Sindh Univ. Res. Jour. (Sci. Ser.) Vol.44 (3) 501-506 (2012)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

A Novel Fault Tolerant Volunteer Selection Mechanism For Volunteer Computing

M. K. KHAN, I. HYDER, B.S. CHOWDHRY*

PAF–Karachi Institute of Economics and Technology, Karachi, Pakistan

Received 18th June 2012 and Revised 26st August 2012

Abstract: Volunteer Computing (VC) is an exciting discipline that has allowed the use of millions of volunteers connected through internet to participate in any project by donating their processing cycles and memory. VC systems have lot of potential but also possess few challenges such as volunteers are not managed centrally, they can submit erroneous results (intentionally or unintentionally) or can quit at any time even while executing assigned tasks. The simple solution is to select volunteer carefully keeping in view the past performance and other measures. In this paper, we have proposed a fault tolerant framework for selecting the best available volunteer. The proposed framework consists of a volunteer grouping and selection mechanism based on the collective impact of processing cycles, memory, spot checking and makespan statistics. To achieve fault tolerance we have also proposed a dynamic replication mechanism that generates the number of replicas of a particular task by analyzing volunteer's behavior.

Keywords- *Volunteer Selection Mechanism, Volunteer Computing, Statistical Mechanism*

1. INTRODUCTION

Volunteer Computing (VC) is an exciting discipline that has established a cheaper path towards high end computing. The million of computers connected on internet can share their idle processing power and memory by becoming part of this community. Another variation of VC is desktop grids (DG) in which hundreds of computers of an organization can be used when their machines are idle or under low load.

The infrastructure of volunteer computing system is based on three entities:

- Master is responsible for maintaining the list of volunteers. Assignment of tasks to the volunteers, getting back the results and communicating it back to user is also master's job.
- Volunteer is responsible to register with master to show the intent of volunteering processing cycles and memory. Its job is to complete the assigned task and send the results to master.
- User is responsible to submit jobs to master which will later sent to volunteers.

There are so many successful projects going on using volunteer computing worldwide such as SETI@home (*Seti 2012*), Folding@home (*Folding 2012*), FightAids@home (*Fight-Aids 2012*), Computing Against Cancer (*Cancer 2012*), Einstein@home (*Einstien 2012*) which shows that the concept started with PARC Worm (*Hupp et al. 1982*) has find its way to the lime light.

Volunteer Computing discipline also enjoy few challenges such as computation time analysis, scheduling, resource management, results verification, fault tolerance, connectivity and bandwidth issues etc. The computers working as volunteers in a volunteer computing environment are inherently volatile, they are heterogeneous, have variations in processing speed and communication bandwidth. It is a fact that these nodes may fail at any time that raises many design and implementation issues.

2. MATERIALS AND METHODS

The most important issue in the volunteer computing is to select the right volunteer or in other word schedule and assign job to such a volunteer that is expected to complete the job and return the results. Numerous mechanisms have been proposed for volunteer selection that can be broadly categorized in the following:

- **Fixed Threshold**: Volunteers are selected on the basis of either a particular processing speed or available memory. Those volunteers that cannot meet the requirements are rejected
- **Makespan Prediction**: Volunteers are selected on the basis of the past performance of completing the assigned job in time.
- **Spot Checking**: Volunteers are selected on the basis of providing correct result against the test job.

All these mechanisms help in selecting the right volunteer. The cost of assigning a job to a wrong

++Corresponding Author: B.S. CHOWDHRY c.bhawani@ieee.org Cell .No.+92-03342639078
*Mehran University of Engineering and Technology, Jamshoro, Pakistan

volunteer is very high in terms of the increase of tasks makespan that results in the dissatisfaction of the job submitter; hence decreasing the credibility of volunteer computing as a discipline.

