Signal Processing and Renewable Energy
March 2018, (pp.1-8)
ISSN: 2588-7327
eISSN: 2588-7335

Resource Allocation in Volunteer Computing Networks Based on Node Reliability

Jafar Khakpour[1], **Mir Mohsen Pedram**[1*]

[1] Department of Electrical & Computer Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran.

Abstract
Volunteer Computing is a special distributed computational architecture as composed of a network of volunteer computational units connected to each other and organized to perform some specific tasks. These networks are distributed in large networks or even vast geographical areas. Rather than lower communication capabilities in these networks, resource management capabilities are limited than compared with other types of computational networks. In this research, we tried to model resource allocation, in these networks considering such limitations in these networks. We put node reliability in this the model as a factor of how much we can rely on a node to complete an assigned task on a specified time and assign one task to multiple nodes in parallel to increase probability of completing task. We model this problem as an economic model to decrease costs and increase revenue on the network based on each tasks priority.

Keywords: Volunteer computing, Desktop grids, Resource management, Grid-resource reliability, Distributed computing, Market model.

1. INTRODUCTION

Distributed Computing System (DCS) is a network of connected computational units cooperating each other to perform a computational task. This network provides sharing of distributed processing units that produce, store and process data on the network. This type of computational power system provides higher computational power with lower costs to scientists, businesses and industries. However, DCS has many limitations and difficulties. One of The major problems in the distributed systems are managing resources and balancing processing tasks on processing nodes [1] to better consume the computational power on the network.

Volunteer Computing (VC) networks are type of distributed networks which uses volunteer resources to perform processing tasks. It They benefits from public interest in scientific research and participation in solving specific problems [2].

These networks use volunteer units' idle resources to process or store data. As expected processing task assigned to these units are not their primary tasks and there is a very limited control over resources on this type of units them. These units are not reliable in terms of a guaranteed amount of processing resources or even commitment to complete the assigned task.

2. RESOURCE MANAGEMENT

Accessible resources in a volunteer computing (VC) network are highly heterogeneous and the infrastructures used for communication between

*Corresponding Author's Email: pedram@khu.ac.ir these nodes are restricted with respect to other types of distributed computing systems. Some important characteristics of VC resources which should be considered on resource allocation are [2] [3]:

• Concurrency of Components: VC networks should be able to process multiple tasks on multiple nodes of network. These tasks could can be part of a bigger task executed by different users for different use-cases.

• Openness and unreliability of systems: Commuting nodes can easily join the network, share resources, accept and process tasks and leave the network. Due to autonomy of nodes, systems unreliability increases for all nodes. There is no guarantee if a device completes assigned tasks before leaving the network. VC resources are not dedicated to process system tasks (VC is using these nodes idle processing power), so processing a task could also take longer than expected on these nodes [4].

• Heterogeneous nodes: Node capabilities vary over a big range on these networks. Some devices could have special abilities which are rare in the network and some just able to process special tasks [5]. When tasks are executed, the network should be able to coordinate these nodes in a way to maximize network utility by assigning tasks and nodes in an optimal way.

To achieve a better efficiency, it is desired to consider these characteristics in resource allocation model. There are many methods developed for resource allocation in these networks in the literature. Nouman [2] listed a group of these methods varying from simple methods like First Come First Served (FCFS) [6] and Random Assignment [7] models to different types of round robin [7] and [8], buffering [9] and Hierarchical [10] resource allocation methods. These models are suitable for BOINC [11] like networks. All mentioned models except hierarchical ones are designed to process a task with huge amount of parallel and similar subtasks which doesn't imply a strict dependency between subtasks. Another group of resource allocation models which is not discussed in this survey (which only includes models which are used in BOINC like networks) is economic models. Hierarchical methods which also have been mentioned in [2] are networks with a tree like graph topology. In these networks, every project is in a parent-child relationship. Each project may request a processing power from lower level projects or provide an idle processing power to upper level project [10]. This type of scheduling methods is deployed in many projects including SZTAKI [12], GLOBUS [13] and LEGION [12].

