IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. VIII (Sep-Oct. 2014), PP 81-94
www.iosrjournals.org

The Architectural Design of Healthcare Service with Big-Data Storing Mechanism

Md. Salahuddin[1], Shaikh Muhammad Allayear[2] and Sung Soon Park[3]
Department of Computer Science and Engineering
East West University, Bangladesh[1,2] and Anyang University, Korea[3]

***Abstract*:** *Healthcare is the diagnosis, treatment, and prevention of disease, illness, injury, and other physical and mental impairments in human beings. Healthcare is delivered by practitioners in allied health, dentistry, midwifery-obstetrics, medicine, nursing, optometry, pharmacy, psychology and other care providers. In healthcare service to provide better performance to identify some diseases like blood pressure, skin cancer, diabetes etc. with the sensor devices those are connected with home server by using wireless sensor or wired network policies. The size of data sets being collected and analyzed in the industry for healthcare business intelligence is growing rapidly. In healthcare service case we have to concern about massive data services. To store and process huge amount of unstructured data by using traditional database system is so difficult. To alleviate this drawback we can use Hadoop architecture that has functionality to store and process huge amount of unstructured data. Hadoop Distributed file system (HDFS) can store and the Hadoop MapReduce framework process huge amount of unstructured data that enables easy development of scalable parallel applications. This paper's main motivation is to ensure better data transmission, data reliability and massive data processing with High Availability (HA) based network storage solution. So, in this paper we proposed MapReduce Agent (MRA) to process Big-Data and also proposed iSCSI Protocol adapted Network Attached Storage (NAS) system for healthcare service.*

Keywords: *HealthCare Service, Big-Data, Hadoop, Sensor Integration and iSCSI.*

I. Introduction

We have entered an era of data explosion, where many data sets being processed and analyzed are called "Big-Data" and it's getting even bigger in healthcare service system. Only the United States-U.S Healthcare service data alone reached 150 exabytes in 2011. Five exabytes (10^{18} gigabytes) of data would contain all the words ever spoken by human beings on earth. At this rate, big data for U.S. Healthcare will soon reach zettabyte (10^{21} gigabytes) scale and even yottabytes (10^{24} gigabytes) [11]. In the present days data's technology is changing rapidly and Big-Data may be used as an alternative solution. The types of Big-Data are structured data, unstructured data and semi-structured data. Structured data include electronic accounting and billings, actuarial data, some clinical data, laboratory instrument readings and data generated by the ongoing conversion of paper records to electronic health and medical records. Unstructured data include office medical records, handwritten nurse and doctor notes, hospital admission and discharge records, paper prescriptions, email, radiograph films, MRI, CT and other images. In healthcare service 80% data are unstructured. NoSQL, MongoDB, and TerraStore process structured big data. Big data not only requires a huge amount of storage, but also demands new data management on large distributed systems because conventional data management systems have difficulty to manage big data. Now a days to store big data of Cloud Healthcare based remote data center is used because of hi-tech machine with more CPU, RAM, disk space, etc. are not capable to store huge amount of healthcare data. To process big data in cloud environment hadoop MapReduce is playing an important role. Hadoop is a popular open-source map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to store and process extremely large data sets on commodity hardware. NAS is a high tech cheap network storage solution that supports only file I/O like CIFS, NFS. The Storage Area Network (SAN) is block I/O protocol but it is very costly to setup. So, if we use iSCSI protocol with NAS system then we may alleviate the traditional NAS behavior on data processing. Also our MRA module has developed on iSCSI block I/O protocol which we may use to modify Hadoop Architecture to ensure better Map and Reduce performance. Because HDFS is file I/O based data system but it works as a block based in the Hadoop architecture.

II. Structure of the paper

The rest of this paper is organized as follows. Overview of the healthcare service, iSCSI protocol and Hadoop MapReduce mechanism are described in section III. At section IV we described our research motivations. We describe our proposed model in brief in section V. We evaluated the performance and result in section VI. Finally, at section VII we provided the conclusion of this paper.

III. Background

In this section, besides the healthcare service we review iSCSI protocol, the MapReduce programming model and describe the salient features of Hadoop, a popular open-source implementation of MapReduce.

A. The Healthcare Data Explosion

The volume of global digital data is increasing exponentially, from 130 exabytes in 2005 to 7,910 exabytes in 2015. By 2020, it is expected to be 35 zettabytes. This is almost four piles of CDs reaching Mars from earth! Specifically for healthcare, in 2012 worldwide digital healthcare date was estimated to be equal to 500 petabytes and is expected to reach 25,000 petabytes in 2020. In other words worldwide healthcare data is expected to grow to 50 times the current total.

kilobyte (kB)	10^3
megabyte (MB)	10^6
gigabyte (GB)	10^9
terabyte (TB)	10^{12}
petabyte (PB)	10^{15}
exabyte (EB)	10^{18}
zettabyte (ZB)	10^{21}
yottabyte (YB)	10^{24}
Table 1: Orders of magnitude of data	

B. Big data in Health Care

There are many sources of big data within the health sector:

- Clinical data – up to 80 per cent of health data is unstructured as documents, images, clinical or transcribed notes;
- Publications – clinical research and medical reference material;
- Clinical references – text-based practice guidelines and health product (e.g., drug information) data;
- Genomic data – represents significant amounts of new gene sequencing data;
- Streamed data – home monitoring, telehealth, handheld and sensor-based wireless or smart devices are new data sources and types;
- Web and social networking data – consumer use of Internet – data from search engines and social networking sites; and
- Business, organizational and external data – administrative data such as billing and scheduling and other non-health data.

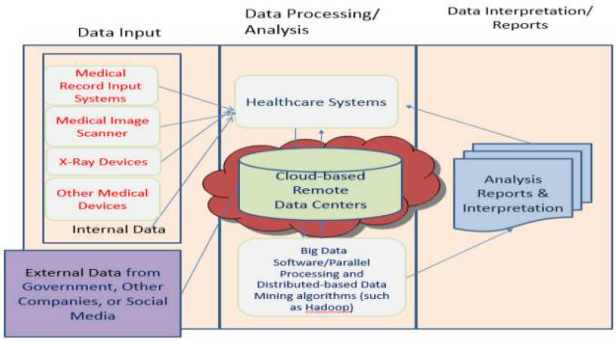

Fig. 1. Relationship between healthcare information systems and big data

C. iSCSI Protocol

iSCSI [2](Internet Small Computer System Interface) is a transport protocol that works on top of TCP. iSCSI transports SCSI packets over TCP/IP. iSCSI client-server model describes clients as iSCSI initiator and data transfer direction is defined with regard to the initiator. Outbound or outgoing transfers are transfer from initiator to the target.

