Representing Job Scheduling for Volunteer Grid Environment using Online Container Stowage

Saddaf Rubab[1] , Mohd Fadzil Hassan[2], Ahmad Kamil Mahmood [3], and Syed Nasir Mehmood Shah[4]

[1,2,3,4]Department of Computer and Information Sciences, University Teknologi PETRONAS, Bandar Seri Iskandar, 32610, Tronoh, Perak, Malaysia

***Abstract*—Volunteer grid computing comprises of volunteer resources which are unpredictable in nature and as such the scheduling of jobs among these resources could be very uncertain. It is also difficult to ensure the successful completion of submitted jobs on volunteer resources as these resources may opt to withdraw from the grid system anytime or there might be a resource failure, which requires job reassignments. However, a careful consideration of future jobs can make scheduling of jobs more reliable on volunteer resources. There are two possibilities; either to forecast the future jobs or to forecast the resource availability by studying the history events. In this paper an attempt has been made to utilize the future job forecasting in improving the job scheduling experience with volunteer grid resources. A scheduling approach is proposed that uses container stowage to allocate the volunteer grid resources based on the jobs submitted. The proposed scheduling approach optimizes the number of resources actively used. The approach presents online container stowage adaptability for scheduling jobs using volunteer grid resources. The performance has been evaluated by making comparison to other scheduling algorithms adopted in volunteer grid. The simulation results have shown that the proposed approach performs better in terms of average turnaround and waiting time in comparison with existing scheduling algorithms. The job load forecast also reduced the number of job reassignments.**

Keywords*—Volunteer grid computing; volunteer resources; container stowage; job scheduling

I. INTRODUCTION

Volunteer grid computing environment is a type of grid consisting of volunteered resources which are distributed and heterogeneous in nature [1]. The volunteer grid is growing day by day as more number of resources is volunteered for high computational research projects. The 'World Community Grid' is a large-scale volunteer computing project supported by IBM [2]. The statistics for number of resources and research projects running under it is shown in Fig. 1.

The resources and jobs submitted to a volunteer grid are unpredictable which affects the performance of volunteer grid and makes resource management more challenging. It is therefore difficult to schedule the jobs as the future job rate and resource availability cannot be anticipated.

The jobs submitted to Volunteer Grid (VG) may vary in terms of requirements and load. The job requirements change due to the nature of tasks to be performed and sometimes based on the time of the day job has been submitted. There might be a few jobs submitted to volunteer grid repeatedly. These repeated jobs may have random load and resource requirements on each job submission. It is hard to forecast the resource demands. There have been studies [3-5] which suggest to allocate a few extra resources to complete a job in VG.

However, it is not an efficient way to over-provision the resources because it will leave the allocated resources as under-utilized. This underutilization of resources is negating the objective of VG to maximize the utilization of allocated resources.

Fig. 1. World Community Grid Volunteer Statistics [2]

Job migration allows a job to be transferred from one resource to another without causing any interruption in job execution. Job migration can help to balance the load and to transfer a job from underutilized resource to another one for achieving VG maximum resource utilization objective where the job can also complete within the specified deadline.

For example, when most of the jobs running on one resource and there is a possibility that a few jobs can miss their deadlines, those jobs can be transferred to other available resources which can complete the jobs within their specified deadlines.

In contrast, when jobs are being executed on a resource which is underutilized, the overall performance of resource is getting low, the jobs can be transferred to other resources which can complete the jobs within deadline and free the underutilized resources. This will not only help to maximize the resource utilization of new resource but also help to free the previous resource for other large job executions. There are practices of migrating jobs to improve the turnaround time of jobs [6-8]. The work presented in [6], proposed a job scheduling strategy which includes history events to make possible a job scheduling scheme which results in fewer job migrations and improve the turnaround time as well. Various job migration strategies are presented in [7], for migrating the unfinished jobs that are delayed or halted on any node.

The job scheduling and resource allocation problem can be demonstrated as container stowage problem where each job is considered as a container and resource as a ship or terminal to pack the containers. Container stowage itself is NP-Hard problem, which requires stowing the containers in vessels or ships in order to reduce the operating costs and deliver the containers at their destination within the budgetary values and time [9, 10]. Container stowage is illustrated in Fig. 2.

The containers are to be stowed to a ship which can deliver the containers to destination within the time and operational cost. A container can only be stowed to one ship at a time whereas many containers can be stowed to one ship. This can depict the job scheduling, as illustrated in Fig. 3. A job can be assigned to one resource at a time and a resource can be allocated to more than one job considering the time and cost constraints associated with the jobs and resources.

