Performance Factor based Local Scheduling for Heterogeneous Grid Environments

G.Sumathi
Professor
Department Of Information Technology
Sri Venkateswara College of Engineering
Sriperumbudur, India

S.Sathyanarayanan
Student
Department of Information Technology
Sri Venkateswara College of Engineering
Sriperumbudur, India

R.Santhosh Kumar
Student
Department of Information Technology
Sri Venkateswara College of Engineering
Sriperumbudur, India

ABSTRACT

Grids [3] have emerged as paradigms for the next generation parallel and distributed computing. Computational Grid can be defined as large-scale high-performance distributed computing environments that provide access to high-end computational resources. Grid scheduling is the process of scheduling jobs over grid resources. Improving overall system performance with a lower turnaround time is an important objective of Local Grid scheduling. In this paper a Performance Factor Based Local Scheduling Algorithm is proposed. In this algorithm priority for each subtask in the Grid is assigned based on two new parameters, Computational Complexity and Performance Factor. The algorithm classifies the subtasks into high, medium and low categories based on their priority. The value for performance factor is assigned based on the value of processing power of each node i.e. Number of operations per cycle per processor and the number of instructions processed per second. The subtasks are then mapped to respective processors based on the assigned priority for execution. A subtask, which requires a very high performance factor and that exhibits high computational complexity, is given a high priority. Prioritizing the subtasks in this way can improve the performance of grid resources that in turn improve the overall efficiency of the computational grid. The effectiveness of this algorithm is evaluated through simulation results.

1. INTRODUCTION

Computational Grids are emerging as a new computing paradigm for solving challenging applications in science, engineering and economics [1].

Computational Grid can be defined as large-scale high-performance distributed computing environments that provide access to high-end computational resources [2].Each of these resources could be a uni- processor machine, a symmetric multiprocessor cluster, a distributed memory multiprocessor system, or a massively parallel supercomputer. Each resource consists of a number of heterogeneous nodes. The resources on the grid are usually accessed via an executing "job".

A Grid scheduling is the process of scheduling jobs over grid resources. A grid scheduler is different from local scheduler in that a local scheduler is in charge of resource allocation, assigning of sub-tasks and subtask execution management.

In heterogeneous grid environment [3] with its multitude of resources, a proper scheduling and efficient load balancing across the grid can lead to improved overall system performance and a lower turn-around time for individual jobs. First Come First Serve (FCFS) algorithm neither considers any of the subtask parameters nor the resource parameters. Shortest Subtask Fastest Node (SSFN) and Longest Subtask Fastest Node (LSFN) algorithms consider computational complexity of subtasks for scheduling and ignore the priority of a subtask. A scheduling algorithm based on the performance factor of the nodes in the resource is proposed and tested. The Performance Factor based local scheduling algorithm assigns a priority to the subtask based on the parameters Computational Complexity and Performance Factor.

The value of the Performance Factor is assigned based on the number of operations per cycle per processor and the number of instructions processed per second. In this a subtask which requires high performance factor and which exhibits high computational complexity is given a high priority. A subtask, which exhibits high computational complexity and requires low performance factor, is given a low priority. A subtask, which exhibits a medium computational complexity and requires medium performance factor, is given a medium priority. The fastest free node available in the resource is allocated to the subtask which has high priority. Prioritizing the subtasks based on their nature can improve the real time performance of computational grids.

The rest of the paper is organized as follows: The grid framework is presented in Section 2. The proposed scheduling algorithm is discussed in Section 3. The performance study is carried out and results are discussed in Section 4. Finally, some concluding remarks are made in Section 5.

2. GRID FRAMEWORK

Fig.1 shows the framework of the grid [4]. The Global and Local Grid Resource Brokers (GGRB & LGRB) and Grid Information Server (GIS) are the three main components of the grid. Each of these components has its own independent functionalities that help in grid management and job scheduling and thus serve the purpose of a grid.

The first author thanks the Defense Research Developmental Organizations (DRDO), Government of India, New Delhi, for **its** financial assistance through extramural research grant.

