Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.8, No.8, 2017
www.iiste.org

A Dynamic Task Allocation Algorithm Based on Weighted Velocity

Chuang-Wei Wang * Wen Wang Ying-Ying Cao Ke-Ming Tang
College of Information Engineering, Yancheng Teachers University, Jiangsu, 224002, China

Abstract
Volunteer computing is a way of supporting people around the world who provide free computer resources, to participate in scientific calculation or data analysis on the Internet. This provides an effective solution to solve the problems of large scale of basic scientific computing and more computing resources requirements. Task allocation is a very important part of volunteer computing. An effective algorithm can significantly improve computational efficiency. At present, most of the existing tasks are divided in term of the computer hardware conditions or the initial state of the computer in the volunteer computing. It seems that this have no obvious impact to calculating efficiency in a short time, but this task will be less flexible when idle resources of the volunteer computing becomes less or more. To make full use of idle computer resources, a dynamic task allocation algorithm (TAA) based on weighted velocity was proposed in this work. The research results showed that the weighted velocity as a parameter can be used to test the computing performance of a computer, dynamically manage task allocation as well.
Keywords: volunteer computing, task allocation, weighted average velocity

1. Introduction

In many areas, some projects with the huge amount of calculations is difficult to find a suitable algorithm, which face a shortage of funds, limited computing resources, low efficiency and other issues. Additionally, my computer is so idle that some resources are in “waste" in our daily lives. When we put the above two issues together, we found a very interesting thing that they are no longer difficult to solve, but gave us a problem-solving ideas (Wu et al., 2016). For scientists, volunteer computing means almost free and unlimited computing resources and volunteers so that they can get opportunities to understand and participate to the field on science to promote the understanding of science.

Volunteer computing can provide an effective way to resolve large-scale scientific computing, amounts of demand of computational resources, and so on (Durrani & Shamsi, 2014). Although the volunteer computing is currently paid the attention by more and more researchers, its development is still in its infancy in China.

Volunteer computing will divides a large operation into a small task, and assign them to volunteer computer (Iftikhar et al., 2016). In the process, the distribution of tasks has a great impact on the operation of computing platforms (Lee et al., 2010). The operating principle of volunteer computing is shown in Fig. 1. Based on the importance of the task allocation, it attracted numerous researchers in many fields to propose a number of related algorithms (Toth & Finkel, 2009). According to the basic principles, most task allocation algorithms (TAA) are currently heuristic algorithm, which may be divided into two categories, simple TAA and knowledge-based TAA.

Simple TAA does not consider the history of request resources for every voluntary computer, but consider each volunteer computer has similar performance. The typical representatives of this algorithm are first come first service (FCFS) algorithm (Estrada et al., 2009), random distribution algorithm, local adjustment strategy, etc.

The knowledge-based TAA may be affected by the history of requesting computing resource and the entire communication in the world. The following examples can illustrate knowledge-based TAA: (1) The world community networks carry out task allocation on the basis of the time of the each client's average return results (Toth & Finkel, 2009); (2) The threshold values can be divided into fixed thresholds and changeable threshold. For fixed thresholds the server must firstly calculate the availability and reliability of volunteer computer. If the calculated value is larger than a prior defined value, the server will distribute this task to volunteer computer. While for the changeable threshold in partitioning strategy, if fewer volunteer computer made a request, and more tasks are waiting for distribution, the system will reduced the defined threshold, otherwise it will increase (Anderson et al., 2005); (3) round robin priority policy is also a TAA based on the computing capacity, which is evaluated by the operation time duration of same computing task. Shorter the operation time, stronger the computing capacity.

Many of the existing TAA can effectively assess volunteer computer, and have made a strategy to manage these volunteer computers (Wu et al., 2016). For some computers have low reliability, and the credibility of its computing results will be so lower that its results will be abandoned. For some computers having weak computing capabilities, although they are willing to participate in volunteer computing, they cannot play its role due to the low performance of the overall platform. However, every same assignment is effectively assigned to the volunteer computer.

