Modeling and Performance Evaluation of Colluding Attack in Volunteer Computing Systems

Kan Watanabe†, Nobuo Funabiki†, Toru Nakanishi† and Masaru Fukushi‡

***Abstract*—This paper proposes a model of a colluding attack in Volunteer Computing (VC), where some of participants are unreliable and may return incorrect results. The colluding attack is one of the most important issues to realize reliable VC because it may ruin the most basic assumption, i.e. "the majority in voting is correct". Especially, check-by-voting archives the largest efficiency by sorting reliable participants based on whether their results are the majority or not. Thus, if some incorrect results become the majority by colluding attack, it may have significant impacts on check-by-voting. In this paper, we perform a Monte Carlo simulation of VC using the proposed colluding model and evaluate the sabotage-tolerance performance of voting methods. Simulation results show that check-by-voting works well if colluding attack happens.**

***Index Terms*—Parallel Computing, Job Scheduling, Mathematical Modeling, Desktop Grids.**

I. Introduction

Volunteer computing (VC) is a type of Internet based parallel computing paradigm, which allows any participants in the Internet to contribute their idle computing resources towards solving large problems in parallel. By making it easy for anyone on the Internet to join a computation, VC makes it possible to build very large and high performance global computing environment with a very low cost. The most popular example of VC is SETI@home [1]. It is currently employing hundreds of thousands of volunteer participants. Nowadays, VC has a major role in scientific computations such as [2].

In VC, a management node (master) divides a computation into small independent jobs and allocates them to participant nodes (workers). Then, the workers execute the allocated jobs in parallel and return their results to the master. Since VC allows anyone in the Internet to join the computation, workers in VC may not be reliable as in grid computing. Those workers may behave erratically due to hardware/software failures or virus infections, or may behave maliciously to falsify the computation, each of which results in sabotage to the computation. Hence, there is a need for sabotage-tolerance mechanisms to improve the reliability of computation in VC.

The basic mechanism for sabotage-tolerance is voting, which collects multiple results for one job and determines the final result through a voting. After voting, collected results are divided into two groups, the majority and the minority. The minority results will be eliminated as incorrect ones and the majority ones are accepted as the final results because the majority seems to be correct.

Manuscript received December 8, 2011; revised January 9, 2012.
† Graduate School of Natural Science and Technology, Okayama University, Japan
‡ Graduate School of Information Sciences, Tohoku University, Japan

Two major voting methods are M-first voting and check-by-voting. M-first voting simply collects M matching results for each job. Because of its simpleness, M-first voting is used in major VC middle-ware such as BOINC[5]. Check-by-voting [3], [4] is the upgraded version of M-first voting based on the idea of weighted voting for performance improvement. Check-by-voting can guarantee that the error rate of computation ϵ is less than given value ϵ_{acc} by determining the weight well. Both of these two voting methods assume the basic requirement that the majority in voting is always correct.

However, the majority in voting may not be correct as pointed out in [7]. For example, VC which distributes large size jobs by P2P communication techniques requires worker-worker communication networks. Some malicious workers (saboteurs) also can communicate with each other using this network. They can generate the incorrect results with the same value and dominate majority of voting by their results. To deal with this threat of colluding attack, verifications and improvements of current sabotage-tolerance methods are needed.

In this paper, we perform simulation-based performance evaluations using a model of colluding attack for verifications of two voting methods, M-first voting and check-by-voting. First, we introduce a colluding probability c into the existing saboteurs' model. In case of $c = 0$, no colluding attack happens like the existing model, while all saboteurs always collude in $c = 1$ as the worst scenario. Then, we perform simulations of VC by changing the parameter c from 0 to 1 and evaluate throughputs and error rates of both M-first voting and check-by-voting.

II. VC model

A. Computation Model

The computation model of VC in this paper is the well known work-pool-based master-worker model. Figure 1 illustrates the model. Details of the model are described as follows.

