Information Sciences Letters
An International Journal

http://dx.doi.org/10.18576/isl/120355

Enhancing Map Reduce Computation Integrity on Hybrid Cloud

Walid Atwa[*], *Doaa Abo Aly and Hamdy Mousa*

Computer Science Department, Faculty of Computers and Information, Menoufia University, Menoufia, Egypt

Received: 1 Jan. 2023, Revised: 2 Feb. 2023, Accepted: 19 Feb. 2023.
Published online: 1 Mar. 2023.

Abstract: MapReduce is one of the most popular distributed programming frameworks. However, MapReduce in the public cloud suffers from a lack of confidence in the participating virtual machines. Also, malicious nodes may purposely cheat the processing result during map tasks or reduce tasks. Thus, the results will be unreliable and erroneous. In this paper, we propose a technique which overlays on a hybrid cloud. We run the master and some of the slave workers on a private cloud that is a trusted cloud, and the remaining workers run on a public cloud. Our technique depends on replicating a subset of each task to reduce overhead. When a malicious worker on the public cloud executes a task and an error is detected as a part of replicated subset, we detect and exclude this worker from the cloud. We carry out several theoretical experiments to investigate the security and performance overhead. The results provide high computation integrity and little performance overhead.

Keywords: Big data; MapReduce; Security; Distributed computing.

1 Introduction

Big data is an expression that depicts the large structured and unstructured volume of data collected about our surroundings. Big data is a data set that is characterized by being big, high in variety, and velocity [1]. The advancement of social networks and information technology lead to the fast increase of data with the coming of cloud computing and big data era [2]. Many political and economic interests are existed in big data, particularly the process of data analysis, integration, and data mining [3]. However, big data faces many security risks and privacy-preserving challenges [4]. The traditional security mechanisms are not able to deal with big data security. This is because of the volume, variety, and velocity of big data. One of these challenges is secure computations in distributed programming frameworks (DPFs).

MapReduce is a popular example for DPFs [5]. MapReduce presents large-scale data processing in parallel [6]. On a public cloud or hybrid cloud, it's utilized to make distributed processing of massive amounts of data more efficient and fault-tolerant without any overprice (MapReduce is presented as platform-as-service by cloud) [7]. Without taking into consideration physical infrastructures and installation of software, there are more public clouds (e.g. Google App Engine, Amazon Elastic MapReduce) that enable users to complete computations of MapReduce. MapReduce on both public cloud and hybrid cloud suffers from various security and attacks threats. The users can cost-effectively process big data using MapReduce on public clouds. But MapReduce on public cloud or hybrid cloud faces integrity vulnerability problem. If the public cloud is evaded because of security problems and running everything on private cloud, this will achieve result accuracy but with less economic benefit.

On the public cloud, MapReduce jobs are executed by a cluster containing hundreds of nodes or thousands of nodes for computation. If an impersonation attack dominated any worker on this cluster, it will control this worker and make it tamper with computations. This worker becomes a malicious worker and may generate the wrong results and the total result of computation becomes incorrect. So, it is necessary to check the result integrity of all workers whether mapper or reducer to satisfy the integrity of computations requirement and eliminate malicious adversary. In addition, more applications (e.g., Yahoo!, Google, Facebook, etc.) use MapReduce to process data, so the integrity of the result of MapReduce computation must be ensured.

In this paper, we propose a technique that deploys MapReduce on the hybrid cloud to combine the advantages of both

[*]Corresponding author e-mail: walid.atwa@ci.menofia.edu.eg

public and private clouds. The private cloud contains a small number of workers known as verifiers and the master, while other workers and Distributed File System (DFS) are deployed on the public cloud. The verifiers, master, and DFS are all trustworthy, but the workers aren't. We describe our technique details of the two phases: map phase integrity check and reduce phase integrity check. In map phase, each map task executes on the public cloud but on the private cloud part of each map task input (called a subset records) will be replicated to verify each map task result. In the reduce phase, we replicate all reduce tasks but, part of reduce tasks will be replicated on the private cloud and the rest of reduce tasks will be replicated on trusted node on public cloud.

We explain the performance overhead and quantitative analyzes on the security of the proposed technique. Also, the ideal value for the subset record ratio that produces the lowest error number within the stated security limits is determined through the security analysis. We implement our technique based on Apache Hadoop MapReduce. We illustrate that the proposed technique is an effective framework for ensuring high computation integrity.

The rest of the paper is arranged as follows. In sections 2, explain MapReduce model. Related work is discussed by section 3. In sections 4 and 5, the proposed technique and its evaluation will be explained respectively. Finally, Section 6 shows the conclusion of the paper.

2 Map Reduce Frameworks

MapReduce presents parallel processing using computing nodes cluster over large-scale data as shown in Fig.1. Three entities are components of MapReduce [8]: master, workers, and distributed file system (DFS). Data blocks are used to store data in a distributed file system (DFS). The master uses staff members to illustrate load balancing, task scheduling, and job management. While the workers are compute resources that complete the tasks assigned by the master.

MapReduce contains two phases: 1) a map phase, in this phase; parallel processing is done by distributing input data to different distributed workers. 2) Reduce phase will be collected the intermediate results together. The master takes the job of MapReduce from users. This job's input text files will be put in DFS in the data blocks form. This job is partitioned into several map and reduce tasks. The number of map tasks depends on how many data blocks are included in the input text files. Only one data block will be taken by each map task as its input.

In the map phase: the map task assignment is sent by the master to the mapper and the data block is read by the mapper from DFS. Then mapper processes the data block and stores its intermediate result in its local storage. Each mapper generates an intermediate result that is partitioned into *k* partitions $p_1, p_2,..., p_k$ by partitioning function. Reduce tasks number is equal to partitions number *k*. In reduce phase, the master will send a notification when a map task is completed. When each mapper has finished its map operation, the reducer will read the intermediate result. Then, the reducer processes its partition that is read from the intermediate result. Finally, each reducer result will be written to the DFS.

3 Related Works

The importance of MapReduce lies in its simple architecture and capability for parallel computation for data-intensive computation in a variety of applications and research domains. Some researchers are interested in how to apply MapReduce to certain application domains to solve problems. Other researches (that will be mentioned in this paper) are interested in computation integrity protection for MapReduce.

To address computation integrity problems for MapReduce, some solutions are proposed such as replication sampling, and verification techniques. Some of these techniques achieve high result integrity but incur high performance overhead and other techniques achieve low-performance overhead but they are more vulnerable to attack.

Secure MapReduce (Secure MR) [8] proposes MapReduce service integrity and prevents replay and denial of service attacks. It adds a set of security components for MapReduce to check the result integrity of map/reduce tasks. This technique imposes small performance overhead while achieving data processing service integrity but it is not able to find collusive malicious workers. Cross Cloud MapReduce (CCMR) [9] offers combining the features of private and public clouds, and it employs the MapReduce framework on a hybrid cloud with single private cloud and single public cloud. It implements three kinds of tasks (original task, replication task, verification task) on both map and reduce phases to check the result integrity of MapReduce. This technique ensures high result integrity but it causes non-negligible performance overhead.

