Journal of Computer Science

Original Research Paper

Static Batch Mode Heuristic Algorithm for Mapping Independent Tasks in Computational Grid

[1]R. Vijayalakshmi and [2]V. Vasudevan

[1]*Department of Computer Applications, Kalasalingam University, Krishnankoil, India*
[2]*Department of Information Technology, Kalasalingam University, Krishnankoil, India*

Article history
Received: 08-02-2014
Revised: 06-06-2014
Accepted: 09-08-2014

Corresponding Author:
R. Vijayalakshmi
Department of Computer Applications, Kalasalingam University, Krishnankoil, India
Email: grjviji@gmail.com

Abstract: Grid computing plays an important role in solving large-scale computational problems in a high performance computing environment. Scheduling of tasks to efficient and best suitable resource is one of the most challenging phase in grid computing systems. Grid environment reveals several challenges in efficient scheduling of complex applications because of its heterogeneity, dynamic behavior and shared resources. Scheduling of independent tasks in grid computing is dealt by a number of heuristic algorithms. This study proposes a new heuristic algorithm for mapping independent tasks in a grid environment to be assigned optimally among the available machines in a grid computing system. Due to the multi-objective nature of the grid scheduling problem, several performance measures and optimization criteria can be assumed to determine the quality of a given schedule. The metrics used here include makespan and resource utilization. This algorithm provides effective resource utilization by reducing machine idle time and minimizes makespan. This algorithm also balances load among the grid resources and produce high resource utilization with low computational complexity. The proposed algorithm is compared with other popular heuristics for performance measures.

Keywords: Grid Scheduling, Heuristics, Resource Utilization, Makespan, Load Balancing

Introduction

Grid computing allows to use remote resources in a high performance computing environment for solving large scale computational problems. Resources in a grid environment may be homogeneous or heterogeneous. Grid environment reveals several challenges in efficient scheduling of complex applications because of its heterogeneity, dynamic behavior and shared resources.

Grid scheduling can be classified as static scheduling and dynamic scheduling. In the case of static scheduling, information related to all resources and tasks in the Grid as application is assumed to be known earlier by the time the application is scheduled. By contrast, in the case of dynamic scheduling, the basic idea is to perform task allocation as the application executes. Both static and dynamic scheduling is widely adopted in Grid computing. One of the major benefits of the static model is that it is easier to program from a scheduler's point of view. Dynamic scheduling is more flexible than static scheduling. But it's hard to include load balance as metric to obtain stable and efficient scheduling algorithm.

Task scheduling is difficult in a heterogeneous environment and the problem of scheduling tasks to the resources is NP-Complete. NP-Complete problems can be solved by heuristic approach. Heuristic scheduling algorithms can be divided into two types: Immediate mode and Batch mode heuristics. A task is scheduled to the resource as soon as it comes to the scheduler in immediate mode. In batch mode scheduling, tasks are not mapped to the resources as they enter, but they are collected as a set of tasks that is examined for mapping at prescheduled times called

Science Publications

© 2015 R. Vijayalakshmi and V. Vasudevan. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

mapping events. Batch mode heuristics are called offline heuristics and immediate mode heuristics is called online heuristics. This research considers static batch mode scheduling.

Due to the multi-objective nature of the grid scheduling problem, several performance measures and optimization criteria can be assumed to determine the quality of a given schedule. The metrics used here include makespan, flowtime, load balance and resource utilization.

The idea of this research work is to device a new batch mode heuristic algorithm for mapping independent tasks with the intention of minimizing makespan, increasing throughput and maximizing average resource utilization rate with balanced load.

Related Study

Many scheduling algorithms are developed for mapping tasks to resources. Braun *et al*. (2001) stated a set of static heuristics for mapping a class of independent tasks onto heterogeneous systems which include Min-Min, Max-Min, MET, MCT and OLB.

Min-Min heuristic selects the task which has minimum execution time and maps the task to the machine that produces minimum completion time. Ready time of the resource is updated and the process continues until all the tasks are mapped. The heuristic complexity of the algorithm is $O(mn^2)$ where m is the number of machines and n is the number of independent tasks. Min-Min heuristics uses batch mode scheduling. It minimizes makespan but provides load unbalance and poor resource utilization as stated by (Alharbi, 2012).

