Journal of Information Processing Vol. 19 39–51 (Feb. 2011)

Regular Paper

Adaptive Group-Based Job Scheduling for High Performance and Reliable Volunteer Computing

Kan Watanabe,[†1] Masaru Fukushi[†1] and Michitaka Kameyama[†1]

This paper presents an adaptive group-based job scheduling method for credibility-based sabotage-tolerance techniques in volunteer computing (VC) systems. Credibility-based technique is a promising approach for reliable VC systems since it guarantees computational correctness mathematically based on the credibility of participants. Check-by-voting reduces the cost of checking credibility in credibility-based technique. However, in some applications where the deadline of the computation is relatively short, current job scheduling methods do not work well for check-by-voting and significantly degrade performance. To improve the performance of VCs, the proposed job scheduling method adaptively groups participants based on the expected-credibility to take into account the participants under job execution. Simulation of VCs shows that the proposed method always outperforms current job scheduling methods regardless of the values of unknown parameters such as population and behavior of saboteurs.

1. Introduction

Volunteer computing (VC) is a type of Internet-based parallel computing paradigm, which allows any participant on the Internet to contribute their idle computing resources towards solving large parallel problems. The most popular example of VC is SETI@home [1),2)], which is currently employing hundreds of thousands of volunteer participants to search massive amounts of radio telescope data for signs of extra-terrestrial intelligence. 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 environments with a very low cost.

While there has been rapidly growing interest in VC, VC systems still have a mandatory issue for reliable computing [3),4)]. Different from the grid computing system which shares managed reliable computers within or among organizations, computing resources of a VC system are owned and managed by volunteer participants. Those resources may behave erratically due to hardware/software failures or virus infection, or may behave maliciously to falsify the computation, each of which results in sabotage to the computation. It is reported in Ref. 3) that a large fraction of participants in real VC (about 35%) actually returned at least one incorrect result.

Against those sabotaging, some sabotage-tolerance methods are proposed and used for reliable VCs. Simple voting is widely used in BOINC [5),6)], the most popular VC middleware, to determine the final result through voting. Credibility-based voting [7)] is an advanced voting method which guarantees the computational correctness mathematically. Check-by-voting [8)] is an efficient checking technique to detect sabotaging and to check credibility without wasting computing resources (e.g., CPU cycles of volunteer participants). Combined with the above two methods, credibility-based voting with check-by-voting is known to be a promising approach for high-performance and reliable VC in that the computational correctness is guaranteed with little redundant computation.

However, the basic job scheduling methods for credibility-based voting used in Refs. 7), 8) do not work well for check-by-voting. Especially, for VCs with severe deadlines, those methods significantly degrade the performance of VC systems due to the presence of "half-finished jobs". The number of half-finished jobs has a significant impact on performance because they do not increase the throughput of VC systems, while wasting the computing resources. Moreover, such half-finished jobs prevent check-by-voting from checking the credibility of participants sufficiently. Since the deadline is given based on numerous factors such as the availability of VC systems and the demand of computation projects, a novel job scheduling method is necessary for credibility-based voting with check-by-voting to support various VCs, especially those with severe deadlines.

The key idea of improving performance is reducing the number of such half-finished jobs by using a grouping technique. The grouping technique decreases the number of half-finished jobs by allocating a job to multiple workers in a group at the same time. Although some adaptive grouping methods [9)] have been proposed for adaptive grouping, these methods are not applicable for credibility-

†1 Graduate School of Information Sciences, Tohoku University

based voting because they focus only on simple voting which can not change the necessary number of results adaptively.

In this paper, we propose an adaptive group-based job scheduling method for credibility-based voting with check-by-voting to improve the throughput of VC systems. The main contributions of our work are the following. (1) We develop a dynamic grouping method, which predicts the optimal group size based on the expected-credibility and reduces the number of half-finished jobs. (2) We reveal the performance of VC systems with several job scheduling methods including the proposed method and m-first voting used in real VCs.

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 an expected-credibility-based grouping method. Section 4 compares the error rates and throughputs of the existing and the proposed job scheduling methods. Finally, Section 5 concludes this paper.

2. Volunteer Computing Systems

2.1 Computational Model

A well known work-pool-based master-worker model [6),7)] is assumed as the computation model of VC systems. This model is used in almost all practical VC systems. 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 computation proceeds in a series of time steps until either all N jobs are finished or the time steps reach a predefined deadline TD.
- Before the computation (at time step $t = 0$), all jobs are placed in a work pool of the master.
- The master gives a job to each idle worker in each time step.
- Each worker executes the allocated job and returns the result to the master. During 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 might return incorrect results. Each saboteur attempts to return an incorrect result with a constant probability s, which is known as the sabotage rate [7)]. The values of f and s are unknown to the master.

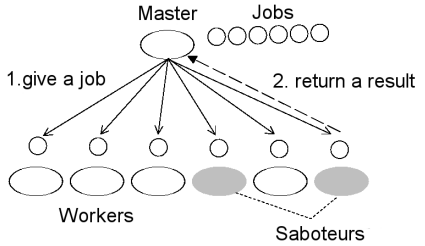


Fig. 1 Computation model of VC systems.

The master manages the execution of a computation and allocates unfinished jobs to idle workers. A computation finishes when the time step t reaches the deadline TD or all N jobs are finished. Jobs that finish with incorrect results are called incorrect jobs. At the end of the computation, an error rate ε can be calculated as the ratio of incorrect jobs to all finished jobs.

Using no sabotage-tolerance mechanisms, the error rate increases in proportion to the number of saboteurs. Assume that n jobs are finished at the end of the computation and all workers function at the same speed. Then, error rate ε is given by $n \times f \times s/n = f \times s$. It is clear that ε is proportional to the number of saboteurs and the sabotage rate s. Therefore, to reduce the error rate, some sabotage-tolerance mechanism must be used.

2.2 Sabotage-tolerance Mechanisms

2.2.1 Basic Mechanisms

Two basic sabotage-tolerance mechanisms are simple voting and spot-checking.

