International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A Review Paper of Improving Task Division Assignment Using Heuristics

Shripadrao Biradar[1], Deepika Pawar[2]

[1]Professor, Department of Computer Engineering, RMD Sinhagad School of Engineering, University of Pune, India

[2]Department of Computer Engineering, RMD Sinhagad School of Engineering, University of Pune, India

Abstract: ***This paper presents an innovative idea of distributing the tasks to their best processor to reduce the execution time of task by using various scheduling techniques. This paper presents hybrid scheduling techniques which provide better solution of scheduling task that means combination of different scheduling provides better performance without degrading the result quality. Scheduling algorithms such as MinMin+, MaxMin+ and Sufferage+ are suitable for overcomes the drawback of previously used scheduling methods such as MinMin, MaxMin and Sufferage as well as scheduling in this paper provides better complexity as compare to previous scheduling methods. This scheduling are also suitable for heterogeneous environment more effectively to execute different set of task on different processors with different configurations.***

Keywords: Task Scheduling, MinMin, MaxMin, Sufferage, Standard Deviation, Load Balancing.

1. Introduction

Distribution of large application into task for faster processing is one of the important process in the area of distributed systems. Although many types of resources can be shared and used in a distributed system, usually they are accessed through an application running in the network. Normally, an application is used to define the piece of work of higher level in the system. An application can generate several tasks, which in turn can be composed of subtasks; this system is responsible for sending each subtask to a resource to be solved. It performs an important step of mappings task to different machines based on the expected execution time. Normally an application is used to define the piece of work of higher level in heterogeneous environment.

Since this application can generate several number of jobs that can be divided into subtasks and provided to different processors that should get completed within minimum time so that the processor use can be made to assign different task. Makespan is one of the most important term in case of mapping task to their processors using different scheduling. Makespan is nothing but turnaround time that is maximum of completion time. An optimal schedule will be the one that minimizes the makespan [1, 2].

Large numbers of task scheduling algorithms are available to minimize the makespan. All these algorithms try to find resources to be allocated to the tasks which will minimize the overall completion time of the jobs. Minimizing overall completion time of the tasks does not mean that it minimizes the actual execution time of individual task. The simple well-known existing algorithms used for scheduling are Min-Min and Max- min and sufferage. These algorithms work by considering the execution and completion time of each task on the each available grid resource. Scheduling is considered to be an important issue in the current distributed system scenario. The demand for effective scheduling increases to achieve high performance computing. Typically, it is difficult to find an optimal resource allocation which minimizes the schedule length of jobs and effectively utilize the resources. The three main phases of scheduling are resource discovery, gathering resource information and job execution. The choice of the best pair of jobs and resources in the second phase has been proved to be NP- complete problem.

The existing scheduling algorithms provide the various techniques for assigning different task to different resources with minimum completion time. These existing scheduling algorithms can be divided into two classes that are online mode and Batch mode scheduling. In online mode, a task is assigned to processor as soon as it arrives at the scheduler [1]. Wherein Batch mode scheduling tasks are not assigned to processor immediately instead they are collected in to set of tasks also called as Metatarsi that are examined for assigning at prescheduled times to different processors also called as mapping events. Since in this system, batch mode is used in very efficient way for mapping different independent tasks to processors. Also these existing algorithms can be applied for heterogeneous environment effectively.

The proposed system in this work contains various scheduling methods along with hybrid technology such as Minmin+, Maxmin+, and Sufferage+. Hybrid technology involves combination of different scheduling methods to overcome disadvantages of minmin and maxmin. Overall, the scheduling algorithms aim to minimize the idle time and makespan of tasks. This paper also involves the concept of Load Balancing [2, 6], wherein, once scheduling of task is done using some scheduling the load balancing algorithm will take place to reschedule the task to utilize all the resources in the heterogeneous environment [5]. Each of this scheduling provides better performance and also decreases time complexity without degrading the solution quality.

To avoid the drawbacks of the existing scheduling algorithm, the proposed system algorithms are being used to enhance the system performance. All the problems discussed in those methods are taken and analyzed to give a more effective schedule. The algorithm proposed in this paper

outperforms all those algorithms both in terms of makespan and load balancing. Thus a better load balancing is achieved and the total response time of the system is improved. The proposed algorithm applies the Min-Min strategy in the first phase and then reschedules by considering the maximum execution time that is less than the makespan obtained from the first phase.

