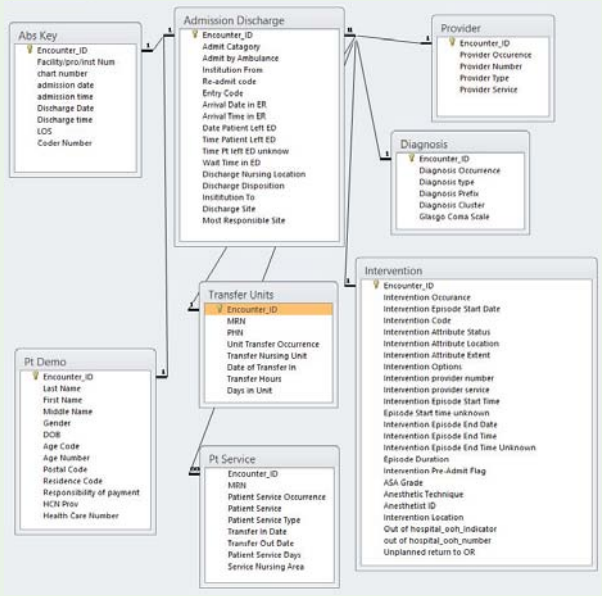

Figure 3: Conceptualized database of Discharge Abstract Database (DAD) from VIHA for simulation.

```
CREATE TABLE IF NOT EXISTS DADS1
    EncounterID BIGINT NOT NULL,
    Admit_by_Ambulance VARCHAR,
    Admit_Category VARCHAR,
    Admission_Date VARCHAR,
    Admission_Time VARCHAR,
    age INTEGER,
    Anesthestic_Grade VARCHAR ,
    Anesthetist_ID VARCHAR,
    Anesthetistic_Technique INTEGER,
    Arrival_Date_in_ER VARCHAR NOT NULL,
    Arrival_Time_in_ER VARCHAR NOT NULL,
    Date_Patient_Left_ED VARCHAR ,
    Date_of_Transfer_In VARCHAR,
    Days_in_Unit VARCHAR,
    Discharge_Date VARCHAR NOT NULL,
    Discharge_Disposition VARCHAR NOT NULL,
    Discharge_Site VARCHAR NOT NULL,
    Discharge_Time VARCHAR NOT NULL,
    Birth_Date VARCHAR,
    Diagnosis_Cluster VARCHAR,
    Diagnosis_Code VARCHAR NOT NULL,
    Diagnosis_Occurrence INTEGER,
    Diagnosis_Prefix VARCHAR,
    Diagnosis_Type VARCHAR,
    Entry_Code VARCHAR,
    Episode_Duration INTEGER,
    First_Name VARCHAR,
    Glasgo_Coma_Scale VARCHAR,
    Gender VARCHAR,
    Health_Care_Number VARCHAR NOT NULL,
    HCN_Prov VARCHAR,
    Institute_From INTEGER,
    Institution_To INTEGER,
    Interven_Attribute_Extent VARCHAR,
    Interven_Attribute_Location VARCHAR,
    Interven_Attribute_Status VARCHAR,
    Interven_Code VARCHAR,
    Interven_Episode_Start_Date VARCHAR,
    Interven_Episode_St_Date VARCHAR,
    Interven_Location VARCHAR,
    Interven_Occurrence INTEGER,
    Interven_Options VARCHAR,
    Interven_Pr_Number INTEGER,
    Interven_Preadmit_Flag VARCHAR,
    Interven_provider_service INTEGER,
    Interven_Start_Time_unknown VARCHAR,
    Interven_St_T_unknown VARCHAR,
    Last_Name VARCHAR,
    LOS INTEGER NOT NULL,
    Middle_Name VARCHAR,
    Most_Responsible_Site VARCHAR,
    MRN VARCHAR,
    Out_of_Hospital_ooh_indicator VARCHAR,
    Out_of_hospital_ooh_number INTEGER,
    PHN INTEGER,
    Postal_Code VARCHAR,
    Pdr_Number INTEGER,
    Provider_Occurrence INTEGER,
    Provider_Service VARCHAR,
    Provider_Type VARCHAR,
    Patient_Service INTEGER,
    Patient_Service_Days INTEGER,
    Patient_Service_Occurrence INTEGER,
    Patient_Service_Type VARCHAR,
    Readmit_Code INTEGER,
    Reporting_Prov INTEGER,
    Residence_Code INTEGER,
    Responsibility_for_payment INTEGER,
    Service_Nursing_Area VARCHAR,
    Time_Patient_Left_ED VARCHAR,
    Time_Pt_left_ED_unknown VARCHAR,
    Transfer_Hours VARCHAR,
    Transfer_Nursing_Unit VARCHAR,
    Transfer_In_Date VARCHAR,
    Transfer_Out_Date VARCHAR,
```

Unit_Transfer_Occurrence INTEGER,

Unplanned_return_to_OR VARCHAR,

Wait_Time_in_ED VARCHAR,

FIN INTEGER NOT NULL,

Encounter_Number INTEGER NOT NULL,

Admit_Source VARCHAR,

Encounter_Type VARCHAR,

Medical_Services VARCHAR,

MostResponProvider VARCHAR,

Address VARCHAR,

Family_Physician VARCHAR,

Location_Building VARCHAR NOT NULL,

Location_unit VARCHAR NOT NULL,

Location_Room VARCHAR NOT NULL,

Location_Bed VARCHAR NOT NULL,

CONSTRAINT PK PRIMARY KEY (EncounterID, Arrival_Date_in_ER, Arrival_Time_in_ER, Discharge_Date, Discharge_Disposition, Discharge_Site, Discharge_Time, Diagnosis_Code, Health_Care_Number, OS, FIN, Encounter_Number,Location_unit), Salt Buckets =5.

