A SECURE FRAMEWORK FOR HADOOP COMMUNICATION

Gobinda Paul,S. M. Monzurur Rahman

Abstract—This paper proposes a framework, namely, HSMF (Hadoop Secure Messaging Framework). Now a day, it requires efficiency in processing billion bytes of data (binary) particularly through business processes. It also has become expensive to ensure reliability in each and every application that processes large amount of datasets (complex or structured). Hadoop platform was designed to address these issues. It was designed mainly for enterprise solution. Our goal is to establish secure communication between hadoop and smart devices or applications, so that applications of smart devices can send queries to hadoop and display respective results. HSMF is based on XML (Extensible Markup Language) message over TCP/IP which gives hadoop more flexibility in order to communicate not only with java based applications but also with applications written in other languages such as C/C++, C#, Python, Dot Net framework etc. Since XML is generated dynamically, client application may choose DML (Data Manipulation Language) of hadoop using HSMF. Client application does not require knowing the design of dataset, it only performs query on particular dataset based on user requirement. During communication via TCP/IP, every XML message header is checked and matched by a messaging header checking mechanism. This is why HSMF is secure. Our results show a successful communication between hadoop and smart devices or applications where smart devices and applications send query to hadoop and display the result received from hadoop.

Index Terms— HSMF, Hive, Hadoop, HDFS, Map-Reduce, Bigdata.

1 Introduction

Hadoop [3] is an open source software framework for storage and large scale processing of data-sets in parallel in a distributed environment (clusters) [2]. Hadoop is not only a software platform but also offers distributed computing and computational capabilities at relatively low cost. It has feature like scalability to meet the anticipated exponential increase in data generated by mobile technology, social media, internet, and other emerging digital technologies. It contains two key components: data storage mechanism with the help of Hadoop Distributed File System [4] and high-performance large-scale data processing using MapReduce framework [1]. MapReduce is the heart of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions.

A new network-levitated xml [7] based communication mechanism HSMF (Hadoop Secure Messaging Framework) improves Hadoop framework, provides additional feature in the original framework. HSMF is implemented based on TCP/IP protocol with security checking mechanism [9].

In this paper, we will explain the details of design and implementation of a TCP/IP implementation of HSMF. Two components, 'server' and 'client' are introduced to realize the TCP/IP connection that can fetch data from end user and send them to server within the new network-levitated xml [7] messaging mechanism. XML is generated dynamically on HSMF, user can select from where to take data for analysis for example: apache hive [10], apache pig [11] etc. Multithreading technologies are used to manage memory pool, send/receive, and merge data segments.

- *Gobinda Paul is currently pursuing masters degree program in Computer Science & Engineering in United International University, Dhaka, Bangladesh. ID-012122002. E-mail: gobinda@live.com*
- *Prof. Dr. S. M. Monzurur Rahman, Professor, School of Science & Engineering in United International University, Dhaka, Bangladesh. E-mail: mrahman99@yahoo.com*

1.1 Background

Efficiency in processing billion bytes of data (binary) particularly through business processes is crucial in today world when we are in one global village. It also has become expensive to ensure reliability in each and every application that processes large amount of datasets (complex or structured). Hadoop platform was designed to address these issues. It was designed mainly for enterprise solution. One of many other challenges at present is how the small smart devices will communicate with hadoop for big data processing and analysis? For example there is a cricket match going on today. I want to predict based on statistical data and analysis with my smart phone that which team is going to win. I have twenty years of big data regarding weather, players' statistical records on their performance, pitch condition and so on. Now I want this prediction to be done using hadoop where the processing time for hive, pig, etc. are different. The obvious questions are: Is it possible to choose the DML dynamically on the basis of user requirements so that we can be able to process faster and how much security I have in order to have the desired result through all the processing. These are the questions that inspired me to start this paper work and eventually reach to solution.

