Advances in Computer Science Research, volume 44
3rd International Conference on Wireless Communication and Sensor Network (WCSN 2016)

File Tracking Method and System Based on Life Cycle Tree

Guang-Yu Gu, Xiang Cai, Shu-Juan Zhang
State Grid Anhui Electric Power Research Institute, Hefei, China
E-mail: gugy2051@ah.sgcc.com.cn, caixiangcx3@163.com, zhangsj202x@ah.sgcc.com.cn

Qing Yi
Anhui Electrical Engineering Professional Technique College, Hefei, China
E-mail: yiqing302@163.com

Kun Ding
SKLOIS, Institute of Information Engineering, CAS, Beijing, China
E-mail: dingkun@iie.ac.cn

***Abstract*-There are many difficult challenges, such as variety, quantity, distribution etc. in the unstructured data tracking management, which lacks an effective file tracking management solution. In this paper, based on the storage mechanism of HBase, we propose a data life cycle tree storage structure, and realize the whole life cycle tracking of the file. At the same time, the system can provide visual analysis services, such as frequency of operation, distribution of hot spots, data types, the breadth and depth of the life cycle tree, and the data volume trend prediction etc. Compared with common data management systems, this system can be more efficient and fast to query and build the data life cycle tree, and quickly realize the visual analysis of the document.**

Keywords- data life cycle; tree storage structure; file tracking management system

I. INTRODUCTION

How to carry on a reliable and effective management to the massive data under the big data environment in the era of comprehensive data explosion has become the hot spot of the data security research. For the text, image, video and other unstructured data, their data type, quantity, distribution, size and other characteristics bring a challenge to the supervision to the data object of the file system.

There are the following problems and challenges in the management of the unstructured data:

- The tracking management of operation behavior in the whole life cycle[1] of data objects is not systematic, which lacks an intuitive structure like the life cycle tree[2][3]. In the face of massive data audit log, however, the efficiency of constructing and querying the life cycle tree is quite low. General systems try to build file life cycle tree by traversing table or querying the index table, which is difficult to compromise on time efficiency and space capacity.
- Lack of effective data visualization analysis. Huge amount of data value is hidden in the audit logs. Data visualization analysis not only allows users to intuitively understand the basic situation of the current data management, but also to view the trend of using frequency, data types, capacity, distribution and other aspects, which helps to improve the efficiency of data real-time management.
- It is difficult to locate event and collect evidence fast and accurately when the data object is abnormal. Once data leak event occurred, a lot of manual work like querying for the audit logs is required to locate the key information, which is time-consuming and less accurate.

To solve the above problem, we proposed a tree-storage structure under HBase [4] mode. In the case of massive data, the efficiency of life cycle [5] tracking and query function is far superior to the common data management system. Also, we implement the unstructured data object visualization analysis efficiently with the tree-storage structure.

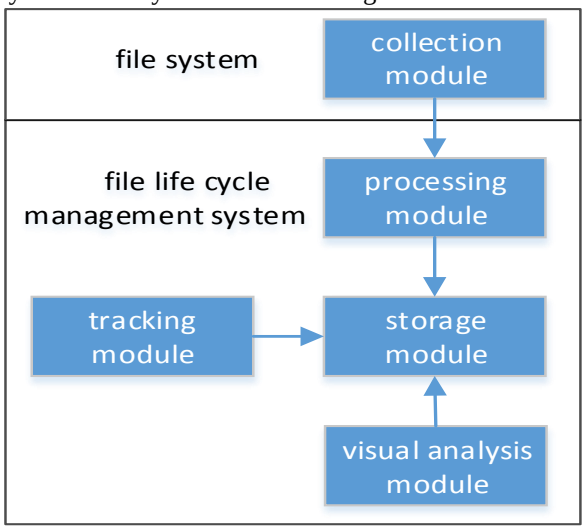

Figure 1. System function modules.

As is shown in Fig.1, the proposed system is composed of collection module, processing module, storage module, tracking module and visual analysis module, which can track and manage the whole life cycle of unstructured data objects. Based on HBase, the system may track the life cycle of data objects efficiently. At the same time, the visual analysis module can carry on statistical analysis to the massive audit log of the storage, display the visually results.

II. DATA LIFE CYCLE TREE

A. Data Object Life Cycle

The life cycle process of unstructured data in a file system may include: create, store, access, transfer, destruction and recovery. Each process in the file management system corresponds to one or more operation behavior.