- A VC system consists of a management node (master) and W different participant nodes (workers).
- A computation to be executed in the VC system is divided into N independent jobs.
- The master gives a job to each idle worker in each time step. Then, each worker executes an allocated job and returns the result to the master.
- The computation finishes when the time step reaches a given deadline P.

B. Sabotage model

In the modeling of workers' behaviors, the presence of saboteur is one of the most important components. To

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

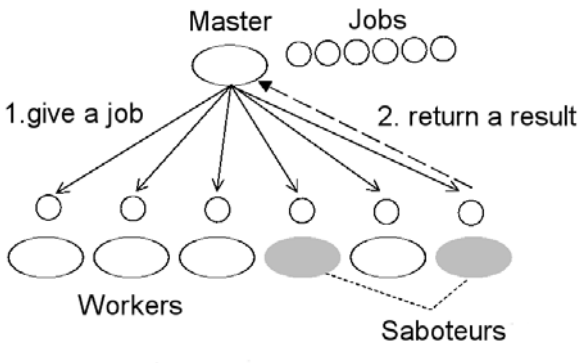


Fig. 1. Computation Model of VC systems

discuss the sabotage-tolerance problem of VC, Sarmenta [6] proposed the following sabotage model.

- A certain faulty fraction f of W workers is assumed to be saboteurs.
- Each saboteur is assumed to return an incorrect result with a constant probability s, which is known as the sabotage rate.
- The values of f and s are unknown to the master.

In the VC system with saboteurs, the performance is evaluated by two performance metrics, the throughput and the error rate. The throughput T is given as the number of finished jobs when the time step reaches the dead line P. The master may accept an incorrect result as final one of a job. In this case, the job is called an incorrect job. The error rate ϵ is given as the ratio of incorrect jobs to finished T jobs.

Using no sabotage-tolerance methods, each job is finished with one result whether the result is correct or not. If all W workers function at the same speed and execute a job in a unit time, T is given by $W \times P$ and ϵ is given by $W \times f \times s \times P/T = f \times s$. It is clear that ϵ is proportional to the number of saboteurs and sabotage rate s. Therefore, to reduce the error rate, some sabotage-tolerance methods must be used.

III. Sabotage-tolerance Methods

A. Basic approach

The most basic approach to sabotage-tolerance is the adoption of redundant computation. Voting is one of the redundant computation techniques and has been widely used in many areas such as fault-tolerant computing.

Two major voting methods are M-first voting and check-by-voting. In these methods, each job is replicated and allocated to several workers so that the master can collect several results and compare their values. The results collected for a job are then classified into groups (called result groups) according to the value of the results. The master decides which result group should be accepted as the final result through voting.

B. M-first-voting

In M-first voting, the result group which collects M matching (the same value) results first is accepted as the final result. Because of its simpleness, M-first voting is used in major VC middle-wares such as BOINC[5]. However, M-first voting has a serious drawback on the performance because the master always need to collect multiple (at least M) results for each job, whether each result is reliable or not. Compared to a VC system with no sabotage-tolerance methods, M-first voting degrades the throughput less than $1/M$. Because the minimum number of M is 2, M-first voting always degrades the throughput less than half.

C. Check-by-voting

1) Basic idea: Check-by-voting [3], [4] is the upgraded version of M-first voting based on the idea of weighted voting. In check-by-voting, the master gives a credibility value to each worker which represents how the worker is reliable. The master changes the credibility value with time based on the workers' behavior in the computation, e.g. the number of returned results. The credibility is used as its weight in voting so that the number of results collected for one job is not fixed. Note that, if the credibility of a worker is enough high, a job can be finished with just one result.This implies the throughput of check-by-voting can get above of the limitation of M-first voting, i.e. the half throughput.

The problems of check-by-voting are how to check the workers' reliability and to calculate the proper value of credibility. For these problems, the authors have proposed the following checking technique and the calculating formula. As a feature, the credibility in the proposed formula is given as a conditional probability so that the expected value of the error rate ϵ is less than an arbitrary value ϵ_{acc} for any cases of s and f. In the proposed method, this condition $\epsilon \leq \epsilon_{acc}$ is called the reliability condition.