The deployment of the Hadoop environment was carried out via a sequence of setup scripts that the user runs after loading modules. These setup scripts created an initial configuration over the number of nodes requested to the batch system. The user could adjust the configurations to match the needs of the job. The services that the master node and slave nodes run are shown in Figure 4.

Making the platform InfiniBand-enabled was challenging. The InfiniBand was desirable as it offered low-latency and high-bandwidth (~40Gbit/s) connectivity between the nodes. However, most of the Hadoop environment services rely on the hostname to get the IP address of the machine. Since the hostnames on a cluster are usually assigned to their management network, the setup scripts and the configuration files needed to be adjusted. Yarn, Hadoop's resource and job manager, unfortunately, still uses the Gig-Ethernet interface when orchestrating between the nodes but the data transfer is still carried out over the InfiniBand. Figure 5 shows peak values in disk space on a Hermes node utilized that is indicative of the maximum reached at the start of file ingestion via InfiniBand. The lows are due to compaction running on the disks after disk space is maximized and this was necessary to maintain a consistent range of data distributed. This was a desired result and showed high performance.

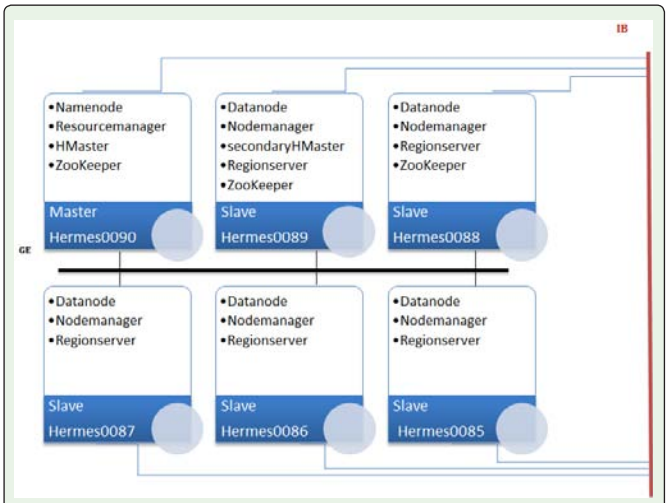

Figure 4: Construction of HBase NoSQL database with dynamic Hadoop cluster, and master and slave/worker services at WestGrid.

The Apache Phoenix allowed the ingestion to set a structured schema into the NoSQL database. The usual SQL queries can run in Phoenix against the HBase data. Similar to HBase shell, Phoenix is equipped with a python interface to run SQL statements. Moreover, Phoenix comes with a Comma-Separated Value (CSV) file bulk loader tool to ingest the large flat CSV files using MapReduce into a big table. Due to throughput and limited wall time, we could not ingest the full 1 billion records using HBase in one run (using the bulk loader) as its MapReduce required more than the available local disk spaces on the nodes to store the intermediate temporary files and timed out. Batch jobs (twenty-eighty) of 50-million records were then ingested one at a time. As explained further in the results, even with that, many ingestion attempts took a longer time than expected or failed. The load balancing between the RegionServers was also a major issue. Additionally, the performance and partitioning of data by MapReduce (its mapping and sorting) is highly dependent on how evenly it can distribute the workload [19,50]. To address it, we had to pre-split the table using the Phoenix "salt bucket" mechanism to evenly balance the distributed data to the nodes.

Data Emulation and Modeling

The established BDA platform was used to benchmark the performance of mining current and future reporting of VIHA's clinical data warehouse, which in archive spans more than 50 years. Currently, the data warehouse has 100 fact and 581 dimension tables that all encompass a total of10 billion records. Huge volumes of health

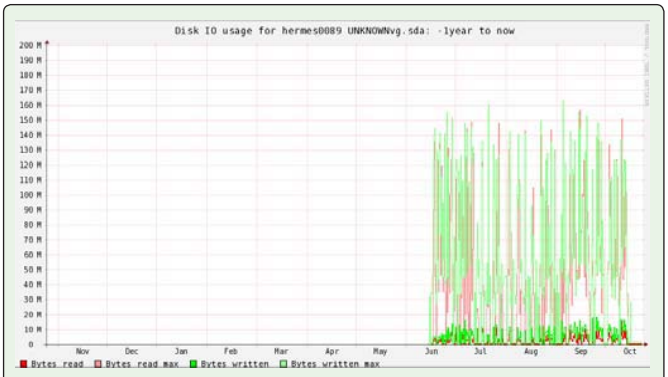

Figure 5: A year of iterations and IO Disk utilization (max 200MB/s) on Hemes89 node showing each duration of ingestion towards three billion during mid-May to October, 2016. The graph shows: bytes read (in red), bytes read max (in light read), bytes written (in green), and bytes written max (in light green). The bytes written and its max represent performance of the InfiniBand (160MB/s was achieved).

data are continuously generated and added to this collection. Within the data warehouse, two of the largest datasets are the ADT and DAD. The former contains individual patient bed-tracking information, while the latter contains Canadian Institute for Health Information (CIHI) diagnostic codes and discharge abstract metadata fields.

Over the span of twelve months in 2014-15, several interviews were conducted with business intelligence data warehouse, clinical reporting, database administrators, application platform, and health informatics architecture teams employed at VIHA. All interviews were recorded and documented. During these interviews, a patient data system generated from hospital admissions (based on encounter types) and a discharge system (based on diagnoses and procedures) were established. Furthermore, data profiles, including dependencies and the importance of the metadata for the reporting performance of the hospital, were confirmed and verified for use over the Hadoop/MapReduce to HBase framework.