Max-Min heuristic is similar to Min-Min but Max-Min maps larger tasks to the machines first. The ready time of the machine is updated and the process repeats until all the tasks are assigned. It takes $O(mn^2)$ to assign tasks to the machines. As reported by (Kamalam and Bhaskaran, 2010) it works better if the number of short tasks is more than long tasks.

Gaurav and Puneet (2013) have reported that Minimum Execution Time (MET) heuristic does not consider resource availability and assigns task using minimum execution time as metric. Resource which can execute the task in minimum time is scheduled. Thus MET causes load imbalance in grid resources. MET takes O(mn) time to assign the tasks where n denote the number of independent tasks and m denotes the number of machines. MET heuristic is categorized under immediate mode scheduling.

Minimum Completion Time (MCT) heuristic is also an immediate mode scheduling heuristic. It considers the task only one at a time. This heuristic searches the machine which has minimum completion time for a particular task as shown by (Hemamalini, 2012). It assigns the task to the machine based on completion time. The ready time of the resource and execution time of the task is summed to compute the completion time. It also takes O (mn) time for scheduling the tasks.

Alharbi (2012) has stated that OLB heuristic maps task in random order to the next available machine without considering the task's expected execution time on the resource. This heuristics balances load among resources but provides poor makespan. Its heuristic complexity is same as MET algorithm.

Elzeki *et al*. (2012) have stated that Sufferage heuristics first calculates the difference between first and second minimum completion time called sufferage. It then maps the task whose sufferage value is more to the machine which will execute it in minimum completion time. The mapped task is removed from unmapped list and the cycle continues till all the tasks are allocated. Its heuristic complexity is same as Max-Min.

Kokilavani and Amalarethinam (2011) proposed a new LBMM algorithm that runs Min-Min heuristics in the first round and then identifies the machine with heavy load by selecting the machine with makespan used by Min-Min. It produces better results than Min-Min heuristic by reducing makespan and balancing load when the tasks are smaller.

Materials and Methods

Problem Definition

Grid scheduling involves assigning of task to any one of the available resource for its complete processing. Tasks are processed without preemption on a resource one by one. Once started processing of a task cannot be stopped or postponed until completion as stated by (Chaturvedi and Sahu, 2011) This heuristic assumes static scheduling in which the capacity of each resource and load of each task in the grid environment is known earlier.

Let $M = \{M1, M2, ... Mn\}$ be the set of M resources that schedules a set of N tasks $T = \{T1, T2,....Tn\}$ in the computational grid. Assume that all the tasks T and resource M are available at the start when time t is 0. The processing time of a task depends on the length and speed of the task and the suitability of the resource for the particular task. This problem assumes that each machine uses FCFS scheduling for performing the received tasks.

As reported by Braun *et al.* (2001) an Expected Time to Compute ETC matrix of size nXm represent the expected processing time of n independent tasks on m machines. ETC matrix denotes the estimated execution time of a given task on each machine along the row and estimated execution time of a machine for each task along the column as reported by (Izakian *et al*., 2009).

The estimated execution time of T_i on M_j is defined by ETC (T_i, Mj). ETC model assumes that the computing capacity of each resource, computational needs of each task is known prior.

The expected execution time E_{ij} of task T_i on resource M_j is the amount of time to execute T_i on M_j when R_j has no load when T_i is mapped.

The expected completion time CT_{ij} of task T_i on resource M_j is the wall-clock time at which T_i is completed by M_j after finishing the previously assigned jobs. It is calculated by Equation 1. As stated by (Sunita and Chittaranjan, 2011):

$$CT\left(T_i, M_j\right) = MAT\left(M_j\right) + ETC\left(T_i, M_j\right) \quad (1)$$

Where:

MAT (M_j) = Machine availability time at which machine M_j completes any previously assigned tasks

ETC (T_i, M_j) = Time taken by a machine M_j to execute T_i when T_i is assigned and M_j is idle.

CT (T_i, M_j) = Overall expected completion time of T_i on M_j

Proposed Heuristic Algorithm

Our proposed algorithm TACT calculates the completion time of each task on the machines first. Then, the average completion time of each task is calculated. Step 6-Step 14, assigns tasks using Min-Min heuristic approach, since Min-Min heuristic attains low makespan and is known as benchmark heuristic. This heuristic calculates the minimum completion time of all unmapped tasks and selects the task which has minimum completion time and assigns that task to the machine. The assigned task is deleted from the unmapped set and the process continues until all tasks are mapped. Step 15 finds the makespan, which is the measure of throughput for heterogeneous systems such as computational grids. Step 18-28 reduces the makespan of machine and reschedules the task to balance the load for better resource utilization. This step reschedules the entire task by comparing the task average completion time with the makespan produced in the Min-Min schedule. If the task average completion time is greater than the makespan produced in Step 15, the tasks with the minimum completion time is assigned to the resource with minimum CT for the job. The process continues until all the tasks are mapped after deleting the assigned task from the task set.