2) Checking technique: Checking technique is the method to check whether a worker is reliable or not through each voting. When a job is finished through a voting, the collected results are divided into two groups, i.e. the majority results and minorities ones. In this technique, a worker w who returned the majority results is regarded as "a reliable worker" and the parameter k_w is counted up, which represents how many times w becomes the majority ($k_w = 0$ at the start of the computation). On the other hand, workers who returned the minority results are regarded as saboteurs. Those workers are eliminated from the system (blacklisting) and all returned results from those workers are invalidated (backtracking).

Using the checking technique, a worker who returns correct results continuously gets larger k_w and gains credibility. The results returned from such reliable workers tend to be accepted, which leads to reduce the error rate. Also, after the worker gains enough credibility, the throughput can be improved because the worker produces reliable results, which can finish one job with just one result.

3) Calculating Formula of Credibility: After checking the worker w, the credibility $C_W(w)$ is given based on the parameter k_w. Eq.(1) shows $C_W(w)$ when all W workers execute a job in a unit time. Because s and f are unknown to the master, the credibility should be given by the upper bound for s and f. The parameter f_{max} is the assumed ratio of saboteurs and given by the master so that the condition $f \leq f_{max}$ is satisfied.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$C_W(w) = \begin{cases} 1 - f_{max} & \text{if } k_w = 0, \\ 1 - \frac{f_{max}}{1-f_{max}} \times max(\frac{1}{k_w e(1-\epsilon_{acc})}, \epsilon_{acc}^{k_w}) & \text{otherwise.} \end{cases} \quad (1)$$

IV. Colluding Attack

A. The definition

In a recent VC, workers can communicate each other for several reasons [7]. For example, VC which distributes large size jobs by P2P communication techniques requires worker-worker communications. Saboteurs in such VC also can communicate with each other using this network. They can generate and return the incorrect results which have the same value to increase the probability that the master accepts saboteurs' results.

In this paper, the colluding attack is defined as returning incorrect results which have the same value. Figure 2 shows an example of colluding attack. For a job "1+1=?", saboteurs can communicate with each other and generate the same incorrect results "3", while the correct result is "2". In this case, the saboteurs' results "3" becomes majority, which are accepted by the master. As shown in this case, the majority in voting is not always correct if colluding attack happens.

B. Colluding Method

The collusion attack is classified into the following three categories depending on how to determine incorrect values.

- Direct colluding method
  The direct colluding method allows a saboteur to communicate with other saboteur directly and determine the incorrect value for every sabotaging. In this method, the number of saboteurs in each colluding group tends to be small since each saboteur must know informations about other saboteurs for direct communications.
- Indirect colluding method
  The indirect colluding method extends direct colluding to enable building an extensive colluding group of saboteurs. Instead of the direct communication, the saboteurs access to a third party, e.g. a colluding server, and obtain the value which should be set to the incorrect result. Thus, saboteurs can communicate with each other indirectly (via the colluding server) without knowing informations of other saboteurs. This method has a possibility of allowing all saboteurs to collude together, and increasing the error rate of a computation dramatically.
- Accidental colluding method
  The accidental colluding happens in case of accident. For example, incorrect values may match when a multiple-choice question is given as a job. This implies a colluding can happen depending on the type of job even if workers have no malicious intent.

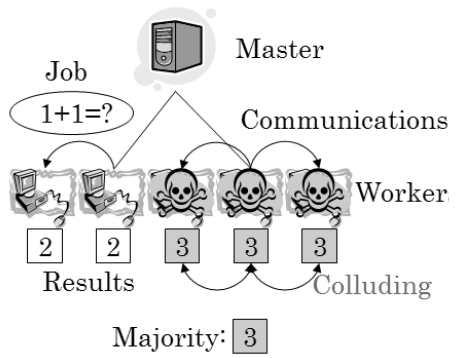


Fig. 2. An Example of Colluding Attack

C. Effect on Sabotage-tolerance

The current sabotage-tolerance methods do not assume the presence of colluding attacks since they focus on traditional VC, which does not need worker-worker commnications, and assume the incorrect values are random (not matches each other). If colluding happens, the performance metrics may be affected by the following reasons.

