Secure, Scalable and Fold-tolerant Personal Health Record Platform

Zaur Fataliyev[1], Khalid Huseynov[2], Elkhan Jabarov[3]

[1]LG Electronics Inc., [2]NFLabs, [3]Korea University, Seoul, Republic of Korea

[1]zaur.fataliyev@lge.com, [2]khalidhnv@nflabs.com, [3]ejabarov@korea.ac.kr

***Abstract*—This paper proposes a new Personal Health Record Platform. Proposed platform has edge in terms of scalability, security and fold-tolerance. The platform is designed to handle information flow among multiple data centers, process raw data, provide API for third-party entities, share information between hospitals, and authenticate access to user data, record data from smart devices and other useful and essential functionality. Multiple data centers are being handled in the platform where each data center utilizes Apache Hadoop ecosystem with all the healthcare system logic implemented on top of it.**

Keywords: personal health record; Hadoop; smart device; distributed processing; Hadoop ecosystem

I. Introduction

Personal Health Record (PHR) is a digital health record of patient that contains all relevant heath related data of patient including medical history and data from wearable devices. Sharing such data between hospitals promises cost reduction, better health care, less paperwork, less medical errors, better disease management and other valuable advantages. Important issues on subject such as security and sharing protocols are is being attempted to be resolved [1, 2], but remains unresolved as security concerns being remained as major issue. Since unauthorized access to such data can have fatal consequences [3], security aspect of the sharing PHR is crucial.

Since there are some unresolved issues related to rules and regulations for sharing PHR data between hospitals and access to patient data, different organizations such as HL7 is attempting to resolve such issues are being developed by HL7 [4], and in future such platforms will be of wide use. Despite the fact that there are many potential benefits of using PHR platforms, alongside with legal issues on the matter, there are numerous management, technology and security issues in the domain as well. The aim of this paper is to address feasible solution to technological side of the PHR Platform, proposing a secure architecture based on Hadoop ecosystem.

Emerging widespread usage of smart devices for activity measurement, portable medical devices as well as other devices which relentlessly generate data require scalable storage and intelligent algorithms for processing. In proposed platform, all kinds of data inflow has been considered and there is room for future expansion of such data streams. In such platforms, the connection of multiple healthcare databases and their synchronization brings in multiple challenges. One of the important issues is the tradeoff between consistency and availability (CAP theorem) in the context of large data volume processing. Considering existing state-of-the-art distributed processing tools, we decided to utilize data center approach for centralizing multiple healthcare systems. There can be multiple data centers, each in charge of certain location, while all of them connected to each other. Each data center utilizes Apache Hadoop "ecosystem" with all the healthcare system logic implemented on top of that. Also our architecture provides web interface with its own API. Besides this, platform empowers custom developed ML algorithm to process raw data for structuring data, personalized recommendations and statistical insight of accumulated health data.

II. Proposed Personal Health Record Platform

In proposed PHR Platform, following issues is being tackled in the design and implementation of proposed framework.

- *All related patient data is included in user profile*
- *Database is scalable*
- *Platform handles data exchange between more than one data centers*
- *Platform contains Machine Learning Engines to process raw datasets*
- *Users have the ultimate power over their data in terms of sharing*
- *Hospitals and doctors are able to push and pull data from user profile in Platform after authorization*
- *There are more than one Data Center*

A. Overview of Platform

Outline of proposed platform is given in Figure 1. Details of implementation is given in succeeding sections. As given in the figure, platform compromised of User, Hospitals and Doctors on client side, platform itself on backend.

Application Programming Interface (API) is for interacting with platforms with different client-side applications on various platforms such as Android, iOS, Windows and etc.

