Heuristic based Independent Task Scheduling Techniques in Cloud Computing: A Review

Puneet Banga
Research Scholar
MMU, Mullana
CSE Deptt.

Sanjeev Rana, PhD
Professor MMU, Mullana
CSE Deptt.

ABSTRACT
Cloud computing, a buzzword of today's that combines the power of both parallel and distributed computing. It delivers its output in the form of service(s) that can be IaaS, SaaS and PaaS (Infrastructure, Software and Platform- as a Service). In Cloud computing, we won't compute on local machines, but on someone premises operated by someone else. Actually Cloud environment deals with dissimilar kinds of virtualized resources. So, to allocate and schedule resources efficiently it requires noticeable efforts. One of the core phases is task scheduling which plays a vital role. It can be seen as the finding an optimal assignment of set of task(s) over the available resource set to obtain desired goals like: cost, quality of service and makespan etc. Even, most of the organizations already started implementing CTQ model (less COST, minimum TIME and assured QUALITY) for attaining maximum return with assured quality. The objective of this paper is to review various independent task scheduling techniques under heuristic mapping category so that we can apply techniques according to current requirement.

Keywords
Cloud Computing, Scheduling, Heuristic, Independent Task, Immediate mode, Batch mode.

1. INTRODUCTION
Cloud computing can be seen as a technology which comprises of many others like: Grid computing, Utility computing, Autonomic computing, SOA and Web services along with hardware virtualization as shown in the diagram below.

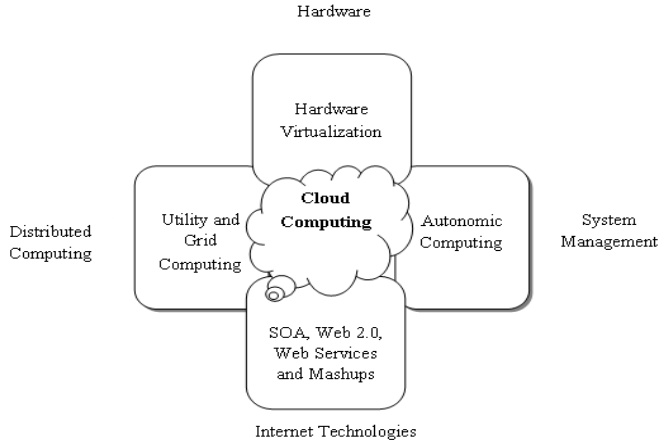

Figure 1: Advent of Cloud computing

It is the distributed computing where everything from tasks to resources are distributed in some manner for the purpose of communication and resource sharing as two prime purposes. In this paper we reviewed various independent task scheduling techniques under heuristic mapping category based on their important parameters like: makespan, resource utilization rate and some QoS parameters. It starts from general model that shares similarity with Grid computing model in general way and then in next section we have described the meaning task scheduling along with its classification. Then this research paper reviewed various techniques of both types of categories under heuristic mapping that are immediate and batch mode with their inherent characteristics.

2. GENERAL MODEL FOR GRID AND CLOUD COMPUTING
Both, Grid and Cloud computing shares a common model which basically consist of four building blocks that are: Clients/Users, Resource Broker/Scheduler, Grid/Cloud Information Service (GIS/CIS) and available Resource(s). Below, mentioned four steps will show their interaction among entities for the purpose of executing user's job respectively.

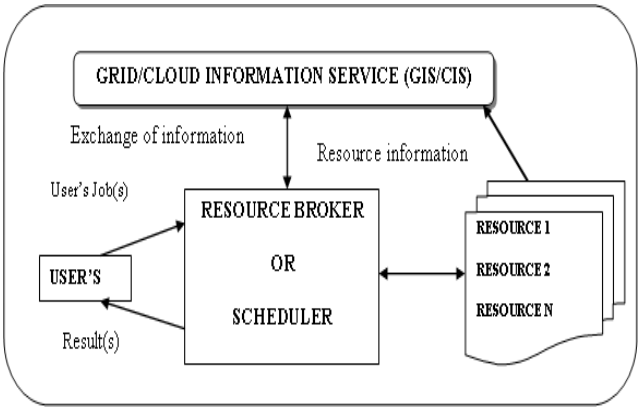

Figure 2: General Model for Grid and Cloud Computing

I. User will submit job(s) to the resource broker/scheduler for execution.
II. Resource broker acquire information about resources from GIS/CIS and then divide the job into various tasks or subtasks if needed. Then map the same to resources distributed geographically according to user's requirements and availability of resources or based on some optimized scheduling technique discussed in next section.
III. GIS/CIS are responsible for providing information about status of available resources which helps the broker for scheduling, monitoring and further communication if required.
IV. After execution of all tasks, result is combined and sends back to user via broker.