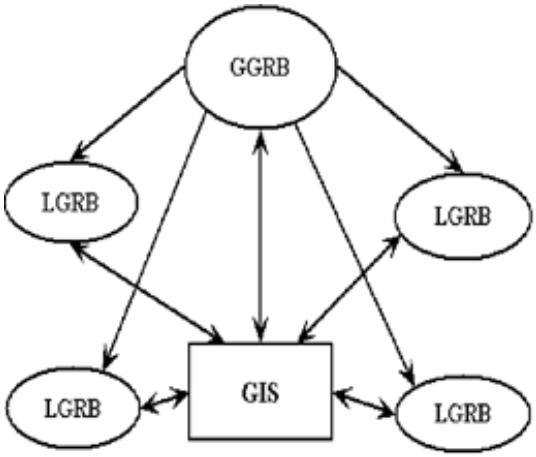

Figure 1. Grid Framework

2.1. Local Grid Resource Broker (LGRB)

The local grid resource broker is a synonym for a grid resource. Each grid resource has been categorized based on its processing speed as follows:

Type 1: TFLOPS Machines Type 2: GFLOPS Machines and Type 3: MFLOPS Machines

This categorization adds to the heterogeneous nature of a grid. Each LGRB in the grid can be any one of the above three resources. There can be many LGRBs possible in the grid. With the addition of every LGRB, the number of resources and consequently the number of processing elements (PEs) are increased. A job submitted to the grid may be migrated to any of the LGRBs in the grid for execution. Once a job has been migrated to a particular LGRB, the LGRB ensures execution of the job on the specified number of processors. Since computational grids have been taken into consideration, the number of processing elements in an LGRB is the actual resource of the grid.

2.2. Global Grid Resource Broker (GGRB)

All the jobs are submitted to the GGRB. A single GGRB takes care of scheduling jobs in the grid based on the resources available as per the scheduling algorithm. Once a task has been scheduled to a particular LGRB, the GGRB migrates the job to that LGRB for execution.

2.3. Grid Information Server (GIS)

The Grid Information Server is the database bank of the grid. It keeps track of the resources available in the grid. Any new LGRB should register itself with the GIS. The GIS provides information regarding free resources to the GGRB based on which the GGRB schedules the jobs.

2.4. Working of the Grid

2.4.1 Registration

Any new LGRB should register itself with the GIS by sending a request. The GIS responds with an acknowledgement, which means that it is ready to accept a new resource as a grid member. Now, it's the LGRBs turn to send the details regarding itself, its type, and number of processing elements and speed of each processing element.

2.4.2 Job Scheduling

The GGRB stores the incoming jobs in a queue. When scheduling is to be done the GGRB requests the GIS with a query for the suitable resources. As soon as the GIS receive a request from the GGRB it sends the IP address of the suitable resource to GGRB, if available. Jobs submitted to the GGRB are migrated to the LGRBs based on a global scheduling algorithm for execution.

3. SCHEDULING ALGORITHM

Proper scheduling algorithm can lead to an improved overall system performance and a lower turnaround time. Since a Grid has heterogeneous resources it is often complex to design an efficient scheduling algorithm.

3.1 Performance Factor Based Local Scheduling Algorithm

The algorithm classifies the subtasks into high, medium and low categories based on their priority. The priority assignment is done by considering the new parameters, Performance Factor of nodes in the resource and Computational Complexity of the subtask.The performance factor is computed by considering the Number of operations per cycle per processor and the Number of instructions processed per second. The priority of each subtask is assigned based on the performance factor calculated and computational complexity of the subtask.The fastest free node available in the resource is allocated to the subtask which has high priority. Prioritizing the subtasks based on their nature can improve the real time performance of computational grids.

3.2 Computational Complexity

Task partitioning algorithm takes care of efficiently dividing an application into tasks of appropriate grain size and an abstract model of such a partitioned application is represented by a Directed Acyclic Graph (DAG). Figure 2 is an example of a DAG that represents an application that has 6 subtasks. Each node of a DAG corresponds to a sequence of operations and a directed arc represents the precedence constrains between the tasks. Each task can be executed on a processor and the directed arc shows transfer of relevant data from one processor to another.