Based on the abovementioned problems, in this work, we focus on the dynamic TAA based on weighted velocity and propose a basis for the module analysis of the volunteer computing platforms, wherein tasks

distribution is based on the weighted average velocity. Compared with the simple average velocity, the weighted average velocity can evaluate a computer's performance over a period of time. And the weighted average velocity can also provide the basis for the next task to make task allocation dynamic implementation.

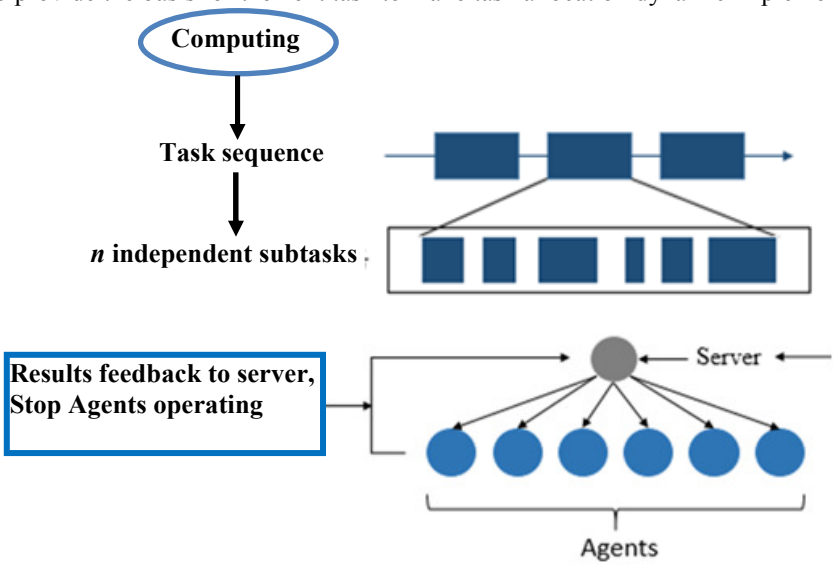

Fig. 1 Operating principle of volunteer computing.

2. A dynamic TAA based on weighted velocity

2.1. The basic idea of TAA

The statistics results showed that a computer revealed different computing capacity at different periods (Anderson et al., 2002). So when we assigned the task to the volunteer computer, we cannot only depend on the hardware conditions of computers. The best way to test a computer's contemporary computing capacity is running the test task (Wu et al., 2016). The method generally returns the total run time. According to the time and the size of the task we can obtain the average velocity. Taking into account the instability of the computer, the average velocity cannot sometimes accurately reflect the computing capacity of a computer. Therefore, we should introduce another parameter, the maximum velocity in running task, to obtain a weighted average velocity, which include three steps.

Step 1: Server firstly allocates the same size of sub-tasks to each volunteer computer. Agents receive sub-task and start processing, and the final results are returned. Depending on computer performance differences and idle level, the speed of return results is not the same.

Step 2: Server receives the returned results from agents, and weights the running speed to obtain the weighted average speed. The weighted average speed is multiplied by the running time to get size of the next task.

Step 3: Server divides the previous task into sub-tasks and assigns to the appropriate agents in a waiting state. After the completion of subtasks, agents still return their operating speeds for the next division of tasks.

In order to effectively manage the agents, agents will be added to the list and be divided (Zhou et al., 2016). The list is divided into IgnoreList and UseList. IgnoreList is usually used to record Agents who have the wrong calculation, client problems, etc. In contrast, UseList is used to record useful Agent. IgnoreList the Agent, Server does not consider to use. For a project, once agent is divided into IgnoreList, it will not be considered until the end of the project. When a new agent to join into a project, the Agent will be checked, and join UseList if useful, otherwise IgnoreList.

In order to ensure orderly implementation of the task, server will save a two-dimensional table, which records the AgentID. This management is effective in Agents and implementation of tasks, convenient to observe the progress of the task (Heien et al., 2009).

2.2 Algorithm Description

Make $agent_j$ represents the j-th Agent, n is the current total number of Agents involved in the calculation, and $j \in [1, n]$. In the $agent_j$, task (i, j) represents the i-th task split and the size of the j-th task, and m represents the total split times of the task, wherein $i \in [1, m]$. w_1 is the weight value of the average velocity $V_{average}$ (i, j), w_2 is the weight value of maximum velocity V_{max} (i, j), and $w_1 + w_2 = 1$. Symbols and illustrations used in the algorithm description and are listed in Table 1.