In the emulated datasets, each row represented encounter-based patient data, with diagnoses, interventions, and procedures specific to that patient, that the current ADT system has in its database schema linked to a bigger data warehouse, which includes DAD (refer to Table 1). This patient-specific structure in the database allowed for active updates for accurate querying. The data emulated for both ADT and DAD were greatly simplified from the real patient records of hospitals (refer to Figures 2 and 3). In fact, the hospitals ADT alone has over 100 tables that could have patient information to query. However, we cross-referenced the most important information from ADT and DAD for clinical reporting with data wanting to be queried as a combined large data.

Unlike relational database systems, HBase itself does not support SQL; in fact, HBase is not a relational data store at all. HBase applications like Apache Phoenix can provide an SQL-like layer over HBase and are written in Java, much like typical MapReduce applications. HBase settings had to be purged or cleaned after each of the ingestions due to unknown tracers or remnants of transactions that then later caused query inaccuracies. Every HBase system components comprises a set of tables. Each table must have an element defined as a Primary Key, and all access attempts to HBase tables need to use this Primary Key; even Apache Spark SQL can be linked to database conversions with HBase [52] and other transformation methods can be applied to healthcare data towards an intact NoSQL database [53]. Additionally, in our study, HBase column represented an attribute of an object; for example, if the table had logs from servers, where each row might be a log record, a typical column would show the timestamp of the time when the log record was written, or perhaps name of the server on which the record was originated. Thus, HBase allowed for many attributes to be grouped into so-called column families and stores the elements of the column family together.

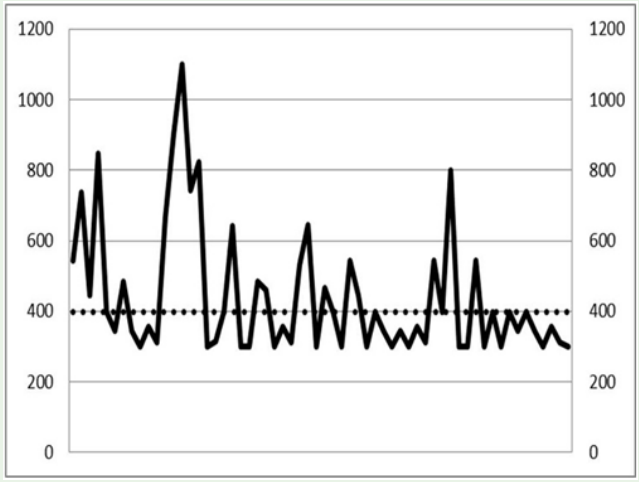

Figure 6: Sixty iterations of HFile to bulkloading via MapReduce to HBase distributed database. Solid line is actual minutes to complete all job tasks and the dotted line is average duration.

Performance Evaluation

The pathway to running ingestions and queries from our build of the BDA platform was as follows: CSV flat files generated → HDFS ingestion(s) → Phoenix bulk loads into HBase → Apache Phoenix Queries. Under this sequence and after loading the necessary module environments for Hadoop, HBase and Phoenix and testing initial results linked to the family qualifiers and HBase key value entries. The SQL code (shown in Table 1) was then iteratively run to ingest 50 million rows to the existing NoSQL HBase database (i.e., names of the columns were shortened but metadata kept exactly the same as ADT and DAD tables shown in Figures 2 and 3).

For performance benchmarking, three metric measures were used: HDFS ingestion(s), bulk loads to HBase and query times via Phoenix. Three flat files in CSV format were ingested to HDFS. One with ten records and was used for quick testing, and two others were used for the actual benchmarking of 50 million records and one billion records. The measurements were automated as much as possible to make easy for the eventual benchmarking of 10 billion records. Figure 6 shows the fluxes of the IE for all 60 iterations to three billion. We also computed the ingestion efficiency (IE) of one billion compared to 50 million records using the formula in equation (1):

$$IE = \frac{1B \times T_i(50\,M)}{50\,M \times T_i\,(1B)} \quad (1)$$

Where $T_i\ (N)$ is the time it takes to ingest N records to either HDFS or HBase.

A SQL script containing all the queries was written and ran using Phoenix sqlline.py. The output of sqlline.py was redirected to a file for further validation. In addition to the query results, sqlline.py reported the time it took for each query (similar to the usual SQL query clients). The total number of queries that were used was 22: two simple queries with wildcard column selection; ten simple queries that did not involve more than three columns in the primary keys (family qualifiers); and, ten complex queries that had >3columns selected. These queries were chosen based on surveys and interviews with VIHA experts on current clinical reporting. Furthermore, the separation between simple and complex queries was also derived to compare to performance of Apache Spark and Apache Drill on the exact same data (Figure 7).

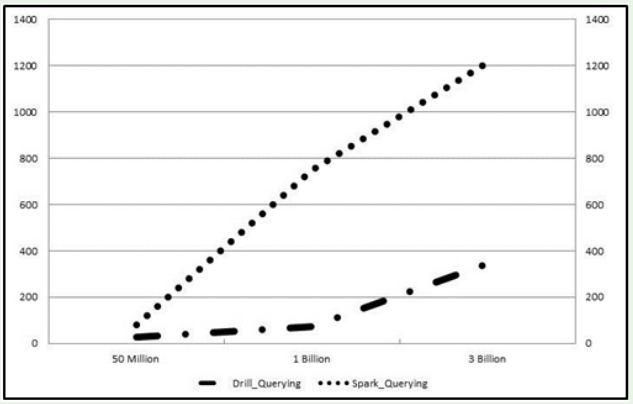

Figure 7: Query durations to running all 22 queries in total in seconds via Apache Spark (dot) and Apache Drill (line-dot).

