Group-Based Scheduling Scheme for Result Checking in Global Computing Systems*

HongSoo Kim[1], SungJin Choi[1], MaengSoon Baik[1], KwonWoo Yang[2], HeonChang Yu[3], and Chong-Sun Hwang[1]

[1] Dept. of Computer Science & Engineering, Korea University,
1, 5-Ga, Anam-Dong, SungBuk-Gu, Seoul, 136-701, Rep. Korea
{hera, lotieye, msbak, hwang}@disys.korea.ac.kr
[2] Gongju National University of Education,
376, BongHwang-Dong, GongJu city, 314-711, Rep. Korea
kwyang@gongju-e.ac.kr
[3] Dept. of Computer Education, Korea University,
1, 5-Ga, Anam-Dong, SungBuk-Gu, Seoul, 136-701, Rep. Korea
yuhc@comedu.korea.ac.kr

Abstract. This paper considers the problem of correctness to fault-tolerance in global computing systems. Global computing system has been shown to be exposed to intentional attacks, where authentication is not relevant, network security techniques are insufficient. To guarantee correctness for computation, fault-tolerance schemes have been used to majority voting and spot-checking but these schemes intend to high computation delay because are not applied scheduling scheme for result checking. In this paper, we propose a new technique called GBSS(Group-Based Scheduling Scheme) which guarantee correctness and reduce computation delay by using fault-tolerant scheduling scheme to the result checking. Additionally, simulation results show increased usable rate of the CPU and increased performance of the system.

1 Introduction

Global computing systems are computing paradigm to run high-throughput computing applications by using idle time of internet connected computers[12]. One of main characteristics of the systems is that computation nodes are freely leaved or joined according to their volatile property. Moreover, there exists different administrator for each nodes. These projects have ventured to study and develop global computing systems such as SETI@home [2], Korea@home [14], Distributed.net [9], Entropia [8], Bayanihan [3], XtremWeb [4][12], Charlotte [6], Cilk [5], GUCHA [13].

Global computing systems assume relatively unreliable computing resources because of the computation interference or submitting bad result of malicious

* This work was supported by grant No. G- 04-GS-01-03R-7 from the Korea Institute of Science and Technology Information.

workers. If malicious workers are submitting bad result, then all results could be canceled. Practically, in the SETI@home, malicious workers returned bad result as change original code[2]. Therefore, global computing systems have to consider a result checking scheme for the computed result.

In previous work, a result checking scheme for the computed result presented majority voting and spot-checking. A majority voting[7] scheme adapt result if results are same value compare to least three result. This scheme has an expected redundancy of 2k+1, so this scheme is becoming insufficient because of a waste of the resource. In spot-checking scheme[3], the master node does not redo all the work objects two or more times, but instead randomly gives a worker a spotter work object whose correct result is already known or will be known by checking it in some manner afterwards. These previous works are not applied a scheduling scheme, so they have been the high computation delay and the high error rate.

In this paper, we propose fault-tolerant scheduling scheme through organized group as credibility of worker. First, we calculate credibility of each workers by spot-checking scheme. Second, we organize group through credibility-based group organization scheme(CBGOS). Finally, we propose group-based scheduling scheme. The GBSS guarantee correctness and reduce computation delay by using fault-tolerant scheduling scheme.

The rest of paper is structured as follows. In section 2, we introduce executing mechanism and assumption for global computing system model. In section 3, this paper describe group-based scheduling scheme using the credibility of each workers. In section 4, we describe implementation and performance evaluation. Finally, in section 5, we discuss conclusions and future work.

2 Global Computing System Model and Assumptions

In this paper, we assume a work pool-based master-worker model. This model is used in practically all internet-based distributed computing systems. As show in **fig. 1**, a computation is divided into sequence of batches, each of which consists of many mutually independent work objects. This work objects are located into work pool by task allocation server(TAS). Work objects are assigned to each workers by group-based scheduling scheme, and then they are executed computation in worker, after then scheduler return results to the TAS. Each workers request new work object in work-pool and then they complete a batch as execute a work objects by work stealing[5]. Also, we assume single-instruction multiple-data(SIMD) model which execute single code through each different data.