Table 1 Symbols and description used in algorithm

Symbols	Illustrations
n	the current total number of Agents
m	the total split times of the task
task(i,j)	the i-th task split and the size of the j-th task
v(i,j)	the i-th task split and the weighted average velocity
t(i,j)	the i-th task split and running time of the j-th Agent
$V_{max}(i,j)$	the i-th task split and maximum velocity of the j-th Agent
$V_{average}(i,j)$	the i-th task split and average velocity of the j-th Agent

In running the algorithm, the first step required all agents to run the test task. The specific process is described as follows. First, server divides big task in the project into small subtasks of the same size. Then these subtasks need to be confirmed to participation volunteer computing in various agents. At this time subtasks in $agent_j$ can be represented as task (1, j), and task (1,1) = task (1, j), $j \in [2, n]$. Secondly, server collects the returned results from agents. The returned results from $agent_j$ contain two data, t (1, j) and V_{max} (1, j). After that, server starts to process the data.

For $agent_j$ data as an example, its main modules are (1) calculation of the average velocity: $V_{average}(1, j) = task(1, j) / t(1, j)$; (2) calculation of the weighted average velocity: $v(1, j) = w_1 \times V_{average}(1, j) + w_2 \times V_{max}(1, j)$; (3) calculation of the size of next task (2, j): task (2, j) = v (1, j) × T (T = 1 min). The final step is to divide a large task according to task (2, j) to prepare the next task allocation. The diagram of first task allocation is presented in Fig. 2.

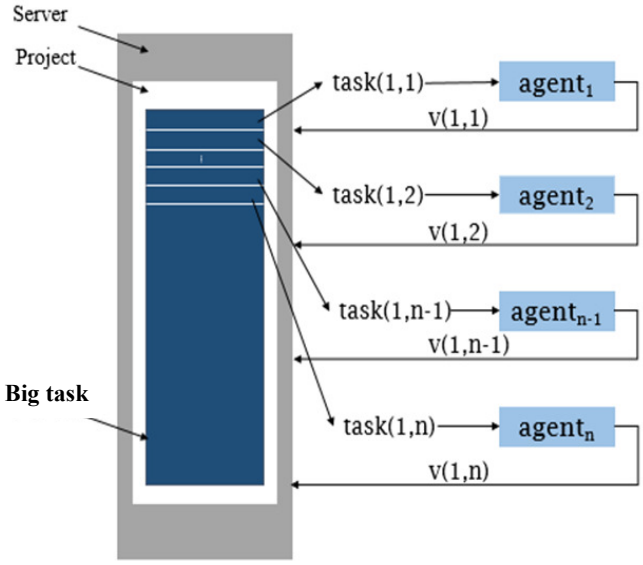

Fig.2 Diagram of the first task allocation

The pseudo code is displayed as follows.

Begin:
Initialize management list of agent，agent total numbers n=0，the split times i=1；
While (Result == false)
n++；
While (! Result)
IF (i=1)
For each $agent_j \in$ UseList
Run task (1, j), and task(1, j)= task(1,1)
Return t (1, j), V_{max} (1, j), Result;
For server
Calculate weighted average velocity v (1, j)
Calculate task (2, j)
Else IF (i !=1)
For each $agent_j \in$ UseList
Run task (i, j)；
Return t (i, j), Vmax (i, j),Result;

For server
Calculate weighted average velocity v (i, j)
Calculate task (i+1, j)
Break;
End

After completion of the test run, we can initially know the computational performance of Agents through a weighted average velocity. When the second distribution of tasks is coming, we can explicitly calculate the task (2, j), and server only make the appropriate allocation of division of big tasks. It should be noted that the second division of tasks will also return similar results, which will be represented as t (2, j) and V_{max} (2, j). $V_{average}$ (2, j), v (2, j) and task (3, j) are correspondingly be obtained as well. And by this analogy, the i-th task allocation is easily deduced. The i-th task allocation diagram is shown in Fig. 3. Idle resources of computer are highly unstable. The user's actions sometimes affect the use of idle resources. Therefore, this dynamic TAA with weighted velocity as a parameter can make better use of computing resources.