All the operations are represented in terms of additions. Node weight represents the amount of computations (in terms of additions) involved in the particular node. This denotes the Computational Complexity of the subtasks.

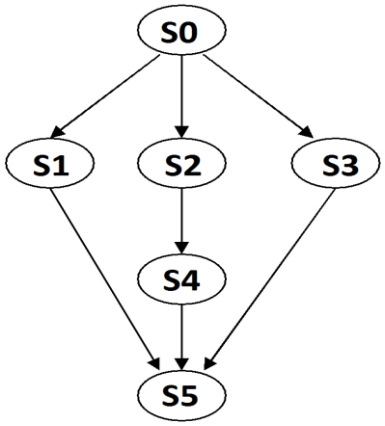

Figure 2. An example DAG

\

3.3 Priority Assignment

A subtask, which requires a high performance factor for execution and exhibits high computational complexity, is given a high priority. A subtask, which exhibits a medium computational complexity and requires a low performance factor, is given a medium priority. The fastest free node available in the gridresource is allocated to the subtask which has the highest priority.

The procedures are given below:

Algorithm

Assign Performance Factor(ResourceListRs_List)

While(Rs_List!=NULL)

For each resource

/*OC = No. of operations per cycle per processor

SP = Speed of the processor

/* Performance FactorList contains the actual value of performance factor for each node available in the resource*/

Performance Factor_List[i] = OC*SP

End While

Find the Max, Min and Mid ranges in Performance Factor_list

For each subtask in Performance Factor_List

If Performance Factor_List[i] >= Maximum

Final_List[i] = High

/*Final_List contains the range of Performance Factor calculated and sorted in descending order for each node available in the resource*/

Else If Factor_List[i] >= Middle

Final_List[i] = Medium

Else Final_List[i] = Low

EndIf

End Assign Performance Factor

Assign Priority Procedure

AssignPriority(Local_List)

While(Local_List !=NULL)

/* Local _ List contains the list of nodes available in the Grid resource */

For each subtask

/* CompC_List contains the Computational Complexity of subtasks*/

If (CompC_List[i] = High AND Final_List[i] = High)

Priority[i] = 1

Else If (CompC_List[i] = High AND Final_List[i] = Medium)

Priority[i] = 2

Else If (CompC_List[i] = High AND Final_List[i] =Low)

Priority[i] = 3

Else If (CompC_List[i] = Medium AND Final_List[i] = High)

Priority[i] = 4

ElseIf (CompC_List[i] =Medium AND Final_List[i] = Medium)

Priority[i] = 5

Else If (CompC_List[i] = Medium AND Final_List[i] = Low)

Priority[i] = 6

Else If (CompC_List[i] = Low AND Final_List[i] = High)

Priority[i] = 7

Else If (CompC_List[i] = Low AND Final_List[i] = Medium)

Priority[i] = 8

ElseIf (CompC_List[i] = Low AND Final_List[i] = Low)

Priority[i] = 9

EndIf

End AssignPriority

Let m represents number of free nodes available in the grid resource and n represents number of subtasks present in the job. The worst case time complexity of the algorithm is O (n logn) when $m \leq n$ and O (m logm) when $m > n$.