Discussion

Essentially, this study proposes a row-column, key-value (KV) model to adopt the Hadoop/MapReduce framework to data distributed over a customized BDA platform for application in healthcare systems. Wang, Goh, Wong, and Montana [54] support this study's claim in their statement that non-relational data models, such as the KV model implemented in NoSQL databases, exhibit accurate multivariate analysis of Big Data in a Hadoop forest using bioinformatics data. Wang et al. [55] further state that NoSQL provides high performance solutions for healthcare, being well suited for high-dimensional data storage and querying, as well as optimized for database scalability and performance. A KV-pair data model can support faster queries of large-scale microarray data and can be implemented using HBase, an implementation of Google's BigTable storage system. This new KV data model implemented on HBase exhibited an average 5.24-fold increase in high-dimensional biological data query performance compared to the relational model implemented on MySQL Cluster and an average 6.47-fold increase in query performance compared to MongoDB [40]. The performance evaluation found that the new KV data model, in particular its implementation in HBase, outperformed the relational model currently implemented, thereby supporting the use of NoSQL technology in BDA platforms for hospital systems.

With HBase, the table schema is dynamic because it can be modified at any time and incorporates different forms of data for different schemas [56]. However, in order for the platform to function, HBase's schema must be predefined (as was done isingSQL-like Phoenix in this study) and the column families specified. However, the schema is very flexible, in that new columns can be added to families or groups at any time. It is therefore able to adapt to changing application requirements, as has been noted by many researchers [57,58]. As Sun [59] states, "Just as HDFS has a NameNode and slave nodes, and MapReduce has JobTracker and TaskTracker slaves, in HBase a master node (HMaster) manages the cluster and RegionServers store portions of the tables and perform the work on the data." HMaster is the implementation of the Master Server, which is responsible for monitoring all RegionServer instances in the cluster. In this study's distributed cluster, the Master started on the NameNode, while HRegionServer started the RegionServer(s). In a distributed cluster, a RegionServer runs on a DataNode [59], as was the case in our implementation. In HBase, through the ZooKeeper, various machines can be selected within the cluster as HMaster (unlike the HDFS architecture, in which NameNode has a single-point-of-availability problem) [59]. HBase clusters can also be expanded by adding RegionServers hosted on commodity class servers. For example, when a cluster expands from 10 to 20 RegionServers, it doubles both in terms of storage and processing capacity. Therefore, more storage space is required as the number of nodes increases. The present study did confirm that HBase and HDFS worked efficiently: HBase supported parallelized processing via MapReduce, and the platform provided a user-friendly Java API for programmatic end-user access.

It is possible to mimic a relational database with an SQL-like structure using NoSQL [52]. The primary key strength of the patient encounter was created using "family" or groups of columns combined. And this combination proved necessary to obtain accurate query results on real patient data. Without the groupings to form the SQL-like primary key, the results from the query did not show the correct number of fields found. In addition, the key values needed to be unique; therefore, when replicating the ingestion, queries had to be modified in order to obtain a result. Chawla and Davis [60] emphasize the importance of the patient-centered framework for a Big Data platform. Since many Big Data platforms are linked to HDFS with non-relational databases, assuring the accuracy of patient data is of the utmost importance. This was achieved in simulating queries overthe NoSQL database, but the accuracy could not be fully validated over non-replicated data of more detailed patient-centric encounters.

Even though it is very difficult to move patient data from a hospital system that updates its clinical model storage in real-time, analyzing patient-level databases can yield population-level inferences or "results" (such as the strength of association between medical treatments and outcomes), often with thousands of outcomes. Other studies indicate that although data from such sources as hospital EHR systems is generally of much lower quality than data carefully collected by researchers investigating specific questions, the sheer volume of data can compensate for its qualitative deficiencies, provided that a significant pattern can be found amid the noise [3,61]. In addition, there is a trend toward higher quality Big Data collections (such as the data produced in genomic analysis and structured data generated from standard-compliant EHR systems) [62,63]. On the other hand, Mayer-Schönberger and Cukie [64] showed that as the population sample approaches 100%, messy data can have greater predictive power than highly cleaned and carefully collected data that represents a smaller sample of the target population. The present study did not analyze the structure of the metadata; ultimately it was designed to replicate massive volumes of patient data.

It was more challenging to use the simulated data in Spark and Drill to validate a proof-of-concept use of these tools with real data. Scott [55] suggests that the competition for the best Big Data software solutions is between Spark and Drill and that Drill can emulate complex data much more efficiently than Spark because Spark requires elaborate Java, Python and Scala coding. Nonetheless, in our study both Spark and Drill were significantly faster than HBase in ingesting files directly into Hadoop. In fact, the ingestion and queries for both Spark and Drill could be run in sequence without

running compaction as well. However, it is difficult to compare the data emulation of HBase to that of Spark and Drill. Neither Spark nor Drill indexed the files, nor did they require the Reducer from MapReduce to complete its task before queries could be performed. Lack of indexing increases the risk of inaccuracies (even though the framework was more fault-tolerant when running Spark and Drill). Therefore, the Big Data tools and their inherent technologies therein strongly affect the health informatics of the data established and resulting queries.

