International Journal of Grid Distribution Computing
Vol.7, No.6 (2014), pp.53-70
http://dx.doi.org/10.14257/ijgdc.2014.7.6.05

Securing MapReduce Result Integrity via Verification-based Integrity Assurance Framework

Yongzhi Wang[1], Jinpeng Wei[2] and Yucong Duan[3]

[1,2]Floridia International University, [3]Hainan University
[1]ywang032@cis.fiu.edu, [2]weijp@cis.fiu.edu, [3]duanyucong@hotmail.com

Abstract

MapReduce, a large-scale data processing paradigm, is gaining popularity. However, like other distributed computing frameworks, MapReduce suffers from the integrity assurance vulnerability: malicious workers in the MapReduce cluster could tamper with its computation result and thereby render the overall computation result inaccurate. Existing solutions are effective in defeating the malicious behavior of non-collusive workers, but are less effective in detecting collusive workers. In this paper, we propose the Verification-based Integrity Assurance Framework (VIAF). By using task replication and probabilistic result verification, VIAF can detect both non-collusive and collusive workers, even if the malicious workers dominate the environment. We have implemented VIAF on Hadoop, an open source MapReduce implementation. Our theoretical analysis and experimental result show that VIAF can achieve high job accuracy while imposing moderate performance overhead.

Keywords: *Distributed systems; Security; Verification; Integrity*

1. Introduction

MapReduce 0 has become the dominant paradigm for large-scale data processing applications such as web search engine, data mining, and scientific simulation. However, given the huge cluster that MapReduce normally requires, many MapReduce users cannot afford or do not want to invest in computer clusters of such a large scale. Therefore, they could resort to public computing service platforms such as Volunteer Computing [1][2] and Cloud Computing [3][4]. However, Volunteer Computing platforms have long been blamed as uncountable due to the existence of malicious nodes [5][6]. And a skillful hacker can also breach the security of commercial clouds. For instance, [7] points out a security vulnerability that Amazon EC2 [3] suffers from: some members of the EC2 community can create and upload malicious Amazon Machine Images (AMIs) which, if widely used, could flood the community with hundreds of infected virtual machine instances. Along the same line, Bugiel et al., [8] perform a systematic study of the security status of public AMIs and report various vulnerabilities in Amazon EC2. Therefore, security concerns make many corporations reluctant to outsource their critical computation to the public services.

More specifically, each MapReduce computation job is carried out via the collaboration of a number of computing nodes. If malicious nodes exist in the MapReduce cluster, the malicious node could tamper with its computation result and thereby renders the overall computation result inaccurate. Hence, we need a solution that can guarantee high computation accuracy in such an outsourced environment.

Several techniques such as replication, sampling, and checkpoint-based solution have been proposed to address such issues in different distributed environments such as P2P Systems, Grid Computing, and Cloud Computing [12-24]. However, existing solutions are hardly applied directly to meet high accuracy requirement of MapReduce due to their limitations. For example, SecureMR [18] that is based on probabilistic replication can

ISSN: 2005-4262 IJGDC
Copyright ⓒ 2014 SERSC

detect collusive workers (workers that return consistent but bad results) with a high probability only when the majority of workers are benign. When collusive worker ratio increases to 80%, SecureMR can only achieve less than 75% of detection rate even if the duplication rate is 1.0. When collusive worker ratio is 100%, the detection rate of SecureMR is 0% regardless of the duplication rate (More details can be found in Section 5). The pure quiz-based idea proposed in [11] incurs high overhead (200% to 450%) when the percentage of malicious hosts increases from 5% to 50%; by introducing a reputation mechanism, the overhead decreases but at the cost of slow reputation accumulation. For example, simulation in [11] suggests that in order to accumulate a reliable reputation, up to 10^5 tasks may be executed by a worker, which is not always feasible in the MapReduce job.

In this paper, we propose VIAF (Verification based Integrity Assurance Framework), a novel MapReduce framework that can detect both collusive and non-collusive workers in an open environment, even if the malicious worker dominates the MapReduce cluster. Since our framework performs the result check on each task, it can effectively defeat the "smart malicious worker" that behaves well to accumulate a good reputation before returning bad results. In VIAF, we replicate each task to detect non-collusive workers. In addition, we add limited number of trusted computation nodes called verifier to verify a small portion of consistent results in a probabilistic manner. We call passing a task replication and probabilistic task verification as passing one *quiz*. We accumulate the number of passed quizzes for each worker. If a worker passes certain amount of quizzes, we believe that this worker is not a malicious one and accept its results. Once a worker fails any quiz, we confirm that it is a malicious node and add it to a blacklist.

We perform theoretical analysis on VIAF and implement it on top of Apache Hadoop MapReduce[9]. Both theoretical analysis and experimental result demonstrate that VIAF can achieve high accuracy while incurring moderate overhead.

The rest of the paper is organized as follows. System assumptions and attacker model are presented in Section 2. System design and theoretical analysis are described in Section 3. System implementation, experiment result and analysis are discussed in Section 4. Section 5 discusses related work. Finally, Section 6 concludes the paper.

2. System Assumption and Attacker Model

2.1. System Assumptions

In this paper, we mainly focus on whether a worker can provide correct task result. Hence we assume that the integrity of other components in the MapReduce cluster (such as data storage and network communication) is well protected. Therefore, we assume the Distributed File System (DFS) and the network communication are trusted.

In VIAF, we divide the computation hosts into two categories: untrusted nodes that take the majority portion in the whole environment and trusted nodes that take the minority portion. We introduce a new type of task in VIAF called *verification*. It is a special task, which is launched by the trusted nodes to verify the task results submitted by the untrusted nodes. In order to retain the control of MapReduce job, we deploy the master on the trusted node and always assign verification task to the trusted node. We assign the map and the reduce tasks to the untrusted node. By doing so, we assume the master and the verification task results are trusted, and the map and the reduce task results are untrusted.

2.2. Attacker Model

The attackers in this system model are malicious workers that try to generate bad results in order to sabotage the job output. They are categorized into two types: *non-collusive malicious worker* and *collusive malicious worker*. (In the rest of this paper, we simply call them *non-collusive worker* and *collusive worker*, respectively.) A non-collusive worker normally returns bad result without consulting other malicious workers. In this case, if the same task is assigned to two workers and at least one of them is non-collusive, the cheat behavior can be easily detected by comparing the returns. In contrast, collusive workers can communicate with each other before cheating. When a collusive worker is assigned a task, it normally consults its collusive partners to see if they are assigned the same task. If yes, they may return consistently bad result; otherwise, it will return correct result in order to avoid detection. Since collusive workers try their best to minimize the inconsistency among their returns, they are much harder to detect than non-collusive workers.

3. System Design and Analysis

3.1. System Design

VIAF combines task replication, probabilistic result verification, and credit-based reputation management. In VIAF, each task is assigned to two workers, and the hash code of the result is returned to the master. (The worker keeps the actual task result in its local storage, the commitment-based protocol such as SecureMR [18] is employed to ensure that the hash code reported to the master is consistent with the worker's local storage). The consistency of two workers' returns will be checked in order to detect the non-collusive worker. In order to detect the collusive worker, the master probabilistically launches a special task called the verification task on the trusted node, in order to verify the correctness of the consistent returns by re-executing the task. In order to determine whether a worker is trusted or not, the master keeps on observing its returned result. Therefore, when a worker finishes one task, the task result is temporarily stored on the worker. And the credit of that worker is incremented by the master. When a worker has accumulated enough credits, the master will accept all its stored results, and its credit is reset to zero.