"Create" refers to the process of data generating in the file system, including a series of data initial behavior. "Storage" refers to the persistent process in the storage device of data objects that have been generated. "Access" refers to the operation of read, modify etc. to the data that from the persistent state to the instantaneous state. "Transfer" refers to the process of data migration. "Destruction" and "recovery" means the removal of data objects and the reduction process of data after deletion.

B. File System Operation Type

For example, the common Windows OS that based on NTFS [6], each life cycle process is corresponding to at least one operation type of data objects.

As is shown in Fig.2, new data objects are generated by "New" operation, which belongs to "create" process. "Save" and "Save as" persistent data objects from memory to disk, which can be considered as "storage" process. "Open", "Preview" and "Rename" operations fetch data from disk into memory for reading or modifying, and pertain to "access" process. "Move", "Cut" and other operations migrate data objects, which can be assigned to "transfer" process. "Delete" operation removes the data object, so that it is belonging to "Destruction" process. "Restore" and "Undo" operations restore the deleted data, which belongs to "recovery" process.

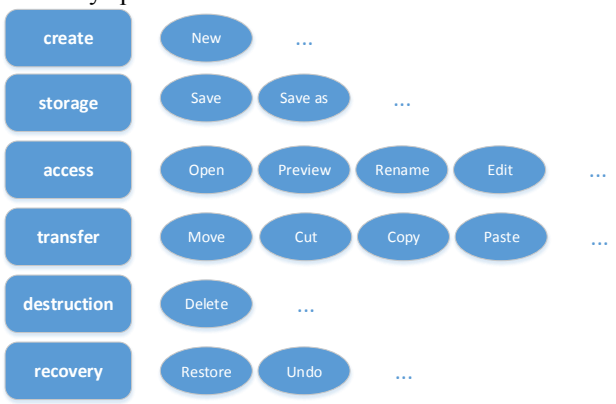

Figure 2. File life cycle and operation type.

C. Tree Structure Algorithm

1) Node classification and storage algorithm

The classifier will receive behavior events and classify them. Events that pertain to the type of data creation belong to class "A". Events that change the "dataId"(which will be described in III.B section) of data objects belong to class "B". And the rest are belonging to class "C". The storage algorithm will firstly convert the behavior event into a log, and then process it according the corresponding classification. The generated log can be stored after adding the field of the root node pointer(i.e., its own row key) if it is an "A" class event. In the case of class "B", the original "dataId" will be extracted from the event. Then the algorithm will find the latest log in the "dataId" from the storage, and copy the root node pointer from the log to the new one. After storing the new log, the algorithm will append its pointer to the corresponding field of the original log. While in the case of class "C", the algorithm will extract the "dataId" from the event and find the latest log in such "dataId", and add the root node pointer (field data:root_rk) to the new log before storing it.

The pseudo code of the algorithm is as followed:

```
// Algorithm received behavioral events
Event e = FUNCTION receivedEvent();
// Classification of events using a classifier
FUNCTION classifier(Event e){
IF(e.type="create")
RETURN class_A;
ELSE IF(e.dataId!=e.original_dataId)
RETURN class_B;
ELSE
RETURN class_C;
}

// Generate log row key
RowKey rk = FUNCTION getRK(e.dataId,e.timestamp);
Log log = FUNCTION convert2Log(rk,e);// Generate log
IF(e==class_A){
log.addColumn("data:root_rk",rk);
FUNCTION save(rk,log);
}
IF(e==class_B){
Log ori_log = FUNCTION findLatest(e.original_dataId);
log.addColumn("data:root_rk",ori_log.root_rk);
FUNCTION save(rk,log);
 //Append node pointer of new log to the original log
ori_log.appendCloumn("data:sub_rk",rk);
FUNCTION update(ori_log.rk,ori_log);
}
IF(e==class_C){
Log pre_log=FUNCTION findLatest(e.dataId);
log.addColumn("data:root_rk",pre_log.root_rk);
FUNCTION save(rk,log);
}
```

The algorithm complexity is O(1). The algorithm is designed to store the tree data structure, as each node has a pointer to the root node. For the nodes whose "dataId"s are not changed, which are stored in order, the parent node can index to the child nodes quickly. For the nodes whose "dataId"s are changed, the parent node has the pointer(s) to the child node(s).