The most impactful technology in this study was MapReduce (and its Java code). MapReduce methodology is inherently complex, as this study discovered, since its default programming framework employs separate Map and Reduce task. Our platform was highly dependent on the efficiency of MapReduce in ingesting files over the six nodes, using this workflow: Input → Map → Copy/Sort → Reduce → Output. A study by Chen, Alspaugh, and Katz [65] reported a similar result. Once the Reducer optimized configurations in Yarn and ZooKeeper with iterations of 50 million rows with data, integrity of the desired clinical event model was established via SQL in Apache Phoenix. According to blogs with technical resolutions, enabling or disabling services or xml settings over the platform are expected to be carried out, because the system relies heavily on InfiniBand (IB) bandwidth and not all default settings can be applied automatically during initialization. Furthermore, slow performance is a known issue when using MapReduce over HBase after additional indexing of data and its store [38,40,66-68].

The data used in this study included diagnosis codes, personal health numbers, medical record numbers, dates of birth, and location mnemonics (to mention only a few of the 90 columns), as these codes are standardized for hospital systems and, compared to genetic data, more easily replicate metadata in large volumes. The use of groups of events allowed the definition of a phenotype to go beyond diagnosis as coded using the International Classification of Disease, version 9 (ICD-9), potentially allowing assessment of the accuracy of assigned codes [62, 69]. In healthcare, the complexity of Big Data storage and querying increases with unstructured sets of data and/or images. The growth in the volume of medical images in modern hospitals has forced a move away from traditional image analysis and indexing approaches towards scalable solutions [70]. In fact, MapReduce has been used to speed up and make possible three large-scale medical image processing use-cases: (1) parameter optimization for lung texture classification using support vector machines (SVM), (2) content-based medical image indexing/retrieval, and (3) dimensional directional wavelet analysis for solid texture classification [71]. In one study, a default cluster of heterogeneous computing nodes was set up using the Hadoop platform, allowing for a maximum of 42 concurrent Map tasks [INSERT CITATION]. However, this study did not test the quantity and efficiency of concurrent Map tasks of MapReduce to process the data to HBase ingestions.

Greeshma and Pradeepini [66] implemented a distributed file system that stored Big Data files and a MapReduce algorithm supported the performance of the analytics on a set of clusters. Performance of the platform queries did not rely on MapReduce, only controlled the actual ingestion of files to form the database. Spark and Drill used some components of MapReduce during the initialization of the queries, but this was minimal and non-reliable. The Hadoop approach used in this study's platform did not consume large-scale data as a whole but broke it into smaller pieces distributed over the collection of servers and iterations. This placed the bulkloading on the AdminNode (deployment) that was deployed to NameNode and then distributed to DataNode. Our findings support Greeshma and Pradeepini's [66] findings that NameNode and AdminNode usally suffice to store, manage, and access Big Data files.

This study did not benchmark the actual performance of the components of the platform to the exiting system that uses analytical tools across EHR. However, to replicate/transfer data from production to domains in the hospital system via secure FTP takes six or more months at 10-15 TB. Thus, our study's demonstration that dataset volumes of 30-40 TB can be processed in three months represents an advance of roughly an order of magnitude over existing technical platforms. Moreover, Yu et al. [68] showed that Mahout's MapReduce architecture was eight to five times slower than other modified MapReduce configurations. More specific to our proposed framework, Greeshma and Pradeepini [66] showed that adding Job Tracker to the MapReduce framework is useful, since Hadoop needs to be running during the queries; this solves the limitations of the relational data warehouse by continuously running with low resources over multiple nodes. Hence, Hadoop is a single point of failure, since it cannot stop running to perform any tasks or functionalities, a weakness demonstrated in the appearance of broken Hadoop clusters in a study by Rabkin and Katz [72].

A large body of research has studied ways of improving storage performance [73] or using compression techniques to reduce the volume of intermediate data [74,75]. The intermediate data movement time during the shuffle phase has been addressed [76,77]. However, none of these studies addressed the issue of memory limitation in the Reduce phase, which can lead to abortion of Reduce tasks, as reported by Yan et al. [78], who stress the importance of memory resources (abortion of Reduce tasks during ingestions occurred in our study, although the platform did not fail when queries were run). Greeshma and Pradeepini [66] propose a memory cached mechanism to be used with MapReduce to improve the Reducer, but this was not tested on other software technologies such as HBase. Additionally, while saturation of storage IO operations or network bandwidth can lead to performance degradation in the Reduce phase (a problem encountered on Regional Servers), the Hadoop framework does not kill the MapReduce application in such cases [79]. However, in the case of an out-of-memory error, the job is usually killed, since the Reduce phase needs to bring large quantities of intermediate data into memory for processing [80].

The present study showed that performing maintenance activities over the platform was essential for ensuring reliability. Some studies have shown that Hadoop can detect task failure and restart programs on healthy nodes, but, if the Region Servers for HBase fail, this process has to be started manually [67,81,82]. Our study showed that compaction improved the number of successful runs of ingestion; however, it did not prevent failure in ingesting the files, a finding that is also supported by other studies [36,38,80,83].

Conclusion

Few previous studies have designed and tested a BDA platform for use in healthcare as this study has done, applying Big Data

technologies to large volumes of simulated data in a clinical healthcare context. This study proposes a framework that not only replicates large amounts of data over the platform but also allows for combining data from different databases during each generation of the replication. The results show that replicating data from source to HBase to form three billion patient encounters required a month of computation time; therefore, the data needs to be stored before running the queries. Moselle [84] states that if data is stored with some permanency for more than a few hours without removal, public disclosure is required (in accordance with privacy laws regarding sensitive personal data). Furthermore, Dufresne, Jeram and Pelletier [85] point out that Canadians have a nationalistic view of their healthcare system, and having public data with publicly provided healthcare makes sense to many citizens, including providers and health professionals. Thus, if the BDA platform can produce accurate query results within seconds while the data is still housed in the hospital and/or is securely encrypted, then existing privacy policies should be changed unless public disclosure is deemed unnecessary.