2.2.1.1 Simple 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 common 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 (the same value) results first is accepted as the final result.

Because of their simplicity, the simple voting methods are widely used in real VC systems such as BOINC [5),6)].

2.2.1.2 Spot-checking

To check whether a worker is a saboteur or not, a master sometimes assigns 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 incorrect result for the spotter job.

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

- Backtracking: The master backtracks (invalidates) all results returned by the saboteur because each might be an incorrect one.
- Blacklisting: The master puts the 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. Note that if a saboteur distinguishes a spotter job from normal ones, the saboteur may temporarily stop sabotaging and return the correct result to avoid being caught by spot-checking. The accuracy of spot-checking (denoted by c) represents the probability that a saboteur does not distinguish a spotter job [8)]. Each saboteur is caught with probability $s \times c$ in each spot-checking.

2.2.2 Credibility-Based Mechanisms

The credibility-based mechanisms are proposed to overcome inefficient redundant computation of the simple voting methods and provides a mathematical guarantee for the upper bound of error rate.

2.2.2.1 Credibility-based Voting

Sarmenta [7)] presents a new voting method using spot-checking, referred to as "credibility-based voting" in this paper. In this method, each system element such as worker, result, result group, and job is assigned a credibility value that represents its correctness. The values of credibility are managed by the master and the most recent values are stored in the master's storage memory.

To check the credibility of workers, the master allocates a spotter job to a worker with probability q, which is known as the spot-check rate. Thus, every worker has a different credibility, which affects the credibility of the result and the result groups. A job is finished when the credibility of any result group reaches a threshold θ. Therefore, unlike the simple voting methods presented in the previous subsection, the necessary number of results to finish a job is not fixed and is generally smaller than that of simple voting.

Furthermore, this method can guarantee that the mathematical expectation of the error rate is below an arbitrary acceptable error rate $\varepsilon_{\mathrm{acc}}$ by setting three conditions [7),8)]: (1) threshold $\theta = 1 - \varepsilon_{\mathrm{acc}}$, (2) unknown parameter f satisfies $f \leq f_{\max}$, and (3) unknown parameter c satisfies $c \geq c_{\min}$. The values of $f_{\max}$ and $c_{\min}$ are known to the master.

2.2.2.2 Check-by-Voting

Although spot-checking is effective to detect sabotaging, spotter jobs are extra jobs for the computation itself. Such extra jobs waste the resources of VC systems (i.e., workers) and degrade performance. Check-by-voting enables the detection of sabotaging and checks the credibility of workers without allocating such extra jobs.

The idea of check-by-voting is to regard voting as spot-checking [8)]. When a job is finished through voting, workers who return the majority (the accepted) results are regarded as non-saboteurs who survive spot-checking, whereas workers who return the minority result are regarded as saboteurs. In check-by-voting, a result in the majority result group of the job is correct with a certain probability P. The actual value of the probability P is unknown: however, in the case where credibility-based voting is employed, the lower bound of P can be guaranteed ($\theta \leq P$).

In credibility-based voting with check-by-voting, the credibility of a worker is given by Eq. (1) as the lower bound of the probability that the worker returns a correct result [8)].

$$C_W(w) = \begin{cases} 1 - f_{\max} & \text{if } k = 0, \\ 1 - \dfrac{f_{\max}}{1 - f_{\max}} \times \max\left(\dfrac{1}{kex_{\min}}, (1 - x_{\min})^k\right) & \text{otherwise,} \end{cases} \quad (1)$$

where k represents the total number of checking (spot-checking and check-by-voting) the worker survives, e is Napier's constant and $x_{\min} = \min(c_{\min}, 1 - \varepsilon_{\text{acc}})$.

Using check-by-voting, the performance of credibility-based voting can be improved for two reasons. The first reason is that check-by-voting allows the master to check workers without allocating spotter jobs. Since spotter jobs are extra jobs, unallocation of spotter jobs leads to higher performance. The second reason is that check-by-voting increases the number of checking workers. Since the results returned from non-saboteurs tend to be the majority, those workers can gain higher credibility, resulting in a reduction of redundant job executions. Therefore, credibility-based voting with check-by-voting is a promising approach to high-performance and reliable VC systems.

3. Expected-Credibility-Based Grouping

3.1 Job Scheduling Problem

Generally, the number of jobs N is extremely-numerous since VC is used for massive scientific projects such as SETI@home [1),2)]. For example, N in a parameter optimization project may reach 2^x, i.e., all possible combinations of x parameters. Although VC system is high-performance, it cannot always compute all of such jobs. Thus, VC systems compute a part of N jobs during a given period, e.g., several months or years. Throughput is important and often used as the performance metric of VC systems [9)]. In the computation model described in Section 2, the throughput, denoted by N_s, is given as the number of finished jobs until the given deadline TD. The ideal value of N_s is $\max(TD \times W, N)$ when all workers execute jobs in each time step.

The throughput of VC systems is highly dependent on the order of execution of jobs. Especially, in credibility-based voting, the necessary number of results for each job depends on the credibility, which changes as the computation proceeds. Hence, the total number of produced results for the computation also depends on the order of execution of jobs. Thus, in credibility-based voting, job scheduling method has a considerable impact on the performance of VC systems.

In sabotage-tolerant VC systems using credibility-based voting, job scheduling problem is summarized as follows:

> *select a job that should be allocated to an idle worker prior to other jobs for increasing throughput N_s to the greatest extent possible, while guaranteeing the computational correctness as $\varepsilon \leq \varepsilon_{\text{acc}}$ for any given ε_{acc}.*

Basic job scheduling methods for credibility-based voting are random and round-robin methods [7),8),10)]. The random method selects a job at random and the round-robin method selects a job in a static order, e.g., the order of a job's ID assigned by the master. Although these job scheduling methods are simple, those methods cause significant performance degradation when the deadline TD is relatively small. The deadline TD is given based on numerous factors such as the availability of VC systems and the demand of computation projects. For an extreme example, the deadline may be within a day for simulation-based tomorrow's weather forecast. Therefore, for broad-ranging utilization of VC systems, a novel job scheduling method for smaller deadline is necessary.