2. Literature Survey

A distributed scheduling algorithm aims to increase the utilization of resources with light load or idle resources thereby freeing the resources with heavy load. The algorithm tries to distribute the load among all the available resources. At the same time, it aims to minimize the makespan with the effective utilization of resources. In classical distributed systems comprised of homogeneous and dedicated resources, scheduling

Algorithms have been intensively studied. But these algorithms will not work well in Grid architecture because of its heterogeneity, scalability and autonomy. This makes load balanced scheduling algorithm for grid computing more difficult and an interesting topic for many researchers. The Non-traditional algorithms differ from the conventional traditional algorithms in that it produces optimal results in a short period of time. There is no best scheduling algorithm for all grid computing systems. An alternative is to select an appropriate scheduling algorithm to use in a given heterogeneous environment because of the characteristics of the tasks, machines and network heterogeneity resources. These scheduling algorithms will work well even for heterogeneous resources also. Heterogeneous systems provides with the facility of utilization of all available resources as load balancing concept that aims at keeping resources busy [5].

Opportunistic Load Balancing (OLB) is specially used to keep all the processors busy that is to make utilization of all the available resources by assigning task in arbitrary order, to the next available processor, without considering task expected execution time on that particular processor but this result in poor makespan [6].

Minimum Execution Time (MET) assigns tasks to processor in arbitrary order with best expected execution time for that task, without taking into consideration processors availability. Since assigning the task to its best processor provides better performance but causes severe load misbalancing and does not provide support for heterogeneous environment [4, 5].

Minimum Completion Time (MCT) assigns tasks to different processors in arbitrary order, with minimum expected completion time for that task. Since this causes some of the task to be assigned to the processor that do not have minimum execution time. For this purpose this minimum completion time is defined in a way that combines the benefit of both opportunistic load balancing (OLB) and minimum execution time (MET) to provide better performance of task mapping [3].

Min-Min algorithm starts with a set of all unmapped tasks. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall minimum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. Compared to MCT this algorithm considers all jobs at a time. So it produces a better makespan. When selected task is assigned to the resource it is removed from the met task that is set of task. Since this method provides easiest way to assign task to processor but has one drawback due to selection of task having minimum expected completion time, the task with largest expected completion time remains unassigned for longer time and also load is not balanced across the systems, due to which some resources remains idle and this also results in increase in makespan.

Max-Min is similar to Min-Min algorithm. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall maximum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. The idea of this algorithm is to reduce the wait time of the large jobs. Since this scheduling algorithm provides the way of mapping task to its best machine with longer execution time first allows this task to be executed concurrently with the task that having shorter execution time. Since this mapping of task to resources is better than minmin scheduling wherein the task with smaller execution time is selected and assigned to the available resource for execution and then task with longer execution time are executed while several machines sit idle. Since maxmin scheduling provides better load balancing across machines as well as better makespan [6, 7].

Sufferage scheduling differs with previous scheduling algorithms in the sense of task selection process. Like minmin and maxmin it also begins with set of unassigned task that has minimum completion time i.e. sufferage scheduling is also based on the concept of minimum completion time, since it differ from the previous scheduling in the sense it selects and assigns the task to the processor on the basis of sufferage value and not minimum or maximum completion time. Since it computes second MCT value instead of computing MCT value for each task and calculates sufferage value which is defined as difference between MCT and second MCT values of a task is taken into account. This scheduling selects the task with largest sufferage value and assigns it to available resource. Thus sufferage scheduling differs from minmin and maxmin scheduling in the task selection policy [4, 5].

3. Enhanced Scheduling Methods

In this work, propose system contains novel algorithms, such as MinMin+, MaxMin+ and Sufferage+.

In these scheduling algorithms, the MCT values that are associated with each processor are separately maintained, instead of being unnecessarily recomputed at each iteration for every unassigned task. In particular, a priority queue $Q_{k\ is}$ being used for each processor $P_{k\ to}$ maintain the completion times of all tasks on that processor. This type of scheduling

algorithms provides better performance than the previous scheduling methods without degrading solution quality. That is previously mentioned scheduling methods increases time complexity by using number of iterations for computing minimum completion time of each task.