In this paper, a job scheduling approach has been presented that uses online container stowage to allocate the volunteer grid resources dynamically based on the job requirements. The proposed approach will also optimize the number of resources used in terms of using the allocated resources to their maximum instead of using excessive resources. The main contributions of this work are:

- To develop an online container stowage job scheduling algorithm that is able to avoid the overloading of resources while ensuring the maximum utilization
- A theoretical proof for optimal value of number of resources in use
- Simulation results to compare with existing job scheduling approaches

The outline of this paper is as follows. Section II describes a literature studied on job scheduling in volunteer grid computing and job reassignments. Section III gives a broad overview of job scheduling approach whereas Section IV focusing mainly on the proposed job scheduling algorithm. The results and simulations are discussed in Section V. Section VI concludes the paper giving the future research direction.

II. Related Literature

This section will give a brief literature review on the job scheduling in volunteer grid which not only will outlines the practices of job scheduling but also the issues, challenges and methods on job reassignment.

Due to the growing use of distributed computing resources, the jobs scheduling becomes an important issue to be studied. Therefore, the job scheduling in volunteer grid has been studied vastly in the literature. A survey [11] has been presented on grid resource management systems, mainly discussing the grid schedulers such as Condor, AppLes, Globus and Nimrod which use batch scheduling heuristics. Few of the scheduling algorithms for volunteer grid are discussed and compared under different input conditions applied using simulation [12].

A 39-days trace of computer availability of 32 machines located in two classrooms has been collected in [13]. These traces were used for scheduling techniques analysis to improve average turnaround time in volunteer grid environment. A tool, named DGSchedSim [14] has also been presented later on to evaluate different other volunteer grid scheduling algorithms using the collected traces by [13].

A stochastic modeling based job scheduler was presented in [15]. A job scheduling architecture using performance prediction was proposed in [11], using the neural network that focuses on local job scheduling on volunteer grid resources. Cost based online job scheduling algorithm is presented by Weng et al. [18]. They have compared the performance of proposed online algorithm with the optimal offline algorithm.

The job scheduling performance in volunteer grid environment can be affected because of resource failure or resource withdrawn. These can be avoided by migrating or reassigning jobs to other available resources [6-8].

