ISSN:0975-9646

Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems

George Amalarethinam. D.I, Vaaheedha Kfatheen .S

Dept of Comp. Sci & IT,Jamal Mohamed College(Autonomous), Trichirappalli, TN, India.

***Abstract*—The Grid computing provides the opportunity to use the remote resources in the network for resolving the large scale tasks. These large tasks can be either scientific or technology related tasks, which require large amount of processing time since they handle large amount of data. The major objective of grid computing is to decompose the larger tasks into smaller tasks, which can be either dependent or independent tasks. Mostly in the grid computing systems the resources can be of two types. They are homogenous system and heterogeneous system. The major problem of task scheduling or job scheduling comes when the resources are of different types. That is, the heterogeneous system, this problem of mapping the task to the resources is called the NP-Complete. The previous experiments and the results have proved that NP-Problem can be best solved by the heuristic approach rather than other approaches. In this paper we have proposed a new heuristic technique called Max-min Average algorithm for task scheduling in the heterogeneous grid computing environment. Usually the performance of the grid computing is measured by the reduction in the idle time and makespan. The proposed Max-min Average algorithm proves that it is efficient than the other heuristic algorithms by reducing the idle time and make span than the other algorithms.**

***Keywords*— Grid computing, Task Scheduling, NP-Problem, Heuristic Algorithm, Load Balancing.**

I. INTRODUCTION

Although many types of resources can be shared and used in a Grid system, usually they are accessed through an application running in the Grid. Normally, an application is used to define the piece of work of higher level in the Grid. An application can generate several jobs, which in turn can be composed of subtasks; the Grid system is responsible for sending each subtask to a resource to be solved.

Scheduling is a process that maps and manages execution of independent tasks on distributed resources. Mapping tasks to machines in an HC Suite is an NP-Complete problem and, therefore the use of heuristics is one of the suitable approaches. Existing scheduling heuristics can be divided into two classes: On-line mode and Batch-mode heuristics. In the on-line mode, a task is mapped onto a host as soon as it arrives at the scheduler. In the batch mode, tasks are not mapped on to hosts immediately and they are collected in to a set of tasks that is examined for mapping at prescheduled times called mapping events. In this paper, we considered batch-mode heuristics. Different criteria can be used for evaluating the efficiency of scheduling algorithms, the most important of which are makespan and flowtime. Makespan is the time when Heterogeneous Computing system finishes the latest task. An optimal schedule will be the one that minimizes the makespan[1].

In this paper, we proposed an efficient heuristic Max-min average algorithm by taking the mean of the completion time of all tasks. Our algorithm aims to minimize the idle time and makespan of the tasks.

The rest of the paper is organized as follows. In Section 2 describes existing heuristics and meta-heuristics based workflow scheduling techniques on distributed systems such as Grid. Section 3 specifies the problem description. The proposed Max-min Average algorithm is presented in Section 4. Experiment results are presented in Section 5. Finally, this paper concludes with the direction for future work in Section 6.

II. LITERATURE SURVEY

A set of static heuristic for task scheduling in heterogeneous computing environments are available. A range of simple greedy constructions heuristic approaches are compared and some of them are briefly described below:-

OLB: Opportunistic Load Balancing (OLB) assigns each task, in arbitrary order, to the next machine that is expected to be available, regardless of the task's expected execution time on that machine [3],[4],[5]. The intuition behind OLB is to keep all machines as busy as possible. One advantage of OLB is its simplicity, because OLB does not consider expected task execution times, the mappings it finds can result in very poor makespans.

MET: In contrast to OLB, Minimum Execution Time (MET) assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability[3],[4]. The motivation behind MET is to give each task to its best machine. This can cause a severe load imbalance across machines. In general, this heuristic is obviously not applicable to HC environments characterized by consistent ETC matrices.

MCT: Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task [3]. This causes some of the tasks to be assigned to the machines that do not have the minimum execution time for them. The intuition behind MCT is to combine the benefits of OLB and MET, while avoiding the circumstances in which OLB and MET perform poorly.