- Error rate ϵ
  The error rates of voting methods will become larger since the incorrect results may become the majority and be accepted by the master. Especially, in check-by-voting, such saboteurs may gain the credibility illegally by returning colluding results. This implies the reliability condition $\epsilon \leq \epsilon_{acc}$ may not be guaranteed because the credibility formula (eq.(1)) does not assume colluding attacks.
- Throughput T
  The throughput T of check-by-voting will become smaller since the correct results returned from non-saboteurs may be invalidated. This invalidation happens when non-saboteurs' results become minority in voting by colluding attacks.

D. Model of Colluding Attack

To evaluate the effect of colluding shown in Section IV-C, colluding attacks should be modeled. In this paper, we propose a colluding model based on the colluding rate c. The details of the proposed model is the following.

- When a saboteur generates an incorrect result with probability s, the saboteur performs colluding attack with the probability c and performs random attack with the probability $1 - c$.
- In case of the colluding attack, the saboteur generates a colluding result which has a predetermined incorrect value.
- In case of the random attack, the saboteurs generates an incorrect result having a random incorrect value.

Note that a saboteur generates a correct results with the probability $1 - s$.

The behavior of a saboteur is summarized as follows. For each job, a saboteur returns a colluding result with the probability sc, returns a random incorrect results with the probability $s(1 - c)$ and returns a correct result with the probability $1 - s$. Using the proposed model and the parameter c, we can generate any case of sabotaging and colluding. If $c = 0$, the computation model is the same as the existing one. On the other hand, the case of $c = 1$ generates the worst case, i.e. all saboteurs always collude and return the same incorrect results.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

V. Simulation

A. Evaluation Method

The error rate ϵ and the throughput T is evaluated in the VC model with saboteurs who may perform colluding attacks. This evaluation can cover any situation of VC by changing the colluding probability c from 0 to 1, including the existing VC model ($c = 0$). The evaluation targets of sabotage-tolerance methods are the following.

- 2-first-voting
  2-first-voting is an instance of M-first voting where M is set to the minimum number 2. As M becomes larger, the throughput tends to be smaller since each job is replicated for more workers. Thus, 2-first-voting shows the best throughput in M-first voting.
- Check-by-voting
  Check-by-voting is the proposed voting method as an improved version of M-first voting shown in Section III-C. For job scheduling, the existing method is used same as in [4].

B. Simulation Condition

Table I shows the simulation parameters in our evaluation. The assumed ratio of saboteurs f_{max} is set to 0.35 based on the real VC experiments [8]. The simulation conditions are as follows. All W workers are assumed to have the same speed and execute a job in a unit time as in [4]. In check-by-voting, the checking technique checks all votings and imposes two penalties to minority workers in voting, where all returned results from minority workers are invalidated (backtracking) and those workers can not get any more jobs from the master (blacklisting) [4].

C. Simulation Results

1) Colluding rate c: Fig.3 shows the error rate ϵ and the throughput T for colluding rate c. Fig.3(a) shows the error rate of 2-first voting increases with the colluding rate c. Since larger c increases the probability of returning colluding results, the colluding results tend to be the majority. Especially, in case of $s = 1.0$, the error rate exceeds the given parameter $\epsilon_{acc} = 0.05$ if c is greater than 0.4. This figure shows 2-first voting can not achieve the required reliability because the values of s and c are unknown to the master.