3. TASK SCHEDULING AND CLASSIFICATION

It is the one of crucial phase which plays a significant role for overall performance in the system. It is the course of action for mapping tasks to the available and selected resources based on requirement(s). The overall performance should be enhanced by reducing the task completion time. This will be achieved by ensuring that selected resources are used without being idle. Another term which plays a vital role here is Meta task: [4] defined as collection of tasks or subtasks. For example: jobs submitted to a supercomputer by different kind of users would be an example of Meta task. Meta task can be seen as batch of tasks which may include either similar kind of attributes or share some basic characteristics.

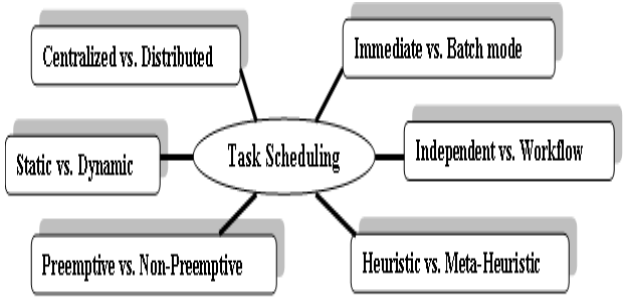

Figure 3: Classification of Task Scheduling

Different categories of task scheduling are:

3.1 Centralized vs. Decentralized – In Centralized scheduling, decision is made by a central node. The advantages are: efficiency, ease of operation and monitoring on resources [5]. On the other hand they have some pitfalls like: scalability issue, inherent complexity and fault tolerance. Decentralized or Distributed type of scheduling is more practical for real cloud environment regardless of its poor efficiency compared to its counterpart. As there is no central entity for overall control, so role of local schedulers plays considerable role for maintaining the state of scheduling.

3.2 Static vs. Dynamic scheduling - In static mode, everything from task execution time to resource capabilities is known in advance. A task assigned once to a resource remains same [5], so it's much simpler to implement specially from scheduler's perspective. In case of dynamic task scheduling, resources are dynamically available for scheduling. It is further flexible than static scheduling as decision is to be taken is not fixed where its receive complexity in addition.

3.3 Preemptive vs. Non-Preemptive – In preemptive scheduling, task can be interrupted while in execution and can be transferred to another machine. If constraints such as priority, deadline and cost are to be imposed then this type of scheduling is become mandatory. In Non-Preemptive scheduling resources [6] are not permissible to be re-allocated until scheduled task(s) finished its execution or willingly they transfer their control.

3.4 Immediate vs. Batch Mode scheduling - In Immediate mode, the task is scheduled to resource immediately without any time lag. It is also known as on-line mode, each task is considered only once [7] i.e., the mapping decision is not altered once it is computed**.** Whereas in Batch mode: tasks are collected into a set (Meta task) and then entire batch is examined for mapping at prescheduled times called mapping events. It is also recognized as offline mode which is the hottest area of research now.

3.5 Heuristic vs. Meta-Heuristic scheduling - Heuristics techniques are specific in terms of solving problems. Meta-heuristic's [8] on the other hand, are problem-independent techniques. They are also known as Guided Random Search or Nature's Heuristic approach. They can be used as black box in a general way for wide range of problems.

3.6 Independent vs. Workflow scheduling – A task which do not require any communication (dependency) with other tasks is called independent task, where as dependent tasks are different as they have some precedence order to be followed [9] during the scheduling process. Let there are 7 tasks: T0 to T7, if they are independent then scheduler can map those independently means without keeping any order. But dependent tasks are executed in order according to precedence in mind like shown in figure:

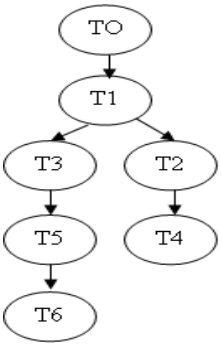

Figure 4: Directed Acyclic Graph (DAG)

DAG can be represented by three types like: Sequence, Parallelism and Choice where in sequence: an ordered chain of tasks where one task will start after a previous task has completed. Parallelism denotes tasks which are running all together. In choice control, a task executed dynamically when it's associated evaluator criteria turns out to be true [10].