Min-min: The Min-min heuristic begins with the set U of all unmapped tasks. Then, the set of minimum completion times, $M=\{\min_{0\leq j<\mu}(ct(t_i,m_j))$, for each t_i # U$\}$, is found. Next, the task with the overall minimum completion time from M is selected and assigned to the corresponding machine (hence the name Min-Min).

At last, the newly mapped task is removed from U, and the process repeats until all the tasks are mapped (i.e., U is empty)[3],[4],[6]. Min-min is based on the minimum completion time, as is MCT. However, Min-min considers all the unmapped tasks during each mapping decision and MCT only considers one task at a time. Min-min maps the tasks in the order that changes the machine availability status by the least amount that any assignment could. Let t_i be the first task mapped by Min-min onto an empty system. The machine that finishes t_i the earliest, say m_j, is also the machine that executes t_i the fastest. For every task that Min-min maps after t_i, the Min-min heuristic changes the availability status of m_j by the least possible amount for every assignment. Therefore, the percentage of tasks assigned to their first choice (on the basis of execution time) is likely to be higher for Min-min than for Max-min (defined next). The expectation is that a smaller makespan can be obtained if more tasks are assigned to the machines that complete them the earliest and also execute them the fastest.

Max-min: The Max-min heuristic is very similar to Min-min. The Max-min heuristic also begins with the set U of all unmapped tasks. Then, the set of minimum completion times, M, is found. Next, the task with the overall maximum completion time from M is selected and assigned to the corresponding machine (hence the name Max-min). At last, the newly mapped task is removed from U, and the process repeats until all the tasks are mapped (i.e., U is empty)[3],[4],[6]. Intuitively, Max-min attempts to minimize the penalties incurred from performing tasks with longer execution times**.** For example, assume that the metatask being mapped has many tasks with very short execution times and one task with a very long execution time. Mapping the task with the longer execution time to its best machine first allows this task to be executed concurrently with the remaining tasks (with shorter execution times). For this case, this would be a better mapping than a Min-min mapping, where all of the shorter tasks would execute first, and then the longer running task would execute while several machines sit idle. Thus, in cases similar to this example, the Max-min heuristic may give a mapping with a more balanced load across machines and a better makespan.

Duplex: The Duplex heuristic is literally a combination of the Min-min and Max-Min heuristics. The Duplex heuristic performs both of the Min-min and Max-min heuristics and then uses the better solution [3],[4]. Duplex can be performed to exploit the conditions in which either Min-min or Max-min performs well, with negligible overhead.

GA : The Genetic algorithm (GA) is a technique used for searching large solution spaces[7],[8],[9],[10],[11]. The GA operates on a population of chromosomes for a given meta-tasks. The initial population is generated by two methods. In the first method, a chromosome is generated randomly from a uniform distribution. In the second method, a chromosome is generated by Min-min and it is called "seeding" the population with a Min-min chromosome.

SA: Simulated Annealing (SA) is an iterative technique that considers only one possible mapping for each meta-task at a time. Simulated annealing uses a procedure that probabilistically allows poor solutions to be accepted to attempt to obtain a better search of the solution space based on a system temperature [8], [12], [13], [14], [15].

GSA: The Genetic Simulated Annealing (GSA) heuristics is a combination of the GA and SA techniques [16],[17]. GSA follows the procedures similar to the GA. For the selection process, GSA uses the SA cooling schedule and system temperature.

Tabu: Tabu search is a solution space search that keeps track of the regions of the solution space to avoid repeating a search near the areas that have already been searched [8],[18],[19]. A mapping of meta-tasks uses the same representation as a chromosome in the GA approach. The implementation of tabu search begins with a random mapping, generated from a uniform distribution.

A*: A* is a tree search technique based on an m-array tree, beginning at a root node that is a null solution [20]. As the tree grows, intermediate nodes represent partial mappings and leaf nodes represent final mappings. Each node has a cost function, and the node with the minimum cost function is replaced by its child node. Whenever a node is added, to reduce the height of the tree, the tree is pruned by deleting the node with the largest cost function. This process is repeated until a complete mapping (a leaf node) is reached.

