Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.1209 – 1220

RESEARCH ARTICLE

An Effective Load Balancing Grouping Based Job & Resource Scheduling Algorithms for Fine Grained Jobs in Distributed Environment

Miss. Hemangi Joshi, Prof. Viraj Daxini

[1]C.E Department, Gujarat Technological University, V.V.P Engg. College, Rajkot, Gujarat, India

[2]C.E Department, Gujarat Technological University, V.V.P Engg. College, Rajkot, Gujarat, India

[1] hemangijoshi18@gmail.com; [2] viraj_daxini34@yahoo.com

Abstract— ***Grid computing is a form of distributed computing that provides a platform for executing large-scale resource intensive applications on a number of heterogeneous computing systems across multiple administrative domains. Therefore, Grid platforms enable sharing, exchange, discovery, selection, and aggregation of distributed heterogeneous resources such as computers, databases and visualization devices. Job and resource scheduling is one of the key research area in grid computing. In a grid computing environment, a scheduler is responsible for selecting the best suitable computing resources in the grid for processing jobs to achieve high system throughput. Further, grouping the fine grained jobs according to the processing capability of available resources results in better throughput, resource utilization and low communication time. This paper focuses on lightweight jobs scheduling in Grid Computing. Therefore, this paper proposes "An Effective Load Balancing Grouping Based Job & Resource Scheduling Algorithms for Fine Grained Jobs in Distributed Environment" with the objective of minimizing overhead time and computation time, thus reducing overall processing time of jobs. The work is verified through various observations made in simulated grid environments. The results obtained shows that the proposed grouping-based scheduling algorithm is on average or comparable to other grouping based scheduling algorithms.***

Keywords— ***grid computing; job grouping; scheduling; load balancing; fine grained jobs***

I. INTRODUCTION

This The growing popularity of the Internet/Web and the Availability of powerful computers and high-speed Networks at low-cost commodity components are Changing the way we do computing and use computers (8).The interest in coupling geographically distributed Resources is also growing for solving large-scale Problems. The word 'grid' was first coined in the mid-1990s to denote a proposed distributed computing infrastructure for advanced science and engineering projects(8). Computational Grid can be defined as large-scale high-performance distributed computing environments that provide access to high-end computational resources. And also it is defined as a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, performance, capability, cost, and user's quality-of-service requirements.

Computational grids provide large-scale resource sharing, such as personal computer, clusters, MPPs, Data Base, and online instructions, which may be cross domain, dynamic and heterogeneous(14). Grid computing needs to support various services security, uniform access, resource management, job scheduling, application composition, Computational economy and accounting (14).

The concept of a computational grid is intended to use high-performance network technology to connect hardware, software, instruments, databases, and people into a seamless web that supports a new generation of computationally rich problem-solving environments for scientists and engineers. Resource sharing therefore is the essence of grid computing. Grid scheduling is the process of scheduling jobs over grid resources. A grid scheduler is in-charge of resource discovery, grid scheduling (resource allocation and job scheduling) and job execution management over multiple administrative domains.

In a Grid computing environment, scheduler is responsible for selecting the suitable machines or computing resources in Grid for processing jobs to achieve high system throughput, but there exist Various scientific and business organizations tend to have increased number of applications with large number of independent jobs, scheduling of these jobs onto the grid is significantly more difficult and complicated than scheduling applications in traditional supercomputer because of the heterogeneous ,dynamic and diverse nature of the Grid resources. Therefore, optimal scheduling of various jobs onto grid is not easy to attain, since optimal scheduling of heterogeneous jobs in heterogeneous environments is known to be NP-Complete problem.

In order to ensure the efficiency and better performance of job scheduling, an effective and near optimal scheduling mechanism has to be developed and implemented to cater the needs of the grid users So to achieve high performance, jobs are to be Scheduled in group instead of light weight jobs. The main purpose of this paper is to develop an efficient job scheduling algorithm to maximize the resource utilization and minimize processing time of the jobs, how they are grouped and allocated to resources in dynamic environment.

II. RELATED WORK

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it. In this section, the various job grouping algorithms proposed in literature for job scheduling in grid environment are discussed.

A scheduling optimization method should consider the following two aspects, one is the application characteristics, and the other is the resource characteristics. Taking into account the characteristics of lightweight job, there are some researches on the fine-grained job scheduling problem.

Hierarchical job scheduling approach used two levels Scheduling [4]global scheduling & local scheduling .The global scheduler uses separate queues for different type of the jobs for scheduling with the FCFS,SJF and first fit (FF) and the local scheduler uses same queue for different type of the jobs.

A dynamic job grouping-based scheduling algorithm [5], groups the jobs according to MIPS (Million Instructions per Second) of the available resources. The proposed job scheduling strategy takes into account: *(i)* the processing requirements for each job, *(ii)* the grouping mechanism of these jobs, known as job grouping, according to the processing capabilities of available resources, and *(iii)* the transmitting of the job grouping to the appropriate resource. This model reduces the processing time and communication time of jobs, but the algorithm doesn't take the dynamic resource characteristics into account and the grouping strategy can't utilize resource sufficiently.

Scheduling framework for Bandwidth-Aware Job Grouping Based strategy [6], groups the jobs according to MIPS and bandwidth of the resource. The principle behind the bandwidth-aware scheduling is the scheduling priorities taking into consideration not only their computational capabilities but also the communication capabilities of the resources. The bandwidth-aware scheduling approach uses the network bottleneck bandwidth of resources to determine the priority of each resource. But the deficiencies of the algorithm are first, grouping strategies does not utilize resource sufficiently, and second, consideration of bandwidth strategy is not efficient to transfer the job.