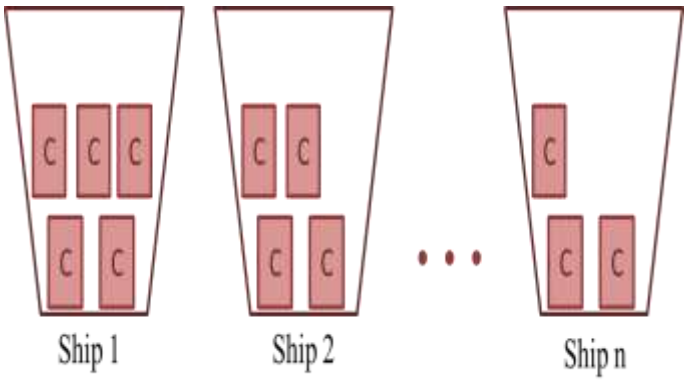

Fig. 2. Illustration of Container Stowage Problem

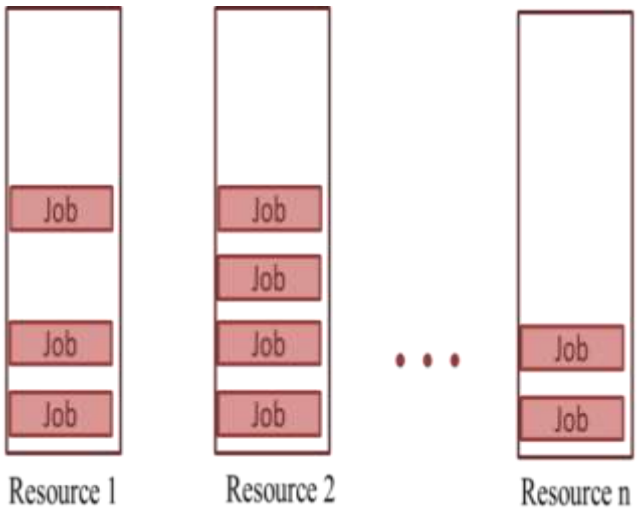

Fig. 3. Illustration of Job Scheduling

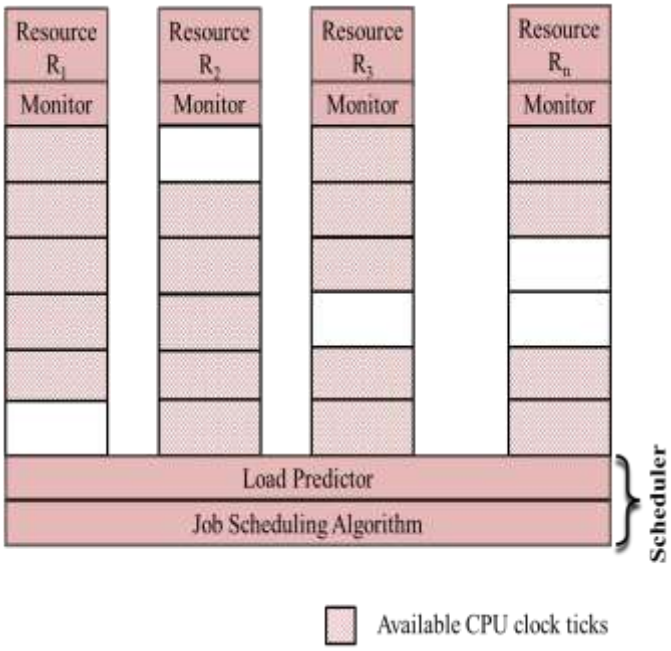

Fig. 4. Scheduling Approach

A job scheduling strategy based on neural network load predictions was proposed in [6] that reassigns the jobs from current resource to another available volunteer resource. Different job scheduling strategies including migration adaptive, wave migration and immediate migration were presented to get better turnaround time for job scheduling using volunteer grid resources [7]. The adaptive scheduling was used by Zhou et al. [7] to reassign the delayed jobs from current nodes to less load classified as night nodes. Most of the job scheduling algorithms and strategies reviewed in this section are greedy and offline algorithms.

The change in volunteer grid environment is mainly due to the resource availability and failure, which is the prime reason for reassignment of jobs. Job scheduling in volunteer grid computing environment can mimic the container stowage problem where the containers need to be stowed in ships and vessels while meeting their time and budgetary constraints. Container stowage problem has been tried to solve by using genetic algorithms, combinatorial optimization and heuristics etc. [9, 10].

Considering this fact, a job scheduling scheme is proposed in this paper which is adaptive in nature and based on container stowage problem concepts. The proposed job scheduling will also use the load prediction/forecast to improve the average turnaround time for scheduling using volunteer resources.

III. Overview of Proposed Job Scheduling Approach

The proposed scheduling approach consists of a job scheduler which can schedule the jobs optimally to complete the submitted jobs within the deadline. The job scheduler is the one responsible for running proposed scheduling algorithm. The volunteer resources will run the submitted and assigned jobs to them individually.

Each of the resource can have different availability times, which illustrates that there can be availability time intervals which can be further sub-divided to fit in more jobs. In such a case multiple jobs can share one resource. The resource monitor attached with each resource will collect the usage information in terms of availability time which means that for how many CPU clock ticks a resource can serve a job and how many available CPU clock ticks are still unassigned with the resource for more jobs. This usage information will be used by scheduler, which is responsible for overall job scheduling. The scheduler required the following three (03) inputs:

- resource demand history of jobs
- capacity and load history of resource
- current allocation of resources to jobs

The scheduler has two main components as well. The load predictor (LP), that is responsible for predicting the resource demands in near future. The second component of scheduler is the proposed scheduling algorithm itself, which can optimally schedule the jobs such that the jobs are completed within the deadline specified and the currently assigned resources are utilized maximum. The overview of job scheduling approach is presented in Fig. 4. The scheduler will be periodically called after a fixed number of CPU clock ticks to see if there is any new job arrived or relocation of job is required. In each of scheduler call, the load predictor will predict the resource demands of new submitted jobs and resource load based on usage information. The prediction results will be represented as CPU clock ticks i.e., the required number of CPU cycles for job and unassigned available CPU cycles.

After the load predictor, the predictions results will be passed to scheduling algorithm to find that there are enough resources available to assign more jobs and to execute the already assigned jobs. If the resources can complete the already assigned jobs and new jobs, the resource allocation will be done locally. In other case, the overload mechanism will be performed.

The scheduling algorithm also reschedules the running jobs from underutilized resources to the nearly optimal maximum utilized resource in order to free the underutilized resource for future jobs which may require more CPU clock ticks and help to use the maximum of resource. The scheduling algorithm will then generate an allocation/re-allocation list (AR list) and pass it to resource monitor to start the job execution on allocated resources. The scheduling approach makes use of live reallocation of jobs, which is itself incurring an overhead but the overall scheduling performance can be improved because the overhead is very negligible.

To analyze the load prediction, in each set of jobs e.g., 250 jobs, the first set of jobs is restricted to 10 jobs and next set is considered as predicted jobs. This scheme can be followed for all the job sets as these are multiples of 10. The load prediction can be done using any time series forecasting method like ARMA, ARCH, GARCH, and Holt-Winters [12-14] etc. The effect of load prediction will be discussed in Section 0

IV. JOB SCHEDULING AS CONTAINER STOWAGE PROBLEM

The container stowage planning for is a core activity of shipping and difficult to solve because of combinatorial nature of alternative mappings of containers to the stowage location in a ship or vessel [10]. Container stowage can be used to demonstrate the job scheduling where each of the resource is a ship and each job is a container to be stowed.