3.2 Expected-Credibility-Based Grouping

The main factor of performance degradation is the presence of "half-finished job". The half-finished job is a job which has some results but are not finished by the end of the computation due to the insufficiency of the number of matching results or the credibility. The amount of half-finished jobs has a significant impact on the performance because such jobs do not increase N_s, while they waste the computing resources (workers' CPU cycles) of VC systems to produce their results. Also, such half-finished jobs affect the performance of credibility-based voting itself because the smaller number of finished jobs N_s leads the smaller number of check-by-voting, which leads the smaller credibility of workers. As the credibility of workers becomes smaller, each job requires much number of results to be finished, resulting in the performance degradation.

The key idea of improving the performance is reducing the number of such half-finished jobs and increasing N_s utilizing the computing resources would be wasted for the half-finished jobs. Based on this idea, we propose a scheduling method, referred as "expected-credibility-based grouping". The proposed job scheduling method is summarized as follows.

Firstly, we use the idea of well-known grouping technique of workers. In the

grouping technique, the master divides idle workers into some groups, each of which executes the same job. By allocating a job to multiple workers in a group at the same time, the master can decrease the number of half-finished jobs.

The simplest grouping method is "simple grouping", in which the master divides all idle workers into groups of the same size. In the simple grouping, the number of workers in a group (group size m_g) is the same among all groups and is a constant value. Since the optimal group size may change dynamically, some adaptive grouping methods [9] are proposed for dynamic grouping of workers. However, those grouping methods are available only for simple voting since they assume that the number of necessary results to finish a job must be fixed in advance. Thus, we propose an adaptive grouping method for credibility-based voting.

Secondly, we use the expected-credibility [10] for an adaptive grouping of workers. In credibility-based voting, a grouping method should consider the credibility in addition to the number of workers since the number of results to finish a job is not fixed and depends on the credibility. However, the credibility itself is insufficient for scheduling metrics since the workers who are currently executing jobs are not taken into consideration. Those workers will return their results, which affect the credibility. Thus, we use the expected-credibility which gives an expectation of the credibility considering the presence of those workers.

The expected-credibility of job j, denoted by $EC_J(j)$, is defined as follows [10]. Suppose that job j has several results which can be grouped into g groups $(G_1, \ldots, G_g)$. Let G_x be the group which has a maximum credibility in the g groups, where $1 \leq x \leq g$. There exist d workers $(w_1, \ldots, w_d)$ who are executing job j as shown in **Fig. 2**. Then, $EC_J(j)$ is given as follows.

$$EC_J(j) = \max_{1 \leq a \leq g} C'_G(G_a), \tag{2}$$

where

$$C'_G(G_a) = \frac{P'_T(G_a) \prod_{i \neq a} P'_F(G_i)}{\prod_{i=1}^{g} P'_F(G_i) + \sum_{n=1}^{g} P'_T(G_n) \prod_{i \neq n} P'_F(G_i)}, \tag{3}$$

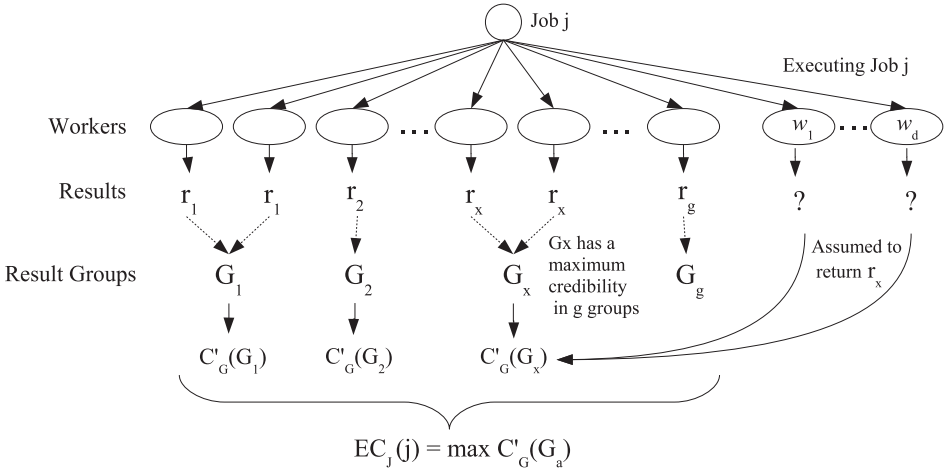


Fig. 2 Expected-credibility EC_J.

$$P'_T(G_a) = \begin{cases} \prod_{r_a \in G_a} C_W(W_{r_a}) & \text{if } a \neq x, \\ \prod_{r_a \in G_a} C_W(W_{r_a}) \times \prod_{i=1}^{d} C_W(w_i) & \text{if } a = x, \end{cases} \tag{4}$$

$$P'_F(G_a) = \begin{cases} \prod_{r_a \in G_a} (1 - C_W(W_{r_a})) & \text{if } a \neq x, \\ \prod_{r_a \in G_a} (1 - C_W(W_{r_a})) \times \prod_{i=1}^{d} (1 - C_W(w_i)) & \text{if } a = x. \end{cases} \tag{5}$$

In those equations, $P'_T(G_a)$ defined by Eq. (4) represents the correctness of result group G_a [10]. Since the correctness of each result r_a is $C_W(W_{r_a})$, $P'_T(G_a)$ is given as the product of $C_W(W_{r_a})$ for all results in G_a. In the definition of expected-credibility [10], all d workers $w_1, \ldots, w_d$ are assumed to return the same results which belong to group G_x. Hence, if $a = x$, credibility of d workers are also included in the product. This is the different point between the credibility and expected-credibility. Similarly, $P'_F(G_a)$ defined by Eq. (5) represents the incorrectness of G_a.

