PAPER *Special Section on Parallel and Distributed Computing and Networking*

Generalized Spot-Checking for Reliable Volunteer Computing

Kan WATANABE[†a)], ***Nonmember and*** **Masaru FUKUSHI**[†b)], ***Member***

SUMMARY While volunteer computing (VC) systems reach the most powerful computing platforms, they still have the problem of guaranteeing computational correctness, due to the inherent unreliability of volunteer participants. Spot-checking technique, which checks each participant by allocating spotter jobs, is a promising approach to the validation of computation results. The current spot-checking is based on the implicit assumption that participants never distinguish spotter jobs from normal ones; however generating such spotter jobs is still an open problem. Hence, in the real VC environment where the implicit assumption does not always hold, spot-checking-based methods such as well-known credibility-based voting become almost impossible to guarantee the computational correctness. In this paper, we generalize spot-checking by introducing the idea of imperfect checking. This generalization allows to guarantee the computational correctness under the situation that spot-checking is not fully-reliable and participants may distinguish spotter jobs. Moreover, we develop a generalized formula of the credibility, which enables credibility-based voting to utilize check-by-voting technique. Simulation results show that check-by-voting improves the performance of credibility-based voting, while guaranteeing the same level of computational correctness.

key words: *spot-checking, sabotage-tolerance, credibility-based mechanisms, volunteer computing*

1. Introduction

Volunteer computing (VC) is a type of Internet based parallel computing paradigm, which allows any participants on the Internet to contribute their idle computing resources towards solving large parallel problems. 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. However, VC still have the problem of guaranteeing computational correctness [1], [2]. Since VC allows anyone to join a computation, participant nodes are not always reliable as in well-controlled grid computing systems. Malicious participants (called saboteurs) may sabotage job execution by returning erroneous results. Therefore, VC systems must alleviate the effect of sabotaging.

Two basic categories of the sabotage-tolerance techniques for VCs are voting and checking. In the voting technique, a job is replicated and allocated to several participants for a redundant computation. Examples of the voting techniques include majority and m-first votings, in which a fixed number of results and m matching results are collected to decide the final result, respectively. Since majority voting is simple but inefficient [3], the most popular VC middle-ware BOINC [4] employs m-first voting.

In the checking technique, a spotter job which is different from a normal computational job is used to check worker's behavior directly. Worker is judged credible or not by verifying the result of a spotter job; hence, the correct result for each spotter job must be known in advance. Examples of the checking techniques include quiz [5] and spot-checking [6]. In quiz, returned results are simply neglected if the worker fails a checking. In addition to this, spot-checking allows to guarantee the computational correctness mathematically based on the total number of checking each worker survives.

Combining the advantages of m-first voting and spot-checking, Sarmenta [6] proposed a novel sabotage-tolerance method, referred to as credibility-based voting in this paper. This method estimates the credibility of results using the function of spot-checking and allows to guarantee correctness of voted results keeping the degree of redundancy as low as possible. Thus, this method is promising for reliable and high-performance VCs.

In spot-checking-based sabotage-tolerance methods, such as credibility-based voting [6], the results of checking are utilized in the estimation of sabotaging frequency. Those methods work well as long as the saboteurs never distinguish spotter jobs; that is, they implicitly assume that the results of spot-checking are fully-reliable. However, generating such indistinguishable spotter jobs is still an open and tough problem because it requires a huge number of reliable nodes or computation time [5] to prepare a number of various spotter jobs and the correct results, which is impractical in real VC systems. The result of spot-checking can not be fully-reliable in real VCs. Saboteurs may return correct results only for spotter jobs, while sabotaging normal jobs to disturb the computation. This makes the spot-checking-based methods useless for guaranteeing the computational correctness of VCs.

In this paper, we introduce the idea of generalized spot-checking to guarantee the computational correctness under the situation that the result of spot-checking is not fully-reliable. This is a feature that no previous methods have considered. The main contribution of our work are the following. (1) We develop a generalized formula of the credibility by introducing the probability c that represents the accuracy of spot-checking. This generalization allows credibility-based voting to guarantee the computational cor-

Manuscript received February 1, 2010.
Manuscript revised June 10, 2010.

[†]The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980–8579 Japan.
a) E-mail: can@ecei.tohoku.ac.jp
b) E-mail: mfukushi@ecei.tohoku.ac.jp

DOI: 10.1587/transinf.E93.D.3164

rectness even if saboteurs distinguish spotter jobs. Through simulations, we verify the accuracy of the generalized formula. (2) We enable to apply check-by-voting method to credibility-based voting to improve the performance of VCs, while guaranteeing the computational correctness. This application has become possible because of our generalized formula. Through simulations, we compare the performance of VCs with and without check-by-voting.

The rest of this paper is organized as follows. Section 2 shows the computational model of VC systems and sabotage-tolerance mechanisms. Section 3 proposes generalized spot-checking and check-by-voting techniques. Section 4 verifies the accuracy of generalized spot-checking and the improvements of performance of VCs with check-by-voting. Finally, conclusions are discussed in Sect. 5.

2. Volunteer Computing Systems

2.1 Computational Model

A well known work-pool-based master-worker model [4], [6] is assumed as the computation model of VC systems. This model is used in almost all VC systems practically. 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.
- At the start of the computation, all jobs are placed in a work pool of the master.
- The master gives a job to each idle worker.
- Each worker executes the allocated job and returns the result to the master. During their execution, no communication exists among workers because jobs are mutually independent.