1.2 Methodology

In this paper, HSMF framework based on TCP/IP socket system through designing a messaging models using hadoop java library. The framework has two major parts. One is 'server' and another is 'client'. A server application was written on hadoop name-node server using hadoop java library with the help of xml and socket library which takes communication request from authorized application based on server side of HSMF model. As a result, if any particular device or application wants to communicate with hadoop dataset, it needs to send query as a request message (format based on HSMF) to server application on a particular port over internet or intranet. On the other hand, server application running on name-node server, accepts request from preauthorized client application (it mainly takes query request from clients) and opens a job request for hadoop and stores the result on an application list. Though, client application sends request to get update notification to server application, server application searches on that application list and reply to client with current status or result.

2 Overview of Proposed HSMF

Hadoop was born out of a need to process mass data. Our proposed framework is not modifying hadoop but to give hadoop more reliability and flexibilities in communication with other applications. According to our proposed framework, we need to write a server socket based application with support of xml, hadoop, hive, pig library or next upcoming library.

Server application always listens on a particular port in order to receive request message from client. Once server application receives xml query request, first tokenize xml message and pickup query message with DML, send that to hadoop to get the result. Once query process completed, query result is stored into a java list or mysql or oracle database or in a flat file.

Client application sends query status message to know whether hadoop process is completed or not. If the requested query is processed successfully by the server, then server application makes an xml message with result and gives reply when client application sends query status message. Else if the query is under process or failed, server application generates xml with query under process or query failure message to client application.

Our HSMF Framework stands on four core messaging patterns. Those are a) Query Request, b) Query Reply, c) Query Status and d) Query Status Reply. HSMF two message-Query Request and Query Status originate from client side and Query Status Reply and Query Reply generate at server side shown on Figure 2.1. Details description of core framework message is given in HSMF Communication section.

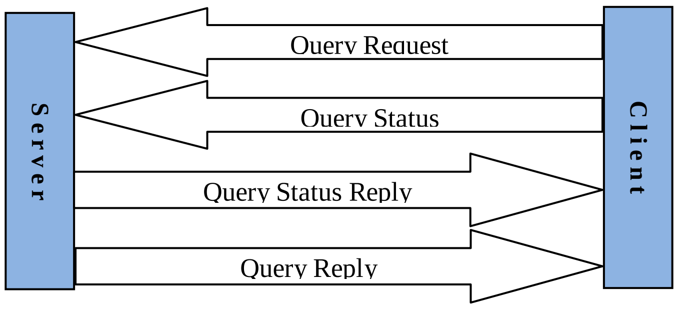

Figure 2.1: Communication between server and client application of HSMF.

2.1 HSMF Communication Messages

The detailed description of HSMF is describe as bellow-

I) Query Request

In this section of the framework, we send the xml message (Table 1) from the client application to server application for sending query request. This message is sent from client application to server application. The format of Query Request message pattern is shown below Table 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QREQ</type>
    <secret>...</secret>
    <session>...</session>
    <clientip>...</clientip>
    <qlang>...</qlang>
    <qpram>...</qpram>
    <query>...</query>
  </message>
</Hadoop>
```

Table 1: Query Request Message Format

If we look closer to the above query request message, it contains different tags. A small detail about the tags is presented below.

<type> helps server application to determine the type of the message. In this section, type value is QREQ to represent Query Request message.

<secret> helps us to determine whether it is a valid message or not. Same secret is also there on server side application. Once an xml message is received to server application first check whether the secret matches or not.

<session>maintains unique session originated from client side to avoid or discard multiple similar request at server side application.

<clientip>determines the server that authenticated client's request its trying to process.

<qlang> used for dynamically select DML to process query.

<qpram> determines the number of column(s) server wants to be processed against the query, this tag is useful for server application for validate the tag <query> and map or bind column(s) . (i.e. possible to search column(s) from query but it's improve processing capability because smart application need to search and map to process)

<query> carries the main query message to server application.

An example of our Query request message is show in table 2

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QREQ</type>

<secret>67333ab60f13242b33408d4242ceac8e</secret>
    <session>w4erg6udtjvdtyj</session>
    <clientip>192.168.1.10</clientip>
    <qlang>HIVE</qlang>
    <qpram>2</qpram>
    <query>select  id,name  from  employee  where
name="gobinda"</query>
  </message>
</Hadoop>
```

