AN IMPROVED CONSTRAINT BASED RESOURCE SCHEDULING APPROACH USING JOB GROUPING STRATEGY IN GRID COMPUTING

Payal Singhal[1], Ravinder Singh [2] and Pinky Rosemarry[3]

[1]Department of Information Technology,
Rajasthan Technical University, Rajasthan, India
payalsinghal8@gmail.com

[2]Assistant Professor, Department of Computer Science and Information Technology,
Rajasthan Technical University, Rajasthan, India
ravikaviya@gmail.com

[3]Department of Computer Science, Rajasthan Technical University, Rajasthan, India
pinkyrose001@gmail.com

ABSTRACT

Grid computing is a collection of distributed resources interconnected by networks to provide a unified virtual computing resource view to the user. Grid computing has one important responsibility of resource management and techniques to allow the user to make optimal use of the job completion time and achieving good throughput. It is a big deal to design the efficient scheduler and is implementation. In this paper, the constraint based job and resource scheduling algorithm has been proposed. The four constraints are taken into account for grouping the jobs, i.e. Resource memory, Job memory, Job MI and the fourth constraint L2 cache are considered. Our implementation is to reduce the processing time efficiently by adding the fourth constraint L2 cache of the resource and is allocated to the resource for parallel computing. The L2 cache is a part of computer's processor; it increases the performance of computer. It is smaller and extremely fast computer memory. The use of more constraint of the resource and job can increase the efficiency more. The work has been done in MATLAB using the parallel computing toolbox. All the constraints are calculated using different functions in MATLAB and are allocated to the resource based on it. The resource memory, Cache, job memory size and job MI are the key factors to group the jobs according to the available capability of the selected resource. The processing time is taken into account to analyze the feasibility of the algorithms.

KEYWORDS

Grid Computing, Job Scheduling, Resource Scheduling, Constraint, Cache

1. INTRODUCTION

In the last 30 years, many research projects were underway in the field of distributed computing. The main research was focused on developing system that can act like one large computer. Many of the early Grid computing concepts were explored to create such programs in which all the resources can be used together. The research was done in this field and a project was led by Ian Foster of ANL and Carl Kesselman of University of Southern California. The project was called Globus and then a suite of tools were created by the team that laid the foundation for Grid computing activities in the academic and research communities. Then, at the 1997 Super Computing Conference, 80 sites worldwide running software based on the Globus toolkit were connected together. This concept came to be known as Grid Computing[1].

Grid computing has emerged as a promising next generation collaborative problem solving platform for industry, science, and engineering. Grid computing is defined as coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [2]. Grid enables this sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations [3]. Grid computing is generally used for problems with large scale collaboration and huge computational and/or data storage requirement. The applications of grid includes large scale simulations in astrophysics, climate modeling, modelling for drug design, high energy physics, infrastructure for multiplayer games etc [4]. The term Grid is chosen as an analogy to a power Grid that provides consistent, pervasive, dependable, transparent access to electricity irrespective of its source.

There are many research areas in Grid computing that needs further work like discovering resources based on their characteristics or name, security, fault tolerance, scheduling of resources and their management etc. This paper mainly focuses on grouping of jobs and allocating resources to the grouped jobs according to the constraints.

The remainder of this paper is organized as follows. The various job and resource scheduling algorithms are surveyed in section II. The scheduling diagram of the proposed algorithm is discussed in section III. The Improved Constraint Based Resource Scheduling Approach using Job Grouping Strategy is described in section IV. With the help of MATLAB, the simulation of scheduling algorithm proposed in the paper is realized in section V. The final section concludes the paper with discussion and analysis of results and lastly references.