As volunteers can behave unexpectedly, there is a chance that whatever mechanism we may use for volunteer selection, it may fail. The volunteer having the best resources in terms of processing power, available memory etc, or the volunteer always completing the task on time or passes the spot checking can stops working at any point in time. This is the inherent unreliability of the volunteer computing environment and to counter this problem, Checkpoint and Replication are used. In checkpoint, regular snapshots are taken of the program execution so that when the execution stops due to node failure it can be restarted / resumed later either on the same node or elsewhere. In replication, the task is assigned to more than one node that will not only ensure fault tolerance but will also help in the verification of the results. If more than two nodes submit different results than the submission from the majority or the submission of identical result by N nodes in a given amount of time (*Boinc 2012, Anderson et al. 2004)* will be taken as correct. Credibility-based voting (*Sermenta et al. 2002)* is another mechanism in which by using a spot job, it is ascertain whether the worker is submitting the correct results or not. In spot checking, master sends a test job (results known to master) to workers to verify the workers.

3. RELATED WORK

Various volunteer selection mechanisms have been proposed in effort to improve upon this critical area. In (*Eric Martic et al. 2009*) a scheduling algorithm is presented for medium grained tasks in volunteer computing system environment having unreliable workers. The proposed mechanism uses pull-style task distribution. Two subsequent algorithms for perfectly reliable workers and computation-reliable workers are also proposed.

In (*Young et al. 2010*) two algorithms RMAX and RMXMN are used to estimate make span and robust scheduling heuristics for dynamic volunteer computing systems. The algorithms use proactive reallocation scheme to achieve better results. In (*Derrick Kondo et al. 2007*) heuristics are proposed based on three scheduling techniques, namely resource prioritization, resource exclusion, and task replication. It is reported that the performance of the best resulting method (clock rate prioritization, adaptive resource exclusion based on make span prediction, reactive task replication based on time-outs) is significantly better than FCFS.

In (*Trilce et al. 2008*) the methodology is based on a distributed, evolutionary algorithm capable of searching a large space of scheduling policies looking for policies that outperform manually-designed scheduling policies. A large set of possible scheduling policies are considered. Each scheduling policy is composed of a variable number of IF-THEN-ELSE rules driving the work assignment. BNF (Backus-Naur Form) grammar is used to model the structure of the rules. The proposed method produces scheduling policies that increase throughput across a variety of different VC projects, in contrast to manually-designed policies that are limited to increasing throughput for single projects.

In (*Li Gaol et al. 2007*) a distributed system for the computation of a massive number of unreliable workers for processing dependent tasks is proposed. Any incorrectly computed task corrupts all dependent tasks. The goal is to determine which tasks should be computed by the (reliable) server and which by the (unreliable) workers, and when, so as to maximize the expected number of correct results, under a constraint on the computation time. Optimal scheduling solutions for the mesh are used with the tightest deadline.

In (*Stefan et al. 2008*) an investigation of the allocation patterns for three different resource allocation mechanisms and their subsequent effect on resource utilization within a simulated Grid system is presented. Typically resources are allocated to volunteers on a pull basis, with blocks of tasks being allocated on request, and collected when the next set of tasks is requested. Where as in Economic resource allocation the negotiation of resources is done via an economic mechanism. The main advantage of auctions for distributed and Grid computing is that they are naturally decentralized which permits many auctioneers to function in a distributed network. Reverse first priced sealed auction (RFPSA) is proposed which is an economic based resource allocation.

In (*Sungjin et al. 2006*) Mobile Agent based Adaptive Group Scheduling Mechanism (MAAGSM) is presented that constructs volunteer groups to perform a scheduling mechanism according to the properties of volunteers such as volunteer autonomy failures, volunteer availability and volunteering service time.

The mechanisms available in the literature use one or two performance measures for selecting the volunteers. To make this decision fully-informed and knowledgeable, we need to add more performance measures and that's what we have done in our proposed framework. Further fault tolerance is not considered by many which should be an integral part of any such framework.

4. PROPOSED FRAMEWORK

The proposed framework adds more knowledge in the volunteer selection decision and consists of the following mechanisms:

1- Volunteer grouping mechanism which is based on three different measures that include collective impact of processing cycles and memory, spot checking and makespan statistics.