Optimal Resource Constraint (ORC) scheduling algorithm [7] includes the combination of both the Best fit allocation and Round Robin scheduling to allocate the jobs in queue pool. This algorithm improved the efficiency of load balancing and dynamicity capability of the grid resources.

In Quan Liu, Yeqing (2009) Liao, the fine-grained jobs [8] are grouped into forming coarse-grained jobs and allocated to the available resources according to their processing capabilities in MIPS and bandwidth in Mb/s. The grouping algorithm integrates Greedy algorithm and FCFS algorithm to improve the processing of Fine-grained jobs. Algorithm maximizes the resource utilization and reduces the total processing time. But the problem of the algorithm is preprocessing scheduling time of the job is high, time complexity of the scheduling algorithm is high and finally, it does not give any attention to the memory requirement of file-size.

A Bandwidth-Aware Job Grouping-Based scheduling strategy [9], that groups the jobs according to the MIPS and bandwidth of resources, but shortcomings of the algorithm is first, the model sends group jobs to the resource whose network bandwidth has highest communication or transmission rate, but the algorithm does not ensure that resource having a sufficient bandwidth will be able to transfer the group jobs within required time.

An agent based dynamic resource scheduling strategy [11] focuses on maximize the processing time of jobs. The process of selecting a job is based on maximum heap tree. The processed outsource model is a hierarchical two layer approach in which top layer is called grid level another is called cluster level. In this model with FCFS-Job Grouping Strategy has of two major parts, first part provides resource management, which selects highest computational power cluster at the grid level and the second part provides FCFS-job grouping strategy that makes efficient utilization of the selected cluster by submitting a matching group of jobs from a FCFS job queue. But this paper does not consider bandwidth and file size constraint except computational power of the resource.

In Constraint-Based Job and Resource scheduling (CBJRS) algorithm [12] grouping is done based on processing capability (in MIPS), bandwidth (in Mb/s), and memory-size (in Mb) of the available resources. The resources are arranged in hierarchical manner where Heap Sort Tree (HST) is used to obtain the highest computational power resource or root node, so as to make balanced and effective job scheduling.

Memory aware job scheduling Model [13] presents and evaluates an extension to Computational-Communication Memory size based job grouping scheduling strategy that tries to maximize the utilization of Grid resources and their processing capabilities, and also reduces processing time and network delay to schedule and execute the jobs on the Grid. The proposed job scheduling is based on job grouping concept taking into account memory constraint together with other constraints such as processing power, bandwidth, expected execution and transfer time requirements of each job.

This study focuses and evaluates an extension to dynamic job grouping based scheduling, which aims to reduce overall processing time of applications by minimizing the job allocation overhead and computation time.

III. GRID SYSTEM: GENERIC VIEW

A. Grid Model

A generic grid computing system infrastructure G in this work is assumed to be the collection of R heterogeneous resources and participation of set of U users connected over high-speed interconnection networks. More formally, the collection of resources is represented as {R1, R2…, Rr}. Each resource Ri owns a set of computing nodes, represented as R= {C1, C2…, Cc} and belongs to a local domain, i.e. a LAN (Local Area Network). Each computing node Cj is composed of a number of processors Pk and represented as C= {P1, P2…, Pp}.Where i≥ 1, j≥ 1and k≥ 1. Hence, the resulting grid resource can be a computing system having a single processor, shared memory multiprocessors (SMP)/ Parallel Processing Systems (PPS), or a distributed computing system/distributed memory cluster of computers.

Multiprocessor systems are defined as computer systems that have several processors sharing a single set of peripherals including the memory. These processors are not autonomous. Distributed Computing Systems, consist of several autonomous processors having their own operating system, with their own local policies. A single processor system or SMP type grid resource provides time-shared environment and the distributed memory multiprocessor systems such as clusters provides a space-shared environment. A cluster/multiprocessor system consists of P processors and let the computation speed of processor j is equal to mj units. A common unit for measuring capacity can be specified in terms of the rating of standard benchmarks such as millions of instruction per second (MIPS).

$$\text{MIPS}= \sum_{k=1}^{p} P_k$$

The total Capacity of a computing node is defined as Cj in and that of Grid resource is defined as Ri :

$$\text{MIPS}= \sum_{j=1}^{c} C_j$$

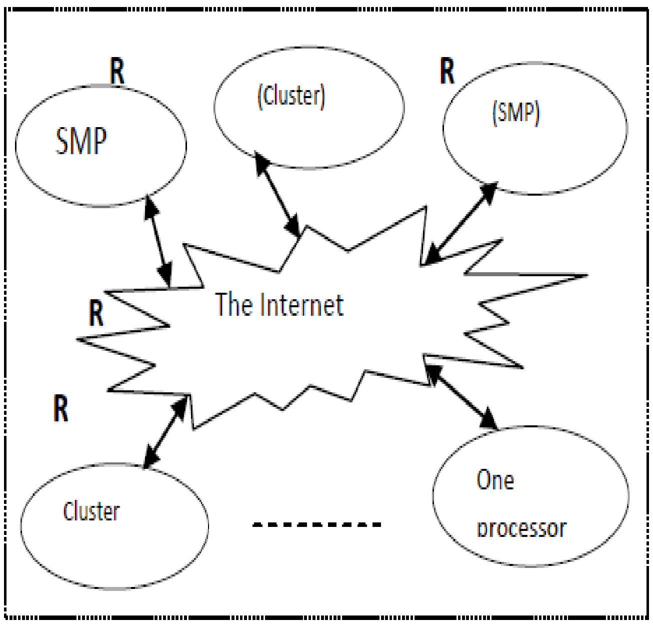

Figure 1: Resources connected across network[11](R denotes combined resources in MIPS available in a cluster)

