Optimized Assignment of Independent Task for Improving Resources Performance in Computational Grid

Sarpreet Singh[1], R.K. Bawa[2],
[1]Assistant Professor, Department of Computer Science& Engineering,
Sri Guru Granth Sahib World University, Fatehgarh Sahib
[2]Professor, Department of Computer Science, Punjabi University Patiala [2]

Abstract

Grid computing has emerged from category of distributed and parallel computing where the heterogeneous resources from different network are used simultaneously to solve a particular problem that need huge amount of resources. Potential of Grid computing depends on my issues such as security of resources, heterogeneity of resources, fault tolerance & resource discovery and job scheduling. Scheduling is one of the core steps to efficiently exploit the capabilities of heterogeneous distributed computing resources and is an NP-complete problem. To achieve the promising potential of grid computing, an effective and efficient job scheduling algorithm is proposed, which will optimized two important criteria to improve the performance of resources i.e. makespan time & resource utilization. With this, we have classified various tasks scheduling heuristic in grid on the basis of their characteristics.

Keywords

Grid Computing, Scheduler, ETC matrix, Makespan, Resource utilization

1. Introduction

Grids computing is evolved from existing technology like distributed & parallel computing , web services, Internet, various cryptography technologies having more intelligent security features and virtualization feature for the next-generation of e-Science and e-business applications [1]. Grid computing is based on the cooperation of multiple processing elements on multiple distributed & dynamic machines, to boost the computational power in the scientific problem and other field which require huge capacity of the CPU cycles & other resources. Due to the dynamic nature of the resources, a grid is another form of heterogeneous computing (HC) system. Different kind of resources used in grid like computation resources (i.e. a machine sharing its CPU), storage resources(i.e. a machine sharing it's RAM or disk space), communication resources (i.e. sharing of bandwidth or a communication path), software and licenses and many other special equipment (i.e. sharing of devices). [2]

The resource management system (RMS) which is central component of a grid computing accepts requests from user and assigns resource from the overall pool of grid resources which fulfill user

requirement. There are number of issues in RMS such as resource discovery, resource scheduling, resource monitoring, resources inventories, resource provisioning, load balancing, fault isolation, service level management system [3]. However, grid scheduling and grid load balancing are the main issues [4].

Scheduling the jobs to the resources in grid computing is also complicated due to the distributed, dynamic and heterogeneous nature of the resources [5].Scheduling is one of the core steps to efficiently exploit the capabilities of heterogeneous distributed resource. A scheduling is a process that maps and manages the execution of inter-dependent tasks/independent task on the distributed resources. It allocates suitable resources to various tasks to satisfy the agreement between resource producer & resource consumer. Poor scheduling will reduce performance of the nodes participating in grid system. Due heterogeneous, dynamic nature of resources, the problem of mapping tasks on distributed services belongs to a class of problems known as NP-hard problem [6]. Due to dynamic & heterogeneous nature of resources, neither algorithm can produce optimal schedule. We can only make our best to find the most suitable solution for users.

Working of grid scheduler passing from four stages i.e. a) Resource discovery b) Resource selection c) Job selection d) Job execution. Scheduler is responsible for selecting resources from pool resources and schedule tasks in such a way that the resource usage policy and task constraints are satisfied, in terms of execution time, deadline & cost of the resources utilized [7]. Scheduler returns the "best" such schedule. "Best" is defined by some performance metric [8]. All scheduling algorithms which were mostly considering only single parameter i.e. makespan(completion time of last finished task or job) which is not sufficient to produce idle grid system. According to Wright [9], a scheduler is designed to satisfy one or more of the following common objectives: (a) maximizing system throughput; (b) maximizing resource utilizations; (c) maximizing economic gains; and (d) minimizing the turn-around time for an application. To minimize the overall turn-around time of the tasks and thus increase the throughput of the system, it is important that right resources be assigned to every task. Since task may have dependency or independency between each other's, so there are two case to assign tasks in grid. First case is for dependent task. In this, split the task into multiple subtasks. Then these subtasks assigned to a resource which fulfill the requirements of task. These tasks may have dependency among themselves. This process is termed matching. After this order of execution of the subtasks is identified. During this ordering, dependencies may be considered & the process of ordering for execution is called scheduling. The overall process of matching and scheduling is termed mapping. Second case is for mapping is for independent tasks. The aim of mapping in both above cases is to maximize an objective function, which is based on QoS attributes such as execution time, response time or those requested by the users of the HC system [10]. To achieve such objective, certain heuristics have been developed for mapping. In the following section, various mapping heuristics will be discussed and the assumptions made while describing the heuristics.