Useful knowledge gained from this study included the identification of the following challenges:

1. Data aggregation. HBase key store qualifiers caused actual storage to double compared to what was expected, although big datasets of clinical, biomedical, and biometric data were processed successfully with Hadoop/MapReduce framework.
2. Data maintenance, Ingestions to the database required the continual monitoring and updating of versions of Hadoop (HDFS), MapReduce and HBase, with limitations persisting for MapReduce.
3. Data integration.Combination of ADT and DAD was shown to be possible and followed current clinical reporting, but a more complex clinical event model of the row keys and column families is required, which would necessitate further testing.
4. Data analysis. A high performance of 3.5 seconds for three billion rows and 90 columns (30TB) was achieved with increasing complexity of queries, but this finding needs to be corroborated when querying real data.
5. Pattern interpretation of application. Health trends could not be discovered using the application; hence further investigation using Hadoop's Machine Learning Libraries (MLLib) is required.

The design of the implemented BDA platform (utilizing WestGrid's supercomputing clusters) is available to researchers and sponsored members. The next step in testing the platform will be to distribute and index the data to ten billion patient data rows across the database nodes and then test the performance using the established methodology. In a later phase, the research plan is to distribute the data across 100 nodes, since Hadoop's HDFS and HBase are theoretically supposed to scale and function better as the number of nodes increases [14,15].

References

1. Chen M, Mao S, Liu Y. "Big Data: A Survey," Mobile Network Application. 2014; 19: 171-209.
2. Viceconti M, Hunter P, Hose R. "Big data, big knowledge: big data for personalized healthcare," IEEE Journal of Biomedical and Health Informatics. 2015; 19: 1209-1215.
3. Hansen MM, Miron-Shatz T, Lau AYS, Paton C. "Big Data in Science and Healthcare: A Review of Recent Literature and Perspectives," Yearbook of Medical Informatics. 2014; 9: 21-26.
4. Manyika J, Chui M, Bughin J, Brown B, Dobbs R, Roxburgh C, Hung B. Big data: The next frontier for innovation, competition, and productivity. Canada Health Infoway, Big Data Analytics in Health - White Paper. 2013.
5. Raghupathi W, Raghupathi V. "Big data analytics in healthcare: promise and potential," Health Information Science and Systems. 2014; 2: 1-10.
6. Foster R. Health Care Big Data is a big opportunity to address data overload. Matchcite.
7. Shah NH, Tenenbaum JD. The coming age of data-driven medicine: translational bioinformatics' next frontier. Journal of the American Medical Informatics Association. 2012; 19: e2-e4.
8. Garrison LPJr. Universal Health Coverage-Big Thinking versus Big Data. Value Health. 2013; 16: S1-S3.
9. Baro E, Degoul S, Beuscart R, Chazard E. Toward a literature-drive definition of big data in healthcare. BioMed Research International. 2015; 9.
10. Wang B, Li R, Perrizo W. Big Data Analytics in Bioinformatics and Healthcare, Medical Information Science Reference. 2014.
11. Kuo MH, Sahama T, Kushniruk AW, Borycki EM, Grunwell D. "Health Big Data Analytics: Current Perspectives, Challenges and Potential Solutions," International Journal of Big Data Intelligence. 2014; 1: 114-126.
12. Grover M, Malaska TJ, Shapira G. Hadoop Application Architectures-Designing Real-World Big Data Applications. O'Reilly Media. 2015.
13. White T. Hadoop - The Definitive Guide: Storage and analysis at internet scale. O'Reilly Publishing, 2015.
14. Lai Wk, Chen YC, Wu TY, Obaidat MS. "Towards a framework for large-scale multimedia data storage and processing on Hadoop platform," J. Supercomp. 2014; 68: 488-507.
15. Dai L, Gao X, Guo Y, Xiao J, Zhang Z. 'Bioinformatics clouds for big data manipulation', Biology Direct. 2012; 7: 1-7.
16. Kumar A, Bawa S. 'Distributed and big data storage management in grid computing', International Journal of Grid Computing and Applications. 2012; 3: 19-28.
17. Merelli I, Pérez-Sánchez H, Gesing S, D'Agostino D, "Managing, Analysing, and Integrating Big Data in Medical Bioinformatics: Open Problems and Future Perspectives," BioMed Research International. 2014; 1-14.
18. Moniruzzaman ABM, Hossain SA. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. International Journal of Database Theory and Application. 2013; 6: 1-14.
19. Lith A, Mattson J. Investigating storage solutions for large data, Master thesis, Department of Computer Science and Engineering, Chalmers University of Technology, Sweden. 2010.
20. Pattuk E, Kantarcioglu M, Khadilkar V, Ulusoy H, Mehrotra S. "BigSecret: A Secure Data Management Framework for Key-Value Stores," IEEE Sixth International Conference on Cloud Computing. 2013; 147-154.
21. Derbeko P, Dolev S, Gudes E, Sharma S. "Security and Privacy Aspects in MapReduce on Clouds: A Survey," Computer Science Review. 2016; 20: 1-28.
22. Martin-Sanchez F, Verspoor K. Big Data in Medicine Is Driving Big Changes. Yearbook of Medical Informatics. 2014; 9: 14-20.
23. Kwakye M. A Practical Approach to Merging Multidimensional Data Models, Master thesis, School of Electrical Engineering and Computer Science, University of Ottawa, Canada, 2011.
24. Chen J, Chen Y, Du X, Li C, Lu J, Zhao S. 'Big data challenge: a data management perspective', Frontiers of Computer Science. 2013; 7: 157-164.

