Journal of Computer Science 9 (5): 660-670, 2013
ISSN 1549-3636

doi:10.3844/jcssp.2013.660.670 Published Online 9 (5) 2013 (http://www.thescipub.com/jcs.toc)

Dynamic Souple Wireless Grid Applications for Horde of Jobs by Sensible Centrality Scheduling with Redite

Vijaya Karthick, P. and V. Vasudevan

Department of Information Technology, Kalasalingam University, Srivilliputhur, India

Received 2013-04-22, Revised 2013-05-17; Accepted 2013-05-28

ABSTRACT

The Grid Computing has emerged as a thorny platform to tackle numerous large-scale issues, particularly in science and engineering domains. One of the primary issues related to the economical and effective utilization of heterogeneous resources in a Grid scheduling. It is mainly due to the dynamic nature of grid. Grid scheduling could be subtle higher cognitive process that operates at totally different levels of grids. Grid Schedulers is employed to map user's job to resources in keeping with their necessities. There are handful programming mechanism for grid environment the realistically wear down this dynamic nature in literature. In this study, Sensible Centrality Scheduling is used to deal with the programming computationally intensive Horde of Jobs (HOJ) applications. Their common and first aim is that they create planning choices while not totally correct performance prediction information. Another purpose to notice is that this Sensible algorithm adopts redite (needless replication) jobs. Our analysis study employs variety of experiments with numerous simulation settings. The results show the efficiency and aggressiveness of our algorithms in comparison to existing ways and we proved that is sensible centrality algorithm is the best algorithm.

Keywords: Grid Computing, Horde of Jobs, Grid Scheduling

1. INTRODUCTION

The Grid allows the development of a virtual computing system that interconnects across worldwide heterogeneous computing systems with a spread of resources. Here, resources refer not solely to physical computers, networks and storage systems however conjointly to abundant broader entities like databases, knowledge transfer and simulation (Casanova *et al*., 2008). The grid makes an attempt to with efficiency integrate various resources that the users will access transparently, as if they're native resources. Therefore, it provides a additional powerful setting compared to the user's native computing system. Additionally to its jobs capability, it is a more cost-effective way in comparison to alternative dedicated superior computer systems.

The Grid has emerged as a concrete platform to tackle large-scale issues, with associate degree increasing range of applications in wide areas being developed and ported to grid surroundings. There are two typical application models that are very famous are Horde-of-Jobs (HoJ) parameter sweep and workflow. A HoJ application consists of freelance tasks and, thus, no specific order of task execution, whereas associate degree application within the advancement model consists of mutual list of tasks. The Horde-of-Jobs (HoJ) applications can be any classified into computationally intensive and knowledge intensive. In this research work, HoJ applications are mentioned as specific interest. HoJ applications are normal parallel type of applications that exist in several scientific and engineering fields, like the essential native Alignment Search Tool (BLAST)

Corresponding Author: Vijaya Karthick, P., Department of Information Technology, Kalasalingam University, Srivilliputhur, India

(Montagnat *et al*., 2008), MCell (Blanquer *et al*., 2005), INS2D (Magnin and Montagnat, 2006) and many data mining applications. Since tasks in a very HoJ application are able to run severally and at a particular time, distributed computing systems like grids are appropriate to run such applications (Casanova *et al*., 2008). Many problems that may be comparatively simple to handle in square computing surroundings become seriously challenging issues in grids, chiefly thanks to the dynamism and heterogeneousness of the grid. Scheduling, particularly, becomes the only most troublesome task. As an example, the primary purpose of a resource collaborating in a very grid is to serve the native users of the organization that it belongs to. Moreover, the resource is possibly controlled by the native scheduler. This means that the capability and availableness of the resource for grid users are volatile, that leads to the grid associate timeserving setting. This places nice emphasis on the standard of the programming methodology. The in recent years, vital efforts like SETI@home (Anderson *et al*., 2002) have been created to alter a colossal quantity of computation (that is, computationally intensive larva applications (CBoT)) by exploiting given laptop cycles across the globe. The success of SETI@home spawned variety of similar follow-up comes (for example, Folding@home (Larson *et al*., 2003; Allen, 2005) and lots of more). Folding@home is a distributed computing project that is used for disease research that simulates protein folding, computational drug designing and other types of molecular proteins dynamics. In this study, we use the idle processing resources of thousands of personal computers owned by volunteers who have installed the software on their machines. Additionally, a number of grid programming algorithms for numerous application models together with the larva application model are proposed (Phan *et al*., 2005; Banino *et al*., 2004; Mohamed and Epema, 2004; Ranganathan and Foster, 2002; Fujimoto and Hagihara, 2003). Inspite of the efforts invested with in creating existing programming algorithms highly economical, most of those algorithms have issue in guaranteeing a decent quality of schedules. It is same that performance prediction info on resources obtained using the Network Weather Service (NWS) (Casanova, 2001) is incorporated into programming algorithms as in Xsufferage (Casanova *et al*., 2000) to make sure sensible worth plan. However, it is impractical to assume that excellent performance information on underlying resources in a very grid is quickly obtainable. In the past, two novel programming algorithms (Lee and Zomaya, 2007), known as the Multi Allocation-Input-data-based Listing (MAIL) formula (Lee and Zomaya, 2006a) and the Multiple Queues with Duplication (MQD) formula (Lee and Zomaya, 2006b) that we have a tendency to recently projected area unit conferred with extra results obtained from a lot of intensive experimental study. The Multi Allocation-Input-data-based Listing (MAIL) formula focuses on programming Data-intensive BoT (DBoT) applications, whereas the MQD formula targets scheduling CBoT applications. The Multi Allocation-Input-data-based Listing (MAIL) formula uses a group of task lists that area unit made by taking the information sharing pattern into consideration which area unit organized dynamically, based on the performance of resources throughout the execution of the appliance. The first goal of this dynamic listing is to minimize knowledge transfer, therefore resulting in shortening the overall completion time of DBoT applications. Multi Allocation-Input-data-based Listing (MAIL) makes an attempt to further scale back serious schedule will increase ensuing from few problematic task/node assignments by adopting task duplication. The MQD formula makes programming choices by implicitly taking the recent employment pattern of resources into account. Like Multi Allocation-Input-data-based Listing (MAIL) it adopts a duplication theme so as to achieve higher resource utilization and to avoid undesirable scheduling choices. By higher resource access, their common and primary strength is that they create programming choices while not correct performance prediction data.