2. Related Work

Mapping heuristics can be either *static* or *dynamic*. In Static mapping information regarding all participating nodes is collected & matching and scheduling decisions are made before the actual execution of the task. In dynamic mapping, matching and scheduling decisions are made on the fly as the application executes. The accuracy of static mapping heuristics depends on the accuracy of these estimates. The result scheduling in the grid systems are based on the mapping heuristics.

The component called mapper maintains a two dimensional matrix known as the expected time to compute (ETC) matrix. It contains the expected execution times of a task that are running on all participating node in the grid. execution time of the tasks on different nodes is represented by row entries in ETC & time taken by a node to execute tasks is represented by column entries in ETC.

2.1 Heuristic descriptions

This section, discuss five heuristics for scheduling tasks to various machines which are participating in grid. Then we propose an efficient heuristic called Optimized Assignment of Independent Task (OAIT) algorithm

Opportunistic Load Balancing: OLB assigns tasks to the next available machine in grid. If more than one machine is available, any one machine is chosen arbitrarily. It does not consider expected execution time of the task on that machine. [11]

Fast Greedy or Minimum Completion Time (MCT): This heuristic assigns each task to the machine which completes in minimum time. MCT heuristic does not consider execution time of a task on that machine, so some time execution time of task will be increase on the machine where the task has to be assigned. [12]

Minimum Execution Time (MET): In this heuristic task is assigned in to machine which give least execution time for that task's execution. Due to this, it can cause load imbalance on resources. MET has advantage over MCT, is its simplicity of implementation. [13]

Min-min: The Min-min heuristic begins with the set of all unmapped tasks. Then, minimum completion time for each task is found on every machine. Next, the task with the overall minimum completion time is selected and assigned to the corresponding machine (hence the name Min-min). Last, the newly mapped task is removed from *U*, and the process repeats until all tasks are mapped (i.e., *U* is empty). Min-min & MCT both heuristic produce minimum completion time. [13][14]

Max-Min: The max-min mapping heuristic is similar to the min-min mapping heuristic. The first step of this heuristic is identical to the min-min heuristic. In the second step, instead of choosing the task having the minimum of earliest completion times among all the tasks, max-min chooses the task having maximum of earliest completion times and assigns it to the corresponding machine. The machine availability time is updated and the process is repeated for every task in the metatask. The max-min mapping heuristics would generally outperform min-min mapping heuristics, when the number of short tasks is greater than that of long tasks. [12][14].

3. Proposed OAIT Algorithm

All discussed heuristics in previous section are known for reducing makspan time based on different set of inputs. But utilization of resources is not up to the mark. While proposed OAIT algorithm reduces makspan as well as optimizes the resource utilization & balance the load on all participating nodes.

When a job is received, mostly heuristic try to find node, which executes job in minimum time without considering current load on that node. As a result some nodes will be overload & load distribution is imbalance. But OAIT heuristic, assigning bigger jobs before assignment of smaller jobs by considering current load on nodes as shown in fig 1. N represent node. Length of Bar represent the execution capacity of node i.e. more length more execution capacity. Let us consider two jobs one is heavy job J_5 (i.e. with maximum number of instruction) assign to fast node N_0 & second light job J_1 (i.e. with minimum number of instruction) assign to low execution capacity node N_3.As a result waiting period for longer jobs will be reduced & also till the completion of longer jobs many smaller jobs will be executed parallel on other light nodes. This will give better resource utilization & reduce the overall turnaround time.