C.1. iSCSI Parameters

The values of iSCSI parameters can be determined during login phase and full feature phase. Each iSCSI connection begins with login phase. The initiator and target can negotiate iSCSI parameters to improve performance during login phase. After that, iSCSI enters the full feature phase, during which iSCSI commands and data are exchanged over the established iSCSI connections. There are two classes of iSCSI parameters. One of them is related to iSCSI read operation while the other is related to iSCSI write operation.

C.1.1. iSCSI Read Operation Parameters

There are three parameters, which are related to an iSCSI read operation.

A. Number of sectors per command: Most SCSI disks define a sector size of 512 bytes, and require I/O operations to be in multiples of a sector. The iSCSI initiator is usually required to declare a limit on the number of sectors in a single SCSI I/O operation. It is the one of important parameters for iSCSI read operation, since the value of this parameter limit the request size of an iSCSI read command. The target can continuously send out the data requested from an iSCSI read command using Data-In PDUs. However, it is not directly related to iSCSI write operation. Though an iSCSI write command request the data transmission from initiator to target, R2T mechanism control the transmission size of the data associated with the command.

B. MaxRecvDataSegmentLength (MRDSL) of initiator: All iSCSI PDUs have one or more header segments and, optionally, a data segment. The Basic Header Segment (BHS) is the first segment in all of the iSCSI PDUs. The BHS is a fixed-length 48-byte header segment. Additional Header Segment (AHS), a Header-Digest, a DataSegment, and a Data-Digest may follow it. The initiator declares the maximum data segment length in an iSCSI PDU. Thus, the DataSegmentLength of a Data-In PDU must not exceed MaxRecvDataSegmentLength of the initiator.

C. Phase Collapse: Normally finishes a read command by sending a separate iSCSI response PDU containing the command's status. However, when the status is success, a non-negotiable option allows the target to use a phase collapse in which it sets a bit in the final Data-In PDU and omits the iSCSI response PDU.

C.1.2. iSCSI iSCSI Write Operation Parameters

There are two parameters, which are related to an iSCSI write operation

A. MaxBurstLength: The initiator and target negotiate maximum SCSI data payload in bytes in a solicited Data-Out iSCSI sequence. A sequence consists of one or more consecutive Data-Out PDUs that end with a Data-Out PDU with the F bit set to one. The end of a sequence of Data-Out PDUs requires the transmission of a R2T PDU by the target back to the initiator before the next data-out sequence can begin. Therefore, the value of the MaxBurstLength parameter limit the total amount of all data segments of all PDUs in a solicited data sequence requested by a R2T PDU.

B. MaxRecvDataSegmentLength (MRDSL) of target: The target declares the maximum data segment length in an iSCSI PDU. Thus, the DataSegmentLength of a Data-Out PDU must not exceed MaxRecvDataSegmentLength of the target.

D. Hadoop Architecture

Hadoop [4] is composed of Hadoop MapReduce, an implementation of MapReduce designed for large cluster, and the Hadoop Distributed File System (HDFS), a file system optimized for batch-oriented workloads such as MapReduce. In most Hadoop jobs, HDFS is used to store both the input to the map step and the output of the reduce step. Note that HDFS is not used to store intermediate results (e.g., the output of the map step): these are kept on each node's local file system.

A Hadoop installation consists of a single master node and many worker nodes. The master, called the Job-Tracker, is responsible for accepting jobs from clients, dividing those jobs into tasks, and assigning those tasks to be executed by worker nodes. Each worker runs a Task-Tracker process that manages the execution of the tasks currently assigned to that node. Each Task Tracker has a fixed number of slots for executing tasks (two maps and two reduces by default).

E. Map Task Execution

Each map task is assigned a portion of the input file called a split. By default, a split contains a single HDFS block (64 MB by default)[4], so the total number of file blocks determines the number of map tasks. The execution of a map task [17] is divided into two phases.

- The map phase reads the task's split from HDFS, parses it into records (key/value pairs), and applies the map function to each record.
- After the map function has been applied to each input record, the commit phase registers the final output with the Task Tracker, which then informs the JobTracker that the task has finished executing.

F. Reduce Task Execution

The execution of a reduce task [17] is divided into three phases.

- The shuffle phase fetches the reduce task's input data. Each reduce task is assigned a partition of the key range produced by the map step, so the reduce task must fetch the content of this partition from every map task's output.
- The sort phase groups records with the same key together.
- The reduce phase applies the user-defined reduce function to each key and corresponding list of values.

G. Pipelining Mechanism

In Google proposed MapReduce mechanism reduce task can not start to work until map task complete it's task. To solve this problem they introduce pipelining version [5] of Hadoop. They used naïve implementation to send data directly from map to reduce tasks. They modified Hadoop so that each reduce task contacts every map task upon initiation of the job and opens a TCP socket which will be used to send the output of the map function. They also developed the Hadoop online prototype (HOP) that can be used to support continuous queries: MapReduce jobs that run continuously. They also proposed a technique known as online aggregation which can provide initial estimates of results several orders of magnitude faster than the final results. Finally the pipelining can reduce job completion time by up to 25% in some scenarios.

H. MapReduce Agent (MRA) Mechanism

In pipelining mechanism there may be some drawbacks occurred in TCP connection and TCP congestion during data transmission. For that reason, TCP connection being disconnected and after that data can be retransmitted which takes long time. So MRA mechanism [17] that can send data without retransmission using iSCSI multi-connection [3] and also manage load balancing of data. Finally to process big data MRA mechanism provide better performance and reduce the time to job completion than pipelining mechanism and Google proposed mechanism.

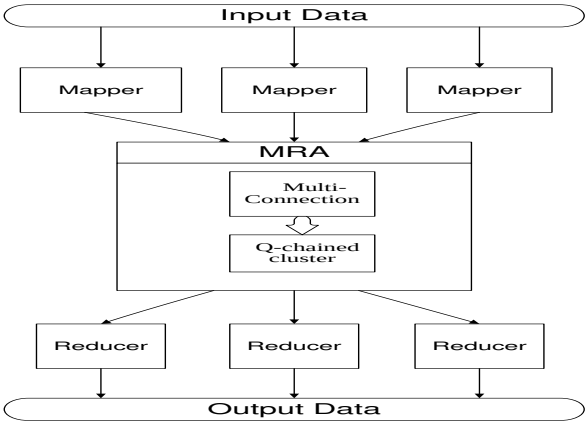

Fig. 2. Map Reduce Agent Architecture (MRA).

IV. Motivations

In a fixed network, a packet loss can in general be considered as an indication of overload and congestion situation. Though communication over high speed wired link better than wireless link but at present mass users are running on ubiquities track. So, communication over wireless link is often characterized by sporadic high bit-error rates, and intermittent connectivity due to handoffs. TCP reacts to packet losses as it would in the wired environment and it drops its transmission window size before re-transmitting packets, initiates congestion control or avoidance mechanisms [15] and resets its retransmission timer. These measures result in an unnecessary reduction in the link's bandwidth utilization, thereby causing a significant degradation in performance in the form of poor throughput and very high interactive delays [16].