Figure 1 depicts the control flow of VIAF. In Figure 1, the Task Queue, History Caches for each worker and the Result Buffer are maintained in the master. W1 and W2 are any two workers performing the replicated tasks. The step "Verify Consistent Result" is executed on the verifier. Whenever the Task Queue is not empty, the master will pick one task from it and send it to any two workers (i.e., W1 and W2), shown in step 1 of Figure 1. After execution, both workers will return their results (in the form of hash code) to the master. The master compares the returned results to see if they are consistent (step2). If the returns are different, the master will reschedule the task to two new workers. (In this case, the master realizes that at least one worker is a non-collusive worker, but it is hard to tell which one is, so it has to reschedule the task to two different workers in the hope that the newly assigned workers are not malicious.) If the returns are consistent, the master may ask the verifier to verify this consistent result (step 3). Due to the limited resource on trusted nodes, verification will be launched in a probabilistic manner (*i.e.*, with certain probability). If the master determines not to perform verification, or if the master determines to perform verification and the verifier returns the same result as the workers, the master increments the two workers' credits and caches their results (hash value) and the task information to their respective *History Caches* (step 4). If the verification is performed but the verifier returns a different result, the master directly accepts the result from the verifier and determines that the two workers are collusive workers and therefore add the two workers to a black list. All the results in the History Caches of the two workers are untrusted. Therefore all the tasks in their History Caches will be rescheduled.

Copyright ⓒ 2014 SERSC

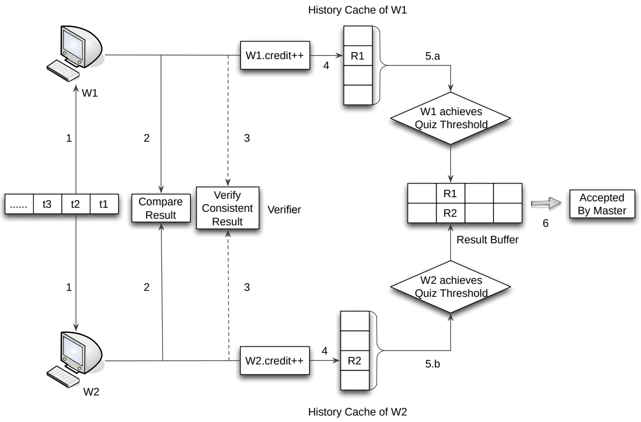


Figure 1. Control Flow Graph of VIAF

Each worker, after accumulating adequate credits, will obtain the master's trust temporarily, and all the results cached in its History Cache will be released to the Result Buffer (step 5.a and 5.b). After releasing the result, the worker's credit is reset to 0 and its History Cache is cleared. Notice that step 5.a and 5.b usually do not happen simultaneously. This is because the replicated tasks are randomly assigned to workers and the credit accumulation of each worker is independent. Therefore, the release of Cache History for each worker happens independently. It is very likely that when two workers (*e.g.*, W1 and W2) execute two replicated tasks and have their credit incremented, one worker (*e.g.*, W1) has accumulated enough credit and can release its History Cache to the Result Buffer, whereas the other worker (*e.g.*, W2) still has not accumulated enough credit.

Our strategy is to release the result of a task to the master only when both workers executing the task are trusted. To enforce that strategy, we design a Result Buffer to buffer the result of the replicate tasks. Only when two results of the same task are received by the Result Buffer, can the task's result be trusted by the master (step 6). In other words, the result accepted by the master is issued by two workers, both of which have passed adequate quizzes and obtained the master's trust. If one worker fails to pass a quiz, not only will the tasks in its History Cache be rescheduled, the relevant replicated tasks already received by the Result Buffer will also be rescheduled.

For the non-collusive workers, since their bad returns will always be detected, their task will always be rescheduled. Therefore, non-collusive worker will not undermine the accuracy of job result. For the collusive workers, since the verification is invoked in a probabilistic way, they cannot predict whether their returns will be verified. The only way to avoid detection is to return the correct result. For those collusive workers who try to take a risk to evade verification, the probability of passing quiz on each task assignment turns out to be very low.

For the job that contains a small number of map/reduce tasks (*e.g.*, the map/reduce tasks are less than the credit each worker has to accumulate in order to temporarily earn the master's trust), VIAF directly assigns the tasks to the verifier since the computation load is not significant.

3.2. . Theoretical Analysis

In this section, we perform a theoretical analysis to measure the VIAF accuracy and the performance overhead. Usually in a parallel computation model such as MapReduce, the result of each task contributes to the overall job result equally. Thus the job accuracy is proportional to the portion of tasks that return correct result. Therefore, we measure the job accuracy based on the probability of each task that offers correct result.

Copyright ⓒ 2014 SERSC

For the performance analysis, we assume each task execution takes the same amount of time. With this assumption, the time spent on a job is proportional to the number of task executions. Therefore, we measure the performance overhead with the portion of extra tasks to be executed in VIAF. Another factor of performance delay comes from the credit accumulation: the master accepts a worker’s result only when the worker’s credit achieves the quiz threshold. Therefore, the tasks executed based on the previous task result can be started only when the worker executing the previous tasks achieves the quiz threshold. However, our experiment result in Section 4.3.B (Figure 5 (b)) shows that such a mechanism will not impose significant delay. Therefore, we evaluate the performance overhead with the number of extra task executions. Based on the above analysis, we define the following metrics for the accuracy and performance overhead analysis:

Definition 1. (*Survival Chance*) The Survival Chance of a collusive worker is the probability that it passes all quizzes and earns trust of the master so that the result of its History Cache can be released to the Result Buffer. It is denoted as Δ.

Definition 2. (*Cheat Probability*) The Cheat Probability of a task is the probability that its result is incorrect but is accepted by the master. It is denoted as CP.

Definition 3. (*Accuracy*) The Accuracy of a task is the probability that its result is correct and is accepted by the master. It is denoted as AC. Apparently, AC = 1-CP.

Definition 4. (*Payload*) The Payload of a task is the average number executions launched on untrusted workers. It includes original task execution and additional task executions due to replication and reschedule. It is denoted as PL.

Definition 5. (*Overhead*) The Overhead of a task is the average number of extra executions launched on untrusted workers. It only includes the executions due to task replication and reschedule. It is denoted as OH. Therefore, we have OH=PL-1, where 1 is for the original task execution.

Definition 6. (*Verification Overhead*) The Verification Overhead of a task is the average number of executions launched on the trusted workers (i.e., verifiers). It only includes verification task. It is denoted as VO.

We model the malicious worker in the environment as follows. Our analysis model assumes an environment that contains a large number of workers. Suppose the environment contains N workers, and M out of which are malicious, we define *malicious node ratio* m = M/N. For simplicity, we assume that m stays constant throughout the job execution. That is, even though some malicious nodes are detected and black-listed during job execution, m decreases very little given the large values of N and M.

Since VIAF replicates each task, each non-collusive worker can be detected immediately by result comparison. Therefore, the detection rate of non-collusive worker is 100%. Inaccuracy introduced by the non-collusive worker can be eliminated. Therefore, in the following analysis, we only focus on the environment consisting of only collusive malicious worker.