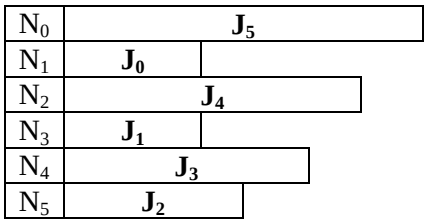

Fig 1: Job distribution

3.1 Proposed grid scheduling architecture

Grid is basically a collection of resources. Each resource consists of number of machine & each machine may have more than one number of processing elements (PE). Fig 1 shows the proposed Grid scheduling architecture. The architecture is composed of Job queue, Grid scheduler (GS), Machines, Local scheduler (LS), processing elements (PE). User submits job to Job queue. The GS analyzes the job's requirements and then select appropriate machine, after gathering information from various Local Scheduler which are connected with GS. Then GS pass job to selected Local Scheduler. LS have complete information about their local PE like execution capacity, current load etc. This information is used by LS, to select the best PE using the proposed scheduling policy & assign the jobs for execution.

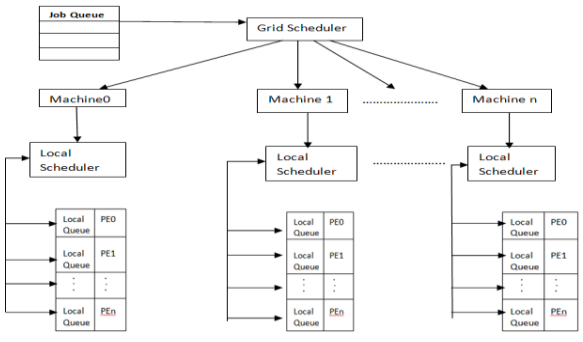

Fig 2: Grid Scheduling Architecture

3.2 Terminology used in Proposed OAIT algorithm:

- JOBQUEUE [id][# instruction]. Its 2D array having two parameters i.e. job Id & number of instructions.
- Job Size: Define by number of instructions.
- Node (M_i,PE_j): Define by i^{th} PE on j^{th} machine.
- Turnaround time (TAT): It is the numeric value of node. Its interval from the time of submission of a job till completion of job. If the TAT value of any node is non-zero, means node is busy to executing task & calculated by:

  TAT = Old TAT of node + (number of instruction /Power of node (Hz))
- Power of node: Capacity to executing jobs. Define by MIPS(millions of instruction per second)
- Expected time to compute (ETC) matrix**:** its 2D array, where rows represent participating machines in grid & column entry represents all PE on that machine. Each entry in ETC represent by TAT of that node. Shown in fig 3.

	PE_1	PE..	PE_n
MC_1			
MC_2			
MC..			
MC_n			

Fig 3. ETC Matrix

- Temp_TAT_F[][]: Its representation is similar to ETC matrix & use to store TAT value of each node (M_i,PE_j) for executing job having minimum number of instructions.
- Temp_TAT_L[][]:Its representation is similar to ETC matrix & use to store TAT value of each node (M_i,PE_j) for executing job having maximum number of instructions.

3.3 OAIT Algorithm

1. Create Grid resource by defining number of machines & PE with MIPS rating.
2. Submit jobs in array JOBQUEUE[id][# instruction]
3. Sort the jobs of JOBQUEUE [id][# instruction] in ascending order, based on second parameter of JOBQUEUE(i.e. # instruction).
4. Initialize entry of ETC [][] matrix to zero.
5. Scheduler component read two jobs at same time, first $Job_{\#min}$ i.e. one with minimum number of instructions & second $Job_{\#max}$ i.e. with maximum number of instructions from JOBQUEUE [][] , until JOBQUEUE will be empty