25. Leeper NJ, Bauer-Mehren A, Iyer SV, Lependu P, Olson C, Shah NH, Practice-based evidence: profiling the safety of cilostazol by text-mining of clinical notes'. PloS One. 2013; 8: e634991- e634998.

26. Lependu P, Iyer SV, Fairon C, Shah NH. 'Annotation analysis for testing drug safety signals using unstructured clinical notes'. Journal of Biomedical Semantics. 2012; 3: S5.

27. Raghupathi W, Raghupathi V. "Big data analytics in healthcare: promise and potential," Health Information Science and Systems. 2014; 2: 3.

28. Schadt EE, Linderman MD, Sorenson J, Lee L, Nolan GP, "Computational solutions to large-scale data management and analysis," Nature Reviews. 2010; 11: 647-657.

29. Deepthi K, Anuradha K. "Big data Mining Using Very-Large-Scale Data Processing Platforms," International journal of engineering research and applications. 2016; 6: 39-45.

30. Hashmi A, Ahmad T. "Big Data Mining: Tools & Algorithms," International Journal of Recent Contributions from Engineering, Science & IT. 2016; 4.

31. Zhang YP "i^2 MapReduce: Incremental MapReduce for Mining Evolving Big Data," IEEE Transactions on Knowledge and Data Engineering. 2015; 27: 1906-1919.

32. Narawade R, Jadhav V. "Data Mining of Big Data: The Survey and Review," International Journal of Innovations in Engineering and technology. 2015; 6: 248-252.

33. Vaidya DP, Deshpande SP, "Parallel Data Mining Architecture for Big Data," International Journal of Electronics, Communication and Soft Computing Science and Engineering. 2015; 208-213.

34. Wu X, Zhu X, Wu GQ, Ding W, "Data Mining with Big Data," IEEE Transactions on Knowledge and Data Engineering. 2014; 26: 97-107.

35. Mohammed EA, Far BH, Naugler C. "Applications of the MapReduce programming framework to clinical big data analysis: current landscape and future trends," BioData Mining. 2014; 7: 1-23.

36. Marozzo F, Talia D, Trunfio P. "P2P-MapReduce: Parallel data processing in dynamic cloud environments," Journal of Computer and System Sciences. 2012; 78: 1382-1402.

37. Taylor RC, "An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics," BMC Bioinformatics. 2010; 11: 1-6.

38. Langkafel P. Big Data in Medical Science and Healthcare Management: Diagnosis, Therapy, side Effects. De Gruyter. 2016.

39. Sakr S, Elgammal A. "Towards a comprehensive data analytics framework for smart healthcare services," Big Data Research. 2016; 4: 44-58.

40. Chang RS, Liao CS, Fan KZ, Wu CM. "Dynamic Deduplication Decision in a Hadoop Distributed File System," International Journal of Distributed Sensor Networks. 2014; 2014: 1-14.

41. Madsen LB. Data-Driven healthcare: how analytics and BI are transforming the industry. Jon Wiley and Sons. 2014.

42. Raghupathi W, Raghupathi V, "Big data analytics in healthcare: promise and potential," Health Information Science and Systems. 2014; 2: 1-10.

43. ZooKeeper, "ZooKeeper - Apache Software Foundation project home page," 2016.

44. Sitto K, Presser M. Field Guide to Hadoop - An Introduction to Hadoop, Its Ecosystem, and Aligned Technologies, O'Reilly Media, San Francisco, CA, 2015.

45. Dunning T, Friedman E, Shiran T, Nadeau J, Apache-Drill, O'Reilly Media, San Francisco, CA. 2016.

46. Jurney R. Agile Data Science: Building Data Analytics Applications with Hadoop, O'Reilly Media, San Francisco, CA. 2013.

47. WestGrid. 2016.

48. Benaloh J, Chase M, Horvitz E, Lauter K. "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records,"Proc. ACM Workshop on Cloud Computing Security. 2009; 103-114.

49. Chrimes D, Kuo MH, Moa B, Hu W. Towards a Real-time Big Data Analytics Platform for Health Applications. Int J Big Data Intel. 2016; 4.

50. Chrimes D, Moa B, Zamani H, Kuo MH. "Interactive Healthcare Big Data Analytics Platform under Simulated Performance," IEEE 14th Int. Conf. Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, 2nd Intl. Conf. on Big Data Intelligence and Computing and Cyber Science and Technology Congress. 811-818.

51. Xu J, Shi M, Chen C, Zhang Z, Fu J, Liu CH. "ZQL: A unified middleware bridging both relational and NoSQL databases," IEEE 14th Int. Conf. Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, 2nd Intl. Conf. on Big Data Intelligence and Computing and Cyber Science and Technology Congress. 2016; 730-737.

52. Yang CT, Liu JC, Hsu WH, Chu WCC. "Implementation of data transform method into NoSQL database for healthcare data," International Conference on Parallel and Distributed Computing, Applications and Technologies. 2013.

53. Wang Y, Goh W, Wong L, Montana G. "Random forests on Hadoop for genome- wide association studies of multivariate neuroimaging phenotypes," BMC Bioinformatics. 2013; 14: 1-15.

54. Wang S, Pandis I, Wu C, He S, Johnson D, Emam I, et al. "High dimensional biological data retrieval optimization with NoSQL technology," BMC Genomics. 2014; 15: S3.

55. Scott J. "Apache Spark vs. Apache Drill. Converge Blog, Powered by MapR.," 2015.

56. Nishimura S, Das S, Agrawal D, Abbadi AE. MD-HBase: design and implementation of an elastic data infrastructure for cloud-scale location services. Distributed and Parallel Databases. 2013; 31: 289-319.