4. IMMEDIATE MODE SCHEDULING

4.1 Opportunistic Load Balancing (OLB) [11] – In this, each task is assigned to machine in arbitrary order that is expected to be available, regardless of the task's expected execution time and its completion time on that machine. The intuition behind OLB is to keep all machines as busy as possible. But it may results in poor load balancing level due to non-consideration of current workload. In some special cases OLB will act as First Come First Serve (FCFS) or Myopic scheduling.

4.2 Minimum Execution Time (MET) [11] – It schedule tasks based on their expected execution time on that machine. Here, a task is assigned to a resource on which it can be executed in minimum time. [5] [6] Allocation a task in such a way (without knowing resource availability time) may

lead to load imbalance among resources. Heuristic MET can solve problem in *O*(tm) time. Here, t denotes number of independent tasks whereas m represents number of allocated machines.

4.3 Minimum Completion Time (MCT) [11]

– Unlike MET which consider execution time only, here task is assigned to a resource that gives minimum completion time Cij. (i^{th} task execution time on the j^{th} resource + availability of j^{th} resource) [5][6] This technique considers availability time as key parameter but allocation in this manner may results in execution of jobs on less high speed Cloud machines. But, it tries to balance the load among resources quite smoothly. Like MET, here problem is solvable in *O*(nm) time is another good point.

4.4 Classifier MCT [12] –

An enhanced form of MCT algorithm which categorized allocated resource(s) and task(s) into two types of classes to achieve better QoS. First one is HIGH class for HIGH computational task(s) and LOW computational task(s) are scheduled to LOW class resources respectively. Each task is examined for its respective class so if it is of HIGH class then it will be scheduled to set of HIGH class resources else it will schedule to LOW class resources and then traditional MCT is applied on it. Experimental results show that CMCT outperforms MET and MCT in terms of makespan and average resource utilization rate. Till now, in this category it is the one the best technique in terms of QoS.

Table 1: Immediate mode independent task scheduling techniques

Parameter/ Technique	OLB	MET	MCT	CMCT
Category	Static	Static	Static	Static
Factor(s) achieved	Keep the resources busy	Minimum execution time	Minimum completion time	Makespan with load balancing
QoS	NO	NO	NO	YES
Phases	ONE	ONE	ONE	TWO
Speciality	Blind approach	To consider fastest resources	Less makespan with resource availability info	Less makespan with assured quality

5. BATCH MODE SCHEDULING

5.1 Min-Min [11] -

It starts with a Meta task (MT) of all unmapped tasks. Out of two phases, its first phase finds minimum expected completion time over all the allocated machines for each task and then in second phase starts. Task with the overall minimum expected completion time from MT is selected in phase two and then scheduled to the corresponding machine [3] [5] [6]. Then, this task is deleted from pool and the process is continues until all tasks are mapped successfully. This heuristic takes O (t^2m) time.

5.2 Max-Min [11] -

Max-Min is analogous to Min-Min, except in second phase which finds maximum expected completion time instead of minimum. As this technique believes to execute longer task first and allows shorter tasks to be executed concurrently. This phenomenon results in lesser makespan, better resource utilization rate and even better load balancing level.

5.3 Sufferage [13] –

As the name indicates, the task suffers the most will be scheduled first. Then, that task will be mapped to machine which will execute it in minimum completion time. In this algorithm, a sufferage value is calculated for every task in the batch based on difference between first and second minimum completion time. Most suffered task is executed first and then process will continue untill all unmapped tasks are mapped successfully.

5.4 RASA [14] –

Resource Aware Scheduling Algorithm is a hybrid scheduling technique which combines the good traits of both Min-Min and Max-Min alternatively depending on resource count. If the number of available resources is odd then Min-Min is executed to assign the first task, else Max-Min strategy is applied. The cycle continues until all unmapped tasks are assigned. Results show better performance over Min-Min and Max-Min.

5.5 TASS [15] -

Task Aware Switcher Scheduling, another hybrid scheduling algorithm which is inspired from RASA. It considers task count rather than resource. In this, if the number of task count is even then Max-Min strategy is performed to assign the first task; otherwise Suffrage strategy is applied till all unmapped tasks are scheduled to respective resources. Experimental results show that TASS perform better compare to Min-Min, Max-Min, Sufferage and RASA in terms of makespan and average resource utilization rate. Further this technique can be tailored according user priority, their deadline and other QoS parameters.