   a) Calculate TAT for $Job_{\#min}$ & $Job_{\#max}$ on every node separately, & store in array "Temp_TAT_f [][] " & "Temp_Tat_L [][]" respectively.
   b) Choose the minimum value ($TAT_{min)}$ from array Temp_Tat_L[][] & select its corresponding Machine(MC_l) & processing element (PE_l)
   c) Choose the maximum value (TAT_{max})from array Temp_Tat_f[][] but less than the TAT_{min} i.e. (TAT_{max}<TAT_{min}) & select its corresponding machine MC_f & PE_f

d) if(MC_f ==MC_l && PE_f ==PE_l)
   i. Choose the second largest value (TAT_{sec_max}) from array Temp_Tat_f[][] but less than the TAT_{min} i.e. (TAT_{sec_max}<TAT_{min}) & select its corresponding MC_f & PE_f
   ii. ETC[MC_f][PE_f]=TAT_{sec_max};
   ETC [MC_l][PE_l]= TAT_{min};

e) Else update array ETC[][]
   i. ETC [MC_f][PE_f]= TAT_{max};
   ii. ETC [MC_l][PE_l]= TAT_{min};
f) Go to step 5.

6. After every second, another thread decrement every TAT value of ETC matrix
7. If all entry in ETC matrix become zero then exit

Initialize all entry of ETC matrix to zero, represent nodes are free. JOBQUEUE receive jobs from user & rearrange in ascending order according to priority. Priority may be defined by any criteria, but in proposed OAIT priority is number of instructions. Proposed OAIT heuristic read two jobs from JOBQUEUE at same time i.e. one with minimum instruction ($Job_{\#min}$) & second with maximum instruction ($Job_{\#max}$). Find the node N_L, which execute $Job_{\#max}$ in minimum time & calculate TAT_{min}. For better resources utilization & reduce the makespan for all pending Jobs , OAIT finds slowest node from pool of nodes which executes $Job_{\#min}$ with maximum time & calculate TAT_{max} , but less than TAT_L i.e. (TAT_{max} < TAT_{min}). Update the ETC matrix by final T_{max} & TAT_{min} corresponding to the node. After assignment of all jobs, decrement thread called, which decrement the TAT value of busy nodes by one after every second, till TAT value of nodes becomes zero.

4. Experimental Detail:

OAIT, MET, Min-Min, Max-Min, MCT heuristics are implemented on Grid simulator (GridSim). Table 1 shows the scenario of Grid, in which four machines are used with different number of PE to execute the Jobs. Each PE has specific MIPS rating. Table 2 shows set of 40 jobs with number of instruction (MI) that are executed on Grid simulator using different scheduling heuristic.

Machine	PE_0 (*mips*)	PE_1 (*mips*)	PE_2 (*mips*)	PE_3 (*mips*)
Mc_0	90	150	600	
Mc_1	80	350		
Mc_2	700			
Mc_3	450	60	200	120

Table 1

Job id	# instruction (MI)	Job id	# instruction(MI)
Job_0	20000	job_{20}	2000
job_1	16000	job_{21}	10000
job_2	12000	job_{22}	3000

job_3	13000	job_{23}	70000
job_4	19000	job_{24}	650000
job_5	15000	job_{25}	30000
job_6	60000	job_{26}	30000
job_7	7000	job_{27}	89900
job_8	2000	job_{28}	32900
job_9	45000	job_{29}	12000
job_{10}	3000	job_{30}	4000
job_{11}	65500	job_{31}	2000
job_{12}	18000	job_{32}	34000
job_{13}	5500	job_{33}	130000
job_{14}	30000	job_{34}	99900
job_{15}	14500	job_{35}	14500
job_{16}	300000	job_{36}	18000
job_{17}	120000	job_{37}	3000
job_{18}	320000	job_{38}	184000
job_{19}	315000	job_{39}	150000

Table2

5. Result & Conclusion

The proposed OAIT algorithm is evaluated for Makespan time, Resource utilization and compared with the other existing heuristics. Table 3 compare the makespan time of the proposed OAIT algorithm and the existing MET, Min-Min, Max-Min, MCT heuristics. From fig 4, it is found that proposed OAIT algorithm reduces the Makespan time i.e. executes all the jobs in minimum time, comparatively to other existing heuristics.