In economic based models, power resources and tasks are modeled as a supply and demand model. Each task owner can demand processing unit's time as a resource. An important study in this group of resource allocation models is Grid Architecture for Computational Economy (GRACE) [14] which organizes these models into eight categories [15]. In this model, task brokers' role is to match requests and demands in a way to maximize revenue on both sides of a deal [16].

In this study, we aim to propose a supply and demand model to include resource reliability of the processing nodes. In order to successfully process a task, each resource needs to meet some minimum requirements. If a node fails to provide this required minimum, we consider this resource as failed. Reliability of a resource is defined as the probability of performing a task in a considered time without any failure [18].

Let T be a random variable showing the occurrence of the first failure in the system, we can define reliability R for a task which needs time t to process as:

$$R(t) = P(T \geq t) = \int_0^t f(T)\,dT \qquad (1)$$

Variable T has a Poisson density function [18]. Because of memory-less characteristic of Poisson distribution, probability of failure happening in a time interval t is independent from its prior time intervals. We also suppose failure of every node in the network is independent from other nodes in the network.

There are two other factors in the node failure management named node maintainability and node availability. Maintainability on a network is defined as probability of a node with failure at timestamp t to join to the network and be ready to process tasks at time $t+1$. For simplicity [19], we suppose that network's maintainability is 100%. Notice that, maintainability of nodes is independent concept from their reliability. To make model simpler we omitted maintainability of nodes by considering this value as 100% for all nodes. We also include node availability factor as part of node reliability [20]. Node availability shows how much is it is probable for a node to be disconnected from network when it's it is scheduled to work on the network. We are going to behave have this factor as part failure/reliability. In other types of distributed systems, this factor's role is different than node reliability, but for VC networks with loosely-managed and large scale networks, there is no difference between failure and unavailability of a node.

Volunteer computing networks are large scale networks with a highly variable size with low reliability nodes (comparing to other grids and distributed computing systems). We should also consider that task failure grows with task size. If a task needs n time slots of nodes with probability p, then probability of a failure in processing the task is np, which makes it difficult for grids to process large tasks. Rollback Recovery and Duplication are two solutions which are mainly used to increase reliability of tasks in a network, which are named [21].

In Rollback Recovery, each node's processing result is backed-up and in case of failure, network is able to restore last checkpoint on another node to continue processing from last checkpoint instead of starting to process from beginning.

In Duplication techniques, a single subtask is assigned to multiple nodes simultaneously, so it is just needed one of the nodes to complete the task successfully in order to avoid processing failure of this subtask. This method has a larger overhead compared to Rollback Recovery methods, but processing power's low cost and high availability in VC networks, encourage to use this type of techniques. Many of famous VC networks use this method for reliability and security reasons [22]. The method is also popular in large scale commercial grids and cloud infrastructures [23].

Most of architectures studied in this research use a combination of both methods to increase reliability and efficiency with lower costs. We also use both of these methods to increase reliability in our model. We suppose that network can assign each subtask to multiple nodes in order to increase successful process of a subtask in determined time interval and keeps each subtask results in another place on the network to be used in case of future failures.

3. MODEL DEFINITION

3.1 Task Graphs

Each task in distributed networks is defined as a Direct Acyclic Graph (DAG) of dependent subtasks. Without loss of generality, we can suppose each tasks DAG has a start and end node (subtask). All subtasks depend on start node and end node depends on all subtasks in the dependency graph. These two subtasks are not counted when assigning each subtask to a network resource. Each subtask has a layer number, which is its path length from start node.

