ACCESS CONTROL AND SECURITY OF DATASETS BY USAGE TRACKING USING BLOCK CHAIN TECHNOLOGY

Piyush Bhatt
Master of Science in Computer Science
Oklahoma State University
Stillwater, Oklahoma

Dr. Johnson Thomas

Thesis Advisor

Committee Members:

Dr. Johnson Thomas

Dr. Christopher Crick

Dr. K.M.George

Introduction to Big Data

- Big Data is the large amount of data which is impractical to be processed using the conventional database systems. It consists of very large datasets.
- Every day 2.5 Quintillion bytes of data is created from myriad of sources.
- Organizations with large number of customer base like Facebook, Amazon or Google have to deal with huge amount of data.
- There are 5 V's of Big Data: Volume, Velocity, Variety, Veracity and Value.
- Big Data technologies focuses on data creation, it's storage, it's retrieval and the analysis that is efficient and effective in terms of the 5 V's mentioned above.
- Example of Big Data Technologies: Hadoop, Apache Pig, Zookeeper, Hortonworks Data Platform(HDP).

What is blockchain?

- Blockchain is the combination of P2P networking and Distributed Database.
- It is also known as Distributed ledger technology and Trust Protocol.
- Blockchains are secure. It is built in a way to prevent hack attacks or unauthorized modification.
- Blocks are linked to each other using their individual hash values(block headers), forming a chain. Thus the name Blockchain.
- A blockchain network consists of multiple nodes and each node has the exact replica of the entire blockchain. If one of the block chain at one of the node gets corrupted, it can be repaired using the copy from the other node.
- Blockchain was first used in Bitcoin. Bitcoin is a cryptocurrency or in simple terms Digital Currency. It works using blockchain as a protocol.

What does a block consist of?

A block consists of 5 kinds of data:

- Version number.
- Hash of the previous block.
- The information of the transaction (the actual data).
- Timestamp of creation of the block
- Number of bits of the data.
- These 5 elements are combined in such a way that they produce a block header which is responsible for creating a connection to the next block.
- Hash value can be calculated using Cryptographic Hash functions. There are many functions available like SHA-256,MD5, RIPEMD-160.

The problem

- Since every dataset has different kinds of data, 2 or more datasets can be combined to fetch the critical or sensitive data which then can be misused for variety of purposes.
- For example one dataset has Candidate_Id, Name, Date of Birth and Place of Birth of an individual. Another dataset might have the Person_Id, Candidate_Id, Date of Birth and Phone Number. The third dataset might have Person_Id and SSN. If an unauthorized user is able to get access to all these 3 datasets, he can retrieve sensitive information of the person.
- This can prove a catastrophe for the business capturing data and can become a legal liability.

Literature Review

Other researches done in this area

Access control of sensitive data in HDFS

- It was done by Yenumula B. Reddy.
- For two businesses to be involved in accessing each other data, a security token is used.
- TokenID={ownerIDc, renewerId, issueDate, maxDate, sequenceNumber, outputCheck}
- ownerIDc={ownerId, accessLimit, KeyId, expirationDate}
- OutputCheck verifies the output generated after the work is done.
- Every authenticated user n_i will have a token which will have a_i : a set of access rights.
- If an unauthorized user tries to access data or the output doesn't match the permissions he will be blocked and reported.
- If a hacker tries to simulate a query as an authorized user he can be caught because of multiple reasons: He might not know about the access rights, the queries might not match, if he is using a particular computer system, he will get locked out of that system.
- In this model there is a chance that the sensitive information might be accessed without any trace of any attack.
- Even read access rights provided to a user will result in the exposure of the data.