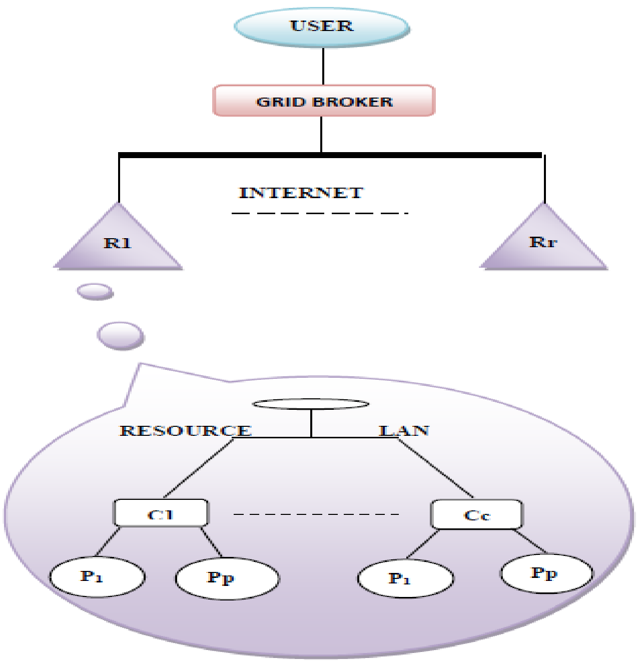

Figure 2: Basic grid model

The Grid Scheduler: The main objective of a scheduler in most systems often is to design a scheduling policy for mapping of submitted jobs to the resources with the goal of maximizing throughput, efficiency, resource utilization, minimizing job completion times, communication overhead and cost or both time-cost etc. In order to achieve the above mentioned objective in high performance computing environment like Grid should provide a comprehensive and versatile environment and a well-defined set of steps to tackle the process of scheduling, which is described in the following steps:

Resource discovery: One of the most and first important goals of the scheduler is to identify a list of authorized resources that can be made available to the users.

Grid Information Service: Most Scheduling algorithms interact with grid information service (GIS) to obtain the initial list of authorized resources, called resource pool. Some of these resources might be meeting the certain minimum requirements of the application such as hardware platform, operating system, RAM or secondary storage space, etc.

Resource selection: Once the information regarding the available resources in the resource pool is obtained, the next task of the scheduler is to select those resources that are expected to meet time. To facilitate user's requirements the scheduler has to gather dynamic information about resource accessibility, system workload, network performance, and price of the resources etc.

Job scheduling: The job scheduling problem is defined as the process of making decision for scheduling set of independent jobs onto best possible matching dynamic resources and services that satisfies requirements of jobs and the constraints imposed by users. In grid computing environment, this is the stage where jobs of applications are allocated to selected physical resources based on user's requirement, resource availability and grid facilities. The scheduling in grid environment has to satisfy a number of constraints on different problems. So, optimal scheduling is a NP-complete problem [5] and different heuristics may be used to reach an optimal or near-optimal solution [14].

Monitoring and reliability: Geographically diverse and dynamic nature of grid makes it difficult to reach the objectives where environmental conditions are subject to unpredictable changes such as system or network failures, system performance degradation, addition or deletion of machines, variations in the cost and computing capability of resources, etc. That is why, it is most important to monitor the tasks running on the nodes of the grid over time and to check reliability factors.

Designing of the Scheduler: Grid scheduling problem can formally be represented by a set of the given jobs, user and resources. Designing of the scheduler involves matching of application needs of the users with availability of the suitable resources and addressing the concern of the quality of the match.

Problem Formulation: An instance of the problem consists of a registered user, number of jobs, resources, MI of the jobs and MIPS of the resources.

B. Basic Job Grouping Framework

When the user creates a list of jobs in the user machine, these jobs are sent to the job scheduler for scheduling arrangement. The information collector gathers resource information from the Grid information service (GIS). The grid information service (GIS) is a facility that provides information about all the registered resources in a grid. Based on the information, the job scheduling algorithm is used to determine the job grouping and resource selection for grouped jobs. Once all the jobs are put into groups with selected resources, the grouped jobs are dispatched to their corresponding resources for computation.

The grouping and selection service serves as a site where matching of jobs is conducted. The strategy for matching jobs is based on the information gathered from the information collector. There are two steps involved during the matching of jobs. They are job grouping and job selection. In the job grouping process, jobs submitted by the user to the scheduler are collected and they are grouped together based on the information of resources. The size of a grouped job depends on the processing requirement length expressed in Million Instructions (MI). At the same time, job selection is also being conducted where a grouped job corresponds to the resource in question. The process is performed iteratively until all the jobs are grouped according to their respective resources. The dispatcher functions as a sender that transmits the grouped jobs to their respective resources. The dispatcher forwards the grouped jobs based on the schedule made during the matching of jobs with resources. The dispatcher also collects the results of the processed jobs from the resources through input ports.