Since iSCSI is used in wireless environments runs on a TCP network, it requires new strategies to evade the forceful decrease of transmission rate from TCP congestion control mechanism without changing existing TCP implementations and without recompiling or re-linking existing applications.

As we have discussed about the NAS drawback shortly into the Introduction section and also focused our proposed method's key point. To adapt iSCSI protocol into the NAS system we have some obstacles also. Those are related TCP congestion and iSCSI parameters on the Wireless Connections.

If we specially focus on wireless network environment, the iSCSI Data-In PDUs are passed to the TCP layer, since the iSCSI PDU is a SCSI transport protocol over TCP/IP. When the iSCSI Data-In PDU size is greater than the MSS (Maximum Segment Size) of TCP layer, the PDU will be further fragmented into smaller segments. If some segments of Data-In PDU are lost due to the high bit error rate in wireless networks, TCP layer would require re-transmitting the segments. At that time all the other segments of those parts of an iSCSI Data-In PDU must wait for being reassembled into an iSCSI Data-In PDU in TCP buffer. It decreases the system performance due to the reduction in the available capacity of TCP buffer. The sender can transmit data segments, which are allowed by the receiver using TCP flow control mechanism. In a wireless network with high bit error rate; if the size of iSCSI PDU is increased by the MRDSL (MaxRecvDataSegmentLength) parameter then many segments would have to wait due to the loss of some segments of iSCSI PDU in TCP buffer. The system performance could further decrease.

In the write operation, it produces the same results as the read operation. In addition, a wireless link generally becomes a bottleneck portion in an end-to-end TCP connection because of its narrow bandwidth, as compared to wired links. Thus a TCP sender's congestion controls are apt to be caused by wireless link congestions.

In healthcare service 80% data are unstructured and those data are sensitive. To process those unstructured data by using traditional database system is so difficult. So we need a proper mechanism to process those data. To process those data properly we used MRA mechanism in Hadoop architecture that can dramatically process huge amount of data.

V. Proposed Model

In healthcare center they have to process huge amount of data to store and process for providing better service. They used different devices to collect information or health data and send to the server for processing data by using wireless or wired network. So we need a mechanism to process and store those data because data is increasing day by day. The following mechanisms those can process huge amount of data and reduce the storage and network connection problem.

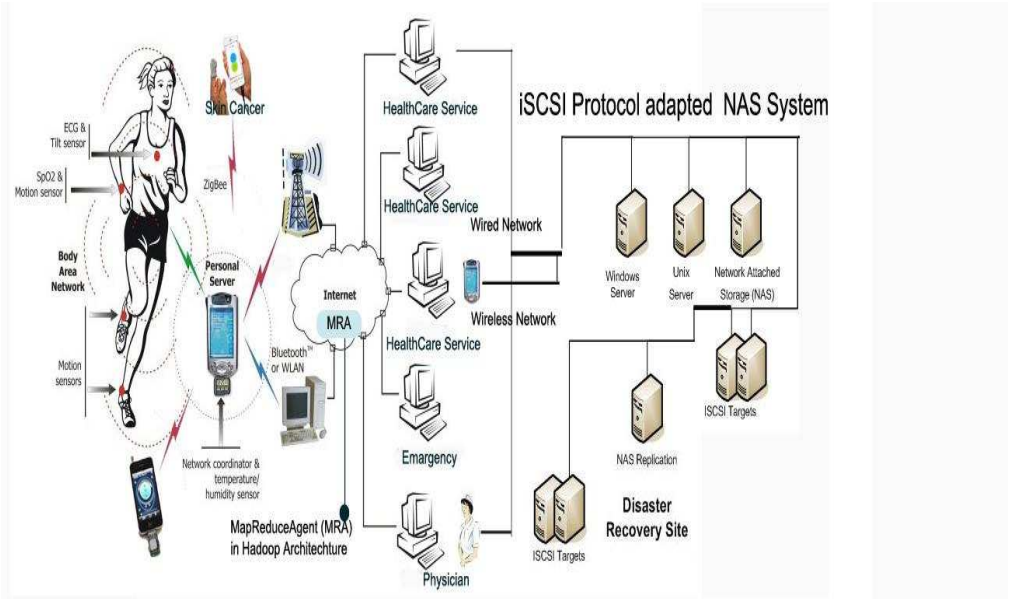

Fig.3. Our proposed healthcare system architectural overview

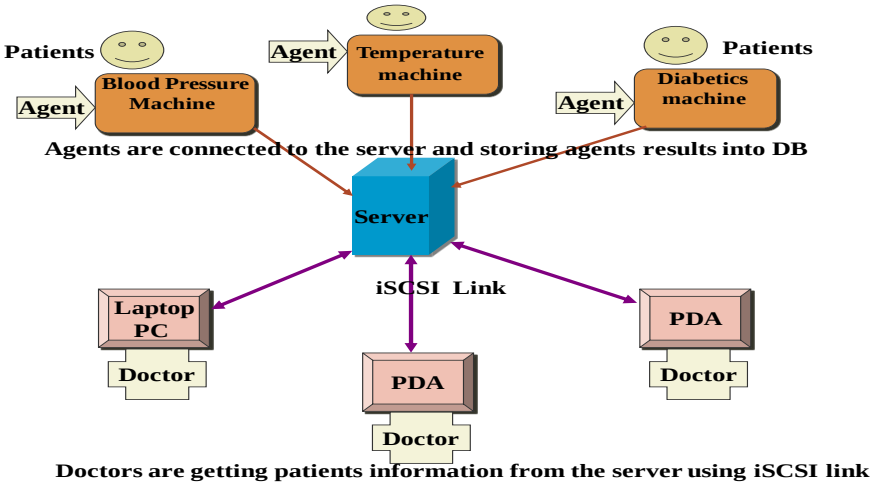

Fig.4. Overall Structure of healthcare system

A. Big data processing mechanism MRA:

Modified Hadoop MapReduce architecture that is MapReduce Agent (MRA) mechanism can process big data properly. Because of MRA mechanism have some properties that can create iSCSI muticonnection to send data map to reduce task directly and also manage data overload by using Q-chain load balancer method.

A.1. Multi-connection and Error Recovery Method of iSCSI:

In order to alleviate the degradation of iSCSI-based remote transfer service caused by TCP congestion control, we propose MRA Multi-Connection and Error Recovery method for one session which uses multiple connections for each session. As mentioned in [8], in a single TCP network connection when congestion occurs by a timeout or the reception of duplicate ACKs (Acknowledgement) then one half of the current window size is saved in sstresh (slow start window). Additionally, if the congestion is indicated by a timeout, cwnd (congestion window) is set to one segment. This may cause a significant degradation in online MapReduce performance. On the other hand in Multi-Connection case, if TCP congestion occurs within connection, the takeover mechanism selects another TCP connection. The general overview of the proposed Multi-Connection and Error Recovery based on iSCSI protocol scheme which has been designed for iSCSI based transfer system. When the mapper (worker) is in active mode or connected mode for reduce job that time session is started. This session is indicated to be a collection of multiple TCP connection. If packet losses occur due to bad channel characteristics in any connection, our proposed scheme will randomly pick out Q-Chained Cluster.