Scheduling Heuristics	**Makespan time(sec)**
OAIT	1082
MET	4222
Min-Min	3413
Max-Min	4899
MCT	1496

Table 3.Compartition of Makespan time

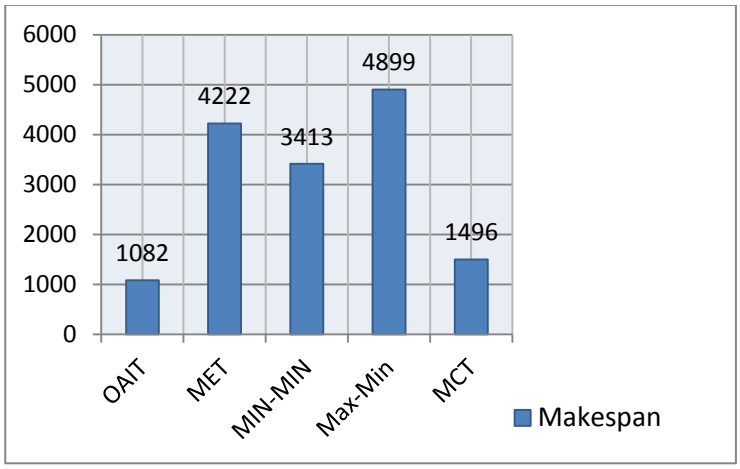

Fig 4 Makespan Time

Table 4 compares Resource utilization time (how much time resources are used to executing the assigned jobs) by proposed OAIT algorithm and other existing MET, Min-Min, Max-Min, MCT heuristics. From fig 5, it is found that proposed OAIT algorithm gives better result as comparatively to other existing heuristics. OAIT algorithm use resources in optimized way & for constant time till the execution of last job. This will also help to distribute computational load among all participating nodes according to capacity of node. While MET, Max-Min, Min-Min shows unbalanced execution load, which lead to poor utilization of resources. MCT shows better result for utilizing resources but not so well as comparatively than proposed OAIT algorithm.

Nodes	**OAIT**	**MET**	**Max-Min**	**Min-Min**	**MCT**
	Time(sec)	Time(sec)	Time(sec)	Time(sec)	Time(sec)
Mc-0 /PE-0	954	0	9604	0	355
Mc-0/PE-1	1066	0	8016	3065	915
Mc-0 /PE-2	1043	0	5566	5743	1191
Mc-1 /PE-0	999	0	8849	0	243
Mc-1 /PE-1	1079	0	7345	2279	1000
Mc-2 /PE-0	1072	4222	7842	3640	1496
Mc-3 /PE-0	1050	0	6759	2004	1170
Mc-3 /PE-1	1007	0	9765	0	116
Mc-3 /PE-2	1010	0	7107	3449	427
Mc-3 /PE-3	1082	0	7921	1399	382