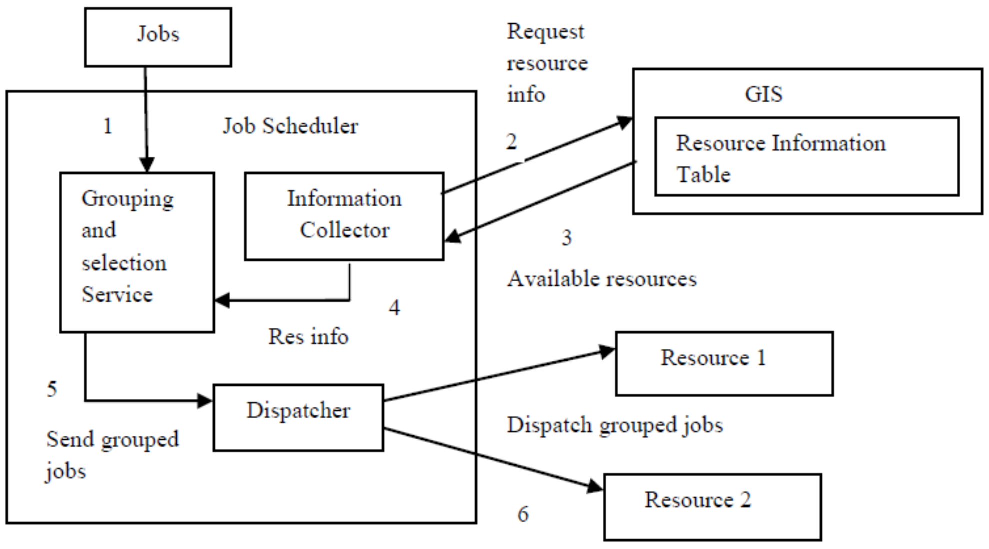

Figure 3: Framework For Job Scheduling

IV. **JOB GROUPING AND RESOURCE SELECTION STRATEGY**.

A. Grouping strategy for jobs

In this paper is based on a resource selection and job grouping strategy. Jobs are grouped according to the ability of the selected resource. Therefore, during job grouping the following condition must be satisfied:

GROUP_JOB_MI<=RESOURCE_MIPS*EXPECTED_EXECUTION_TIME (1)

Where, MI (Million Instruction) is job's required computational power, MIPS (Millions Instruction per Second) is processing capability of the resources and Expected_Execution_Time is user defined time, used to measure total amount of accumulated MI of jobs that can be completed within a specific time window are put together into a group. Expected_Execution_Time is much like granularity size defined in[5].

B. Overall Processing Time of Job Calculation

To evaluate the total processing time of an application, an analytical performance model is defined in terms of overhead time and computation time of the grouped jobs.

The total processing time is calculated in seconds based on the overhead time for processing each Jobs, and the time taken for performing Job (Gridlet) grouping process, sending Jobs to the resources, processing the Jobs at the resources and receiving back the processed Jobs. This time computation is depicted in Figure 4. In real world, the overhead time for each job depends on the current network load and speed

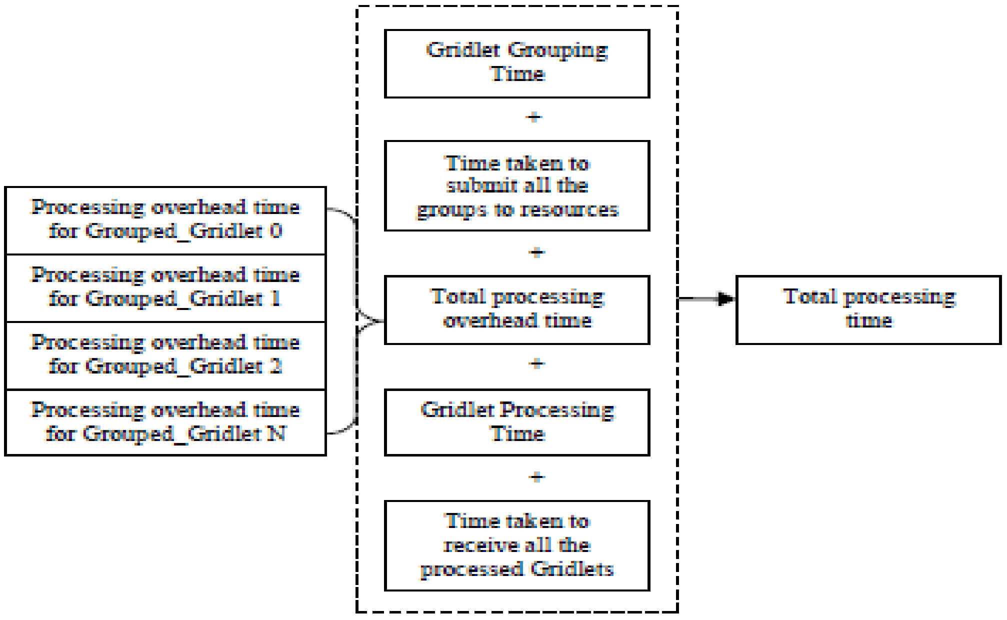
Gridlet Grouping Time
+
Time taken to submit all the groups to resources
+
Processing overhead time for Grouped_Gridlet 0
Processing overhead time for Grouped_Gridlet 1
Processing overhead time for Grouped_Gridlet 2
Processing overhead time for Grouped_Gridlet N
Total processing overhead time
+
Gridlet Processing Time
+
Time taken to receive all the processed Gridlets
Total processing time

Figure 4: Processing time[5]

C. Resource Utilization Calculation

Resource Utilization: To Maximizing the resource utilization of the Grid system is another important objective. Resource utilization in terms of computational capability of Resource can defined as follows:

RESOURCE UTILIZATION =(TOTAL PROCESSING LOAD AT RESOURCE/ TOTAL PROCESSING POWER OF RESOURCE)*100 (2)

By using above formula, we can find Resource utilization of overall system.

D. Scheduling Strategy

The resources and jobs are sorted out in descending order of their processing power and job's computational requirement respectively. The resources are taken one after another in FCFS order from the reverse sorted resource list. Once a resource is selected in this manner, jobs are added into job group according to the processing capability of this resource by alternatively taking jobs from front end i.e. job with higher computational requirement and then rear end of the job list i.e. job with smaller computational requirement. It should be noted that the front end and rear end pointers are updated to point to the next job in the list. While grouping the jobs taken in order of the grouping strategy from the job list, the above strategy falls into one of the two cases given below.

