IEICE TRANS. INF. & SYST., VOL.E93–D, NO.2 FEBRUARY 2010

PAPER

Expected-Credibility-Based Job Scheduling for Reliable Volunteer Computing

Kan WATANABE[†a)], ***Nonmember***, **Masaru FUKUSHI**[†b)], ***and*** **Susumu HORIGUCHI**[†c)], ***Members***

SUMMARY This paper presents a proposal of an expected-credibility-based job scheduling method for volunteer computing (VC) systems with malicious participants who return erroneous results. Credibility-based voting is a promising approach to guaranteeing the computational correctness of VC systems. However, it relies on a simple round-robin job scheduling method that does not consider the jobs' order of execution, thereby resulting in numerous unnecessary job allocations and performance degradation of VC systems. To improve the performance of VC systems, the proposed job scheduling method selects a job to be executed prior to others dynamically based on two novel metrics: expected credibility and the expected number of results for each job. Simulation of VCs shows that the proposed method can improve the VC system performance up to 11%; It always outperforms the original round-robin method irrespective of the value of unknown parameters such as population and behavior of saboteurs.

key words: *volunteer computing, job scheduling, sabotage-tolerance, credibility-based voting, expected credibility*

1. Introduction

Volunteer computing (VC) is an Internet-based parallel computing paradigm that enables any Internet participant to contribute the idle computing resources of their desktop PCs (such as CPU cycles and storage) to efforts aimed at solving large parallel problems. Today, VC has become a large-scale distributed computing platform. It is used to address difficult quantitative problems in many scientific research areas such as astronomy, biology, physics, and mathematics [1]–[7]. The most popular example is SETI@home [1], [2], which searches massive amounts of radio telescope data for signs of extra-terrestrial intelligence using hundreds of thousands of desktop PCs.

In such large-scale parallel computing systems, job scheduling is an important issue to use the huge number of available computing resources efficiently. Numerous job-scheduling methods have been proposed for use in VC systems such as resource prioritization [8], redundant requests [9], and analysis of volatility [10], [11]. Most previous job scheduling methods specifically examined reduction of the overall execution time of a parallel computation, without consideration of the correctness of the computational results.

It was recently pointed out in [12]–[14] that verification of results is crucial for any VC system. Because VC enables anyone on the Internet to join a computation, participants are not reliable. Some malicious participants (saboteurs) might falsify the results of some computations to render the overall computation useless. It is reported in [15] that a large fraction of participants (about 35%) actually returned at least one erroneous result in a real VC. Therefore, VC systems should employ an efficient job-scheduling method incorporating sabotage-tolerant mechanisms to achieve high performance and reliable computation results.

The basic mechanism for sabotage-tolerance is voting, in which a management node replicates a job and allocates them to several participants (workers) for a majority decision. Simple majority voting is examined deeply in [15]–[18]. BOINC [19], [20], the most popular VC middleware, uses *m*-first voting method, which collects *m* matching results. In these voting methods the necessary number of redundant results, i.e. redundancy, is fixed in advance. Each job requires a fixed number of results (two or more), no matter how reliable each result is. Consequently, the performance of VC systems is decreased to less than half even if ideal job scheduling is realized through the use of some scheduling method.

To improve these simple but inefficient voting methods, Sarmenta [21], [22] proposed a novel voting method called credibility-based voting [18], which enables determination of the necessary number of redundant results for each job dynamically. This method introduced "credibility" for each system element, such as a worker, job and result, to represent their correctness, and attained to guarantee the computational correctness mathematically.

Although credibility-based voting [18] is a promising approach to sabotage-tolerance of VC systems, it relies on simple round-robin job scheduling. This simple job scheduling, however, degrades the performance of VC systems because it ignores the order of execution of jobs. Different from the simple voting methods such as *m*-first voting, the necessary number of redundant results in the credibility-based voting changes dynamically through the computation. Therefore, the order of execution of jobs markedly affects the total number of redundant results produced for a computation.

As described in this paper, we propose an expected-credibility-based job scheduling method to improve the overall execution time of credibility-based voting [18]. The key idea is to introduce novel metrics for job scheduling: expected credibility and the expected number of results for each job. These metrics take account of the workers execut-

Manuscript received April 2, 2009.

[†]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
c) E-mail: susumu@ecei.tohoku.ac.jp

DOI: 10.1587/transinf.E93.D.306

ing jobs at the moment. Different from the simple round-robin method used in [18], the proposed method selects a proper job dynamically based on these metrics, thereby enabling reduction of the overall execution time of a computation.

The remainder of this paper is organized as follows. Section 2 describes the computational model of VC systems and sabotage-tolerance mechanisms. In Sect. 3, we propose the expected-credibility-based job scheduling method. Section 4 presents a demonstration of the performance of the proposed and the conventional job scheduling methods using Monte Carlo simulations. Finally, conclusions are discussed in Sect. 5.

2. Volunteer Computing Systems

2.1 Computational Model

A well known work-pool-based master–worker model [12], [16], [18]–[21] 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 an 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 might return erroneous results. Each saboteur is assumed to return an erroneous result with a constant probability s, which is known as the sabotage rate [18]. The values of f and s are 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.