With the correctness and incorrectness of each result group, i.e., $P'_T(G_a)$ and

```
1  Push idle workers in Q;
2  //Q is a queue of workers who wait job allocation
3  Initialize S as the set of all unfinished jobs;
4  //S is a set of candidate jobs for allocation
5
6  while ( Q is not empty ) do
7    Pop worker w from Q;
8    Allocate a spotter job to w with rate q;
9    while ( No job is allocated to w ) then
10     Select job j which has a maximum EC_J in S;
11     if(EC_J(j) ≥ θ) then
12       S = S − {j};
13       //eliminate j from candidate set S
14     else
15       Allocate job j to w;
16       //add w to the group of workers who are executing j
17       Update EC_J(j);
18     end if
19   end while
20 end while
```

Fig. 3 Expected-credibility-based grouping.

$P_F'(G_a)$, $C_G'(G_a)$ defined by Eq. (3) represents the relative correctness of a result group; that is, the correctness of G_a under the all possible cases. For g result groups, there are $g+1$ possible cases; either all g groups are incorrect or one of g result groups, i.e., G_n, is correct and all others are incorrect for $n = 1, \ldots, g$ (g cases). In Eq. (3), first term of the denominator represents the former and the second term represents the latter. Similarly, the numerator represents the case where G_a is correct and all others are incorrect.

$EC_J(j)$ predicts the credibility of job j supposing the best-case where all d workers return their results for a result group currently having a maximum credibility.

3.3 Job Scheduling Algorithm

Using the grouping technique and the metric EC_J, we propose expected-credibility-based grouping method for credibility-based voting with check-by-voting. **Figure 3** shows the algorithm of the proposed method.

First, all idle workers are stored in a worker queue Q (lines 1–2 in Fig. 3). Let S be the set of candidate jobs waiting for allocation. At the start of job scheduling, the master initializes S as the set of jobs which are unfinished at that time (lines 3–4). As long as the worker queue is not empty, the master extracts a worker from the queue and allocates a job to the worker (lines 6–20). When a worker w is extracted from the queue (line 7), by spot-checking, a spotter job is allocated to w with spot-check rate q (line 8). If spotter job is not allocated, one unfinished job is allocated to w (lines 9–19). The master selects job j, which has a maximum EC_J in the set of candidate jobs S (line 10). Note that if $EC_J(j)$ exceeds θ, j will be finished with results from d workers who are currently executing j. Then, the master eliminates j from S to avoid excess executions of j (lines 11–13). On the other hand, if $EC_J(j)$ does not exceed θ, j requires more results and workers in addition to current group of d workers. Then, the master adds w to the group by allocating j to w (lines 15–16). Once j is allocated to w, then $EC_J(j)$ is updated immediately to reflect the allocation of j in the subsequent scheduling (line 17).

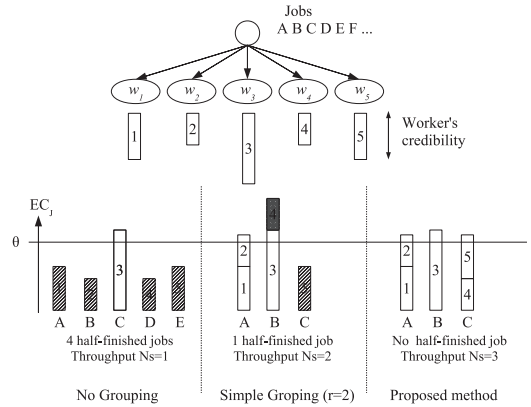


Fig. 4 An example of job scheduling.

Using this scheduling method, a job is allocated to multiple workers in a group and executed simultaneously. Compared to the job scheduling method with no grouping such as the round-robin method [7], this scheduling policy enables to decrease the number of half-finished jobs by the grouping. Also, compared to the simple grouping method, this scheduling policy enables to increase N_s because the group size is adaptively decided based on EC_J.

Figure 4 shows the differences of job scheduling and the throughput N_s among

several job scheduling methods. Suppose that there are several unfinished jobs $A, B, \ldots$ and 5 idle workers $(w_1, \ldots, w_5)$. Each of the workers has different credibility depending on the number of spot-checking and check-by-voting the worker survives. In this case, w_3 has the largest credibility which is large enough to finish a job, that is, $C_W(w_3) \geq \theta$. In no grouping method (the left case in Fig. 4), the master allocates each job to workers one-by-one; then, 5 jobs, A, B, C, D and E are allocated to each worker. After the results of those jobs are returned, job C is finished because the credibility of C reaches θ, while other 4 jobs are not finished at this time. If the deadline is too short to reallocate these unfinished jobs to workers, those jobs becomes half-finished ones at the end of the computation and throughput $N_s = 1$ for job C. This is why the job scheduling methods without groping cause significant performance degradation when the deadline TD is relatively small.

The grouping method such as simple grouping (the middle case in Fig. 4) can decrease the number of half-finished jobs by allocating the same job to multiple workers. In this case, workers w_1 and w_2 are grouped to execute job A; then job A will be finished when those workers return their results and the credibility of job A reaches θ. However, this method may cause performance degradation due to excess job allocations. This example shows that the result from w_4 is unnecessary to finish job B, since the credibility of job B can reach θ without the result. The simple grouping method and other existing grouping methods [9] can not avoid such job allocations since they assume that the number of necessary results to finish a job must be fixed, while it changes depends on the credibility in credibility-based voting.

The proposed method works well to avoid such excess job allocations based on EC_J. In this case (the right case in Fig. 4), instead of job B which will be excess, another job C is allocated to w_4. The master groups w_1 and w_2 for job A as the simple grouping does since the master can make a decision based on EC_J that job A will not be finished with the result from w_1. Thus, compared to no grouping and simple grouping methods, the proposed method can improve the number of jobs finished during the same period. Also, the proposed method can improve the efficiency of credibility-based voting by the effect of increased number of check-by-voting. In this example, jobs A, B and C are finished after the workers return their results. Then, at this time, the master utilizes these results as 3 check-by-votings and increases the credibility of workers. The larger credibility of workers enables the smaller number of results to finish jobs in credibility-based voting. Therefore, for all jobs executed in subsequent time periods, the number of necessary results becomes smaller, resulting in larger N_s.