Case I. At some stage in grouping, if given condition in eq. 1 fails, while adding a job into the group from front end of the job list, then it is removed from the group and if possible jobs are taken from the rear end of the list till it satisfies the eq. 1.

Case II. If condition in eq. 1 fails while adding a job from rear end of the job list during the grouping operation, then it stops grouping for that resource removing the last job that was added and sends the job group to the dispatcher. Then, it takes next highest resource to perform another job grouping. The above job grouping process is repeated for each resource taken in given order as long as either Resources or Job exist.

When there is no more resource left, first come first served (FCFS) algorithm is used, once a resource node finishes its job, the mediator will get the status of the idle resource, and it will be assigned a new grouped job to that newly discovered resource.

E. Sorting Algorithm for Resource and Job list

1). Job Sorting: Light weight Jobs are much in count than resource so best mechanism for sorting that large number of jobs' list is necessary

When deciding on the best sorting algorithm we often look at its worst-case running time, and base our decision solely on that factor. So quick sort as a viable option because of its T(n^2) worst-case running time, which could be made exponentially unlikely with a little effort. In fact, quick sort is the currently fastest known sorting algorithm and is often the best practical choice for sorting, as its average expected running time is O(n log(n)).

Quick sort is a divide and conquer algorithm which relies on a partition operation: to partition an array, we choose an element, called a pivot, move all smaller elements before the pivot, and move all greater elements after it. This can be done efficiently in linear time and in-place. We then recursively sort the lesser and greater sub lists.

```
function quicksort(array)
    if length(array) ≤ 1
        return array  // an array of zero or one elements is already sorted
    select and remove a pivot element pivot from 'array'  // see '#Choice of pivot' below
    create empty lists less and greater
    for each x in array
        if x ≤ pivot then append x to less
        else append x to greater
    return concatenate(quicksort(less), list(pivot), quicksort(greater)) // two recursive calls
```

Quick Sort for Sorting Jobs

2.) Resource Sorting: Resources are comparatively small in number than job . so to sort the resources in descending order we are using another sorting technique heap sort.

Heap sort is a much more efficient version of selection sort. It also works by determining the largest (or smallest) element of the list, placing that at the end (or beginning) of the list, then continuing with the rest of the list, but accomplishes this task efficiently by using a data structure called a heap, a special type of binary tree. Once the data list has been made into a heap, the root node is guaranteed to be the largest element. It is removed and placed at the end of the list, then the heap is rearranged so the largest element remaining moves to the root . Using the heap, finding the next largest element takes O(log n) time, instead of O(n) for a linear scan as in simple selection sort. This allows Heap sort to run in O(n log n) time.

F. Algorithm of Job Scheduling

1. The scheduler receives the job_List, Job [ ] created by the user.
2. Sort the jobs according to their Job_Processing_Requirement(Job_MI) in decreasing order.
3. The scheduler receives the Resource_List, Res [ ].
4. Sort the Resources in descending order according to their MIPS.
5. Get the MIPS of the Resource I and set resource id number I =1.
6. Calculate Resource_MI=Resource_MIPS *EXPECTED_EXECUTION_TIME.
7. Set the Grouped_job_MI to zero.
8. Head and Tail pointer pointing to the first and last Jobs in the job_List respectively

```
9. While (Grouped_job_MI <= Resource_MI, and if there are ungrouped jobs in the job_List ,job[] and
   unused Resources in Resource_List Res[ ] )
BEGIN
    • Set Grouped_Job_MI to the summation of previous Grouped_job_MI and current job_MI pointed
      by the head pointer.
    • If Grouped_job_MI <=to Resource_MI
        ◦ update the head pointer.
    • else
        ◦ Remove the last Job added to the Grouped_job_MI.
    • Now Set Grouped_Job_MI to the summation of previous Grouped_Job_MI and current Job_MI
      pointed by the tail pointer
    • If Grouped_Job_MI< Resource_MI
        ◦ update tail pointer.
        ◦ Go to BEGIN
    • else
        ◦ Remove the last Job added to Grouped_Job_Length.
        ◦ Stop grouping.
END.
10. Set a new ID for the Grouped_ Job.
11. Submit the Grouped_job to Resource, R[i].
12. Increment i (for next resource in Resource-List) and goto step 5.
13. If there are still more jobs after grouping and if there is not any resource free than job will go to waiting
    list after that whenever any resource will be free, will provide to jobs in which are in waiting list with
    the same following above grouping procedure.
```

Explanation**:** After receiving the Job list and resource list, the Jobs and resources are sorted according to the decreasing order of their MI and MIPS respectively by the scheduler. Then, the resources are selected one by one in FCFS order from the reverse sorted resource list and their equivalent MIs are obtained by multiplying with the given Expected_Execution_Time. After that the grouped job MI is initialized to zero and the two pointers head and tail are initialized to point to the first job and last job of the Job list respectively. The grouping strategy is implemented in step-9 While adding and extracting the jobs to and from the group, the head and tail pointers are updated accordingly. After that the resource ID will be set for the grouped Job and is sent to the corresponding resource for computation. After that next resource is selected for another grouped job. The above grouping-based scheduling of jobs continues till jobs and resources are available.

V. EXPERIMENTAL SETUP

A. Implementation with Gridsim Tool kit

Here Figure 4 depicts the simulation strategy of the proposed dynamic job grouping-based scheduler which is implemented using the GridSim simulator.