In this study, a specialized algorithm Known as Sensible Centrality Scheduling algorithm (SCS) is projected to mainly concentrate on CHoJ application. In which dynamic listings of jobs are created primarily based upon their workloads that ends up in minimize the general finishing time of associate application.

1.1. MODELS

1.1.1. System Model

The grid G in our analysis consists of variety of location in each of that a group of P process node is taking part in a grid. Where Li is that the i^{th} location taking part in G and Ni is a set of nodes:

$$G = \{L_1, L_2, L_3, \ldots L_n\}$$

And:

$$L_i; 1 \le i \le n = \{N_{i,1}, N_{i,2}, \ldots N_{i,p}\}$$

Each location is an autonomous administrative domain that has its own native users, who use the resources in it. These locations are connected with one another through a Wide Area Network (WAN). Nodes are composed of each space-shared and time-shared machine with numerous process speeds, that is, CPU speed. These resources aren't entirely dedicated to the grid. In alternative words, they're used for both native and grid jobs (Banino *et al*., 2004). Every of those nodes have one or additional processors, memory, disk, so forth. The availability and capability of resources, as an example, nodes and network links, varies over time. Therefore, the accurate completion time of jobs on a selected node is difficult, if possible, to work out a priori. Moreover, the job might fail to finish since the resource failure on that it's running.

1.2. Compute Intensive Horde-of-Jobs Model

HoJ applications are normal parallel type of applications that exist in several scientific and engineering fields. An application K of this model consist of r heterogeneous jobs $\{J_1, J_2 \ldots . J_r\}$ without any job dependencies. It is assumed that the work (computation time) of every jobs within the CHoJ model is understood which it varies between jobs. The input file transfer for every job is negligible. The size of the jobs itself is additionally tiny and, thus, transferring it does not influence a lot of the finishing time of the jobs.

1.3. Scheduling Crisis

The grid programming crisis self-addressed during this study may be a job programming of a group K of r freelance jobs, comprising a HoJ application, onto N heterogeneous nodes dispersed across multiple location in a grid. The first goal of this programming is to form as several applicable job node matches as attainable in order that the makespan, conjointly referred to as schedule length, of a HoJ application is decreased. The makespan during this study is outlined because the quantity of your time taken from the time the primary computer file transfer starts to the time the last job accomplished. The function of the resource broker is to allocate the resources to the requesting users. The resources and the users will be dynamic in the wireless grid architecture. The resources can also be provided for the intermittent users. The resource broker is responsible for scheduling.

1.4. Related Work

Grid programming is one among the foremost wide investigated topics in recent times with the aim of their effectiveness in use and its performance. A number of programming algorithms that may be used for Horde of Jobs based applications are projected.

Because of the NP-complete nature of the job programming drawback (Grama, 2003), the majority of projected solutions are heuristic algorithms. These heuristics embrace Max-Min, Min-Min, Sufferage (Lang *et al*., 2006; Maheswaran *et al*., 1999), XSufferage (Casanova *et al*., 2000) and Storage Affinity (SA). However, they make associate arguable assumption that excellent performance prediction information on assets and jobs is thought at the time of scheduling; thus, they're Performance-Prediction Information-Dependent Algorithms (PPIDA). In contrast to these heuristics, a recently projected approximation algorithmic program, list scheduling with Round-robin with Duplication, does not need any performance prediction info on assets or jobs (Lee and Zomaya, 2007) focused on Practical Scheduling with bag of tasks. The extension of this research work is carried out from the job allocation. Max-Min selects the unexpected jobs whose minimum earliest finishing time over all of the nodes is that the longest among all of the unexpected jobs. The chosen job is then allotted to the host on that the minimum earliest finishing time is anticipated. The sole distinction distinctive Min-Min from Max-Min is that the job choice scheme. Specifically, Min-Min provides priority to the job that has the shortest earliest finishing time. Moreover it observes that, at the time of every programming instance, Max-Min tends to schedule the longest job, whereas it's more doubtless that Min-Min processes the shortest job. Sufferage makes programming verdict by the sufferage value of jobs (Ranganathan and Foster, 2002). The sufferage price of a task is outlined as the distinction between its earliest finishing time and its second earliest finishing time. At every planning call, it computes the sufferage values of all of the unscheduled jobs and schedules the jobs whose sufferage value is that the largest. This approach is effective because of the serious increase of makespan is decreased. We cannot come to conclusion that this does not guarantee that the general makespan is shortened.