57. Nguyen AV, Wynden R, Sun Y. "HBase, MapReduce, and Integrated Data Visualization for Processing Clinical Signal Data," AAAI Spring Symposium: Computational Physiology. 2011.

58. Sun J. "Scalable RDF store based on HBase and MapReduce," Advanced Computer Theory and Engineering (ICACTE), 3rd Int. Conf., Hangzhou, China. 2013; 20-22.

59. Chawla NV, Davis DA. "Bringing Big Data to Personalized Healthcare: A Patient- Centered Framework," J Gen Intern Med. 2013; 28: S660-S665.

60. Sullivan PO, Thompson G, Clifford A. "Applying data models to big data architectures," IBM J Res & Dev 2014; 58: 1-12.

61. Freire SM, Teodoro D, Wei-Kleiner F, Sundsvall E, Karlsson D, Lambrix P. "Comparing the Performance of NoSQL Approaches for Managing Archetype-Based Electronic Health Record Data," PLoS One. 2016; 11: e0150069.

62. Frey LJ, Lenert L, Lopez-Campos G. "EHR Big Data Deep Phenotyping Contribution of the IMIA Genomic Medicine Working Group," Year Med Inform. 2014; 15: 206-211.

63. Mayer-Schönberger V, Cukie K. Big data: A revolution that will transform how we live, work, and think, Houghton Mifflin Harcourt. 2013; 256.

64. Chen Y, Alspaugh S, Katz R. "Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads," Proceedings of the VLDB Endowment. 2012; 5: 1802-1813.

65. Greeshma AL, Pradeepini G. "Input split frequent pattern tree using MapReduce paradigm in Hadoop," J. Theo. App. Inform. Tech. 2016; 84: 260-271.

66. Maier M. "Towards a Big Data Reference Architecturem," MSc Thesis. Eindhoven University of Technology, Netherlands, 2013.

67. Yu SC, Kao QL, Lee CR. "Performance Optimization of the SSVD Collaborative Filtering Algorithm on MapReduce Architectures," IEEE 14th Int. Conf. Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress. 2016; 612-619.

Citation: Chrimes D, Kuo MH, Moa B and Kushniruk AW. Interactive Big Data Analytics Platform for Healthcare and Clinical Services. SM J Biometrics Biostat. 2018; 3(2): 1030.

68. Hripcsak G, Albers DJ. Next-generation phenotyping of electronic health records. J Am. Med. Inform. Assoc. 2013; 20: 117-121.

69. Wang F, Lee R, Liu Q, Aji A, Zhang X, Saltz J. "Hadoop-GIS: A high performance query system for analytical medical imaging with MapReduce," In: Atlanta-USA: Technical Report, Emory University. 2011; 13: 1-13.

70. Markonis D, Schaer R, Eggel I, Müller H, Depeursinge A. "Using MapReduce for large-scale medical image analysis," Healthcare Informatics, Imaging and Systems Biology (HISB), IEEE Second International Conf. La Jolla, CA, USA, September. 2012; 28: 1-10.

71. Rabkin A, Katz RH. "How Hadoop Clusters Break," IEEE Software, July/August. 2013; 88-95.

72. Moise D, Trieu TTL, Bouge L, Antoniu G. "Optimizing intermediate data management in MapReduce computations." Proceedings of the first international workshop on cloud computing platforms, ACM. 2011; 1-7.

73. Ruan G, Zhang H, Plale B. "Exploiting MapReduce and data compression for data-intensive applications." Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, ACM. 2013; 1-8.

74. Xue Z, Shen G, Li J, Xu Q, Zhang Y, Shao J. "Compression aware i/o performance analysis for big data clustering." Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, ACM. 2012; 45-52.

75. Wang Y, Xu C, Li X, Yu W. "JVM-bypass for efficient Hadoop shuffling." In: IEEE 27th International Symposium on Parallel & Distributed Processing (IPDPS). 2013; 569-578.

76. Yu W, Wang Y, Que X, Xu C. "Virtual shuffling for efficient data movement in MapReduce." IEEE Transactions on Computers. 2015; 64: 556-568.

77. Yan D, Yin XS, Lian C. "Using memory in the right way to accelerate big data processing." Journal of Computer Science and Technology. 2015; 1; 30-41.

78. Ahmad F, Chakradhar ST, Raghunathan A, Vijaykumar T. "Shufflewatcher: Shuffle-aware scheduling in multi-tenant MapReduce clusters." Proceedings of the USENIX conference on USENIX Annual Technical Conference. USENIX Association, 2014; 1-12.

79. Nabavinejad SM, Goudarzi M, Mozaffari S. "The Memory Challenge in Reduce Phase of MapReduce Applications," J. Latex Class Files. Transactions on Big Data IEEE. 2016. 14.

80. Chung WC, Lin HP, Chen SC, Jiang MF, Chung YC. "JackHare: a framework for SQL to NoSQL translation using MapReduce," Autom. Softw. Eng. 2014; 21: 489-508.

81. Dutta H, Demme J. Distributed Storage of Large Scale Multidimensional EEG Data using Hadoop/HBase, Grid and Cloud Database Management, New York City: Springer. 2011.

82. Dean J, Ghemawat S. "MapReduce: A Flexible Data Processing Tool," Comm. ACM. 2010; 53: 72-77.

83. Moselle K. "Data Management in the Island Health Secure Research Environment," Enterprise Architecture at Vancouver Island Health Authority, Working Draft 5, Victoria, BC. 2015.

84. Dufresne Y, Jeram S, Pelletier A. "The True North Strong and Free Healthcare? Nationalism and Attitudes Towards Private Healthcare Options in Canada." Canadian Journal of Political Science. 2014; 47: 569-595.