The system accepts total number of user jobs, processing requirements or average MI of those jobs, allowed deviation percentage of the MI, processing overhead time of each user job on the Grid, Expected_Execution_Time of the job grouping activity and the available Grid resources in the Grid environment (step 1-3). Details of the available Grid resources are obtained from Grid Information Service entity that keeps track of the resources available in the Grid environment. Each Grid resource is described in terms of their various characteristics, such as resource ID, name, total number machines in each resource, total processing elements (PE) in each machine, MIPS of each PE, and bandwidth speed. In this simulation, the details of the

Grid resources are store in a file which will be retrieved during the simulations. After gathering the details of user jobs and the available resources, the system randomly creates jobs according to the given average MI and MI deviation percentage (step 4). The scheduler will then select a resource and multiply the resource MIPS with the given granularity size (step 5). The jobs will be gathered or grouped according to the resulting total MI of the resource (step 6), and each created group will be stored in a list with its associated resource ID (step 7). Eventually, after grouping all jobs the scheduler will submit the job groups to their corresponding resources for job computation (step 8).

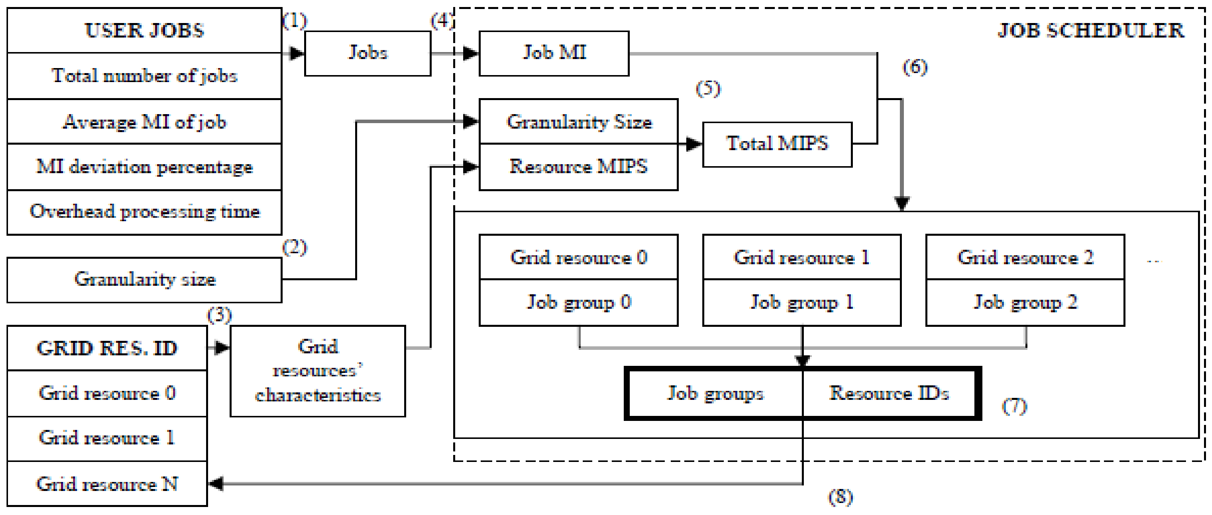

Figure 5: The simulation strategy for dynamic job grouping-based scheduler[5]

GridSim has been used to create the simulation of grid computing environment. The simulation is conducted in heterogeneous environment to verify the improvement of proposed model over other scheduling models. The scheduling algorithm is implemented on a laptop with Core 2 Duo processor and 2 GB RAM. The inputs to the simulations are total number of randomly generated jobs with and MI deviation percentage, resource MIPS, different Expected_Execution_Time and Job processing overhead time. Simulations are conducted using five resources of different MIPS, where each resource is composed of some machines and each machine contains one or more processing elements (PEs). Resources associated with different Expected_Execution_Time such as 10, 20, 30 are taken for various observations. The details of the resource list are shown in table-1.

The MIPS of each resource is computed as follows:

*Resource_MIPS=Total_PE*PE_MIPS*, where *Total_PE*=Total number of PEs at the resource, *PE_MIPS=MIPS of PE.*

Table -1 Resource its Mips and Expected_Execution_Time

Resource Name	**No of Nodes, (PEs)**	**Expected_Execution_Time**	**Resource MIPS**
R1	1(2)	10-30	100
R2	2(3&5)	10-30	400
R3	3(2&3&5)	10-30	500
R4	1(4)	10-30	200
R5	2(6,1)	10-30	350

In this simulation, the total processing time is obtained in sim seconds by adding together the overhead time and computation time of the each grouped Job. The processing overhead time of each grouped Job is set to 10 seconds. The purpose of this Simulation is to analyze and compare the performance difference between two scheduling algorithms**:** “A Dynamic Job Grouping-Based Scheduling” (DJGBS) and proposed “An Effective Load Balancing Grouping Based Job & Resource Scheduling Algorithms For Fine Grained Jobs In Distributed Environment” (ELBGJRSF). The scheduling and grouping strategy of DJGBS is adopted from [5].

B. . Simulation Results

Using gridsim toolkit, simulation environment is created to study the behavior and performance of the proposed grouping-based scheduling approach in comparison to DJGBS in heterogeneous and dynamic grid environments. The results are obtained through various observations under all possible grid environments with different Expected_Execution_Time and different MI percentage deviation. As is illustrated total MIPS is the main factor to constrain the sizes of coarse-grained jobs.

Simulation Environment: (Jobs are created with deviation of 20%) In this simulation jobs are created with average MI of 200 and deviation of 20%. Expected_Execution_Time of 10, 20 and 30 are taken for observation 1, observation 2 and observation 3 respectively to analyze the processing time of submitted Jobs. Observation 4 is performed over different Expected_Execution_Time to study the processing time of two scheduling algorithm with average MI of 200.

Observation:-1

The Figure-4 given below represents the processing time of two algorithms for the Expected_Execution_Time 10 with different user jobs of average MI 200 with deviation of 20%.