In addition to the advantage in performance discussed above, the proposed method is also capable of tolerating saboteurs' malicious behavior. In VCs where saboteurs intentionally behave, some saboteurs may return incorrect results without executing jobs. Such saboteur can produce many incorrect results by getting more jobs than non-saboteurs who execute jobs for correct results. This type of sabotaging strongly influences the error rate of voting method. Against such malicious behavior, spot-checking is known to work well. In spot-checking, a saboteur who gets more jobs tends to be caught more easily because the saboteur gets more spotter jobs. The proposed method contains the mechanism of spot-checking (line 8 in Fig. 3) and thus it can also work well for such malicious behavior. Even though saboteurs return many incorrect results without execution, the proposed method can catch such saboteurs with the mechanism of spot-checking and eliminate the returned incorrect results by backtracking. Once a saboteur is caught, blacklisting can prevent the saboteur from joining the system again. Therefore, saboteurs cannot continue such malicious behavior in the proposed method.

3.4 Scheduling Cost

In the proposed method, the master must manage and update values of EC_J in addition to the values of credibility in credibility-based voting. Thus, the proposed method requires the master to cost some additional space to store the values of EC_J and time to calculate EC_J. The space complexity of the proposed method is $O(N)$, where N is the number of jobs, because the master keeps only the latest value of EC_J for each job. The time complexity of the proposed method is complicated to derive analytically because the definition of EC_J includes many unknown parameters such as the number of results in each result group. However, the actual time to calculate EC_J for a job is not so large (e.g., magnitude of microseconds) since it consists of simple arithmetic operations shown in Eqs. (2)–(5). Therefore, the time to update EC_J is sufficiently-small

and negligible compared to the time to execute one job (e.g., several hours).

4. Performance Evaluation

4.1 Simulation Conditions

In this section, we evaluate the effectiveness of the proposed job scheduling method through the simulation of VCs. The throughput N_s and error rate ε are evaluated as the average of 1,000 simulation results for five different methods: the proposed method, no grouping, simple grouping, m-first voting and spot-checking. The relation among those methods are summarized as follows.

- Credibility-based voting with check-by-voting
  - No grouping:
    The master uses the round-robin method [7] and allocates each job to one worker.
  - Simple grouping:
    The master uses the simple grouping method and allocates a job to m_g workers at the same time.
- Traditional sabotage-tolerance methods
  - 2-first voting:
    The master allocates a job to m_g workers at the same time. Since the simple voting methods require at least 2 results for a job, 2-first voting is the fastest method.
  - Spot-checking:
    Spotter jobs are allocated with rate q and a job is finished with one result (no redundant computation).

The parameters used in our simulation are shown in **Table 1**. Because some parameters are unknown to the master and are uncontrollable, such as s and f, 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 [3]. To compare the best case of spot-checking method and the proposed method, we assume that spotter jobs are never distinguished ($c = 1$) and $q = 0$ for the proposed method.

Also, we make existing assumptions [7),8),10),11)] for fair evaluations between different methods. First, all workers have the same processing speed. Therefore, jobs are distributed equally among the workers and a job is executed in one time step. Second, we assume "random attack with probability s" as sabotaging of saboteurs. In this attack model, the value of an incorrect result is set as a random value and never matches each other. Third, we assume workers' defection to model real workers' unexpected behavior; that is, workers join and leave the system freely in real VCs. We assume that a worker leaves the system with probability p_{down}, which is called the defection rate, and that a worker rejoins the system with probability p_{up} in every time step. The value of p_{up} is set corresponding to p_{down} so that 80% of workers are participating in the system, on average. For instance, when $p_{down} = 0.1$, p_{up} is set to 0.4. A job allocated to a worker is discarded when the worker leaves the system.

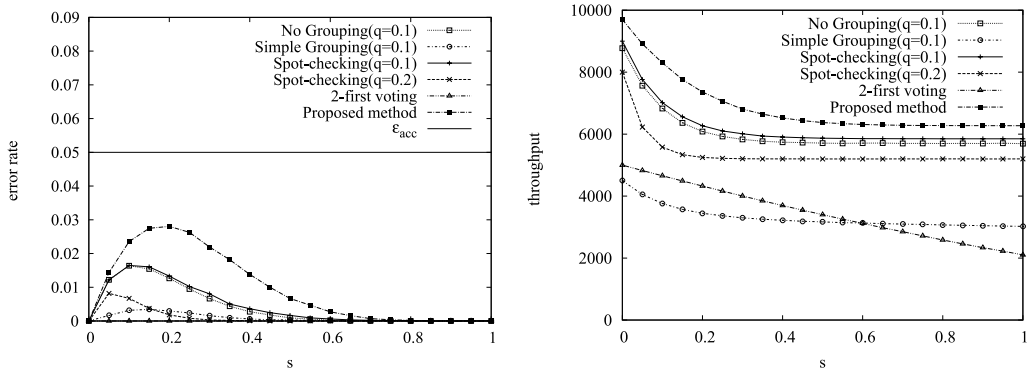
Table 1 Simulation parameters.

the number of jobs (N)	10,000
the number of workers (W)	100
faulty fraction (f)	$0 \sim f_{\max}$
upper limit of f ($f_{\max}$)	0.35
checking accuracy (c)	1
lower limit of c ($c_{\min}$)	1
acceptable error rate (ε_{acc})	0.01, 0.05
sabotage rate (s)	$0 \sim 1$
defection rate (p_d)	$0 \sim 0.8$
deadline (TD)	$100 \sim 200$
group size (m_g)	2