Vigiles: Fine-grained Access Control for Map Reduce Systems

- Done by Huseyin Ulusoy, Murat Kantarcioglu, Erman Pattuk, Kevin Hamlen.
- Vigiles is like a firewall which provides Fine Grained Access Control Predicates. It acts as a middleware between users and the Mapreduce system.
- These predicates run Access Control Filters(ACF) which provide access based on the security policies and protocols.
- A user can communicate with Vigiles by using the computer Screen (OS interface).
- Vigiles executes the job on behalf of the user and since it has all the file access parameters, it does the work and provides the output.
- Although Vigiles provide a way to identify sensitive data, it all depends on the admin. Admin decides which data is sensitive. There's no way which track which user access which data.

Access Control for Big Data using Data Content

- Content based access control (CBAC) for big data.
- It is an additional access control scheme along with the existing Database access control rules.
- Policy: ACR = {subject, object, action, f (u, di) where u is the subject and d is the data object.
- It is designed for areas where a certain amount of approximation regarding access control is applicable.
- A user can access a few more or less records than it is assigned by the administrator.
- CBAC provides the users a special role which allows them to access a certain amount of data.
- It uses a function: f (u, di) = {true, false}. Access to a record will be provided if the output of the function is true.
- This model is good but there's always the risk of a user can access a record which he was not supposed to access and there would be no record of him accessing it since CBAC provides the access based on approximation.

An Access Scheme for Big Data processing

- Big Data access control requires additional access control capabilities.
- This Access control scheme has some components:.
- Security Agreement(SA) between Big Data source and the Master System defining the security classes.
- Trust Cooperated Systems List(TCSL) is a list of trusted Cooperated Systems which are recognized by the master system.
- Master System Access Control Policy(MSP) is a set of rules imposed by Master System to enforce Access Control on Cooperated Systems.
- Cooperated System Access Control Policy(CSP) allows the Cooperated Systems to control the access to the distributed Big Data data/process by considering the processing capabilities and security requirements of the systems.
- This model is very good and secure but someone can bypass the security classes or the security officer can modify the policies. Human error can be a factor in this.

Conclusion of literary review

- There is very limited work on detecting malicious or unauthorized modifications to big data. The aspect of security when datasets are merged or joined has also not been addressed in previous works. None of the above approaches provide a proper and accurate mechanism for securing big data.

Proposed Solution

- A system is required to track the usage of datasets.
- Every time a dataset is accessed, the usage of that data shall be captured.
- This system should be secure so that no one can make any unauthorized changes to it.
- Block chain comes into picture. A blockchain will be created every time a dataset will be introduced. Every dataset will have it's own block chain. Every block in the dataset will have username of the user accessing the dataset, the name of the dataset, the type of access used(Command Line Interface or Map Reduce) and command line operation(cat, copy, move, put). Since the block chain is quite secure and immune to any kind of cyber attacks, the usage of the datasets would remain safe.

Creation of block chain

Creation of Genesis Block

- Name of the dataset created, username, method using which the dataset was created (Command Line Interface or Map Reduce) and the command line operation using which the dataset was created (copy, move, put) will be captured.
- The values captured above would be hashed using SHA-256.
- A version number would be assigned to the genesis block.
- The number of bits of the data in the genesis block would be calculated.
- A timestamp would be entered for this block. The timestamp would be the creation time of the block.
- Once the hashed value of the data is calculated, the version number, the number of bits and the time stamp, will be concatenated and then the hash of this string will be calculated.
- This will be the blockheader of the genesis block. This hash value will be used in the next block.