Figure 1 illustrates the master-worker model of VC systems. To produce a sabotage model, a certain faulty fraction f of the W workers are assumed to be saboteurs who may return erroneous results. Each saboteur attempts to return an erroneous result with a constant probability s, which is known as the sabotage rate [6]. Note that if a saboteur distinguishes spotter job, the saboteur may give up sabotaging to avoid detection by checking. The values of f and s are

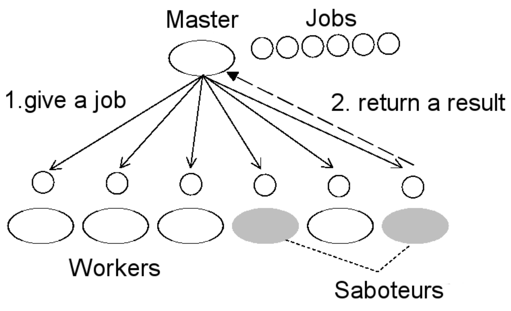


Fig. 1 Computation model of VC systems.

unknown to the master.

The master manages the execution of a computation and allocates unfinished jobs to idle workers. Computation finishes when all jobs are finished. Jobs that finish with erroneous results are called erroneous jobs. At the end of the computation, an error rate ϵ can be calculated as the ratio of erroneous jobs to all jobs.

Using no sabotage-tolerance mechanisms, the error rate increases in proportion to the number of saboteurs. Let T be the time taken to finish all jobs of a computation, i.e. computation time. If all workers function at the same speed and execute a job in a unit time, then T is given by $\lceil N/W \rceil$ unit times and error rate ϵ is given by $N \times f \times s/N = f \times s$. It is clear that ϵ is proportional to the fraction f and sabotage rate s. Therefore, to reduce the error rate, some sabotage-tolerance mechanism must be used.

2.2 Sabotage-Tolerance Mechanisms

2.2.1 Voting and Checking

The basic sabotage-tolerance mechanisms are voting and checking.

(1) Voting: Each job is replicated and allocated to several workers so that a 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 of the job through voting. Two major voting methods are majority and m-first votings.

- Majority voting: The result group which collects the largest number of results is accepted as the final result.
- m-first voting: The result group which collects m matching results first is accepted as the final result.

By a simple principle, these voting methods are widely used in real VC systems such as BOINC [4].

(2) Checking: To check whether a worker is a saboteur or not, a master allocates a spotter job whose correct result is already known to the master. The master can catch the worker as a saboteur if a worker returns an erroneous result for the spotter job. Two major checking methods are quiz [5] and spot-checking [6].

- Quiz: Spotter jobs are inserted into a package of normal jobs and are allocated to a worker. If the worker survives all spotter jobs in the package, the results of jobs in the package are accepted.
- Spot-checking: A spotter job is allocated to a worker directly. When the worker survives the checking, the number of checking is counted up. If the worker survives checking enough times, all the results returned by the worker are accepted.

When the master catches a saboteur by a checking, the following two methods can be used:

- Backtracking: The master backtracks (invalidates) all results returned by the saboteur because each may be an erroneous one.
- Blacklisting: The master puts saboteur's identification information (e.g. IP address) on a blacklist to prevent the saboteur from returning results or getting any more jobs.

Backtracking and blacklisting can be used simultaneously for efficient sabotage-tolerance.

2.2.2 Credibility-Based Voting

In traditional voting methods such as m-first voting, each job requires a fixed number of matching results (at least two or more), no matter how reliable each result is. Consequently, the performance of VC systems diminishes to less than half. Using only spot-checking is also inefficient, since it requires the executions of huge number of spotter jobs, which wastes the computational resources of workers.

To improve these inefficiency, Sarmenta [6] proposed a new sabotage-tolerance method (called "credibility-based voting"), by combining m-first voting and spot-checking. In this method, each system element such as worker, result, result group, and job is assigned a credibility value that represents its correctness. Every worker and result has different credibility, which affects the credibility of the result groups. A job is finished when the credibility of any result group of the job reaches a threshold θ. Therefore, the necessary number of matching results to finish a job is not fixed and is generally smaller than that of the m-first voting.

Furthermore, based on spot-checking, this method can guarantee that the mathematical expectation of the error rate is below an arbitrary acceptable error rate ϵ_{acc} by setting two conditions: (1) threshold $\theta = 1 - \epsilon_{acc}$, and (2) unknown parameter f satisfies $f \leq f_{max}$, where f_{max} is the upper limit of f and is a known value. This condition implies that the number of saboteurs in W workers is, at most, $\lfloor f_{max} \times W \rfloor$. For spot-checking, each worker gets a spotter job with probability q, which is known as spot-check rate.

In credibility-based voting, the definitions of the credibility are as follows [6], [9]. Since parameters s and f are unknown to the master, the credibility of a worker w, $C_W(w)$, is given as the lower bound of the probability that a result returned by w is correct. The master calculates $C_W(w)$ based on the number of checking w survives under the assumption that w never distinguish spotter jobs. If worker w survives spot-checking k times, $C_W(w)$ is given by Eq. (1)

$$C_W(w) = \begin{cases} 1 - f_{max}, & \text{if } k = 0, \\ 1 - \dfrac{f_{max}}{(1 - f_{max}) \times ke}, & \text{otherwise} \end{cases} \quad (1)$$

where e is Napier's constant.

The credibility of a result r produced by worker w is equal to $C_W(w)$. The credibility of a result group G_a, $C_G(G_a)$, is given as the conditional probability that the results in G_a is correct and all other results are erroneous ones. Thus, the result group which contains more results tends to have larger credibility. The credibility of a job j, $C_J(j)$, is defined to be $C_G(G_x)$, where G_x has a maximum credibility in all result groups of job j. When the credibility of job j reaches the threshold θ, the result group G_x is accepted as the final result of job j, and job j is finished.

3. Generalized Spot-Checking and Check-by-Voting

3.1 Motivation and Key Idea

Although spot-checking-based mechanism is a promising approach for the sabotage-tolerance of VC systems, it still has some practical issues, e.g. generating spotter jobs. Therefore, the current VC systems such as BOINC [4] mainly employ a simple but inefficient voting method. The goal of our work is realizing high performance and reliable VC systems by a practicable and efficient sabotage-tolerance method.