On the other hand, the error rate of check-by-voting is almost constant for any c and is smaller than the given ϵ_{acc}. To discuss this result, we should reconfirm the simulation conditions. In this simulation, all workers produce the same number of results and the ratio of incorrect results is $f \times s$. As long as f is smaller than 0.5, the incorrect results are the minority in all results, whether saboteurs collude or not. Even if the colluding workers become the majority in some jobs, their results for other jobs may be the minority. Therefore, those incorrect results become the minority in a voting at any point of the computation. When an incorrect result from a saboteur becomes the minority, all the results returned from the saboteur are invalidated by backtracking. The master reevaluates the past all votings of jobs which include the results from the saboteur. In such reevaluation, the correct results tend to be majority because the incorrect results have been invalidated. This is the reason why the error rate of check-by-voting becomes smaller.

TABLE I
Simulation Parameters

Computation deadline P	100
The number of workers W	100
The maximum number of jobs N	10000
The assumed ratio of saboteurs f_{max}	0.35
The actual ratio of saboteurs f	$0 \sim f_{max}$
Sabotage rate s	$0 \sim 1$
Acceptable error rate ϵ_{acc}	0.01, 0.05

Fig.3(b) shows the throughput of check-by-voting is almost constant for any c. This implies almost all saboteurs become the minority in any voting and take the penalty (blacklisting). In this situation, the rest few saboteurs do not affect the throughput even if s and c are large. On the other hand, the throughput of 2-first voting depends on c (from 2500 to 3200 at $s = 1$) because all saboteurs survive until the end of the computation. Note that the throughput of 2-first voting is always less than $W \times P/2 = 5000$ because 2-first voting requires at least two results for each job, while check-by-voting outperforms 2-first-voting for any c and achieves over $T = 6000$ for any c.

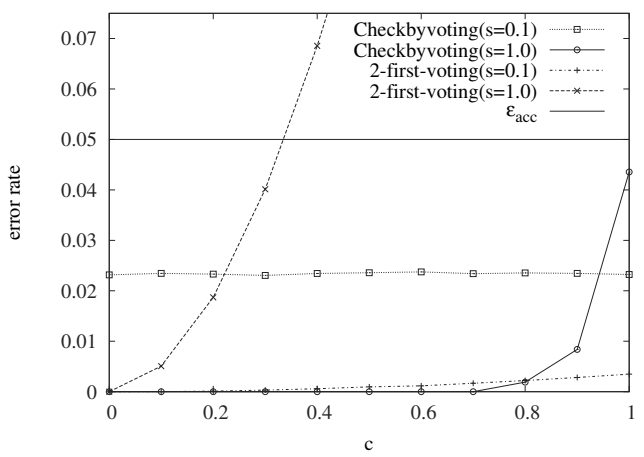
Fig.4 shows the simulation results in case of $\epsilon_{acc} = 0.01$. The result of 2-first-voting is the same as in Fig.3 because 2-first-voting does not use ϵ_{acc}. Fig.4(a) shows that check-by-voting guarantees the reliability condition $\epsilon \leq \epsilon_{acc}$ for any c, despite ϵ_{acc} changes from 0.05 to 0.01. This is a feature of check-by-voting. The credibility changes depends on given ϵ_{acc} (Eq.1) and then the number of collecting results (redundancy) changes depending on the credibility. Generally, if ϵ_{acc} becomes smaller, the credibility values become smaller and the required redundancy becomes larger to eliminate incorrect results through voting. As shown in Fig.4(b), the throughput also changes depending on ϵ_{acc}. For instance at $s = 1$, $T = 6500$ at $\epsilon_{acc} = 0.05$ and $T = 5800$ at $\epsilon_{acc} = 0.01$. This implies check-by-voting obtains less error rate to satisfy $\epsilon \leq \epsilon_{acc}$ at the expense of throughput.

2) The actual ratio of saboteurs f: Fig.5 shows the error rate and the throughput for the actual ratio of saboteurs f. In this figure, the colluding rate is assumed as the worst case, i.e. $c = 1.0$. Fig.5(a) shows the error rates of both 2-first voting and check-by-voting increase with f. Especially, for $s = 1$, the error rate rapidly increases with f. This implies the powerful influence of saboteurs on the error rate, who always return incorrect and colluding results. However, in check-by-voting, the reliability condition $\epsilon \leq \epsilon_{acc}$ is satisfied as long as f is smaller than the assumed f_{max}.