2) Constructing tree algorithm

The constructing tree algorithm is used to construct the data tree. The input of the algorithm is "dataId". The algorithm will find the latest log in the received "dataId" from the storage firstly, and get the root node pointer from the log. And then the algorithm will obtain the root node log

by the pointer. After that, the algorithm will construct the data tree by iteration from the root node. For those "dataId"-changed nodes, the algorithm traverses the sub nodes by the sub node set of the parent node. While for those "dataId"-not-changed nodes, the algorithm can index to the child nodes directly. After the iteration, the constructing of data tree will be completed.

The pseudo code of the algorithm is as followed:

```
FUNCTION buildTree(DataId dataId){
Log log = FUNCTION findLatest(dataId);
// Get the root node row key
RowKey root_rk=log.root_rk;
// Find the root node log
Log log = FUNCTION find(root_rk);
FUNCTION addRootNode(log);// Add root node
FUNCTION buildTree(root_rk);
}
FUNCTION buildTree(RowKey rk){
Log log = FUNCTION find(rk);
IF(log.sub_rk!=null){
FOR EACH rk IN log.sub_rk{
Log log = find(rk);
//Add sub node
FUNCTION addChildNode(log);
// Iterative build tree
FUNCTION buildTree(rk);
}
}

Log next_log = FUNCTION findNext(rk);
IF(next_log!=null){
//Add sub node
FUNCTION addChildNode(next_log);
// Iterative build tree
FUNCTION buildTree(next_log.rk);
}
}
```

The algorithm complexity is O(n), and n is related to the number of nodes.

III. MODULE DESIGN AND SYSTEM IMPLEMENTATION

A. Collection Module

The collection module is installed in each user terminal as a behavior monitoring and recording program, which can monitor file objects in the specified disk directory, record a series of data manipulation actions by the operator in the file system, and transmit the relevant behavior events recorded by the system to the processing module in a specific format.

The information collected by the collection module mainly includes the filename, the absolute path, the data type, the operation, the operation time, the operator etc. For the filename, the absolute path and data type, if they are changed by the operation, the collection module will record the changes before and after. For example, the module will record that the data object is renamed from "TestA.txt" to "TextB.txt".

There are two ways of data transmission between the collection module and the processing module: real time synchronization and pseudo real time synchronization. When it runs in real time synchronization, the collection module will send the relevant information to the processing module immediately once detects data operations, which is suitable for the situation that the operation frequency under the monitoring directory is moderate and the scenes that the terminal cluster is smaller. When it runs in pseudo real time synchronization, the relevant information will be cached to the local side, and will be packaged at a certain time to send out to the processing module. The time interval is so small that the user will be difficult to perceive. Such way is suitable for the situation with high operation frequency and the scenes that the terminal cluster is large.

The detailed work flow of the collection module is as follows:

- Start module, collect the physical address, network address, the OS user and other relevant information. Then the monitoring directory will be configured in order to capture file operation behavior.
- Capture file operations, and extract the related information of the file, and then generate an operation event.
- If runs in real time synchronization, send the event to the processing module immediately; if run in pseudo real time synchronization, cache the event to the local side.
- The pseudo real time synchronization algorithm calculates the time for synchronization, at which the collection module package the local events and send them out to the processing module.

B. Storage Module

The storage module is used to store the file life cycle logs. The HBase table structure is used in the storage module. The row key is designed in the form of "RowKey = dataId + (MAX_VALUE – timestamp)", while "dataId = Hash(Mac + path + filename + extension)". Such design can meet the requirements of high performance: "dataId" is processed by 128 bit hash operation, so the row keys can achieve load balancing with a fixed 16 byte output. And the timestamp uses the "Long" type with 8 byte array. Thus the total length of the row key is 24 bytes. The design of "MAX_VALUE - timestamp" can ensure the log flashback storage by the time. The row key elements are shown in TABLE 1.

TABLE I. ROW KEY ELEMENT

Definition	Description
Mac	Record the physical address of the hardware of the file system
timestamp	Record the time stamp of the current event
path	Record the absolute path of the data object
filename	Record the name of the data object
extension	Record the file type(extension) of the data object

In the storage module, each line in the audit table represents an audit log, and columns in each log are corresponding to the timestamp, data name, id, extension name, path, OS user, physical address, network address, OS, file system, tree structure node pointer and other information. HBase is a scalable columnar storage database, which may not write for every column of the table in a certain stored procedure, for that additional columns can be added dynamically.