In the current spot-checking-based mechanisms such as credibility-based voting [6], it is implicitly assumed that the allocation of spotter jobs is never distinguished by saboteurs. Especially, in credibility-based voting, the credibility of a worker is calculated based on this assumption as shown in Eq. (1). However, generating such spotter jobs is still an open problem because generating many kinds of spotter jobs requires more job executions on reliable nodes, which in turn degrades the performance of the system. If there is a small repertoire of spotter jobs, a saboteur may distinguish spotter jobs easily because the same spotter job is repeatedly allocated to the saboteur. In this case, credibility-based voting becomes almost impossible to guarantee the reliability condition $\epsilon \leq \epsilon_{acc}$ because a saboteur can slip through spot-checking and gain higher credibility illegally by returning correct results only for spotter jobs.

The key idea of solving the aforementioned problem is generalization of spot-checking technique by introducing the idea of imperfect checking. We define a new parameter c, which represents the accuracy of spot-checking. This generalization allows to use imperfect spot-checking for spot-checking-based mechanisms, while maintaining the features of those methods (i.e. guaranteeing the computational correctness in credibility-based voting).

In the generalized spot-checking, a checking can detect an attempt of sabotaging by a saboteur with probability c. However, the actual value of c for each spot-checking is unknown to the master like the sabotage rate s and the fraction f. One way to solve this problem is using the lower bound of the probability, c_{min}, as is done for f [6]. The value of c_{min} is proportional to the number of spotter jobs a master can prepare for the computation. If saboteurs never distinguish spotter jobs, the generalized spot-checking sets c_{min} to 1 and works as well as the original (perfect) spot-checking. On the other hand, c_{min} is set to a value smaller than 1 if the spot-checking is imperfect, i.e. the master can not prepare a variety of spotter jobs.

From the next section, we describe two methods applying the proposed generalized spot-checking technique to credibility-based voting. First, we give a new calculation formula of the credibility using probability c to guarantee the reliability condition $\epsilon \leq \epsilon_{acc}$ even when spotter jobs are distinguished by saboteurs. Next, we propose a new checking technique, which utilizes the result of each voting as an imperfect checking to improve the performance of credibility-based voting.

3.2 Calculating Formula of Credibility

If a master uses generalized spot-checking and blacklisting, the master can calculate the credibility of worker w who survives spot-checking k times as follows. Let c_i $(i = 1, \ldots, k)$ be the accuracy of i-th spot-checking, that is, w can not distinguish i-th spot-checking as a spotter job with probability c_i. In this situation, w distinguishes the i-th spot-checking and return a correct result with probability $1 - c_i$. On the other hand, even if w does not distinguish, w returns a correct result with probability $1 - s$ because the job may be spotter one. Thus, the probability that w survives i-th spot-checking is $(1 - c_i) + c_i(1 - s) = 1 - s \times c_i$. Here, c_{min} is a lower bound of c_i ($c_{min} \leq c_i$ for $i = 1, \ldots, k$).

Since parameters s and f are unknown to the master, the credibility of w, i.e. $C_{W_{gen}}(w)$, is given as the lower bound of the probability that w returns a correct result for any s and f. To calculate $C_{W_{gen}}(w)$, we define $\epsilon_{result}(s, f, k)$ as the probability that an arbitrary result returned by a worker who survives k spot-checking is erroneous one, under the situation that $(1 - f) \times W$ non-saboteurs always return correct results and $f \times W$ saboteurs return erroneous results with probability s.

When $k = 0$, the upper bound of $\epsilon_{result}(s, f, k)$ is given by Eq. (2) since $0 \leq s \leq 1$ and $0 \leq f \leq f_{max}$.

$$\begin{aligned}\epsilon_{result}(s, f, 0) &= s \times f \\ &\leq f \\ &\leq f_{max}. \end{aligned} \tag{2}$$

When $k \neq 0$, the upper bound of $\epsilon_{result}(s, f, k)$ is given as follows. The probability that a saboteur survives all k spot-checking is $\prod_{i=1}^{k}(1 - s \times c_i)$. Since w is either a non-saboteur or a saboteur who survives k spot-checking, $\epsilon_{result}(s, f, k)$ is given by Eq. (3).

$$\epsilon_{result}(s, f, k) = s \times \frac{f \times \prod_{i=1}^{k}(1 - s \times c_i)}{(1 - f) + f \times \prod_{i=1}^{k}(1 - s \times c_i)}. \tag{3}$$

In Eq. (3), $0 \leq f$ and $0 \leq (1 - s \times c_i)$ for all i; then, $\epsilon_{result}(s, f, k)$ satisfies Eq. (4).

$$\epsilon_{result}(s, f, k) \leq s \times \frac{f \times \prod_{i=1}^{k}(1 - s \times c_i)}{(1 - f)}. \tag{4}$$

In Eq. (4), $(1 - s \times c_i) \leq (1 - s \times c_{min})$ for all i; then, $\epsilon_{result}(s, f, k)$ satisfies Eq. (5).

$$\epsilon_{result}(s, f, k) \leq \frac{f}{1 - f} \times s(1 - s \times c_{min})^k. \tag{5}$$

In Eq. (5), Eq. (6) is satisfied since $0 \leq f \leq f_{max}$.

$$\epsilon_{result}(s, f, k) \leq \epsilon_{result}(s, f_{max}, k). \tag{6}$$

Also, Eq. (7) is satisfied since $s(1 - s \times c_{min})^k$ has a maximum point at $s = \frac{1}{c_{min}(1+k)}$. Here, $\frac{(\frac{k}{1+k})^k}{1+k}$ is strictly and asymptotically bounded from above by $\frac{1}{ke}$, where e is Napier's constant [9].

$$\begin{aligned} s(1 - s \times c_{min})^k &\leq \frac{1}{c_{min}(1 + k)}\left(1 - \frac{1}{1 + k}\right)^k \\ &= \frac{1}{c_{min}(1 + k)}\left(\frac{k}{1 + k}\right)^k \\ &\leq \frac{1}{kec_{min}} \end{aligned} \tag{7}$$