A. Minmin+ scheduling

Minmin+ scheduling uses different methods such as MinMin+Select function invokes a MIN (Q_k) operation on each priority queue Qk to find a candidate task for processor Pk. The candidate task Ti selected for processor Pk is effectively the task that will increase the current completion time of P_k (i.e., e_k) by the smallest amount if Ti is assigned to Pk. For each processor P_k, the execution time of the candidate task Ti on P_k is added to e_k to compute the updated e_k value for P_k if Ti is assigned to P_k. A running-min operation performed over these K updated e_k values gives the minimum MCT value (min) for the current iteration as well as the task-to-processor assignment (i', k') that achieves this minimum MCT value. At the end of each iteration of the main loop, the assigned task $T_{i'}$ is deleted from all priority queues. For the implementing this priority queue two alternatives are being considered that are binary heap and sorted linear array, and also some operations are being used like sorting operation, deletion operation, and necessary check is made on the queue to know which task has not being yet assigned to available resource that is with minimum completion time. Hence the overall running time complexity is reduced.

B. Maxmin+ scheduling

The solution quality obtained in the earlier iterations is likely to deteriorate due to the late assignment of very large tasks. In case of the MaxMin scheduling, the larger tasks are assigned in earlier iterations, but not necessarily to their favorite processors. Since, in the first few iterations of MaxMin, the first iteration assigns the largest task to its favorite processor. It is assumed that the second largest task has the same favorite processor as the largest task. In the second iteration, the task selection policy of MaxMin prevents the assignment of the second largest task to its favorite processor. In the next iteration, the third largest task loses the flexibility of being assigned to the favorite processors of the largest two tasks and so on. To alleviate the above-mentioned drawbacks of the MinMin and MaxMin scheduling, these scheduling algorithms are combined under a hybrid scheduling, which referred to as MaxMin+. Like MinMin and MaxMin, the MaxMin+ scheduling involves a main loop that assigns a selected task to a processor at each iteration. Within an iteration, the scheduling first computes a task- to-processor assignment according to the MinMin scheduling. The computed assignment is realized only if it does not lead to an increase in the makespan of the previous iteration. If, however, the computed assignment increases the makespan, the task-to-processor assignment is recomputed according to the MaxMin scheduling. This scheduling overcome drawback of maxmin scheduling of task assignment problem to same processor by doing the combination of maxmin with minmin+ under a hybrid scheduling that is maxmin+.

C. Sufferage+ scheduling

The main idea behind the Sufferage+ scheduling is to perform critical assignment decisions by Sufferage so that the solution quality is not significantly degraded and perform non-critical assignment decisions by the fast MinMin+ algorithm. With this approach, it is expected a considerable decrease in the execution time of Sufferage with a small potential degradation in the solution quality. Since Sufferage+ working is similar to sufferage scheduling since to make applicable sufferage scheduling to large datasets it is combine with minmin+ scheduling under a new scheduling that is sufferage+. As in MaxMin+, in this scheduling also the MinMin+Init function performs the necessary initializations. It computes the assignment according to MinMin+. The comparison operation checks whether makespan will change if the computed assignment is used. Then this algorithm computes the task-to-processor assignment according to Sufferage. This scheduling differs from previous scheduling in the sense that when assignment is computed, it is realized only if it does not, lead to increase in makespan of previous iteration otherwise assignment is recomputed using sufferage scheduling.

D. Switcher Scheduling

As the name indicates it is the combination of different scheduling also known as hybrid scheduling. Switcher scheduling is based on concept of standard deviation value comparison with threshold value. Based on this it switches between scheduling that is if standard deviation [6] value is less than threshold value than tasks are considered to be with minimum execution time and minmin scheduling is applied to assign the tasks to available resources, otherwise maxmin scheduling is used to assign the tasks to available resources. This process is repeated until all the tasks are assigned to their respective available resources [3, 13].

Standard deviation concept is specially used for hybrid scheduling that is combination of different scheduling. Wherein, the standard deviation value is compared with threshold value to check which scheduling to be applied for mapping of task to different resources [6]. Since the standard deviation value is calculated on the basis of average of completion time of all tasks, as mention below:

$$avgCT = \frac{\sum_{i-1}^{s} CT_{ij}}{s}$$

Where avgCT denotes average of completion time that is sum of all completion time of given tasks and s is nothing but index of task. Using this average value standard deviation is calculated as:

$$sd = \sqrt{\frac{\sum_{i-1}^{s}(CT_{ij} - avgCT)^2}{s}}$$

Based on above mentioned formulae standard deviation is calculated. Since, this is compared with the threshold value in case of hybrid scheduling wherein the multiple scheduling are called by algorithm alternatively for mapping task to processors. This hybrid scheduling will use standard deviation concept wherein if the calculated standard deviation is less than threshold value then that particular task is assigned using the minmin scheduling to available

resource otherwise the task is assigned to available resource using maxmin scheduling. Since after the assignment of task to resource it will be deleted from set of task that is metatask and the hybrid scheduling will repeat all the process until all the task are assigned to processors.