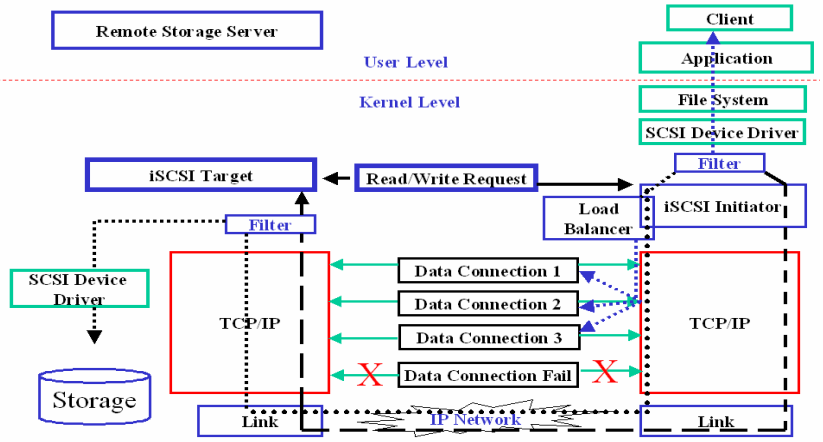

Fig.5. Overview of Multi-connection and Error Recovery Method of iSCSI

A.2. Q-Chained Cluster Load Balancer

Q-chained cluster is able to balance the workload fully among data connections in the event of packet losses due to bad channel characteristics. When congestion occurs in a data connection, this module can do a better job of balancing the workload which is originated by congestion connection, will be distributed among N-1 connections instead of a single data connection. However, when congestion occurs in a specific data connection, balancing the workload among the remaining connections can become difficult, as one connection must pick up the workload of the component where it takes place. In particular, unless the data placement scheme used allows the workload, which is originated by congestion connection to be distributed among the remaining operational connections.

Figure 6 illustrates how the workload is balanced in the event of congestion occurrence in a data connection (data connection 1 in this example) with Q-chained cluster. For example, with the congestion occurrence of data connection 1, primary data Q1 is no longer transmitted in congestion connection for the TCP input rate to be throttled and thus its recovery data q1 of data connection 1 is passed to data connection 2 for conveying storage data. However, instead of requiring data connection 2 to process all data both Q2 and q1, Q-chained cluster offloads 4/5ths of the transmission of Q2 by redirecting them to q2 in data connection 3. In turn, 3/5ths of the transmission of Q3 in data connection 3 are sent to q3. This dynamic reassignment of the workload results in an increase of 1/5th in the workload of each remaining data connection.

Data connection	0	1	2	3	4	5
Primary Data	Q_0	F	$1/5Q_2$	$2/5Q_3$	$3/5Q_4$	$4/5Q_5$
Recovery Data	$1/5q_5$	F	q_1	$4/5q_2$	$3/5q_3$	$2/5q_2$

Fig. 6. Q-Chained Load Balancer.

B. Architecture of iSCSI Module for Block I/O Service in NAS:

Figure 7 illustrates the different OS platforms data and file components those are of interest for data transfer through the I/O path. The NAS metadata server will provide the I/O interface to the user-space world.

The I/O system will call in metadata server use MUVFS (or generic file system) to perform their tasks, while individual file system, such as EXT3 or NFS and HDFS Big-data plugs their code into the MUVFS handlers.

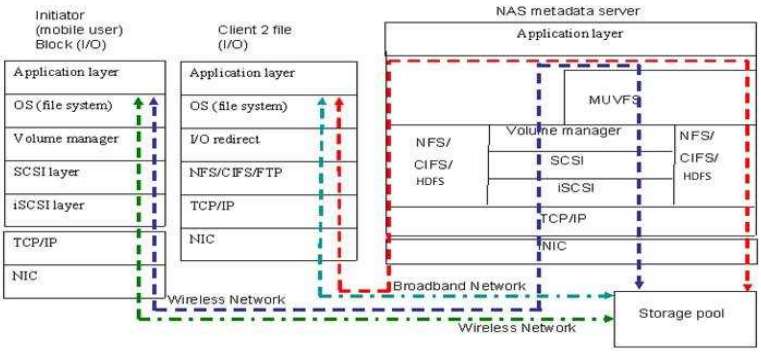

Fig. 7. Software Architecture of NAS for Block I/O and File I/O Service.

When NAS offer the block I/O and HDFS(Big-Data) services, it invokes the iSCSI module, the asker is Initiator. Concrete data read/write process is as follows: (1) The block I/O commands (SCSI commands) sent by the users in Initiator are encapsulated into the IP data packets with the iSCSI device driver and the TCP/IP protocol stack, then transferred via network; (2) When the encapsulated packets arrive at the NAS metadata server, they are restored to the previous SCSI commands in an unpacked process, then passed to the MUVFS layer. Having been processed by the MUVFS, the previous I/O commands are packed up once again and sent to an appointed NAS via an inner network; (3) required data blocks are packed into the iSCSI protocol data units by the NAS and returned to the Initiator via a network(wired and wireless). The way, which the users appliance communicates with the NAS via a high-speed IP channel, is similar to that the users appliance communicates with the metadata server. When NAS offers the file I/O service, the data read/write process is almost the same as that in a usual NAS.

B.1. Multi User Virtualization File System (MUVFS)

To support Unix Client, Windows Client, iSCSI mobile appliance and Big-Data file system (HDFS), we develop MUVFS system. The Multi User Virtualized File System (MUVFS) is the most important core modular of our proposed network NAS storage model, which enables the centralized management of many NAS nodes and provides a unified file system view for clients. Figure 8 shows the structure of MUVFS. It consists of Unfsd, UiSCSId, Samba, the management partition, an iNAS configuration table, and a virtual partition (logic volume) map table. Unfsd is a wrapper of the global NFS daemon and provides file services for UNIX/Linux clients. Samba is used for windows clients. UiSCSId is a wrapper of the global iSCSI daemon and provides block I/O services for iSCSI systems like mobile, PDA, Laptop.

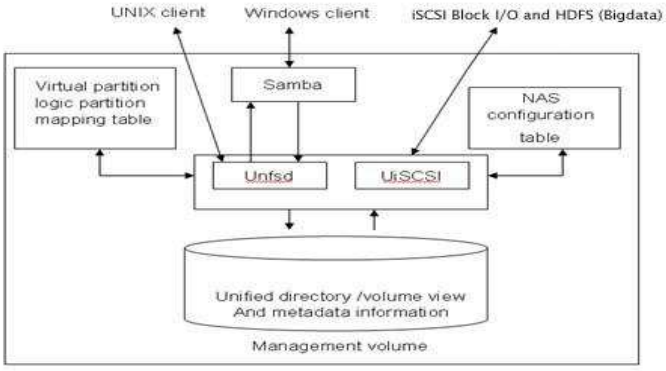

Fig.8. Structure of Multi user Virtualization File System MUVFS

The NAS configuration table contains the system information such as hostnames, independent IP of NAS members, export points, and so on. The virtual partition (logic volume) map table specified the data partition which stores file entities, etc.