For some applications, the acceptable error rate can be relatively high, about 1% or more. These include applications such as image or video rendering. But also applications as climate forecast, acceptable error rate can be relatively low, near 0%. Therefore, global computing systems need the result checking mechanism to correctness for result. In this paper, we assume malicious failure model as follows. Malicious failure means worker return malicious bad results. So, we focus this problem that worker return bad results by malicious failure. Therefore, we propose schemes to solve this problem.

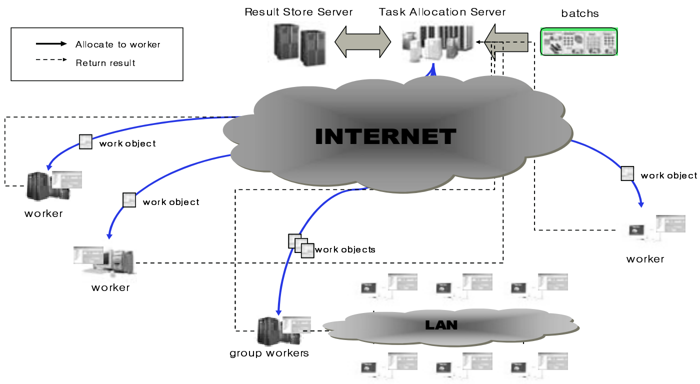

Fig. 1. Global Computing System Model

3 Group-Based Scheduling Scheme for Result Checking

We describe the CBGOS and the GBSS proposed in this paper. Although work has been made on the credibility-based fault tolerance scheme using spot checking and majority voting proposed in [3], this scheme is not applied fault-tolerant scheduling scheme. In our work, we propose scheduling scheme as credibility-based grouping of each workers. Therefore, we can be guaranteed correctness for computed result and reduce the computation delay by result checking scheme.

3.1 Overview

The GBSS is executed by grouping using the credibility threshold as **fig. 2**. We apply GBSS as credibility-based grouping to guarantee correctness of executed result by workers. First of all, this scheme have to calculated credibility C_v of a worker v_i by majority voting and spot-checking before assign work object to the workers. If we only accept a result for a work object when the probability of that

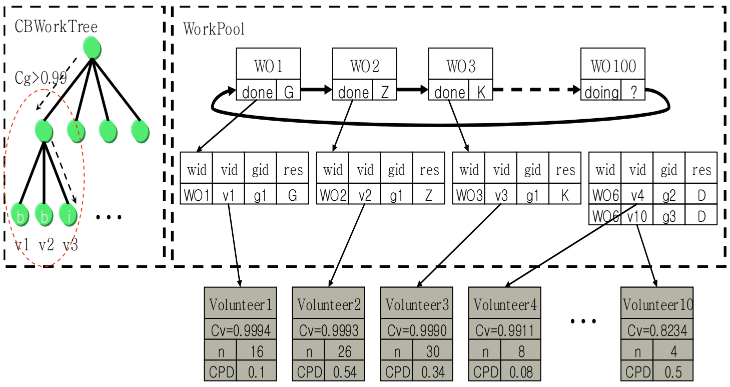
CBWorkTree
Cg>0.99
b b i
v1 v2 v3
WorkPool
WO1 done G
WO2 done Z
WO3 done K
WO100 doing ?
wid vid gid res
WO1 v1 g1 G
WO2 v2 g1 Z
WO3 v3 g1 K
WO6 v4 g2 D
WO6 v10 g3 D
Volunteer1 Cv=0.9994 n 16 CPD 0.1
Volunteer2 Cv=0.9993 n 26 CPD 0.54
Volunteer3 Cv=0.9990 n 30 CPD 0.34
Volunteer4 Cv=0.9911 n 8 CPD 0.08
Volunteer10 Cv=0.8234 n 4 CPD 0.5

Fig. 2. Work Pool and Work Tree to Group-based Scheduling

result being correct is at least threshold θ, then the probability of accepting a correct result would be at least θ. θ has different value as each applications and calculated as $\theta = 1$ - ε by acceptable error rate ε.