XSufferage widen the Sufferage planning heuristic (Maheswaran *et al*., 1999) by taking information sharing into consideration. It makes planning decisions supported the sufferage worth of jobs. The sufferage worth of a job in XSufferage is outlined because the difference between its earliest location-level completion time and its second earliest location-level finishing time.

The sufferage values utilized in Sufferage are node level, those adopted by XSufferage are location level. The sufferage worth of a job is employed as a live of the doable increase on makespan, that is, a job with an oversized sufferage worth implies that the finishing time of the job seriously increases, inflicting a doable increase of makespan if it's not allotted to the node on that the earliest location-level finishing time is achievable. Therefore, the larger the sufferage worth of a job, the upper the planning priority the job gets.

Storage Affinity (SA) primarily aims at minimizing information transfer by creating scheduling choices that incorporate the situation of knowledge previously transferred. Additionally, it considers job replication as presently as a number becomes obtainable between the time once the last unexpected job gets allotted and the time once the last running job finishes its execution.SA resolve job/node assignments depends on the SA metric. The SA of a job to a node is to quantity the jobs which was stored in the node belongs. Though the programming verdict SA makes is between a job and a node. SA is calculated between a job and a location. This can be as a result of, within the grid model used for SA, each location within the grid uses one information repository that may be fairly accessible by the nodes within the location. For each programming verdict, the SA calculates SA values of all unexpected jobs and dispatches the job which has high value of SA. If none of the jobs contains a positive SA value, one among them is selected in arbitrary manner. By the time the programming of all unexpected jobs is complete, there would be as several as |N| running jobs, departure all |N| node busy. On the completion of any of those running nodes, SA starts job duplication. Now, every of the remaining running jobs is taken into account for duplication and also the best one is selected. The selection verdict is predicated which depends on the SA value and the variety of replicas.

RR could be a grid programming rule for freelance coarse grained jobs. Because the term implies, its uniqueness comes from the round-robin order duplication theme that makes duplicates of running jobs in an exceedingly round-robin fashion after conducting list programming for all of the special jobs. RR initial every which way assigns a job to every node within the grid and so waits till one or additional of these assigned nodes complete their jobs. On the completion of a job, the next special job is sent to the node on that the completed job has run. This tends to end in quick assets obtaining additional jobs. Once all of the jobs are dispatched, RR starts duplicating running jobs, hoping that these replicas end prior to their novels. RR performs programming with none dynamic data on assets and nodes. The rule is comparable to alternative programming heuristics that need such performance data. The new Multi Allocation-Input-data-based Listing (MAIL) algorithm rule cluster jobs into variety of dynamic lists supported their information distribution modes. Each of these lists is meant to be scheduled onto identical location in the grid so as to attenuate convey the details, that is vital to shortening the finishing time of DBoT applications in explicit. Since the performance of grid resources fluctuates over time, the lists square measure organized dynamically during application runtime. In a trial to with efficiency contend with the dynamism of grid resources, the Multi Allocation-Input-data-based Listing (MAIL) adopts a job duplication that's particularly useful in avoiding serious schedule will increase. For example, one or two of jobs is also running unexpectedly long, increasing the schedule considerably due to the overload or irregular behaviors of the assets on which they're running or being transferred. A same duplication approach is found in RR. Note that Multi Allocation-Input-data-based Listing (MAIL) doesn't use any prediction data on the performance of assets and its use, apart from the information on input file, that is, size and placement, which is Multi Allocation-Input-data-based Listing (MAIL) rectifiable by the computer hardware whereas planning the jobs of associate application. However, it's not assumed that the information is offered for following invocation of the application. The Multi Allocation-Input-data-based Listing (MAIL) consists of 2 major phases: Job Grouping part-group's jobs into a set of lists supported their information sharing pattern, associates these job lists with location information and breaks and/or associates them with nodes. Scheduling part-assigns jobs to nodes, dynamically reorganizing job lists and duplicates jobs once all jobs square measure scheduled and a few jobs are still running.

The MQD will proceed with the programming method. On completion of jobs, the performance ranking of the host on which the jobs is finished is computed. The performance of a bunch used for computing its performance ranking is quantified by dividing the employment of the last job the node finished by the job total finishing time. The above performance ranking decides that a queue future job for the node is chosen from it.

1.5. Proposed Work

It is observed that good performance data on underlying resources during a grid is extremely thorny, if not impossible to get. Therefore, a best schedule generated by a programming rule might not truly be deliverable if the programming choices are created victimization performance prediction data. On the opposite hand, if programming is meted out while not intuitive judgments, as an example, in a first-come, first-serve manner, the standard of the schedule can simply become poor.

The Grid computing facilitates flexible, secure, coordinated large scale resource sharing among dynamic collections of individuals, institutions and resource sharing in a geographical distributed area.

It is an evolving Technology of set of open standards for Web services and interfaces that make services, or computing resources, available over the Internet. These days the grid technologies are used on homogeneous clusters and heterogeneous clusters and they can add value on those clusters by assisting, for example, with scheduling. The criteria for Grid Computing involves by coordinating the resources that are not subject to centralized control. It uses standard, open, general-purpose protocols and interfaces and delivers nontrivial qualities of service.