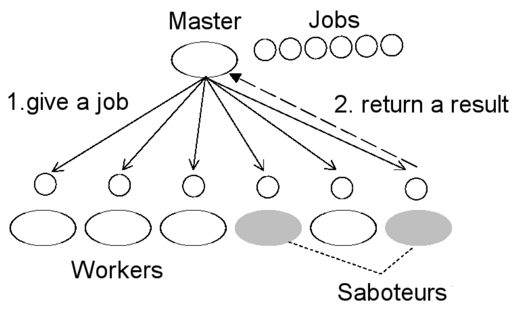


Fig. 1 Computation model of VC systems.

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: the execution time of the computation. If all workers function at the same speed and execute a job in a unit time, then T is given by N/W unit times and error rate ϵ is given as $N \times f \times s/N = f \times s$. It might be readily apparent that ϵ is proportional to the number of saboteurs 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 Spot-Checking

The basic sabotage-tolerance methods are voting and spot-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 voting and m-first voting.

- 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) 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 [19], [20].

(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 erroneous 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 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. However, blacklisting is not always effective for VC systems. Although saboteurs can be blacklisted based on their IP addresses, it is not difficult for a saboteur to rejoin the computation as a new worker. Requiring more reliable identification information such as home address will make the blacklisting more effective. However, it also diminishes volunteers' incentive to join the VC system. Based on this trade-off, we must consider both VC systems with and without blacklisting to meet the variant condition of real VCs.

2.2.2 Credibility-Based Voting

In these traditional voting methods such as majority voting and m-first voting, each job requires a fixed number of redundant results (at least two or more), no matter how reliable each result is. Consequently, the performance of VC systems diminishes to less than half even if ideal job scheduling is realized using some scheduling methods.

To improve these inefficient voting methods, Sarmenta [18] 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 a worker, result, result group, and job is assigned a credibility value that represents its correctness. Every worker and result has different credibility, which affect the credibility of the result groups. A job is finished when the credibility of any result group reaches a threshold θ. Therefore, different from the m-first voting, the necessary number of results to finish a job is not fixed and is generally smaller than that of the m-first voting.

Furthermore, 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, $f_{max} \times W$. Because of these two major advantages, for this study, we use credibility-based voting as a sabotage-tolerance mechanism to reduce the error rate and guarantee the computational correctness.

2.3 Job Scheduling Problem

A sabotage-tolerance mechanism generally requires some extra execution time (overhead) because of a redundant computation. For example, when each job is replicated and allocated to m workers for a voting, it increases the overall execution time by m times compared to the case of non-redundant computation. Similarly, spot-checking with the spot-check rate q [18] increases the execution time by $1/(1-q)$ times.

In basic voting methods, the necessary number of results for each job is fixed in advance (e.g. m matching results are required in m-first voting). In this case, the order of execution of jobs does not affect the total number of results produced for a computation. Even if the order of execution of jobs is changed, the overall execution time remains unchanged.

In contrast, with credibility-based voting, the necessary number of results for each job is not fixed in advance because it depends on the credibility of each result. The credibility of results changes as the computation proceeds, depending on the result of spot-checking. In this case, the order of execution of jobs directly affects the total number of results. Therefore, in credibility-based voting, job scheduling method have a considerable impact on the overhead and the execution time of the computation.

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

> *select a job that should be allocated to an idle worker prior to other jobs for reducing execution time T to the greatest extent possible, while guaranteeing the computational correctness as* $\epsilon \leq \epsilon_{acc}$ *for any given* ϵ_{acc}.

Basic job scheduling methods for credibility-based voting are the random and round-robin methods [18]. 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, they are not efficient because they do not take account of a job's progress.

3. Expected-Credibility-Based Job Scheduling Method

3.1 Outline of the Proposed Method

We propose a new job scheduling method for credibility-based voting [18] that can consider the progress of each job to reduce the overhead and execution time T. We define two novel metrics for each job: the expected credibility and the expected number of results. Using these two metrics, the proposed scheduling method can select a proper job to be executed prior to others, thereby achieving a reduction in the execution time of a computation.

The proposed method employs spot-checking with a constant rate q and backtracking, as in credibility-based voting [18].

3.2 Definitions of Credibility

Credibility is defined for each system element to represent their correctness [18]. The credibility of a worker w (denoted by $C_W(w)$) is determined by how many times w survives spot-checking (how many times w returns the correct result for spotter jobs). When blacklisting is used, if w survives spot-checking k times, $C_W(w)$ is given by Eq. (1).

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

Therein, e is Napier's constant. $C_W(w)$ is given by Eq. (2) when blacklisting is not used.

$$C_W(w) = \begin{cases} 1 - f_{max}, & \text{if } k = 0 \\ 1 - \frac{f_{max}}{k}, & \text{otherwise.} \end{cases} \tag{2}$$

The credibility of a result r produced by worker w (denoted by $C_R(r)$) is equal to $C_W(w)$.

$$C_R(r) = C_W(w). \tag{3}$$

The results collected for a job are grouped into several result groups, $G_1, \ldots, G_g$, each of which includes all results having the same value. The credibility of a result group G_a (denoted by $C_G(G_a)$) is given by the following Eq. (4):

$$C_G(G_a) = \frac{P_T(G_a) \prod_{i \neq a} P_F(G_i)}{\prod_{i=1}^{g} P_F(G_i) + \sum_{n=1}^{g} P_T(G_n) \prod_{i \neq n} P_F(G_i)}, \tag{4}$$

in which the following are used.

$$P_T(G_a) = \prod_{r \in G_a} C_R(r), \tag{5}$$

$$P_F(G_a) = \prod_{r \in G_a} (1 - C_R(r)). \tag{6}$$

Therein, $P_T(G_a)$ ($P_F(G_a)$) is the probability that all results in G_a are correct (incorrect). In Eq. (4), $C_G(G_a)$ represents the conditional probability that the results in G_a are correct and those in all other groups are incorrect, for a given combination of the results groups.

