RFID Based Biometrics Analysis Over Cloud

Faisal Rahman, Mohiuddin Solaimani, Latifur Khan
Department of Computer Science
The University of Texas at Dallas
Richardson, TX
{fxr140430, mxs121731, lkhan}@utdallas.edu

ABSTRACT

Radio Frequency Identification (RFID) is still a new technology when it comes to data acquisition. RFID comes in many different shapes and forms, as well as in operating frequencies. In this project we have used RFID technology to acquire biometrics information into the tags. Ultra-high frequency or UHF (860-960 MHz) tags have been used in the project. These tags are well known for generating long read/write ranges relative to other operating frequencies. Later we will scan this tag and collect patient's information. We will build a statistical model using this massive amount of continuous information using Big data technology. We have used Apache Spark, a distributed framework for processing patient's biometrics. Spark is a fast and general engine for both large-scale batch and stream data processing. It is faster than Hadoop MapReduce. We will implement learning algorithms to build a model of the patient's normal characteristics in our distributed Spark based framework. Later we will use this model to analyze the future patient's biometrics. If it deviates from the normal characteristics of the model, we will give proper medicare or suggestions accordingly.

Categories and Subject Descriptors

1 [**Healthcare application in distributed system**]: Metrics—*RFID based data acquisation, statical healthcare data analysis using Big Data.*

General Terms

Theory

Keywords

RFID, Biometrics, Big Data, Apache Spark

1. INTRODUCTION

Due to its unique advantages, RFID technology already has wide applications. Imagine going shopping before a long weekend. What probably comes to mind are overcrowded stores, impatient shoppers and long checkout lines. Waiting in these lines can be tiring and frustrating. Now imagine a situation in which you put the items you wish to purchase in your shopping bag and simply exit the store. Without even opening your bag, the appropriate amount of money is deducted from your bank account. In the midst of this process, the supermarket automatically updates its inventory, and the system orders more of the items needed [14].

Several companies are interested in correlating RFID tags with internet links [10]. Suppose John Doe wants to go to a famous restaurant and order food to go. But he wants to walk in and walk out with his order without the wait and avoid the rush. He places the order over the internet and when he walks in the restaurant and passes the antenna, the RFID tag in his pocket trigger the restaurant personal of his arrival and delivers his food without the wait. In this process, his credit card is charged and his order is saved in a system for future reference.

In Health-Care the application of the RFID has a broader usage [12]. A person's medical history or biometrics can be stored in RFID tags. In 3rd world country where hospital are overcrowded with limited medical personal, patientâĂŹs medical history can easily be updated by reading the tag. During natural calamities, i.e., earthquake, hurricane emergency personal can obtain biometrics for the affected people in case they are unconscious or not able to speak due to injury.

We should reshape the way we think about RFID. We mean that we should not fixate on acquiring data using RFID, but rather we can think about the applications that can be improved with the availability of RFID data. As an example, retailers should think about how they can get data about shoppers' behavior from the moment they walk in a store and the kind of analysis they can do to optimize processes accordingly. Beyond the process identification, retailers can benefit from the availability of full RFID solutions that help them incorporate data generated from RFID systems with the data warehousing, analytics and mostly homegrown software applications used to manage operations. Another example may be the statistical analysis of patient's bio-metric characteristics. We can capture patient's bio-metric information using RFID. Later we can build statistical model using this data. Afterward, we can use this model to analyse patient's health condition and give them suitable medicare accordingly.

As we envision a complete data analysis solution using RFID technology, we cannot think without BIG Data. Big data is used for vast amounts of information that can be interpreted by building a statistical model to analyze some patterns or trends. Organizations leverage Big data by gathering records and information captured and then interpreting it with analytics. Like other industries, Big data has become a key factor in healthcare. It has several applications like predicting over an entire state's health records to pinpoint people who are at risk for certain ailments. Using Big data we can help people giving early warning signs and

also improve patient safety.

Currently we are capturing patient's biometrics information using RFID. We can build statistical model with this information. We can suggest informative lifestyle choice for certain patient with the help of this model. For example, we can suggest a diabetic patient suitable dieting based on glucose level.

We can do the trend analysis of patient's biometrics statistics and try to correlate them. For example, we can take blood pressure or heart rate of patients and build a model. For a particular patient, we can do the trend analysis by showing his blood pressure or heart rate graphically at different time intervals and see how much deviation he has from the expected behavior. Furthermore, we can build the statistical model from these data and also can find the relationship between them. Later we use this model to predict the probable healthcare for a particular patient.

In this paper, we have presented RFID based distributed framework that can fetch continuous bio-metrics data from patients, and store in persistent storage. Next, in the long run analytics will be carried out on gathered data in real time (near) for trend analysis and recommendation.