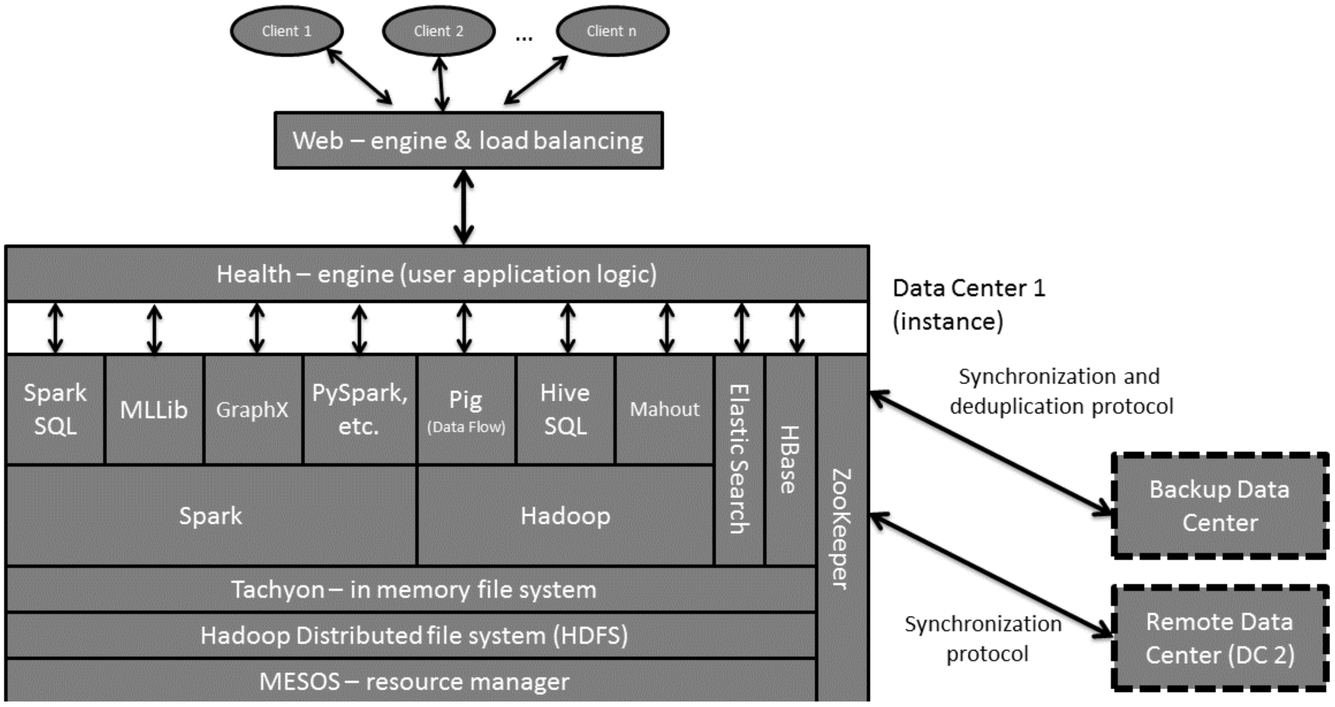

Figure 1: Distributed Computing System of Proposed PHR Platform based on Hadoop Ecosystem

PHR Platform contains Distributed Computing System, Engines and functionality of handling remote Data Centers (Figure 1).

Authorization is for controlling hospital's or doctor's access to user data (Figure 2).

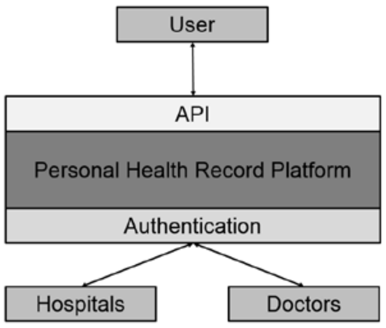

Figure 2: Outline of Proposed PHR Platform

B. *User Data flow to PHR Platform*

User data includes following sections of data sets and streams (Figure 3)

- **Personal Information** entries contains but not limited to following data
  - Identification Values: Name, Surname, Age, Address and others
  - Physical Information: Blood type, Height, Weight and others
- **Insurance Information** entries include information about insurance such as company or national health insurance.
- **Wearable Devices** data contains data sets from wearable smart devise such as physical activity measuring devices, blood test devices, blood pressure measuring devices.
- **Medical History** entry contains data of previous medical treatments, allergies, drug history, addictions and other data in the domain.

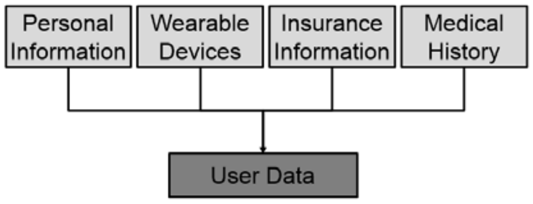

Figure 3: Data inflow to PHR Platform

C. *Personal Health Record Platform Architecture*

Figure 4 shows the details of PHR Platform. Platform is designed in a way that can handle data inflow from users. Our choices of environments for implementation has been justified in discussion section. In a nutshell, all collected data is managed in platform, providing environment and protocols for secure and intelligent processing of incoming data. Distributed approach with Hadoop ecosystem to processing of large volumes of data ensures the system can handle large data considering health related data is growing very fast.

D. User profile in Platform

User profile in database contains following sections of information in database.