2- Replication mechanism that suggest the count of the replicas to be generated of a particular tasks by evaluating the volunteer behavior on group wise basis.

Volunteers are grouped using three measures independently that include:

- Collective Impact of Resource (CI)

Analyzing the impact of processing speed, processor availability, harddisk span and available RAM for job execution.

- Spot Check Result (SC) Checking the volunteer against a test job (result already known to server) to verify the results
- Makespan Statistics (MS) How long the volunteer take to complete the assigned job.

CI categorize volunteers in two groups (Accept or Reject) where as the other two measures will categorize the available volunteer in three groups i.e. Platinum, Golden and Silver. The final placement of volunteer on any of the list is based upon the fact that it has got acceptance from CI. If a volunteer is rejected by CI than it will not be considered for the other two measures i.e. SC or MS. It is quite possible that a volunteer gets a place in Platinum list according to one measure, whereas the other measure put the same volunteer in Silver list. This is due to the fact that it is not necessary that a volunteer getting through the spot check also have good makespan statistics. Or the volunteer with better resources would always pass the spot check.

As compared to (*Sermenta et al. 2002)* our mechanism is different in two ways; first we are improving the credibility based voting which is based on spot checking by adding two more measures i.e. impact of collective impact and makespan statistics. Second the degree of redundancy is better controlled in our framework because of volunteers grouping i.e. the probability of replicas required for a job which is assigned to platinum list members would be much less as compare to silver list members.

Now we will explain the volunteer grouping mechanism i.e. collective impact of resources and replication mechanism in detail

4.1. Collective Impact Of Resources

When the volunteers get registered with a server and show the intent of volunteering for a particular task, it is first evaluated against the provided resources that include processing cycles and available memory. Volunteers having resources below than a particular threshold are assigned the task for the evaluation. This mechanism is called "exclusion on fixed threshold" and is a commonly used mechanism but it does not consider the established fact that the RAM and others indicators also play an important part in speeding up the processing. Now we will present the step by step working of this mechanism.

4.1.1. Service Request

Job (task) is submitted by the users to the master (server) who maintains the list of all those workers (volunteer) who are volunteering their services. To analyze the workers volunteered resources at the time when they are joining in, master is suppose to poll the systems parameters from them but as the worker has the liberty to change the volunteered resources at any time, the polling is required to be repeated often.

4.1.2. Service Assessment

Following function is used by the master to assess the expected performance of the available workers.

$$R_{ij} = \begin{cases} 1, & \text{if } Rij > dj \\ 0, & \text{if } Rij < dj \end{cases}$$

In the above equation, R_{ij} R_{ij} represents the $j_{th} i_{th}$ i_{th} worker having j_{th} parameter, the threshold value for the particular parameter is represented by dj. Master can set the minimum threshold of every parameter by analyzing user's requirements. If all the parameters on the worker are above the minimum level set by the master than R_{ij} = 1. Following parameters will be used:

Parameter Title	Parameter Representation
CPU – Speed	X1
CPU – Availability	X2
Storage – Capacity	X3
RAM – Availability	X4

4.1.3. Scoring Scheme

A simple scoring scheme has been used to evaluate the above mentioned parameters. The ranking in this scoring scheme improves with the increase in the donated resources. For example if the available CPU on a worker is 3.0 GHz than this service is ranked 4. Likewise other parameters will be treated in the similar fashion. Following tables declare the scoring scheme of all the parameters.

CPU – Speed (X1) Scoring Scheme	Value
X1 is less than 1.8 GHz	1

X1 is greater than 2.8 GHz	4

CPU – Availability (X2) Scoring Scheme	Value
X2 is less than 25%	1
X2 is in between 26% to 50%	2
X2 is in between 51% to 75%	3
X2 is greater than 75%	4

Storage – Capacity (X3) Scoring Scheme	Value
X3 is less than 500 MB	1
X3 is in between 501 MB to 1000 MB	2
X3 is in between 1001 MB to 1500 MB	3
X3 is greater than 1500 MB	4