Extensive literature is available on the container stowage problem however those presented solutions are not feasible to apply in grid and volunteer grid environment specifically. The traditional container stowage solutions can show performance ratio approximately one which suggests having an approach to call upon the stowage planning solution after a fixed interval of time to assign and reassign the jobs. Due to the reassignment, there is a possibility of having many job migrations from one resource to another when the resource is underutilized or overloaded because the traditional scheduling algorithms does not take new jobs arriving to the system in consideration that can affect the overall system performance. These all algorithms are usually termed as offline algorithms.

The online container stowage algorithms can be a solution to reduce the number of job migrations. Although there are online container stowage algorithms, which does not take the following container details to avoid container/job migrations. A few online container stowage algorithms do not allow migration of already stowed container to a new ship/vessel location. If such an online container stowage algorithm is applied, this will be a limitation to our job scheduling approach although the volunteer grid environment allows migration of the jobs. Using the authors' experimental setup, it will be proved that job migration using online container stowage algorithm using volunteer grid resources will help to achieve nearly optimal results.

It must be considered that at time of job migration from one resource to another, the required CPU burst time of job might have been reduced as the job has an opportunity to utilize the current resource for its execution. This presents that a job requirement can be changed after its first assignment to the current resource. The changed job requirements compel to have an online algorithm which can accommodate not only the future jobs but also ensures the maximum utilization of resources in use and free the nearly idle resources. The scheduling algorithm proposed in this study is named as Online Container Stowage Job Scheduling (onCSJS). onCSJS illustrates the container stowage planning by ensuring the behavior of containers and ships/vessels. The ships/vessels are not allowed to stow the containers to the maximum limit as it unfavorable for the ship stability. The resources in onCSJS will mimic the same behavior by not allowing the jobs to fully utilize the volunteer resource as it will not only provides a possibility of overloading but also the migration of large number of jobs in case of resource failure.

A. Job Scheduling Algorithm

The main objective of onCSJS is to improve the job scheduling for volunteer grid environment to complete the jobs within deadline and to use maximum of the resources in use by freeing the nearly idle resources. The onCSJS is online relaxed job scheduling algorithm utilizing the concept of container stowage by not considering the new jobs when reassigning the old jobs and only a few reassignments are acceptable. The proposed algorithm onCSJS not only assign the new jobs but also perform the reassignment of old jobs currently running on volunteer resources. The job assignment to the available volunteer resources will be performed by calling assign(job) function. reassign(oldjobs, job) will be called to reassign the old jobs from current volunteer resource to new resource. During the reassignment operation the old jobs which have been executed partially on current resources will be considered as new jobs because the remaining CPU burst time are changed and will require less CPU clock ticks to complete the job. The description of algorithm will be clearer by understanding the following representations firstly in container stowage:

- Let c as a container to be stowed and $size(c)$ as the size of container, where $size(c) \epsilon\ (0, 1]$
- Let sv as the ship/vessel. The $total\_space(sv)$ is the total space available in a ship for container stowage and $space(sv)$ as the space available in sv ship for more containers, where $total\_space(sv)\ \epsilon\ (0, 1]$

The container stowage problem must satisfy the equation (1) such that the total space of ship/vessel $\boldsymbol{sv}$ must be greater than or equal to the total of already stowed containers and space left after stowing a new container $\boldsymbol{c_i}$.

$$space_i(sv) + c_i \leq total_{space_i}(sv) \quad (1)$$

The representation of container stowage has been translated for job scheduling algorithm in volunteer grid environment.

- Let c as a job to be assigned and $size(c)$ as the size

of job, where $size(c) \epsilon (0, 1]$

- Let sv as the resource. The $total\_space(sv)$ is the total space available in a resource for scheduling job and $space(sv)$ as the space available in sv resource for more jobs to be assigned, where $total\_space_{(sv)\epsilon}(0, 1]$

The job scheduling must satisfy the equation (1) such that the total space of resource $\boldsymbol{sv}$ must be greater than or equal to the total of already assigned jobs and space left after assigning a new job $\boldsymbol{c_i}$.

The $\boldsymbol{total\_space}$ of resource is translated in the range of 0 to 1, which requires classifying the jobs. Following are the four (04) classes of jobs:

- S-con: Small jobs $size(c) \epsilon (0, 1/3]$
- M-con: Medium jobs $size(c) \epsilon (1/3, 1/2]$
- L-con: Large jobs $size(c) \epsilon (1/2, 2/3]$
- VL-con: Very Large jobs $size(c) \epsilon (2/3, 1]$

The system performance can be increased if a threshold is set for the maximum resource to be utilized. In simulation test run of onCSJS, the 0.75 or 75% of the total resource available will be set as CPU clock ticks that a resource can contribute.