Table 2: An Example Of Query Request Message

In this example (Table 2), we see that client application send a request to the server application to know the id and name from the table employee where as employee table store in hive warehouse. User request sending device ip 192.168.1.10 and md5 share secret of UIU is 67333ab60f13242b33408d4242ceac8e. Originating session is w4erg6udtjvdtyj. Clients notify to server application that it's sending only two columns (id & name) via qpram tag. Given query should process by hive representing qlang tag.

II) Query Reply

In this section of the framework, we send the xml (Table 3) message from the server application to client application for sending query request result. This message only has been sent when server application received Query Status message to know the update or output of the previous message Query Request. This message has been sent from server application to client. The format of Query Reply message is shown as bellow table 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QREP</type>
    <secret>...</secret>
    <session>...</session>
    <clientip>...</clientip>
    <queryrow>...</queryrow>
    <queryout1>
      <qpram1>...</qpram1>
      <qpram2>...</qpram2>
    </queryout1>
    ...
    ...
    <queryoutN>
      <qpramN>...</qpram N>
      <qpramN>...</qpram N>
    </queryoutN>
  </message>
</Hadoop>
```

Table 3 : Query Reply Message Format

If we look closer to the above query reply message, it contains different tags. A small detail about the tags is presented below.

<type> helps client application to determine the type of the message. In this section type value is QREP to represent Query Reply message.

<secret> helps us to determine whether it is a valid message or not. Same secret is also there on client side. Once an xml message is received to client application first check whether the secret matches or not.

<session> maintains unique session to avoid or discard multiple similar request at client side.

<clientip>determines the client that authenticated server's request, it's trying to process. In this section clientip tag carry the value of server ip.

<queryrow> helps the client application to determine the number of rows will contain in this message. So the client application can easily maintain a loop to collect data iteratively. If say <queryrow> value 2, then client application can easily generate a tokenized mechanism for first row value under <queryout1> and next <queryout2>.

<queryout1> represents the first row values if multiple rows generated then the suffix of row number added on the tag <queryout1…N>

<qpram1> represents first row's first column value, similarly <qpram2> to <qpramN> contains next column value based on user request.

A example of Query Reply message show in Table 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QREP</type>

<secret>67333ab60f13242b33408d4242ceac8e</secret>
    <session>w4erg6udtjvdtyj</session>
   <clientip>192.168.1.254</clientip>
    <queryrow>2</queryrow>
    <queryout1>
      <qpram1>1</qpram1>
      <qpram2>gobinda</qpram2>
    </queryout1>
    <queryout2>
      <qpram1>2</qpram1>
      <qpram2>yemon</qpram2>
    </queryout2>
  </message>
</Hadoop>
```

Table 4: An Example Of Query Reply Message

In this example (Table 4), we see that client application receives a xml message from the server application which contains two rows and two columns. First row contains 1, gobinda and second row contains 2, yemon. User request sending device ip 192.168.1.10 and md5 share secret of UIU is 67333ab60f13242b33408d4242ceac8e.Originating session is w4erg6udtjvdtyj.