From Eq. (7), Eq. (8) is satisfied between $0 \leq s \leq 1$.

$$s(1 - s \times c_{min})^k \leq min\left(1, max\left(\frac{1}{kec_{min}}, (1 - c_{min})^k\right)\right). \tag{8}$$

Finally, we obtain the upper bound of $\epsilon_{result}(s, f, k)$ as Eq. (9) when $k \neq 0$ from Eqs. (5), (6) and (8).

$$\begin{aligned} &\epsilon_{result}(s, f, k) \\ &\quad \leq \frac{f_{max}}{1 - f_{max}} \times min\left(1, max\left(\frac{1}{kec_{min}}, (1 - c_{min})^k\right)\right). \end{aligned} \tag{9}$$

From Eqs. (2) and (9), $C_{W_{gen}}(w)$ is given by Eq. (10) as the lower bound of the probability that a result returned by w is correct for any s and f.

$$C_{W_{gen}}(w) = \begin{cases} 1 - f_{max}, & \text{if } k = 0, \\ 1 - \frac{f_{max}}{1 - f_{max}} \times min\left(1, max\left(\frac{1}{kec_{min}}, (1 - c_{min})^k\right)\right), & \text{otherwise.} \end{cases} \tag{10}$$

When the master uses imperfect spot-checking, the reliability condition $\epsilon \leq \epsilon_{acc}$ can be guaranteed by using Eq. (10) instead of Eq. (1).

In the derivation of Eq. (10), we suppose that if a saboteur distinguishes a spotter job, then it returns a correct result. This is quite natural behavior to avoid being caught by the master. On the other hand, against this assumption, some saboteurs may return erroneous results even if it distinguishes spotter jobs. Even in such situation, Eq. (10) is applicable because it is derived based on the upper bound of the probability of surviving spot-checking ($1 - sc_{min}$ in Eq. (5)). Returning erroneous results for distinguished spotter jobs reduces the probability of surviving spot-checking but does not affect the upper bound.

3.3 Check-by-Voting

While the generalization of spot-checking allows the master to use imperfect checking, it also allows the master to utilize a technique, referred to as check-by-voting in this paper. The idea of check-by-voting is common to all voting methods and summarized as follows. When a job is finished through a voting, a result in the majority result group of the job is correct with a certain probability P. Here, by supposing that the finished job is a spotter one, the worker who returns the majority result can be assumed to survive a checking (with accuracy $c = P$). On the other hand, the worker who returns the minority result can be assumed as a saboteur.

Although the idea of check-by-voting is well-known, adapting the idea to credibility-based voting has a serious problem to guarantee the computational correctness. That is, the result of check-by-voting is imperfect; an erroneous result becomes the majority with probability $1-P$. A check-by-voting is an imperfect checking with accuracy P.

Since the generalization of spot-checking allows to use imperfect checking, we can use check-by-voting while guaranteeing the condition $\epsilon \leq \epsilon_{acc}$. Although the actual value of the probability P is unknown, the lower bound of P is guaranteed in credibility-based voting (i.e. $\theta \leq P$). In the case of check-by-voting, a saboteur survives a voting with probability $1-sP$. Hence, $\epsilon_{result}(s, f, k)$ for check-by-voting is given by

$$\epsilon_{result}(s, f, k) = s \times \frac{f \times (1-sP)^k}{(1-f) + f \times (1-sP)^k}. \tag{11}$$

Equation (11) is the same as Eq. (3) in the sense that the probability of surviving spot-checking is $(1 - sP)$ instead of $(1 - sc_i)$. Other equations and the upper bound of $\epsilon_{result}(s, f, k)$ can be derived in the same way for spot-checking except that P is bounded by θ. Thus, the credibility of a worker who survives check-by-voting k times is given by Eq. (12) by setting $c_{min} = \theta$ in Eq. (10).

$$C_{W_{check\text{-}by\text{-}voting}}(w) = \tag{12}$$
$$\begin{cases} 1-f_{max}, & \text{if } k = 0, \\ 1-\dfrac{f_{max}}{1-f_{max}} \times min\left(1, max\left(\dfrac{1}{ke\theta}, (1-\theta)^k\right)\right), & \text{otherwise.} \end{cases}$$

Using check-by-voting, the performance of credibility-based voting can be improved by two reasons. The first reason is that check-by-voting increases the number of checking each worker gets. Since the results from non-saboteurs tend to be majority ones, those workers can gain higher credibility, resulting in reduction of redundant job executions and the computation time. The second one is that check-by-voting allows the master to check workers without allocating spotter jobs. Since spotter jobs are extra ones, the decrease of allocating spotter jobs leads to smaller computation time.

Figure 2 shows the algorithm of check-by-voting. In this algorithm, the states of "used" and "not used" are defined for each result, whereas the states of "finished" and "unfinished" are defined for each job. The state of result represents whether the result is already used for evaluating worker's credibility in check-by-voting. To avoid duplicate

```
 1  //$S(w_i)$ is the set of normal jobs allocated to a worker $w_i$
 2  //$r_j(w_i)$ is the result of job $j$ returned by a worker $w_i$
 3  //$Q$ is a queue of jobs for checking
 4
 5  procedure check-by-voting($j$):
 6  update the credibility of job $C_J(j)$;
 7  If($C_J(j) \geq \theta$) then
 8    mark $j$ as finished;
 9    For each result $r_j(w_i)$ of $j$ do
10      If($r_j(w_i)$ is not used) then
11        If($r_j(w_i)$ is majority) then
12          $k(w_i)$=$k(w_i)+1$;
13          update the credibility of $w_i$;
14        else
15          blacklisting $w_i$;
16          invalidate all results returned by $w_i$;
17        end if
18        mark jobs in $S(w_i)$ as unfinished;
19        add jobs in $S(w_i)$ into $Q$;
20        mark $r_j(w_i)$ as used;
21      end if
22    end for
23  end if
24
25  If ($Q$ is empty) then
26    exit;
27  else
28    Pop a job $j_{next}$ from $Q$;
29    call procedure check-by-voting($j_{next}$);
30  end if
31  end procedure
```

Fig. 2 Check-by-voting algorithm.

evaluation, only when the result is not used, the worker who returns the majority result can gain the credibility.