In the global computing system, the executing applications have each different error rate *f*. An executed result by worker could be accepted if results have verify the reliability by the majority voting. First of all, when scheduler assigns work object, it execute concurrently assignment for point of group by credibility of group. As error rate, it determine a number of redundancy $k/(1-f)$ and assign to itself or to upper group when execute k+1 majority voting scheme. k+1 majority voting adapt more than half in k+1, this is to reduce redundancy. If it lead to bad result, executed result by majority voting scheme as a redundancy, then we have one more redundancy. In the next section, we guarantee more higher performance than previous schemes by group-based scheduling scheme.

3.2 Credibility

We calculate credibility of workers to guarantee correctness for executed results by workers. This paper defines credibility of a worker as follows.

Definition 1. *Credibility(*C*). The credibility is a determining foundation of correctness as executed computation result by worker. This use error rate* f*, participated degree in computation and the number of spot-checks passed by a worker,* n*, to estimate likely a worker is to give a good result.*

$$C_{v_i} = 1 - \frac{f}{n} \cdot CPD(v_i)(n > 0) \tag{1}$$

$$C_{v_i} = 1 - f \cdot CPD(v_i)(n = 0) \tag{2}$$

Definition 2. *Computation Participation Degree(*CPD*). The CPD define that worker is the rate of how long does it participate at the computation, so we calculate really participated time at the computation. The participation degree at computation $CPD_k(v_i)$ for a worker v_i calculate as follows.*

$$CPD_k(v_i) = 1 - \frac{CJT_k(v_i)}{CJT_k(v_i) + CLT_k(v_i)} \tag{3}$$

In **equation 1**, C_{v_i} is the credibility of the worker v_i and n is the number of spot-checks returned by a worker and *f* is the probability that a worker chosen at random would be bad. If n is 0, then it would be calculated by **equation 2**. And participation degree of computation $CPD_k(v_i)$ is calculated by the rate of really participated term for really participated term of worker and leaved term of worker. In **equation 3**, $CJT_k(v_i)$ is participated term of v_i in computation and $CLT_k(v_i)$ is leaved term of v_i in computation. Therefore, if leaved term in computation is more than really participated term in computation, then credibility of v_i is relatively the less. Also, the higher participation degree of computation have the higher credibility.

```
var
    n := theNumberOfWorkers; //the number of workers
    $\theta$ := credibilityThreshold; //credibility threshold
    $v_i$ := worker; //ith worker
    $g_{id}$ := groupID; //ID of group
    $C_{v_i}$ := credibilityOfWorker; //credibility of worker $v_i$

OrganizeGroup()
    for i := 0 to n do
        $C_{v_i}$ := ClassifyWorkersByCredibility();
        $g_{id}$ := organizeCBWT($C_{v_i}$);
    endfor
```

Fig. 3. Group Organization Algorithm

3.3 Credibility-Based Group Organization Scheme

The CBGOS organize group by credibility of a worker. Before assign a work object, it complete work tree as group organization algorithm with information of the workers as calculated credibility value C_{v_i} by spot-checking. If the credibility of group is more higher than the least threshold when it organize group, then it is place at the highest level. And, workers update work tree by CBGOS whenever they return result. As **fig. 2**, the credibility-based work tree determine that it assign to which worker selected next work object in cycle linked list of work pool. A state of worker in the work tree have idle(i) which is computation available state, busy(b) which is computation unavailable state and die(d) which is stop failure state.

After organized credibility-based work group when assign computation to each group, scheduler assign identifier to know which work object assigned to which worker and organized in which group. The elements of work object is as follows.

$$WO(v_{id}, w_{id}, g_{id}) \tag{4}$$

A v_{id} is the identifier of a worker and a w_{id} is the identifier of a work object and a g_{id} is the identifier of a group. Also, these identifiers are divided into CBWT.