1.6. Architecture of Grid Environment

The main components of grids are:

- Grid Information Server
- Global Grid Resource Broker
- Local Grid resource Broker
- Grid Users
- Grid resources like computers, laptops, Servers, Printers

In **Fig. 1**, the Architecture of the Grid is depicts the various components of Grid. The role of Global Grid Resource Broker is the client Registration of jobs to process and the role of Resource nodes is to donate the resources at local Grid resource Broker and process the client request as per the instruction given by Local Grid Resource Broker. All the resource statics like resource node, resource node size, resource header information will be collected from all the LGRB by Grid Information server and it is forwards to the GGRB. The main component in which scheduling will takes place in global grid resource broker.

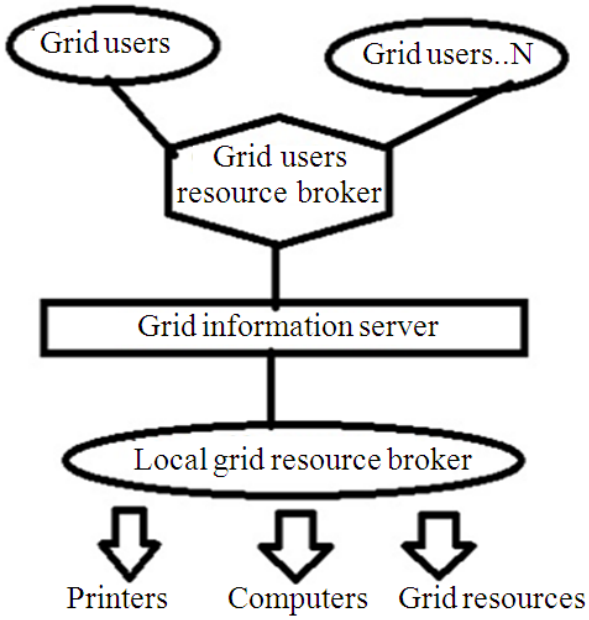


Fig. 1. Architecture of grid environment

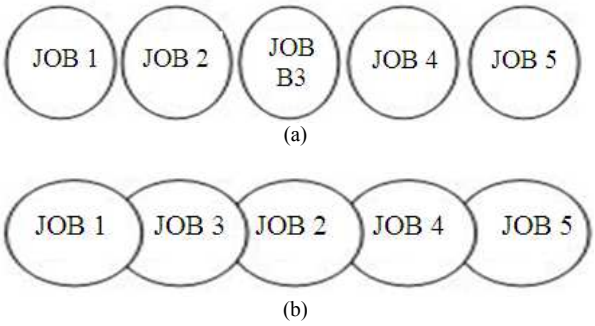


Fig. 2. Initial set of k jobs, (a) Initial set of jobs, (b) Preprocessed jobs

This GGRB provides all the information like resource type, resource variants, resource allocations and the corresponding nodes like nodes 1, node2, node3 and the information of the nodes will be acquired by GGRB. The Grid Scheduling takes place in the time sequence. To provide the efficient scheduling with the available resources is the one of the top issues in the Grid Computing environment.

The mechanism of Sensible Centrality Scheduling (SCS) algorithm is explained in the **Fig. 2** with initial set of K jobs. Initially jobs are organized in descending order by workload and programming of jobs are mentioned.

Table 1. Workload allocated to nodes

Nodes	----------SCS---------		----------MDQ----------		----------MAX-MIN--------		---------MIN-MIN------	
1	34	48.57%	54	77.07%	44	62.875%	34	48.57%
2	48	48.00%	76	76.00%	48	48.00%	48	48.00%
3	22	40.74%	34	62.96%	22	40.74%	34	62.96%

Table 2. Comparison of Algorithm by Makespan

Algorithm	Total time (min)	Average makespan (min)
SCS	10.8	3.60
MDQ	15.3	5.10
MAX-MIN	11.1	3.70
MIN-MIN	11.5	3.83

From the well known workload we compute the centrality value by dividing the sum of maximum and minimum workload and bi as sown in the step. Initially overall processing speed of each node is calculated with the help of node processing speed in various time limits. Based on these values we assign rank to the node. In this algorithm we use three queues (i.e.,) MajQ, MinQ, RepQ. The job assigned to the MajQ and MinQ are depends upon the centrality. RepQ is used to avoid the job redite (needless replication) by deleting the job from the queue (RepQ) once it assigned for processing. The jobs in the MinQ are only assigned to the nodes which have highest rank value.

The jobs in the MajQ are assigned to the remaining nodes based on the node rank. Either MajQ or MinQ jobs are get finished, it go for RepQ to find out the unscheduled jobs. If exits it process those jobs in the above procedure. This is shown below:

Input: A set of k of jobs, a set N of nodes.
Output: A schedule of K onto N

Algorithm of SCS

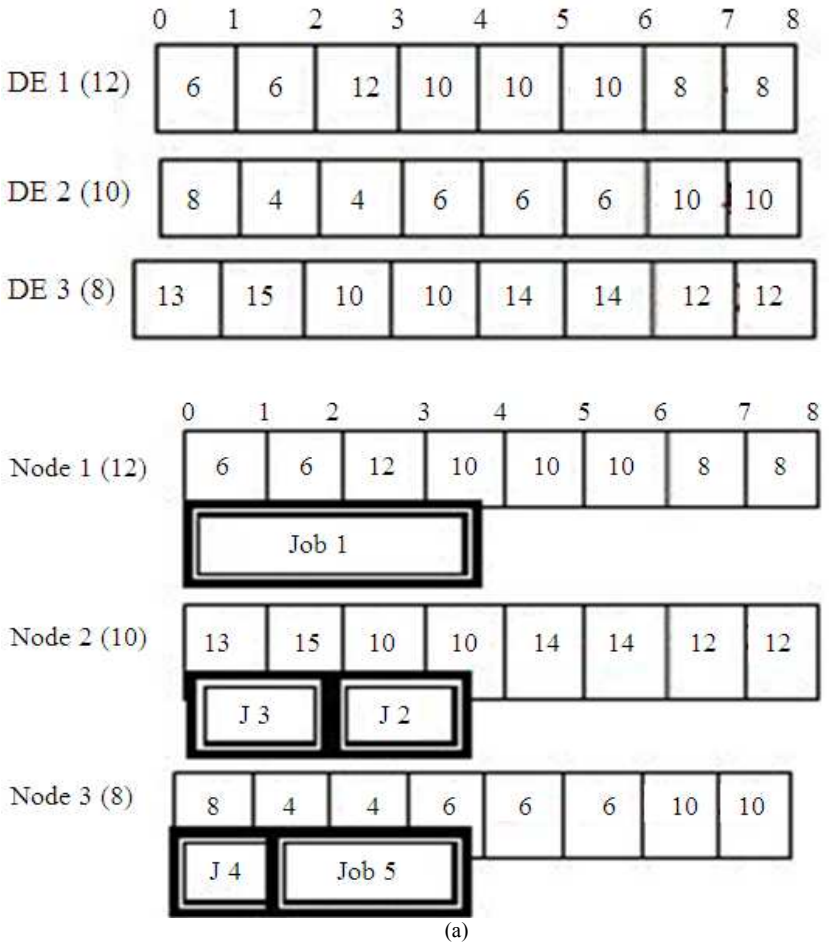
1. Sort k in decreasing order by workload
2. Let centrality = max (K)+min (K)/2
3. Create 3 queues/*majQ,min Q,Rep Q*/
4. for each k then
5. Rep Q = k
6. if (k> = centrality)
7. then
8. Assign maj Q=k
9. else
10. Assign min Q=k
11. end if
12. end for
13. Compute the processing speed of all nodes and assign rank.
14. Let O = Nodes which are sorted in ascending order based on ranking factor.
15. Let m be the minimum ranking node
16. for each value (vi) of min Q
17. Let m = vi
18. Delete vi from Rep Q
19. End for
20. for each value vj of maj Q
21. Allocate vj to the nodes in 'O' order except M
22. Delete Vj from Rep Q
23. end for
24. if last job in min Q is completed then
25. Check Rep Q to find unscheduled jobs
26. if any
27. goto step 2127. vi = min (RepQ)
28. goto step17
29. end if
30. if last job in Max Q is completed then
31. check Rep Q to find unscheduled jobs
32. if any
33. vj = max(RepQ)
34. end if

1.7. Experimental Evaluation

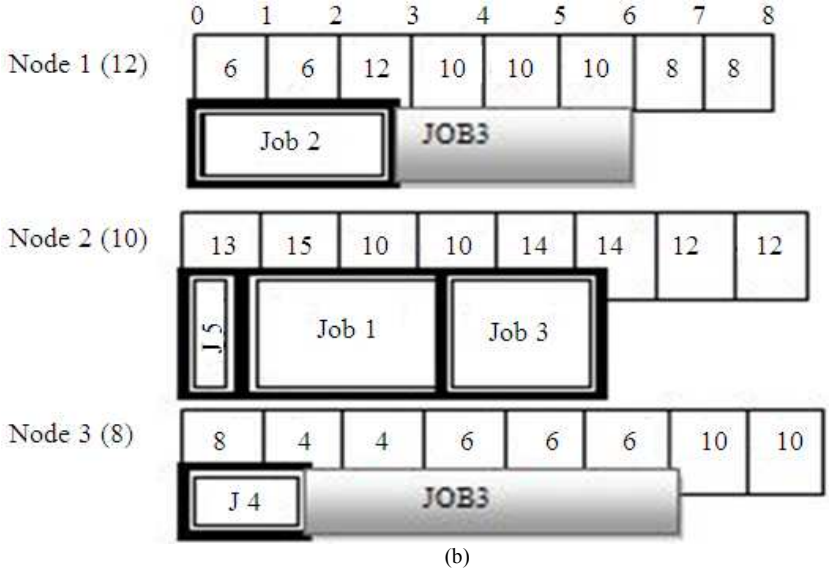
The primary role of the scheduling algorithm is to minimize the makespan as much as possible. In order to attenuate makespan, one of the important key issues is to avoid repetition. The various workloads assigned to the nodes by each algorithm are offered in **Table 1** when compared to other algorithm, SCS acquire minimum makespan which is clearly clarified from the **Table 2**.

1.8. Grid Simulator Tool

The grid simulator Tool used for this study is enforced with GridSim tool due to its made set of simulation facilities that Multi Allocation-Input-data-based Listing (eaMAIL) y permits the event and analysis of planning procedures for heterogeneous distributed computing environments in simulating grids is Tiers an arbitrary constellation generator that fabricate arbitrary network models analogous to the structure of the web. Properties of resources and jobs within the simulations conducted during predefined set of assets and job factors shown in **Table 2**. This was proved by writing the various test cases for every node and network link is simulated by employment traces obtained from actual systems deployed because the GrADS test bed, where the end to end testing was carried out.