The paper is organized in the following ways. Section 2 presents basic components of our framework for RFID data acquisition, and NoSQL storage. Section 3 presents our RFID distributed framework. Section 4 presents current status of our work. Section 5 presents future work. Section 6 has acknowledgement.

2. FRAMEWORK

Our framework relies on RFID for data acquision and Apche Spark for data management. In the following subsection We will present each of these and argue why we are using these for our framework.

2.1 RFID

RFID tags are used for data storage and data acquisition for patient biometrics information. For the data acquisition part, the hardware is composed of a reader, antenna, and the UHF tags. The application software is provided by the RFID manufacturer, RFID Inc. The software is customized for this application by writing additional codes in Visual Basic. In the front-end of the source code, a user interface (UI) is implemented to input patient information. This information is then written to the individual tags. A snapshot of the UI is shown in figure 1. As of today, there are eight biometrics information such as blood pressure, glucometer reading, ECG, etc. are stored in the tags. This information can be easily read and edited (re-written) by the UI. We will analyze this information using Big data that will be discussed later sections.

RFID tag uses 96 bits of memory to store the EPC (Electronic Product Code) as highlighted in yellow in figure 2. This is enough space to store 24 hexadecimal characters (0-9, A-F). The 1st column in the figure 2 shows 3 memory banks i.e. 0-3. The memoty bank is divided into sub-address. Each sub address contains memory bits. The EPC is stored in memory bank 01 from sub address 02-07. The user memory highlighted in green is located 3rd memory bank and for these tags the user memory is 512 bits.

The biometric and patient information is stored in both the EPC and in the user memory of the individual tags. Instead of product identifier, our source code uses EPC bits to

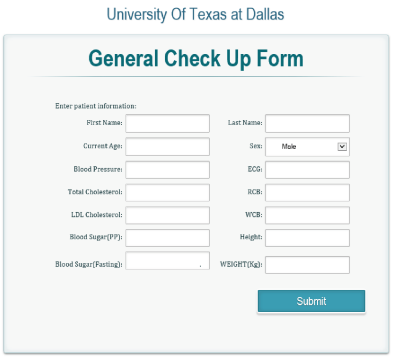

Figure 1: Snapshot of the User Interface

flag each of the biometric fields. For example if the patient has blood pressure then the bit representing the blood pressure field will be set to "1". 64 EPC bits are used for this purpose and the remaining 32 bits are used for tag identification/ manufacturer reserved bits. Individual or multiple tags in the RF field are identified independently through the 32 Tag identifier (id) bits.

Memory Bank	Sub Address	Description	Memory Size
00	00	Tag manufacturer only	32 bits
	01	Tag manufacturer only	32 bits
01	00	CRC	16 bits
	01	Protocol 3000=No Data; 3400=Data	16 bits
	02	Biometric	16 bits
	03	Biometric	16 bits
	04	Biometric	16 bits
	05	Biometric	16 bits
	06	Tag id	16 bits
	07	Tag id	16 bits
10		Tag manufacturer only	96 bits
11	00-31	User Memory	512 bits

Figure 2: Organization of the Tag memory

2.2 Apache Spark

Apache Spark [6] is an open-source distributed framework for data analytics. Spark also uses Hadoop [1] for distributed file system. It can work on the top of next generation Hadoop cluster called YARN [9]. Spark avoids the I/O bottleneck of the conventional two-stage MapReduce programs. It provides the in-memory cluster computing that allows user to load data into a cluster's memory and query it efficiently. This increases its performance over traditional MapReduce to a great extent.

Choosing the appropriate framework for statistical analysis using Big data is a challenging task. It solely depends on the nature of the application. Several interactive and offline applications in healthcare system use this distributed statistical model. Recently we have seen several Big Data frameworks (e.g., Hadoop [1], Map Reduce [13], HBase [2], Mahout [4], etc.) that address the scalability issue. However, they excel at batch-based processing. Apache Storm [8] and Apache S4 [5] are distributed frameworks that process only stream data. Apache Spark [6, 7] is the distributed framework that offers both stream data processing and offline batch processing using next generation Hadoop cluster-

ing. Spark has an advanced DAG (Direct Acyclic Graph) execution engine that supports cyclic data flow and in-memory computing. It makes Spark faster than other distributed frameworks. Spark is 100x faster than Hadoop MapReduce in memory, or 10x faster on disk [6]. It is also faster than Storm and S4 [18]. Overall, it generates low latency, real-time results. Its has instant productivity and no hidden obstacles. It also has easy cluster setup procedure.

Apache Spark has simple architecture (in Fig 3). It has two key concepts: Resilient Distributed Datasets (RDD) and directed acyclic graph (DAG) engine.