There are many different types of hybrid algorithms that call alternatively different scheduling for mapping tasks to their best processors. This type of scheduling also maintains the proper load balance across the processors due to which all the available resources get fully utilized and no resource remains an idle.

4. Balancing Load In Distributed Systems

To better utilize the machines, the load should be balanced to minimize the machine idle time. To balance the load, the sizes of tasks must be taken into consideration. In a heterogeneous system, execution time of a task varies on different machines. The task size can simply be measured as the average of the execution times over all machines. Tasks can be assigned with priorities based on their sizes, either favorable to the smaller tasks or to the larger tasks. Or the priority can be assigned independently of task sizes. It has been found that the load is severely imbalanced when smaller tasks are mapped first. If large tasks are given higher priorities, it generally leads to a better balanced load. Also, a priority independent of tasks sizes has a fairly good chance for load balancing. Scheduling mentioned above like minmin and maxmin maps different tasks to different available resources efficiently but it does not maintain proper load balancing among the resources due to which some resources are utilized and some remain idle. This load balancing concept can be applied to this scheduling to get done execution faster [6].

Minmin scheduling selects tasks with minimum completion time and allocates it to available resource, due to which task with longer execution time remains unassigned although the resource is available that causes resources to remain idle. Similarly in maxmin scheduling tasks with maximum completion time is selected and assigned to processor, due to which smaller tasks are assigned after long time to available processors [7].

Solution for above is to apply load balancing concept with this types of scheduling. This can be done when tasks are assigned to their resources that is once minmin scheduling is applied to assign tasks to available resources using minimum completion time, the load balancing method is applied again on this assigned task for rescheduling it i.e. it may happen that minmin scheduling will use some resources to assign task and some remain idle then load balancing method will select the task with
maximum completion time that will be less than makespan produced by Minmin scheduling and reschedule it to the resource that is available and not utilized yet so that execution of tasks will be more faster [11]. Other tasks maximum completion time is not less than makespan. So whichever task has maximum completion time less tan makespan is selected and rescheduled to available resource.

5. Example Of Balancing Load in Distributed Systems

Consider a heterogeneous environment with two resources R_1 and R_2 and metatask that contain four different tasks T_1, T_2, T_3 and T_4 as shown below in table1 that contains tasks, resources along with expected execution time for mapping tasks to their respective resources.

Table 1: Resources and Tasks with Expected Execution Time

Resources		*Tasks*
R_2	R_1	
2	7	T_1
4	13	T_2
1	14	T_3
3	11	T_4

As shown in above table task assignment is done to different processors using minmin scheduling is done in following way.

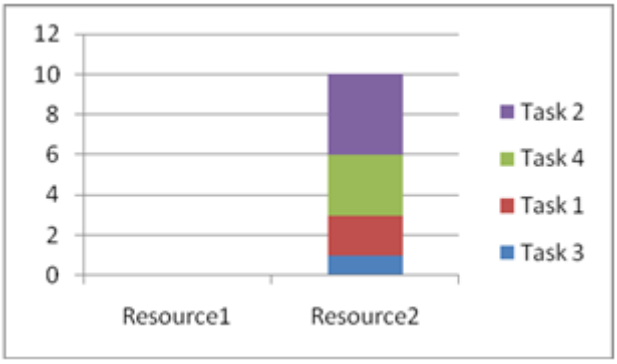

Figure 1: Mapping of Tasks to Resources with Minmin Algorithm

As shown in figure1 minmin scheduling will select tasks according to given execution time so task T_3 will be assigned first to Resource R_2, then task T_1 will be assigned again on resource R_2, then task T_4 will be assigned again on resource R_2 and finally the remaining task that is task T_2 will also be assigned to resource R_2 only according to given expected execution time in Table1. Since on resource R_2 task completion is faster than on resource R_1, so all the task will be assigned on resource R_2 only.

Once minmin scheduling is applied for mapping tasks to available resources, load balancing technique is applied for again rescheduling tasks to make utilization of idle resources that minimizes overall task completion time that is makespan.