Integrity MR [10] is the same architecture of CCMR but it contains single private cloud and multiple public clouds. It also has the same kinds of tasks on CCMR (original task, replication task, verification task) but in this technique, the

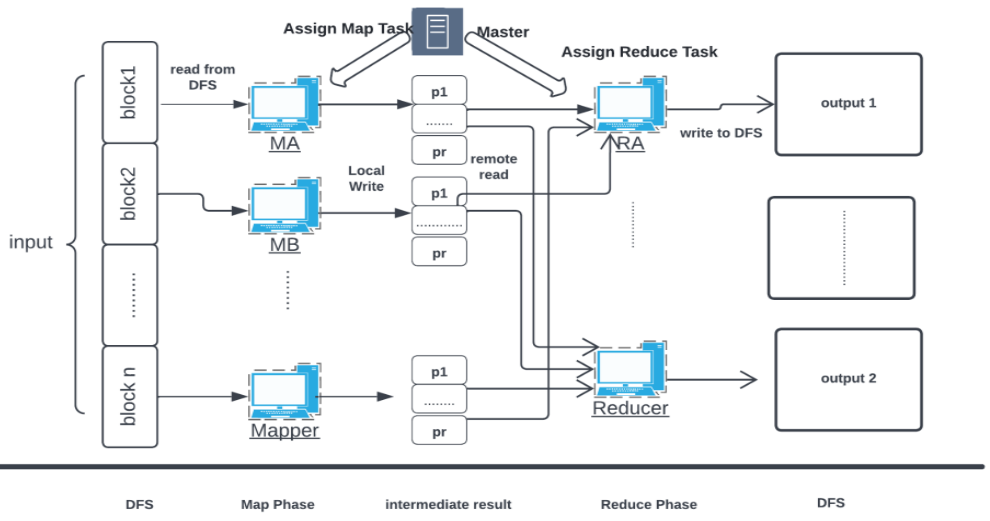

Fig. 1: The MapReduce data processing reference model.

original task is executed on public cloud, the replication task is run on another public cloud and the verification task is run on the private cloud but, on CCMR the original task and the replication task are run on one public cloud. This technique achieves high integrity with a non-negligible performance overhead. Accountable MapReduce [11] will detect malicious workers through the master submits the input data and output of each mapper and reducer to auditor group that their members will repeat the input data processing and match its output with the original one for inconsistency check.

In TrustMR [12] both map and reduce tasks are divided into smaller computation pieces. To ensure the result integrity of each task, TrustMR depends on replicating a subset of pieces of computation two times. This technique achieves a high detection rate by decreasing the overhead of replication but a map verifier that checks result integrity of intermediate results may be malicious and provide wrong input to the reducer. Result verification mechanism is proposed based on the value of reputation-threshold using voting method [13]. This technique is based on the reputation of a worker whereas the result that is received from a worker is true if the reputation of this worker surpasses the minimum worker score ($R(P_1) > W_S$).

MtMR [14] applies on a hybrid cloud that contains of single private cloud (consisting of master and verifiers) and one public cloud (consisting of DFS and slave workers). In MTMR, the checker (the private cloud) wants to check if the result executed by the prover (the public) is correct or not. In Credibility-Based Result Verification [15], the credibility of nodes is a factor for result verification whereas the result with the highest level of credibility is picked above the one with the lowest level of credibility.

4 Proposed Technique

4.1 System Overview and Architecture

As illustrated in Fig.2, Our solution performs MapReduce on a hybrid cloud that combines single private cloud and single public cloud. The master node and a few slave nodes are operated by the secure private cloud (referred to as verifiers). The Distributed File System (DFS) is scattered throughout the public cloud, as are other slave nodes, also referred to as workers. The DFS integrity can be ensured by the techniques proposed in Ref [16, 17]. The verifiers, master, and DFS are all trustworthy, but the workers aren’t.

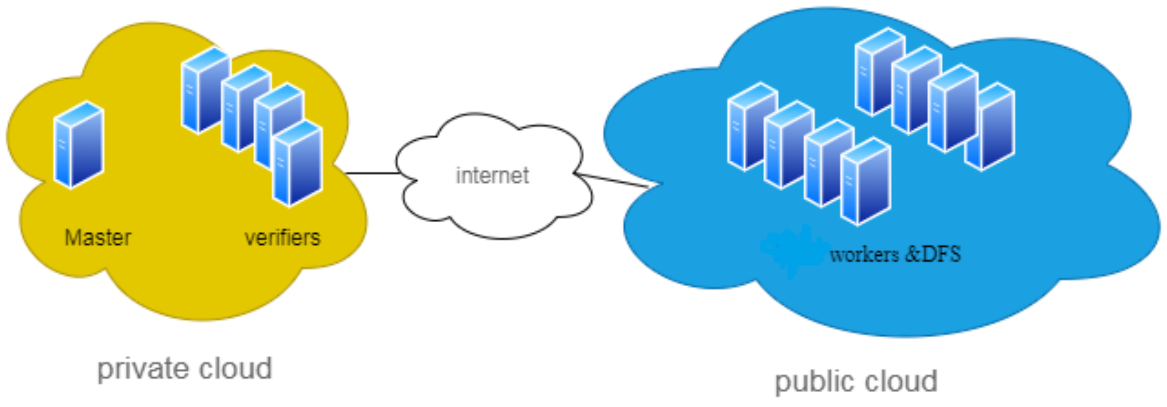

Fig. 2: Architecture of our technique.

Our technique presents two kinds of tasks in both the map and reduce phases: the original task and the verification task. The workers in the public cloud execute the original task. While the verifier on the private cloud executes the verification task. Because the original task cannot be trusted, the verification task only repeats a portion of the original task to check the result returned by the original task. In a map task, each repeated portion is termed a subset, however in the reduce phase, it's called reduce sampled records. Our technique performs one-layer check on each returned original task result in both the map and reduce phases: verification. The worker just sends the result's hash value to reduce the communication cost.

4.2 Map Phase Integrity Check

In our technique, we break map phase integrity check into four steps: the original, replication, request, and response step.

In the original step, outputs of each map task are a group of key-value pairs. The hash value (h_1) of each pair and the hash value (H) of the concatenation of these hash values for each map task will be calculated. H of each map task will be sent to the master. In the replication step, our technique applies one-layer check on each returned original map task result. In this layer, our technique replicates the execution of a subset of each map task input by verifier on the private cloud. In the request step, the master sends to the public cloud the hash values of the output of the execution of subsets to request from all map tasks on the public cloud the complementary hash values for these sent hash values. In the response step, all map tasks on public cloud response and send the complementary hash values to the verifier (on the private cloud). Then, the private cloud calculates the hash value of concatenation of complementary hash values and the hash value(s) of the output of subset execution for a map task as H^* and sends H^* to the master. Then, the master compares H, H^*. If H equals H^*, the results of these workers (mappers) are accepted.