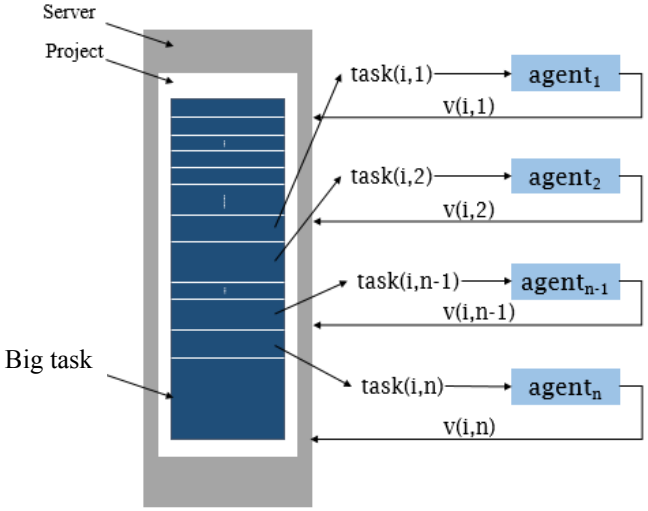

Fig. 3 Diagram of the i-th task allocation

3. Experimental

To test the performance of the proposed algorithm, the dynamic TAA with weighted velocity as a parameter was built, and HCEP (HashCatEnigma Project) was also run on volunteer platform three experiments, which includes (1) collection of the subtasks size assigned to agents on heterogeneous volunteer computing platform, (2) comparison experiment with static task allocation algorithm, (3) comparison experiment with the existing volunteer computing platform.

3.1. Experimental data explanation

HCEP is an experimental project run on a volunteer computing platform. It crack the password by using a dictionary (Zhang & Meng, 2009). The project has the following main feature, large task, low complexity of task unit and the split task. When performing the experiment, we need several computers, and which is installed agent client. You can use the account, password to log agent client. After successful completion of tests, agents can be registered in server, followed by HCE tasks allocation.

The first part of the experiment is to build HCEP experimental environment by using dynamic TAA with a weighted average velocity as a parameter, followed by collection of records for the subtasks size in eight assignments. The second part of the experiment is to compare two TAA, static TAA and dynamic TAA with a weighted average velocity as a parameter. The third part of the experiment is to carry out a platform comparison experiment to test the availability of algorithm proposed in this study.

3.2. Results and analysis

3.2.1. The first part of the experiment

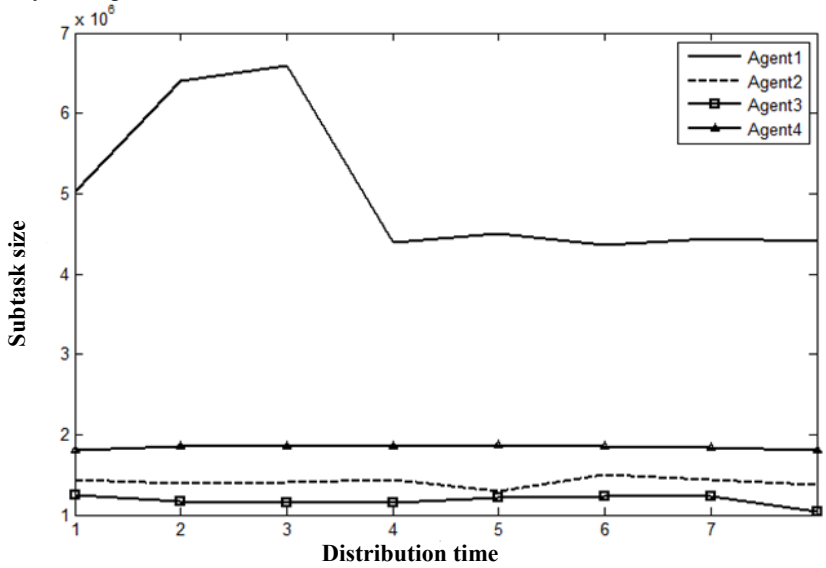

Fig .4 The change of subtask size of Agents

In the part, four agents (Agent 1, Agent 2, Agent 3, Agent 4) were selected to record the data in once password cracker project. The data primarily included the subtasks size for i to i + 7 task allocations. Fig. 4 displayed the change of subtask size during the execution of task for eight times. It can be observed that the amounts of tasks assigned to each Agent1 are higher than the other agents. This reason is that Agent1 has excellent hardware condition and GPU performance.