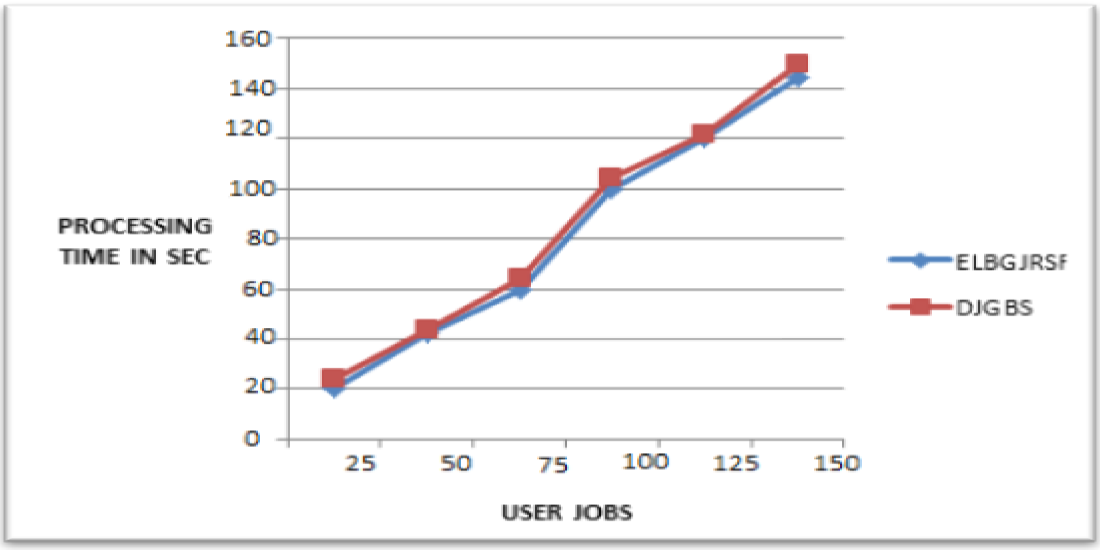

Figure 6: Processing time with Expected_Execution_Time=10

Observation:-2

The Figure-5 corresponds to the processing time of two algorithms for the Expected_Execution_Time 20 with different user jobs of average MI 200 with a deviation of 20%.

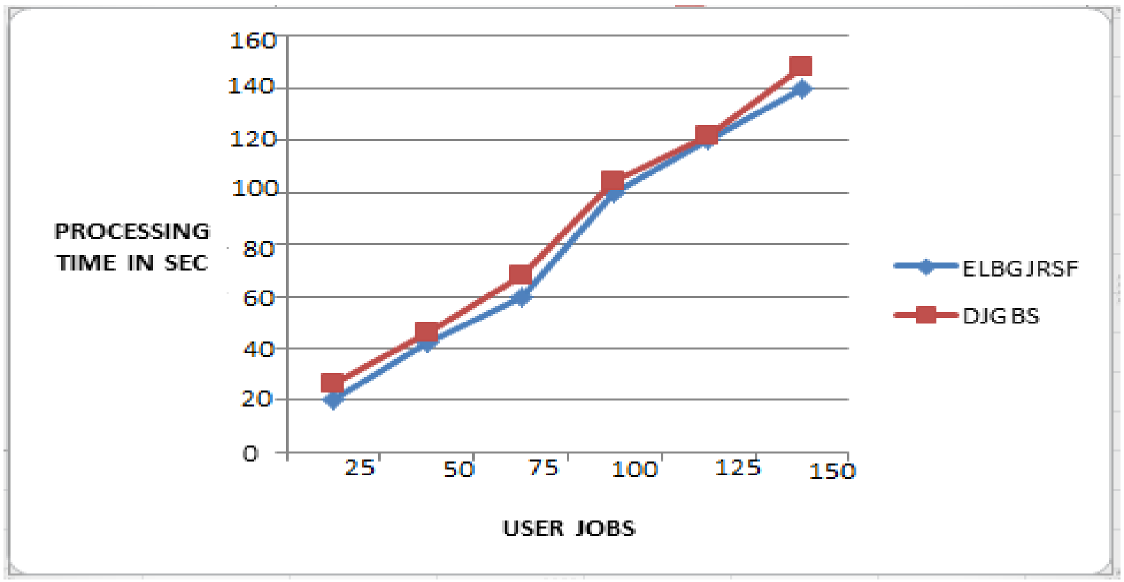

Figure 7: Processing time with Expected_Execution_Time=20

Observation:-3
The observation in Figure-6 signifies the processing time of two algorithms for the Expected_Execution_Time 30 with different user jobs of average MI 200 with deviation of 20%.