4. PERFORMANCE STUDY

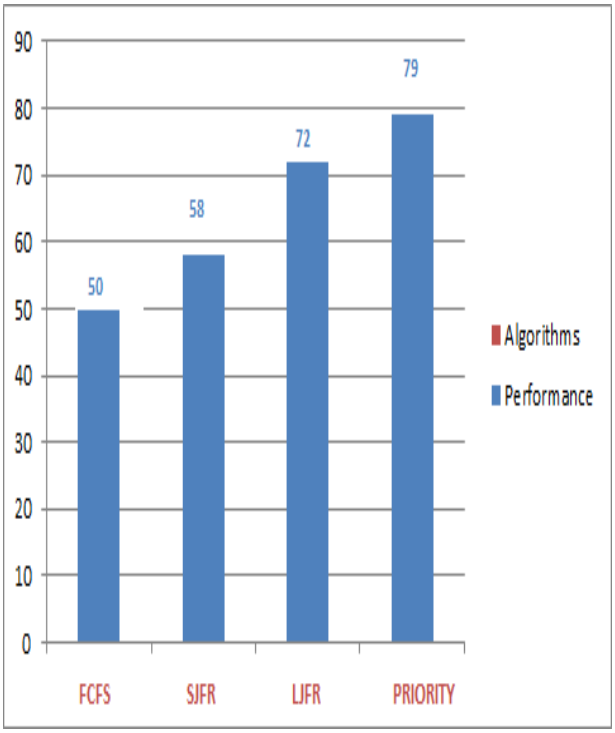

Figure3. Performance Chart

We compare the performance of our algorithm with First Come First Serve, Shortest Subtask Fastest Node and Longest Subtask Fastest Node algorithms.

3.4 First Come First Serve (FCFS)

This scheduling algorithm schedules the subtask on a "First come First serve" basis. We learn that FCFS algorithm is basic and is not based on the factors like computational complexity and performance factor. It shows very low computation results when compared to any of the other local scheduling algorithms in Grid.

3.5 Shortest Subtask Fastest Node (SSFN)

The Shortest Subtask Fastest Node algorithm assigns the subtask exhibiting a diminutive value of computational complexity to the fastest node available in the resource.Shortest Subtask Fastest Node is a scheduling algorithm, which tries to reduce the overall turnaround time of the subtask. From Figure 3 we can decipher that SSFN is more stable in handling subtask and hence outperforms FCFS scheduling algorithm.

3.6 Longest Subtask Fastest Node (LSFN)

The scheduling algorithm, commonly used for the assigning of complex subtasks to high efficiency nodes available in a resource is the Longest Subtask Fastest Node (LSFN) algorithm. It tries to reduce the overall execution time of the subtask.

From Figure3 we can infer that LSFN outperforms FCFS and SSFN as the subtasks are assigned to the fastest node available in the resource which leads to shorter execution time.

3.7 Performance Factor based Local Scheduling Algorithm (PF)

The value of the Performance Factor is assigned based on the processing power i.e. number of operations per cycle per processor and the number of instructions processed per second. In performance factor based local scheduling algorithm a subtask which requires high performance factor and exhibits high computational complexity is given a high priority.

A subtask, which exhibits high computational complexity and requires low performance factor, is given a low priority.

A subtask, which exhibits medium computational complexity and requires nodes in the resource having medium performance factor, is given a medium priority. The fastest free node available in the resource is allocated to the subtask which has high priority.

From Figure4, we can state that PF based algorithm outperforms FCFS, SSFN and LSFN due to its enhanced usage of resources.

Performance evaluation is done based on execution time.

Execution time for each subtask is calculated by calculating the elapsed time between submission time and the completion time of the subtask.

5. CONCLUSIONS

Design of a proper scheduling algorithm with an aim to improve the performance of a grid has indeed been a complex task with a lot of parameters to be taken into consideration. The SSFN and LSFN algorithms take the computational complexity of the subtasks, speed of the nodes in the resource into consideration while scheduling the jobs. In the Performance factor based algorithm a parameter named

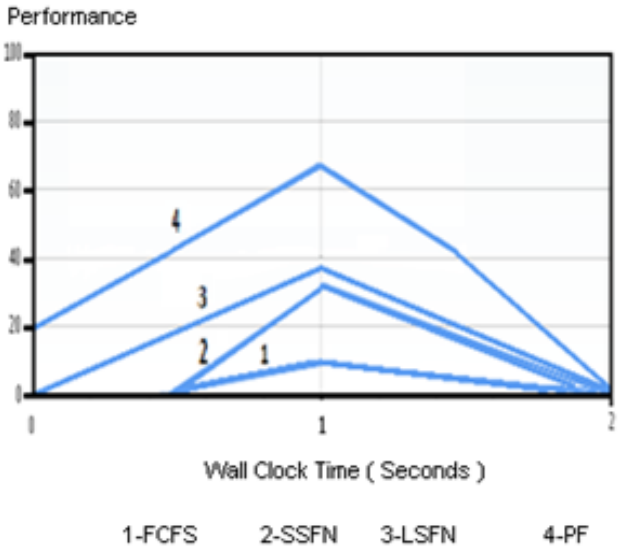

Figure4. Performance Graph

"priority" has been used in the analysis and consequently, the jobs are classified into high, medium and low categories. A subtask, which needs nodes having a high performance factor, exhibiting a high computational complexity is given a higher priority. Prioritizing the subtasks in this way can improve the performance of computational grids. The effectiveness of our algorithm is evaluated through simulation results and its superiority over other known algorithms is demonstrated.