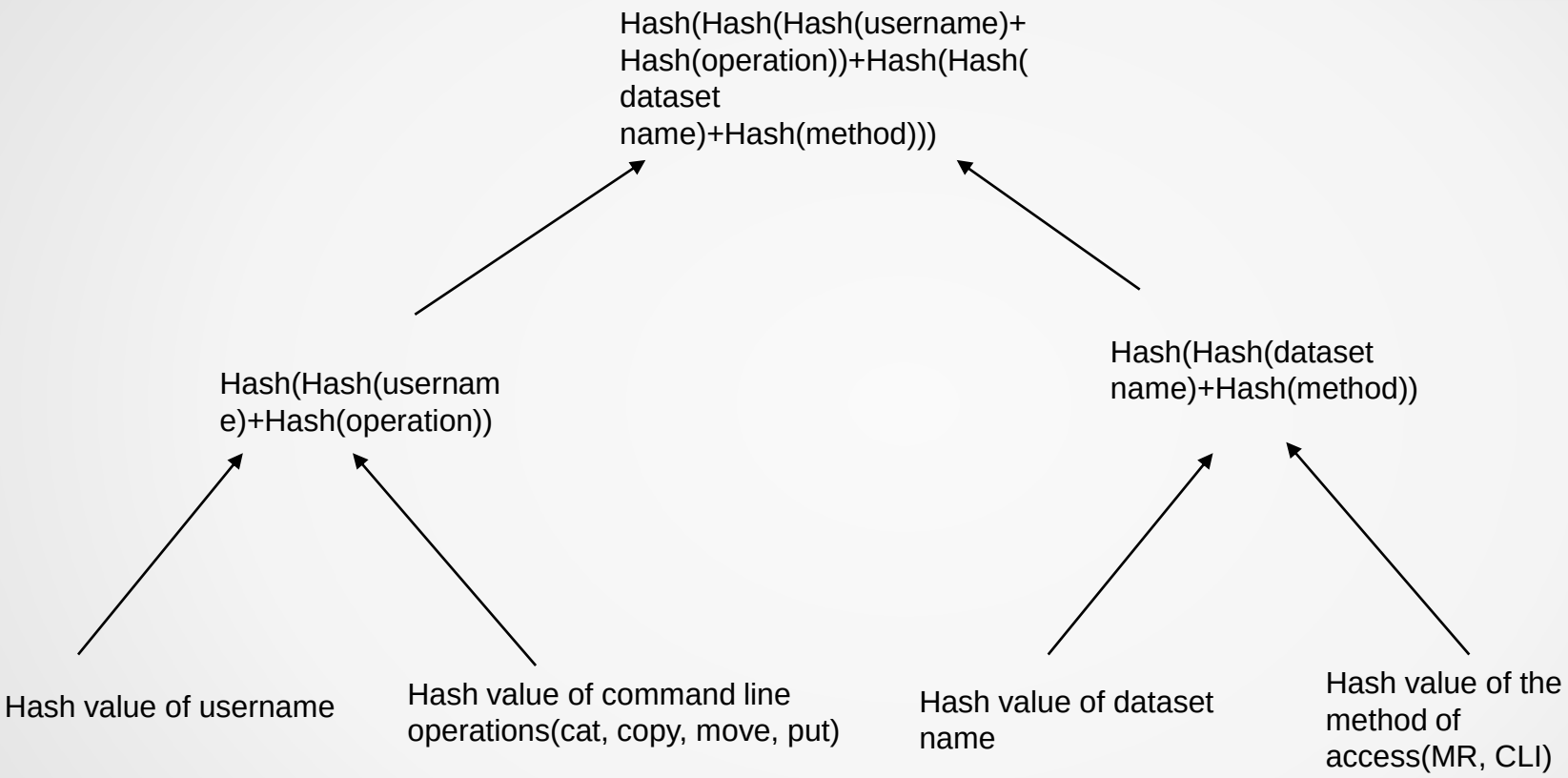
Creation of the first block

- The block header of the genesis block will be used.
- A version number will be calculated based on the version number of the genesis block.
- Timestamp of creation of this block will be captured.
- The username of the user accessing the dataset will be captured.
- The name of the dataset will be acquired.
- The method which was used to access the dataset(Command Line Interface or Map Reduce) will be fetched.
- The command line operation with which the user accessed the dataset (view, copy, move, put).
- Number of bits of the above data(dataset name, username, method and operation) will be calculated.
- Merkle root will be calculated using the above data(dataset name, username, method and operation).