In this algorithm, when a master collects a result of job j from a worker, the master calls procedure check-by-voting(j). First, the master calculates the credibility of job j and marks j as finished if the credibility reaches the threshold θ (lines 7–8). Let $r_j(w_i)$ be the result of job j returned by worker w_i. When w_i returns $r_j(w_i)$, $r_j(w_i)$ is marked as "not used" for check-by-voting.

For all results of j ($r_j(w_1)$, $r_j(w_2)$, ...), the master repeats the checking process (lines 9–22). If $r_j(w_i)$ is the majority result of job j, the master assumes that worker w_i survives a spot-checking and increases the credibility of w_i (lines 11–13). On the other hand, if $r_j(w_i)$ is minority, the master blacklists w_i and invalidate all results returned by w_i (lines 15–16). Let $S(w_i)$ be the set of jobs allocated to w_i. Since both updating credibility of w_i and invalidating results returned by w_i affect the credibility of jobs in $S(w_i)$, the master adds them into queue Q to for rechecking (lines 18–19). Since the check-by-voting assumes each job allocation as one spot-checking, the master marks the used result in the check-by-voting to avoid duplicate evaluation of worker's credibility (line 20). After checking all results of j, there may be some jobs in queue Q (line 25). As long as the queue is not empty, the master repeats check-by-voting recursively (lines 28–29).

Spot-checking and check-by-voting can be used simultaneously. In this case, the value of c_{min} in Eq. (10) is set to the smaller value, either the lower bound of c or θ. The value of k is the total number of spot-checking and check-by-voting a worker survives.

4. Performance Evaluation

4.1 Simulation Conditions

In this section, we evaluate the accuracy of proposed formula Eq. (10) and the effectiveness of proposed check-by-voting method through the simulation of VCs. First, in Sect. 4.2.1, we compare error rates ϵ of credibility-based voting using the original formula Eq. (1) with those using the proposed formula Eq. (10). Error rates ϵ are evaluated using the round-robin job scheduling method since it is originally employed in [6]. Then, in Sect. 4.2.2, we evaluate the performance of credibility-based voting with check-by-voting in terms of error rates ϵ and computation times T. Because computation time is highly dependent on job scheduling algorithm, we employ three job scheduling algorithms: the round-robin, the random and the $ENR + ECJ$ [8] methods. The $ENR+ECJ$ method [8] is known as an efficient job scheduling method for credibility-based voting. Error rates ϵ and computation times T are evaluated as the average of 1000 simulation results.

The parameters used in our simulation are shown in Table 1. Because some parameters such as f and s are unknown to the master and are uncontrollable, we use variant values for such parameters to simulate various cases of VCs. The upper limit of f, i.e. f_{max}, is set to 0.35 reflecting the result of an experiment in a real VC environment [1]. Because the optimal value of q depends on f and s [7], q is set to 0.1 as in [6]. In addition, we employ the blacklisting and the backtracking methods and assume that the backtracking method invalidates all results of jobs returned by the detected saboteur even if the jobs are finished.

The simulator developed for this experiment follows that in [6]. At the start of each iteration of simulation, we create a list of W worker entries and randomly select $f \times W$ workers to be saboteurs. We then simulate a computation done by these workers by going through the list in round-robin manner. In each unit time, a worker contacts the master to return a result (for the job it received in the previous turn) and to get a new job. This assumes that all workers run at the same speed.

Table 1 Simulation parameters.

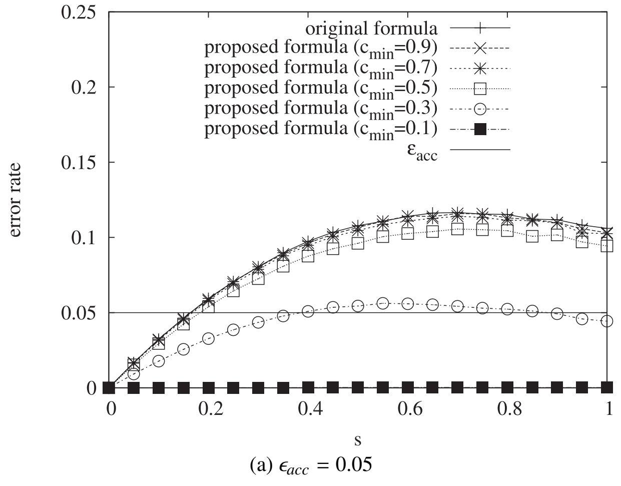
# of jobs (N)	10000
# of workers (W)	100
faulty fraction (f)	$0 \sim f_{max}$
upper limit of f (f_{max})	0.35
sabotage rate (s)	$0 \sim 1$
checking accuracy (c)	$0 \sim 1$
lower limit of c (c_{min})	$0 \sim 1$
acceptable error rate (ϵ_{acc})	0.01, 0.05
spot-check rate (q)	0.1

4.2 Simulation Results

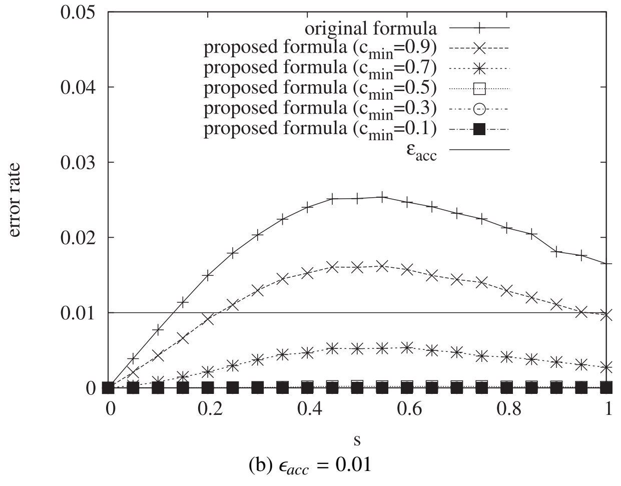
4.2.1 Error Rates of Original and Proposed Formula