2. RELATED WORK

Several works has been done in the area of resource management and job scheduling as this is the main challenging task in Grid computing. In this section, some work done in this area is discussed. A dynamic job grouping based scheduling algorithm reduces the processing time and communication time of jobs. But the deficiencies of this algorithm are that it does not take dynamic resource characteristics into account and does not utilize resources efficiently[5]. To remove these deficiencies, Improved Resource Scheduling approach using job grouping strategy was developed. This algorithm works mainly at three levels: user level, global level and local level and then minimize the processing time of jobs thereby increasing the resource utilization but does not improve the performance of the grid system to a much extent[6]. Scheduling Framework for Bandwidth-Aware Job Grouping-based scheduling approach uses the bandwidth in scheduling framework to enhance the performance of job scheduling, but this algorithm does not utilize the resources efficiently[7]. Bandwidth Aware Job Grouping based Scheduling algorithm groups the jobs according to MIPS and bandwidth of resources thereby reducing the total job processing time but the shortcoming of this algorithm is that the model sends the group jobs to the resource whose network bandwidth has highest communication or transmission time and it does not ensure that the resource having sufficient bandwidth will be able to transfer the jobs within required time[8]. Grouping based Fine-Grained job scheduling algorithm focuses on scheduling lightweight jobs. This algorithm utilizes resources efficiently and integrates greedy and FCFS algorithm to improve undertake of fine grained jobs. But the problem of the algorithm is pre-processing scheduling time of the job and time complexity of the scheduling algorithm are high[9]. Constraint Based Job and Resource Scheduling algorithm groups the jobs taking three constraints into consideration and reduces the processing time, processing cost and enhance the resource utilization but it does not provide quality of service(QoS)[10]. To solve the problems mentioned above, an Improved Constraint Based Job and Resource Scheduling algorithm with QoS in grid computing is presented. This paper focuses on grouping the jobs taking four constraints into account and then allocating resources to the grouped jobs.

3. SCHEDULING DIAGRAM

Figure 1 demonstrates the scheduling activity of the proposed algorithm. At the beginning of the scheduling activity, open the pool of MATLAB sessions for parallel computations. Then get the information about the computer on which MATLAB software is running i.e. finding the resource memory. After that, find the available RAM memory. The CHKCPU utility provides information about CPU i.e. its speed, cache, type etc. and we find L2 cache. Then jobs are created using the co-distributor that usually gives the best performance for solving linear systems. Declare the matrix sizes ranging from 1000-by-1000 up to 45% of system memory available to each worker. We solve the linear system a few times and calculate the job MI based on the best time. Based on the above constraints calculated, the jobs are grouped according to their job memory size and job MI. Lastly, the execution time of each task and the total time required for completion of each job is also calculated.

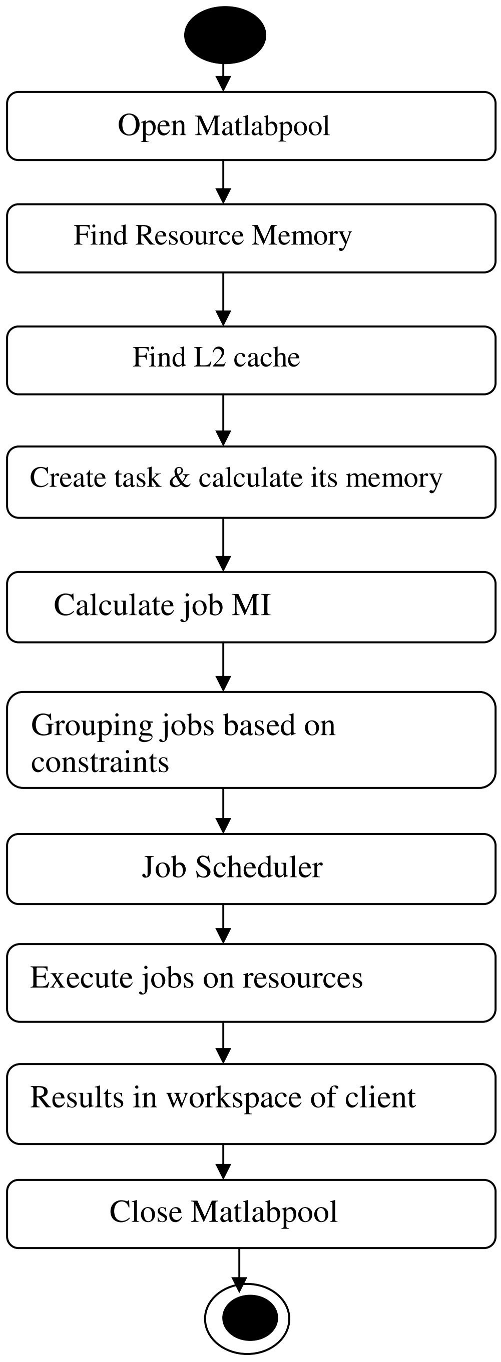
Open Matlabpool
Find Resource Memory
Find L2 cache
Create task & calculate its memory
Calculate job MI
Grouping jobs based on constraints
Job Scheduler
Execute jobs on resources
Results in workspace of client
Close Matlabpool