Creation of first block continued

- The version number, the timestamp and the number of bits would be converted into Hexadecimal.
- The hexadecimal values would be converted into Little Endian format. The characters are replaced in pairs from the end of the string to the front of the string.
- These little endian values would be hashed and concatenated together.
- The blockheader of the previous block which is in hashed format would be concatenated with the hash values obtained in the previous step.
- The output of the above step would be concatenated with the merkle root which is a hash value as well.
- The final output of the above entire process would provide the block header of the current block.

Merkle Tree of the data



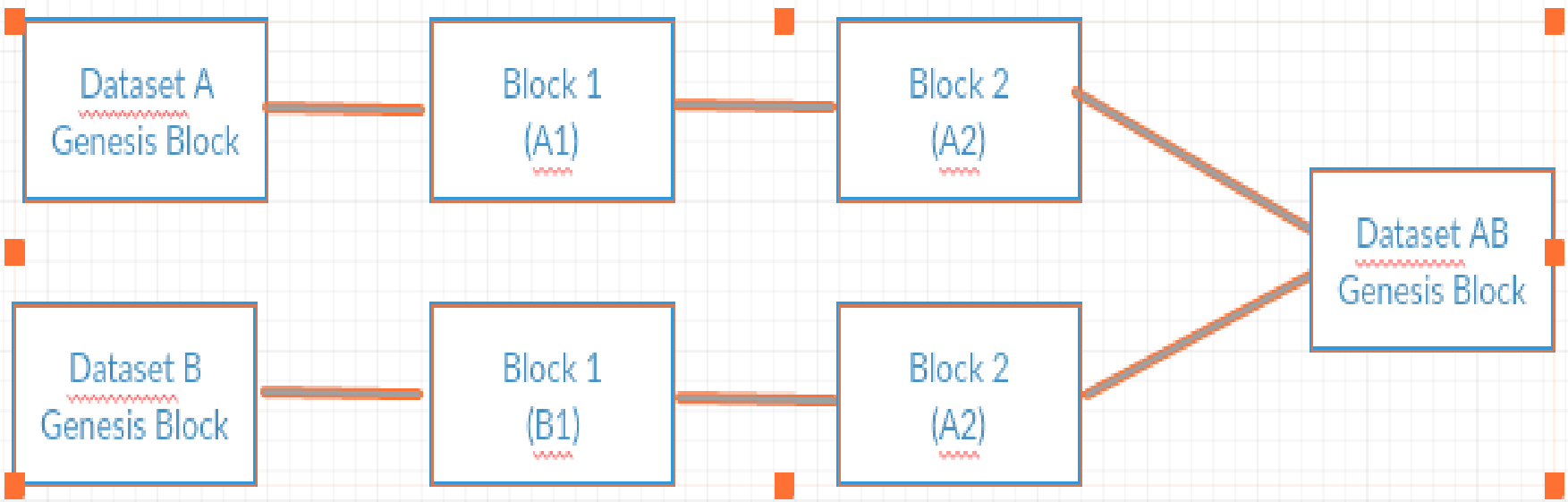
Validation of the genesis block and first block

- Any changes in the Genesis block would change the block header.
- Any change in the version number would change the block header.
- Any change in the data in the block would change the block header.
- Any change in the data in the block would change the bits, thus changing the block header.
- Any change in the current block would be reported as well.

Creation of a new block chain

- The new block chain would be created by combining the two block chains. The block which would be created using the 2 blocks of the other block chains would be the genesis block of the new block chain.
- **The data in this block chain will have the following fields:**
- **Version Number:** Version number would be calculated for the new genesis block.
- **Time stamp:** It would be the time of creation of this new block.
- **Data :** The data in this new block would comprise of the following components:
- **Username:** Username who accessed the two blockchains together.
- **Dataset:** The dataset name would be concatenated using '_' as a delimiter.
- **Type of Access:** CLI or MR depending on the usage.
- **Command Line Operation:** cat, copy, move, put.