The volunteer resources are arranged considering the classification of jobs each of it will have the $\boldsymbol{total\_space_{(sv)\epsilon}(0, 1]}$

- S-ship: Small resource
- M-ship: Medium resource
- L-ship: Large resource
- VL-ship: Very Large resource

Since the resources are heterogeneous in volunteer grid environment, a few combinations of these resource classes are also considered including SL-ship, MM-ship and ML-ship. There are some constraints to be considered while assigning or reassigning jobs to the resources. The S-ship resource can only have a few S-con jobs only. M-ship resource can be allocated to only two M-con jobs, whereas L-ship is for only two L-con job. The VL-ship can have two VL-con jobs only. SL-ship will be allocated to one L-con job and few S-con jobs. MM-ship will only have two M-con jobs. ML-ship can have one M-con job and one L-con job.

Further, there are groups formed from the S-con submitted jobs such that the group size of each is 1/3 of the total of S-con jobs submitted to be scheduled. This will help to assign the S-con jobs (small jobs) in a few steps and waiting time for these jobs will be less. It will also reduce the overhead in case of reassignment of jobs as overhead will be more for L-con or VL-con jobs in comparison with S-con jobs if require reassignment.

V. Results and Simulation

A. Experimental Setup

SETI@home [15] has been selected for resources and LCG1 [16, 17] dataset has been used for jobs submitted to be scheduled on volunteer resources. SETI@home project has recorded activities of 60883 nodes for a period of 10 months [15]. In SETI@home there are missing values such as zero or negative values in RAM size, null values are saved in location, null values in time zone and other components. These all null, zero and negative values were removed before starting the simulation using a pre-processing method. After pre-processing, only 38,166 nodes are having complete data. Table 1 shows population of nodes after pre-processing the missed values.

TABLE I. Number of Nodes after Pre-processing

Type of Nodes	Number of Nodes
Initial	60883
After pre-processing Location	38180
After pre-processing location RAM size, Time zone	38166

The LCG1 dataset contains 11 days of recorded node activities with 188,041 jobs of 53y 179d 7h 26m 46s CPU time The details can be studied in an online published report [17]. For benchmarking only top 15 nodes activities for 5 days from processed resource dataset has been selected and total of 1000 jobs from LCG1 has been selected randomly.

B. Benchmarking of onCSJS

The performance of proposed algorithm onCSJS has been compared with EDF (Earliest Deadline First), LLF (Least Laxity First), RM (Rate Monotonic), FCFS (First Come First Serve) and RR (Round Robin) [18-20] using trace datasets available online.

C. Active Resources

In the simulation run, a resource with less than two jobs is considered as non-active. Fig. 5 shows the number of active resources during 5 days with 5 hours difference. It has been assumed in onCSJS that the active resources are those which are allocated to two or more jobs. The reason of a resource being non-active could be the reassignment of jobs or resources are idle from the start. If a resource has been assigned only one job, it will be reassigned to another active resource if can execute, and current resource will become non-active. It has been observed that if the number of jobs is less there will be less number of active resources.

On the contrary, if the number of jobs is increased preserving the same amount of resources, the number of active resources will be more.