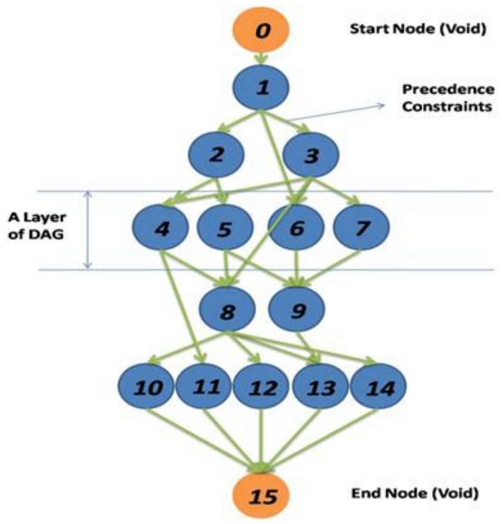

***Fig. 1.** A task's Direct Acyclic Graph with additional start and end nodes.*

3.2 The Proposed Model

Each buyer i pays value R_i if his/her task completes successfully. Each node n has a small cost C_n at each time slot (time is discrete), if device is used to process tasks. Let us suppose x_{ijsn} is a binary variable showing if we have assigned subtask j in task i to node n at time slot s. We name failure rate of node n as f_n which shows the probability of a failure happening for device n at a single time slot. Now we can show probability of successful processing subtask T_{ij} with scheduling matrix $X = \left[x_{ijsn}\right]$ as:

$$\Pr(\text{successfully process}\, T_{ij} \,|X) = 1 - \prod_n f_n^{\,x_{ijsn}} \tag{2}$$

If we define:

$$e^{h_{ijs}} = \prod_n f_n^{\,x_{ijsn}} \tag{3}$$

Then we have:

$$h_{ijs} = \sum_n x_{ijsn} \ln f_n \tag{4}$$

Thus, probability of scheduling X to process task i and gain revenue R_i is:

$$\Pr\left(\text{process task}\, i\, \text{successfully}\right) = \prod_{s,j}\left(1 - \prod_n f_n^{\,x_{ijsn}}\right) = \prod_{s,j}(1 - e^{h_{ijs}}) \tag{5}$$

Cost of all resources scheduled to process task i (and its subtasks T_{ij}) is then:

$$\text{total processing cost of task}\, i = \sum_{j,s,n} x_{ijsn} Q_n \tag{6}$$

where C_n is node $n's$ expenses for a single time slot. Now expected revenue of this task is shown as:

$$\text{task}\, i\, \text{expexted revenue} = R_i \prod_{s,j}(1 - e^{h_{ijs}}) - \sum_{j,s,n} x_{ijsn} C_n \tag{7}$$

which is a concave function on variables h_{ijs} and x_{ijsn} where:

$$0 \le e^{h_{ijs}}, x_{ijsn} \le 1$$

Now we can write our model as:

$$\max F = \sum_i R_i \prod_{s,j}(1 - e^{h_{ijs}}) - \sum_{j,s,n} x_{ijsn} C_n \tag{8}$$

Subject to:

$$\begin{aligned}
&1)\ x_{ijsn} \le Q_{ijn} \\
&2)\ \sum_{i,j} x_{ijsn} \le 1 \qquad \forall s,n \\
&3)\ \sum_{s,n} x_{ijsn} \le M\, a_i \quad \forall i,j \\
&4)\ a_i \le \sum_{s,n} x_{ijsn} \qquad \forall i,j \\
&5)\ \sum_n x_{ijsn} \mathrm{U}_n \le h_{ijs} \\
&6)\ x_{ijsn}, a_i \in \{0,1\} \\
&7)\ h_{ijs} \le 0
\end{aligned} \tag{9}$$

Variables in this model are:

F : Total expected revenue as model's objective function,

x_{ijsn} : Binary variable showing if subtask j of task i is assigned to node n at time slot s ,

a_i : Binary slack variable showing if scheduler is going to process task i or not,

h_{ijs} : Slack variable as logarithm of failure rate of a subtask T_{ij} .