3.2.2. The second part of the experiment

Through several experiments, static TAA was compare with dynamic TAA in this work. The time used six times in cracking the password by two algorithms was shown in Fig. 5. Due to the influence from the number and properties of the agents, the differences of final results have certain randomness. Overall, the efficiency of dynamic TAA was higher than static TAA.

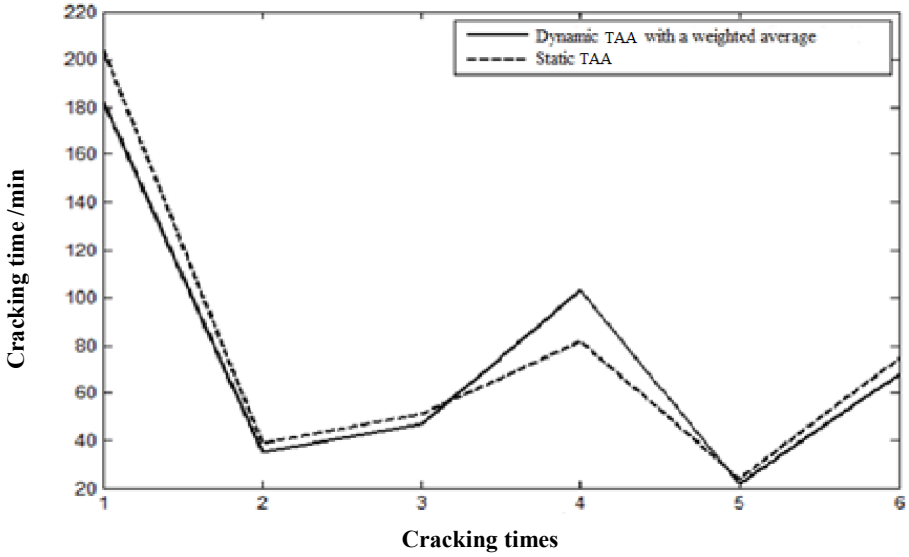

Fig. 5 The time used in cracking the password by two TAA

3.2.3. The third part of the experiment

A comparison experiment of dynamic TAA platform and EWSA (Elcomsoft Wireless Security Auditor) by cracking a password program was performed, and we can get the time they use, respectively. To ensure the reliability, the experiment will be conducted several times. Fig. 6 showed the record of the experimental results by six times. It can be seen that heterogeneous volunteer computing platform with dynamic TAA using a weighted average velocity spent less time than EWAS in cracking same password.