4.2 Simulation Results

4.2.1 Performance for Sabotage Rate s When $\varepsilon_{\text{acc}} = 0.05$

Figure 5 (a) shows error rate ε as a function of sabotage rate s for $\varepsilon_{\text{acc}} = 0.05$. This figure clearly shows that error rate of each method is less than the required value $\varepsilon_{acc} = 0.05$ for any s; that is, the reliability condition $\varepsilon \leq \varepsilon_{\text{acc}}$ is guaranteed. Error rate of 2-first voting is zero since incorrect results never matches each other and only correct results will be accepted through voting. Error rates of other methods increase at first at certain values of s (around $0.05 \sim 0.2$) since saboteurs with larger s increase the number of incorrect results. However, when s is larger than such values, error rate decreases with s. This is true because saboteurs with larger s will be caught with larger probability by checking techniques such as spot-checking and check-by-voting. Saboteurs with too large s will be caught



(a) Error-rate ε (b) Throughput N_s

Fig. 5 Error-rate ε and throughput N_s vs. sabotage rate s for $\varepsilon_{\rm acc} = 0.05$, $f = 0.35$, $p_d = 0$ and $TD = 100$.

until the end of the computation and all results produced by the saboteurs are invalidated by backtracking. Thus, there is a maximum point of error rate for s when checking techniques are used.

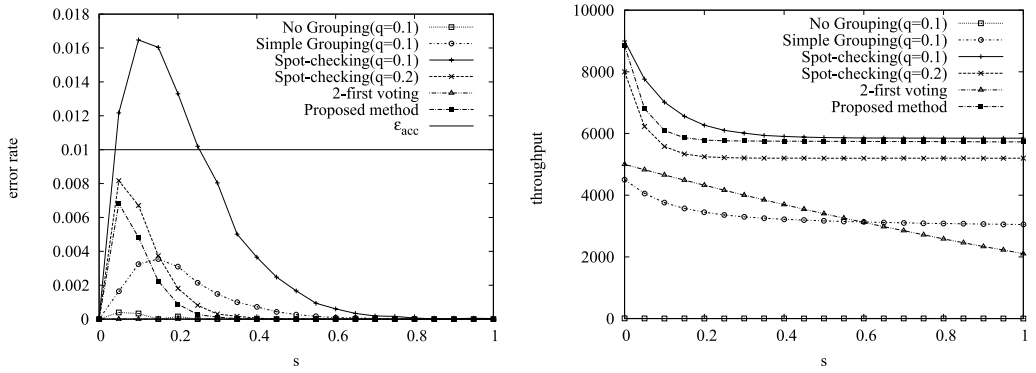
Figure 5 (b) shows the throughput N_s as a function of s. As s becomes larger, the throughput of 2-first voting decreases linearly, in proportion to the number of correct results which should be accepted. Note that, even if $s = 0$, the throughput of 2-first voting is at most 5,000 (i.e., $\lfloor TD \times W/m \rfloor$) since each job requires at least $m = 2$ results to be finished. The throughput of the simple grouping is also at most 5,000 since it allocates each of jobs to $r = 2$ workers at the same time. This result indicates that the throughputs of m-first voting and simple grouping with any m_g are less than the half of the ideal value.

On the other hand, throughputs of other methods including the proposed method exceed 5,000 as shown in Fig. 5 (b). This is true because some jobs are finished with only one result in those methods. Especially, each of jobs in spot-checking method finishes with only one result; hence, spot-checking methods show better performance than 2-first voting. No grouping method also shows better performance since it utilizes the credibility-based voting which enables to finish some jobs with only one result based on the credibility. Since the credibility depends on the number of checking (spot-checking and check-by-voting), executing many check-by-voting in the early stages of the computation leads much better performance, as does in the proposed method. Thus, as shown in this figure, the proposed method shows the best performance for any s.

4.2.2 Performance for Sabotage Rate s When $\varepsilon_{\rm acc} = 0.01$

Figure 6 (a) shows the error rate ε as a function of sabotage rate s for $\varepsilon_{\rm acc} = 0.01$. This figure clearly shows that the error rate of the credibility-based method never exceeds $\varepsilon_{\rm acc}$, while that of spot-checking method can exceed $\varepsilon_{\rm acc}$. This is true because some of saboteurs with smaller s survive spot-checking until the end of the computation. In spot-checking method, all incorrect results produced by such saboteurs are accepted without redundant computation, resulting in large error rate. Using larger q decreases the number of such saboteurs and the error rate. For example, at $s = 0.05$, the error rate of spot-checking method with $q = 0.2$ is 0.008 ($\leq \varepsilon_{\rm acc}$), while that with $q = 0.1$ is 0.012.

However, such larger q degrades the performance as shown in Fig. 6 (b). The throughput of the spot-checking method with $q = 0.2$ is less than that with $q = 0.1$. This is true because larger q decreases the number of normal jobs executed in the computation. In each time step, $W \times q$ workers get spotter jobs and $W \times (1-q)$ workers get normal jobs on average; then the number of executed jobs in spot-checking method is $TD \times W(1-q)$. The throughput is proportional



(a) Error-rate ε (b) Throughput N_s

Fig. 6 Error-rate ε and throughput N_s vs. sabotage rate s for $\varepsilon_{\rm acc} = 0.01$, $f = 0.35$, $p_d = 0$ and $TD = 100$.

to the number of executed normal jobs (they are the same at $s = 0$). Thus, it is clear that larger q decreases the throughput.

Figure 6 (b) also shows that the proposed method outperforms other methods, except for the spot-checking method with $q = 0.1$. Note that, as mentioned above, the spot-checking method with $q = 0.1$ does not guarantee the reliability condition. Although larger q (e.g., $q = 0.2$) decreases the error rate, the throughput of the spot-checking method with such large q is less than that of the proposed method. This result indicates that the proposed method shows the best performance among the methods which guarantee the reliability condition.