RAM – Availability (X4) Scoring Scheme	Value
X4 is less than 256 MB	1
X4 is in between 257 MB to 512 MB	2
X4 is in between 513 MB to 1 GB	3
X4 is greater than 1 GB	4

The flexibility presented in this method is quite obvious. The threshold value of the parameters can be adjusted in two ways. First the user who is submitting the job can set the threshold value as per the need, second the master can adjust the parameters by looking at the over performance of the volunteers. Let us assume that the user is not novice and ask the master to send the task to only those volunteers who have a CPU speed of greater than 2.8 GHz. The user also expect that the task should only be given to a worker whose 70% CPU is available. Now the master can decide to adjust the requested parameters by analyzing the overall conditions of the volunteered resources such as availability of RAM etc.

4.1.4. Criteria of the Worker's Selection

Once the threshold test is applied, we will get the result in the form of logical values for each parameter. These logical values will lead to the correct prediction of the acceptability level of the workers. Scoring scheme and threshold test are used to obtain these predicted values, also called (Y values). as of now, we have fixed the passing marks of the threshold test at 75% that practically means that if 3 parameters out of 4 are satisfactory than the worker is put on a pre-selection list. The members of this list are than spot checked for the final selection.

4.1.5. Logistic Regression Model and Its Implementation

Modeling of continuous-value functions is usually done by linear regression model whereas for categorical response variables, generalized regression models are used. One of them is known as Logistic Regression Model that estimates the probability p that the dependent variable will have on a given value. One limitation of above mentioned model is that it could be used when the output variable can only hold binary data i.e. 0 and 1.

either accepting the workers or rejecting them. This is also one of the reasons for selecting Logistic Regression Model for our analysis. Let us assume that output Y is coded as 0 and 1 as two possible categorical values. We can compute the probabilities for both values that will be based on the available data (four parameters). Here is the given input sample:

$P(yj = 0) = 1 - pj$ and $P(yj = 1) = pj$.

We will use linear regression model on which we will fit these probabilities:

$$\log\left(\frac{p_j}{1-p_j}\right) = \alpha + \beta_1.X_{1j} + \beta_2.X_{2j} + \beta_3.X_{3j} + \cdots + \beta_n.X_{nj}$$

This above equation is called the linear logistic model. The function log (pj(1 - pj)) can also be written as logit(p). There is a chance that the probabilities that we will be predicting in the form of Pj can go out of range whereas the range is (0,1). Logit form of output is precisely used to take care of the above mentioned.

If the final value for probability p is below 0.5 that it can be said that output value Y = 0 is more probable than the other categorical value Y = 1. It is possible to establish the logistic regression model by using one set of data (training set) but to analyze whether the model is correctly predicting categorical values, we need to another set of data (testing set). (*Kantardzic et al. 2003*)

Coming back to our solution, now we need to estimate the Logistic Regression Model's coefficients values. First we need to poll all the parameters from the workers and then use SPSS software. SPSS uses given data set to produce the required output in the form of estimation of the coefficients i.e. $\beta_1, \beta_2, \beta_3 and\ \beta_4$ and constant values i.e. β_0 as shown in **(Table 1).**

Table 1 - Parameters, Coefficient, Values and Constant Table

Parameters	Variable	Coefficient	Values
CPU - Speed	X1	β4	14.010
CPU-Availability	X2	β3	.086
Storage-Capacity	X3	β2	.050
RAM-Availability	X4	β1	.048
Constant		βo	-53.133

It is stated earlier as well that the Logistic Regressions Model equation is:

$$Y = \beta o + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4$$

We get our own model, by replacing the values of table 1 in the above equation.

$$Y = -53.133 + 14.010\ X1 + 0.86\ X2 + 0.050\ X3 + 0.048\ X4$$

Now we can test our model as we have the values of coefficients. We just need to put the values of X1, X2

4.1.6. Scheduling Results

According to classification table **(Table 2)** generated by the SPSS on the basis of training data set of 326 volunteers, we got 93 % correct prediction results. It is also evident that the model has erroneously selected 14 such volunteers that should be rejected and rejected 8 such volunteers that should be selected.