VI. Performance evaluation

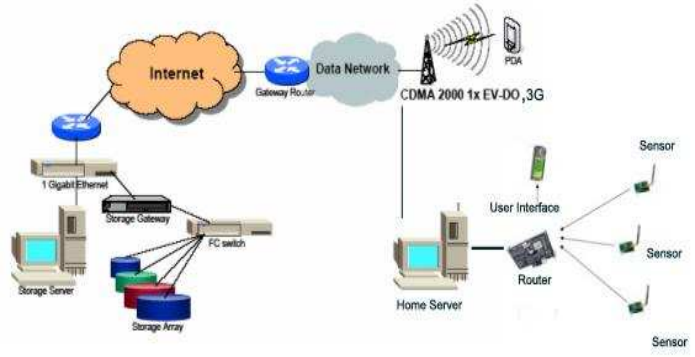

Fig.9. Experimental Setup

A. Experimental environment setup for Network Storage.

Our experiment setup consists of a Smart phone and Storage server connected on to Internet with 3G CDMA 2000 1xEV-D0 network. The initiator module embedded in smart phone based on Android can transmit user interfacing data those are generating from sensor based health devices. iSCSI command and Big-data will be stored in the target NAS Storage server for file and block I/O. We set the experimental environment as Table 2.

	Storage Server or Target	**Mobile Device or Initiator**
OS	Linux Red hat 5.0 (kernel version 3.14.4)	Android Jelly Bean 4.3
CPU	Intel Core i5-2400 CPU	2.2 GHz QualComm SnapDragon 800
Memory (RAM)	4GB	1GB
NIC	1Gbps LAN	3G CDMA 2000
	802.15.4 zigbee sensor device	

Table 2. Experimental Environment

Experimental methodology for iscsi based nas system

Let us make one think clear before the explanation of experimental methodology for this section. In this section we have focused on iSCSI parameter analysis those defined by its standard to investigate those effects. We performed two kinds of experiments to find out the best performance parameters setting to adapt mobile appliances with NAS system for large amount of storage facilities via wireless network. We picked up MRDSL parameter to optimize iSCSI protocol for large and small size of data transmission between initiator (Healthcare service) to the target (NAS) storage.

We consider throughput as performance metrics, which is the total number of application level bytes carried over an iSCSI connection divided by the total elapsed time taken by the application, as expressed in Bytes per millisecond (B/ms). We used the system timer of Smart Phone, which is based on Android Operating System in order to measure throughput. Total elapsed time was measured as the time interval from the initial time when the experiment program started generating the first byte of data to the time when the last byte of data was confirmed to have been sent (received). The elapsed time therefore includes all data transfers and all read and write commands as well as responses at all levels of the protocol stack. We perform two kinds of experiments. In the first experiment, we generate an I/O stream of 5 megabytes, and then measure throughput. Our request for a 5 megabytes I/O operation is passed through the file system and the SCSI subsystem to the iSCSI initiator as a number of SCSI commands. Then the initiator sends the commands and data to the target device. At that time, we adapt various settings of the iSCSI parameters to investigate the best parameter values for performance in wireless network. Each data point plotted in every graph of this paper was calculated as the average of 5 runs with identical parameter settings. We also perform the same experiment again with a different size of I/O stream, which is a 100 megabytes I/O stream. We perform the second experiment in order to examine the effect of the characteristics of the high bit error rate within the commercial CDMA network and the variable bandwidth of the network. Our experiment program generates read I/O bursts continuously for 15 hours, and then measures throughput every 1000 seconds. The second experiment shows that the increase of the MRDSL parameter value cannot always bring performance improvement to iSCSI-based NAS System in unstable wireless network with high bit error rate and variable bandwidth

Experimental Result

Figure 10 illustrates that the increase of the Number of sectors per command causes the performance improvement in the wireless network. It is very similar to the experiment results in a wired network. However, there are the performance falling at the Number of sectors per command value of 2048 with MRDSL of 1Kbytes and 4Kbytes. At the MRDSL values of 1Kbytes and 4Kbytes, there are 1% and 15% de-creases respectively in throughput as Number of sector per command increases from 1024 to 2048. In the wireless network, the results related to the values of MRDSL from our experiment are different from those in a stable wired network. The throughputs are better at the MRDSL values of 1Kbytes, 2Kbytes and 4Kbytes than the values of 512bytes and 8Kbytes. At the Number of sectors per command value of 2048, the throughput of 2Kbytes MRDSL is increased by 48% and 52% respectively compared with those of 512bytes and 8Kbytes MRDSL. However, when MRDSL value is among 1Kbytes, 2Kbytes and 4Kbytes, it is difficult to say which one is the best value, for different MRDSL values may outperform others at different values of Number of sectors per command. For example, MRDSL of 2Kbytes can bring better throughputs when Number of sectors per command is between 256 and 512, while MRDSL of 1Kbytes is better when Number of sectors per command is around 1024 as we can see from figure 9. Therefore, it is not always correct in wireless network to increase the value of MRDSL to get performance improvement. Correspondingly, the default value 8Kbytes of MRDSL in standard is also not suitable for a wireless network. The reason of performance dropping at the MRDSL value of 512bytes is due to extra PDUs overhead. When MRDSL is 8Kbytes, however, the reason of performance falling is due to the increase of data segments which are transmitted continuously by TCP until detecting congestion even though congestion occurred in the wireless network with narrow bandwidth. Therefore, we suggest that you should use the parameters settings of the MRDSL of 1Kbytes, 2Kbytes and 4Kbytes with the Number of sectors per command values of 1024 or 2048 when per-forming a small file read operation in CDMA network