3.4 Group-Based Scheduling Scheme

A work scheduling of this paper apply basically eager scheduling scheme with added GBSS. We firstly execute spot-checking in order to calculate credibility of a worker before allocate a work object. After then, we organize credibility-based group. As **fig. 4**, an allocation server of task allocates a task W_i to a worker through GBSS. If allocated worker's credibility is more less than credibility threshold, then it execute majority voting. Executed result R_i by two workers confirm result and return to task allocation server if scheduler give good result through CBGOS. In this time, GBSS allocate a same work object to a worker of group to more upper level through CBWT. When it execute scheduling, we have to consider state of work object as follows. A work object of work pool have three

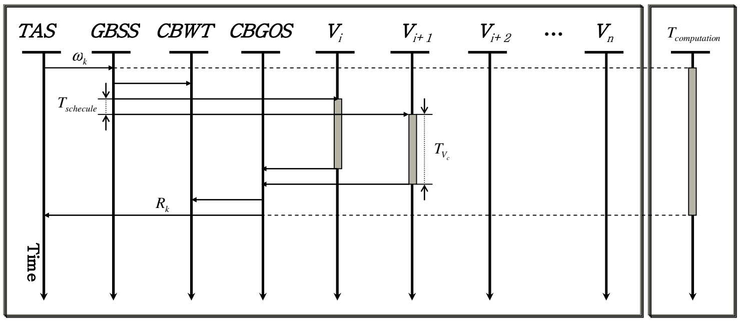
TAS
GBSS
CBWT
CBGOS
V_i
V_{i+1}
V_{i+2}
V_n
$T_{computation}$
ω_k
$T_{schecule}$
T_{V_c}
R_k
Time
TAS : Task Allocation Server
GBSS : Group-Based Scheduling Scheme
CBWT : Credibility-Based Work Tree
CBGOS : Credibility-Based Group Organization Scheme
V : Worker
$T_{computation}$:Total Computation Time

Fig. 4. Scheduling Sequence Diagram through GBSS

```
var
  wo := WorkObject;
  wp := WorkPool;
  wt := CBWT;
  $w_{id}$ := IDofWorkObject;
  $v_{id}$ := IDofWorker;
  $g_{id}$ := IDofGroup;

while wp.nextWorkPoolObject() != null &
     wp.nextWorkObjectState == "undone" do
   wo := wp.nextWorkObject();
   if initial computation then
      allocateWorkObject(wo);
   else
      for wt.everyCredibilityGroup do
         if( wt.$g_{id}$ > wo.$g_{id}$ )
            $v_{id}$ := wt.searchWorker("idle");
         endif
      endfor
      wo.$g_{id}$ := wt.getGroupID($v_{id}$);
      allocateWorkObject(wo);
   endif
endwhile
```

Fig. 5. GBSS Algorithm

states such as "done", "undone" and "doing". The "done" state means return correct result after executed by worker. The "undone" state means worked state or work state before execution. Also, the "doing" state means working state by a worker. In **fig. 5**, we present a GBSS algorithm.

4 Implementation and Performance Evaluation

We measure performance of added component to global computing system, "Korea@home"[14]. This system is able to data centric distributed computing system because of consisted of huge amount of data in set of single instruction.

Type	The Number of Workers	Rate
Xeon	50	1.20%
P4 2.0G over	1,145	27.37%
P4 2.0G under	602	14.39%
P3	1,067	25.48%
P2 under	211	5.05%
Intel Celeron	148	3.54%
Intel Mobile	73	1.75%
AMD Athlon XP	296	7.07%
AMD Athlon	138	3.30%
AMD Duron	38	0.91%
AMD	32	0.76%
AMD Mobile	4	0.10%
Etc	2	0.05%
Not Aware	378	9.03%
Total	4,184	100%

Fig. 6. Organizing Distribution as CPU type

Execution between each data have to independent relation to mutual exclusive execution.

In Korea@home, applications such as "fundamental research to protein folding" and "fundamental research to discover a new medicine" are executed during from November 2003 to February 2004 by participation of 1,845 users and 4,184 workers. In **fig. 6**, workers of Korea@home have various CPU types. We present statistical information of workers's CPU type as follows.