III) Query Status

In this section of the framework, we send the xml message (Table 5) from the client application to server application for get updates of desire query result. After receiving Query Status message, server application first check it's process list, if the query already processed, then server application generate and send Query Reply xml message else if any error occurred by processing query or requested query already under process then generate and send Query Status Reply xml to notify the client application. . The format of Query Status message is show in bellow table 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QSTAT</type>
    <secret>...</secret>
    <session>...</session>
    <clientip>...</clientip>
    <qlang>...</qlang>
    <qpram>...</qpram>
    <query>...</query>
  </message>
</Hadoop>
```

Table 5: Query Status Message Format

If we look closer to the above query reply message, it contains different tags. A small detail about the tags is presented below.

<type> helps client application to determine the type of the message. In this section type value is STAT to represent Query Satus message.

<secret> helps us to determine whether it is a valid message or not. Same secret is also there on client side. Once an xml message is received to client application first check whether the secret matches or not.

<session> maintains unique session to avoid or discard multiple similar request at server side.

<clientip> determines the server that authenticated client's request its trying to process. In this section clientip tag carry the value of client ip.

<qlang> is used for dynamically select DML to process query.

<qpram> determines the number of column(s) server wants to be processed against the query, this tag is useful for server application for validate the tag <query> and map or bind column(s) . (i.e. possible to search column(s) from query but it's improve processing capability because smart application need to search and map to process)

<query> carries the main query message to server application.

This Query Status message is similar to Query Request message except type. In any network circumstances, if server application is not able to receive Query Request message but server application receive Query Status message on same session of Query Request, then server application convert Query Status message into Query Request message and process accordingly. Query Status message send to server application periodically after thirty second (maximum twenty times) until client application get positive reply at server end. After sending these messages to server client application wait thirty sec for getting reply from server. Server application send two type of reply against this Query Status message, Server application check its internal list of query result, if process completed successfully then server reply Query Reply message else reply Query Status Reply message to client application. A example of Query Status message show in Table 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QSTAT</type>

<secret>67333ab60f13242b33408d4242ceac8e</secre
t>
    <session>w4erg6udtjvdtyj</session>
    <clientip>192.168.1.9</clientip>
    <qlang>HIVE</qlang>
    <qpram>2</qpram>
    <query>select id,name from employee where
name="gobinda"</query>
  </message>
</Hadoop>
```

Table 6: An Example Of Query Status Message

In this example (Table 6), we see that server application receives a xml message from the client application to know the status of process id and name from the table employee where as employee table store in hive warehouse. User device ip 192.168.1.9 and md5 share secret of UIU is 67333ab60f 13242b33408d4242ceac8e.Originating session is w4erg6udtj vdtyj. Clients notify to server application that it's sending only two columns (id & name) via qpram. Given query should process by hive representing qlang tag.

IV) Query Status Reply

In this section of the framework, we send the xml message (Table 7) from the server application to client application for giving the update of query processing status. Both Query Status Reply and Query Reply message is generated by the server application when server application receives Query Status message but main difference is ,Query Reply message is generated when request query already processed and result stored into a list else any error or query already in progress mode then server side generated Query Status Reply message. The format of Query Status reply message like as bellow table 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QSTATREP</type>
    <secret>...</secret>
    <clientip>...</clientip>
    <reason>...</reason>
  </message>
</Hadoop>
```

Table 7: Query Status Reply Message Format

If we look closer to the above Query Status Reply message, it contains different tags. A small detail about the tags is presented below.

<type> helps client application to determine the type of the message. In this section type value is QSTATREP to represent Query Reply message.

<secret> helps us to determine whether it is a valid message or not. Same secret is also there on client side. Once an xml message is received to client application first check whether the secret matches or not.

<session> maintains unique session to avoid or discard multiple similar request at client side.

<clientip>determines the client that authenticated server's request, it's trying to process. In this section clientip tag carry the value of server ip.

<reason>helps client application to determine the reason of processing failure or if query is under process.

If client application receives Query Status Reply message then stop sending Query Status message and display result of the query user UI.A example of Query Status Reply message show in Table 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<Hadoop>
  <message>
    <type>QSTATREP</type>

<secret>67333ab60f13242b33408d4242ceac8e</secret>
    <clientip>192.168.1.254</clientip>
    <reason>INVALID QUERY FORMAT</reason>
  </message>
</Hadoop>
```

Table 8: An Example Of Query Status Reply Message

In this example (Table 8), we see that client application receives a xml message from the server application which contains a reason invalid query. Originating session is w4erg6udtj nvdtyj. User request sending device ip 192.168.1.254 and md5 share secret of UIU is 67333ab60f13242b33408d4242ceac8e.