In write operation, the Number of sectors per command is kept constant at 1024(512Kbytes) and the MaxBurstLength (MBL) varies from 512Bytes to 256Kbytes. We use the fixed Number of sectors per command, because the value of the parameter MaxBurstLength limits the total amount of all data segments of all PDUs which were requested by R2T. Figure 11 shows that the throughput is increased as MBL increases from 512Bytes to 128Kbtyes in a wireless network, after that there is a slight decrease in throughput when MBL is around 256Kbytes. The same kinds of decrease in throughput happen in read operations too, which is caused by the narrow bandwidth of CDMA 3G networks. The number of iSCSI PDUs which are continuously passed to the TCP layer can be increased by increasing either the value of MaxBurstLength for write operation or Number of sectors per command for read operation, then the sender will transmit more segments until recognizing congestion in wireless network, and thus causes the performance falling of iSCSI based NAS for mobile appliances. In a write operation, the result is more obvious because the CDMA 2000 network has the narrower bandwidth for upload than that for download. Therefore it is not always true that the increase of MBL value would cause the performance improvement. The MBL value of 256Kbytes in standard is not suitable too. When the MRDSL value is 2Kbytes or 4Kbytes, the throughputs are better than that is 512bytes or 1Kbytes or 8Kbytes. At the MBL of 128Kbytes, the throughput of MRDSL of 2Kbytes is 26% and 31% increases respectively than those of MRDSL of 512bytes and 8Kbytes. However, when MRDSL is at a value of 2Kbytes or 4Kbytes, it is hard to say which one is the best value. From the figure 10 we can know that either of them may achieve better performance within a certain value range of Number of sectors per command. From these we can also see the standard value for wired network is not suitable here for a wireless network. Therefore, we suggest that you use the parameters settings of the MRDSL of 2Kbytes and 4Kbytes with the MBL of 128Kbytes when performing a small file write operation in CDMA network.

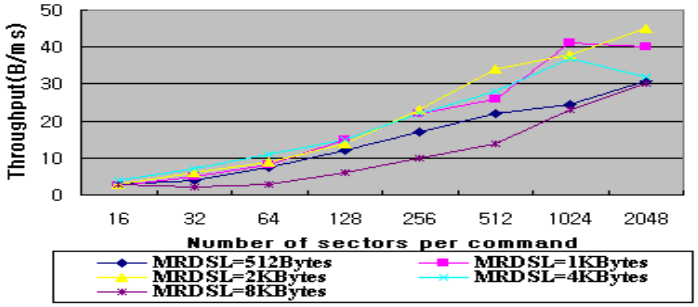

Fig.10. Effect of Number of sectors per command on throughput for 5 Mbytes read operation in CDMA network

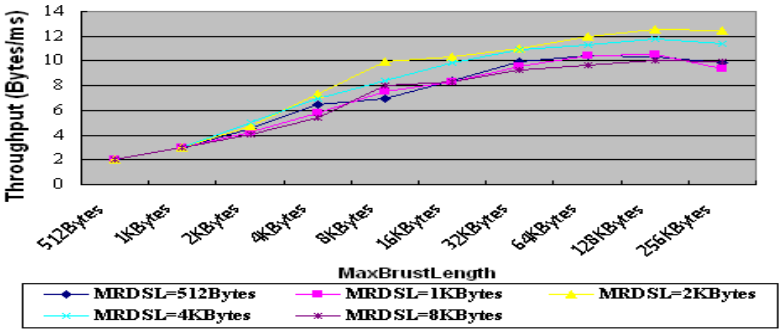

Fig.11. Effect of MaximumBrustLength (MBL) for 5 Mbytes write operation in CDMA network

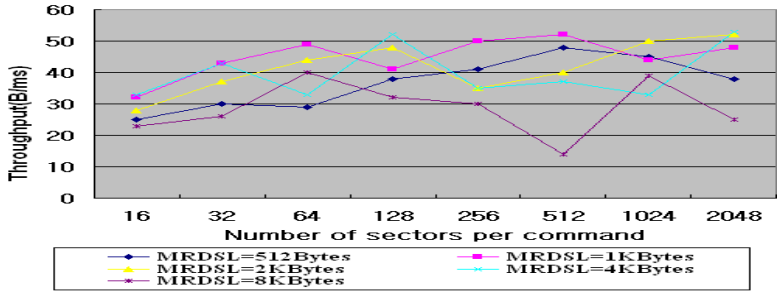

Fig.12. Effect of Number of sectors per command on throughput for 100 Mbytes read operation in CDMA network

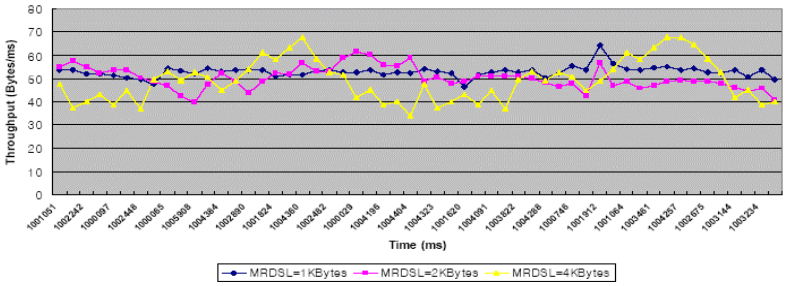

Fig.13. Effect of MRDSL on throughput about 15 hours in wireless network

Figure 12 shows the results from our experiment of 100 megabytes read operation. When MRDSL is at a value of 1Kbyte, the throughput is better than that at the value of 512bytes or 8Kbytes. When the MRDSL value is 1Kbytes, 2Kbytes and 4Kbytes, however, it is difficult to say which one is the best value. In the case of large size of I/O burst, such as 100 megabytes read operation, the best value for performance is also different from the standard value. When the MRDSL value is 2Kbytes, 4Kbytes and 8Kbytes, the performance is sharply dropped or increased with the increase of Number of sectors per command. At 8Kbytes of MRDSL value, there is a drastic decrease by 64% in throughput as Number of sectors per command increases from 64 to 512, after that there is an increase in throughput suddenly when the Number of sectors per command is around 1024. This experiment has different results from 5 megabytes I/O operation because the characteristics of a high bit error rate within the wireless channel, and a narrow and variable bandwidth of the wireless channels affect the large file read operation which was performed for a long time. Therefore, we perform the second experiment in order to examine the effect of the characteristics of CDMA network more clearly. In the experiment for continuous read I/O bursts, we measure throughput every 1000 seconds in order to observe the degree of performance increasing or decreasing with respect to 1Kbytes, 2Kbytes and 4Kbytes MRDSL parameter values respectively, and then the Number of sectors per command is kept constant at 1024(512Kbytes). Among the 1Kbytes, 2Kbytes, and 4Kbytes MRDSL, there are the fewest changes in throughput of 1Kbytes of MRDSL in figure 13. From the results, we found out that the smaller size of iSCSI PDU could be less affected by the characteristics of a high bit error rate within the wireless channel, and a narrow and variable bandwidth of the wireless channels as we explained in section IV. Therefore we suggest that you should set the MRDSL parameter value at 1Kbytes when performing a large file I/O operation while moving in CDMA network.

B. Experimental environment setup for MRA

As per as [5] we also evaluate the effectiveness of online aggregation, we performed two experiments on Amazon EC2 using different data sets and query workloads. In their first experiment [5], they wrote a "Top-K" query using two MapReduce jobs: the first job counts the frequency of each word and the second job selects the K most frequent words. We ran this workload on 5.5GB of Wikipedia article text stored in HDFS, using a 128MB block size. We used a 60-node EC2 cluster; each node was a "high-CPU medium" EC2 instance with 1.7GB of RAM and 2 virtual cores. A virtual core is the equivalent of a 2007-era 2.5Ghz Intel Xeon processor. A single EC2 node executed the Hadoop Job- Tracker and the HDFS NameNode, while the remaining nodes served as slaves for running the TaskTrackers and HDFS DataNodes.