For simplicity, we assume that there exists only one collusion group in the untrusted workers. In other words, all the collusive workers can collaborate with each other. Even though a task is assigned to two collusive workers, collusion can happen only when they are executing the same task simultaneously (for example, collusion cannot happen if the two workers fall out of sync in processing the same task). Due to the uncertainty of temporal sequence, we define p (*collusion opportunity*) as the probability that one collusive worker can find another collusive worker executing the replicated task and be able to commit a cheat. We assume the collusive workers to be conservative and intelligent: they only commit a cheat when they find a collusive partner. In addition, in order to reduce the risk of detection, they do not cheat on every task. Since VIAF perform task verification randomly, a collusive worker cannot predict which task result it submits will be verified. Therefore, when they find a collusive partner, they cheat randomly in a hope of not to be caught. We define q (*collusion tendency*) as the probability that the collusive partners decide to commit a cheat.

We model the VIAF system setting as follows. Since verification is launched with a certain probability, we define such a probability as the *verification probability*, denoted as v. We define the quizzes each worker has to pass in order to temporarily earn the master's trust as the *quiz threshold*, denoted as k.

The model parameters and measurement metrics are summarized in Table 1.

We start with the analysis of survival chance Δ. The survival chance of a collusive worker is the probability that such a worker can pass k quizzes in the k task assignments. Remember that since our analysis environment contains no non-collusive workers, the replicated task results are always consistent. In order to survive, each task execution should not fall into the following situations: *"The replicated task is assigned to another collusive worker (with probability m). They are able to collude (with probability p), and they determine to cheat (with probability q). The consistent results are verified by a verification task (with probability v)."* The probability of this case is mpqv.

Table 1. Notation for Theoretical Analysis Model

Symbol	Name	Explanation
m	malicious node ratio	Malicious worker ratio out of all workers
p	collusion opportunity	When a task is assigned to two collusive workers, the probability that two workers can discover each other to commit a cheat.
q	collusion tendency	The probability that two collusive workers determine to commit a cheat when discovering each other.
v	Verification Probability	The probability that a task returning consistent results is verified.
k	Quiz Threshold	Number of quizzes a worker must pass in order to obtain the trust of the master.
Δ	Survival Chance	The probability that a worker passes all the k quizzes and earns trust of master.
CP	Cheat Probability	The probability that a task returns a bad result and accepted by the master.
AC	Accuracy	The probability that a task returns a good result and accepted by the master.
PL	Payload	The average number of executions launched on untrsuted workers for each task.
OH	Overhead	The average number of extra executions (excluding the original task execution) launched on untrusted workers for each task.
VO	Verification Overhead	The average number of executions launched on trusted workers (verifiers) for each task.

For a collusive worker, the probability of k consecutive task executions that avoid falling into the above situation is therefore

$$D=(1- mpqv)^{k} \qquad (1)$$

The cheat probability CP can be derived with the information of survival chance. The master accepts a bad result from a malicious worker only in one of the following two scenarios:

a) The task is assigned to two collusive workers. They both survive in k quizzes and they commit a collusive cheat. The probability in this case is $m^2pq\Delta^2$.

b) The task is assigned to two collusive workers. Any of them fails to pass the k quizzes, so the task has to be rescheduled. Knowing the Cheat Probability of a rescheduled task as CP, we have the cheat probability as $m^2(1-\Delta^2)CP$ in this case.

Combining the two cases, we have the cheat probability CP as:

$$CP = m^2 pq\Delta^2 + m^2(1-\Delta^2)CP$$

Therefore,

$$CP = \frac{m^2 pq\Delta^2}{1-m^2(1-\Delta^2)} \tag{2}$$

Since AC = 1-CP, we have

$$AC = 1-\frac{m^2 pq\Delta^2}{1-m^2(1-\Delta^2)} \tag{3}$$

Payload PL can be calculated with the same principle. In our model, rescheduling happens in two cases:

a) The task is executed by two collusive workers and at least one of them fails to pass the k quizzes. The probability of this case is $m^2(1-\Delta^2)$. The payload is 2+PL, where 2 is for the replicated task and PL is the payload after reschedule.

b) For the other cases, the payload is 2 since there's no reschedule needed. The probability is $1-m^2(1-\Delta^2)$

Adding above overheads together, we have $PL = m^2(1-\Delta^2)(2+PL)+(1-m^2(1-\Delta^2))*2$

Therefore, we have derived the payload as:

$$PL = \frac{2}{1-m^2(1-\Delta^2)} \tag{4}$$

Since the overhead OH=PL-1, we have

$$OH = \frac{2}{1-m^2(1-\Delta^2)} - 1 \tag{5}$$

Similarly, we can calculate the verification overhead. Suppose the average verification overhead of each task is VO, consider the following cases:

a) The task is assigned to two collusive workers and at least one of them fails to pass the k quizzes. The probability in this case is $m^2(1-\Delta^2)$.When the task is verified (with probability v), the verification result will be directly accepted by the master. The verification overhead is $m^2(1-\Delta^2)v$. However, when the task is not verified (with probability of 1-v), it will be rescheduled. Since after reschedule, the average verification overhead is VO, we have the verification overhead as $m^2(1-\Delta^2)(1-v)VO$. Combining the two cases, we have the verification overhead $m^2(1-\Delta^2)v+m^2(1-\Delta^2)(1-v)VO$.

b) For the other cases, where the probability is $(1- m^2(1-\Delta^2))$, the task will not be rescheduled but can be verified with probability v. The verification overhead is therefore $(1-m^2(1-\Delta^2))v$.

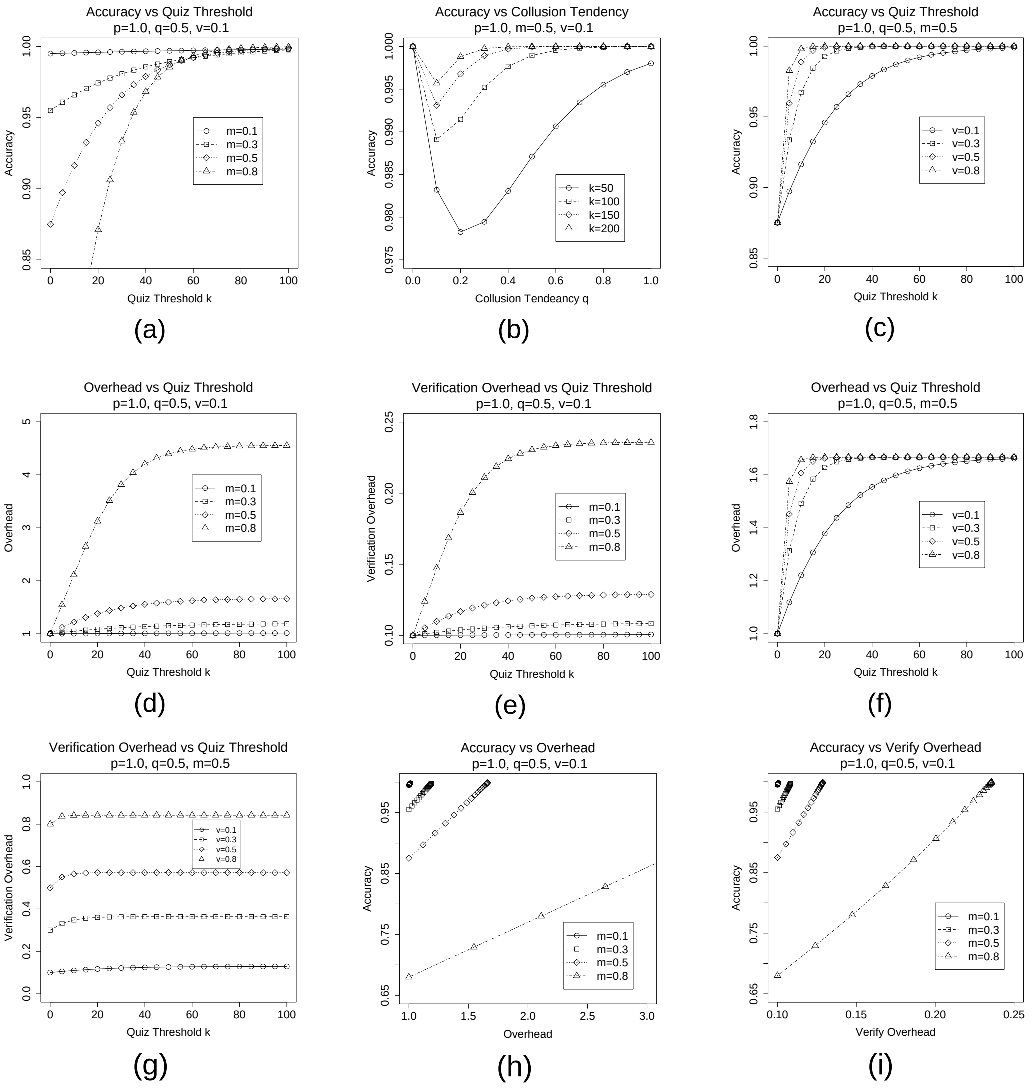
The verification overhead VO consists of the above two cases. We have