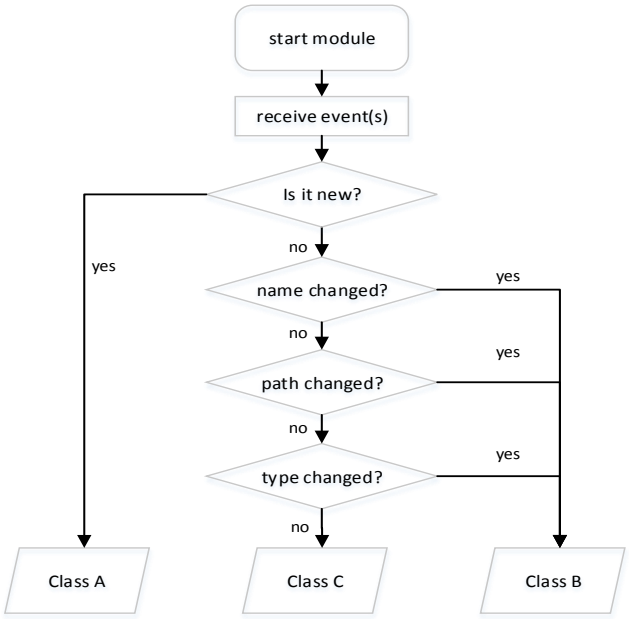

Figure 3. Classification logic.

C. *Processing Module*

1) *Classifier*

According to the tree structure classification algorithm, the classification of the processing module is designed as follows:

Definition 1. In the file system, if a new data objects is created, the operation behavior is belonging to class "A".

Definition 2. In the file system, if the path, or the name, or the type (extension) of the data is changed after the operation, such operation behavior is belonging to class "B".

Definition 3. In the file system, operations excluding class "A" and "B" are belonging to class "C".

The detailed work flow of classifier is shown in Fig.3.

2) *Processing procedure*

The changes of data path, name and type before and after the operation will be recorded by the collection module, based on which the classifier of the processing module is to classify the received behavior events. For example, "new" operation events are divided into class "A"; "move", "rename", "modified extension" operation events are divided into class "B"; "open", "edit" operation events are divided into class "C".

For class "C" operation events, the row key will be generated according to generation rules mentioned in section III.B. The collected information will be filled to the corresponding column before the item are written to the audit log table of the memory module.

For class "B" operation events, at least one of the data path, name and type will be changed, which means the "dataId" will be changed according the row key generation rules. The processing module will find the latest log in the audit log table by the original "dataId", and copy the "data:root_rk" column to the new log. After storing the new log, the module will append its row key to the "data:sub_rk" column of the original "dataId" log. "data:root_rk" and "data:sub_rk" can be regarded as node pointers of the tree data structure, pointing to each other.

As described in III.A, the collection module can record the changes before and after. For those "dataId"-changed events, it is feasible to obtain the original "dataId" according to the information before the change.

D. *Tracking Module*

This module is used to locate the relevant audit log and construct the life cycle tree. According to the row key generation rules, "dataId" is composed of the data path, name, type, and the physical address of the terminal. Combined with the timestamp to get the row key, the module can quickly locate the corresponding audit log.

Each node of the life cycle tree corresponds to a log record in the audit log table of the memory module. For each data object recorded by the log, the module can query any nodes in the tree according to the row key and node pointers without traversing all the records of the entire storage module, which is fast and efficient.

Due to the characteristic of row key in HBase, it no longer needs to traverse the table when constructing the life cycle tree. As a result, the constructing efficiency is superior to the common file management system under big data situation. Also, the index table is not needed, which saves the system memory.

An example of the life cycle tree is shown in Fig.4. When a new file object is copied from the original one, it will be described as a fork. Fig.5 shows the related table in HBase according to Fig.4.

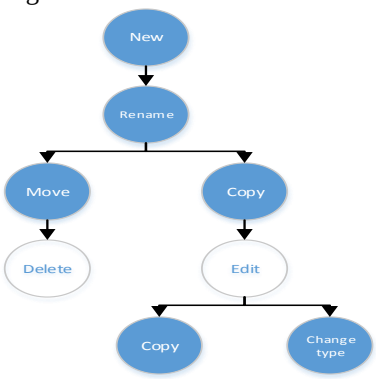

Figure 4. Life cycle tree of the data object.