We took performance result using both large (512MB) and small (32MB) HDFS block sizes using a single workload (a word count job over randomly-generated text). Since the words were generated using a uniform distribution, map-side combiners were ineffective for this workload. We performed all experiments using relatively small clusters of Amazon EC2 nodes. We also did not consider performance in an environment where multiple concurrent jobs are executing simultaneously.

Experimental Results For Iscsi Multi-Connection And Q-Chained Load Balancer

Our proposed scheme throughputs in different RTTs are measured for each number of connections in Figure 14. We see the slowness of the rising rate of throughput between 8 connections and 9 connections. This shows that reconstructing the data in turn influences throughputs and the packet drop rates are increased when the number of TCP connections is 9 as the maximum use of concurrent connections between initiator and target.

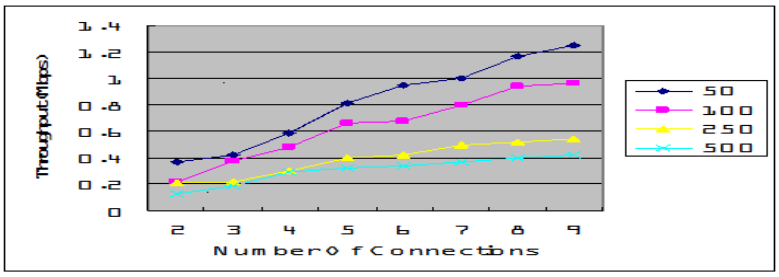

Fig.14. Throughput of Multi-Connection iSCSI System. Y axis is containing Throughput easurement With Mbps & X axis is for number of connections. 50,100,250 and 500 RTT are measured by ms.

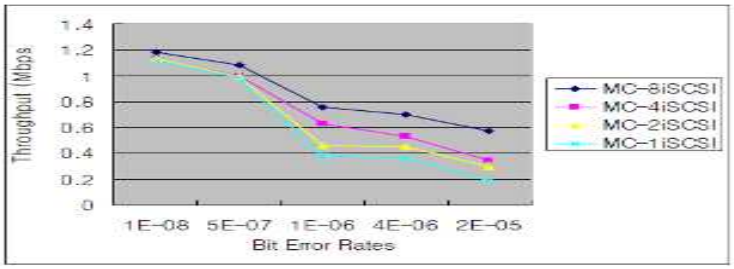

Fig. 15. Throughput of Multi-Connection iSCSI vs iSCSI At different error rates. Y axis is Throughput & X axis is for Bit error rate.

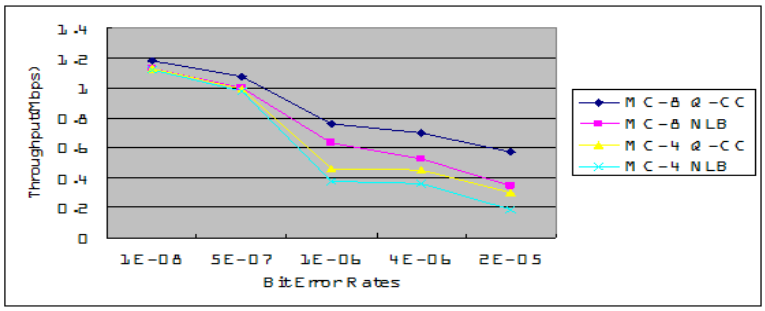

Fig.16. Q-Chained Cluster Load Balancer vs No Load Balancer. MC: Multi Connection, Q-CC: Q-Chained Cluster NLB: No Load Balancer.

Therefore, 8 is the maximum optimal number of connections from a performance point of view. Multi-Connection iSCSI mechanism also works effectively because the data transfer throughputs increase linearly when the round trip time is larger than 250ms.

In Figure 15, the performance comparison of Multi-Connection iSCSI and iSCSI at different bit-error rates is shown. We see that for bit-error rates of over 5.0×10-7the Multi-Connection iSCSI (2 connections) performs significantly better than the iSCSI (1 connection), achieving a throughput improvement about 24 % in SCSI read. Moreover, as bit-error rates go up, the figure shows that the rising rate of throughput is getting higher at 33% in 1.0×10-6, 39.3% in 3.9×10-6and 44% in 1.5×10-5. Actually, Multi-Connection iSCSI can avoid the forceful reduction of transmission rate efficiently from TCP congestion control using another TCP connection opened during a service session, while iSCSI does not make any progress. Under statuses of low bit error rates (< 5.0×10-7), we see little difference between Multi-Connection iSCSI and iSCSI. At such low bit errors iSCSI is quite robust at handling these.

In Figure 16, Multi-Connection iSCSI(8 connections) with Q-Chained cluster shows the better average performance about 11.5%. It can distribute the workload among all remaining connections when packet losses occur in any connection. To recall an example given earlier, with M = 6, when congestion occurs in a specific connection, the workload of each connection increases by only 1/5. However, if Multi-Connection iSCSI (proposed Scheme) establishes a performance baseline without load balancing, any connection, which is randomly selected from takeover mechanism, is overwhelmed.

Experimental results of mra

In the Hadoop map reduce architecture [4, 5]; their first task is to generate output which is done by map task consume the output by reduce task. The whole thing makes the process lengthy because reduce task have to wait for the output of the map task. In pipelining mechanism [5], they send output of map task immediately after generation of per output to the reduce task so it takes less time than Hadoop MapReduce[4]. During the transmission (TCP) if any problem occurred then they retransmit again which took more time and drastically reduce the performance of MapReduce mechanism. On the other hand our proposed mechanism (MRA) recovers the drawback by using multi-connection and Q-chained load balancer method. In these circumstances MRA may prove its better time of completion.

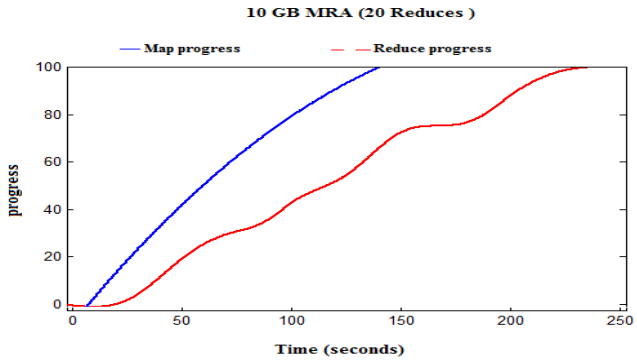

Fig. 17. CDF of map and reduce task completion times for a 10GB wordcount job using 20 map tasks and 20 reduce tasks (512MB block size). The total job runtimes were 240 seconds for MRA.