And coefficients are:

R_i : Revenue of successfully processing task i ,

C_n : Costs of using node n to process a task at any time slot,

M : Big-M, a big enough value,

Q_{ijn} : Binary value showing if node n is able to process subtask T_{ij} ,

$U_n = \ln f_n$: Logarithm of failure rate of node n .

As discussed before, objective function is the expected total revenue of network from processing assigned tasks. Constraint 1 of model is added to prevent a task from assigning to a node without minimum required processing power.

Constraint 2 prevents simultaneously assigning multiple subtasks to a single node. Constraint 3 and 4 say one task is either activated or not activated. If it is activated, there should be at least one assignment for each subtask of this task and if it is not activated, no subtask of this task should be processed on any nodes. Constraint 5 is a rewrite of equation (5).

The model tries to maximize a concave function with a set of linear constraints. So, if we relax this model from Mixed-Integer model to a real-valued problem, it would be a concave optimization problem. Thus, many of convex optimization problems can help us on finding good enough results for this problem. In the following part, we will describe an algorithm to find solutions for the proposed model.

3.3. Solving the Problem

We are going to use Benders decomposition [24] to find a solution for the problem. In this technique, a complex problem of the form:

$$\min_{X,Y} f(x,y)$$

subject to:

$$\begin{aligned} &C(x,y) \le 0 \\ &x^L \le x \le x^U \\ &y^L \le y \le y^U \\ &x \in X, y \in Y \end{aligned} \qquad (10)$$

is solved, where x and y are model parameters. Now, if we suppose y is the complicating variable that by removing this variable, our model forms a simple model which we are able to solve. Now we can define function V as:

$$V(y=\hat{y}) = \min\left\{ f(x,\hat{y}) | C(x,\hat{y}) \le 0, x \in X \right\}$$

Then the main problem is rewritten as:

$$\min_{y,\theta} \theta$$

subject to:

$$\begin{aligned} &\theta \ge V(y) \\ &y \in Dom(V) \end{aligned} \qquad (11)$$

which $Dom(V)$ is function $V's$ domain (all y values which $V(y)$ exists for these values). The above problem is solved using the following approximation on objective function and adding benders cuts for each iteration of algorithm to add more cuts and create a better approximation of the main function:

$$\min_{y,\theta} \theta$$

subject to:

$$\begin{aligned} &y \in S_p \\ &S_p = \\ &S_{p-1} \bigcap \left\{ y | \theta \ge V\left(\hat{y}^p\right) + (\nabla V^p)^T \left(y - \hat{y}^p\right)\right\} \end{aligned} \qquad (12)$$

which p is step counter and $S_1 = Dom(V)$. So, first decomposition problem of our model is:

$$\min_{y,\theta} \theta$$

subject to:

$$\begin{aligned} &x_{ijsn} \le Q_{ijn} \\ &\sum\nolimits_{i,j} x_{ijsn} \le 1 \quad \forall s,n \\ &\sum\nolimits_{s,n} x_{ijsn} \le M\, a_i \quad \forall i,j \\ &a_i \le \sum\nolimits_{s,n} x_{ijsn} \; \forall i,j \\ &x_{ijsn}, a_i \in \{0,1\} \\ &\theta \ge \theta^L \end{aligned} \qquad (13)$$

And the second problem would be a problem with obvious solution of:

$$h_{ijs} = \sum\nolimits_n x_{ijsn} U_n \qquad (14)$$

where U_n is logarithm of failure rate of node n ($U_n = \ln f_n$). And at each step p we add this optimality cut constraint to the model:

$$\theta \ge \theta^p + \frac{R_i \prod_{s,j} (1 - e^{h_{ijs}^p}) - C_n}{1 - e^{x_{ijs}^p}} \left(x_{ijsn} - x_{ijsn}^p\right)$$
$$\forall i, j, s, n \qquad (15)$$

which θ^p, h_{ijs}^p and x_{ijs}^p are values we found from step p. It is worth to mention that above constraints are linear inequalities. We repeat these steps until θ converges to a constant value or it get into ε-neighborhood of V^p.