– **Organized Distribution as CPU type**. The workers of 27% for all have Pentium4 2.0Ghz and workers of about 25% have performance of Pentium3. The Pentium4 and Xeon type have relatively high performance as about 43%. Also, AMD CPU has about 12%.

4.1 Performance Evaluation of Group-Based Scheduling Scheme

Group-based scheduling scheme for result checking proposed in this paper presents more low computation delay than previous scheme. First, this scheme to result checking guarantee the reliability for a result as consist of group by the credibility of a worker and then assigns a work object to the organized group. Therefore, this scheme shows more low computation delay than previous scheme. Second, We measured computation delay. As shows in **fig. 4**, we can be reduced computation delay as execute fault-tolerant scheduling in the same time.

– **Guarantee of correctness for result**. We could verify to guarantee correctness for result by our proposed GBSS. First of all, we measure credibility of worker and then apply scheduling scheme by measured credibility of workers. Therefore, we describe equation of proposed scheme in this paper as follows.

$$R_T = \prod_{i=1}^{n} R_{g_i} \tag{5}$$

$$Also, R_g = 1 - \prod_{j=1}^{m}(1 - R_i) \tag{6}$$

In **equation 5**, R_{g_i} denoted reliability for group, R_T denoted reliability for all system. Also, in **equation 6**, R_i denoted reliability for a worker. In previous scheme, it is dependent of reliability of each workers to guarantee correctness, but as **equation 5**, we scheme can shows reliability for all computation by reliability of group R_g. Therefore, we does show reliability for all computation of group by reliability of group verify higher than previous scheme. So, probability to voting normal workers is presented by the "***Bernoulli Trials***". Therefore, we verify group-based scheduling scheme have voting probability worker of more higher credibility than previous scheme.

$$P_x = \binom{n}{s} \left| \frac{F}{N - \sum_{j=k}^{n} N_G} \right|^{s} \left| 1 - \frac{F}{N - \sum_{j=k}^{n} N_G} \right|^{N-S} \tag{7}$$

In **equation 7**, P_x is probability to normal worker and n denoted n trials, sequence of independent repetitions, and s denoted s successes and F denoted the number of normal worker and N denoted the number of all workers. In **equation 7**, N_{G_j} denoted the number of workers of jth group and F is probability of voted normal worker in rest workers of N minus other groups. We are compared with two equation. In previous scheme, it present the number of assigned worker to prior to executed worker among total workers, but in proposed scheme, it has probability the higher correctness because of voted worker in more upper group than itself by GBSS.

- **Measurement of CPU usable rate applied group-based scheduling scheme.** We measure CPU usable rate applied group-based scheduling scheme. **fig. 7** shows experimental results we plot ranking of CPU usable rate of participated worker's PCs during a month period, from SEP. 2004 to OCT. 2004 at Korea@home.