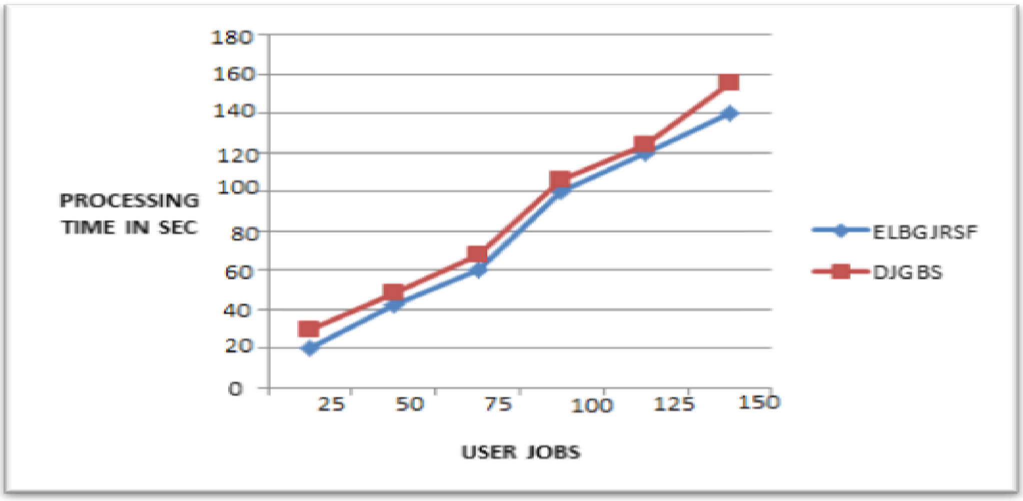

Figure 8: Processing time with Expected_Execution_Time=3

Observation:-4
The Figure-7 illustrates the behavior of ELBGJRSF and DJGBS algorithms in terms of processing time for 150 Jobs with different Expected_Execution_Times The performance gain of the proposed scheduling algorithm over DJGBS in simulation environment is from 4.7% to20.9% in terms of processing time.

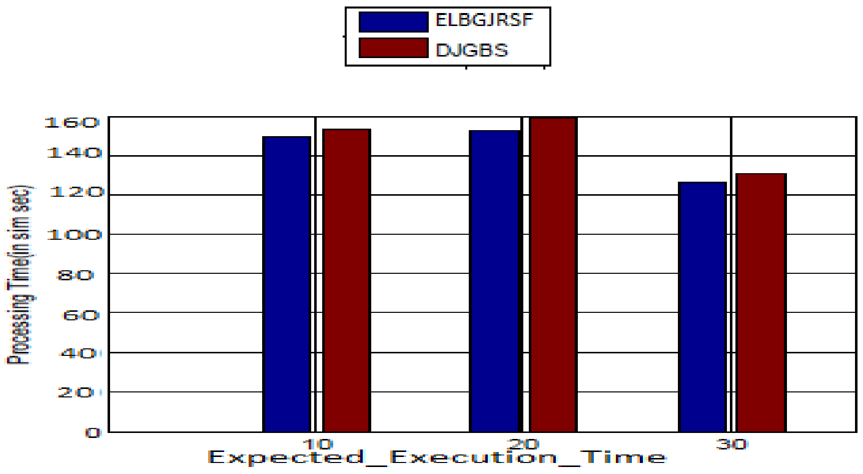

Figure 9. Processing time with different Expected_Execution_Time

VI. CONCLUSIONS

The job grouping strategy results in increased performance in terms of low processing time and cost if it is applied to a Grid application with a large number of jobs where each user job holds small processing requirements. Sending/receiving each small job individually to/from the resources will increase the total communication time. In addition, the total processing capabilities of each resource may not be fully utilized each time the resource receives a small scaled job. Job grouping strategy aims to reduce the impact of these drawbacks on the total processing time. The strategy groups the small scaled user jobs into few job groups according to the processing capabilities of available Grid resources. This reduces the communication overhead time and processing overhead time of each user job.

ACKNOWLEDGMENT

I wish to thank all who helped me in this paper .I thank my Head of the department **Prof. Tejas P. Patalia** for helping me in sorting out the procedural work and his guidance. I also thank **Asst. Prof. Viraj Daxini** for his kind support and guidance during paper preparation. Books, Internet and computing facilities have been a treasure in developing this paper. So, words of gratitude for the staff of library and computer department of V.V.P Engineering College, Rajkot.

REFERENCES

[1] Ian Foster and Carl Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Elsevier Inc., Singapore, Second Edition, 2004.

[2] GridSim 5.0 Beta Application Programming Interface (API). http://www.buyya.com/gridsim/doc/api/.

[3] Raj Kumar Buyya and Srikumar Venugopal." A gentle introduction to grid computing and tech- neologies". CSI Communications, 29(1):9{19, July 2005. Computer Society of India (CSI).

[4] J. Santoso; G.D. van Albada; B.A.A. Nazief and P.M.A. Sloot:"Hierarchical Job Scheduling for Clusters of Workstations", ASCI 2000, pp. 99-105. ASCI, Delft, June 2000.

[5] Muthuvelu. N, Liu. J, Lin Soe. N, Venugopal. S, Sulistio. A and Buyya. R, 2005, "*A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids*", in Proceedings of Australasian Workshop on Grid Computing and e-Research (AusGrid2005), , pp. 41-48,2005.

[6] 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

[7] K.Somasundaram, S.Radhakrishnan, "Node Allocation In Grid Computing Using Optimal Resource Constraint (ORC) Scheduling",VOL.8 No.6, IJCSNS International Journal of Computer Science and Network Security, June 2008

[8] 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

[9] 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

[10] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan ,"A Survey of Job Scheduling and Resource Management in Grid Computing", World Academy of Science, Engineering and Technology 40 2010

[11] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra,"An Agent Based Dynamic Resource Scheduling Model With FCFS-Job Grouping Strategy In Grid Computing", Waset, ICCGCS 2010.In Press.

[12] M.K.Mishra,V.K.Soni,R. Sharma,Sarita Das,***"**Constraint-Based Job and Resource scheduling in Grid Computing**"*** ,3rd International Conference On Computer Science and Information Technology,IEEE,2010

[13] M.K.Mishra, R. Sharma, V. K. Soni, B. R. Parida, R. K. Das(2010): A Memory-Aware Dynamic Job Scheduling Model in Grid Computing. *International Conference on Computer Design and Applications, 2010 IEEE*, vol.1-545.

[14] P. Rosemarry, R.Singh, P.Singhal, And D. Sisodia,"Grouping Based Job Scheduling Algorithm Using Priority Queue And Hybrid Algorithm in Grid Computing." International Journal of Grid Computing & Applications (IJGCA) Vol.3, No.4,December 2012.