New Block Chain



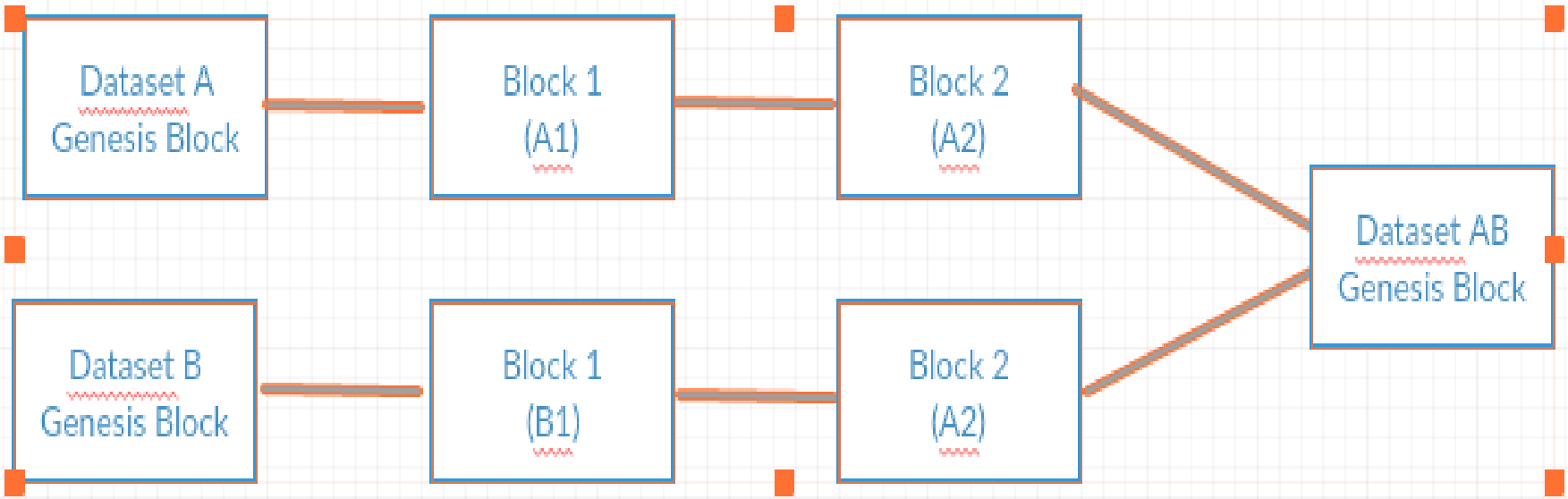
Deletion cases

- Case of deletion of a blockchain or a block in case of corruption.
- Case of deletion of the entire dataset

Deletion due to corruption

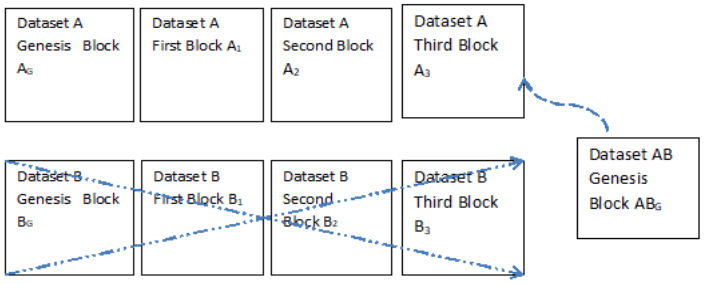
- If an unauthorized user tries to alter any block of the block chain, the block will become corrupt and thus corrupting all the blocks following the block.
- The entire block chain is being replicated across multiple nodes and when the corrupt block has to be deleted, the same block could be copied from the backup of the block chain and inserted again.
- This would prevent any loss of data.