Figure 1. Activity Diagram for An Improved Constraint Based Resource Scheduling Approach using Job Grouping Strategy

4. PROPOSED ALGORITHM

4.1. Steps of the Algorithm

```
1.  Start parallel job that connects this MATLAB client with a number of resource in Grid using
    maltlabpool.
2.   [userview sys] = memory;               %resource ms
3.  Resource_Memory=sys.VirtualAddressSpace.Available;  %1st constraint
4.  [RESULT,RESOURCE]=system('chkcpu /v');
5.  RESOURCE is the character array last two constraints are considered as:
6.  inc1=1099;
7.  inc2=1066;
8.  for i=1:length of the constraint parameter;
9.     getting L1 Cache of cpu
10.    getting L2 cache of cpu                                  %2nd Constraint
11. end
12. Convert the cache of CPU from string to number.
13. Declare the matrix sizes ranging from 1000-by-1000 up to 45 of system memory available to
    each worker.
14. for i = 1:length of job size
15. if((maxMemUsagePerWorker<=Resource_Memory) && (gflops(i)<=Resource_cache))
16. start=tic;
17. ----------------------------------------------calling bench function-----------------------------------
    ---------------------
18. numReps = 3;                      %no. of matrix in each job
19. [A, b] = getData(n);
20. ----------------------------------------------calling getdata function-----------------------------------
    --------------------
21.  Spmd                    %Execute code in parallel on MATLAB pool
22.  codistr = codistributor2dbc(codistributor2dbc.defaultLabGrid,……
    codistributor2dbc.defaultBlockSize……..
    'col'); % It distributes matrices along two subscripts over a rectangular computational grid of
    labs
23. A = codistributed.rand(n, n, codistr); % creates an n-by-n-by-... codistributed array of
    underlying class double.
24. b = codistributed.rand(n, 1, codistr);
25. end
26. ---------------------------------------------end of getdata function-------------------------------------
    -------------------
27. time = inf;    %initailly assign time is infinity
28. for itr = 1:numReps
29.   tcurr = timeSolve(A, b);               %3rd Constraint
30. ---------------------------------------------calling timesolve function---------------------------------
    ---------------------
31. Spmd
32. tic;
33. time = gop(@max, toc);                  % Global operation across all labs and Time for all to
    complete.
```

```
end
-----------------------------------------------end of timesolve function----------------------------------
---------------------
flop = 2/3*n^3 + 3/2*n^2;            % the floating operations count of the HPC Challenge
gflops = flop/time/1e9;              %4th Constraint
-----------------------------------------------end of bench function--------------------------------------
---------------------
End;
```