3 Architecture of Proposed Framework

Our proposed Framework can easily communicate with hadoop over internet or intranet to almost all platform like a apps written in android or symbian or in desktop application. In the architecture of this framework a java and hadoop library based socket server application stay at the namenode server of hadoop which main goal to accept client xml messaging request on a particular socket port (say 9090) and reply accordingly after client request been processed. A communication model of our HSMF is shown in the figure 3.1.

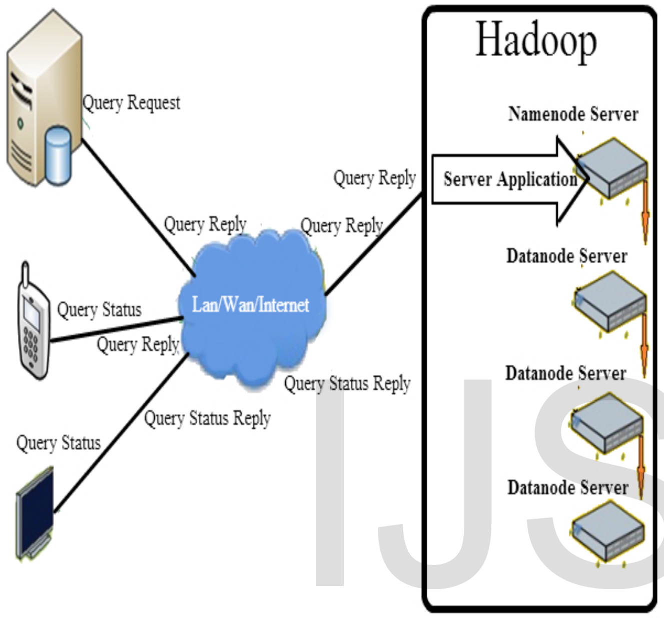

Figure 3.1: HSMF Communication

3.1 Server Side

According our HSMF, a server application was run as demon to take request, process it and reply to client application. The data flow diagram of server application are given in bellow figure 3.2

Data flow Diagram

A details data flow diagram of HSMF server application process is shown as bellow.

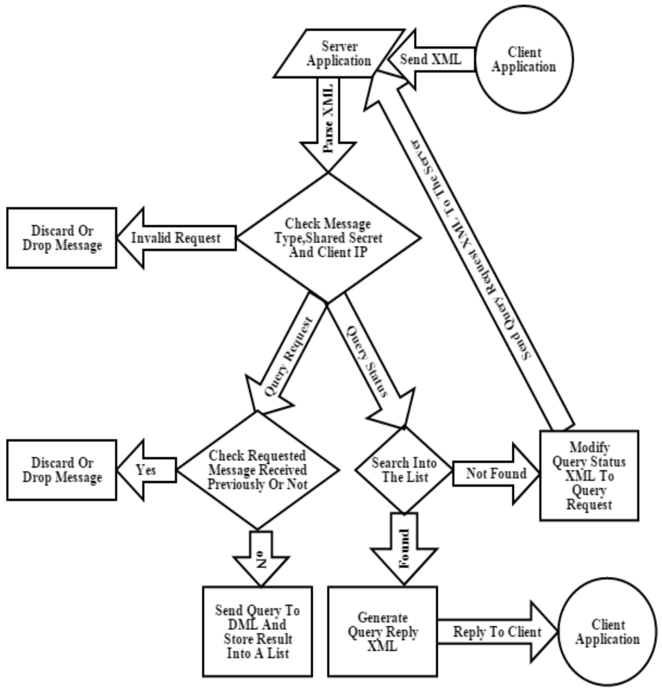

Figure 3.2: Server Side Data Flow Diagram.