5.6 QoS Guided Min-Min [16] -

The term QoS defines different things to different people, but here it means network bandwidth. In his research, one dimension QoS is considered that is network bandwidth along with conventional Min-Min. If a normal task can be executed on both high QoS and low QoS resources, but a task that requests a high QoS can only be executed on a resource providing high quality of service is the key to success. This algorithm provides improved results than the conventional Min-Min. Further the work can be extended by adding more QoS parameters to it.

5.7 QoS Priority Grouping Scheduling [17]

- Rather than bandwidth as a major parameter, this algorithm looks into deadline and acceptation rate of the tasks as foremost factor. It is based on the concept that, if a task can be executed on fewer hosts then that task has higher priority. If we compare this with Min-Min and QoS guided Min-Min, this

algorithm achieves better acceptance rate and reduction in makespan. In next section of this paper author has introduced another parameter i.e., deadline of task, so QD-Sufferage is applied to task with deadline and those tasks without deadline are scheduled according to QoS priority grouping. Result shows better throughput but it still required more QoS parameters as future work.

5.8 Scheduling Framework for Bandwidth-Aware Job Grouping-Based Scheduling [18] – It considers jobs by two factors that are computational and the communication capabilities of the resources. It uses bandwidth capacity of links to decide the priority of each resource. The resources are selected based on priority by scheduler itself and it groups independent fine-grained jobs together based on chosen resources processing capability (MIPS). The purpose behind grouping of jobs is to maximize the resource utilization rate and to decrease transmission time.

5.9 QoS Sufferage [19] – A refinement of Sufferage algorithm which takes network bandwidth into account and schedules the tasks based on their network bandwidth. But this algorithm tries to achieve smaller value of Makespan as compared with the Max-Min, Min-Min, QoS guided Min-Min and QoS priority grouping algorithms. But the proposed work considers only bandwidth as prime parameter for QoS, so in future multiple factors for QoS can be considered to maximize the overall performance.

5.10 A Min-Min Max-Min Selective Algorithm [20] – For every decision this algorithm takes the advantageous of both Min-Min and Max-Min. Here, all tasks are sorted according to execution time in ascending order. Then, it calculates the standard deviation (sd) of all unmapped tasks based on their completion time. Then a place is found in the sorted list where the difference between the two consecutive values is more than sd, if this place is in the first half of the list, then conventional Min-Min is applied else Max-Min is used to map the task. Result shows better performance over Min-Min and Max-Min in each round.

5.11 Load Balanced Min-Min (LBMM) [21] – Another variant of Min-Min which takes load balancing factor as one of the vision in this work. LBMM in its first round executes Min-Min followed by selecting the heavy load resources and then reassigns them to the resources with light load in its second round. Then that task's completion time is calculated for all resources and the maximum completion time of that task is compared with the makespan produced by Min-Min. If calculated value is shorter than makespan produced then the task is rescheduled to resource that produces it. Now ready time of both resources (old and latest) are updated else steps are repeated again for a task with next maximum completion time. When all resources and all tasks assigned are considered for rescheduling then the process stops. Experimental results show that LBMM outperforms Min-Min with respect to makespan and average resource utilization rate.

5.12 Improved Max-Min Algorithm in Cloud [22] - In this, we picked longest task (in terms of MI) first as this algorithm considers expected execution time instead of completion time for mapping user's task. Allocation to the slowest resource for longest task permits us to finish other smaller tasks concurrently on high speed resources. In future, proposed work can be improved by applying some Meta heuristic techniques like: GA, PSO and ACO etc and can be tested on real environment too.

5.13 Mid-Max Algorithm [23] - The Mid-Max heuristic starts with all unmapped tasks by calculating their completion. Then the task having overall midst completion time is picked and allocated to fastest resources (highest MIPS). The newly mapped task is removed from the set (MT) and the process repeats until all tasks are mapped successfully.

5.14 Grouping based User Demand Aware Job Scheduling approach for Computational Grid (GUDA) [24] - It is based on the concept of grouping user jobs and taking user's deadline as QoS parameter. Here fine grained jobs are grouped into coarse grained jobs where group of jobs is prepared according to some grouping scheme and then that group is scheduled to resources based on user's deadline. Result shows reduction in makespan and communication time compare to users' demand aware scheduling (UDDA) by means of grouping concept. Further the work can be extended by accounting load balancing as new angle of research.