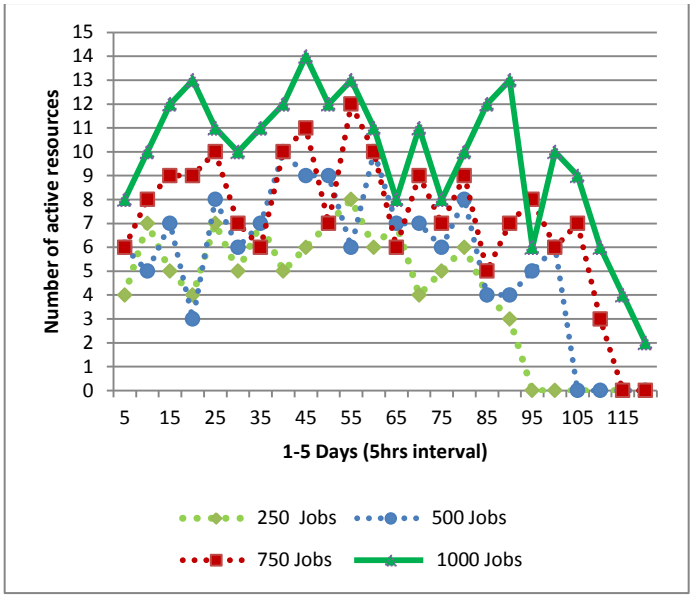

Fig. 5. Active Resources 1-5 Days

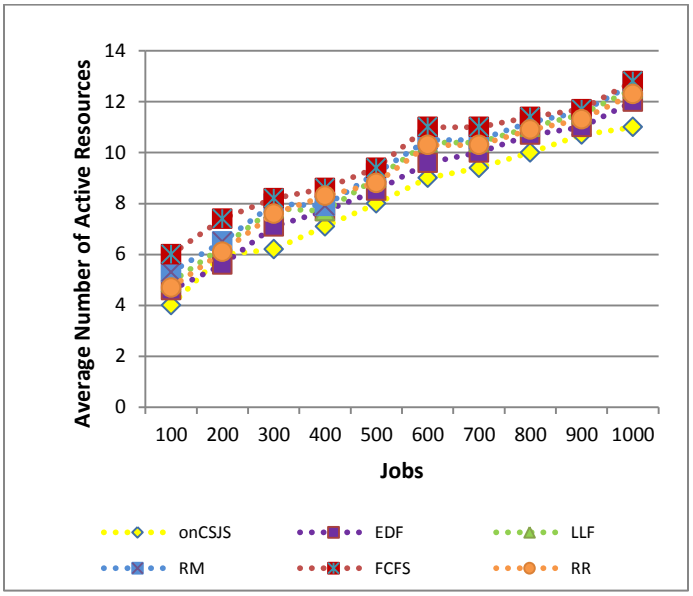

Fig. 6. Load on Active Resources

It proves our claim that active resources must be ensured that they are being used to the maximum and rest of the resources are saved for any future jobs. The scheduling algorithms chosen for benchmarking do not consider the factor of active and non-active resources; therefore, it is not a valid justification to make a comparison with the proposed algorithm.

D. Load on Active Resources

From Fig. 6, the behavior of active resources is easy to study. Fig. 3 gives more detail analysis of active resources with respect to the jobs scheduled using different algorithms. The proposed algorithm onCSJS used less number of active resources and tried to use the maximum of the active resources.

The difference will be clearer if more number of resources is used. In the simulation, only 15 volunteer resources are selected, however of the number of resources are more, the difference in number of active resources using onCSJS as compared with other scheduling algorithms will be more evident.

E. eassignment of Jobs

Fig. 7 presents the number of reassignments in case the job has to be migrated from one resource to another depending on the scheduling algorithm being used to test the performance. The number of reassignments is increasing as the number of jobs submitted increased. The number of job migrations is more in onCSJS as compared to the RR and RM because the proposed algorithm focuses more on the overall performance rather than on individual job runs.

F. Performance Comparisons

The performance of job scheduling algorithm can be explained briefly with the help of average waiting time and average turnaround time. The onCSJS scheduling algorithm performs better than the baseline scheduling algorithms for both the average waiting time and average turnaround time.

The average waiting time of onCSJS and baseline scheduling algorithms is presented in Fig. 8. The average waiting time of onCSJS and EDF is very close and there is a significant difference with other scheduling algorithms.