Table 2- Result Classification Table

Observed Y	Predicted Y		Correctness Percentage
	0	1	
0	72	14	100 - (14/326x100) = 95.7
1	8	232	100 - (08/326x100) = 97.5
Overall Percentage			100 - (22/326x100) = 93.3

The mechanism either accepts or rejects the volunteers. The selected volunteers are than distributed among Platinum, Golden and Silver groups by using the other two measures i.e. spot checking and makespan statistics.

4.2. Replication Mechanism

As stated earlier, in volunteer computing replication represents the assignment of one job to more than one volunteer to achieve fault tolerance. This addresses two issues, volunteers submitting erroneous results intentionally or un-intentionally can be identified and the impact of volunteers becomes unavailable while executing the task can be reduced. In first some sort of results are produced but their verification is required whereas in later no results are produced. The crucial impact is similar in both cases i.e. the increase of the task's makespan.

Replication also has a down side as it is duplication of effort that may result in waste of precious processing cycles. So the important question is how many replicas to be created of a particular task? This is dependent on the behavior of volunteers in terms of submitting erroneous results or to become unavailable while executing the task. These volunteer properties would vary in different systems.

In the proposed replication mechanism, we have tried to keep the number of replicas to the minimum. The numbers of replicas can be adjusted by analyzing the behavior of the volunteers of a particular system. Poor volunteer behavior would impact in a negative fashion and will increase the number of replica's to be generated for a particular task. Our replication mechanism is dynamic because of the aforementioned. For the sake of simplicity, two-state-discrete-time markov chain (DTMC) is used to model the system. We assume that the volunteer in volunteer computing environment toggles between good and bad state.

- Good State: Volunteer is available, completes the execution and returns the result
- Bad State: Volunteer is not available, doesn't complete the execution or doesn't submit the result
- Current State: Before getting the task for execution

Future State: After getting the task for execution

The transition of two states can be shown as:

Current State	Future State
Good	Good
Good	Bad
Bad	Bad
Bad	Good

We considered 0 for the successful execution of task and 1 for the failed execution of task. The transition matrix T is given as:

$$\mathbf{T} = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \quad (1)$$

In the above matrix, we are only considering the absolute probability of bad state at any given time. We need to get a product of initial vector and the transition matrix of that step to get the Absolute Probability. This can be represented as:

$$\mathrm{P}^n = \mathrm{P}^{(0)}\mathrm{T}^n \quad (2)$$

In volunteer computing, volunteer's probability of moving to a bad state can vary. For the sake of calculation and simplicity, we are taking it as 20%[1] but this percentage can be adjusted any time by getting the actual input of the system. Total probability must be 1, if $\mathrm{P}_i^{(0)}$ is the initial probability and can be given as:

$$\sum_{i=1}^{m} p^{(0)} = 1 \quad (3)$$

The absolute probability after considering m number of finite states will be:

$$\mathrm{P}_j^{(n)} = \mathrm{P}^{(0)}\mathrm{T}^{(n)} \quad (4)$$

Now, we have the absolute probability vector and we need to dig the bad state probability out of it which will be used for finding the quantity of Replicas required to be generated of a particular task at a particular time. Binomial distribution is used to fix **n** (the maximum number of Replicas, a system should send). The absolute probability of bad states is compared in the cumulative probability column of **(Table 3)** to choose the closest value. The required output is the **NoR** (Number of Replicas) of the respective row possessing the closest value of the bad state absolute probability. Now we list the sources of input for binomial distribution:

[1] Different percentages can be used for the bad state as per the requirements of actual conditions

$x \rightarrow$ It ranges from 0 to n and is a random variable

$n \rightarrow$ It is the fixed number of trial (attempts) defined in the system

$p \rightarrow$ It is the good state probability computed from initial vector

$NoR \rightarrow$ Represent the number of Replicas to be generated for the successful delivery of the task to such volunteer that will complete it and will send the results.