2.2.1 *Resilient Distributed Dataset (RDD)*

RDD [17] is a distributed memory abstraction. It allows in-memory computation on a large distributed clusters with high fault-tolerant. Spark has two types of RDDs: parallelized collections that are based on existing programming collections (like list, map, etc.) and files stored on HDFS. RDDs performs two kinds of operations: transformations and actions. Transformations create new datasets from the input or exiting RDDs (e.g. map or filter), and actions return a value after executing calculations on the dataset (e.g. reduce, collect, count, saveAsTextFile, etc.). Transformations are lazy operations that define only the new RDD while actions are performed the actual computation and calculate the result or write to the external storage.

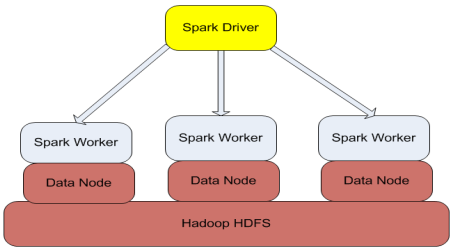

Figure 3: Spark Architecture

2.2.2 *Directed acyclic graph (DAG) engine*

Whenever the user runs an action on RDD, a directed acyclic graph is generated considering all the transformation dependencies. This eliminates the traditional MapReduce multi-stage execution model and also improves the performance.

3. DISTRIBUTED FRAMEWORK; RFID

Our RFID-based data analytic system is described in figure4. Each patient will be identified by each tag independently through the tag id bits. The tag can be in a form of a wrist-band and when it passes though the antenna, the biometric information gets read into the UI or, new information can be written to the tag. Information in the tag can easily be updated and edited as long the tag/tags are within the writable distance from the antenna, which is about 15 feet. The snapshot of the UI is shown figure1.

We capture the patient's biometrics information using RFID tag. The patient's information can be written locally into the RFID. When the RFID antenna reads the RFID tag, the data inside it passes through IP network. We have used Apache Kafka (a message broker) [3] which ships these data

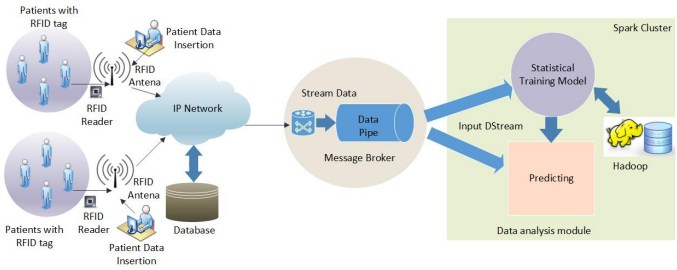

Figure 4: RFID based distributed framework

to our data analysis module. In this project we have used Apache Spark for data analysis. Our data analysis module collects data from all RFID reader. It builds the statistical model with machine learning technique and this model will be used to predict the behavior or trend of patient's certain bi-metric statistics. We will also use patient's additional information like names, address, etc. to a database.

4. WORK PROGRESS

4.1 Environment Setup

4.1.1 *RFID*

At present we have implemented the serial version of the code. This version the RFID system, i.e. reader antenna, tags are all connected with a single PC. The TCP/IP version of the source code is in code writing/testing stage. This version will have the capability to interact with multiple reader, antenna and tags and be able to transfer the data to a server in real-time.

The original version of the source code is provided to us by RFID Inc. The code had limitation so we modified the code. We implemented several functionalities such as a loop write so multiple words can be sequentially written rather than before one word at a time. Now, with a single write instruction, multiple words can be written in the user memory of the tag. We also added a CRC for data validation on the EPC bits. A polynomial of degree 4 is used to calculate the cyclic redundancy check for the flags in the EPC bits.

4.1.2 *Spark Cluster*

Our Spark cluster consists of 5 VMware ESXi[15](VMware hypervisor server) 5.x systems. Each of the system has Intel(R) Xeon(R) CPU E5-2695 v2 2.40GHz processor, 64 GB DDR3 RAM, 4 TB hard disk and dual NIC card. Each processor has 2 sockets and every socket has 12 cores. So, there are 24 logical processors in total. All of the ESXi systems have 5 virtual machines installed in it. Each of the virtual machine has configured with 8 vCPU, 16 GB DDR3 RAM and 1 TB Hard disk. As all the VMs are sharing the resources, performance may vary in runtime. We have installed Linux Centos v6.5 64 bit OS in each of the VM along with the JDK/JRE v1.7. We have designed a patient's biometrics analysing system on this distributed system using Apache Spark. We have installed Apache Spark version 1.0.0. We have also installed Apache Hadoop NextGen MapReduce (YARN) [9] with version Hadoop v2.2.0 and form a cluster. Apache Spark uses this YARN cluster for distributed file system.