$$VO = m^2(1-\Delta^2)v + m^2(1-\Delta^2)(1-v)VO + (1-m^2(1-\Delta^2))v$$

Reorganizing the above equation, we have

$$VO = \frac{v}{1-m^2(1-\Delta^2)(1-v)} \tag{6}$$

(3), (5) and (6) indicates that the accuracy, overhead and verification overhead are determined by both the VIAF system configuration and the malicious node behavior. Specifically, the system configuration includes the quiz threshold k, and the verification probability v. The malicious node behavior includes the malicious node ratio m, the collusion opportunity p and the collusion tendency q. In order to better study their effects on the accuracy, overhead and verification overhead, we perform a series of simulation based on (3), (5) and (6). In this set of simulation, we assume the collusion opportunity to be 1.0 for simplicity. Such a setting represents the worst-case scenario, *i.e.*, the malicious node can always collude. The simulation result is in Figure 2.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 2. Simulation based on Theoretical Analysis Result

Figure 2 (a) shows the relationship between quiz threshold and accuracy based on (3) and (1). In this figure, we set the collusion opportunity p as 1.0, the collusion tendency q as 0.5, and the verification probability v as 0.1. The malicious node ratio m is set to 0.1, 0.3, 0.5 and 0.8, respectively. For each malicious node ratio setting, we increase the quiz threshold k from 0 to 100. The simulation result shows that the accuracy increases with the increase of the quiz threshold. Even if when m is large, VIAF can still guarantee high accuracy by setting a higher quiz threshold. For example, when m is 0.1, setting k as 20 would guarantee accuracy close to 100%. When m is 0.8, accuracy close to 100% can still be gained by setting a k value more than 80.

Figure 2 (b) shows the relationship between the accuracy and collusion tendency q. The simulation shows that when the quiz threshold k is fixed, the malicious node can achieve the lowest accuracy at a value of q between 0 and 1. Such a result reflects that VIAF forces the malicious node to make a trade-off. On one hand, collusion with a very high collusion tendency would be detected easily; on the other hand, collusion with a very low collusion tendency does not inject many errors to the computation job. The figure also shows the minimal accuracy VIAF can guarantee: When m is 0.5, p is 1.0, v is 0.1, setting k as 50 can guarantee more than 97.5% of accuracy. With the same environment setting, setting k as 200 can guarantee more than 99.5% of accuracy.

Figure 2 (c) shows how accuracy is affected by the verification probability v. A higher value of v could guarantee a higher value of accuracy when k is not big enough. For example, when k is 20, VIAF can guarantee accuracy less than 95% when v is 0.1, compared to accuracy close to 100% when v is 0.8. However, when k is big enough (*e.g.*, k=100), the value of v does not significantly affect the value of accuracy.

Figure 2 (d) and (e) show that the overhead and verification overhead both increase with the increase of the quiz threshold. However, both the overhead and the verification overhead have an upper bound when m, p, q and v are fixed. When quiz threshold increases to a certain value (*e.g.*, k=100), the overhead and the verification overhead increase to their upper bounds. The figures also show that an environment with a higher malicious node ratio incurs a higher upper bound of overhead and verification overhead. Figure 2 (f) and (g) show how the verification probability v affects the overhead and verification overhead. The figures show that when the quiz threshold is large (*e.g.*, bigger than 100), different value of v does not incur significantly different overhead. However, a higher value of v incurs a higher value of verification overhead.

Figure 2 (h) and (i) show the relationship between the accuracy and the overhead/verification overhead. For each curve in the figures, the bottom-left most point corresponds to the setting where k is 0, and the top-right most point corresponds to the case where k is 100. The difference of k values between adjacent points on each curve is 5. Generally, the accuracy increases with the increase of the overhead/verification overhead. The figures show that when k is small (*e.g.*, 0), a higher value of m results in a lower accuracy and a lower overhead and verifier overhead. And the accuracy increases with the increase of the overhead and the verification overhead. We find that on each curve, the points become denser with the increase of k and eventually concentrate to their outmost limits. This suggests that when k is big enough (*e.g.*, bigger than 40), increasing k further would bring neither additional accuracy benefit, nor additional overhead or verifier overhead cost.

The simulation indicates the following suggestions in setting parameter of the VIAF:

1). According to Figure 2 (a), when other parameters are fixed, increasing the quiz threshold k would increase the accuracy. Meanwhile, Figure 2 (h) and (i) suggest that the overhead and the verification overhead would be increased with the increase of the accuracy. Therefore, when a system does not have overhead and verification overhead restriction, setting k as big as possible would guarantee high accuracy.

2). According to Figure 2 (c) and (f), increasing the verification probability v does not help boosting the accuracy when the quiz threshold k is large enough (*e.g.*, k>100). On the other hand, Figure 2 (g) shows that setting a higher value of v would incur a higher value of verification overhead. Therefore, in order to reduce the verification overhead, v should be set to a reasonably small number based on the value of k. For example, according to Figure 2 (c), when k is set to 100, setting v as 0.1 could guarantee the accuracy close to 100%. However, when k is set to 40, in order to guarantee the accuracy close to 100%, v should be set above 0.5.

4. Implementation and Evaluation

We have implemented a prototype system based on Hadoop MapReduce 0.20.2. With this implementation, we did a series of experiments.