4. EXPERIMENTAL RESULTS

To show the importance of device reliability in our model, we show a small-scale sample and try to solve the optimization model. In this example we have 10 nodes with 3 processing tasks. Each of these tasks has 4 subtasks (other than Start and End nodes of task DAG). Node usage costs are random numbers with normal distribution $N(\mu = 5,\ \sigma^2 = 2)$ and revenue of completing each task is drawn from $N(\mu = 125,\ \sigma^2 = 20)$.

We tried to schedule this problem on two different cases: first when failure rate of each node is a random number between 0.9 and 0.999, and second case, when failure rate is zero (nodes are completely reliable).

For the first model we found a scheduling with total revenue of 255. The algorithm has found the solution in 4 iterations.

For the models without failure rates, we got total revenue of model as 324.3. But when we applied same failure values to the model used for the first case, the revenue decreased from 324.3 to 243.8. This result is because the second case just tries to find the combinations with least cost to assign tasks and there is no replica scheduling in the model. Results here are not so big, but as you can see in Fig. 3, if we increase task size and reliability of successful processing a task without other replicas, decreases drastically.

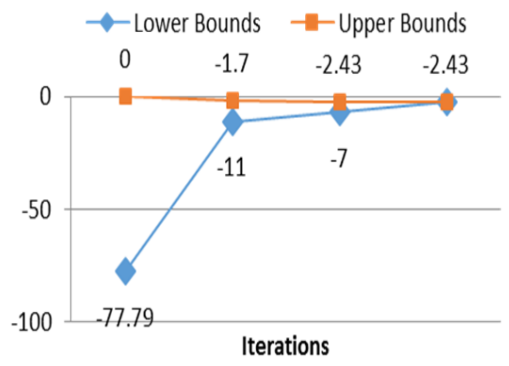

Fig. 2. Benders decomposition value convergences in 4 iterations for the test problem.

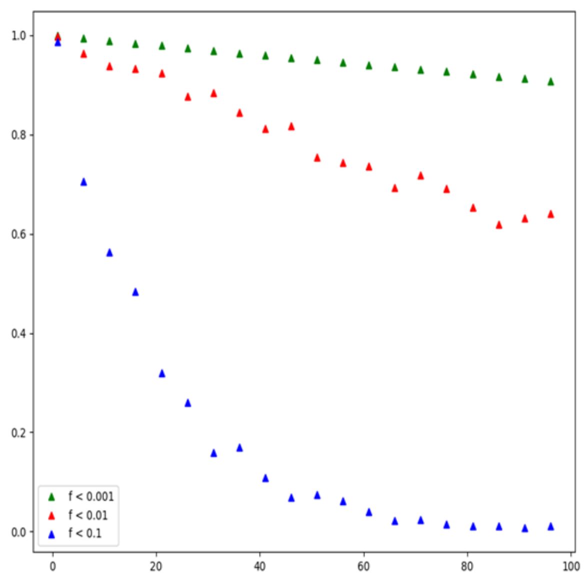

Fig. 3. Reliability of tasks and their reliability regarding to different average failure value of nodes.

5. CONCLUSION

In this research, we tried to model reliability and show how it can change a VC network's expected reliability and revenue. Resource scheduling was supposed as a centralized management system in the study, and converting this model to a market model, let us assign resources in a distributed mechanism using trading agents. GAMS software was used to implement the algorithm and to solve the decomposed problem. Unfortunately, standard free solvers we used are not efficient to solve this problem with large tasks and it fails to find optimum solution for large tasks containing more than 10 subtasks. Thus, metaheuristic algorithms could be a good choice of to solve these problems.

REFERENCES

[1] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 2, pp. 585–599, 2016.

[2] M. N. Durrani and J. A. Shamsi, "Volunteer computing: requirements, challenges, and solutions," J. Netw. Comput. Appl., vol. 39, pp. 369–380, 2014.