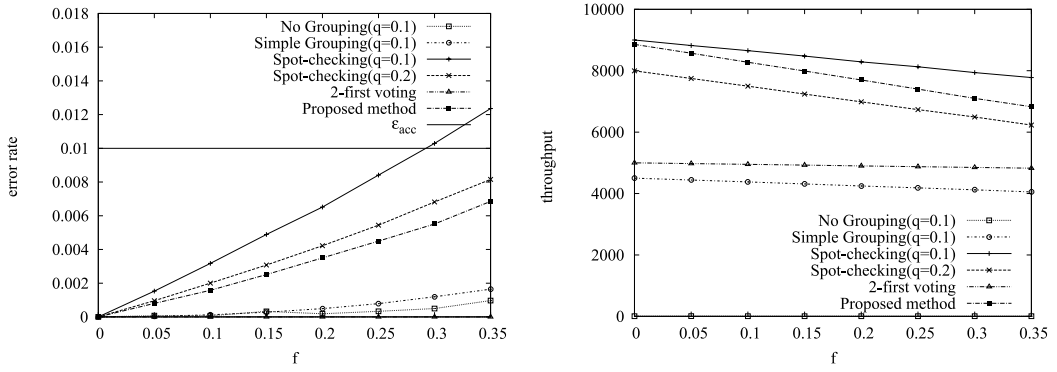
Compared to the case of $\varepsilon_{\rm acc} = 0.05$, the throughput of no grouping method is quit small (almost 0) when $\varepsilon_{\rm acc} = 0.01$. This is true because smaller $\varepsilon_{\rm acc}$ increases the required credibility of a worker (i.e., $\theta = 1 - \varepsilon_{\rm acc}$) to finish a job with only one result. Hence, no grouping method requires enough large rate q to gain such large credibility, while too large q degrades the throughput like in cases of spot-checking methods. On the other hand, the proposed method works well even if $q = 0$ (no spot-checking) since it can gain the credibility of workers enough by causing check-by-voting in a proactive manner.

4.2.3 Performance for Faulty Fraction f

Figure 7 shows error rate ε and throughput N_s as functions of faulty fraction f. As f increases, error rates of all methods tends to be large since the numbers of both saboteurs and incorrect results increase. As shown in Fig. 6 (a), the error rate of spot-checking method with $q = 0.1$ can exceed $\varepsilon_{\rm acc}$. Although larger q decreases error rate, it also decreases the throughput as shown in Fig. 7 (b). These figures show that, for any f, the proposed method shows the best performance among the methods which guarantee the reliability condition. Although the actual values of s and f are unknown to the master, the proposed method can improve the throughput irrespective of the values of s and f, i.e., the behavior and the number of saboteurs. This very important feature indicates that the proposed method is applicable to VC systems of various environments.

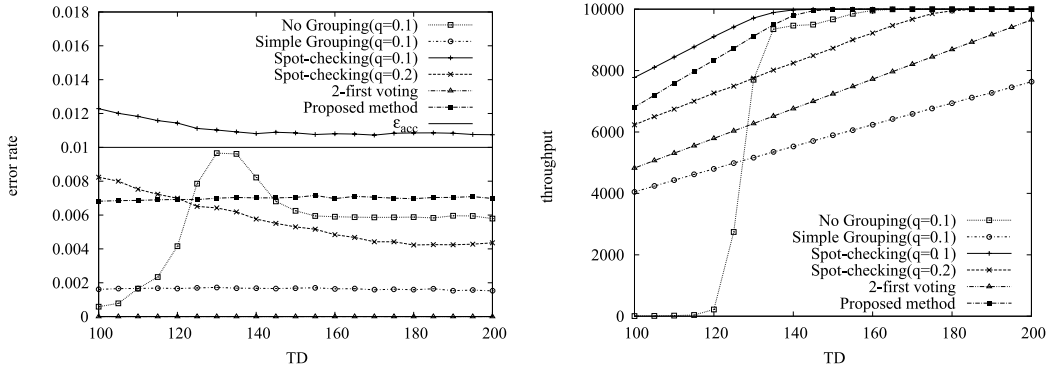
4.2.4 Performance for Deadline TD

Figure 8 (a) shows error rate as a function of deadline TD. This figure shows that error rates of spot-checking methods tend to be small as TD becomes larger. This is true because the number of spot-checking is proportional to TD. As TD increases, the master allocates more spotter jobs to saboteurs; then saboteurs are be caught more easily. The error rates of the proposed method and simple grouping are almost constant for TD since the number of spot-checking (e.g., less than 20) is negligibly small compared to the number of check-by-voting (e.g., more than 4,000).



(a) Error-rate ε (b) Throughput N_s

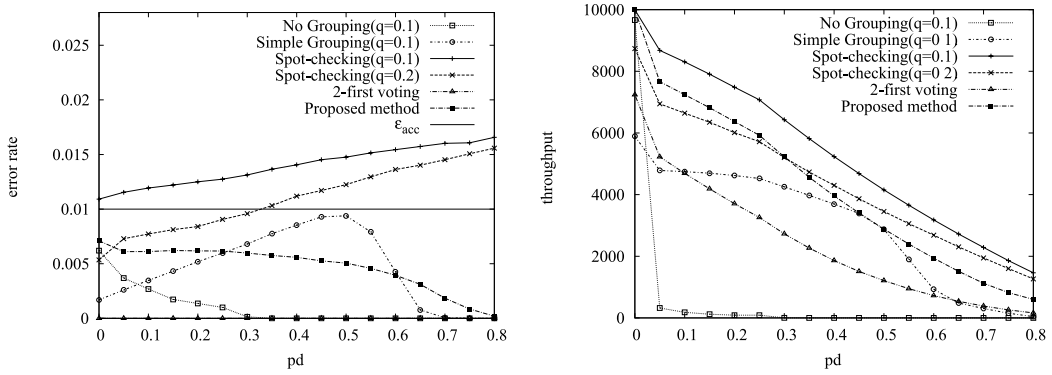
Fig. 7 Error-rate ε and throughput N_s vs. faulty fraction f for $\varepsilon_{\rm acc} = 0.01$, $s = 0.05$, $p_d = 0$ and $TD = 100$.