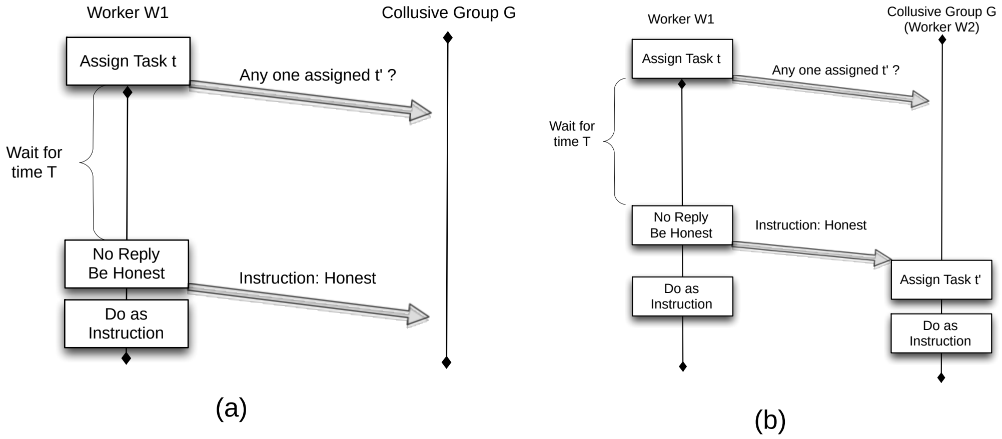
4.1. Implementation Details

Following the design in Figure 1, VIAF is implemented based on existing Hadoop implementation. In VIAF, each worker calculates the MD5 hash code of the task result and sends it back along with the task status. The master maintains the credit and the history cache of each worker, as well as the result buffer. We modify the task scheduler in the master to enforce the VIAF task assignment design. Verification tasks are assigned to the trusted workers. For each original and replicated task to be assigned, instead of passively waiting for the worker to ask for an assignment, the master always randomly chooses a worker from untrusted workers to assign the task. The original task and the corresponding replicated task are always assigned to two different untrusted workers. In addition, each rescheduled task is always assigned to a different untrusted worker than the one that has executed it before. When a worker is added to the black list, no task will be assigned to it. Adding a worker to the black list requires a list of rescheduling: Once a worker is added to the black list, each task in the worker’s execution cache will be rescheduled. Besides, all the tasks buffered in the result buffer and executed by the worker will be taken out and rescheduled as well (refer to Figure 1).

4.2. Collusive Worker Model

We implement the collusive workers in MapReduce environment in order to launch the experiment described in section 4.3. The collusive workers work as in Figure 3.

When a collusive worker W1 is assigned a task t, it first connects to all its collusive group members (the group member list is stored at each collusive worker) and queries if any other collusive worker is assigned the replicated task of t, let’s say t’. It will wait for a certain amount of time T for some collusive workers to respond to him. (It is worth noting that the collusion opportunity p in Table 1 will be higher if the waiting time is set as a large value). If W1 does not get any feedback from its entire collusive group when the waiting time T expires, it knows that no other collusive worker in its group is assigned t’ so far: maybe t’ is assigned to a worker not belonging to the collusive group (Figure 3(a)), or maybe t’ is not assigned yet (Figure 3(b)). In this case, W1 will behave honest to reduce the detection risk. It will compute honestly and announce his decision to the entire collusive group in case t’ is assigned to another collusive worker later (Figure 3(b)).



(a)

(b)

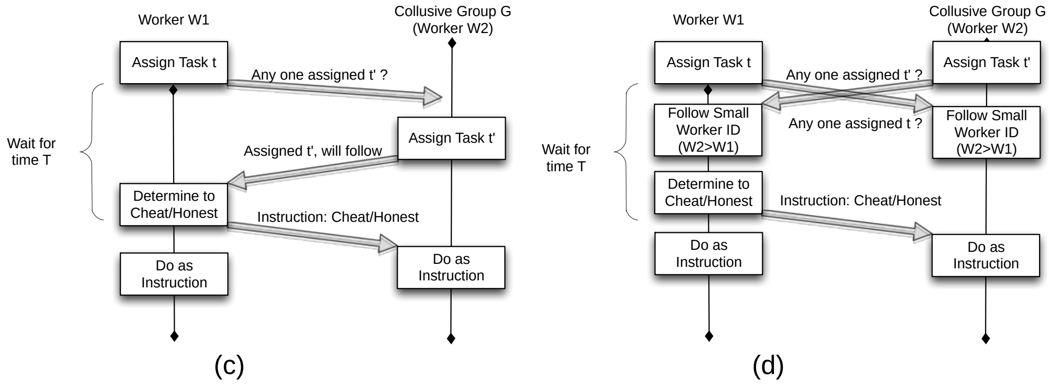


Figure 3. Collusive Worker Model

If W1 receives a response from a collusive worker W2 within waiting time T, indicating that t’ is assigned to W2 and W2 will follow the decision of W1, W1 will decide whether to cheat or to be honest, and send the decision to W2. Both workers will execute the tasks as the decision (Figure 3(c)). When t and t’ are assigned to two collusive workers simultaneously, the two workers will query each other simultaneously. In this case, the worker with a smaller node id (*e.g.*, W1) will decide whether to cheat and send the instruction to the other worker (*e.g.*, W2) (Figure 3(d)).

4.3. Experiment and Result Analysis

We launch a series of experiments to test the accuracy, overhead and performance of VIAF. The experiments are performed on a MapReduce cluster consists of nine virtual machines (512 MB of RAM and 40 GB of disk each). Each virtual machine runs on Debian 5.0.6 “lenny”. Out of the nine nodes, one is running as both a master and a trusted verifier; the remaining eight nodes are untrusted workers. All nine virtualized instances are deployed via VMware Workstation 7.11 on a Linux server with a 2.93 GHz, 8-core Intel Xeon CPU and 16 GB of RAM.

We observe the accuracy, overhead and verification overhead of VIAF by manipulating the collusive worker’s collusion tendency q and the quiz threshold k. In order to measure the performance overhead, we perform a set of experiments to compare the VIAF execution time with a baseline. The baseline is deployed on the same cluster as VIAF (*i.e.*, nine virtual machines deployed on a Linux server; one master runs on one virtual machine and the other eight workers run on the other eight virtual machines). It runs the standard Hadoop 0.20.2 framework. For each set of experiments, we launch the experiment under each configuration setting for 10 trials and compute the average value. Since each job that we test contains only one reduce task, we directly assign the reduce task to the verifier. Hence the replication and probabilistic verification only happens on the map phase. In other words, our measurement on VIAF is only on the map phase. However, since the VIAF design is suitable for both the map and the reduce phase. The measurement of the map phase also represents the fact of the reduce phase.

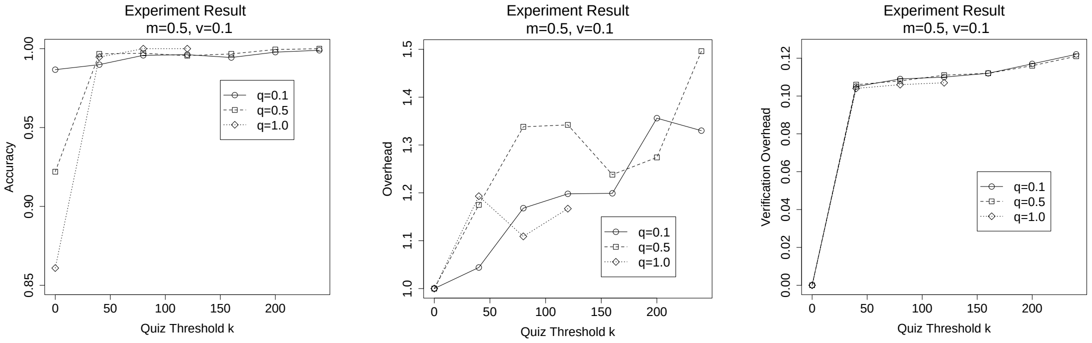
4.3.A. Accuracy, Overhead and Verification Overhead

We use TeraSort [10], a Hadoop benchmark application to test the accuracy, overhead and verification overhead of VIAF. In this experiment, each job sorts one gigabyte of randomly generated string records by the value of the record key. The entire job executed in standard Hadoop consists of 980 map tasks and one reduce task. Since the job contains only one reduce task, we directly assign it to the verifier to execute.