- **Personal Information**: basic personal information of user
- **Insurance information**: sponsoring insurances
- **Medical history**: available previous medical records
- **Examination**: examination results such as blood test or eye sight test
- **Contacts**: personal and emergency contacts
- **Sensitive Information**: sensitive medical information such as HIV results, pregnancy results and any other information labeled as sensitive data upon choice of user.
- **Nutrition**: daily nutrition records
- **Smart device data**: data from smart devices such as running activity, portable smart blood monitoring device
- **Appointments**: hospital or doctor appointments
- **Medications**:medication routines, tablet schedules and diet agenda by doctor
- **Recommendations and Warnings**: diet, physical activity recommendations and warnings by doctor

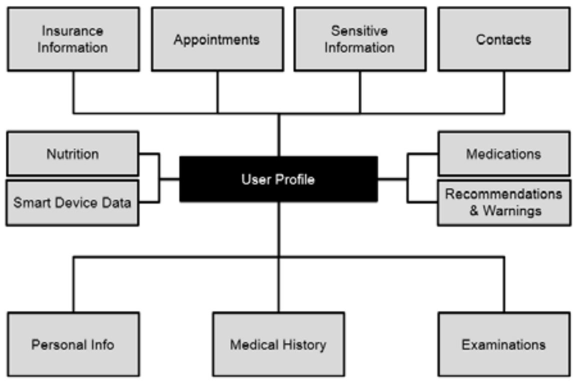

Figure 4: Details of PHR Platform

E. Machine Learning Engines

Machine Learning (ML) Engines is included into the PHR Platform for intelligent decision-making, recommendation for patients and statistical information measurements.

1) Optical Character Recognition (OCR) Engine

This unit processes images taken with client camera. Primary purpose of OCR Engine is to record nutrition content of food or beverages, contents and side effects of drugs and other related information on intake.

2) Recommendation Engine

a) Doctor & Hospital Recommendation

This recommendation engine recommends doctors and hospitals to patient based on user-feedback, price-range, distance and resource availability.

b) Health Diet Recommendation

This recommendation engine recommends fruits, vegetables and other edible items based on patient condition.

c) Workout Recommendation

This recommendation engine recommends specific workout routine based on user preference or patient condition.

3) Statistical Measurements

This engine monitors user-base, extracts general statistical insights about health, birth-rate, and other summarized information.

F. Client-side Applications

1) General User Application – applications for user side based on various platforms such as android, iOS, windows and etc.

2) Hospital Application – application for hospitals, integrated into hospital ERP.

3) Doctor Application – application for doctors for direct communication between doctor and patient.

4) Special-purpose Application – special-purpose oriented application such as maternity application, infant care application and etc.

5) Third-party Applications – applications developed by third-party entities.

III. Discussion

Nowadays, the need for sharing health data is becoming more important than ever. Although it brings several issues with it in terms of security and processing. Emerging widespread usage of fitness devices, mobile devices and other smart devices creates a perfect environment to be centeralized processing and sharing of such health data in a way that will change healthcare system for better for everyone.

Proposed platform takes these into account and proposes a secure and scalable healthcare platform for connected all patient data with hospitals and doctors in a secure way. The platform has ML engines for intelligent decision-making that will process user data and give recommendations to users.

Our choice of cluster resource manager fell onto Mesos [5] as opposed to YARN [6]. Mesos uses Linux container groups [7], whereas YARN utilizes simple Unix processes. It gives Mesos more fine-grained resource management capabilities. Moreover, Mesos provides a choice to the client process for running the job, whereas YARN has monolithic choice without any feedbacks from the client. It makes Mesos more scalable resource manager in comparison with the YARN.

Hadoop Distributed File System (HDFS) [8] is a distributed storage system on top of cluster of commodity hardware. Tachyon [9] is in-memory file system that's built on top of HDFS for acceleration purposes. Normally HDFS writes into persistent memory (e.g. HDD) whereas, Tachyon is made to work with the RAM and serves as a cache layer on top of HDFS. This accelerates the performance of HDFS.

The computing framework of Apache Hadoop [10] is based on the MapReduce [11] paradigm proposed by Google. A MapReduce program is composed of a Map() procedure that performs filtering and sorting using key values and a Reduce() procedure that performs a summary operation on the values with the same key. Using this approach we can process large amounts of data in parallel in short amount of time, making our system highly scalable.

Pig is a procedural and higher level language for programming on Hadoop framework. The language for this platform is called Pig Latin [12]. Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for RDBMS systems. Pig Latin can be used for implementing some data flows, making Hadoop to look as a combination of functional programming with SQL.