0 1 2 3 4 5 6 7 8
DE 1 (12)
6 6 12 10 10 10 8 8
DE 2 (10)
8 4 4 6 6 6 10 10
DE 3 (8)
13 15 10 10 14 14 12 12
Node 1 (12)
6 6 12 10 10 10 8 8
Job 1
Node 2 (10)
13 15 10 10 14 14 12 12
J 3
J 2
Node 3 (8)
8 4 4 6 6 6 10 10
J 4
Job 5

(a)


0 1 2 3 4 5 6 7 8
Node 1 (12)
6 6 12 10 10 10 8 8
Job 2
JOB3
Node 2 (10)
13 15 10 10 14 14 12 12
J 5
Job 1
Job 3
Node 3 (8)
8 4 4 6 6 6 10 10
J 4
JOB3

(b)

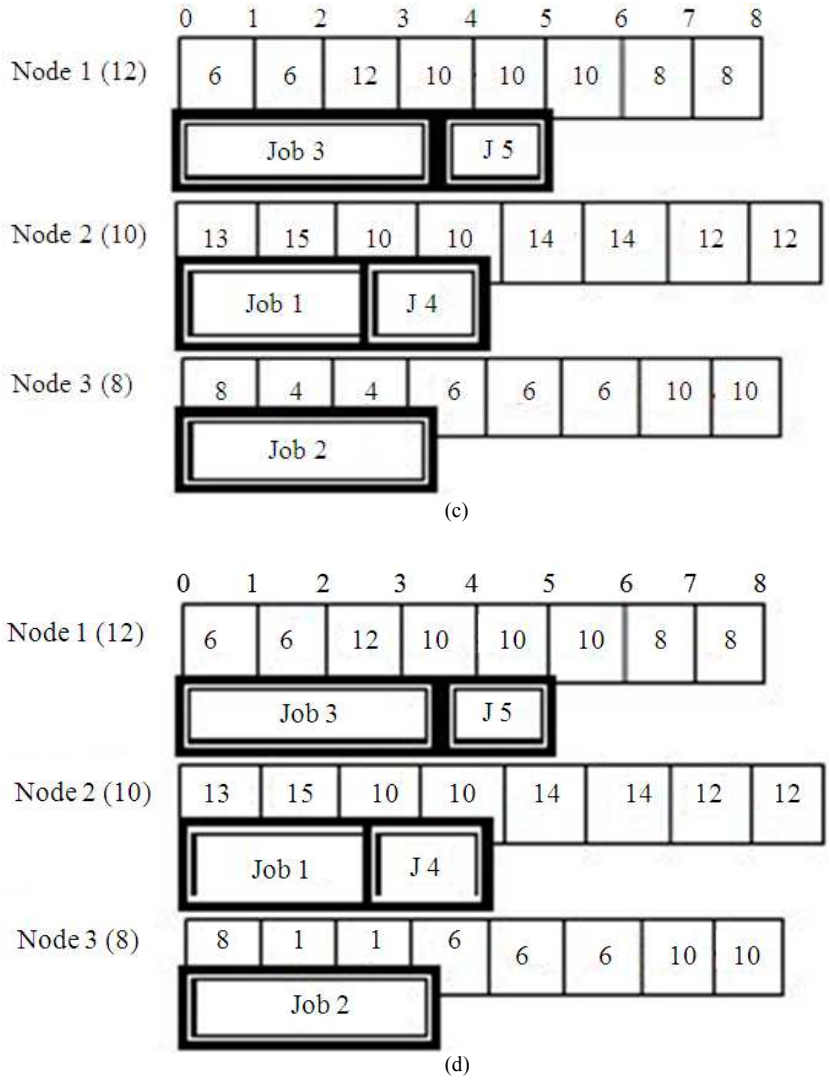


Fig. 3. Programming of Jobs, (a) Node processing speed, (b) SCS (c) MDQ, (d) Max-Min

1.9. Simulation Results

The SCS algorithm and three previously proposed algorithms, MDQ, Max-Min, Min-Min, are compared by using a total of 20,000 simulations for each **Fig. 3**. The 10,000 simulations are composed of 150 simulated grids and 30 simulated jobs and each of these 3,000 grid-job pairs is run 10 times with different host workload traces. The simulation results presented in this study clearly show the promising performance of the SCS algorithm compared to the other three. The experimental results of Max-Min, Min-Min and MDQ shown in **Fig. 4**. The normalized average makespan is shown in the **Fig. 5**. It is defined as the average makespan of an algorithm over that of SCS that generates the shortest makespan among the three algorithms presented in this study models analogous to the structure of the web. Properties of resources and jobs within the simulations conducted during this study are random and uniformly scattered between a predefined set of assets and job factors shown in **Table 2**. Every node and network link is simulated by employment traces obtained from actual systems deployed because the GrADS test bed and the virtual test has been conducted.