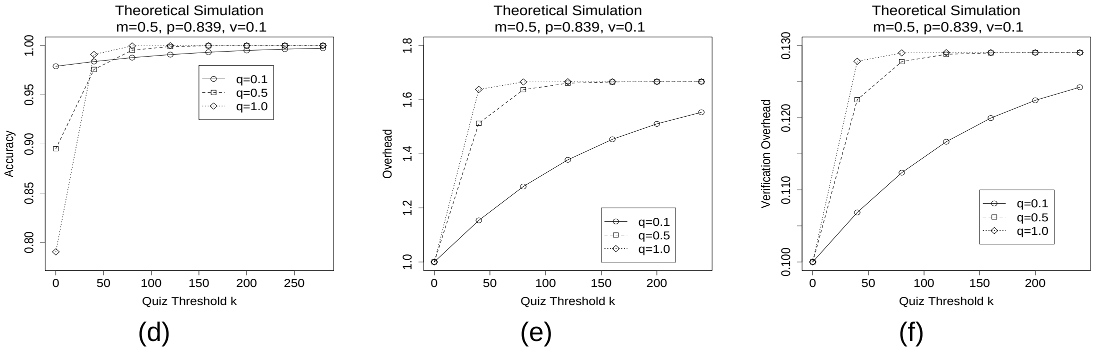
In the experiment, we set 4 out of 8 workers as collusive workers, which means that in this environment, the malicious node ratio m is set to 0.5. All the collusive workers are in one collusive group and are implemented as described in Section 4.2. We set verification probability v as a constant value of 0.1. We alter the collusion tendency q from 0.1 to 1.0, and change the quiz threshold k from 0 to 240, with a step size of 40. In our experiment, we found when q is 1.0 and k is greater than 80, the Accuracy constantly stays at 100%. Therefore, we skip the cases when q is 1.0 and k is greater than 120. The Accuracy, Overhead and Verification Overhead of experiment results are shown in Figure 4 (a), (b) and (c), respectively.

During the experiments, we have the collusive workers report to the master once they are able to collude. Thereby we collect the value of p (collusion opportunity) in each run of experiment and calculate the average value of p as 0.839 and an acceptable small standard deviation as 0.0284. With this value, we perform a simulation based on (3)(5) and (6) and generate the theoretical result of accuracy, overhead and verification overhead with the experimental setting in Figure 4 (d) (e) and (f). We point out that the simulation result shown here is just a reference for the reader to compare the general changing trend of accuracy, overhead and verification overhead. We do not expect exact match due to the following two reasons: 1). The experiment has randomness, which is abstracted away in the simulation. 2). The average value of p we use in the simulation is just a statistical evaluation, which could not accurately reflect each single experiment.

In this set of experiments, we measure Accuracy as the percentage of correct map results out of all the map results accepted by the master in a job. In Figure 4 (a), when q is 1.0, accuracy increases quickly from 86.09% to 100% when k increases from 0 to 80. When q is 0.1, Accuracy increases slowly from 98.67% to 99.90% when k increases from 0 to 240, but the initial Accuracy (i.e., 98.67%) when k is 0 is much larger compared with the initial value when q is 1.0 (i.e., 86.09%). The result indicates that VIAF is more efficient to quench the aggressive collusive workers that have higher collusion tendency. However, by setting a bigger value of quiz threshold k, VIAF can still effectively prevent the less aggressive collusive workers from injecting errors. By comparing Figure 4 (a) with (d), we can see that the increasing trends in two figures are similar. Notice that the accuracy of the experiment result is higher than the simulation value with the same environment and configuration setting. This is because our experiment has a limited number of malicious workers, so once all the collusive workers are added to the black list, the malicious node ratio m decreases from 0.5 to 0. We also notice that the accuracy decreases at some point. For example, when q is 0.1, accuracy decreases from 99.62% to 99.44% when k increases from 120 to 160. We believe this is due to some randomness during the experiment.



(a) (b) (c)



(d) (e) (f)

Figure 4. Accuracy, Overhead and Verifier Overhead of VIAF, Compared to the Theoretical Simulation

Data in Figure 4 (b) indicates that when increasing k from 0 to 240, Overhead increases from 100% to 133% when q is 0.1, and from 100% to 149% when q is 0.5. The Overhead consists of two parts: the replicated tasks and the rescheduled tasks. The former takes 100% and the latter includes the rescheduled tasks executed by the collusive workers that are added to the black list. The figure shows that a larger value of quiz threshold incurs a higher reschedule overhead. This is because a higher value of quiz threshold means more tasks are buffered in each worker before they are accepted by the master. Once a worker is added to the black list, all the tasks buffered in the worker will be rescheduled. Finally, we have to point out that the experimental randomness makes the overhead increase trend in Figure 4 (b) not as regular as the theoretical simulation in Figure 4 (e). However, the general increasing trend in the experiment is still consistent with the theoretical analysis. For instance, in Figure 4 (b), the slope of the curve with q as 0.1 is generally smaller than the curve with q as 0.5, which is consistent with Figure 4 (e). In addition, when k is 240, the overhead with q as 0.5 (*i.e.*, 149%) is higher than the overhead with q as 0.1 (*i.e.*, 133%).

Figure 4 (c) shows the Verification Overhead with different quiz threshold and collusion tendency. When q is 0.1, Verification Overhead increases from 0% to 12.2% as k increases from 0 to 240. When q is 0.5, Verification Overhead increases from 0% to 12.1%. Comparing Figure 4 (c) with Figure 4 (f), the verification overhead in the experiment is smaller than the simulation when q is 0.5 and 1.0. It is because when a blacklist is applied, the malicious node ratio will gradually decreased from 0.5 to 0. Therefore, less task reschedule will happen and the number of task verification is reduced.

4.3.B. Performance

In order to test VIAF performance, we compare its execution time with standard MapReduce. We measure *performance slow down*, which is defined as the extra percentage of execution time of VIAF compared to the baseline execution time.

In the performance test, we do not introduce malicious workers to the cluster environment. It is because according to the collusive worker model described in Section 4.2, the collusive workers may take longer time (due to the waiting time for the response of query) to return result compared to the benign workers. Hence, it is hard to distinguish whether the delay is caused by VIAF or is caused by collusive workers. Moreover, customers are usually willing to accept performance slow down for detecting malicious nodes in the cluster, but they are reluctant to suffer too much performance penalty in an environment free of malicious workers.

In order to perform a comprehensive measurement, we first measure the performance slow down of different MapReduce applications. After that, we choose one of the applications, Terasort, and adjust the quiz threshold k to observe the performance change.