4.2 Experimental Results

We performed our experiments using commercial FCC-compliant equipment, namely Ultra High Frequency (UHF) readers from RFID Inc. (model IP67), for an antenna, circular-polarized, (model UHF-A18C-10) and a liner vertical polarized (model UHF-A78V-10). We used UHF tags from RFID Inc. as well

The experiments were conducted in a lab environment with moderate radio reflection / interference anomalies. We tested single tag and multiple tags. We positioned tags perpendicular to each other whenever possible, and spread the tags far apart in space in order to minimize tag occlusion by other tags and/or objects.

As with linear antennas, circular antenna experiments show a dramatic double-digit average improvement in reading and writing capabilities. The detection probabilities for circular antennas are higher than for linear ones because the orientation of tags with respect to the reader antennas varies widely [11].

We have identified several test subjects both male, female and childs with existing medical condition for testing the system. We have created a chart shown in figure 5 to reference while inputing data in the UI. The biometrics information have catagorized as Field Name. Each field is further divided into age groups with corresponding maximum and minimum range of that field.

Field Name	*Category*	*Range*
Systolic Blood Pressure	Adult (>18)	100-180
(mm Mercury)	Baby: 0-5 yrs	80-125
	Child: 6-12 yrs	90-135
	Teen: 13-17 yrs	100-150
Diastolic Blood Pressure	Adult (>18)	60-110
(mm Mercury)	Baby: 0-5 yrs	30-85
	Child: 6-12 yrs	50-95
	Teen: 13-17 yrs	60-100
Sugar, Fasting	Adult (>12)	70-140
(milligrams/deciliter)	Baby: 0-5 yrs	70-200
	Child: 5-11 yrs	70-160
Sugar, Postprandial	Adult (>12)	70-200
(milligrams/deciliter)	Baby: 0-5 yrs	150-220
	Child: 5-11 yrs	120-200
EKG		0-5
(millivolts)		
Total Cholesterol	Adult (>12)	80-280
(milligrams/deciliter)	Child: 0-11 yrs	70-230
LDL Cholesterol	Adult (>12)	30-200
(milligrams/deciliter)	Child: 0-11 yrs	25-160
Red Blood Count	Men	4.7-6.1
(millions/microliter)	Women	4.2-5.4
	Children	4.6-4.8
White Blood Count	Adult (>18)	4.5-11
(thousands/microliter)	Baby (0-3 days)	9-35
	Baby (4-30 days)	5-21
	Baby (1-24 months)	6-17.5
	Child (2-8 years)	5-15.5
	Child (9-18 years)	4.5-13.5

Figure 5: Biometrics classification

5. FUTURE WORK

One of the issues with RFID tags is it cannot be read with 100 percent accuracy (Rothfeder, 2004) [16] in the real world due to factors such as limitations in the reading range, tag orientation or interference. To mitigate the problem of imperfect read-rates, multiple readers or a reader with multiple antennas are commonly used for entry way, portal or overhead scanning. We would like to implement a multiple antenna/reader system that would send data through the TCP/IP protocol. We also like to experiment with multi-tag objects and analyze the test results. At present, we are experimenting with passive tags and we would like to see the effect on semi-passive tags with current system configuration. A typical test scenario will be, each of the subjects will be given a tag. Biometrics information will be first loaded to the tag though UI. Then tags in the field will be read. Once the data is loaded back to the UI, it will be processed for big data evaluation.

6. ACKNOWLEDGEMENT

This material is based upon work supported by NSF Award No. CNS 1229652. We would like to acknowledge Colorado based company RFID Inc. for their support in this project.

7. REFERENCES

[1] Apache hadoop. http://hadoop.apache.org/.
[2] Apache hbase. https://hbase.apache.org/.
[3] Apache kafka. http://kafka.apache.org/.
[4] Apache mahout. https://mahout.apache.org/.
[5] Apache s4. http://incubator.apache.org/s4.
[6] Apache spark. http://spark.apache.org/.
[7] Apache spark streaming. http://spark.apache.org/streaming/.
[8] Apache storm. http://storm.incubator.apache.org/.
[9] Yarn. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/yarn.html.
[10] S. Ahuja and P. Potti. An introduction to rfid technology. *Communications and Network*, 2(03):183, 2010.
[11] L. Bolotnyy, S. Krize, and G. Robins. The practicality of multi-tag rfid systems.
[12] D. Clarke and A. Park. Active-rfid system accuracy and its implications for clinical applications. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 21–26. IEEE.
[13] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
[14] A. N. Nambiar. Rfid technology: A review of its applications. In *Proceedings of the world congress on engineering and computer science*, volume 2, pages 20–22, 2009.
[15] VMware. vsphere esx and esxi info center.
[16] L. Wang, B. A. Norman, and J. Rajgopal. Placement of multiple rfid reader antennas to maximise portal read accuracy. *International Journal of Radio Frequency Identification Technology and Applications*, 1(3):260–277, 2007.
[17] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing.
[18] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: A fault-tolerant model for scalable stream processing. Technical report, 2012.