Figure 3 shows error rate ϵ as a function of s. This figure clearly shows that the error rates of credibility-based voting with the original formula exceed the required rate ϵ_{acc} when s becomes large. Since the value of s is unknown to the master, this result indicates that the credibility-based voting with the original formula can not guarantee the condition $\epsilon \leq \epsilon_{acc}$. This is true because the original formula assumes that the spotter jobs are never distinguished by saboteurs, while saboteurs in the simulator sometimes ($c = 0.1$ in this case) distinguish them as in real VCs and gain improper credibility. On the other hand, the error rate with the proposed formula at $c_{min} = 0.1$ is less than ϵ_{acc} for any s. This result indicates that the proposed formula enables the master to calculate proper credibility of workers even if saboteurs distinguish spotter jobs.

This figure shows that, when the condition $c_{min} \leq c$ is not satisfied, the error rate may exceed ϵ_{acc} even if the proposed formula is used. For example, in Fig. 3 (a), the error rates in some cases exceed $\epsilon_{acc} = 0.05$ when $c_{min} = 0.9$, 0.7, 0.5 and 0.3. This is true because the master gives

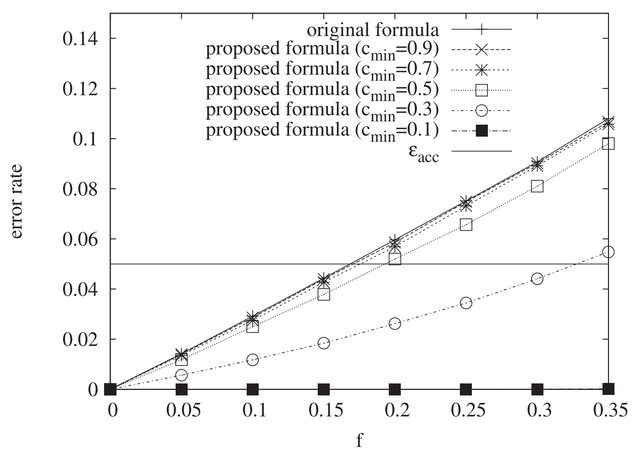


(a) $\epsilon_{acc} = 0.05$

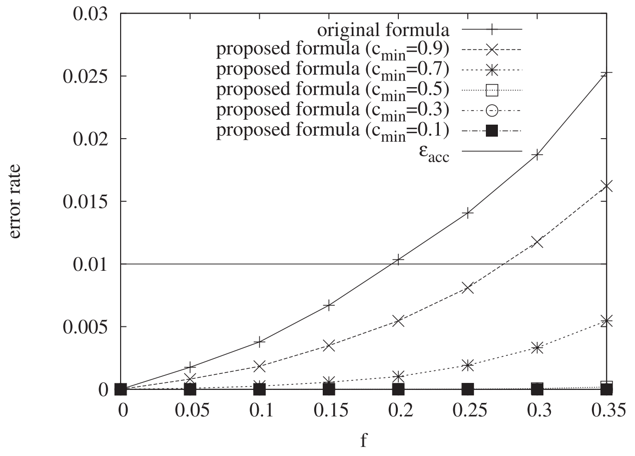


(b) $\epsilon_{acc} = 0.01$

Fig. 3 Error-rate ϵ vs. sabotage rate s for $f = 0.35$ and $c = 0.1$ (round-robin).

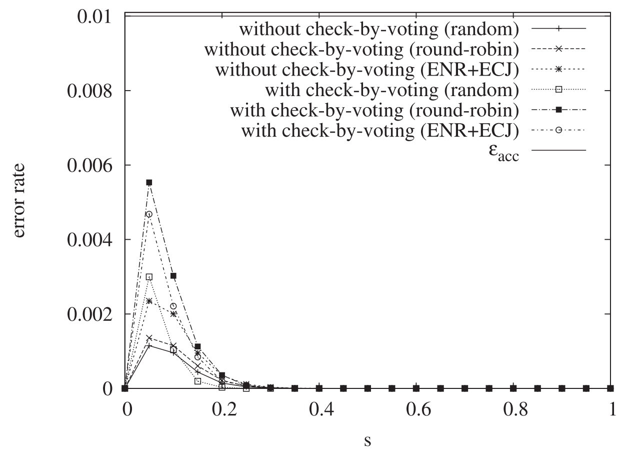


(a) ϵ_{acc} = 0.05

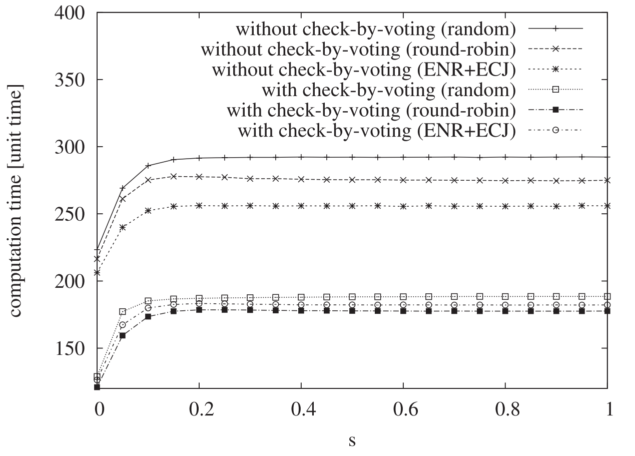


(b) ϵ_{acc} = 0.01

Fig. 4 Error-rate ϵ vs. faulty fraction sf for $s = 0.5$ and $c = 0.1$ (round-robin).



(a) Error-rate ϵ



(b) Computation time T

Fig. 5 Computation time T and error-rate ϵ vs. sabotage rate s for $\epsilon_{acc} = 0.01$, $f = 0.35$ and $c = c_{min} = 0.8$.