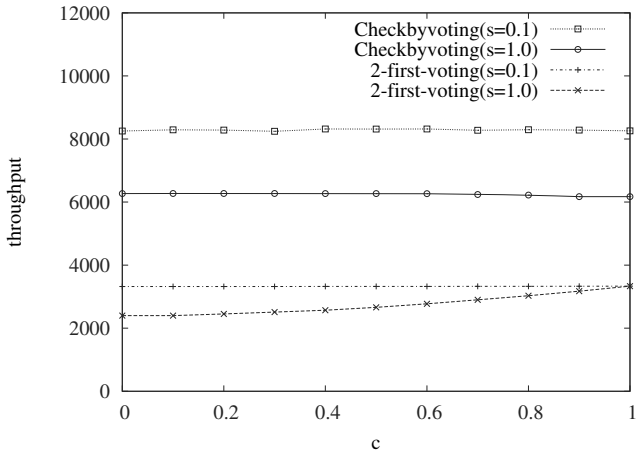
As shown in Fig.5(b), for larger f, the throughput of check-by-voting becomes smaller because the ratio of non-saboteurs $(1 - f)$ becomes smaller. In check-by-voting, the throughput mainly depends on the number of non-saboteurs because almost all saboteurs are blacklisted by the checking technique. On the other hand, the throughput of 2-first voting is almost constant for any f since saboteurs will not be checked. However, the throughput of check-by-voting is almost two times and outperforms 2-first voting for any f.

3) Sabotage rate s: Fig.6 shows the error rate and the throughput for the sabotage rate s. In this figure, the colluding rate c is set to 1 to assume the worst case. Fig.6(a) shows the error rate of check-by-voting has two local maximal

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

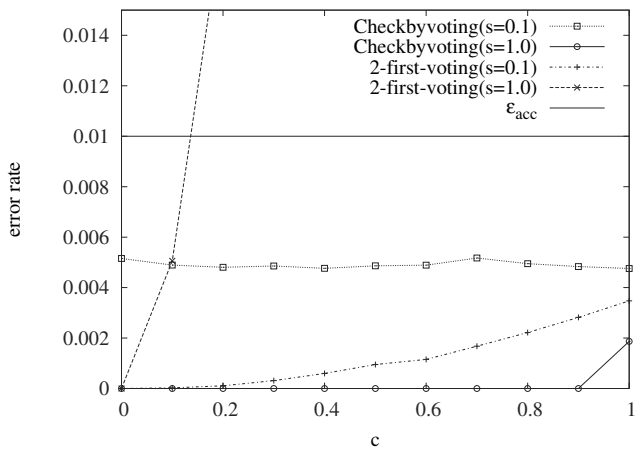


(a) Error rate ϵ

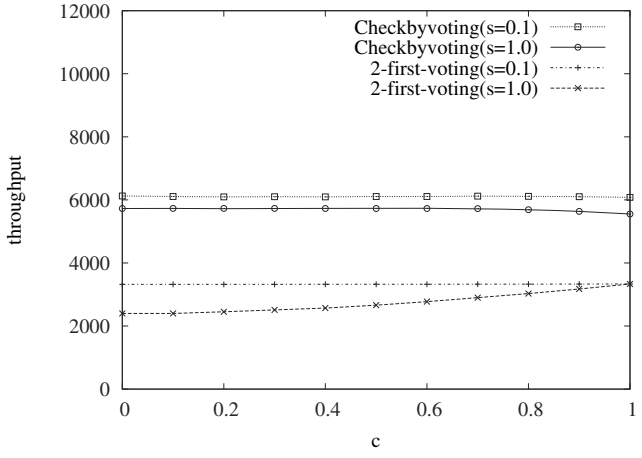


(b) Throughput T

Fig. 3. Error rate ϵ and throughput T for colluding rate c ($\epsilon_{acc} = 0.05$, $f = 0.35$)



(a) Error rate ϵ



(b) Throughput T

Fig. 4. Error rate ϵ and throughput T for colluding rate c ($\epsilon_{acc} = 0.01$, $f = 0.35$)

values at $s = 0.2$ and $s = 1.0$. This result implies the difficulty of guaranteeing the reliability condition, which means the worst case is not always at $s = 1.0$. To guarantee $\epsilon \leq \epsilon_{acc}$ for any s, we must investigate the reason why these local maximal points arise.