I. In the server side , a server application is always listening on a particular port say 9090 on demon mode, once is get any xml request from the client it's it send to XML parser to parse the message to get message type, session, origination ip, shared secret and query message.

II. A checker process check message type from tokenize message ,if the message type not match with Query Request (QREQ),Query Status (QSTAT) then send to Discard Or Drop the client message .

III. If the Message Type Match with Query Request (QREQ) then check the session whatever same message accept previously or not.

If same message receive and it's under process simply discard with no reply else store request in an array or could be in database like mysql or flat file system with query for feature process.

A background thread being run on system to check the list and to collect unprocessed query message with query processing language request sent by the client .

IV. If the Message Type Match with Query Status (QSTAT) then check the session whatever same message accept previously or not.

If same message receive server application search internal list whatever is process successfully or not.

If requested query status already process then server application generate Query Status Reply (QSTATREP) xml with proper data and send that to client via socket (9090 port).

If server application find out that Query Status (QSTAT) message not process previously then generate Query Status Reply(QSTATREP) xml with reason Not Found and modify XML like Query Request (QREQ) message and send that to server application unprocessed list .

This scenario could be happen only if somehow server application not received Query Request (QREQ) previously but client send Query Status (QSTAT) message to get the latest update.

3.2 Client Side

According our HSMF, a client application should follow XML messing pattern, so that client application easily communicate with server to get requested hadoop data. The data flow diagram of client application is given in bellow figure 3.3.

Data flow Diagram

A details data flow diagram of HSMF client application process is shown as bellow.

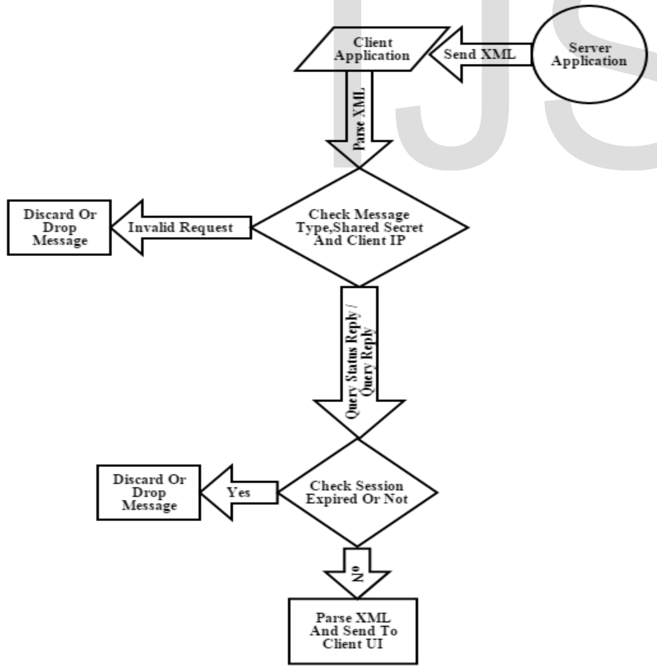
Client Application
Send XML
Server Application
Parse XML
Discard Or Drop Message
Invalid Request
Check Message Type,Shared Secret And Client IP
Query Status Reply / Query Reply
Discard Or Drop Message
Yes
Check Session Expired Or Not
No
Parse XML And Send To Client UI

Figure 3.3: Client Side Data Flow Diagram.

I. In the client side, client application is not always listening on a particular port. Once a query request received by the client application, it's generate a Query Request (QREQ) XML with message type, session, origination ip ,shared secret send to server.

II. After sending Query Request (QREQ) message to server, client application periodically send Query Status (QSTAT) every 10 second until received Query Status Reply (QSTATREP) message from server application.

III. When client application received Query Status Reply (QSTATREP) xml reply from the server it's it send to XML parser to parse the message to get message type, session, origination ip ,shared secret and query message.