The credibility of a job j (denoted by $C_J(j)$) is equal to $C_G(G_x)$, where G_x is a result group that has a maximum credibility among all result groups for job j.

$$C_J(j) = C_G(G_x) = \max_{1 \leq a \leq g} C_G(G_a). \tag{7}$$

When $C_J(j)$ reaches threshold θ ($= 1 - \epsilon_{acc}$), the result of the group G_x is accepted as the final result of job j; then job j is finished.

3.3 Expected-Credibility-Based Job Scheduling

3.3.1 Definitions of Expected Credibility

To select a job for prior execution, it is important to consider the progress of each job. The simple metric to represent the progress might be the credibility of the job because a job is finished when the credibility of the job reaches threshold θ. However, this metric is insufficient to grasp the proper progress because there might be several workers who are engaged simultaneously in the execution of the job. Consider the case in which job j is allocated to several workers, as shown in Fig. 2. According to Eq. (7), $C_J(j)$ is calculated from the credibility of returned results. In the calculation, the workers who are currently executing job j are not examined ($w_1, \ldots, w_d$ in Fig. 2). Those workers will return their results, which affect $C_J(j)$; therefore, the consideration of such workers is necessary to grasp the proper progress of jobs.

Based on the idea described above, we define two new metrics for each job: the expected number of results and the expected credibility. Presuming that job j has several results which can be grouped into $G_1, \ldots, G_g$ and presuming that there exist d workers ($w_1, \ldots, w_d$) who are executing job j as shown in Fig. 2, then the expected number of results and the expected credibility of job j, denoted by $ENR(j)$ and $EC_J(j)$ respectively, are defined as follows.

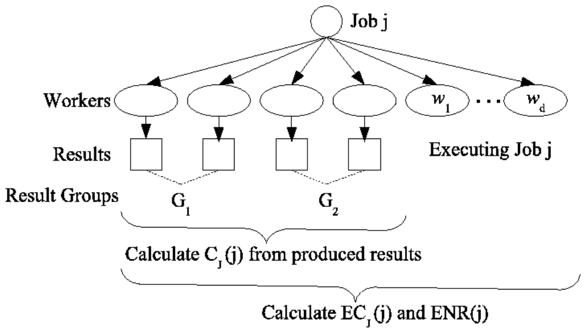

Job j
Workers
w_1
w_d
Results
Executing Job j
Result Groups
G_1
G_2
Calculate $C_J(j)$ from produced results
Calculate $EC_J(j)$ and ENR(j)

Fig. 2 Calculation of $EC_J(j)$ and $ENR(j)$.

- The expected number of results (ENR)
  $ENR(j)$ is defined as
  $$ENR(j) = \sum_{i=1}^{g} |G_i| + d, \tag{8}$$
  where $|G_i|$ represents the number of results in G_i.
  The metric $ENR(j)$ represents the number of results of job j when all d workers return their results for job j.
- Expected credibility (EC_J)
  $EC_J(j)$ is defined as
  $$EC_J(j) = \max_{1 \leq a \leq g} C'_G(G_a), \tag{9}$$
  where
  $$C'_G(G_a) = \frac{P'_T(G_a) \prod_{i \neq a} P'_F(G_i)}{\prod_{i=1}^{g} P'_F(G_i) + \sum_{n=1}^{g} P'_T(G_n) \prod_{i \neq n} P'_F(G_i)}, \tag{10}$$
  $$P'_T(G_a) = \begin{cases} P_T(G_a), & \text{if } a \neq x \\ P_T(G_a) \times \prod_{i=1}^{d} C_W(w_i), & \text{if } a = x, \end{cases} \tag{11}$$
  $$P'_F(G_a) = \begin{cases} P_F(G_a), & \text{if } a \neq x \\ P_F(G_a) \times \prod_{i=1}^{d} (1 - C_W(w_i)), & \text{if } a = x. \end{cases} \tag{12}$$
  In those equations, $P'_T(G_a)$ ($P'_F(G_a)$) is the probability that all results in G_a and all d workers are correct (incorrect).
  The metric $EC_J(j)$ predicts the credibility of job j supposing that all d workers return the same result, which joins the result group G_x, where G_x is the result group which has a maximum credibility in all groups.