Map tasks are carried out on the public cloud during the map phase. The output is a list of <*key, value*> pairs for each map task. The protocol between the private cloud and one map task (i.e., the public cloud) is shown in fig.3. First, the map task calculates the hash value for each key value pair of its output then computes (H) and sends it to master (on the private cloud). (H) is defined in equation (1). Second, the private cloud runs the map task on verifiers with the subset of map input $<k_2,v_2>$ and produces key value pair $<k'_2,v'_2>$ and its hash value h'_2. Third, Verifier requests complementary hash values for subset (h'_2) from the map task on the public cloud. After the map task receives a request, it sends a response (h_1,h_3) to prove the correctness of the computation. Finally, the verifier concatenates the hash value of subset (h'_2) and complementary hash values (h_1,h_3) then, computes the hash value of this concatenation as H^* then, H^* will be sent to the master that checks if H=H^* or not.

$$H = \text{hash value of } (h1 + h2 + h3) \qquad (1)$$

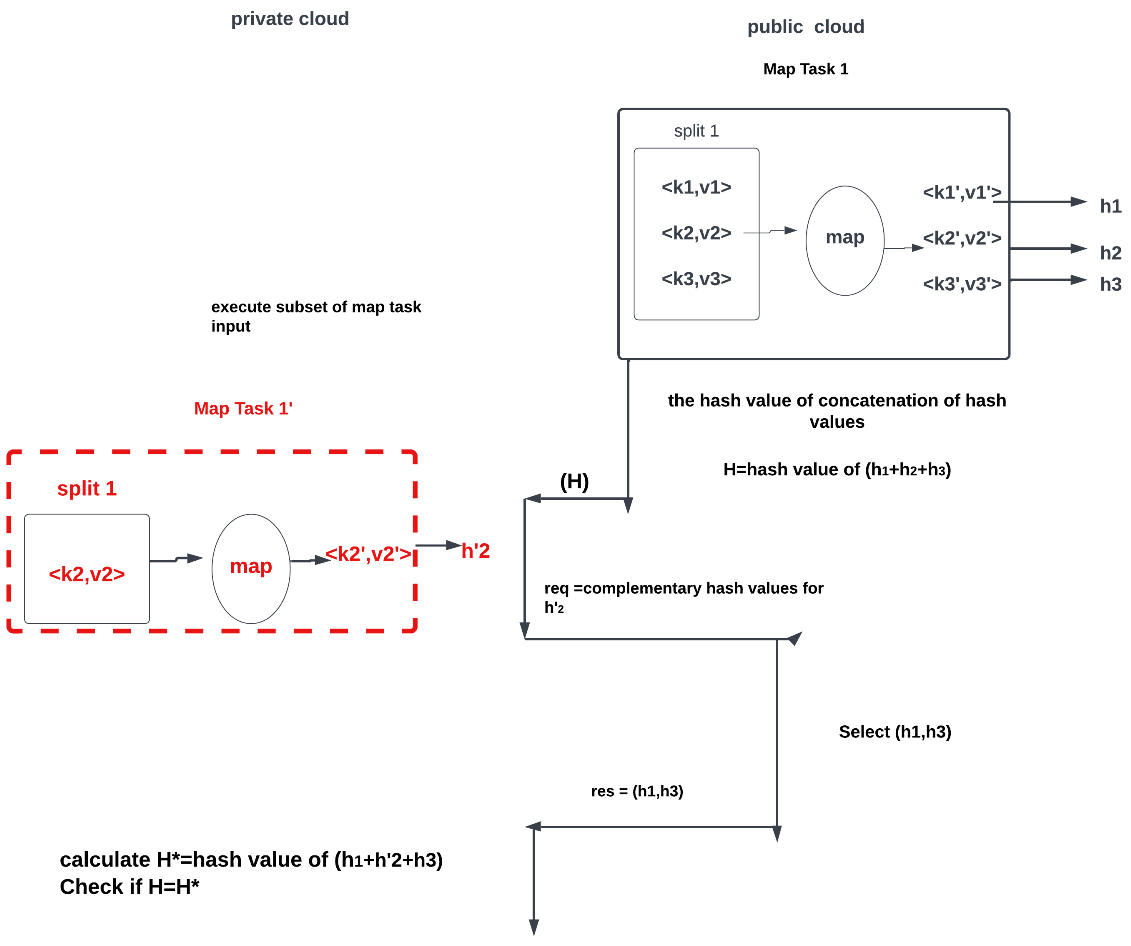

Fig. 3: The Protocol of Map Phase Integrity Check.

4.3 Reduce Phase Integrity Check

We break reduce phase integrity check into five steps: the original, request, response, verification, and rest verification step.

In the original step, reduce tasks normally begin their works on the public cloud whereas output records produced by map tasks are the input of all reduce tasks. After each reduce task finishes, the hash value of its output will be sent to the master as H_i. In the request step, after the private cloud verifies from all map tasks, it will take some distinct keys records (as reduce sampled records) that are outputs of re-computation of subsets of map tasks input on the private cloud. Then, the private cloud requests from output of all map tasks on the public cloud all records that belong to these distinct keys. Whereas all map tasks are verified from their outputs in map phase integrity check. In the response step, the public cloud responds and sends all records that belong to the received distinct keys.

In the verification step, these records that are received in the response step will use as input of reduce tasks that are executed and computed the hash values of their outputs as H'_i by verifier on the private cloud whereas these reduce tasks executed on private cloud are a sample of original reduce tasks to check reduce tasks integrity (i.e. some of reduce tasks on the public cloud will be taken as sample to execute on private cloud to check reduce phase integrity). These hash values of outputs that are computed in this step will be sent to the master. The master selects from the received hash values of reduce tasks in the original step the hash values (H_i) that belong to the reduce tasks of the same selected distinct keys in the request step on the private cloud. Then, the master compares H_i, H'_i. In the rest verification step, if H_i equals to H'_i, the nodes that execute these reduce tasks on the public cloud are trusted so, the rest of the reduce tasks on the public cloud that are not checked on the private cloud will be replicated their execution on these nodes to reduces communication overhead and will be computed the hash values of their output. Then, these hash values will be sent to the master. The master compares these hash values with the hash values of the original reduce tasks that belong to the same keys.