Though the above stated heuristic algorithms have advantages, they do have their own disadvantages. OLB leads to poor makespan since it does not consider the expected execution time while mapping the meta-tasks to the machines and it is also hard to achieve dynamic load balance of jobs. MET results in severe load imbalance across the machines. Static mapping of meta-task to machine using MCT heuristic algorithm leads to poor makespan since it takes more time for a job to map to the particular machine. Max-min is appropriate only when most of the jobs arriving to the grid systems are shortest and also Max-min outperforms Min-min. The experimental results show that Duplex, SA, GSA, and Tabu do not produce good mappings. Min-min, GA, and A* were able to deliver good performance. GA is better than Min-min by few percents, and also it has to be "seeding" the population with a Min-min chromosome to obtain its good performance. In different situations, A* produce better or worse mappings than Min-min and GA. Among the three algorithms, Min-min is the fastest algorithm, GA is much slower, and A* is very slow. Among the stated algorithms, Min-min is the simple and fastest algorithm and its good performance depends on the choice of mapping the meta-tasks to the first choice of minimum execution time. However the drawback of Min-min is that, it is unable to balance the load because it usually assigns the small task first and few larger tasks, while at the same time, several machines sit idle, which leads to poor utilization of resources. The proposed algorithm retains the advantage of Min-min algorithm and reduces the idle time of the resources, which in turn leads to better makespan.

III. PROBLEM DESCRIPTION

We consider a scheduling problem where tasks are to be allocated immediately to the resources in a global, heterogeneous and dynamic environment. The allocation should be as fast as possible, while at the same time optimizing several criteria such as response time, utilization and slowdown. The tasks have to be completed on a unique resource. There are no dependencies between tasks (each task is independent). The arrival rate of tasks determines the

system load. Since we consider a heterogeneous environment, the processing capacity of each resource in the system may vary significantly and thus yields different runtimes for a particular task on different machines.

An instance of the problem consists of the following:

- A number of independent tasks to be scheduled
- A number of heterogeneous machines (resources)
- The task's expected time to compute on each machine. (This depends on the workload of each task and computing capacity of each machine). The ETC matrix: ETC[*i*][*j*] is the expected execution time for task *i* on machine *j*.
- Ready time (*ready*[*m*]), the time when machine *m* will finish the previously assigned tasks.

In this work, we assume that the computation time for each task is known accurately before the task begins the execution.

IV. MAX-MIN METHODOLOGY AND IMPLEMENTATION

Task scheduling system is the most important part of grid resource management system. The scheduler receives the task request, and chooses appropriate resource to run that requested task. In this paper, the formulation of task scheduling is based on the expected time to compute (ETC) matrix. A meta-task is defined as a collection of independent task (i.e. task doesn't require any communication with other tasks) [1, 9]. Tasks derive mapping statically. For static mapping, the number of tasks, t and the number of machines, m is known as priori. *ETC (i,j)* represents the estimated execution time for task t_i on machine m_j. The expected completion time of the task t_i on machine m_j is

$$\mathbf{ct\ (t_i, m_j) = mat(m_j) + ETC(t_i, m_j)}$$

where mat (m_j) is the machine availability time. i.e. the time at which machine m_j completes any previously assigned tasks. The main aim of the heuristic scheduling algorithm is to minimize the makespan

$$\mathbf{makespan = max\ (ct\ (t_i, m_j))}$$

The proposed heuristic scheduling algorithm Min-mean works in two phases.

- In phase 1, the task allocation is done based on the Max-min algorithm.
- In phase 2, the mean of completion time of all the machines are taken. The machine whose completion time is greater than the mean value is selected. The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the mean value.

The related definition of proposed Max-min Average heuristic scheduling algorithm is as follows:

- ET_{ij} - the amount of time taken by machine M_j to execute $Task_i$ given that M_j is idle when $Task_i$ is assigned.
- CT_j - the expected completion time of M_j
- Mat (m_j) - the machines availability time i.e. the time at which Machine j completes any previously assigned tasks.
- Group (CT_i, Machines) –The function "f1" is used to group all the tasks and machines that has minimum completion time.
- The best minimum task/machine pair (m, n) is selected from the Group
- MeanCT- is used to find the mean completion of all the machines.