3.3.2 Job Scheduling Algorithm

Using metrics EC_J and ENR, we propose a new job scheduling method called "expected-credibility-based job scheduling". Figure 3 shows the algorithm of the expected-credibility-based job scheduling. First, all idle workers are

```
1  Push idle workers in worker queue $Q$;
2  Group all unfinished jobs into $S_i$;
3    // $S_i$ is a set of unfinished jobs of which $ENR$ are $i$
4  $i = 0$;
5    // Select jobs from $S_0$ at first
6
7  while ( Q is not empty ) do
8    while ( $S_i$ != $\phi$ ) do
9      Pop worker $w$ from $Q$;
10     Allocate a spotter job to worker $w$ with rate $q$;
11     if( No job is allocated to $w$ ) then
12       Select job $j$ which has a minimum $EC_J$ in $S_i$;
13       Allocate job $j$ to $w$;
14       Update $EC_J(j)$;
15       $S_i = S_i - \{j\}$;
16       $S_{i+1} = S_{i+1} + \{j\}$;
17     end if
18     if ( Q is empty ) then
19       exit;
20     end if
21   end while
22   $i = i + 1$;
23 end while
```

Fig. 3 Expected-credibility-based job scheduling algorithm.

stored in a worker queue (line 1 in Fig. 3) and all unfinished jobs are grouped based on the value of ENR (line 2). Let S_i be the set of unfinished jobs of which ENR are equal to i. 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 7–23). When a worker w is extracted from the queue (line 9), by the spot-checking mechanism, a spotter job is allocated to w with spot-check rate q. One unfinished job is allocated to w if a spotter job is not allocated. In this case, the master selects job j, which has a minimum $EC_J(j)$ in S_i (line 12). Once job j is allocated to w, then $EC_J(j)$, S_i, and S_{i+1} are updated immediately to reflect the allocation of job j in the subsequent scheduling (lines 14–16).

Using this scheduling method, jobs in S_i are selected prior to those in S_{i+1}, and among S_i, a job having the minimum EC_J is selected prior to others. That is, jobs are selected in the ascending order of ENR and EC_J.

This scheduling policy arises from the careful observation that there exist many unnecessary job allocations to satisfy a job's termination condition: i.e. $C_J(j) \geq 1 - \epsilon_{acc}$. This occurs because C_J changes dynamically during a computation. In fact, $C_J(j)$ changes when the master (1) collects new results for job j, or (2) collects results for a spotter job from the workers which have returned results for job j. Regarding the second case, when a worker w survives spot-checking, $C_W(w)$ and $C_R(r)$ increase as in Eqs. (1)–(3), which in turn increases the job's credibility $C_J(j)$. Because spot-checking is executed with a constant rate q, $C_J(j)$ tends to continue to increase during the computation, which implies that job j might satisfy the termination condition with no new results. Meanwhile, using a job scheduling mechanism, the master continues to allocate the unfinished job j and collects new results so that $C_J(j)$ satisfies the termination condition. If $C_J(j)$ exceeds the threshold without the newly collected results, then the new results will turn out to be unnecessary for job j. In such a case, such additional job allocations can be said to be "unnecessary job allocations".

The key factor to reduce the overall execution time is how to save such unnecessary job allocations. It is, however, difficult to find during a computation which results will be unnecessary at the end of the computation because the credibility of jobs changes dynamically depending on the unpredictable results of jobs returned for normal and spotter jobs. Note that jobs which have lower credibility and fewer results require more results to increase the credibility; that is, to save unnecessary job allocations, a job which has lower EC_J and ENR should be selected prior to others during job scheduling. If jobs are selected in the ascending order of EC_J without considering the number of results, the jobs for which erroneous results have been returned will be selected many times since the presence of erroneous results makes it more difficult to increase the jobs' credibility . Such job scheduling will cause performance degradation because of unnecessary job allocations, as verified in our performance evaluation presented in Sect. 4.2.2. Therefore, as described above, the proposed job scheduling method selects jobs in the ascending order of ENR and EC_J.

3.4 Scheduling Cost

Next, the scheduling cost is analyzed for both the round-robin and our proposed methods. In VC systems, scheduling is the repetitive process of allocating jobs and receiving results. As described in this paper, the scheduling cost is defined as the time (worker's waiting time) to select one job for a worker when the master receives a result from the worker.

Let t_c be the time required for the master to calculate a credibility of one job, C_J; also, t_e is the time to calculate the ENR of one job. The scheduling cost at time step t (t-th unit time) is denoted as $Cost(t)$. The scheduling cost varies according to when it is calculated. Therefore, we calculate $Cost(t)$, and then average of $Cost(t)$ for both methods.

In the round-robin method [18], when the master receives a result for a normal job j, the master simply calculates only $C_J(j)$ in t_c to check whether it reaches the threshold or not. If the master receives a result for a spotter job, the result will affect C_J of several jobs because of backtracking or updating of the credibility. The number of jobs which need the update of the credibility at time step t is, at most, t because the number of results produced by w is, at most, t even if w can return a result in every time step. The scheduling cost for the round-robin method, i.e. $Cost_{orig}(t)$, is calculated as

$$Cost_{orig}(t) = t_c((1-q) + qt). \tag{13}$$

Time step t changes from 1 to T, where T is the overall execution time of the computation represented in unit times. The average of $Cost_{orig}(t)$, i.e. $E(Cost_{orig})$ is given as

$$\begin{aligned} E(Cost_{orig}) &= \frac{1}{T}\sum_{t=1}^{T} Cost_{orig}(t) \\ &= t_c\left((1-q) + \frac{q(T+1)}{2}\right). \end{aligned} \tag{14}$$

Because the round-robin method is the simplest scheduling method, this cost is the minimum required for credibility-based voting.