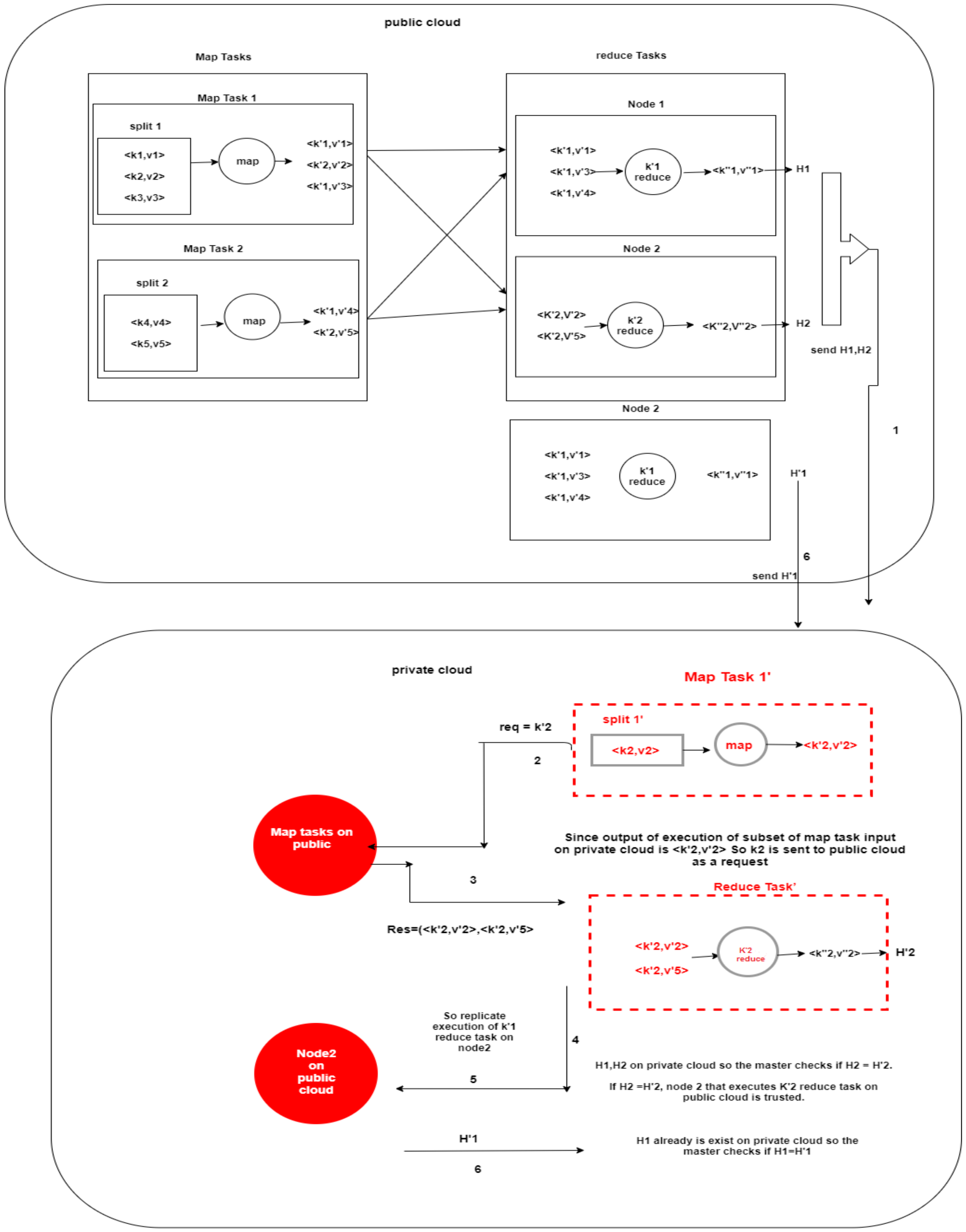

Fig. 4: The Protocol of reduce Phase Integrity Check.

Fig.4.shows the connectivity between the private and public clouds in reduce phase integrity check. First, all reduce tasks on the public cloud are normally executed whereas all outputs of all map tasks that have been verified on map phase integrity check are the inputs of reduce tasks whereas ($<k'_1,v'_1>,<k'_1,v'_3>,<k'_1,v'_4>$) belong k'_1 reduce tasks and($<k'_2,v'_2>,<k'_2,v'_5>$) belong k'_2 reduce tasks. These reduce tasks generate their outputs ($<k''_1,v''_1>,<k''_2,v''_2>$), compute the hash values of these outputs as H_1, H_2 and send them to the master. Then, since the output of the subset of map task on private cloud $<k'_2,v'_2>$, the master will take k_2 and send it as a request to the public cloud. On the public cloud, all records that belong to k2 are selected from all output of all map tasks ($<k'_2,v'_2>,<k'_2,v'_5>$) and sends these records to the private cloud in response.

After the private cloud receives these records, it uses these records as input of reduce task (k'_2 reduce) that is executed on verifier as checked sample of original reduce tasks. After k'_2 reduce task finishes, it computes the hash value of its output $<k''_2,v''_2>$ as H'_2 and sends this hash value to the master. Then, the master checks if H_2 equals H'_2 or not. If H_2 equals H'_2, node2 that executes k'_2 reduce task is trusted so, k'_1 reduce task that is on public cloud and not checked will be replicated its execution on this node. The hash value of the output of k'_1 reduce task will be sent as H'_1 to the master. The master checks if H_1 equals H'_1 or not. therefore, we check all reduce tasks whereas part of these tasks checks on the private cloud and the rest of reduce tasks on trusted nodes on the public cloud.

5 System Evaluations

The security and the performance overhead are two aspects that our analysis is based on them. They will be discussed.

5.1 Security Analysis

The quantitative and the qualitative analysis are performed on the security of the system. In the qualitative analysis, we show that a semi-honest worker is not computationally able to cheat safely. Error detection probability and the number of predicted incorrect records are analyzed through the quantitative analysis if a cheating by a semi-honest worker occurs with a certain probability. The relationship is also modelled between the expected security guarantee and the optimal parameter value.

5.1.1 Qualitative Analysis

For each map task, only some records from the public cloud can be sampled and checked in the private cloud because It's challenging for the private cloud to gather all input records for each map task from the public cloud. In our technique, if the malicious worker returns wrong results when the input records that correspond to these wrong results are taken as a sample, it has to guarantee that the sample processing can produce the same wrong results. But, our following proofs illustrate that it is computationally impossible.

Theorem 1. Assume a malicious worker commits a cheat output the records set S', while the correct records set is S. If a record $p_{si}\in(S-S')$, an input is chosen as a subset of map input is replicated its execution by verifier on the private cloud in our technique, such as a cheat can be detected through the map phase integrity check.

Proof. Suppose S is set of the correct output records for a map task, yet a malicious worker commits a cheat and outputs wrong output records set S' where $S'\neq S$. In the original step, the master get H of S' = hash value of $\|^{n}_{i=1} h_i$, where h is the hash value of each output record (key value pair) on this mapper and n is the number of output records that this mapper generates. Suppose the master chooses the input record of output record $p_{si}\in(S-S')$ as a subset of map input that is replicated its execution by verifier on private cloud, the master thus sends the hash value of p_{si} in a request req=[h_i] to the original map task on public cloud. To surpass the verification, the map task must return complementary hash values for the hash value of psi in the response res, so that

$$H\ o\,f\ S' = \textit{hash value for (the hash value of } p_{si} + \textit{Complementary hash values for the hash value of } p_{si})$$

If $p_{si}\notin S'$, there is no hash value of p_{si}. Therefore it is computationally infeasible to find complementary hash values for the hash value of p_{si}. Thus, the hash value for (the hash value of p_{si} + Complementary hash values for the hash value of p_{si}) = H. As a result, the malicious worker cannot evade the detection.

The worker has no way of knowing which record will be chosen as a subset of map input that is replicated its execution by verifier on the private cloud because the original step is executed before the request step. Thus, to safely pass checks in the map phase integrity check, the best method for a worker is to work rightly

Theorem 2. If any reduce task cheats in its result, this cheat can be detected by the reduce phase integrity check.