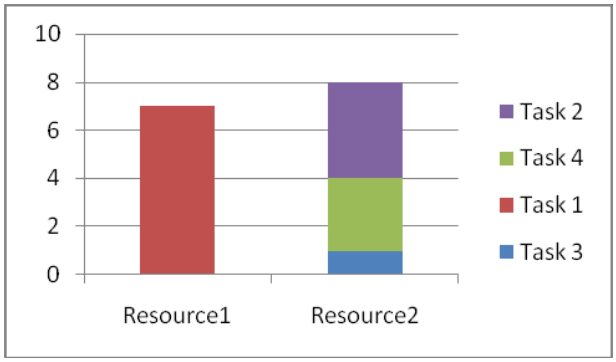

Figure 2: Rescheduling of tasks to Resources with Load Balancing method

As shown in figure2 task T_1 is rescheduled to balance the load as it provides maximum completion time on resource R_1 as shown in Table1 as well as it is less than makespan produced by minmin scheduling. While remaining tasks although have maximum completion time but are not less than makespan. So task T_1 is rescheduled on resource R_1 that results in better makespan as compared to minmin scheduling.

6. Conclusion

The aim of this paper was to present various scheduling methods like minmin, maxmin, sufferage, hybrid, load balancing techniques in the field of distributed systems. The scheduling like minmin and maxmin are suitable for small scale distributed systems but when number of tasks increases than these scheduling cannot schedule task appropriately that affects on makespan which relatively become large. To overcome limitations of these scheduling and make them applicable for large scale distributed systems, a new task scheduling algorithm like minmin+, maxmin+ and sufferage+ along with hybrid scheduling are used that also maintains proper load balancing across the systems. This scheduling uses advantages of minmin and maxmin and covers there disadvantages. This study can be further extended by considering task heterogeneity and machine heterogeneity.

References

[1] T. D. Braun,H. J. Siegel,N. Beck, L. L. Boloni, “A comparison of eleven static scheduling for mapping a class of independent tasks onto heterogeneous distributed computing systems”, J. Parallel Distrib. Comput., vol. 61, no. 6, pp. 810837, 2001.
[2] P. Luo, K. Lu, and Z. Shi, “A revisit of fast greedy scheduling for mapping a class of independent tasks onto heterogeneous computing systems”, J. Parallel Distrib. Comput., vol. 67, pp. 695714, 2007.
[3] Kamali Gupta, Manpreet Singh, “Scheduling Based Task Scheduling In Grid”, International Journal of Engineering and Technology (IJET), vol. 4, pp. 254260, Aug-Sep 2012.
[4] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, “Dynamic mapping of a class of independent tasks onto heterogeneous computing systems”, J. Parallel Distrib. Comput., vol. 59, pp.107131, 1999.
[5] H. J. Siegel and S. Ali, “Techniques for mapping tasks to machines in heterogeneous computing systems”, J. Syst. Archit., vol. 46, no. 8, pp. 627639, 2000.
[6] T. Kokilavani, Dr. D.I. George Amalarethinam, “Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing”, International Journal of Computer Applications, vol. 20, April 2011.
[7] George Amalarethinam. D.I, Vaaheedha Kfatheen .S, “Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems”, International Journal of Computer Science and Information Technologies, Vol. 3, pp. 3659-3663, 2012.
[8] K. Kaya, B. Ucar, and C. Aykanat, “Scheduling for scheduling file-sharing tasks on heterogeneous systems with distributed repositories”, J. Parallel Distrib. Comput., vol. 67, no. 3, pp. 271285, 2007.
[9] Doreen Hephzibah Miriam. D and Easwarakumar. K.S, “A Double MinMin Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010.
[10] Kamalam.G.K and Muralibhaskaran.V, “A New Scheduling Approach: MinMean Algorithm For Scheduling Meta-Tasks On Heterogenous Computing Systems”, International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010.

Author Profile

S.S. Biradar received the B.E. degree in Computer Science & Engineering from PDACOE Gulbarga Karnataka & M.E. degree in DC&N from Dr. AIT Bangalore Karnataka in 2010 & 2012 , respectively. Currently he is working as Assistant Professor of Computer Engineering Department in RMD SSOE Pune, India.

Deepika R. Pawar Research Scholar RMD Sinhagad School of Engineering Warje, Pune, University of Pune. She received B.E. in Computer Engineering from Cummins College of Engineering, Karvenagar, Pune from PU. Currently she is pursuing M.E. in computer engineering from RMD Sinhagad School of Engineering Warje, Pune University of Pune.