Combination of datasets



Dataset deletion

- If the new block chain is created by combination of 2 existing block chains, it would have all the columns of the two block chains.
- If the dataset B is deleted, it would affect the dataset AB.
- So if dataset B is deleted, the new block chain AB would just become a following block of block chain A



Algorithm for merkle root calculation

Algorithm 2 merkle_root_calc
Input: dataset_name, operation, user_name, method_Of_Access;
output: merkle_root;
1: Compute Hash(dataset_name and operation) = Hash(Hash(dataset_name)+Hash(operation));
2: Compute Hash(user_name and method_Of_access) = Hash(Hash(user_name)+Hash(method_Of_Access));
3: Compute merkle_root = Hash((dataset_name and operation)and(user_name and method_Of_access)) = Hash(Hash(dataset_name and operation)+Hash(user_name and method_Of_access)) ;

Block creation algorithm

Input: version_number, timestamp, merkle_root, prev_block_header, data_size, dataset_name, operation, user_name, method_of_access;
Output: block_header;
1: Compute merkle_root = merkle_root_calc (dataset_name, operation, user_name, method_of_access);
2: Compute block_header=Hash (LittleEndian (Hex (version_number)) LittleEndian (Hex (timestamp)) +LittleEndian (Hex (data_size)) +merkle_root + prev_block_header);
if (blockType==Genesis block)
prev_block_header=0
3: insert to database= Encrypted (dataset_name, operation, user_name, method_of_access), LittleEndian (Hex (version_number), LittleEndian (Hex (timestamp), LittleEndian (Hex (data_size), merkle_root, block_header;

Algorithm to show n blockchain combinations

Input: dataset_name (1 to n), operations, version_number, data_size, timestamp, user_name, method_of_access, prev_block_header (1 to n);
output: block_header;
1: merkle_root= merkle_root_calc ((dataset_name1+dataset_name2+....+ dataset_namen), operations, user_name, method_of_access);
2: block_header = Hash(LittleEndian(Hex(version_number)+LittleEndian(Hex(data_size) + LittleEndian (Hex(timestamp) + merkle_root + Hash(prev_block_header1 +prev_block_header2 +.....prev_block_headern));