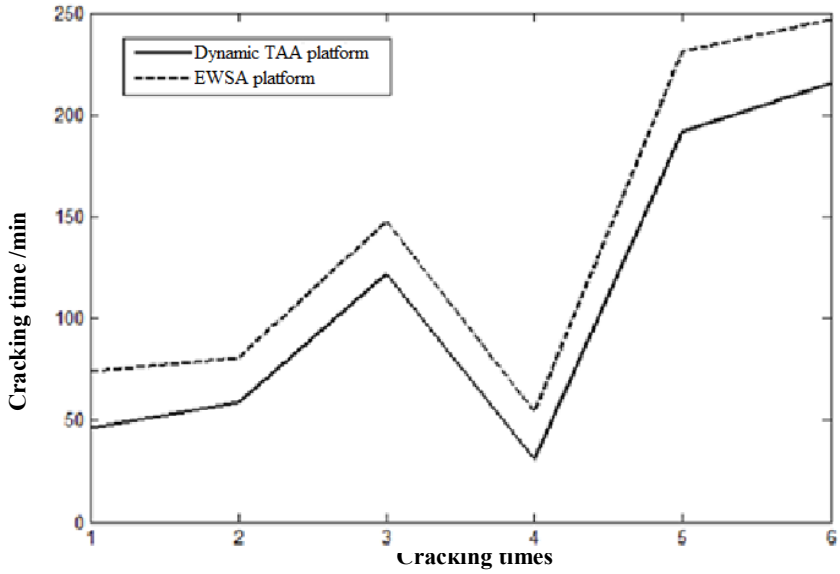

Fig. 6 The comparative experiments of heterogeneous volunteer computing platform with dynamic TAA and EWSA.

By comparison of the above three experiments，it can be clearly seen that dynamic TAA with the weighted velocity revealed better performance. And for unknown performance of the agents and real-time resource status changed frequently, dynamic TAA with the weighted velocity exhibits high stability, high availability. To some extent, dynamic TAA reduce the over-reliance on agent performance.

4. Conclusion

This work studies task allocation strategy of the volunteer computing platform. The weighted velocity was found to effectively evaluate the computing capacity of a computer. Although the state of the computer changes frequently, we can change the subtasks size to accommodate different computers. A link between the subtasks size and the weighted velocity is that the returned weighted velocity determines the next subtasks size. In order to improve the efficiency of volunteer computing, we may build a group for volunteer computers. Grouping for similar resources is not only to facilitate the management of agents by server, but also to increase the accuracy of volunteer computing (Zhou et al., 2016). Volunteer computing is computing model with the features of low-cost, simple to build, high performance. Task allocation as an important part of volunteer computing has high research value (Khan et al., 2016).

Acknowledgments

This study was funded by Natural Science Research Project of Yancheng Teachers University (No.10YCKL032), and Jiangsu Province College Students' Innovative Training Project (201410324008Z). All authors declare no financial/commercial conflicts of interest.

References

Durrani MN, Shamsi JA. Volunteer computing: requirements, challenges, and solutions. Journal of Network and Computer Applications, 2014, 39:369-380.

Lee YC, Zomaya AY, Siegel HJ. Robust task scheduling for volunteer computing systems. The Journal of Supercomputing, 2010, 53(1):163-181.

Toth D, Finkel D. Improving the productivity of volunteer computing by using the most effective task retrieval policies. Journal of Grid Computing, 2009, 7(4):519-535.

Estrada T, Taufer M, Anderson DP. Performance prediction and analysis of BOINC projects: an empirical study with EmBOINC. Journal of Grid Computing, 2009, 7(4):537-554.

Anderson DP, Korpela E, Walton R. High-performance task allocation for volunteer computing. In: Proceedings of the first IEEE international conference one-science and grid computing, 2005.

Anderson DP, Cobb J，Korpela E，Lebofsky M，Werthimer D. SETI@home: An Experiment in Public-Resource Computing. Communications of the ACM, 2002, 45(11):56-61.

Heien EM, Anderson DP, Hagihara K. Computing low latency batches with unreliable workers in volunteer computing environments. Journal of Grid Computing 2009,7(4):501-518.

Iftikhar S, Tariq A, Khan SA. Optimal Task Allocation Algorithm for Cost Minimization and Load Balancing of GSD Teams. Lecture Notes on Software Engineering, 2016, 4(1): 16-19,

Wu DF, Zeng GP, He D, Qian ZP, Zhang QC. Task Coordination Organization Model and the Task Allocation Algorithm for Resource Contention of the Syncretic System. Tsinghua Science & Technology, 2016, 21(4):459-470.

Zhang Y, Meng Y. Dynamic multi-robot task allocation for intruder detection. International Conference on Information and Automation. IEEE, 2009:1081-1086.

Zhou Y, Fei C, Wang W. Task similarity-based task allocation approach in multi-agent engineering software systems. Journal of Information Science and Engineering, 2016, 32(4): 1021-1039.

Khan MZ, Devi G, Alginahi YM. Study and Analysis of Cloud Aided Remote Sensing Multiprocessing System (CARMS) with Task Allocation. International Journal of Future Computer and Communication, 2016, 5(1): 47-52.