The detailed work flow of the tracking module is as followed:

1) If receiving a location instruction, the module will generate the row key according to the received data path,

name, type, physical address, timestamp and other information to index and return the log entry;

2) If receiving a tree constructing instruction, the module will locate to the latest log of the data object firstly, find the root node by its "data:root_rk" column, and then construct the tree by the algorithm.

E. Visual Analysis Module

This module can statistically analyze the audit log in the storage module. The operation frequency ranking, hot spot distribution ranking, data type ranking, life cycle tree width and depth ranking, and the trend prediction of the data can be obtained.

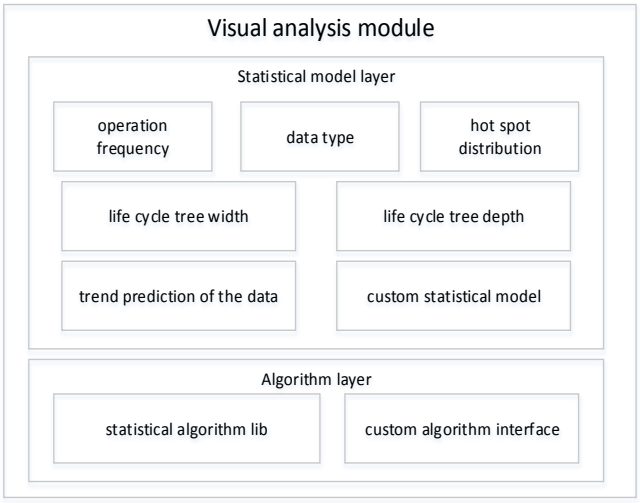

Figure 5. Schematic diagram of visual analysis module.

As is shown in Fig.6, this module is composed of the statistical model layer and the algorithm layer. The former is used to define statistical requirements, and the latter define the relevant algorithm logic according to the requirements. In addition, a custom interface is reserved to add new statistical model and related algorithm.

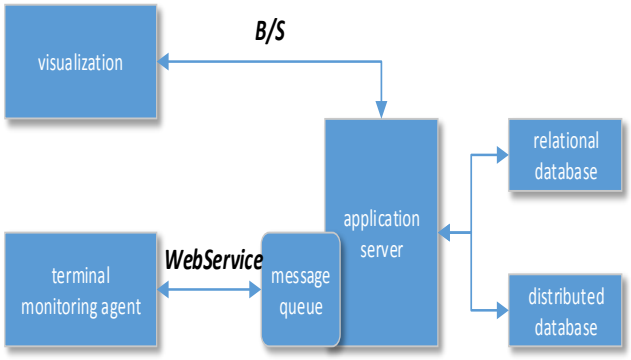

Figure 6. System architecture.

This module uses MapReduce, a distributed computing tool, to do the corresponding statistical analysis of the data for different requirements. Distributed computing can be implemented in parallel computing, which greatly reduces the time required for statistical analysis.

IV. IMPLEMENTATION OF THE SYSTEM

The implementation of the life cycle tracking and visual analysis system is based on SOA multilayer framework [7]. The system architecture is shown in Fig.7.

Rowkey	data								
	dataId	filename	path	extension	class	op_type	root_rk	sub_rk	…
					……				
3003df…fffeb9832	3003df…	Newtext	C:/	txt	A	New	3003df…fffeb9832		…
					……				
639d60…fffeb9700	639d60…	TestA	C:/	txt	B	Rename	3003df…fffeb9832	755ee8… fffeb9402; 8004a9… fffeb8856	…
					……				
698a83…fffeb7044	698a83…	TestA	F:/	txt	B	Copy	3003df…fffeb9832		…
					……				
755ee8…fffeb9009	755ee8…	TestA	D:/	txt	C	Delete	3003df…fffeb9832		…
755ee8…fffeb9402	755ee8…	TestA	D:/	txt	B	Move	3003df…fffeb9832		…
					……				
78add2…fffeb73da	78add2…	TestA	E:/	doc	B	ChangeType	3003df…fffeb9832		…
					……				
8004a9…fffeb8332	8004a9…	TestA	E:/	txt	C	Edit	3003df…fffeb9832	698a83… fffeb7044; 78add2… fffeb73da	…
8004a9…fffeb8856	8004a9…	TestA	E:/	txt	B	Copy	3003df…fffeb9832		…
					……				

Figure 7. Table of HBase.