Psuedocode for TACT Algorithm is given below:

Step1. for all tasks T_i in MT
Step2. for all machines M_j
Step3. Calculate completion time of task i on machine j, CT (T_i, M_j) = MAT (M_j) + ETC (T_i, M_j)
Step4. for all tasks N
Step5.Compute average completion time $ACT_i=\sum CT_{ij}/N$
Step6. do until all tasks in *MT* are mapped
Step7. for each task T_i in *MT*
Step8. for all machines M_j
Step9. Choose the task T_i with minimum CT_{ij} and re source that obtains it
Step10. Assign T_i to resource M_j that has minimum completion time
Step11. Delete assigned task T_i from MT
Step12. Update machine M_j availability time
Step13. Update completion time CT_{ij} of all unmapped tasks
Step14. end do
Step15. for all machines M_j
Step16. Calculate makespan = max (CT_{ij})
Step17. end for
Step18. do until all tasks T_i in MT are mapped
Step19. Search the task T_i with maximum ACT_i
Step20. if maximum ACT_i>makespan
Step21. Search the machine M_j with minimum CT_{ij} for the task *Ti*
Step22. end if
Step23. Assign T_i to resource M_j with minimum CT_{ij} for the task *Ti*
Step24. Delete assigned task T_i from *MT*
Step25. Update machine M_j availability time
Step26. Update completion time CT_{ij} of all unmapped tasks
Step27. Update ACT_i
Step28. end do

Illustrative Example

Consider a grid environment with three resources and three tasks. The ETC matrix for the grid system is defined in the Table 1.

Table 1. ETC matrix (3 tasks and 3 machines)

Tasks/machines	M1	M2	M3
T1	50	20	15
T2	20	60	15
T3	20	50	15

Min-Min heuristics assigns tasks T1 and T3 to resource R3 and task T2 to R1 achieving a makespan of 30 time units. Similarly Max-Min heuristics also assigns tasks T1 and T3 to resource R3 and task T2 to R1 with a makespan of 30. MET algorithm produces a makespan of 45 by assigning all the tasks T1, T2 and T3 to R3. MCT assigns T2 to R1 and T1 and T3 to R3 with a makespan of 30 time units.

The Sufferage heuristic, produces similar result as that of Max-Min, Min-Min and MCT and achieves a 30 time units makespan. The makespan of LBMM algorithm is also 30 with T2 assigned to R1 and T1 and T3 mapped to R3. All the discussed heuristic assigns tasks to resources R1 and R3 and left R2 unassigned with any task. However the proposed heuristic is able to achieve a makespan of 20 time units by mapping T1 to R2, T2 to R3 and T3 to R1. Figure 1 illustrates the mapping of tasks to resources using TACT heuristic.

Experimental Results

Programs for existing and proposed heuristics are implemented in C++ language. Programs define schedule for assigning tasks to available machines and calculates makespan, resource utilization, average resource utilization, flow time and fitness value based on the ETC matrix supplied to it. This section shows the actual results after executing the code for the given example. Results obtained are discussed as follows. Better results are produced in terms of makespan, resource utilization and fitness value compared to other algorithms. Minimization of makespan and greater average resource utilization rate is achieved by the proposed TACT scheduling algorithm.

Makespan

Different heuristics algorithms performance based on makespan is shown in the Fig. 2. Makespan is the time a heuristic takes to finish a batch of jobs. It is a measure of efficiency and throughput of a grid computing system as stated by (Saeed and Entezari-Maleki, 2009). The following graph illustrates that TACT reduces makespan compared to other heuristics. Makespan is evaluated by Equation 2 as follows:

$$Makespan = max\ (CT(T_i, M_j) \tag{2}$$

Resource Utilization

The main objective of grid computing systems is to maximize resource utilization. This metric improves the utilization of resources by minimizing the idle time of resources. It is defined as the percentage of time that resource R_j is busy during the scheduling process as stated by (Rafsanjani and Bardsiri, 2012). It is calculated using the formula in Equation 3.

$$RU_j = MAT(R_j)\ for\ j = 1,2,3,...N \tag{3}$$

Equation 4 computes average resource utilization as follows:

$$ARU = \sum_{j=1}^{N} \frac{RU_j}{N} \tag{4}$$

The following Fig. 3 illustrates the utilization of resources by different heuristics. All the heuristics except OLB left any one of the resources idle; whereas OLB produces an unbalanced resource utilization by allocating low load to R3. TACT produces an optimum utilization compared to all other algorithms by using all the machines without keeping the resources idle.