[3] G. Coulouris, J. Dollimore, and T. Kindberg,

Distributed Systems: Concepts and Design (5th Edition). Addison-Wesley, 2011.

[4] S. Haghtalabi, R. Javidan, and A. Harounabadi, "A New Resource Allocation Model for Grid Networks based on Bargaining in a Competitive Market," J. Soft Comput. Appl., vol. 2014, no. 1, pp. 1–17, 2014.

[5] Y. Jiang and Z. Huang, "The rich get richer: Preferential attachment in the task allocation of cooperative networked multiagent systems with resource caching," IEEE Trans. Syst. Man Cybern.-Part Syst. Hum., vol. 42, no. 5, pp. 1040–1052, 2012.

[6] T. Estrada, M. Taufer, and D. P. Anderson, "Performance prediction and analysis of BOINC projects: An empirical study with EmBOINC," J. Grid Comput., vol. 7, no. 4, pp. 537–554, 2009.

[7] D. P. Anderson, "Emulating volunteer computing scheduling policies," in Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, 2011, pp. 1839–1846.

[8] S. H. Ngo, M. Fukushi, X. Jiang, and S. Horiguchi, "Efficient scheduling schemes for sabotage-tolerance in volunteer computing systems," in Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on, 2008, pp. 652–658.

[9] D. Toth and D. Finkel, "Improving the productivity of volunteer computing by using the most effective task retrieval policies," J. Grid Comput., vol. 7, no. 4, pp. 519–535, 2009.

[10] A. C. Marosi, "Challenges and formal aspects of volunteer computing," 2016.

[11] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on, 2004, pp. 4–10.

[12] Kacsuk, A. M., "SZTAKI Desktop Grid – a Hierarchical Desktop," in Proceedings of the 3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC–16'), 2016, pp. 183–212.

[13] L. Ferreira et al., "Introduction to grid computing with globus," Ibm Redb., vol. 9, 2003.

[14] R. Buyya, "Economic-based distributed resource management and scheduling for grid computing," ArXiv Prepr. Cs0204048, 2002.

[15] M. B. Qureshi et al., "Survey on grid resource allocation mechanisms," J. Grid Comput., vol. 12, no. 2, pp. 399–441, 2014.

[16] L. Ding, L. Chang, and L. Wang, "Online auction-based resource scheduling in grid computing networks," Int. J. Distrib. Sens. Netw., vol. 12, no. 10, DOI: 10.1177/1550147716673930.

[17] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," Inf. Sci., vol. 357, pp. 201–216, Aug. 2016.

[18] Y.-S. Dai and X.-L. Wang, "Optimal resource allocation on grid systems for maximizing service reliability using a genetic algorithm," Reliab. Eng. Syst. Saf., vol. 91, no. 9, pp. 1071–1082, 2006.

[19] C. Dabrowski, "Reliability in grid computing systems," Concurr. Comput. Pract. Exp., vol. 21, no. 8, pp. 927–959, 2009.

[20] M. R. Lyu and others, "Handbook of software reliability engineering," 1996.

[21] M. Chtepen, F. H. Claeys, B. Dhoedt, F. De Turck, P. Demeester, and P. A. Vanrolleghem, "Adaptive task checkpointing and replication: Toward efficient fault-tolerant grids," IEEE Trans. Parallel Distrib. Syst., vol. 20, no. 2, pp. 180–190, 2009.

[22] K. Budati, J. Sonnek, A. Chandra, and J. Weissman, "Ridge: combining reliability and performance in open grid platforms," in Proceedings of the 16th international symposium on High performance distributed computing, 2007, pp. 55–64.

[23] O. A. Ben-Yehuda, A. Schuster, A. Sharov, M. Silberstein, and A. Iosup, "ExPERT: Pareto-Efficient Task Replication on Grids and a Cloud."

[24] A. M. Geoffrion, "Generalized benders de-

composition," J. Optim. Theory Appl., vol. 10, no. 4, pp. 237–260, 1972.