In the proposed method, when the master receives a result for a normal job j, the master calculates $EC_J(j)$ and $ENR(j)$ in addition to $C_J(j)$. $EC_J(j)$ and $ENR(j)$ must also be updated when a normal job j is allocated to a worker (as shown in lines 13–16 in Fig. 3). Therefore, for each allocation of a normal job (probability $1-q$), the master calculates $EC_J(j)$ and $ENR(j)$ twice. If the master receives a result for a spotter job, the master calculates EC_J and ENR in addition to C_J for, at most, t jobs as described above. The scheduling cost for the proposed method, i.e. $Cost_{prop}(t)$, is calculated as

$$\begin{aligned} Cost_{prop}(t) &= (1-q)(t_c + 2(t_c + t_e)) + qt(t_c + (t_c + t_e)) \\ &= Cost_{orig}(t) \\ &\quad + (t_c + t_e)(2(1-q) + qt). \end{aligned} \tag{15}$$

By definition, the time to calculate EC_J is almost identical as t_c (the time to calculate C_J).

The average of $Cost_{prop}(t)$, i.e. $E(Cost_{prop})$, is given as the following.

$$\begin{aligned} E(Cost_{prop}) &= \frac{1}{T}\sum_{t=1}^{T} Cost_{prop}(t) \\ &= 2 \times t_c\left((1-q) + \frac{q(T+1)}{2}\right) \\ &\quad + t_c(1-q) \\ &\quad + t_e\left(2(1-q) + \frac{q(T+1)}{2}\right). \end{aligned} \tag{16}$$

The value of q is between 0 and 1 and $T >> 1$ (several orders of magnitude). Therefore, the second term in Eq. (16) is negligibly small compared to the first. The third term in Eq. (16) is also negligibly small compared to the first because calculation of ENR involves the simple addition shown in Eq. (8), then t_e is negligibly small. Therefore, $E(Cost_{prop})$ can be represented as follows.

$$E(Cost_{prop}) \approx 2 \times E(Cost_{orig}). \tag{17}$$

From Eqs. (14) and (17), it is apparent that the scheduling costs of both methods are negligibly small compared to the execution time of a job in general VC systems (e.g. several hours in [1]), because t_c, the calculation cost of one credibility, is a very small value (on the order of microseconds). Furthermore, both scheduling costs (for a single worker) are independent of the system scale, such as the number of jobs and the number of workers. Consequently, both methods will work well in large-scale VC systems, without a considerable increase in the workers' waiting time.

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. Execution times T of VCs are evaluated as the average of 10000 simulation results for five different job scheduling methods: the proposed method, random, round-robin [18], and two variants of the proposed method. The variants of the proposed method are used as a reference against the proposed method. Those methods select jobs using either ENR or EC_J. They are called, respectively, "ENR method" and "EC_J method".

- ENR method: This method selects a job that has a minimum ENR in all unfinished jobs. A round-robin selection is used if some jobs have the same ENR.
- EC_J method: This method selects a job that has a minimum EC_J in all unfinished jobs.

Table 1 Simulation parameters.

# of jobs (N)	10000
# of workers (W)	100
spot check rate (q)	0.1, 0.2
acceptable error rate (ϵ_{acc})	0.01
faulty fraction (f)	0 ~ f_{max} (0.4)
sabotage rate (s)	0 ~ 1
defection rate (p_{down})	0 ~ 0.4

The parameters used in our simulation are shown in Table 1. Because some parameters are unknown to the master and are uncontrollable, such as f and s, 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 0.4 reflecting the result of an experiment in a real VC environment [15].

We make two existing assumptions as in [18] for fair evaluations between different job-scheduling methods. First, all workers have the same processing speed. Therefore, jobs are distributed equally among the workers and a job is executed in a unit time. Second, a saboteur knows when it is caught by spot-checking in a system without blacklisting. After having been caught, the saboteur immediately rejoins to the system as a new worker.

In addition to these existing assumptions, 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 turn of the computation. 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.

4.2 Simulation Results

4.2.1 In Cases with Blacklisting

Simulation results for VC systems with blacklisting are shown in Figs. 4–7.

Figure 4 shows execution time T as a function of f

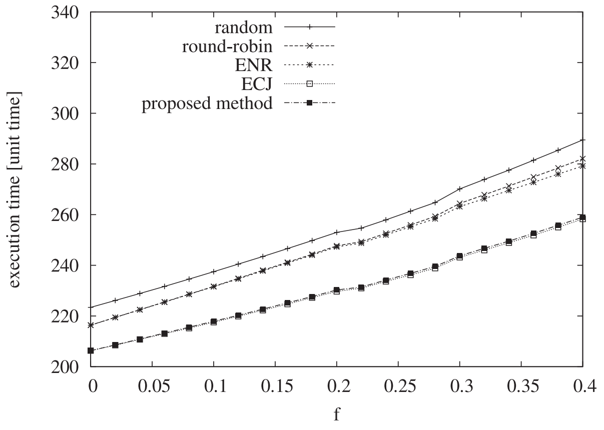


Fig. 4 Execution time vs. faulty fraction f for $s = 0.1$, $q = 0.1$, and $p_{down} = 0.0$ (with blacklisting).