Algorithm Max-Min Average

(1) While there are tasks to schedule
(2) For all $Task_i$ to schedule
(3) For all $Machine_j$
(4) Compute $CT_{i,j}$; $CT_{i,j}$ = Mat (m_j) + T_{ij}
(5) End for
(6) Group (CT_i, Machines) =f1 (CT_i, 1, T_i, 2 ...)
(7) End for
(8) Select the best maximum pair (task, machine) from the Group
(9) Compute maximum $CT_{m,n}$
(10) Reserve task m on n
(11) End while
//Optimization based on MeanCT
(12) Calculate MeanCT= (ΣCT j)/No of machines
(13) For all $Machine_j$
(14) if (CT_j<=MeanCT)
(15) Select tasks reserved on the host
(16) End for
(17) For all $Task_i$ reselected
(18) For all $Machine_j$ with (CT_j >=MeanCT)
(19) Compute $NewCT_{i,j}$ = CT($task_i$, $host_j$)
(20) if($NewCT_{i,j}$>= MeanCT)
(21) Group (CT_i, Machines) =f1 (CT_i, 1, CT_i, 2 ...)
(22) End for
(23) Select the best maximum pair from the Group
(24) Reschedule ($task_i$ on $machine_j$)
(25) Compute $NewCT_{m,n}$
(26) End for

V. EXPERIMENT RESULTS

TABLE I : Sample ETC matrix

TASK #	M1	M2
T1	1	2
T2	2	4
T3	5	9

Min-min heuristic scheduling algorithm executes all the shortest tasks first and then the longest tasks. Table 1 gives a sample ETC matrix, with the expected execution time of three tasks (t1, t2, t3) on the two machines (m1, m2). This sample ETC matrix clearly explains how proposed Max-min average heuristic scheduling algorithm performs better than the Min-min algorithm. It is assumed that both the machines are idle at the start.

The sequence of the execution of Min-min algorithm and the proposed Max-min average heuristic scheduling algorithm is as follows:

Step 1: Static mapping of tasks to machines based on Min-min is shown in Figure 1. Min-min algorithm gives a makespan of 8 sec.

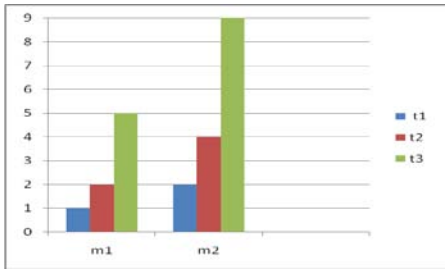

Figure 1: Results of Min-min algorithm

Step 2: The mean completion time for the sample ETC matrix can be calculated by using the following relation:

MeanCT = CTm1+CTm2

where CTm1: Completion time of all tasks on machine m1, CTm2: Completion time of all tasks on machine m2.

MeanCT = 4 sec.

Step 3: Tasks on machines m1 are selected as shown in the Figure 2 because of CTm1 > MeanCT.

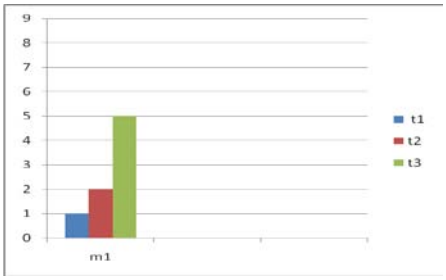

Figure 2: Selected tasks on machine m1

Step 4: Rescheduling of the tasks on machine m1 to the machine m2 is done, whose expected execution time is

ET_i >= MeanCT

The final scheduling of the tasks (t1, t2, t3) on two machines (m1, m2) using the proposed Min-mean heuristic scheduling algorithm is shown in the Figure 3. Min-mean heuristic scheduling algorithm gives a makespan of 7 sec.

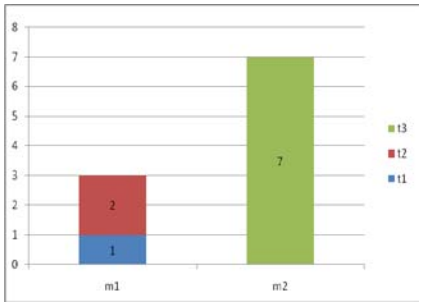

Figure 3 : Results of Max-Min Average Algorithm

The Figure 1 and Figure 3 clearly show that the Max-min Average heuristic scheduling algorithm performs better than Min-min algorithm.