Proof. Suppose r_1,r_2,r_3,r_4 are all reduce tasks whereas each reduce task belong one key. If reduce sampled records (output of re-execution of subsets of map tasks on private cloud) are belong r_1,r_2 reduce tasks, these reduce tasks of r_1,r_2 will be replicated on private cloud so if these reduce tasks that are referred as r_1,r_2 cheat in their outputs, the reduce phase integrity check phase can detect such a cheat. After the nodes that executes r_1,r_2 reduce tasks on public cloud are verified(because they are replicated on private cloud), the rest of reduce tasks (r_3,r_4) will be replicated on public cloud on these nodes because they become trusted so also if these reduce tasks that are referred as r_3,r_4 cheat in their outputs, the reduce phase integrity check can detect such a cheat.

5.1.2 Quantitative Analysis

Malicious workers may cheat at random in the hopes of avoiding detection. To analyze this case, we perform quantitative studies. Measurement metrics and the parameters for quantifying system security are defined in Table 1.

Table 1: Metrics and Parameters of Quantifying System Security

Notation	Definition
N	The count of input records in a map task
N_r	The count of all reduce tasks
M_I	The count of input records for all map tasks
M_O	The count of output records for all map tasks
S_m	The ratio of records number of all subsets of all map tasks input to M_I
S_r	The ratio of reduced sampled records to M_O
R_O	The number of output records of all reduce tasks
α	The average number of input records in a reduce task.
c_m	The worker probability to cheat whenever it executes a map task input record.
c_r	The worker probability to cheat whenever it executes a reduce task input records or when it generates a reduce task output.
w_h	The workload of a single hash function invocation.
w_t	The workload of processing or generating single input record.
s_h	The size of one hash value.
s_k	The key size of a task input/output record.
w_r	The workload associated with accessing single record in the map task input/output records based on its position.
s_r	The length of a single input/output record in a map task.

We show the analysis results on both map phase integrity check and reduce phase integrity check in Theorem 3 and 4 respectively

Theorem 3. In a map task with N input records. If subset records ratio for this map task is S_m (whereas S_m is here the ratio of records number of the subset of a map task input to input records number N in a map task). Then, we can compute the map phase detection probability (D_m) and map phase error number (E_m) as follow:

$$Dm = 1 - (1 - CmSm)^N \quad (2)$$

$$Em = Cm(1 - Sm)\, N(1 - CmSm)^{N-1} \quad (3)$$

The results for reduce phase are similar to the map phase. Thus, in reduce phase with N_r reduce tasks, the reduce phase detection probability (D_r) and reduce phase error number (E_r) are computed as follow:

$$Dr = 1 - (1 - CrSr)^{Nr} \tag{4}$$

$$Er = Cr(1 - Sr)Nr(1 - CrSr)^{Nr-1} \tag{5}$$

In our technique, we assume the ratio of reduced sampled records S_r equal 1. So, D_r and E_r are computed as follow:

$$Dr = 1 - (1 - Cr)^{Nr} \tag{6}$$

$$Er = 0 \tag{7}$$

Decreasing the number of incorrect records to a small value is very beneficial because completely removing the incorrect records is very difficult. So it is useful to search for parameters of the system to assure certain security guarantee. In our technique, in map phase integrity check our analysis (shown below) refers that Sm has an impact on the security guarantees in terms of map phase integrity check error number (i.e., E_m). Our performance analysis for map phase integrity check in Section 5.2 refers that Sm determines overhead of the performance of the map phase. Therefore, finding optimal values for Sm can give a perfect trade-off between performance overhead and the security guarantee. The result of Theorem 3 is used to guide our analysis for optimal values.

In Table 2, we present the numerical analysis of map phase error number E_m with different records ratio (i.e., S_m = 0.01, 0.02, and 0.03, respectively) and cheat probabilities $C_m \in [0, 0.1]$. Since E_{ms} under various S_{ms} are all near to 0 when C_m is bigger than 0.1.

We also present a graphical analysis to explain the map phase integrity check error number E_m under various map subset records ratios Sm. The maximal map phase integrity check error number, marked as E_{mmax} will reduce because of increasing the map subset records ratio Sm. For example, as shown in Fig. 5, when Sm rises from 0.01 to 0.03, E_{mmax} reduces from 36 to 12. Also, from Fig. 5, we can notes that the records ratio S_m effect the value of E_{mmax}. Thus, we need to find the minimal map subset records ratio S_{mmin} for maximal map phase integrity check error number E_{mmax}.

$$Smmin = \begin{cases} \dfrac{(1-1/N)^{N-1}}{\text{Emmax}+\left(1-\frac{1}{N}\right)^{N-1}}, & if\ \dfrac{N(1-1/N)^{N-1}}{\text{Emmax}+\ (1-1/N)^{N-1}} \geq 1 \\ 1 - \sqrt[N]{\text{Emmax}/N}, & if\ \dfrac{N(1-1/N)^{N-1}}{\text{Emmax}+\ (1-1/N)^{N-1}} < 1 \end{cases} \tag{8}$$

Table 2: Numerical Analysis of E_m with different S_m and C_m.

S_m=0.01		S_m=0.02		S_m=0.03	
C_m	E_m	C_m	E_m	C_m	E_m
0	0	0	0	0	0
0.004	26.54552328	0.004	17.6145408974	0.004	11.6868963966
0.01	36.4218857854	0.01	13.26285767	0.01	4.828621067
0.02	26.7963859022	0.02	3.588429085	0.02	0.480300955
0.04	7.2500914159	0.04	0.131185943	0.04	0.002369682
0.06	1.4706122036	0.06	0.003591167	0.06	0.0000087370
0.08	0.2650491507	0.08	0.0000872439	0.08	0.0000000285
0.1	0.0447663789	0.1	0.0000019838	0.1	0.0000000001

Table 3: The Minimal Map Subset Records Ratios S_{mmin}.

E_{max}=10		E_{max}=5		E_{max}=3		E_{max}=1	
N	S_{mmin}	***N***	S_{mmin}	***N***	S_{mmin}	***N***	S_{mmin}
100	0.0356547	100	0.0688544	100	0.1097209	100	0.2699289
20000	0.0354835	20000	0.0685351	20000	0.1092342	20000	0.2689463
40000	0.0354830	40000	0.0685343	40000	0.1092330	40000	0.2689439
60000	0.0354829	60000	0.0685340	60000	0.1092326	60000	0.2689431
80000	0.0354828	80000	0.0685339	80000	0.1092324	80000	0.2689427
100000	0.0354828	100000	0.0685338	100000	0.1092323	100000	0.2689424