One of the reasons may be that larger s increases both the ratio of incorrect results to all results and the probability of being detected as saboteurs. First, at $s = 0$, the error rate is 0. As s increases, the error rate tends to increase because the number of incorrect results increases. However, larger s also increases the probability of producing incorrect results, which may be minority ones in the checking technique. Thus, when s reaches a certain value ($s = 0.2$ in this case), the error rate becomes smaller as s increases due to the invalidation of incorrect results by backtracking. After s reaches 0.7, the ratio of incorrect results becomes enough large so that those incorrect results tend to be majority. Therefore, the error rate increases with s between $s = 0.7$ to 1.0. The above speculation seems to be true in our simulation. However, the condition of arising such local minimal is not known and should be in our future work.

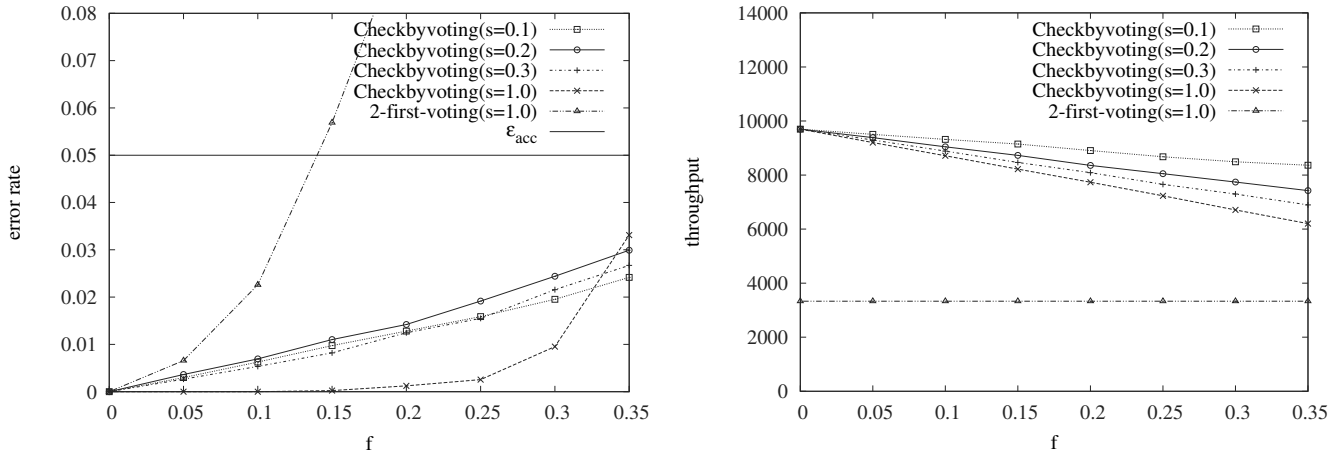
VI. Conclusion

In this paper, we perform simulation-based performance evaluation of both M-first voting and check-by-voting. For the evaluation we introduce a colluding probability c into the existing saboteurs' model and propose colluding model. In case of $c = 0$, no colluding attack happens as the existing model, while all saboteurs always collude in $c = 1$ as the worst scenario. The simulation results show that check-by-voting guarantees the reliability condition $\epsilon \leq \epsilon_{acc}$ for any s,f and c, while the throughput always outperforms that of 2-first voting. However, there are still unknown parts of colluding. One of our future work is analyzing these behaviors and guaranteeing the condition $\epsilon \leq \epsilon_{acc}$ in any case of saboteurs' behavior.