Table 3. NoR calculation table

NoR	Probability b(x; n, p) $P(NoR \le x) = \sum_{NoK=0}^{x} \binom{n}{x} p^x q^{n-x}$	Cumulative Probability
0	0.0003	0.0003
1	0.0064	0.0067
2	0.0512	0.0579
3	0.2048	0.2627
4	0.4096	0.6723
5	0.3277	1.0000

4.2.1. Replication Results

This mechanism suggests how many replicas of a particular task are to be generated in a volunteer computing environment as per volunteer's behavior. Now the suggested replicas are sent to various volunteers and verification of the results will automatically be handled.

For example, the system is facing tough conditions that resulted in the increase of unavailable state probability to 0.39. The corresponding NoR of the closest value 0.6723 is 4 (Table 3). The results from the NoR table can be interpreted as 4 Replicas of a particular task should be generated and sent to volunteers to overcome the unavailable state probability of 0.39. That is how the behavior of volunteers is mapped on the probable quantity of Replicas to be generated for the successful execution of the task and to overcome the impact of volunteers having bad state.

5. RESULTS DISCUSSION AND CONCLUSION

In this paper we have proposed a novel volunteer selection framework consisting of volunteer grouping and selection mechanism. The proposed framework is fault tolerant as a dynamic replication mechanism is also coupled with it. Volunteers selected from "Collective Impact of Resources" are than spot checked and their makespan statistics are evaluated. Based on the above mentioned, volunteers are divided in three groups i.e. Platinum, Golden and Silver. The replication mechanism is dynamic and robust as it suggests the number of replicas to be generated of any particular task according to the overall volunteer behavior.

Scheduling mechanism results have shown that out of 326, the job is wrongly scheduled to only 22 states the required number of job replicas as per need.

These days we are incorporating the said framework in BOINC which is one of the leading middleware for volunteer computing worldwide.

REFERENCES:

Anderson, D.P. (2004) "BOINC: a system for public-resource computing and storage". In: 5th IEEE/ACM International Workshop on Grid Computing, 2004,4–10.

BOINC: http://boinc.berkeley.edu/ accessed in 2012

CANCER: www.computeagainstcancer.org/ accessed in 2012

Derrick K., A. A. Chien, and H. Casanova (2007) "Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids". J Grid Computing **(5)**: 379–405

Einstein@home: www.einstein.phys.uwm.edu/ accessed in 2012

Eric M., H. David P. Anderson, and K. Hagihara (2009) "Computing Low Latency Batches with Unreliable Workers in Volunteer Computing Environments". J Grid Computing **(7):** 501–518

FightAids@home: www.fightaidsathome.org/ accessed in 2012

Folding@home: www.folding.stanford.edu/ accessed in 2012

Hupp, S. (1982) "The Worm programs – early experience with distributed computation". ACM.

Kantardzic. M. (2003) "Chapter 5. In Data Mining: Concepts, Models, Methods, and Algorithms". John Wiley and Sons.

Li G., and G. Malewicz: (2007) "Toward Maximizing Quality Results of Dept. Tasks Computed Unreliably". Theory of Computing Systems **(41):** 731–752.

Sarmenta, L.F.G. (2002) "Sabotage-tolerance mechanisms for volunteer computing systems". Future Gener. Comput. Syst. 18 **(4):** 561–572

SETI@home:www.setiathome.ssl.berkeley.edu/ accessed in 2012

Stefan K. (2008) Kris Bubendorfer: "Grid Resource Allocation: Allocation Mechanisms and Utilization Patterns". The Sixth Australasian Symposium on Grid Computing and e-Research.

SungJin C., M. Baik, J. Gil, S. Y. Jung, and C. Hwang (2006) "Adaptive group scheduling mechanism using mobile agents in peer-to-peer grid computing environment" Apple Intel **(25):**199–221.

Trilce E., O. Fuentes, and M. Taufer: (2008) "A Distributed Evolutionary Method to Design Scheduling Policies for Volunteer Computing". ACM.

Young C. L. (2010) Albert Y. Zomaya · Howard Jay Siegel: "Robust task scheduling for volunteer computing systems". J Supercomputing **(53)**:163–181.