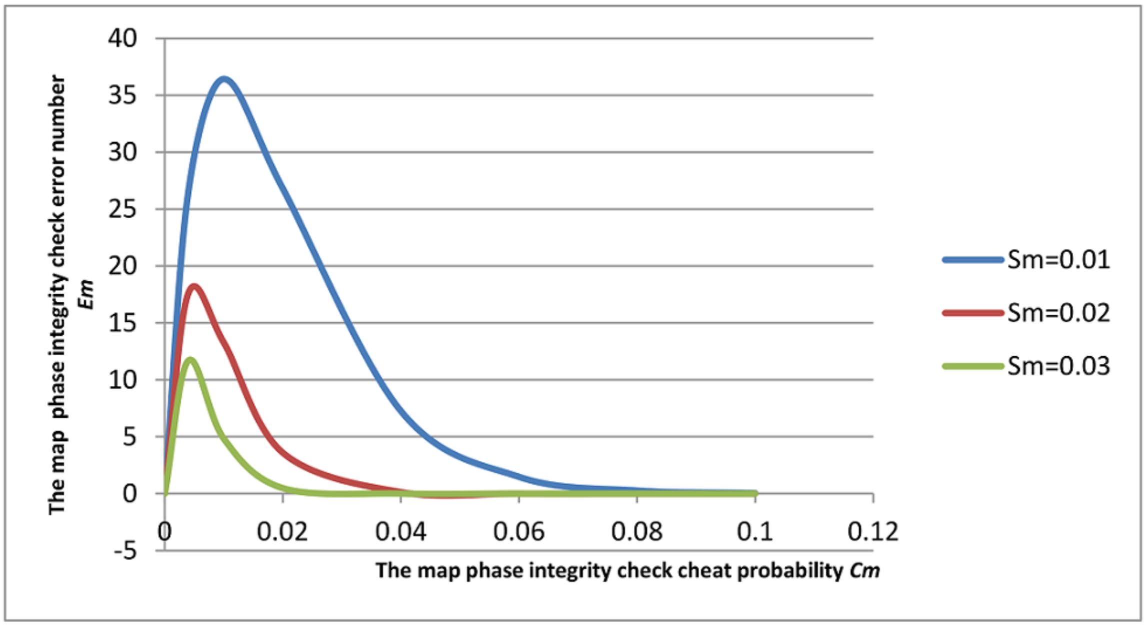

Fig. 5: Graphical Analysis of E_m with different Sm and c_m.

Fig. 6, 7, and 8 show the minimal map subset records ratios/numbers under various requirements of system, including the number of input records and the maximal map phase integrity check error number. As shown in Fig 6 and Table 3, our technique can get the minimal map subset records ratios with various number of input records under different maximal map phase integrity check error number requirements. It also refers that a lower value of E_{mmax} needs a higher S_{mmin}. For example, when E_{mmax} are set to 1, 3, 5, and 10 the minimal map subset records ratio S_{mmins} are 0.27, 0.11, 0.07. and 0.04 respectively. Thus, when 4 % of map input records is taken as a sample, it can ensure that no more than 10 wrong output records will be put to the map output and also sampling 27% of input records of map can ensure that no more than 1 wrong output record will be put to the map output.

Under the same E_{mmax}, we also find that when the number of input records N raises, the minimal map subset records ratio S_{mmin} actually decreases slowly. As shown in Fig.7, when E_{mmax} is set to 10, the minimal map subset records ratio reduces from 0.0354835 to 0.0354828 when N increases from 20000 to 100000. Now, we can compute the minimal map subset records number (i.e. SN_{mmin}=S_{mmin}·N) as explained in Table 4. Fig. 8 displays minimal map subset records numbers under various E_{mmaxs} when N raises from 100 to 100000. The minimal map subset records number is proportional to the number of input record N. For example, at E_{mmax} is 10, when the number of input records grows from 100 to 100000 the minimal map subset records number increases from 4 to 3548. At E_{mmax} is 5 when N grows from 100 to 100000, the minimal map subset records number grows from 7 to 6853.

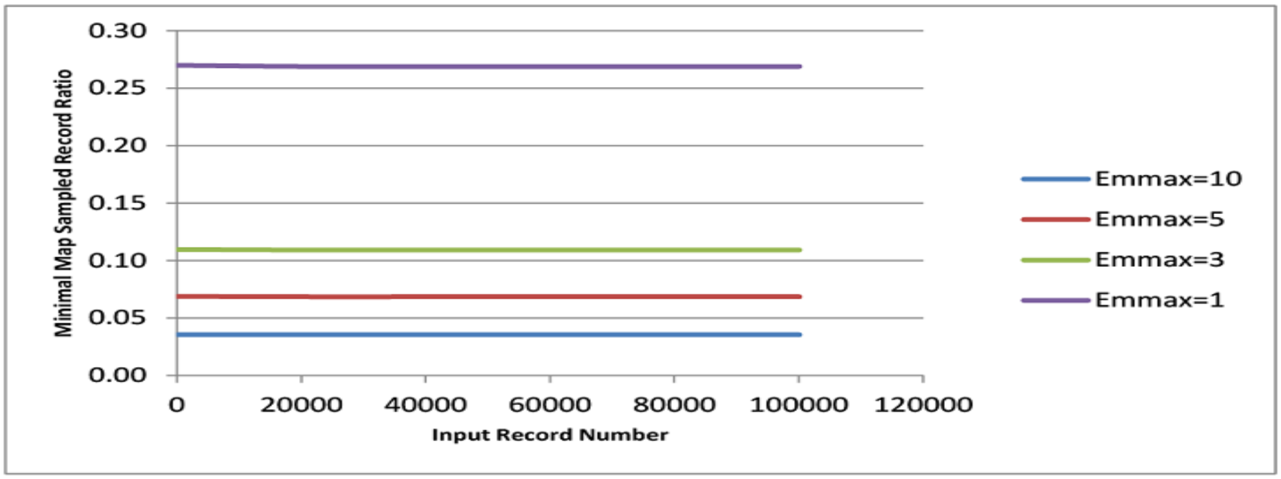

Fig. 6: S_{mmin} with different E_{mmax} and N.

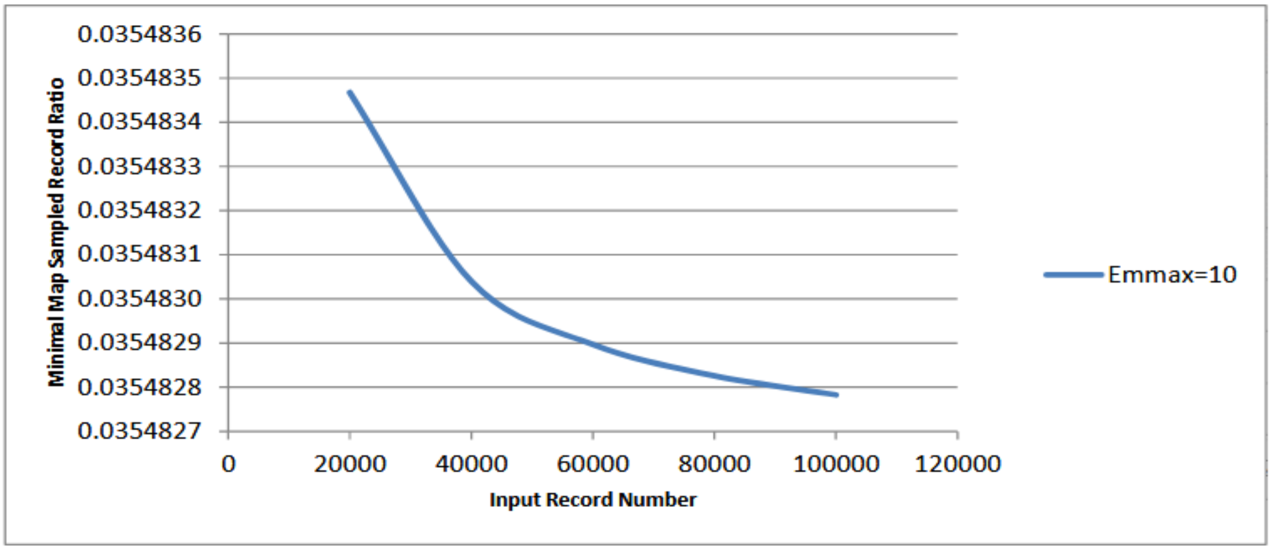

Fig. 7: S_{mmin} with E_{mmax} =10.