4.2. Description of the Algorithm

This algorithm begins with starting a parallel job that connects the MATLAB client with a number of resources in Grid using Matlabpool. The workers are connected to the matlab according to the pool size and the amount of system memory is allocated to each worker. Then we find the amount of available and total virtual memory for the MATLAB process. The amount of memory that will be required by a resource is calculated which is the first constraint of the algorithm. The CHKCPU utility gives information about the CPU i.e. its speed, CPU mode, size of L1 cache, L2 cache etc. The second constraint of the algorithm is the L2 cache that we get from the RESOURCE variable. The L2 cache is a part of computer's processor which increases its performance. It is smaller and extremely fast computer memory. As reading and writing operation from cache memory is faster than from main memory, so to reduce the processing time, we use L2 cache of the resources in the grid as our second constraint. We convert the L1 and L2 cache values from string to numeric values. Then the tasks are created and job is a collection of individual tasks. For creating jobs that are complex and require more processing and to increase memory, we create matrices. We declare the matrix sizes ranging from 1000-by-1000 up to 45% of the system memory available to each worker. The jobs of different matrix size are created having different memory size which is the third constraint of the algorithm. Whenever matrix is created, get_data function is called. It takes the values of different matrices created above. The data is generated using the 2-D block cyclic co-distributor. Then the gflop function is used to calculate the computation time of the jobs. It is the fourth constraint of the algorithm. Once the task to be executed is defined, the jobs are grouped based on the constraints and sent to the labs based on the condition defined in for loop and in if statement. If the memory of the worker is less than or equal to the available memory used by MATLAB and if the job memory is less than the resource cache, the respective jobs are grouped and computed on the respective resource. Then the result is sent back to the client. In the last, the Matlabpool is closed. It stops the worker pool, destroys the parallel job and makes all parallel language features revert to using the MATLAB client for computing their results.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

Simulation is done to prove the proposed algorithm can work efficiently in grid environment. MATLAB is used to create the simulation environment[11]. MATLAB stands for Matrix Laboratory. Varying number of jobs is used to analyze processing time of jobs. The simulation is carried out in heterogeneous environment. MATLAB is a high performance language for technical computing. It integrates computation, visualization and programming environment. The client should have the parallel computing toolbox for grid computing. The scheduler or job manager manages the job sent from the client and sends to the worker and should have MDCE installed.

In the simulation, we performed scheduling experiments by setting different values to the number of jobs; the number of jobs is varied from 1000 to 2000 in steps of 3. The processing time is recorded to analyze the feasibility of the algorithm.

Table 1. Comparison between ICBRSJGS and CBJRS according to their processing time

No. Of Jobs	**Processing Time**	
	ICBRSJGS	**CBJRS**
1000	1.471	1.480
1473	1.180	1.312
1943	2.181	2.200