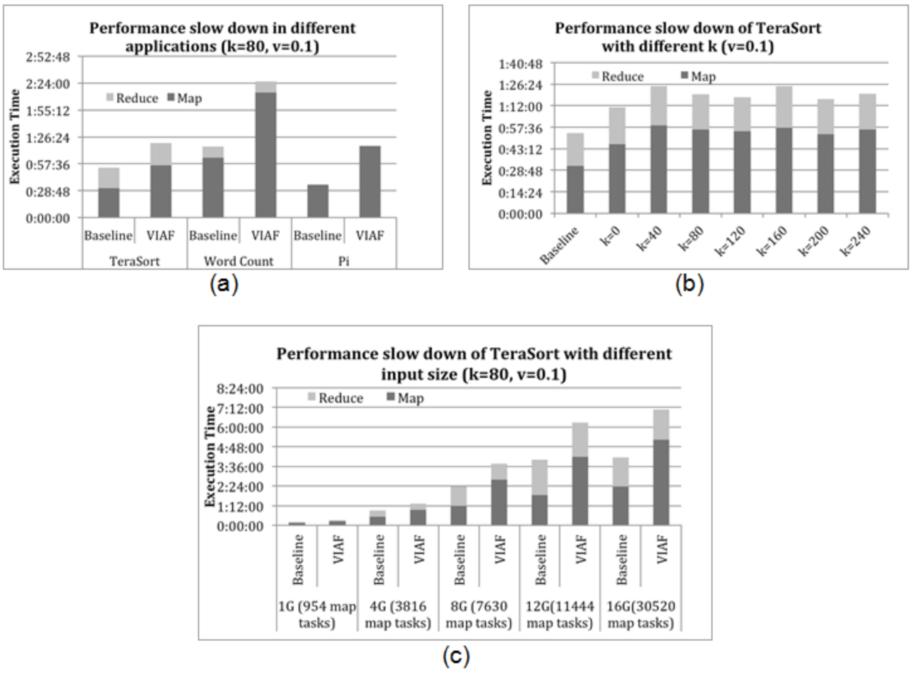
Finally, we launch scalability test to measure the execution time change when the input data size increases. The performance slow down on each experiment is shown in Figure 5.

We use three MapReduce applications to compare the VIAF execution time with baseline: Terasort, Word Count and Pi. The applications details and the job configurations are listed in Table 2. Among the three applications, Terasort and Word Count are I/O intensive jobs, and Pi is a CPU intensive job. In this set of experiments, we set quiz threshold k as 80, verification probability v as 0.1. Since each job executed in this set of experiments contains only one reduce task, we directly assign the reduce task to the verifier. Therefore, in each job execution, the reduce phase would not incur performance delay.

Table 2. MapReduce Applications for Performance Test

App. Name	Description	Job Configuration
Tera Sort	Sort Text Strings in Alphabetical Order	3,816 map tasks, input size is about 4G.
Word Count	Count the occurance of each word in the input text files	9,216 map tasks, input size is about 9G.
Pi	comput the value of Pi with Monte-Carlo Method	10,000 map tasks, each task takes 1,000,000 steps.

The execution time of each application is shown in Figure 5(a). For each application, the execution time of map phase in VIAF is doubled compared to the baseline. This extra execution time is attributed to the task replication and probabilistic task verification. The reduce phase execution time in the baseline is close to VIAF. Therefore, the overall job performance slow down is determined by the portion of reduce phase execution time in each job. For example, in the baseline, the portion of map phase execution time in Terasort, Word Count and Pi are 59%, 84% and 100%, respectively. And the job performance slow down due to VIAF is 49%, 91% and 118%, respectively. The result shows that the VIAF performance slow down is determined by how much portion of job will be replicated. The application type (*i.e.*, whether the job is CPU intensive or I/O intensive) does not affect the performance slow down significantly.



(a) (b) (c)

Figure 5. Performance Slow Down Evaluation of VIAF

Copyright ⓒ 2014 SERSC

We also measure the performance slow down under different quiz threshold k. In this set of experiments, we use the same TeraSort job described in Section 4.3.A. We set the verification probability v as 0.1, and increase the value of k from 0 to 240. The results are shown in Figure 5(b). Comparing with the baseline, we can see that VIAF incurs 32% to 59% of performance delay when k increases from 0 to 240. Such a delay is attributed to the map phase execution (since we directly send reduce task to the verifier). When k is 0, the map phase has 46% of performance delay and the performance delay increases until k achieves 40. When k is above 40, the performance delay becomes stable (i.e., 76% on average). We can see that the credit accumulation does introduce performance delay. However, when the quiz threshold is set above 40, further increasing the quiz threshold will not further increase the job execution time.

In the scalability test, we measure the execution time of TeraSort with different input sizes. In this set of experiments, we fix the quiz threshold k as 80, verification probability v as 0.1, and change the input data size from one Giga bytes to 16 Giga bytes (the input data is generated via the TeraGen program). The experimental result in Figure 5(c) shows that when input data size increases, the execution time on baseline and on VIAF both increase. For each input data size, the performance delay in the map phase is 112% on average. And no increasing trend is observed when the input data size increases. As a result, the performance slow down for the entire job is 61% on average, and no increasing trend is observed when input data size increases.

In summary, the performance experiment shows that the major performance slow down of VIAF is caused by the deterministic task replication. The credit accumulation and probabilistic verification does not impose significant performance delay. Meanwhile, our experiment shows that the VIAF performance is not affected by the input data size. When input data size scales up, the slow down of VIAF is stable.

5. Related Work

Integrity assurance of computation has been studied in a wide range of distributed environments, such as P2P systems, Grid Computing, and Cloud Computing [12-24]. Golle *et al.*, [16] propose to guarantee correctness of distributed computation result by duplicating computations. Following this direction, Sonnek [22] employs a replication-based scheme that allows the degree of redundancy to be adaptively adjusted based on the dynamically calculated reputation as well as reliability of each worker. To solve the related problem in distributed storage realm, Maniatis *et al.*, [19] proposed a majority voting based protocol to guarantee the integrity of replicated storage. Wei et al. propose SecureMR [18], an integrity assurance framework for MapReduce. By applying non-deterministic duplication, it can defeat both collusive workers and non-collusive workers with certain probability. Their work is close to ours. However, the effectiveness of SecureMR largely depends on the assumption that the majority of workers are benign. For example, according to formula (2) in [18], when n, p_m, b, l are set to 50, 0.5, 20, and 15, respectively, in order to achieve 90% detection rate when m is 25 (50% workers are collusive malicious), the duplication rate must be higher than 0.6; with the same environment setting, when m is 40 (80% collusive worker ratio), setting the duplication rate to 1.0 can only achieve less than 75% detection rate; and when m is 50 (100% collusive worker ratio), the detection rate is 0% regardless of the setting of duplication rate.

The sampling based idea "ringer" [17] is proposed by Golle *et al.*, in order to solve the problem of verifying computation completion for the "inversion of one-way function" class of computations. Following that idea, Zhao *et al.*, [11] propose to insert indistinguishable *Quizzes* to the task package that is going to be executed by the untrusted worker and verify the returned result for those Quizzes. Their simulation results show that by combining reputation system, Quiz approach gains higher accuracy and lower

Copyright ⓒ 2014 SERSC

overhead than replication-based approach. However, their simulation also suggests that reputation accumulation is a long-term process that requires too many tasks to be performed. For example, according to Figure 9 in [11], in order to defeat the “smart malicious worker” that performs well for a while to accumulate a good reputation before returning bad results with a fixed probability, up to 10^5 tasks must be executed by a worker. Varalakshmi [21] proposes a similar idea based on sampling technique to use tasks of the given jobs themselves as quiz questions. Du et al. [12] propose to insert several sampled tasks to the task package and check the sampled task returns using Merkle-tree based commitment technique. Michalakis *et al.*, [20] present the Repeat and Compare system to detect errors from untrusted replicas in P2P Content Distribution Networks. The idea is to randomly pick some tasks that are executed by untrusted replicas and send them to other replicas (or trusted host) to verify the result.