Average resource utilization produced by Min-Min, Max-Min, Sufferage, LBMM and MCT is 55.5%. OLB utilizes 69.4% and MCT produces a low utilization rate of 33.3%. TACT overcomes all algorithms by having an average resource utilization of 91.6 by using R1, R2 to cent percentage and R3 with load 75%.

Figure 4 shows the average resource utilization for all the heuristics. TACT has the best utilization rate among all the other algorithms.

Flow Time

Flow time is used to measure the QoS of the grid systems. It is defined as the sum of finishing time of all the tasks as reported by (Chaturvedi and Sahu, 2011). Flow time calculated for all the heuristics is shown in Fig. 5. Flow time is minimum when tasks are processed in increasing order of processing time on a machine. Equation 5 is used to calculate flowtime:

$$Flowting = \sum_{i=1}^{M} E_{ij} \tag{5}$$

Fitness

Fitness metric is used to calculate the performance of the scheduling algorithm to optimize makespan and flowtime. It is computed using the following relation as stated by (Alharbi, 2012).

$$Fitness = p * makespan + (1-p) * \frac{Flowtime}{N}$$

where, p ranges from 0 to 1 based on the importance of the metric and N denotes the number of machines. This study assumes 0.5 for experimental evaluation. Figure 6 shows the comparision results of fitness value for different heuristics.