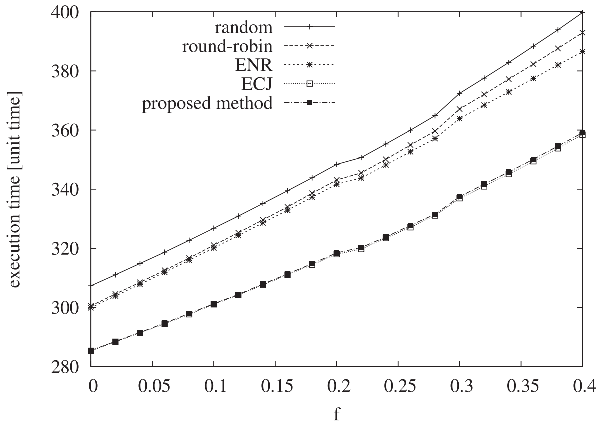


Fig. 5 Execution time vs. faulty fraction f for $s = 0.1$, $q = 0.1$, and $p_{down} = 0.1$ (with blacklisting).

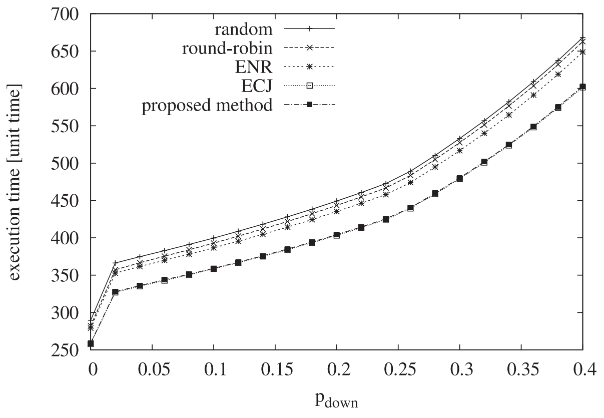


Fig. 6 Execution time vs. defection rate p_{down} for $s = 0.1$, $q = 0.1$, and $f = 0.4$ (with blacklisting).

for $p_{down} = 0$. As f increases, the execution time of each method increases because the erroneous results returned from saboteurs increase proportionally to f.

The proposed method outperforms the round-robin method for any f. Although the actual value of f is unknown to the master before the computation, the proposed method can improve the execution time even if no saboteur exists: $f = 0$. This is true because different jobs have different credibility by the effect of spot-checking. Our proposed method performs well in this situation.

In addition, the performance curve of the EC_J method is close to that of the proposed method, whereas the curve of the ENR method is close to that of the round-robin method. The job scheduling method based only on the number of results, the ENR method, does not improve the execution time markedly because the credibility of jobs is different even though those jobs have the same ENR. To improve the execution time efficiently, a job scheduling method must take account of the job's credibility, as the proposed method does.

Figure 5 displays execution time T as a function of f for $p_{down} = 0.1$. Compared to the case of $p_{down} = 0$ (in Fig. 4), the execution time of each method increases because the number of results collected in a unit time decreases as a result of workers' defection. The difference between the proposed method and the round-robin method increases compared to the case of $p_{down} = 0$.

We evaluate execution time T for various values of defection rate p_{down} to verify the effect of workers' defection on each scheduling method. Figure 6 portrays the execution time T as a function of defection rate p_{down}. As p_{down} increases, workers leave and rejoin the system more frequently, the execution time of each method increases (e.g. the execution time T for each method at $p_{down} = 0.4$ is more than twice that at $p_{down} = 0$). The proposed method outperforms the round-robin method for any p_{down}.

Figure 7 portrays the execution time T as a function of sabotage rate s. The execution time of each method increases at first at a sabotage rate s of 0.2, then retains almost identical values for larger s. This is true because, when s is large, all saboteurs are caught until the end of the computation and all results produced by the saboteurs are invalidated, whether erroneous or not.

The proposed method outperforms the round-robin method for any s, even if saboteurs never return erroneous

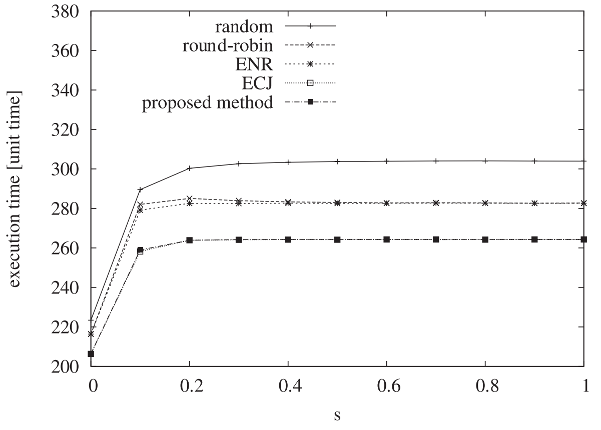


Fig. 7 Execution time vs. sabotage rate s for $f = 0.4$, $q = 0.1$, and $p_{down} = 0.0$ (with blacklisting).