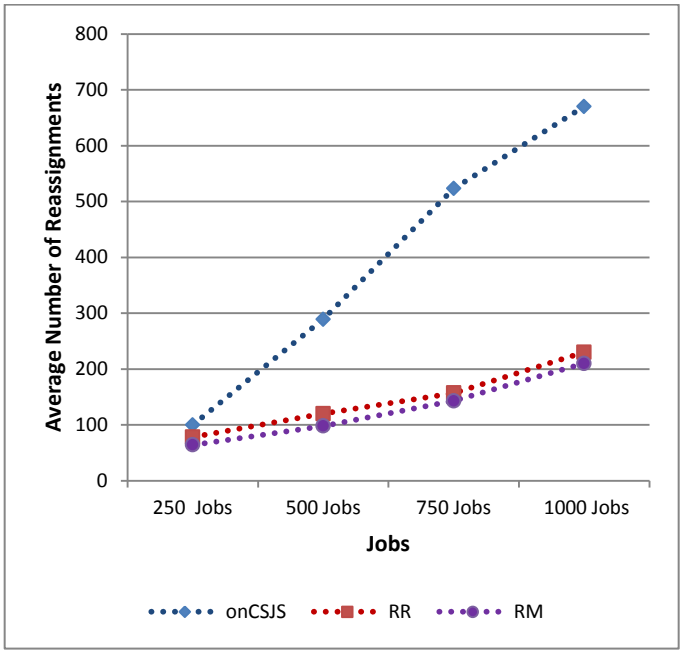

Fig. 7. Reassignments on Jobs

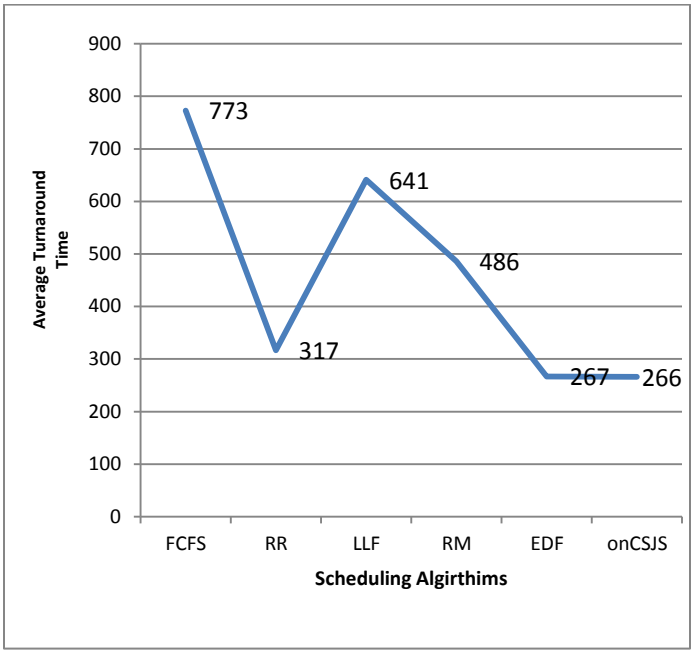

Fig. 9. Average Turnaround Time

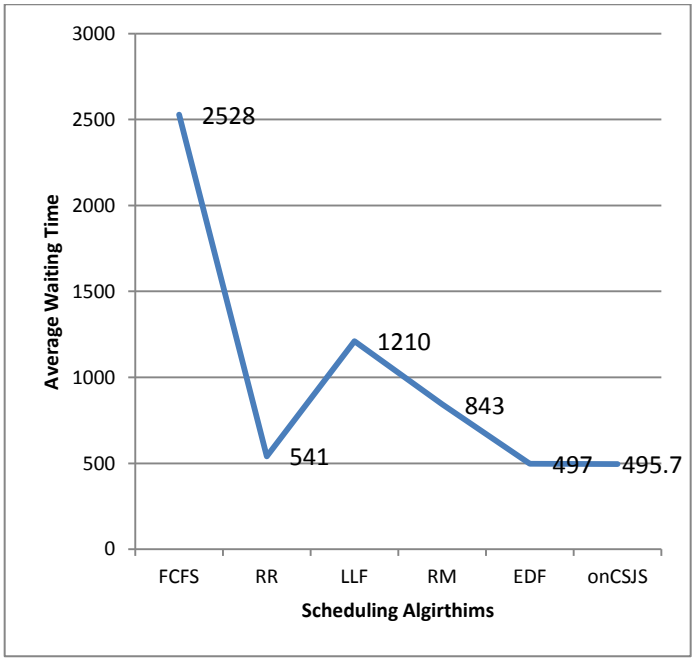

Fig. 8. Average Waiting Time

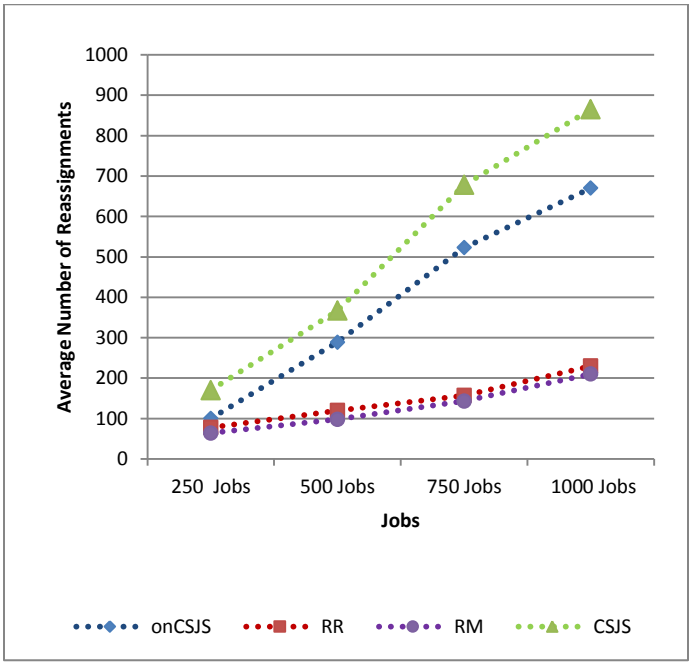

Fig. 10. Reassignment of Jobs with onCSJS

The turnaround time of onCSJS is presented in Fig. 9. It has been observed that average turnaround time calculated using onCSJS is very less than the baseline scheduling algorithms. The less turnaround time can be a similar effect for less waiting time. The less waiting time and less turnaround time are due to the reason of scheduling the jobs to a resource with available CPU clock ticks that can be allocated to submitted jobs.

G. Load Predictor Analysis

If the load prediction is not included in the proposed job scheduling algorithm onCSJS, let's call that CSJS. In CSJS the number of job reassignments will be more because there is no consideration for the future job load on volunteer re-sources and the job scheduling will be different. The number of job reassignments is very less in onCSJS (Fig. 7) as compared to CSJS (Fig. 10). This proves that if job prediction is made using any forecasting method, it can help to reduce the job reassignments overhead. The results are tabulated in Table II.