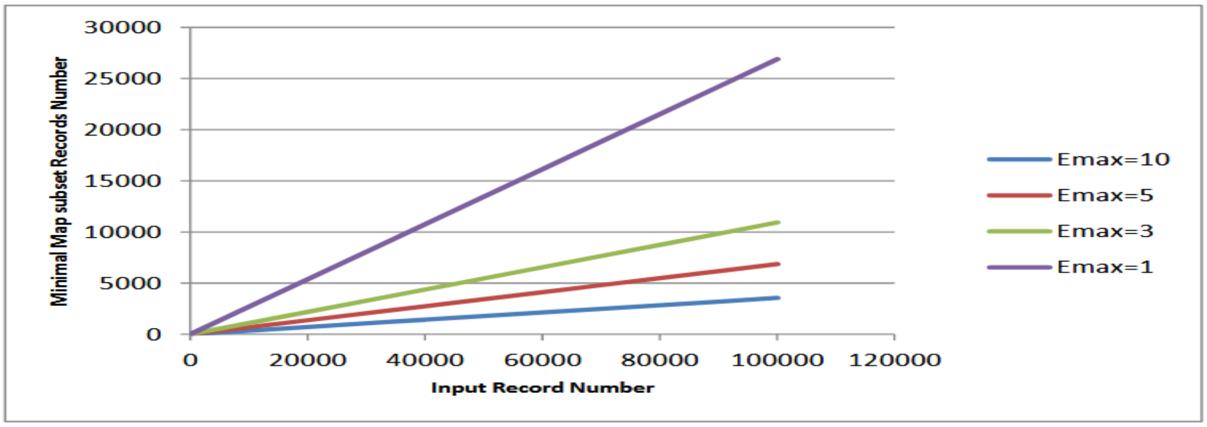

Fig. 8: SN_{mmin} with different E_{mmax} and N.

Table 4: The Minimal Map Subset Records Number SN_{mmin}

E_{max}=10		E_{max}=5		E_{max}=3		E_{max}=1	
N	*SNmmin*	*N*	*SNmmin*	*N*	*SNmmin*	*N*	*SNmmin*
100	3.56547	100	6.885442	100	10.97209	100	26.99289
20000	709.6693	20000	1370.701	20000	2184.684	20000	5378.927
40000	1419.322	40000	2741.371	40000	4369.32	40000	10757.76
60000	2128.974	60000	4112.041	60000	6553.955	60000	16136.58
80000	2838.626	80000	5482.71	80000	8738.59	80000	21515.41
100000	3548.278	100000	6853.38	100000	10923.23	100000	26894.24

5.2 Performance Analysis

The performance analysis is performed on both all map tasks and all reduce tasks but their overheads are analyzed separately. Both reduce task's performance and map task's performance are dependent on a few factors, which are found in Table 1. In the analysis we assume the following parameter w_n, α, w_t , w_r, s_r, s_k, w_h, and s_h are constants. We show the analysis details for map phase integrity check and reduce phase integrity check.

In map phase integrity check, we explain the details of analysis for the four steps explained before. In the original step, each map task on the public cloud generates output as a set of key value pairs. For each map task we need to generate a hash value for each key value pair. The workload of all map tasks for generating a hash value for each key value pair is w_hM_o. Thus, the time complexity in this step is O(M_o) and the communication overhead is O(1) as each map task send the hash value *H* to the master.

In the replication step, the private cloud has to retrieve a subset of each map task input from the public cloud. Therefore, the communication overhead is $sr\ S_m\ M_I$ = O($S_m\ M_I$). Then, the verifiers will replicate the execution of each subset of each map task input. We suppose the computation workload of these replications is proportional to subsets records number of all map tasks input that are retrieved from the public cloud. Thus, the computation overhead for the verifiers will be $w_t\ S_mM_I$ = O($S_m\ M_I$). After the verifiers finish these replications computation, they will generate hash values for all key value pairs that are outputs of these replications computation. Since outputs records number that verifiers need to generate from them hash values are proportional to records number of subsets of map tasks input so the overhead of generating the hash values from these outputs that are on verifier is $w_hS_m\ M_I$ = O($S_m\ M_I$). So the total overhead on private cloud $S_mM_I\ (w_t +w_h)$ = O($S_m\ M_I$).

In the request step, the master will send all outputs records hash values of subsets of map tasks input. The communication overhead in this step is $s_hS_mM_I$ = O(S_mM_I). As we mentioned before, the verifiers that need to generate the hash values from outputs records number is proportional to records number of subsets of map tasks input.

In the response step, upon receiving the request from the private cloud, each map task should locate the complementary hash values for the hash values that is received from the request step. Then, the complementary hash values are sent to the private cloud. As a result, in order to find all complementary hash values, search through the outputs of all map tasks. The number of all map tasks output records hash values equal to the number of all map tasks output records M_o (i.e. each record (key value pair)has one hash value). Thus, locating the complementary hash values for one record of subsets output takes O(M_o). Since outputs records number of subsets execution on verifier is proportional to records number of subsets of map tasks input (S_mM_I) records. Then, locating the complementary hash values for all output records of all subsets execution will incur a computation overhead of O($S_mM_I\ M_o$). The complementary hash values that the total size of them is $s_h(M_o – S_mM_I)$ will be sent to the private cloud. Thus, the cross-cloud communication overhead is $s_h(M_o – S_mM_I) = O(M_o – S_mM_I)$.

In reduce phase integrity check, we explain the details of analysis for the five steps. In the original step, each reduce task
on the public cloud will generate the hash value of its output so the overhead is w_hR_o = O(R_o). Since each reduce task only has to send the hash value of its output H_k (whereas k is a distinct key that belongs to one reduce task) to the master on the private cloud. Thus, the communication overhead is O(1).

In the request step, distinct keys in output records of subsets execution are sent to the public cloud by the master as a request. In the worst case, each output record includes a distinct key, at most S_rM_o records (whereas the output of subsets of map tasks input on the private cloud are part of the output of mappers on the public cloud) are sent to the public cloud by the master. Therefore, the communication overhead in this step is $s_kS_rM_o = O(S_rM_o)$. In the response step, each map task on the public cloud needs to locate from their output records all records for each received key in the request step. So, locating all records for each key from the output of all map tasks takes $w_rM_o = O\ (M_o)$ time. Thus, there exists $O(S_rM_o)$ keys in the request step, locating all records from all map tasks output for all received keys will incur a computation overhead of $w_rS_rM_oM_o = O(S_rM^2o)$. Also in the response step, output records number of all map tasks that belongs to the received keys to be returned is at most $O(S_rM_o\alpha)$. The communication overhead is $O(S_rM_o\alpha)$.