6. REFERENCES

[1] Foster, I., Kesselman, C: The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann (1998)

[2] Foster, I., Kesselman, C: The Globus Project: a Status Report In Poc. IPPS/SPDP'98 Workshop on Heterogeneous Computing pp. 4-18,1998

[3] G.Sumathi,R.SanthoshKumar,S.Sathyanarayanan,"MidJ FR Global SchedulingAlgorithm for HeterogeneousGridEnvironment",IJRTET,November,2011

[4] G.Sumathi, N.P. Gopalan, "PriorityBased Scheduling For Heterogeneous Grid Environments", Proc. Of 10th IEEE International Conference on Communication Systems (ICCS 2006), October, 2006.

[5] AmitAgarwal, Padam Kumar, Economical Task SchedulingAlgorithm for Grid Computing Systems, GJCST Classification(FOR) D.4.1, F.1.2

[6] T.Kokilavani, Dr. D.I. George Amalarethinam, Applying Non-Traditional Optimization Techniques to Task Scheduling in GridComputing an Overview International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 1, No. 4, December,2010

[7] Zhan Gao, SiweiLuo and Ding Ding, A Scheduling Approach Considering Local Tasks in the Computational Grid International Journal of Multimedia and Ubiquitous EngineeringVol. 2, No. 4, October, 2007

[8] Kousalya.K and Balasubramanie.P, Ant Algorithm for GridScheduling Powered by Local Search Int. J. Open Problems Compt.Math., Vol. 1, No. 3, December 2008

[9] Fangpeng Dong and Selim G. Akl, Scheduling Algorithms for Grid Computing: State of the Art and Open Problems TechnicalReport No. 2006-504 .

[10] S.Padmavathi, S.MercyShalinie and R.Abhilaash, A MemeticAlgorithm Based Task Scheduling considering Communication Coston Cluster of Workstations Int. J. Advance. Soft Computing.Appl.,Vol. 2, No. 2, July 2010 ISSN 2074-8523.

[11] Topcuoglu,H.,S.Hariri and M.Y.Wu, "Performance Effectiveand Low Complexity Task Scheduling Algorithm scheduling forheterogeneous computing ", IEEE Transaction on Parallel andDistributed Systems, Vol.13,No.3,(2002).

[12] P.J. Huang, H. Peng, X.Z. Li, Macro adjustment based task scheduling inhierarchical Grid market, in: Proceedings of the 7th International Conferenceon Computational Science, ICCS 2007, Beijing, China, in: Lecture Notes inComputer Science, vol. 4487, Springer-Verlag, 2007, pp. 430_433.

[13] G. Stuer, K. Vanmechelena, J. Broeckhovea, A commodity market algorithm forpricing substitutable Grid resources, Future Generation Computer Systems 23 (5) (2007) 688_701.

[14] C.L. Li, L.Y. Li, QoS based resource scheduling by computational economy incomputational Grid, Information Processing Letters 98 (3) (2006) 119_126.

[15] R. Subrata, A.Y. Zomaya, B. Landfeldt, Artificial life techniques for loadbalancing in computational Grids, Journal of Computer and System Sciences73 (8) (2007) 1176_1190.

[16] C.H. Hsu, T.L. Chen, K.C. Li, Performance effective pre-scheduling strategy forheterogeneous Grid systems in the master slave paradigm, Future GenerationComputer Systems 23 (4) (2007) 569_579.

[17] M. Kalantari, M.K. Akbari, A parallel solution for scheduling of real timeapplications on Grid environments, Future Generation Computer Systems(2009), in press (doi:10.1016/j.future.2008.01.003).