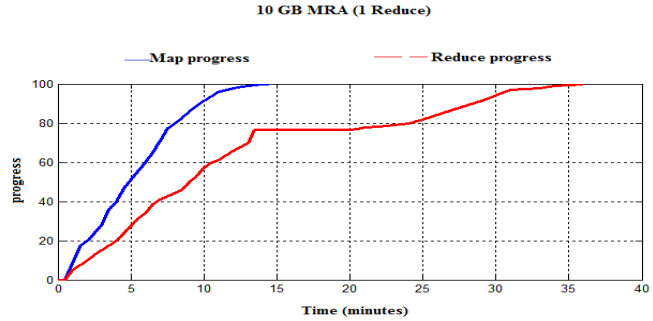

Fig. 18. CDF of map and reduce task completion times for a 10GB wordcount job using 20 map tasks and 1 reduce tasks (512MB block size). The total job runtimes were 36 minutes for MRA.

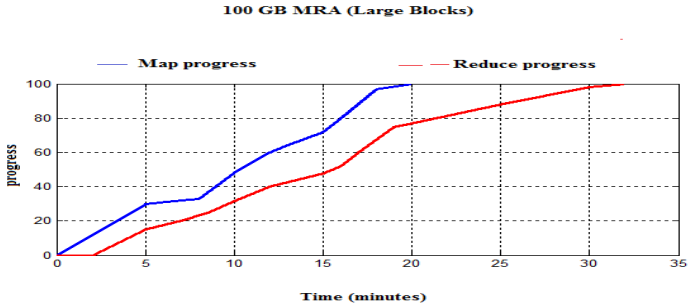

Fig.19. CDF of map and reduce task completion times for a 100GB wordcount job using 240 map tasks and 60 reduce tasks (512MB block size). The total job runtimes were 32 minutes for MRA.

VII. Conclusion

In this paper, we described our proposed healthcare service method which can be a suitable choice to handling large scale data like Big-Data. The types of Big-Data are structured data, unstructured data and semi-structured data. In the field of healthcare service most of the data are unstructured data. Also the size of data is increasing rapidly. That's why Hadoop MapReduce mechanism is the good solution to handle Big-Data. Though our paper sketched the design of healthcare service, but we strongly focused on Big-Data's reliable transmission and data balancing mechanism. To fulfill our goal and for good designed healthcare service, we adapted iSCSI protocol with Hadoop MapReduce mechanism and proposed it as MRA (MapReduce Agent). With good performance result, we also described how do we adapt and why do we use iSCSI Protocol with Hadoop? Our paper demonstrated that MapReduce can be more useful if we use MRA and also demonstrated Hadoop implementation in healthcare service.

In our paper, the other proposed part was also iSCSI protocol adaptation with traditional NAS massive network storage systems. In this case the motivated focus object was Block I/O. We analyzed iSCSI protocol supports block I/O and HDFS follows Block I/O procedures. So our iSCSI multi-connection with load balancing procedures may good solution to work with Hadoop MapReduce architecture and also ensures good data transmission. On the other hand, NAS is cheaper than SAN (Storage Area Network) but SAN gives well performance because of it supports block I/O instead of file I/O like NAS. This paper also solves this issue by adapting iSCSI protocol with NAS system.

In this paper, we performed two kinds of experiments to find out the best performance parameters setting to adapt mobile appliances with NAS system for large amount of storage facilities via wireless network. So, We are suggesting to use parameters settings of the MRDSL of 1Kbytes, 2Kbytes and 4Kbytes with the Number of sectors per command values of 1024 or 2048 to perform a small file read operation in CDMA network and for performing a small file write operation in CDMA network the parameters settings of the MRDSL of 2Kbytes and 4Kbytes (MBL of 128Kbytes). For large files operation, we found that the smaller size of iSCSI PDU could be less affected by the characteristics of a high bit error rate within the wireless channel, and a narrow, variable bandwidth of the wireless channels.

Acknowledgements

- This iSCSI research (Grants No.2009-0075576) was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Ministry of Education, Science and Technology 2013.
- This iSCSI research work (Grants No. 47504) was supported by Business for Cooperative R&D between industry, Academy and Research Institute funded Korea small and Medium Business Administration 2011.
- Research Lab: Gluesys Lab, Anyang University, Korea and East West University, Bangladesh.

References

[1] Le DEAN, J., AND GHEMAWAT, S. MapReduce: Simplified dataprocessing on large clusters. In OSDI (2004).

[2] SAM-3 Information Technology – SCSI Architecture Model 3, Working Draft, T10 Project 1561-D, Revision7, 2003.

[3] S.M.Allayear, Sung Soon Park: iSCSI Multi-Connection and Error Recovery Method for Remote Storage System in Mobile Appliance. The 2006 International Conference on Computational and It's Applications (ICCSA2006), Glasgow- Scotland. Springer-Verlag Berlin Heidelberg 2006, (SCI Indexed) LNCS 3981, pp.641-650.

[4] Hadoop, http://hadoop.apache.org/mapreduce/

[5] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein UC Berkeley: MapReduce Online. Khaled Elmeleegy, Russell Sears(Yahoo! Research)

[6] S.M Allayear, S.S Park and Jaechun No: iSCSI Protocol Adaptation with 2-way TCP Hand Shake Mechanism for an Embedded Multi-Agent Based Health Care Service. Proceedings of the 10th WSEAS international conference on Mathematical methods, computational techniques and intelligent systems, Corfu, Greece , 2008.

[7] S.M Allayear, S.S Park: iSCSI Protocol Adaptation With NAS System Via Wireless Environment. International Conference On Consumer Electronics (ICCE), Las Vegus, USA. 2008.
[8] R.Caceres and L. Iftode: Improving the Performance of Reliable Transport Protocols inMobile Computing Environments, IEEE JSAC,
[9] RFC 3270 "http://www.ietf.org./rfc/rfc3720.txt.
[10] Kuo Lane Chen, Huei Lee:The Impact of Big Data on the Healthcare Information Systems
[11] Institute for Health Technology Transformation: Transforming Health Care Through Big Data
[12] Bonnie Feldman, Ellen M. Martin, Tobi Skotnes: Big Data in Healthcare Hype and Hope
[13] Silvia Piai, Massimiliano Claps: Bigger Data for Better Healthcare
[14] Canada Inforoute: Big Data Analytics in Health,white paper
[15] V.Jacobson: Congestion avoidance and control, In SIGCOMM 88, August (1988).
[16] Mobile Computing Environments, IEEE JSAC, June (1995).
[17] "Introducing iSCSI Protocol on Online Based MapReduce Mechanism***",** Shaikh Muhammad Allayear, Md. Salahuddin, Fayshal Ahmed, and Sung Soon Park, ICCSA 2014, Part V, LNCS 8583, pp. 691–706, 2014. © Springer International Publishing Switzerland 2014