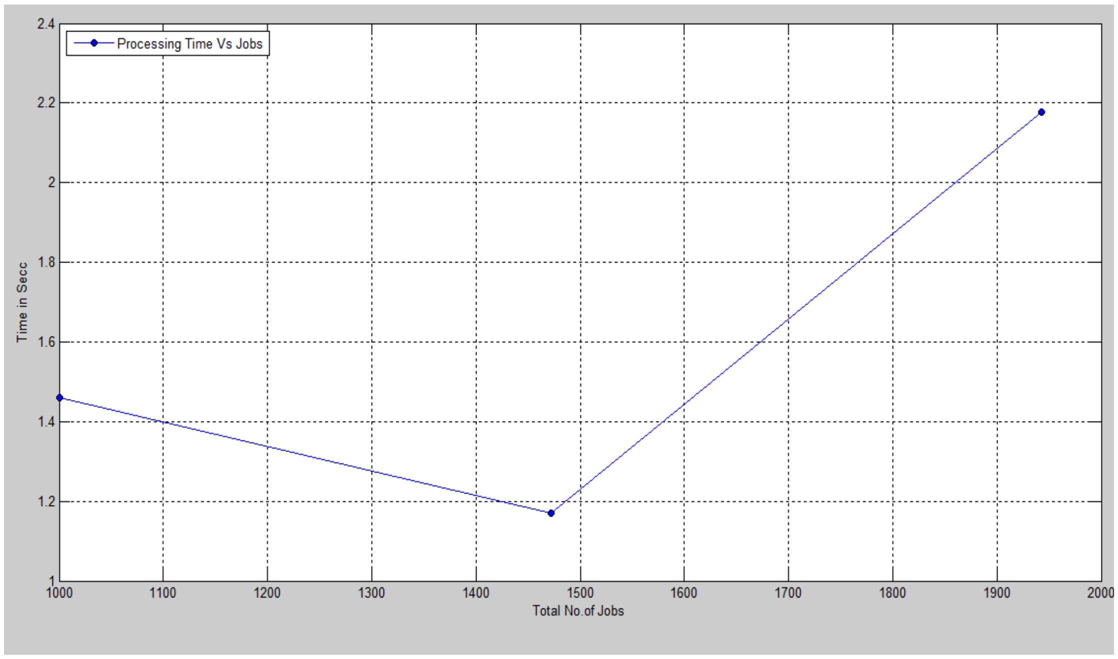

Figure 2. Processing time of the ICBRSJGS algorithm

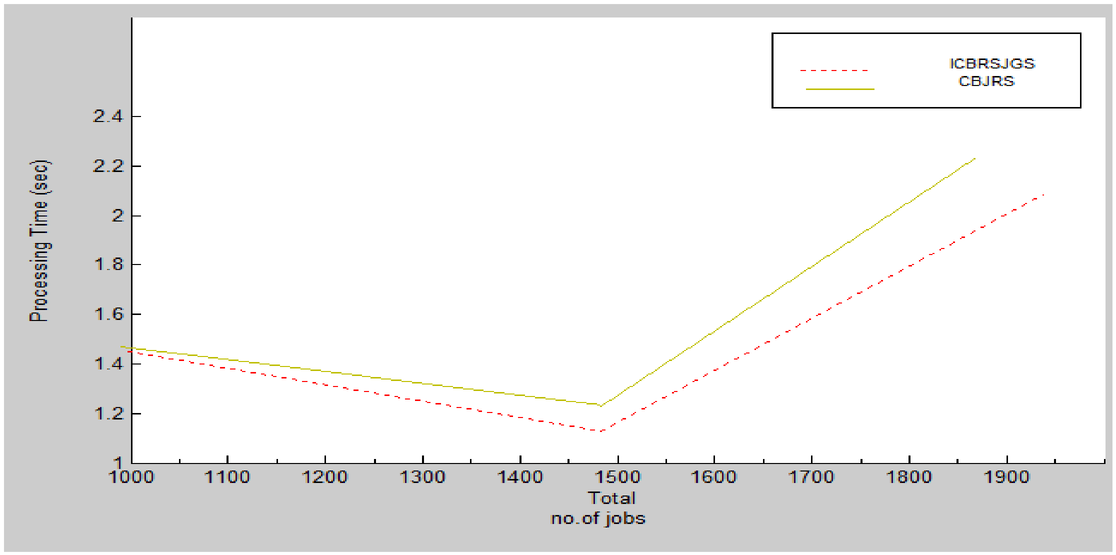

Figure 3. Processing times of the algorithms

5.2 Experimental Results

Simulation is conducted to analyze and compare the processing time of jobs between ICBRSJGS and CBJRS algorithms. Table 1 shows comparison between proposed ICBRSJGS and CBJRS algorithms in terms of processing time. Figure 2 shows implementation results in simulation environment. Figure 3 shows that the processing time decreases in proposed model when number of jobs increases as compared to CBJRS.

6. CONCLUSION

In this paper, we have proposed an Improved Constraint based job scheduling algorithm taking into account four constraints resource memory, job memory, job MI and the L2 cache. In order to reduce the processing time of the jobs instead of three, four constraints have been considered. The simulation results have shown the proposed implementation has achieved its objective in grid environment.

The implementation of CBJRS was done using only three constraints and hence reduced the processing time. The constraint resource memory i.e. main memory, it takes more time to execute from main memory. To reduce this time and for fast reading and operation of memory we have used the constraint L2 cache. The L2 cache is a part of computer's processor; it increases the performance of computer. It is smaller and extremely fast computer memory.