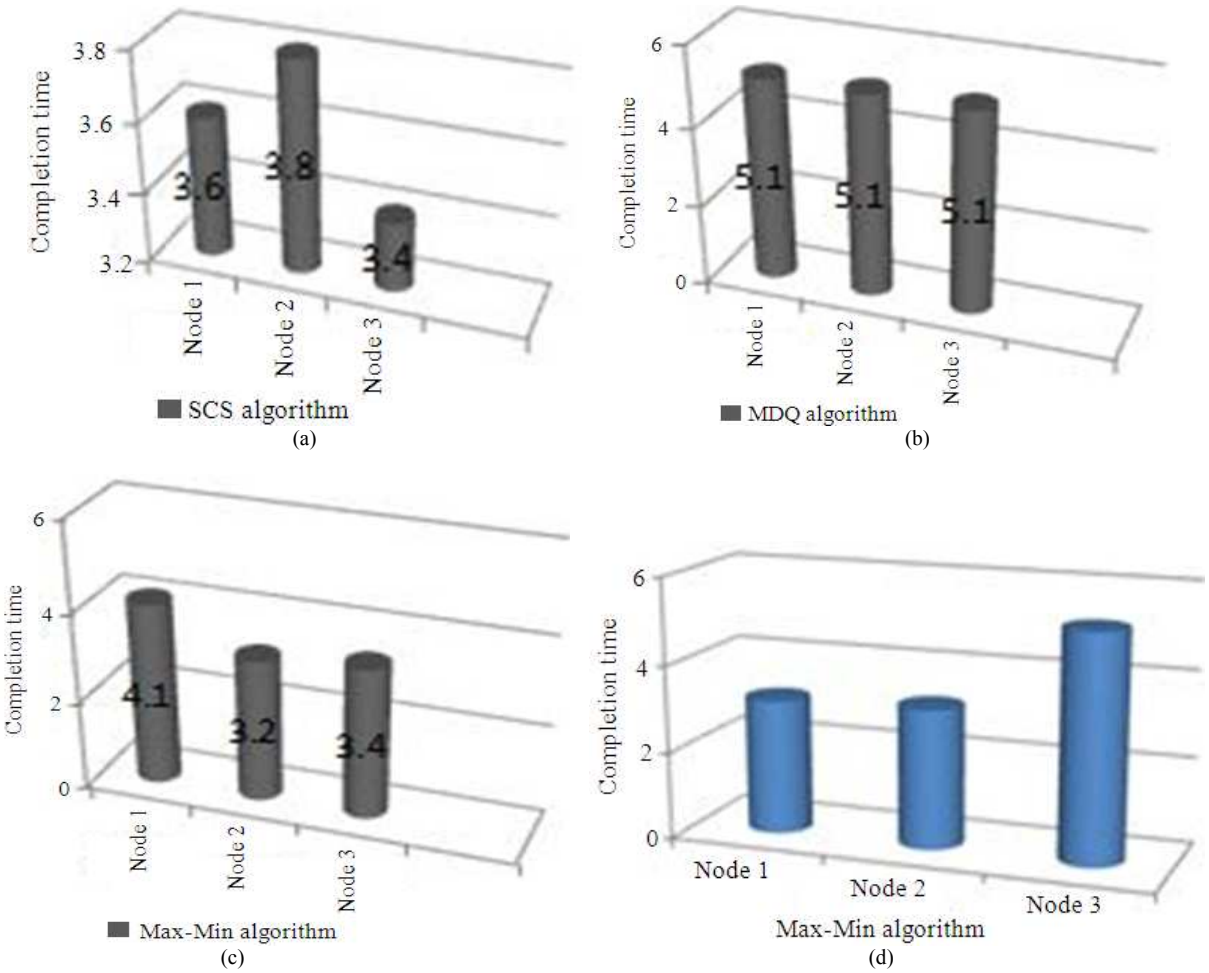


Fig. 4. Simulation results for completion of jobs (a) SCS (b) MDQ (c) MAX-MIN (d) MIN-MIN

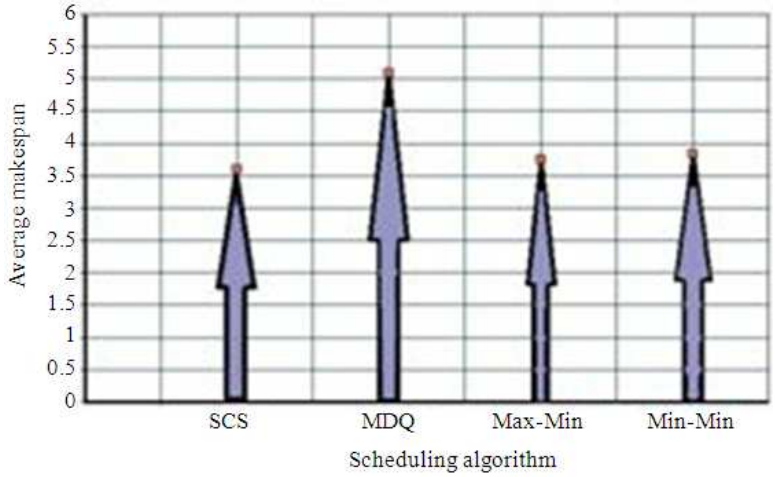


Fig. 5. Simulation results for average makespan in various algorithm

2. CONCLUSION

This study bestowed an algorithm called Sensible centrality Scheduling (SCS) for HOJ application in grid environment. They are fastidiously designed to include the fundamental features of the grid (that is, vitality and heterogeneity) into the decision-making process. Practicability and performance are the two main design goals. The projected algorithms achieve these goals by victimization intuitive approaches. SCS takes the work pattern of resources into thought for programming choices and conjointly it neglect job duplication. Extensive experiments with numerous take a look at superior performance of SCS. It mostly delivers higher schedules compared to those generated by the four algorithms (that is, Max-Min, Min-Min and MQD). Further focus to be carried out to enhance this algorithm to process the suspended jobs.