IV. A checker process check message type from tokenize message ,if the message type not match with Query Status Reply (QSTATREP) then send to Discard Or Drop the server message .

V. If the Message Type Match with Query Status Reply (QSTATREP) then check the session whatever same message accept previously or not. If same message receive and it's under process simply discard or drop else display receiving data from server.

4 Conclusion

This paper described the implementation and evaluation of an alternative communication between hadoop and smart application over TCP/IP. Our HSMF can be used as a plug-in of hadoop which is useful for commercial cloud systems [5] to gain good benefits from HSMF.

We have presented some key technologies that deal with large scale of data sets in HSMF including multi-threading, xml messaging mechanism and buffer allocation management. Then we introduced two important components- XML messaging based Socket Server and Client. When client gets a any request from user, it will send that request to the server. In server side there is a thread to read client xml request by shared secret based security checking mechanism and keep the request in a pool. Another thread will be wake up to pull the request and send data to hadoop hive or pig depending on the choice of clients and store the result in a flat file or memory pool for feature response in output request. Once server side receives output request reply instantly via TCP/IP socket. One receive thread in client side receives data and display to client User Interface. All of the buffers are allocated to a memory pool in the beginning, threads can get empty buffers from this memory pool to do their work and return them when they don't need them.

The discussion regarding our HSMF outlines the possible ways of how a user or device can easily communicate with hadoop data. From the experimental results we find that our HSMF can achieve better performance and the scalability is good.

In the future, we will need to test our HSMF on much larger commercial cloud systems, which can better demonstrate the benefits of our work. Currently HSMF works with text based data or query. We will also work with HSMF for image and voice type data processing. .

ACKNOWLEDGMENT

The authors wish to thank Prof. Dr. Monzurur Rahman. This work was supported in part by a grant from United International University.

REFERENCES

[1] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” in Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6, ser. OSDI’04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[2] White T (2010) Hadoop: the definitive guide. Yahoo Press.

[3] Hadoop: The Definitive Guide, Third Edition by Tom White (O'Reilly - 2012)

[4] Apache, “Hdfs”, http://apache.hadoop.org/hdfs/ Accessed: 02/12/2014.

[5] Cloud computing, http://en.wikipedia.org/wiki/Cloud computing Accessed: 02/12/2014.

[6] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google File System,” pub. 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October,2003

[7] Fadika, Z. ,”Parallel and distributed approach for processing large-scale XML datasets” in Grid Computing, 2009 10th IEEE/ACM International Conference, Banff, AB

[8] Peisen Yuan, Chaofeng Sha, Xiaoling Wang, Bin Yang, Aoying Zhou, Su Yang , “XML Structural Similarity Search Using MapReduce” in 11th International Conference, WAIM 2010, Jiuzhaigou, China, July 15-17, 2010. Proceedingson.

[9] Karnati Hemanth, Talluri Ravikiran, Maddipati Venkat Naveen,Thumati Ravi "Security Problems and Their Defenses in TCP/IP Protocol Suite",International Journal of Scientific and Research Publications, Volume 2, Issue 12, December 2012

[10] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang 0002, Suresh Anthony, Hao Liu, and Raghotham Murthy, “Hive - a petabyte scale data warehouse using hadoop,” In ICDE, pages 9961005, 2010..

[11] Apache Pig Wiki. http://wiki.apache.org/pig/PigPerformance. Accessed: 02/12/2014.

[12] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins, “Pig latin: a not-so-foreign language for data processing” In SIGMOD08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 10991110, New York, NY, USA, 2008. ACM.

[13] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears, “MapReduce Online” In NSDI, pages 312328, April 2010.

[14] Yandong Wang, Xinyu Que, Weikuan Yu, Dror Goldenberg, Dhiraj Sehgal, “Hadoop Acceleration Through Network Levitated Merge” Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011. ACM.