The results of the comparative study shows that the proposed constraint based job scheduling model using four constraints gives better performance than the model using three constraints in terms of processing time. The proposed model provides a real time grid computing environment. Additionally, the simulation environment is semi-dynamic and it can't reflect in the real computational grid environment sufficiently that promotes further research in the proposed area. In future, this work can be implemented in the real computational grid environment.

ACKNOWLEDGEMENT

We are thankful to everyone who supported and guided us in our work.

REFERENCES

[1] Foster I. and Kesselman C., “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann, 2004.

[2] Baker M., Buyya R., and Laforenza D., “Grids and Grid Technologies for Wide-Area Distributed Computing”, Software: Practice and Experience, Dec 2002, vol. 32, no. 15, pp. 1437-1466.

[3] Berman F., Hey A. J. G., and Fox G., “Grid Computing: Making the Global Infrastructure a Reality”, John Wiley & Sons, 2003.

[4] Gibbins H., Nadiminti K., Beeson B., Chhabra R., Smith B., and Rajkumar Buyya, “The Australian BioGrid Portal: Empowering the Molecular Docking Research Community”, APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch (APAC 2005), Sept. 2005.

[5] N. Muthuvelu, Junyan Liu, N.L.Soe, S.venugopal, A.Sulistio, and R.Buyya, “A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids,” in Proc of Australasian workshop on grid computing, vol. 4, pp. 41–48, 2005.

[6] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, “An Improved Resource Scheduling Approach Using Job Grouping strategy in Grid Computing”, IEEE International Conference on Educational and Network Technology, pp. 94-96, 2010

[7] Ng Wai Keat, Ang Tan Fong, “Scheduling Framework For Bandwidth-Aware Job Grouping-Based Scheduling In Grid Computing”, Malaysian Journal of Computer Science, vol. 19, No. 2, pp. 117-126, 2006.

[8] T.F Ang, W.K.Ng, T.C Ling, “A Bandwidth-Aware Job Grouping-Based Scheduling on Grid Environment”, Information Technology Journal, vol .8, No.3, pp. 372-377, 2009.

[9] Quan Liu, Yeqing Liao, “Grouping-based Fine-grained Job Scheduling in Grid Computing”, IEEE First International Workshop on Educational technology And Computer Science, vol.1, pp. 556-559, 2009.

[10] Vishnu Kant Soni, Raksha Sharma, Manoj Kumar Mishra, Sarita Das, ”Constraint Based Job and Resource Scheduling in Grid Computing”, IEEE 3rd International Conference on Computer Science and Information Technology, vol. 4, pp. 334-337, 2010.

[11] The MathWorks Inc. MATLAB 7.0 (R14SP2). The MathWorks Inc.,2005

Authors

Payal Singhal has received her B.Tech degree in CSE (Computer Science & Engineering) in 2010 from Ajmer Institute of Technology, Ajmer affiliated to Rajasthan Technical University, Kota and M.tech from Govt. Engineering College, Ajmer in 2012 (RTU-Kota). She has presented several papers in international & national conferences. Her research interest includes Grid Computing.

Mr.Ravinder Singh (Assistant Professor) received his B.E degree in C.S.E (Computer Engineering) from Sobhasaria engineering college, Sikar in 2006. M.E (CSE) from, Motilal Nehru National Institute of Technology, 2009 India. He has three years teaching experience. He is presently working as Assistant Professor (CS & IT) in Govt. engineering College, Ajmer. He has published several papers in international and national journals/conferences. He is the guide of various projects of B.Tech. students and also the guide of more than 5 students of M.Tech. thesis.

Pinky Rosemarry has received her B.Tech degree in CSE (Computer Science & Engineering) in 2010 from Ajmer Institute of Technology, Ajmer affiliated to Rajasthan Technical University, Kota and M.tech from Govt. Engineering College, Ajmer in 2012 (RTU-Kota). She has presented several papers in international & national conferences. Her research interest includes Grid Computing.