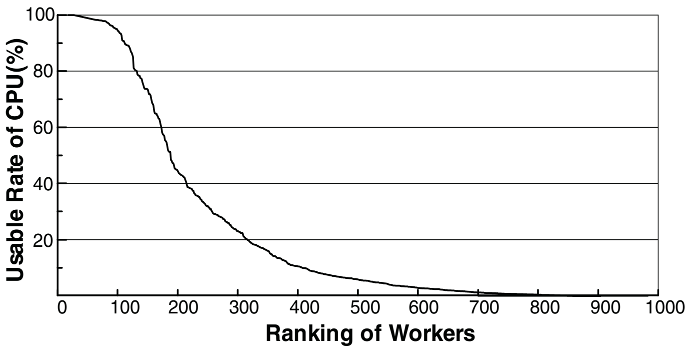

Fig. 7. CPU Usable Rate of Workers

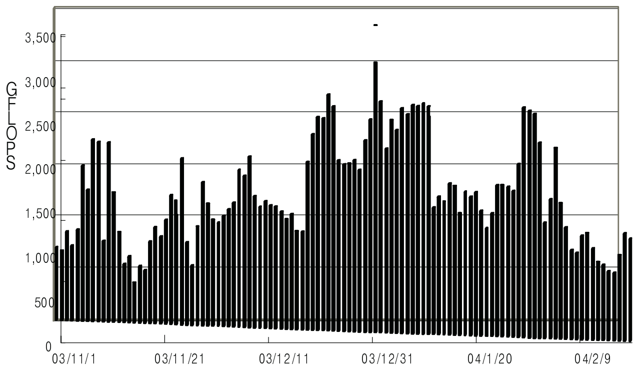

Fig. 8. Performance of Korea@Home measured during about three month

The usable rate of CPU means rate of turnaround time, computation time of all at worker's PC. In here, turnaround time of worker's PC presents normally executing time of worker's PC. First of all, we are shown as follows when we are not apply GBSS. Generally, while measured term of CPU usable rate, worker's PCs of 958 executed task from server and shown CPU usable rate of average 9.68%. But, when we are apply group-based scheduling scheme, worker's PCs of 1598 executed task from server and shown CPU usable rate of average 38.19%. We can know discarded results in previous work, and also we can know high usable rate by reduction of discarded results by GBSS. Therefore, we can getting high performance by GBSS.

- **Performance Evaluation by applied GBSS**. We measured performance by apply GBSS in Korea@home. **fig. 8** shows experimental results it shows measured result during about three month period.

The workers participated average 412(max 942) while a day. We can get 3,185 GFlops more 13% than before and performance of average 1,457 GFlops. In this way, GBSS can be guaranteed the higher correctness.

5 Conclusion and Future Work

In this paper, we proposed GBSS by credibility-based group organization in global computing system. This scheme apply scheduling scheme through organization of group as participated degree in computation and credibility of worker. We can be guaranteed correctness for executed result by workers and also reduced computation delay as result checking.

In near future, we should research for advanced scheduling scheme and dependency model between work objects to implement in Korea@home. Also, we should research for the least turnaround time to the fast response time by scheduling scheme using availability of workers.

References

1. M. O. Neary, P. Cappello. "Advanced Eager Scheduling for Java Based Adaptively Parallel Computing," JGI'02, November 3-5, 2002.
2. D. Molnar. "The SETI@home Problem".
3. L. Sarmenta. "Sabotage-Tolerance Mechanism for Volunteer Computing Systems," FGCS, 18(4). 2002.
4. C. Germain, N. Playez. "Result Checking in Global Computing Systems," ICS'03, June 23-26, 2003.
5. R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou. "Cilk: An Efficient Multithreaded Runtime System," In 5th ACM SIAPLAN Symposium on Principles and Practice of Parallel programming (PPOPP '95), pages 207-216, Santa Barbara, CA, July 1995.
6. A. Baratloo, M. Karaul, Z. Kedem, and P. Wyckoff. "Charlotte: Metacomputing on the Web," In Proceeding 9th International Conference on Parallel and Distributed Computing Systems, 1996.
7. L. LAMPORT, R. SHOSTAK, M. PEASE. "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982.
8. Entropia home page. http://www.entropia.com.
9. A. L. Beberg, J. Lawson, D. McNett, distributed.net home page. http://www.distributed.net.
10. M. O. Neary, A. Phipps, S. Richman. "Javelin 2.0: Java-Based Parallel Computing on the Internet". In Proceedings of Euro-Par 2000. Munich, GERMANY, August 28 - September 1, 2000.
11. N. Camiel, S. London, N. Nisan, and O. Regev. "The POPCORN Project: Distributed Computation over the Internet in Java," In 6th International World Wide Web Conference, Apr. 1997.
12. C. Germain et al, "Global Computing Systems," In Proceeding of SciCom01. LNCS 2179. Springer, 2001.
13. L. F. Lau, A. L. Ananda, G. Tan, W. F. Wong, "GUCHA: Internet-based Parallel Computing using Java," ICA3PP, pp. 397-408. December 2000.
14. Korea@home home page. http://www.koreaathome.com.