Table 4: Comparison of Resource utilization

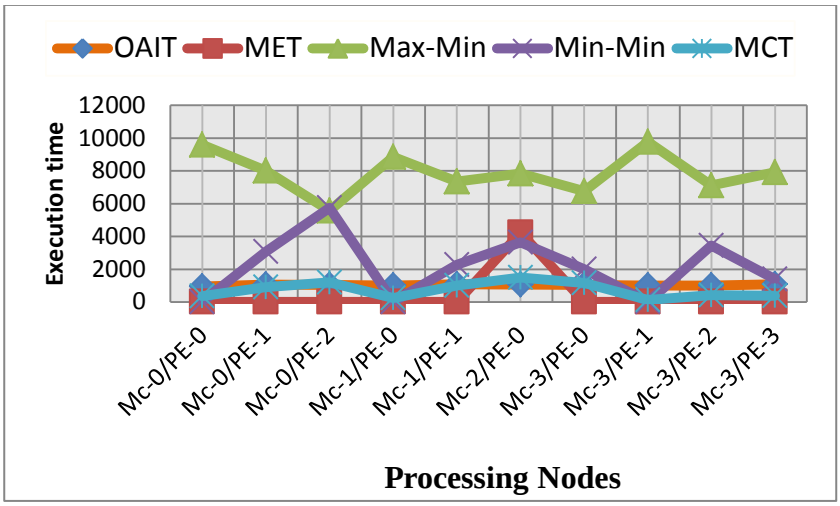

Fig. 5 Resources utilization

REFERENCES

[1] I. Foster and C. Kesselman (Eds.). *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[2] Jarek Nabrzyski, Jennifer M. Schopf & Jan Weglarz, Grid Resource Management– State of the art and Future trends, Kluwer Academic Publisher

[3] Sharma, A., & Bawa, S. Comparative analysis of resource discovery approaches in grid computing.
Journal of Computers, 3(5), 60-64, 2008.

[4] Moallem, A., & Ludwig, S. Using artificial life techniques for distributed grid job scheduling. Proceedings of the 2009 ACM Symposium on Applied Computing, 1091-1097, (2009).

[5] Li, Y. A bio-inspired adaptive job scheduling mechanism on a computational grid. International Journal of Computer Science and Network Security, 6(3B), 1-7., 2006

[6] J. D. Ullman, NP-complete Scheduling Problems, *Journal of Computer and System Sciences*, 10:384-393, 1975

[7] Maozhen Li, Mark Baker, The Grid Core Technologies, A John Wiley & Sons, Inc., 2005.

[8] H. Dail, H. Casanova, and F. Berman, A Decoupled Scheduling Approach for the GrADS Environm ent, in Proc. 2002 ACM/IEEE conference on Supercomputing,pp.1-14, Baltimore, Maryland USA, November 2002

[9] D Wright, Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor, Proceeding of HPC Revolution 01, Illinois, 2001.

[10] Howard Jay Siegel and Shoukat Ali. Techniques for mapping tasks to machines in heterogeneous computing systems. *Journal of Systems Architecture*, 46(8):627–639, 2000. Available online at: citeseer.ist.psu.edu/siegel00techniques.html (accessed January 1st, 2009).

[11] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson,M. Theys, B. Yao, D. Hensgen and R. Freund, A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, in J. of Parallel and Distributed Computing, vol.61, No. 6, pp. 810-837, 2001

[12] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Ladislau Bni, Muthucumaru Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, Bin Yao, Debra A. Hensgen, and Richard F. Freund. A comparison study of static mapping heuristics for a class of metatasks on heterogeneous

computing systems. In *Heterogeneous ComputingWorkshop*, volume 8, pages 15–29. IEEE Computer Society, 1999.

[13] Technical Report No. 2006-504 "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems '', Fangpeng Dong and Selim G. Akl ,School of Computing, Queen's University Kingston, Ontario ,January 2006

[14] Sameer Singh Chauhan, and R. C. Joshi," A Weighted Mean Time Min-Min Max-Min Selective Scheduling Strategy for Independent Tasks on Grid", In Advance Computing Conference (IACC), IEEE, pp. 4-9 ISBN: 978-1-4244-4790-9 , 2010

Authors

Er. Sarpreet Singh has done Master in the area of Grid Computing from Department of Computer science, Punjabi University, Patiala, INDIA. Presently he is working as Assistant professor at Sri Guru Granth sahib World University, Fatehgarh sahib & doing his Ph.D from Punjabi University, Patiala, INDIA. His area of interest is Grid & distributed computing

Dr. Rajesh K. Bawa has done Master's and Ph.D degree in the area of Numerical Computing from IIT Kanpur INDIA. Presently he is working as Head & Professor in Department of Computer science, Punjabi University, Patiala, INDIA. His present area of interest is Parallel and Scientific Computation. He has published many research papers in international & national journals.