TABLE II. NUMBER OF REASSIGNMENTS

Scheduling Algorithms	**Jobs**			
	250	**500**	**750**	**1000**
onCSJS	100	289	523	670
RR	78	120	157	230
RM	64	98	143	210
CSJS	170	367	678	865

VI. CONCLUSION

There are job scheduling policies which can make use of history events. The job scheduling in volunteer grid computing environment can be aided with container stowage considering the jobs as containers and resources as ships or vessels. A job scheduling algorithm using the container stowage has been proposed for volunteer grid computing environment. The design and evaluation has been discussed in details by making comparisons with the other job scheduling algorithms including EDF, RM, RR, LLF and FCFS. The proposed algorithm considers job reassignments dynamically that's why it is named as onCSJS. The effect of not including reassignments has also been discussed. The onCSJS takes history events into account at time of assigning jobs to volunteer resources. If the history events are not taken in considerations, it will increase the number of reassignments and we call it as CSJS.

In future, the onCSJS can be incorporated in the middleware of volunteer grid to study its impact in real environment. A more accurate forecasting method can be engaged rather than taking the next batch of jobs as forecasted jobs.

ACKNOWLEDGMENT

Authors appreciatively acknowledge the High Performance Computing Center (HPCC) at Universiti Teknologi PETRONAS (UTP), Malaysia for providing the volunteer grid computing environment.

REFERENCES

[1] M. Nouman Durrani and J. A. Shamsi, "Volunteer computing: requirements, challenges, and solutions," Journal of Network and Computer Applications, vol. 39, pp. 369-380, 2014.

[2] J. W. Taylor and R. D. Snyder, "Forecasting intraday time series with multiple seasonal cycles using parsimonious seasonal exponential smoothing," Omega, vol. 40, pp. 748-757, 2012.

[3] N. P. Preve, Grid Computing: Towards a Global Interconnected Infrastructure: Springer-Verlag London, 2011.

[4] K. Chard, K. Bubendorfer, and P. Komisarczuk, "High occupancy resource allocation for grid and cloud systems, a study with DRIVE," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010, pp. 73-84.

[5] S. Krawczyk and K. Bubendorfer, "Grid resource allocation: allocation mechanisms and utilisation patterns," in Proceedings of the sixth Australasian workshop on Grid computing and e-research-Volume 82, 2008, pp. 73-81.

[6] S. Naseera and K. M. Murthy, "Prediction Based Job Scheduling Strategy for a Volunteer Desktop Grid," in Advances in Computing, Communication, and Control, ed: Springer, 2013, pp. 25-38.

[7] D. Zhou and V. Lo, "Wave scheduler: Scheduling for faster turnaround time in peer-based desktop grid systems," in Job Scheduling Strategies for Parallel Processing, 2005, pp. 194-218.

[8] J. Zhang, "Flexible distributed computing with volunteered resources," 2010.

[9] P. Giemsch and A. Jellinghaus, "Optimization models for the containership stowage problem," in Operations Research Proceedings 2003, ed: Springer, 2004, pp. 347-354.

[10] M. Zeng, M. Y. H. Low, W. J. Hsu, S. Y. Huang, F. Liu, and C. A. Win, "Automated stowage planning for large containerships with improved safety and stability," in Proceedings of the Winter Simulation Conference, 2010, pp. 1976-1989.

[11] N. Ding, C. Benoit, G. Foggia, Y. Besanger, and F. Wurtz, "Neural Network-Based Model Design for Short-Term Load Forecast in Distribution Systems," 2015.

[12] D. C. Montgomery, C. L. Jennings, and M. Kulahci, Introduction to time series analysis and forecasting vol. 526: John Wiley & Sons, 2011.

[13] P. J. Brockwell, Introduction to time series and forecasting vol. 1: Taylor & Francis, 2002.

[14] C. Chatfield, "The holt-winters forecasting procedure," Applied Statistics, pp. 264-279, 1978.

[15] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@ home: an experiment in public-resource computing," Communications of the ACM, vol. 45, pp. 56-61, 2002.

[16] Z. Balaton, Z. Farkas, G. Gombas, P. Kacsuk, R. Lovas, A. C. Marosi, et al., "EDGeS: the common boundary between service and desktop grids," Parallel Processing Letters, vol. 18, pp. 433-445, 2008.

[17] J. Basney, M. Livny, and T. Tannenbaum, "Deploying a high throughput computing cluster," High performance cluster computing, vol. 1, pp. 356-361, 1999.

[18] S. Teng, W. Zhang, H. Zhu, X. Fu, J. Su, and B. Cui, "A Least-Laxity-First Scheduling Algorithm of Variable Time Slice for Periodic Tasks," Breakthroughs in Software Science and Computational Intelligence, p. 316, 2012.

[19] Y. Xu, H. Su, Y.-J. Pan, Z.-G. Wu, and W. Xu, "Stability analysis of networked control systems with round-robin scheduling and packet dropouts," Journal of the Franklin Institute, vol. 350, 2013.

[20] A. Sirohi, A. Pratap, and M. Aggarwal, "Improvised Round Robin (CPU) Scheduling Algorithm," International Journal of Computer Applications, vol. 99, pp. 40-43, 2014.