3. ACKNOWLEDGEMENT

This research work is supported in part by the Kalasalingam University, srivilliputhur, India under the research and development scheme to promote the research work.

4. REFERENCES

Allen, B., 2005. Einstein@Home. LSG.

Anderson, D.P., J. Cobb, E. Korpela, M. Lebofsky and D. Werthimer, 2002. SETI@home: An experiment in public-resource computing. ACM Commun., 45: 56-61. DOI: 10.1145/581571.581573

Banino, C., O. Beaumont, L. Carter, J. Ferrante and A. Legrand *et al*., 2004. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. IEEE Trans. Parallel Distribut. Syst., 15: 319-330. DOI: 10.1109/TPDS.2004.1271181

Blanquer, I.B., V.H. Hernandez and J.D. Segrelles, 2005. An OGSA middleware for managing medical images using ontologies. J. Clin. Monit. Comput., 19: 295-305. PMID: 16328944

Casanova, H., 2001. Simgrid: A toolkit for the simulation of application scheduling. Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, May 15-18, IEEE Xplore Press, Brisbane, Qld., pp: 430-437. DOI: 10.1109/CCGRID.2001.923223

Casanova, H., A. Legrand and M. Quinson, 2008. SimGrid: A generic framework for large-scale distributed experiments. Proceedings of the 10th International Conference on Computer Modeling and Simulation, Apr. 1-3, IEEE Xplore Press, Cambridge, UK., pp: 126-131. DOI: 10.1109/UKSIM.2008.28

Casanova, H., A. Legrand, D. Zagorodnov and F. Berman, 2000. Heuristics for scheduling parameter sweep applications in grid environments. Proceedings of the 9th Heterogeneous Computing Workshop, May 1-1, IEEE Xplore Press, Cancun, pp: 349-363. DOI: 10.1109/HCW.2000.843757

Fujimoto, N. and K. Hagihara, 2003. Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid. Proceedings of the International Conference on Parallel Processing, Oct. 9-9, IEEE Xplore Press, Kaohsiung, pp: 391-398. DOI: 10.1109/ICPP.2003.1240603

Grama, A., 2003. Introduction to Parallel Computing. 2nd Edn., Addison Wesley, Harlow, ISBN-10: 0201648652, pp: 636.

Lang, B., I. Foster, F. Siebenlist, R. Ananthakrishnan and T. Freeman, 2006. Attribute based access control for grid computing. Mathematics and Computer Science Division.

Larson, S.M.. C.D. Snow, M. Shirts and V.S. Pande, 2003. Folding@Home and Genome@Home: Using distributed computing to tackle previously intraceable problems in computational biology. Biophysics Program, Stanford University.

Lee, Y.C. and A.Y. Zomaya, 2006a. Data sharing pattern aware scheduling on grids. Proceedings of the International Conference on Parallel Processing, Aug. 14-18, IEEE Xplore Press, Columbus, OH., pp: 365-372. DOI: 10.1109/ICPP.2006.30

Lee, Y.C. and A.Y. Zomaya, 2006b. A grid scheduling algorithm for bag-of-tasks applications using multiple queues with duplication. Proceedings of the 1st IEEE/ACIS International Workshop on Computer and Information Science, Jul. 10-12, IEEE Xplore Press, Honolulu, HI., pp: 5-10. DOI: 10.1109/ICIS-COMSAR.2006.7

Lee, Y.C. and A.Y. Zomaya, 2007. Practical scheduling of bag-of-tasks applications on grids with dynamic reMAIL(Multi Allocation-Input-data-based Listing) ience. IEEE Trans. Comput., 56 815-825. DOI: 10.1109/TC.2007.1042

Magnin, I.E. and J. Montagnat, 2006. The grid and the biomedical community: Achievements and open issues. Proceedings of the EGEE User Forum, (EGEE' 06), Geneva, Switzerland.

Maheswaran, M., S. Ali, H.J. Siegel, D. Hensgen and R. Freund, 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings of the 8th Heterogeneous Computing Workshop, Apr. 12-12, IEEE Xplore Press, San Juan, pp: 30-44. DOI: 10.1109/HCW.1999.765094

Mohamed, H. and D. Epema, 2004. An evaluation of the close-to-files processor and data co-allocation policy in multiclusters. Proceedings of the IEEE International Conference on Cluster Computing, Sep. 20-23, IEEE Xplore Press, pp: 287-298. DOI: 10.1109/CLUSTR.2004.1392626

Montagnat, J., A. Frohner, D. Jouvenot, C. Pera and P. Kunszt *et al*., 2008. A secure grid medical data manager interfaced to the gLite middleware. J. Grid Comput., 6: 45-59. DOI: 10.1007/s10723-007-9088-2

Phan, T., K. Ranganathan and R. Sion, 2005. Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm. Proceedings of the 11th International Conference on Job Scheduling Strategies for Parallel Processing, Jun. 19-19, Springer Berlin Heidelberg, Cambridge, MA, USA., pp: 173-193. DOI: 10.1007/11605300_9

Ranganathan, K. and I. Foster, 2002. Decoupling computation and data scheduling in distributed data-intensive applications. Proceeding of the 11th IEEE International Symposium on High Performance Distributed Computing, Jul. 24-26, IEEE Xplore Press, pp: 352-358. DOI: 10.1109/HPDC.2002.1029935