(a) Error-rate ε (b) Throughput N_s

Fig. 8 Error-rate ε and throughput N_s vs. deadline TD for $\varepsilon_{\rm acc} = 0.01$, $s = 0.05$, $f = 0.35$ and $p_d = 0$.

Figure 8 (b) shows the throughput as a function of deadline TD. As TD increases, the throughput tends to be large and it will reach N in each method. The throughput reaches $N = 10{,}000$ around $TD = 150$ in the proposed method; that is, all N jobs are finished within 150 time steps. On the other hand, it takes over $TD = 160$ in simple grouping or other methods (except for spot-checking method with $q = 0.1$). Since spot-checking method with $q = 0.1$ does not guarantee the reliability condition as shown in Fig. 8 (a), it is true that the proposed method shows the best performance for any TD. This figure also shows that the proposed method works well even if TD is relatively small, i.e., severe deadline.



(a) Error-rate ε (b) Throughput N_s

Fig. 9 Throughput N_s and error-rate ε vs. defection rate p_d for $\varepsilon_{\rm acc} = 0.01$, $s = 0.05$, $f = 0.35$ and $TD = 150$.

4.2.5 Performance for Defection Rate p_d

Figure 9 (a) shows error rate as a function of defection rate p_d. This figure clearly shows that, even if $q = 0.2$, spot-checking method does not guarantee the reliability condition where p_d exceeds 0.3. As p_d increases, the error rate of spot-checking method tends to be large since a saboteur can survive spot-checking by defecting from the system. This result indicates that, even if q is large, the spot-checking methods never guarantee the reliability condition since the error rates of those methods depend on the unknown parameter p_d.

Figure 9 (b) shows the throughput as a function of p_d. As p_d increases, the throughput of each method decreases since it decreases the number of results returned from workers. This figure also shows that, for any p_d, the proposed method shows the best performance among the methods which guarantee the reliability condition.

5. Conclusion

As described in this paper, we propose an adaptive group-based job scheduling method for credibility-based voting with check-by-voting to improve the throughput of VC systems, and to achieve high performance and reliable computations in VC systems. The proposed method dynamically groups workers based on the expected-credibility to decrease the number of half-finished jobs and to avoid excess job allocations. Simulation results described herein show that the proposed method improves the throughput compared to the original no grouping and the simple grouping methods, irrespective of the value of unknown parameters such as the population of saboteurs, the sabotage rate, and the defection rate.

References

1) SETI@home. http://setiathome.berkeley.edu/
2) Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D.: SETI@home: An experiment in public-resource computing, *Comm. ACM*, Vol.45, No.11, pp.56–61 (2002).
3) Kondo, D., Araujo, F., Malecot, P., Domingues, P., Silva, L.M., Fedak, G. and Cappello, F.: Characterizing Error Rates in Internet Desktop Grids, *13th European Conference on Parallel and Distributed Computing*, pp.361–371 (2007).
4) Domingues, P., Sousa, B. and Silva, L.M.: Sabotage-tolerance and trust management in desktop grid computing, *Future Generation Computer Systems*, Vol.23, No.7, pp.904–912 (2007).
5) BOINC. http://boinc.berkeley.edu/
6) Anderson, D.P.: BOINC: A System for Public-Resource Computing and Storage, *5th IEEE/ACM International Workshop on Grid Computing*, pp.4–10 (2004).
7) Sarmenta, L.F.G.: Sabotage-Tolerance Mechanisms for Volunteer Computing Systems, *Future Generation Computer Systems*, Vol.18, No.4, pp.561–572 (2002).

8) Watanabe, K. and Fukushi, M.: Generalized Spot-checking for Sabotage-tolerance in Volunteer Computing Systems, *Proc. 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (*CCGrid2010*), pp.655–660 (2010).
9) Sonnek, J., Chandra, A. and Weissman, J.: Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures, *IEEE Trans. Parallel and Distributed Systems*, Vol.18, No.11, pp.1551–1564 (2007).
10) Watanabe, K., Fukushi, M. and Horiguchi, S.: Expected-credibility-based Job Scheduling for Reliable Volunteer Computing, *IEICE Trans. Inf. Syst.*, Vol.E93-D, No.2, pp.306–314 (2010).
11) Watanabe, K., Fukushi, M. and Horiguchi, S.: Optimal Spot-checking for Computation Time Minimization in Volunteer Computing, *Journal of Grid Computing*, Vol.7, No.4, pp.575–600 (2009).

(Received May 31, 2010)
(Accepted November 5, 2010)
(Released February 9, 2011)

Kan Watanabe received his B.E. degree in information engineering, and 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 his B.S. and M.S. degrees from Hirosaki University in 1995 and 1997, respectively, and Ph.D. degree in information science from the Graduate School of Information Science at Japan Advanced Institute of Science and Technology (JAIST), in 2002. He is currently an assistant professor in the Graduate School of Information Sciences at Tohoku University. His research interests include fault-tolerant architectures, nanoscale circuits and systems, and dependable parallel and distributed systems.

Michitaka Kameyama received his B.E., M.E. and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1973, 1975, and 1978, respectively. He is currently the dean and professor in the Graduate School of Information Sciences, Tohoku University. His general research interests are intelligent integrated systems for real-world applications, advanced VLSI architecture, and new-concept VLSI including multiple-valued VLSI computing. He received the Outstanding Paper Awards at the 1984, 1985, 1987 and 1989 IEEE International Symposiums on Multiple-Valued Logic, the Technically Excellent Award from the Society of Instrument and Control Engineers of Japan in 1986, the Outstanding Transactions Paper Award from the IEICE in 1989, the Technically Excellent Award from the Robotics Society of Japan in 1990, and the Special Award at the 9th LSI Design of the Year in 2002. He is an IEEE Fellow.