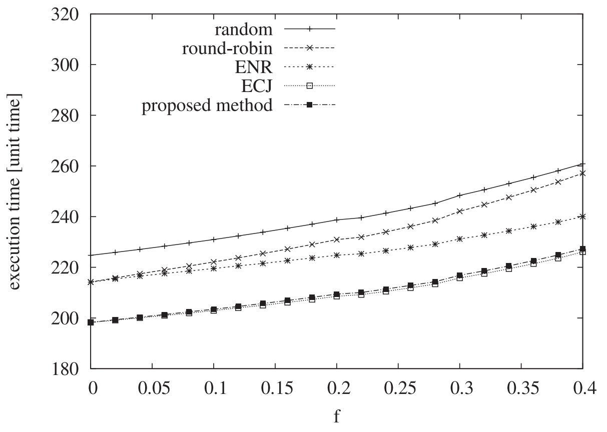


Fig. 8 Execution time vs. faulty fraction f for $s = 0.1$, $q = 0.2$, and $p_{down} = 0.0$ (without blacklisting).

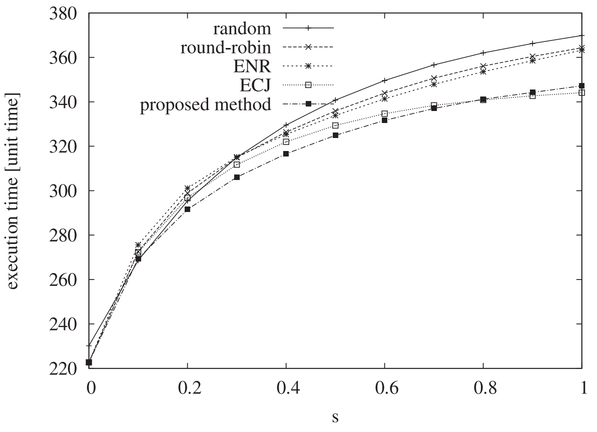


Fig. 10 Execution time vs. sabotage rate s for $f = 0.4$, $q = 0.1$, and $p_{down} = 0.0$ (without blacklisting).

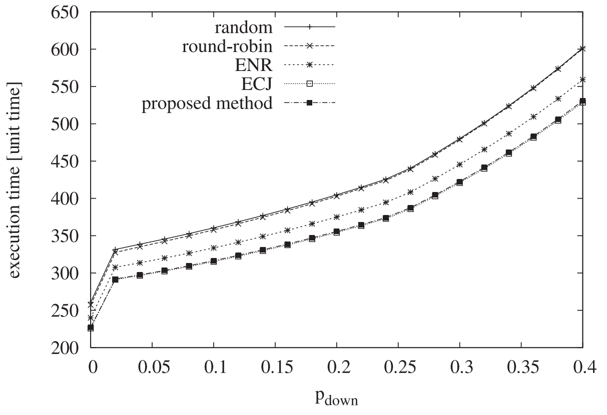


Fig. 9 Execution time vs. defection rate p_{down} for $s = 0.1$, $q = 0.2$, and $f = 0.4$ (without blacklisting).

results ($s = 0$). Although the actual value of s is also unknown to the master like f, the proposed method can improve the execution time irrespective of the value s, i.e., the behavior of saboteurs.

As shown in Figs. 4–7, the proposed method always outperforms the round-robin method, irrespective of the unknown parameters f, s and p_{down}. This very important feature indicates that the proposed method is applicable to VC systems of various environments.

4.2.2 In Cases without Blacklisting

Simulation results for VC systems without blacklisting are shown in Figs. 8 to 10.

Figure 8 shows execution time T as a function of f. The proposed method outperforms the round-robin method for any f, just as in the cases with blacklisting. Particularly, when $f = 0.4$, the difference between the proposed method and the round-robin method is about 11%.

Figure 8 also shows that the difference between the ENR method and the round-robin method is large, different from cases with blacklisting (Fig. 4). In cases without blacklisting, the saboteurs can rejoin the system and produce erroneous results permanently, so that more results are invalidated by backtracking and each job's ENR becomes widely disproportionate. This result indicates that consideration of ENR becomes an important reason to select jobs in cases without blacklisting.

Figure 9 depicts execution time T as a function of defection rate p_{down}. The proposed method outperforms the round-robin method for any p_{down}, as in cases with blacklisting.

Figure 10 shows execution time T as a function of sabotage rate s. In cases without blacklisting, the sabotage rate s affects the execution time, different from cases with blacklisting. This is true because the sabotage rate s is the ratio of erroneous results to all results produced using a saboteur. In cases without blacklisting, although a saboteur will be caught, the saboteur rejoins the system and produces erroneous results permanently. Some of produced erroneous results are retained in the system until the end of the computation, making it more difficult to increase the jobs' credibility. The number of these erroneous results is proportional to s. For that reason, the execution time increases as s increases in cases without blacklisting.