The collection module is developed by C language as the application service component, while the server using Java.

The back-end development uses Spring MVC architecture [8], integrating with the front-end in B/S mode. The web layer respond HTTP requests by servlet(s) [9], and call the corresponding service(s) in the back-end to complete the business logic before returning the result to the front-end. The front-end uses JavaScript for processing and rendering after receiving the results, achieving RIA (Rich Internet Applications) by using JSP and AJAX [10].

A unified service framework is adopted in the service interface layer, docking with Webservice, HTTP and other interface protocol, using JSON format for transmission of data.

Mysql is used as a relational database in the data storage layer, storing the user, permissions, configuration, model and other system metadata, integrating with the back-end by Hibernate [11]. HBase is used as a distributed database in the data storage layer to store the audit logs and other business data, which is based on HDFS [12], and integrates with the back-end well by the simplehbase plug-in.

The detailed work flow of the life cycle tracking and visual analysis system is as followed:

1) The specified disk directory is monitored by the terminal agent. All of the file operations are recorded and converted to events. These events will be sent to the message queue [13] of the application server in the real time or pseudo real time manner.

2) The message queue submits the receiving events to the application server in chronological order. The application server calls the classifier to classify these file operation events.

3) The application server processes the classification results and converts the event(s) to log(s), and then stores log(s) to the distributed database.

4) Once receiving the relevant instruction, the application server locates the right log in the storage, or constructs life cycle tree according to request parameters. Users can view the log or tree through the browser.

5) Users can send a visual analysis request on the browser, and the application server tries to find the statistical models from the relational database, call the relevant algorithms to analyze the information in the distributed database, and get the final analysis results.

V. CONCLUSION

It is a common problem that how to manage and track the life cycle of unstructured data with different variety, quantity and distribution. We propose a data life cycle tree storage structured which is suitable for HBase. On the basis of it, we realizes the life cycle tracking and visual analysis system. Compared with common data management system, this system can efficiently and quickly query and construct the data life cycle tree, and has a satisfactory adaptability to the increment of the data object log. Meanwhile, the system can statistically analyze massive logs according to the requirements of the user, and obtains the visual analysis results.

REFERENCES

[1] H.V Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J.M. Patel, R. Ramakrishnan, and C. Shahabi, “Big data and its technical challenges,” Communications of the ACM, 2014, 57(7),pp. 86-94.

[2] P. Hsiao and W. Feng, “Using a multiple storage quad tree on a hierarchical VLSI compaction scheme,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1990, 9(5), pp. 522-536.

[3] M.H. Pan and Z.H. Yao, “The application of multilayer tree structure in DICOM,” Medical Imaging. International Society for Optics and Photonics, 2006, 614519-614519-11.

[4] L. George, HBase: the definitive guide. " O'Reilly Media, Inc.", 2011.

[5] A. Simonet, G. Fedak, and M. Ripeanu. "Active data: a programming model to manage data life cycle across heterogeneous systems and infrastructures," Future Generation Computer Systems, 2015, pp. 25-42.

[6] G. Fellows, “NTFS volume mounts, directory junctions and $Reparse,” Digital Investigation, 2007, 4(3), pp. 116-118.

[7] L. Wu, G. Barash, and C. Bartolini, “A service-oriented architecture for business intelligence,” IEEE international conference on service-oriented computing and applications (SOCA'07). IEEE, 2007, pp. 279-285.

[8] G. Warin, Mastering Spring MVC 4, Packt Publishing Ltd, 2015.

[9] J. Hunter and W. Crawford. Java servlet programming. " O'Reilly Media, Inc.", 2001.

[10] N.C. Zakas and J. Fawcett, Professional Ajax. John Wiley & Sons, 2007.

[11] Y. Ren, D. Jiang, T. Xing, and P. Zhu, “Research on software development platform based on SSH framework structure,” Procedia Engineering, 2011, pp. 3078-3082.

[12] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). IEEE, 2010, pp. 1-10.

[13] J.A. Zounmevo, A. Afsahi, “A fast and resource-conscious MPI message queue mechanism for large-scale jobs,” Future Generation Computer Systems, 2014, pp. 265-290.