The comparison results of Figure 1 and Figure 3 is as follows:

- The idle time of the machine m2 is reduced.
- The load is well balanced in both the machines m1 and m2.
- The measure of the throughput of the heterogeneous computing systems is termed as makespan. The makespan can be calculated as makespan = max (CT_i) Ti ε meta-tasks makespan = 7 sec. Figure 1 and Figure 3 shows that the makespan using Max-Min Average is reduced compared to that of the makespan using Min-min.

CONCLUSION AND FUTURE WORK

The experimental results show that the Max-min Average heuristic scheduling algorithm performs better than the existing heuristic algorithm in various systems and settings and also it delivers improved makespan on various heterogeneous environments. The future research will be directed towards the factors such as CPU workload, communication delay and so on.

REFERENCES

[1] Hesam Izakian, Ajith Abraham, Vaclav Snasel, "Performance comparison of six efficient pure heuristics for scheduling meta-tasks on Heterogeneous Distributed Environments", September 22, 2009.

[2] Hesam Izakian, Ajith Abraham, Vaclav Snasel, "Metaheuristic Based Scheduling Meta-Tasks in Distributed Heterogeneous Computing Systems", Sensors 2009, 9, 5339-5350,7 July 2009, ISSN 1424-8220.

[3] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions", in 7th IEEE Heterogeneous Computing Workshop (HCW '98), pp. 79_87, 1998.

[4] R. F. Freund, M. Gherrity, S. Ambrosius, et al., "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", in 7th IEEE Heterogeneous Computing Workshop (HCW '98), pp. 184_199, 1998.

[5] R. F. Freund and H. J. Siegel, Heterogeneous processing, IEEE Comput. 26, 6 (June 1993),13_17.

[6] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors", J. Assoc. Comput. Mach. 24, 2 (Apr. 1977), 280_289.

[7] J. H. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, MI, 1975.

[8] Z. Michalewicz and D. B. Fogel, "How to Solve It: Modern Heuristics," Springer-Verlag, New York,2000.

[9] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms", in 5th IEEE Heterogeneous Computing Workshop (HCW '96), pp. 86_97, 1996.

[10] Y. G. Tirat-Gefen and A. C. Parker, "MEGA: An approach to system-level design of application specific heterogeneous multiprocessors", in 5th IEEE Heterogeneous Computing Workshop (HCW '96), pp. 105_117, 1996.

[11] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach", J. Parallel Distrib. Comput. 47, 1 (Nov. 1997), 1_15.

[12] M. Coli and P. Palazzari, "Real time pipelined system design through simulated annealing", J. Systems Architecture 42, 6_7 (Dec. 1996), 465_475.

[13] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Optimization by simulated annealing, Science 220, 4598 (May 1983), 671_680..

[14] S. J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach"', Prentice_Hall, Englewood Cliffs, NJ, 1995.

[15] A. Y. Zomaya and R. Kazman, "Simulated annealing techniques, in Algorithms and Theory of Computation Handbook" (M. J. Atallah, Ed.), pp. 37-1_37-19, CRC Press, Boca Raton, FL, 1999.

[16] H. Chen, N. S. Flann, and D. W. Watson, "Parallel genetic simulated annealing: A massively parallel SIMD approach", IEEE Trans. Parallel Distrib. Comput. 9, 2 (Feb. 1998), 126_136.

[17] P. Shroff, D. Watson, N. Flann, and R. Freund, "Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments", in ``5th IEEE Heterogeneous Computing Workshop (HCW '96)," pp. 98_104, 1996.

[18] I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro, "Improving search by incorporating evolution principles in parallel tabu search", in 1994 IEEE Conference on Evolutionary Computation, Vol. 2, pp. 823_828, 1994.

[19] F. Glover and M. Laguna, "Tabu Search", Kluwer Academic, Boston, MA, 1997.

[20] K. Chow and B. Liu, "On mapping signal processing algorithms to a heterogeneous multiprocessor system", in 1991 International Conference on Acoustics, Speech, and Signal Processing (ICASSP'91),Vol. 3, pp. 1585_1588, 1991.