Table 5: The Overhead of our technique.

Phase	Step	Public Cloud Computation	Private Cloud Computation	Cross-Cloud Communication
Map Phase Integrity Check	Original	$O(M_O)$	-	O(1)
	Replication	-	$O(S_m\ M_I)$	$O(S_m\ M_I)$
	Request	-	-	$O(S_m\ M_I)$
	Response	$O(S_m\ M_I\ M_O)$	-	$O(M_O - S_m\ M_I)$
Reduce Phase Integrity Check	Original	$O(R_O)$	-	O(1)
	Request	-	-	$O(S_r\ M_O)$
	Response	$O(S_r\ M^2{}_O)$	-	$O(S_r\ M_O)$
	Verification	-	$O(S_r\ M_O)$	-
	Rest_verification	$O(R_O+M_O(1 - S_r))$	-	O(1)
Total Overhead		$(S_m+S_r)M^2{}_I+M_IS_r+R_O$	$(S_r+S_m)\ M_I$	$(S_r+S_m)\ M_I$

In the verification step, after the master receives the records from the previous step, it uses these records to be executed by reduce tasks on the private cloud (these reduce tasks are replication of some reduce tasks on the private cloud). The computation overhead is proportional to the received records. So the time complexity for this computation is $w_t\ S_rM_o\alpha$. After the reduce tasks finish on the private cloud, it computes the hash value of their outputs. Since each reduce task generates its output as one record, the overhead for computing the hash value of all reduce task's output $w_hS_rM_o$. The total overhead in this step is $(S_rM_o(w_t\ \alpha+w_h)) = O\ (S_rM_o)$.

In the rest verification step, the rest of reduce tasks that are not verified from their result on the private cloud will be replicated its execution on the public cloud on verified nodes. Since the reduce tasks that are verified from its result on private cloud takes some of map tasks outputs as inputs of these reduce tasks, the rest of map tasks outputs are used as inputs of the rest reduce tasks that are not verified from its result. So the overhead of this computation is $w_t\ (M_o\text{-}S_rM_o\alpha)$. The overhead of generating the hash values of outputs of these reduce tasks (the rest of reduce tasks) is $w_h(R_o - S_rM_o)$ whereas S_rM_o presents the number of outputs of reduce tasks on the private cloud because each record in S_rM_o presents distinct key so the total overhead of computation is $o(M_o - S_rM_o\alpha)+o(R_o - S_rM_o) = o(R_o+M_o(1 - S_r))$ whereas α is a constant value. Then, each reduce task from these reduce tasks (the rest of reduce tasks) will only send the hash value of its output to the master as H'_K so the overhead of sending is O(1). The total overhead can be simplified by assuming $O(M_I) = O(M_o)$. Table 5. shows overhead for different aspects and the total overhead for different aspects are shown in the last row in Table 5.

6 Conclusions

In this paper, we present our technique that checks the result's integrity of MapReduce computation to ensure the result accuracy. Our technique deploys MapReduce on a hybrid cloud to combine the advantages of both public and private clouds. We design our technique in two phases: map phase integrity check and reduce phase integrity check respectively. In the map phase, all the map tasks execute on the public cloud, but subset records are replicated on the private cloud to verify each map task result. While, in the reduce phase, only a portion of reduce tasks will be replicated on the private cloud, and the remainder will be replicated on a trusted node on the public cloud. The performance overhead and quantitative analyzes of the proposed technique provide high computation integrity and little performance overhead.

Conflict of Interest

The authors declare that there is no conflict regarding the publication of this paper.

References

[1] J. Wang, Y. Yang, T. Wang and J. Zhang, ”Big Data Service Architecture: A Survey,” Journal of Internet Technology., **21(1)**, 393-405, 2020.

[2] G. M. Vaidya and M. M. Kshirsagar, ”A Survey of Algorithms, Technologies and Issues in Big Data Analytics and Applications,” in 4th International Conference on Intelligent., 2020.

[3] W. Fang, X. Z. Wen, Y. Zheng and M. Zhou, ”A survey of big data security and privacy preserving,” IETE Technical Review., **34(3)**, 544-560, 2016.

[4] M.M.El-Gayar , M. EL-Hasnony, Intelligent System for Ranking Big Data in Search Engine, Journal of Intelligent Systems and Internet of Things., **3(2)** , 43-56, (2021).

[5] D. N, S. B and V. S, ”Big data Hadoop MapReduce job scheduling: A short survey.,” Information Systems Design and Intelligent Applications., **5**, 349-365, 2019.

[6] K. Shankar, Improving the Security and Authentication of the Cloud with IoT using Hybrid Optimization Based Quantum Hash Function, Journal of Intelligent Systems and Internet of Things., **1(2)** , 61-71, 2020.

[7] J. Dean and S. Ghemawat, ”MapReduce: simplified data processing on large clusters,” in the 6th Conference on Symposium on Operating Systems Design Implementation., 2004.

[8] W. Wei, J. Du, T. Yu and X. Gu, ”Securemr: A service integrity assurance framework for mapreduce,” in Annual Computer Security Applications Conference, 2009.

[9] Muhammad Edmerdash, Waleed khedr, Ehab Rushdy, An Overview of Cloud-Based Secure Services for Enterprise Drug–Drug Interaction Systems, International Journal of Wireless and Ad Hoc Communication., **2(2)** , 49-58, 2021.

[10] Y. Wang, J. Wei, M. Srivatsa, Y. Duan and W. Du, ”IntegrityMR: Integrity assurance framework for big data analytics and management applications,” in International Conference on Big Data., 2013.

[11] Z. Xiao and Y. Xiao, ”Achieving accountable MapReduce in cloud computing,” Future Generation Computer Systems., **30**,1-13, 2014.

[12] H. Ulusoy, M. Kantarcioglu and E. Pattuk, ”TrustMR: Computation integrity assurance system for MapReduce,” in International Conference on Big Data (Big Data)., 2015.

[13] M. Ilayaraja, Particle Swarm Optimization based Multihop Routing Techniques in Mobile ADHOC Networks, International Journal of Wireless and Ad Hoc Communication., **1(1)**, 47-56, 2020.

[14] Y. Wang, Y. Shen and H. Wang, ”Mtmr: Ensuring mapreduce computation integrity with merkle tree-based verifications,” IEEE Transactions on Big Data., **4**, 418-431, 2016.

[15] T. A. Samuel and A. N. M, ”Credibility-based result verification for Map-reduce,” in Annual IEEE India Conference (INDICON), 2014.

[16] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters”. Communications of the ACM., **51(1)**, 107- 113, 2008.

[17] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, ”Enabling security in cloud storage SLAs with CloudProof.” Proceedings of 2011 USENIX Annual Technical Conference, Portland, OR. 2011.