Figure 10 also shows that the EC_J method takes a longer execution time than other methods when s is small. This is true because the EC_J method does not take account of the number of results for each job. By selecting a job in the ascending order of EC_J, a job having erroneous results is selected prior to other jobs because the credibility of such a job is smaller, as shown in Eqs. (4)–(7). On the other hand, when erroneous results are invalidated by backtracking, the credibility of such jobs becomes considerably larger; some results of the job will turn out to be unnecessary. Particularly, when s is small, it is difficult to catch saboteurs and invalidate erroneous results. Consequently, jobs having erroneous results are selected repeatedly and the number of unnecessary job allocations increases. For that reason, the EC_J method requires a longer execution time than other methods, as shown in Fig. 10.

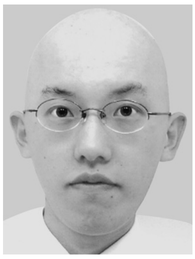
By selecting jobs in the ascending order of ENR, a job having numerous results is not selected in job scheduling even if the job has erroneous results. In this case, the unnecessary job allocations do not become numerous, as with the EC_J method. Therefore, the proposed method, which selects jobs in the ascending order of ENR at first and then selects a job based on EC_J, shows better performance for any s.

5. Conclusion

As described in this paper, we proposed expected-credibility-based job scheduling for credibility-based voting [18] to improve the performance of the original round-robin job scheduling, and to achieve high performance and reliable computations in VC systems. The proposed job scheduling method selects jobs to save unnecessary job allocations based on two novel metrics: the expected credibility and the expected number of results. Simulation results described herein show that the proposed method reduces the execution time compared to the round-robin method, 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/, Sept. 2008.
[2] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An experiment in public-resource computing," Commun. ACM, vol.45, no.11, pp.56–61, 2002.
[3] Folding@home, http://folding.stanford.edu/, Sept. 2008.
[4] FightAIDS@home, http://fightaidsathome.scripps.edu/, Sept. 2008.
[5] GIMPS, http://www.mersenne.org/, Sept. 2008.
[6] distributed.net, http://www.distributed.net/, Sept. 2008.
[7] Einstein@Home, http://einstein.phys.uwm.edu/, Sept. 2008.
[8] D. Kondo, A.A. Chien, and H. Casanova, "Scheduling task parallel applications for rapid turnaround on enterprise desktop grids," J. Grid Computing, vol.5, no.4, pp.379–405, 2007.
[9] H. Casanova, "Benefits and drawbacks of redundant batch requests," J. Grid Computing, vol.5, no.2, pp.235–250, 2007.
[10] E.J. Byun, S.J. Choi, M.S. Baik, and C.S. Hwang, "Scheduling scheme based on dedication rate in volunteer computing," 4th International Symposium on Parallel and Distributed Computing, pp.234–241, 2005.
[11] D. Kondo, G. Fedak, F. Cappello, A.A. Chien, and H. Casanova, "Characterizing resource availability in enterprise desktop grids," Future Gener. Comput. Syst., vol.23, no.7, pp.888–903, 2007.
[12] 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.
[13] F. Martins, M. Maia, R.M. de Castro Andrade, A.L. dos Santos, and J. Neuman de Souza, "Detecting malicious manipulation in grid environments," 18th International Symposium on Computer Architecture and High Performance Computing, pp.28–35, 2006.
[14] S. Zhao, V. Lo, and C. GauthierDickey, "Result verification and trust-based scheduling in peer-to-peer grids," 5th IEEE International Conference on Peer-to-Peer Computing, pp.31–38, 2005.
[15] 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.
[16] J. Sonnek, A. Chandra, and J. Weissman, "Adaptive reputation-based scheduling on unreliable distributed infrastructures," IEEE Trans. Parallel Distrib. Syst., vol.18, no.11, pp.1551–1564, 2007.
[17] 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.
[18] L.F.G. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems," Future Gener. Comput. Syst., vol.18, no.4, pp.561–572, 2002.
[19] BOINC, http://boinc.berkeley.edu/, Sept. 2008.
[20] D.P. Anderson, "BOINC: A system for public-resource computing and storage," 5th IEEE/ACM International Workshop on Grid Computing, pp.4–10, 2004.
[21] Bayanihan, http://www.cag.lcs.mit.edu/bayanihan/, Sept. 2008.
[22] L.F.G. Sarmenta and S. Hirano, "Bayanihan: Building and studying web-based volunteer computing systems using Java," Future Gener. Comput. Syst., vol.15, no.5–6, pp.675–686. 1999.



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.

Susumu Horiguchi received his M.E. and D.E. degrees from Tohoku University in 1978 and 1981, respectively. He is currently a full professor in the Graduate School of Information Science, Tohoku University. He was a visiting scientist at the IBM Thomas J. Watson Research Center from 1986 to 1987 and a visiting professor at The Center for Advanced Studies, the University of Southwestern Louisiana and at the Department of Computer Science, Texas A&M University in the summers of 1994 and 1997. He was also a professor in the Graduate School of Information Science, JAIST (Japan Advanced Institute of Science and Technology). He has been involved in organizing many international workshops, symposia and conferences sponsored by the IEEE, IEICE and IPS. His research interests have been mainly concerned with interconnection networks, parallel computing algorithms, massively parallel processing, parallel computer architectures, VLSI/WSI architectures, and Multi-Media Integral Systems. Prof. Horiguchi is a senior member of the IEEE Computer Society, and a member of the IPS, and IASTED.