An early version of this work is presented in [24]. However, in this paper, we improve our system design so that both the map and the reduce tasks can be executed on the untrusted worker. Also, we loosen our environment assumption so that the malicious worker could take majority portion in the MapReduce cluster. In addition, we perform further theoretical study and experimental evaluation. Specifically, we perform more theoretical studies and suggest the system setting strategy. In addition, more experiments are performed to comprehensively evaluate the efficacy and performance of VIAF under different environment setting, application type, and job input size.

6. Conclusion

To guarantee high accuracy of MapReduce calculation, we propose the VIAF (Verification based Integrity Assurance Framework), a MapReduce framework that can defeat both collusive and non-collusive malicious workers. We present the design and implementation detail of such a framework, along with theoretical analysis and experimental evaluations. Our analysis and evaluation result shows that this framework can guarantee high result integrity while incurring moderate performance overhead.

Acknowledgements

This material is based upon work supported by the U.S. Department of Homeland Security under grant Award Number 2010-ST-062-000039. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, Communications of the ACM, vol. 1, no. 51, **(2008)**.

[2] P. Anderson, “BOINC: a system for public-resource computing and storage”, Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing, Pittsburgh, USA.

[3] SETI@home. http://setiathome.ssl.berkeley.edu/.

[4] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/.

[5] Windows Azure Compute. https://www.windowsazure.com/en-us/home/features/compute.

[6] Andrew Colley. Cheats wreak havoc on seti@home: participants. ZDNet, Australia, **(2002)**.

[7] D. Molnar, “The seti@home problem. E-Commerce”, **(2000)**.

[8] Cloud Security: Amazon’s EC2 serves up 'certified pre-owned' server images. http://dvlabs.tippingpoint.com/blog/2011/04/11/cloud-security-amazons-ec2-serves-up-certified-pre-owned-server-images.

[9] S. Bugiel, S. Nürnberger, T. Pöppelmann, A.-R. Sadeghi and T. Schneider, “AmazonIA: when elasticity snaps back”, Proceedings of the 18th ACM conference on Computer and communications security, Chicago, USA, **(2011)** October, 17-21.

[10] MapReduce Tutorial. http://hadoop.apache.org/mapreduce/docs/ current/mapred_tutorial.html.

[11] O. ’Malley and C. Murthy, “Winning a 60 Second Dash with a Yellow Elephant. http://sortbenchmark.org/Yahoo2009.pdf.

[12] S. Zhao, V. Lo and C. Gauthier Dickey, “Result verification and trust based scheduling in peer-to-peer grids. Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing, Konstanz, Germany, **(2005)** August 31- September 2.

[13] W. Du, J. Jia, M. Mangal and M. Murugesan, “Uncheatable grid computing”, Proceedings of the 24th International Conference on Distributed Computing Systems, Tokyo, Japan, **(2004)** March 23-26.

[14] L. F. G. Sarmenta, “Sabotage-tolerance mechanisms for volunteer computing systems”, Future Generation Computer Systems, vol. 4, no. 18, **(2002)**.

[15] C. Germain-Renaud and D. Monnier-Ragaigne, “Grid result checking”, Proceedings of the 2nd conference on computing frontiers, New York, NY, USA, **(2005)**.

[16] P. Domingues, B. Sousa and L. Moura Silva, “Sabotage-tolerance and trust management in desktop grid computing”, Future Generation Computer Systems, vol. 7, no. 23, **(2007)**.

[17] P. Golle and S. Stubblebine, “Secure distributed computing in a commercial environment”, Proceedings of the 5th International Conference Financial Cryptography, Grand Cayman, British West Indies, **(2002)** February 19-22.

[18] P. Golle and I. Mironov, “Uncheatable distributed computations”, Proceedings of the 2001 Conference on Topics in Cryptology, San Francisco USA, **(2001)** April 8-12.

[19] W. Wei, J. Du, T. Yu and X. Gu, “SecureMR: A Service Integrity Assurance Framework for MapReduce”, Proceedings of the 2009 Annual Computer Security Applications Conference, Honolulu, Hawaii, USA, **(2009)** December 7-11.

[20] P. Maniatis, D. S. H. Rosenthal, M. Roussopoulos, M. Baker, T. J. Giuli and Y. Muliadi, “Preserving peer replicas by rate-limited sampled voting”, Proceedings of the 19th ACM symposium on Operating systems principles, New York, USA, **(2003)** October 19-22.

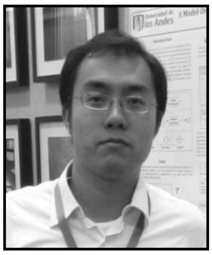
[21] N. Michalakis, R. Soul and R. Grimm, “Ensuring content integrity for untrusted peer-to-peer content distribution networks”, Proceedings of the 4th USENIX conference on Networked systems design & implementation, Cambridge, USA, **(2007)** April 11-13.

[22] P. Varalakshmi, S. Thamarai Selvi, K. Anitha Devi, C. Krithika and R. M. Kundhavai, “A Quiz-Based Trust Model with Optimized Resource Management”, Proceedings of 13th Asia-Pacific Grid Computer Systems Architecture Conference, Hsinchu, Taiwan, China, **(2008)** August 4-6.

[23] J. Sonnek, A. Chandra and J. B. Weissman, “Adaptive reputation-based scheduling on unreliable distributed infrastructures”, IEEE Transaction of Parallel and Distributed System, vol. 11, no. 18, **(2007)**.

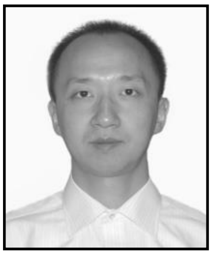
[24] Y. Wang and J. Wei, “VIAF: Verification-Based Integrity Assurance Framework for MapReduce”, 2011 IEEE International Conference on Cloud Computing, Washington DC, USA, **(2011)** July 4-9.

Authors



Yongzhi Wang, he received his BS and MS degree in School of Computer Science and Technology from Xidian University, China in 2004 and 2007, respectively. He is currently a PhD student in School of Computing and Information Sciences from Florida International University, USA. His research interests include big data, cloud computing security and outsourced computing security.

Jinpeng Wei, he received a PhD in Computer Science from Georgia Institute of Technology, Atlanta, GA in 2009. He is currently an assistant professor at the School of Computing and Information Sciences, Florida International University, Miami, FL. His research interests include malware detection and analysis, information flow security, cloud computing security, and file-based race condition vulnerabilities. He is a member of the IEEE and the ACM.



Yucong Duan, he received a PhD in Software Engineering from Institute of Software, Chinese Academy of Sciences, China in 2006. He is currently a full professor and vice director of Computer Science Department in Hainan University, P.R.China. His research interests include software engineering, service computing, cloud computing, and big data. He is a member of IEEE, ACM and CCF (China Computer Federation).