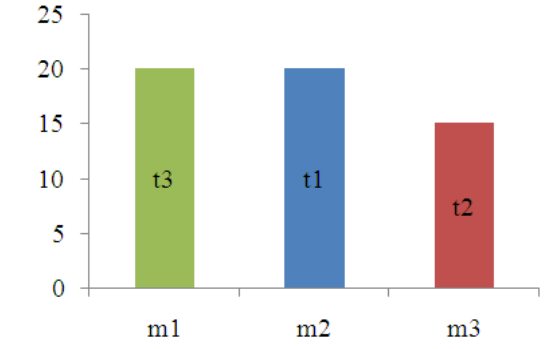

Fig. 1. Gantt chart for ACT heuristic

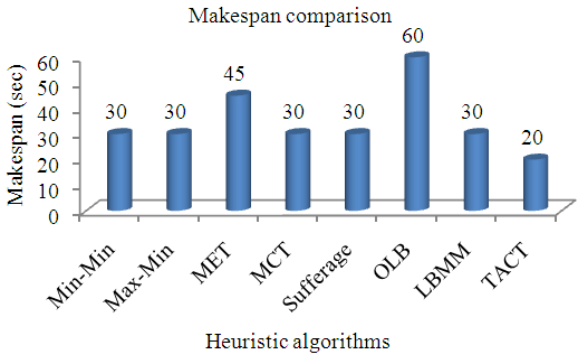

Fig. 2. Comparison of makespan among heuristics

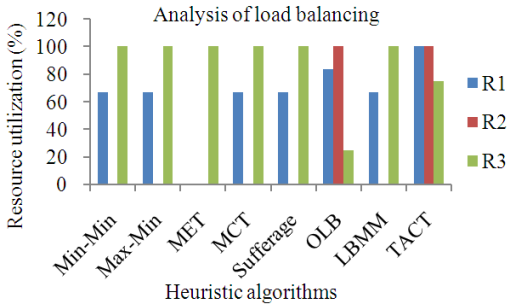

Fig. 3. Comparison of resource utilization among heuristics

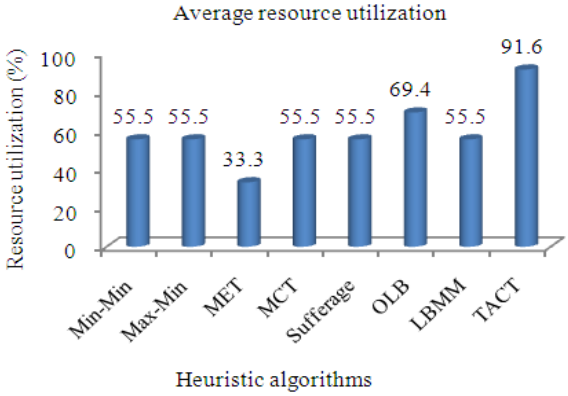

Fig. 4. Comparison of average resource utilization

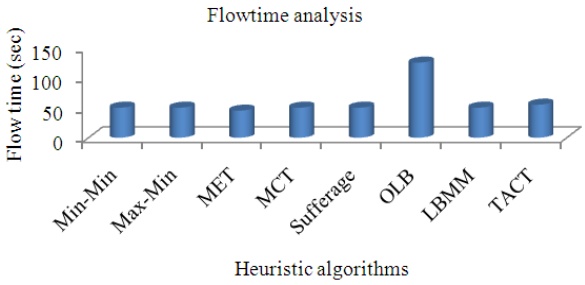

Fig. 5. Comparison of flowtime among heuristics

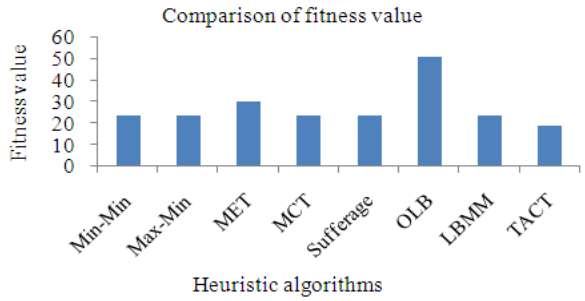

Fig. 6. Comparison of fitness value among heuristics

Conclusion

Since scheduling in grid computing environment is an NP-Complete problem, heuristic algorithms is a suitable method to cope with its solution. This study presents a new heuristic with the aim of reducing makespan, increasing throughput and resource utilization. Experimental results show that the proposed heuristic scheduling algorithm performs better than the existing heuristics and provides improved makespan and resource utilization. Considering QoS factors, communication delay and CPU workload are a part of future research.

Funding Information

The authors have no support or funding to report.

Author's Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Alharbi, F., 2012. Multi objectives heuristic Algorithm for Grid Computing. Int. J. Comput. Applic., 46: 39-45.

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni and M. Maheswaran *et al*., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distributed Comput., 61: 810-837. DOI: 10.1006/jpdc.2000.1714

Chaturvedi, A.K. and R. Sahu, 2011. New heuristic for scheduling of independent tasks in computational grid. Int. J. Grid Distributed Comput., 4: 25-36.

Elzeki, O. M., M.Z. Rashad and M.A. Elsoud, 2012. Overview of scheduling tasks in distributed computing systems. Int. J. Soft Comput. Eng., 2: 470-475.

Gaurav, S. and B. Puneet, 2013. Task aware switcher scheduling for batch mode mapping in computational grid environment. Int. J. Adv. Res. Comput. Sci. Softw. Eng., 3: 1292-1299.

Hemamalini, M., 2012. Review on grid task scheduling in distributed heterogeneous environment. Int. J. Comput. Applic., 40: 24-30.

Izakian, H., R.A. Abraham and V. Snasel, 2009. Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. Proceedings of the International Joint Conference on Computational Sciences and Optimization, Apr. 24-26, IEEE Xplore Press, Sanya, Hainan, pp: 8-12. DOI: 10.1109/CSO.2009.487

Kamalam, G.K. and V.M. Bhaskaran, 2010. An improved min-mean heuristic scheduling algorithm for mapping independent tasks on heterogeneous computing environment. Int. J. Computational Cognition, 8: 85-90.

Kokilavani, T. and D.I.G. Amalarethinam, 2011. Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. Int. J. Comput., 20: 43-49.

Rafsanjani, M.K. and A.K. Bardsiri, 2012. A new heuristic approach for scheduling independent tasks on heterogeneous computing systems. Int. J. Machine Learn. Comput., 2: 371-376. DOI: 10.7763/IJMLC.2012.V2.147

Saeed, P. and R. Entezari-Maleki, 2009. RASA: A new task scheduling algorithm in grid environment. World Applied Sci. J. 7 152-160.

Sunita, B. and H. Chittaranjan, 2011. Efficient refinery scheduling heuristic in heterogeneous computing systems. J. Adv. Inform. Technol., 2: 159-164. DOI: 10.4304/jait.2.3.159-164