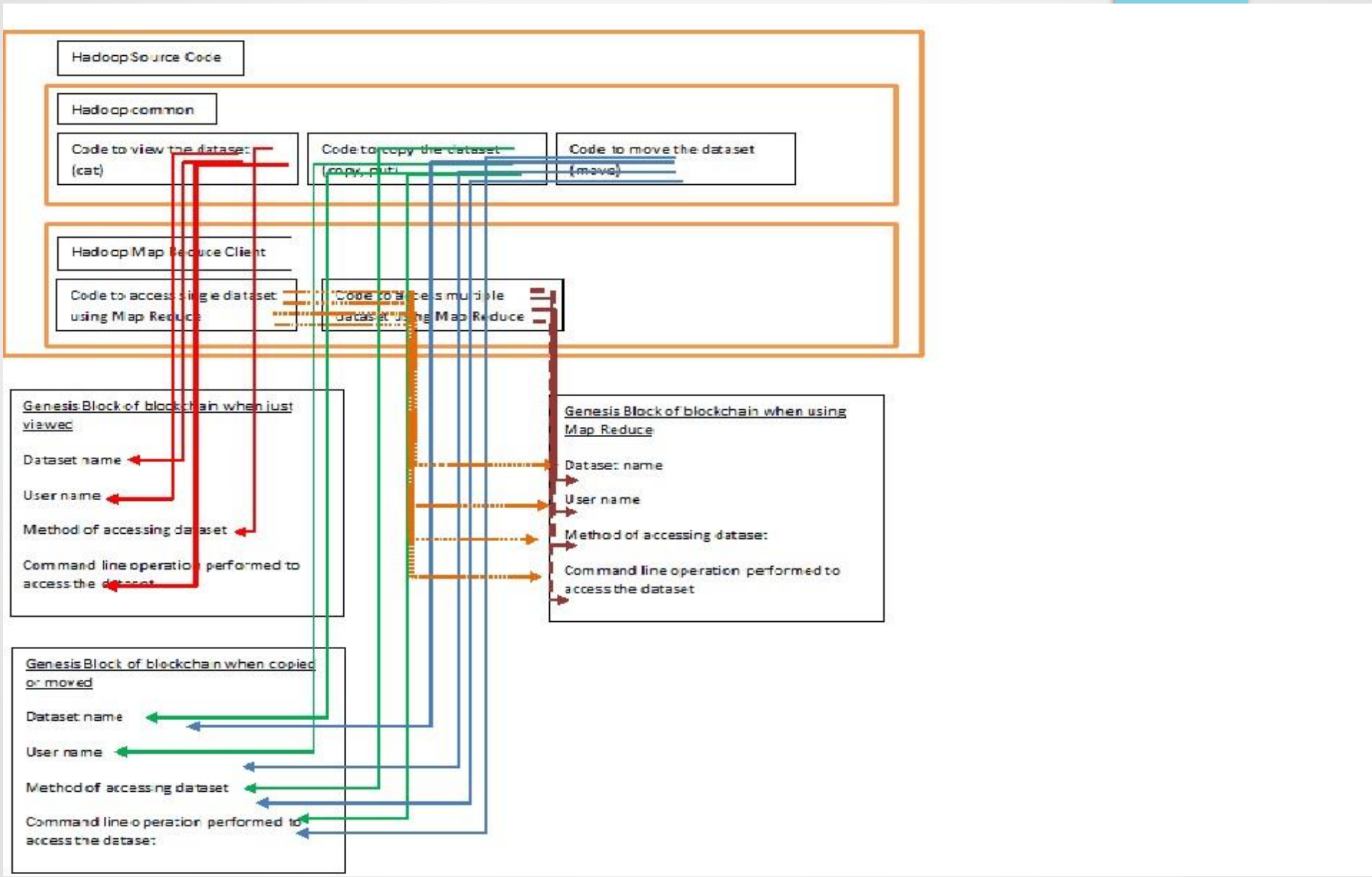
Algorithm to show deletion of dataset

input: dataset_name1, dataset_name2, dataset_name12;
1: Dataset_name12 = dataset_name1+dataset_name2;
2: if dataset_name2 present == dataset_name2 blockchain
then do nothing
else dataset_name2 deleted
dataset_name12 = dataset1;

Algorithm to show validation of blockchain

input(from database): prev_block_header, LittleEndian(Hex(version_number),LittleEndian(Hex(timestamp),LittleEndian(Hex(data_size),merkle_root, block_header;
output: boolean_flag;
1: if (Hash (prev_block_header+LittleEndian (Hex (version_number)
+LittleEndian (Hex (timestamp) +LittleEndian (Hex (data_size)
+merkle_root)) ==block_header) {boolean_flag=true} else {boolean_flag=false};
if (boolean_flag = false) {delete_block}

Implementation



Findings and Results

- The data usage tracker works as expected. When an existing dataset is accessed for the first time, a new blockchain is created. Every time the dataset is accessed again a new block is added to the blockchain.
- The average execution time for creation of Genesis block/Blockchain creation in the AWS EC2 cluster with 4GB RAM and 8GB hard drive system is 4000 milliseconds. 2 processes (HDFS and Hiveserver2) were running simultaneously. The sample for this result is 25 executions. The size of datasets varied between 10000-100000 lines.
- If a dataset is deleted, its corresponding blockchain is deleted as well. The average execution time for deletion of a blockchain in the AWS EC2 cluster with 4GB RAM and 8GB hard drive system is 1000 milliseconds. The sample for this result is 10 executions. The size of datasets varied between 10000-100000 lines.
- At every regular intervals the blockchain is validated for validity. The values are recomputed with the previous values and if there is an anomaly, it is recorded in the report.
- The average execution time for validation of 16 blockchains with an average of 40 blocks in each blockchain in the AWS EC2 cluster with 4GB RAM and 8GB hard drive system is 19500 milliseconds.

Thank you.

Questions?