Acknowledgment

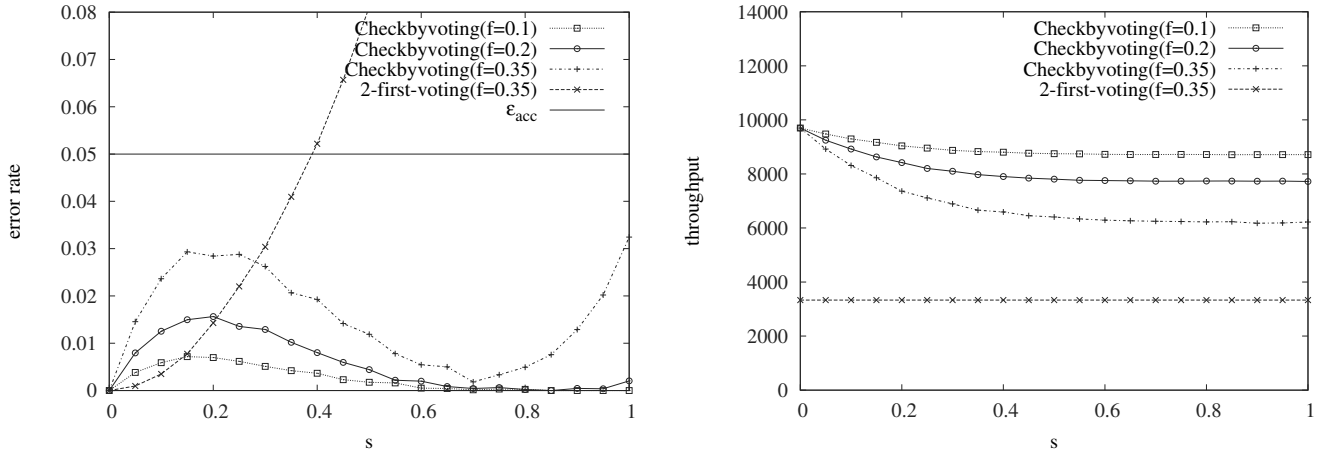
This work was supported by KAKENHI 23800041.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



(a) Error rate ϵ (b) Throughput T

Fig. 5. Error rate ϵ and throughput T for the actual ratio of saboteurs f ($\epsilon_{acc} = 0.05$, $c = 1.0$)



(a) Error rate ϵ (b) Throughput T

Fig. 6. Error rate ϵ and throughput T for sabotage rate s ($\epsilon_{acc} = 0.05$, $c = 1.0$)

REFERENCES

[1] http://setiathome.ssl.berkeley.edu/
[2] B. Knispel. et.al, “Pulsar Discovery by Global Volunteer Computing”, Science, Vol.329, no.5994, 2010.
[3] K. Watanabe and M. Fukushi, “Generalized Spot-checking for Reliable Volunteer Computing”, IEICE Transactions on Information and Systems, Vol.E93-D, No.12, pp.3164 – 3172, 2010.
[4] K. Watanabe, M. Fukushi and M. Kameyama, “Adaptive Group-Based Job Scheduling for High Performance and Reliable Volunteer Computing”, Journal of Information Processing, Vol.19, pp.39 – 51, 2011.
[5] http://boinc.berkeley.edu/
[6] Luis F. G. Sarmenta, “Sabotage-Tolerance Mechanisms for Volunteer Computing Systems”, Future Generation Computer Systems, Vol. 18, Issue 4, pp.561-572, 2002.
[7] F. Araujo, J. Farinha, P. Domingues, G.C. Silaghi, D. Kondo, “A maximum independent set approach for collusion detection in voting pools”, Journal of Parallel and Distributed Computing, Vol. 71 (10), pp. 1356 – 1366, 2011
[8] D. Kondo, F. Araujo, P. Malecot, P. Domingues, L. M. Silva, G. Fedak, F. Cappello, “Characterizing Error Rates in Internet Desktop Grids”, 13th European Conference on Parallel and Distributed Computing, pp. 361–371, 2007.

ISBN: 978-988-19251-9-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)