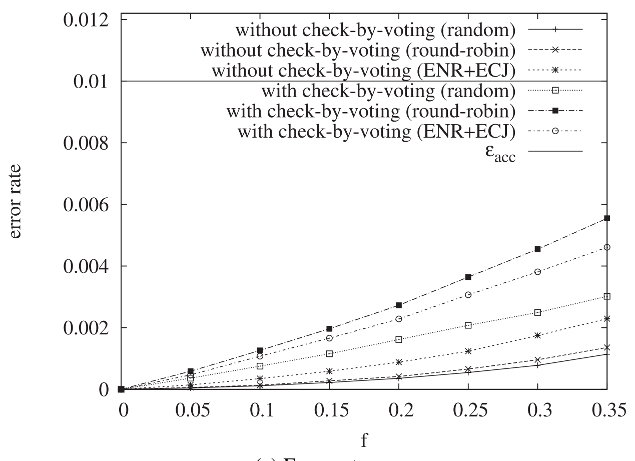
the credibility to workers based on c_{min} as shown in Eq. (10), under the assumption that the unknown parameter c satisfies $c_{min} \leq c$. As shown in this figure, the error rate tends to increase as c_{min} increases. Thus, when the lower bound of c can not be estimated, it seems that c_{min} should be set to smaller value to guarantee $\epsilon \leq \epsilon_{acc}$ for any situations.

Figure 4 shows error rate ϵ as a function of f. This figure also shows that the error rates with the original formula may exceed ϵ_{acc}. Since error rate is proportional to the number of saboteurs $f \times W$ and f is unknown to the master, the condition $\epsilon \leq \epsilon_{acc}$ should be satisfied for any f. Here, the error rate of VCs with the proposed formula at $c_{min} = 0.1$ satisfies $\epsilon \leq \epsilon_{acc}$ for any f, even if $f = f_{max}$. From Figs. 3 and 4, it is shown that the credibility-based voting with the proposed formula can guarantee $\epsilon \leq \epsilon_{acc}$ for any s and f as long as the condition $c_{min} \leq c$ is satisfied.

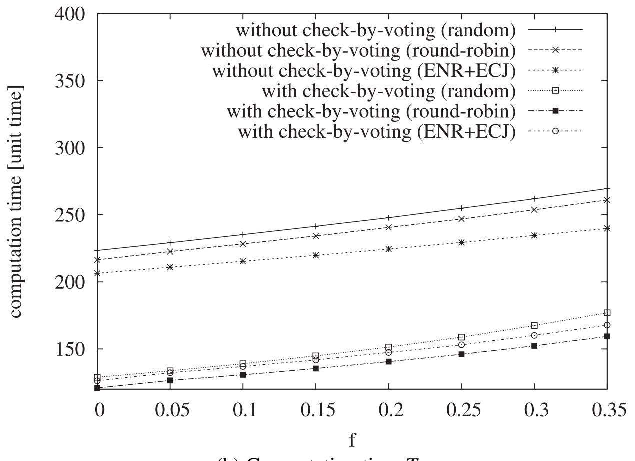
4.2.2 Improvement with Check-by-Voting

Figure 5 shows computation time T and error rate ϵ as a function of s in cases with check-by-voting and without check-by-voting. Figure 5 (a) shows that the error rates of all methods are less than ϵ_{acc} for any s. This means, even if check-by-voting is used, credibility-based voting with the proposed formula guarantees the reliability condition $\epsilon \leq \epsilon_{acc}$. Figure 5 (b) shows that the computation times of VCs with check-by-voting are decreased compared to those without check-by-voting. This is true because workers can gain higher credibility by check-by-voting in addition to the spot-checking with rate q. For example, when $s = 1$, the minimal computation time of VCs without check-by-voting is 255 (ENR+ECJ), while that with check-by-voting is 180 (round-robin). Note that the $ENR + ECJ$ method is the fastest scheduling method in credibility-based voting without check-by-voting. This result indicates that using check-by-voting is more effective to improve the computation time than devising a new job scheduling method without check-by-voting.

Figure 6 shows error rate ϵ and computation time T as a function of f in cases with check-by-voting and with-

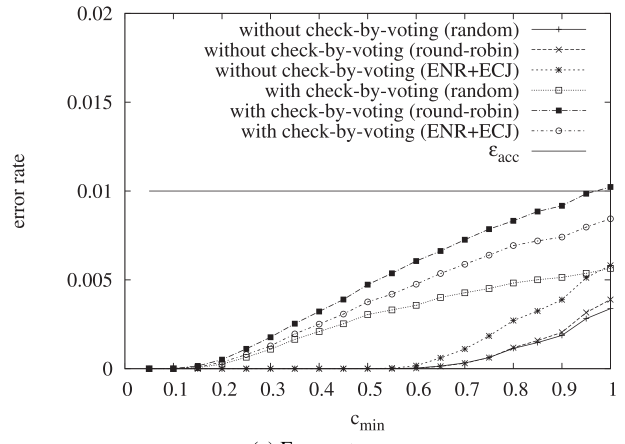


(a) Error-rate ϵ

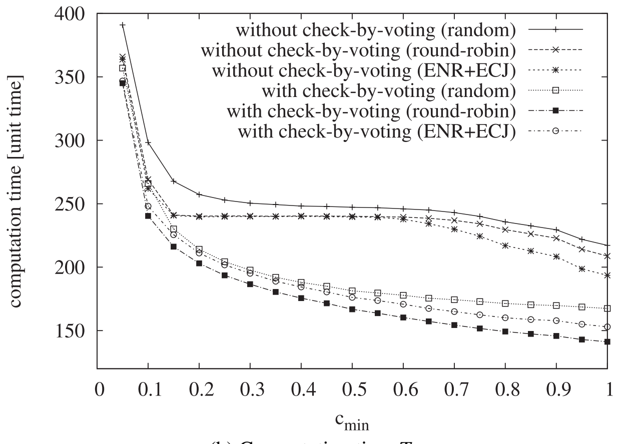


(b) Computation time T

Fig. 6 Computation time T and error-rate ϵ vs. faulty fraction f for $\epsilon_{acc} = 0.01$, $s = 0.05$ and $c = c_{min} = 0.8$.