Apache Hive is a database engine built on top of Hadoop ecosystem for providing data summarization, query, and analysis [13]. It provides an SQL-like language called HiveQL [14] with schema on read and transparently converts queries to map/reduce jobs. Furthermore, Mahout is a machine learning framework built on top of Hadoop ecosystem. It can serve multiple purposes by providing API for applications running in the health-engine.

An Apache Spark project is an in-memory implementation of the Apache Hadoop project. Spark's in-memory primitives support up to 100 times faster performance for certain applications [15]. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well suited to machine learning algorithms. The machine learning library of Spark is called MLLib [16] as opposed to Mahout of Hadoop ecosystem. Spark SQL [17] is the counterpart of Hive SQL in the Spark ecosystem. Moreover, Hive SQL can be run on top of Spark SQL as well. GraphX [17] library is a special graph processing API on top of Spark. This API can be effectively utilized by applications running in the health-engine. Finally Spark supports API for Python, Scala and few others languages. This would be useful for implementing some logic not existing in current APIs of Spark, and Hadoop.

The Elastic Search [18] and HBase [19] don't perform any MapReduce processing. On the other hand, they're built on top of HDFS (or Tachyon in our case) for directly querying some large tables directly from the HDFS.

This entire stack describes the architecture of each host in our data center. There can be different architectures [20] for data centers as well. In case of failures, we should have back-up data center as well. The back-up data center should be continuously synchronized with our currently working data center. The remote data center is required for serving the remote healthcare institutions. Basically web-engine should choose the closest data center for directly the queries.

IV. Conclusion

In the paper, a secure, scalable and fold-tolerant PHR Platform has been proposed. Platform has tackled all technical-theoretical side of all problems in the design of such platforms. Proposed platform collects all types of patient/user related data, processes those data, makes decisions based on processed data, defines protocols for sharing data between multiple data sources and authorizes access to data by hospitals and doctors and provides API for third-party developers. Proposed platform is built on Hadoop, thus making data storage and processing scalable and secure. Integration of multiple PHR databases gives rise to multiple problems in terms of their connection and synchronization. Consistency and availability is a tradeoff when it comes to large volume processing. Thus, distributed approach to processing is a feasible in such processing. Proposed platform uses multiple datacenter approach to stated problem and integrates all data centers in Hadoop ecosystem. All system logic implementation is done on top of this ecosystem and furthermore, it provides web interface with its API for users, hospitals and other third-party entities.

References

[1] R. Mahncke, P. Williams, "Secure transmission of shared electronic health records: A review", 4th Australian Information Security Management Conference, Australia, 5th December, 2006

[2] "Securing Medical SaaS Solutions using a Novel End-toEnd "J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2.Oxford: Clarendon, 1892, pp. 68-73.

[3] Zhang R, Liu L. "Security models and requirements for healthcare application clouds",The 3rd IEEE International Conference on Cloud; July 5-10, 2010; Miami, FL, USA. New York, NY: IEEE; 2010

[4] Health Level Seven International. Available: http://www.hl7.org

[5] Apache Mesos. Available: http://mesos.apache.org/

[6] Hadoop YARN. Available: http://hortonworks.com/hadoop/yarn/

[7] Linux Containers. Available: https://linuxcontainers.org/

[8] Hadoop Distributed File System. Avaiable: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[9] Tachyon. Avaiable: http://tachyon-project.org/

[10] Apache Hadoop. Available: http://hadoop.apache.org

[11] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, in: Proc. 6th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2004, San Francisco, USA, Dec. 2004

[12] Hadoop: Apache Pig. Avaiable: http://pig.apache.org

[13] Venner, Jason (2009). Pro Hadoop. Apress. ISBN 978-1-4302-1942-2.

[14] HiveQL Language, Manual https://cwiki.apache.org/confluence/display/Hive/LanguageManual

[15] R. Xin, J. Rosen, M. Zaharia, M. Franklin, S. Shenker, I. Stoica, "Shark: SQL and Rich Analytics at Scale." June 2013.

[16] Spark MLLib. https://spark.apache.org/docs/1.2.1/mllib-guide.html

[17] GraphX – Apache Spark. Avaiable: https://spark.apache.org/graphx/

[18] Elastic Search. Avaiable: https://www.elastic.co

[19] Apache HBase. Avaiable: http://hbase.apache.org

[20] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architectures," Concurrency and Computation: Practice and Experience, vol. 25, no. 12, pp. 1771-1783, 2013.