(a) Error-rate ϵ



(b) Computation time T

Fig. 7 Computation time T and error-rate ϵ vs. c_{min} for $\epsilon_{acc} = 0.01$, $f = 0.35$, $s = 0.05$ and $c = 0.1$.

out check-by-voting. As f increases, the error rate of each method increases because the number of erroneous results returned by saboteurs increases proportionally to f. However, as shown in Fig. 6 (a), the error rates of all methods are less than ϵ_{acc} as long as f is less than $f_{max} = 0.35$. Figure 6 (b) shows that the computation times of VCs with check-by-voting are decreased compared to those without check-by-voting even if $f = 0$, that is, there are no saboteurs. From Figs. 5 and 6, it is shown that check-by-voting can decrease the computation time, while guaranteeing the condition $\epsilon \leq \epsilon_{acc}$ for any s and f.

Figure 7 shows error rate ϵ and computation time T as a function of c_{min} in cases with check-by-voting and without check-by-voting. This figure also shows that the condition $\epsilon \leq \epsilon_{acc}$ is satisfied for all methods, as long as the condition $c_{min} \leq c = 0.1$ is satisfied. Only when the condition $c_{min} \leq c$ is not satisfied, error rate can exceed ϵ_{acc} (e.g. $c_{min} = 1$).

As shown in Figs. 3 and 7 (a), the error rate tends to increase as c_{min} increases. This is true because the credibility of a worker given by a master is proportional to c_{min} as shown in Eq. (10). When the master gives a saboteur larger credibility, it increases the chance of accepting erroneous results produced by the saboteur.

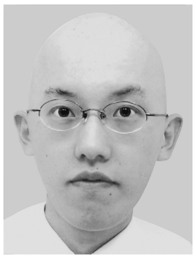
Figure 7 (b) shows that, as c_{min} increases, the computation time decreases. This is true because larger credibility (given by larger c_{min}) reduces the mean number of results to finish a job, resulting in smaller computation time to finish all jobs. This result indicates that, by setting c_{min} larger as much as possible (ideally $c_{min} = c$), we can minimize the computation time while guaranteeing the condition $\epsilon \leq \epsilon_{acc}$. However, estimating the lower bound of c is a difficult problem because the actual value of c differs between saboteurs and depends on several factors such as the frequency and the number of spot-checking. Thus, we simply conclude in this paper that; by setting c_{min} to very small value to satisfy $c_{min} \leq c$, we can guarantee the condition $\epsilon \leq \epsilon_{acc}$ even if c is unknown. The optimization of the parameter c_{min} is required for the computation time minimization, and will be discussed in other papers.

5. Conclusion

In this paper, we generalize spot-checking by introducing probability c, i.e. the accuracy of spot-checking. This generalization allows to use imperfect checking for spot-checking-based mechanisms such as credibility-based voting. First, we develop a generalized formula of the credibility considering the probability c to guarantee the computational correctness even if spotter jobs may be distinguished. The simulation results show that credibility-based voting with the proposed formula always hold the required reliability condition $\epsilon \leq \epsilon_{acc}$. Next, we propose check-by-voting, which utilize the result of each voting as an imperfect checking to improve the performance of credibility-based voting. The simulation results show that check-by-voting method can improve the performance of VC systems, while guaranteeing the condition $\epsilon \leq \epsilon_{acc}$ for any parameters s and f.

References

[1] D. Kondo, F. Araujo, P. Malecot, P. Domingues, L.M. Silva, G. Fedak, and F. Cappello, "Characterizing error rates in internet desktop grids," 13th European Conference on Parallel and Distributed Computing, pp.361–371, 2007.
[2] P. Domingues, B. Sousa, and L.M. Silva, "Sabotage-tolerance and trust management in desktop grid computing," Future Gener. Comput. Syst., vol.23, no.7, pp.904–912, 2007.
[3] Y.A. Zuev, "On the estimation of efficiency of voting procedures," Theory of Probability and Its Applications, vol.42, no.1, pp.71–81, 1998.
[4] BOINC http://boinc.berkeley.edu/, Jan. 2010.
[5] S. Zhao, V. Lo, and C.G. Dickey, "Result verification and trust-based scheduling in peer-to-peer grids," 5th IEEE International Conference on Peer-to-Peer Computing, pp.31–38, 2005.
[6] L.F.G. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems," Future Gener. Comput. Syst., vol.18, no.4, pp.561–572, 2002.
[7] K. Watanabe, M. Fukushi, and S. Horiguchi, "Optimal spot-checking for computation time minimization in volunteer computing," J. Grid Computing, vol.7, no.4, pp.575–600, 2009.
[8] K. Watanabe, M. Fukushi, and S. Horiguchi, "Expected-credibility-based job scheduling for reliable volunteer computing," IEICE Trans. Inf. & Syst., vol.E93-D, no.2, pp.306–314, Feb. 2010.
[9] L.F.G. Sarmenta, "Volunteer computing," Ph.D. thesis, Massachusetts Institute of Technology, 2001.



Kan Watanabe received the B.E. degree in information engineering, and the M.S. degree in information sciences from Tohoku University, Japan, in 2006 and 2008, respectively. He is currently working toward the Ph.D. degree. His research interest includes parallel and distributed computing systems.

Masaru Fukushi received the M.S. degree from Hirosaki University in 1997 and a Ph.D. in Information Science from the School of Information Science at JAIST (Japan Advanced Institute of Science and Technology) in 2002. He is currently a research associate in the Graduate School of Information Sciences, Tohoku University. His research interests are dependable multi-processor systems, reconfigurable systems and parallel image processing.