5.15 Best-Min [25] –The basic idea was build around the conventional Min-Min, but this algorithm considers the rescheduling unlike fixed scheduling procedure in Min-Min. In Best-Min algorithm minimum completion time is compared against the makespan produced by Min-Min. If makespan value by Min-Min is smaller, then the task is rescheduled on the resource that produced it and the available time for all resources is updated. Otherwise task is scheduled in current resource as usual. No doubt rescheduling lead's to cost more.

5.16 Load Balance Improved Min-Min (LBIMM) and User-Priority Awared LBIMM (PA-LBIMM) [26] - A new scheduling algorithm for load balancing in cloud with respect to user's priority that was based on conventional Min-Min mapping. The experimental results show that under all possible situations both the LBIMM and PA-LBIMM are capable of decreasing completion time of tasks, improving load balance of resources and gain better overall performance than Min-Min algorithm.

5.17 Enhanced Max-Min [27] – It is a refinement of Improved Max-Min based on conventional Max-Min. In Improved Max-min algorithm, task is assigned to resource produces minimum completion time (slowest resource in terms of MIPS) while Enhanced Max-min assign task with average execution time (average or nearest greater than average task) to resource produces minimum completion time which leads to reduces overall makespan and balance load across resources over Improved Max-Min.

5.18 Enhanced Load Balanced Min-min (ELBMM) algorithm [28] - It is based on LBMM but with one difference that rather than choosing minimum execution time of task on heavy load resource, it chooses maximum execution time while rescheduling process. Results show ELBMM produces better result compare to LBMM.

5.19 Credit based scheduling algorithm in Cloud [29] - Another static scheduling technique under batch mode mapping which consider mainly two parameters that are: (1) Task Length (MI) and (2) User Priority (an

integer number). Here the algorithm is duly based on credit system, where each task is assigned a credit based on their task length and priority. From the simulation results it is concluded that, the proposed algorithm works efficiently in terms of makespan. In future, the proposed scheme can be enhanced by considering task's deadline and other vital QoS parameters.

5.20 Credit based scheduling using deadline in Cloud [30] - This algorithm is enhanced version of its predecessor [29]. Here each task credit is calculated based on three parameters that are: (1) Task's Length (2) User's Priority and (3) User's Deadline using a predefined formula. Results show proposed work performs better in terms of makespan compare to its predecessor, but further it requires cost of data transfer and rescheduling as future work.

Table 2: Batch mode independent task scheduling techniques

Technique/Parameters	Factors achieved	QoS	Hybrid	Remarks
1. Min-Min	Makespan	×	×	2 phases with smallest overall MCT
2. Max-Min	Makespan	×	×	2 phases with maximum overall MCT
3. Sufferage	Makespan	×	×	Execute a task that suffers the most
4. RASA	Makespan	×	YES	Based on resource count, applies Min-Min and Max-Min one by one
5. TASS	Makespan, Resource Utilization rate	×	YES	Based on task count, applies Max-Min and Sufferage one by one
6. QoS Guided Min-Min	Makespan	YES	×	Considers network bandwidth into account with Min-Min
7. QoS Priority Grouping	Makespan, Deadline, Acceptance rate	YES	×	Achieved better throughput
8. BAJGS	Transmission time, Resource utilization rate	YES	×	Based on both computation and communication
9. QoS Sufferage	Makespan	YES	×	Considers network bandwidth into account with Sufferage
10. Min-Min Max-Min Selective	Makespan	×	×	Considers task length (MI) and based on Standard deviation
11. LBMM	Load balancing, Makespan	×	×	Rely on re-scheduling
12. Improved Max-Min	Load balancing	×	×	Based on execution time, for future it can opt meta-heuristic
13. Mid-Max	Makespan	×	×	Based on fastest resource(s)
14. GUDA	Deadline	YES	×	Grouping of fine grained jobs into coarse grain
15. Best-Min	Makespan	×	×	Re-scheduling
16. LBIMM and PA-LBIMM	User's priority, Load balancing	YES	×	Consider both load balancing and user's priority
17. Enhanced Max-Min	Makespan, Load balancing	×	×	Enhancement of Max-Min
18. ELBMM	Makespan	×	×	Re-scheduling
19. Credit based	Makespan, User's priority	YES	×	Based on credit= Task length + user's priority
20. Credit based with Deadline	Makespan, User's priority, Deadline	YES	×	Here deadline is added as one of the QoS parameter

6. CONCLUSION

This paper reviews famous independent task scheduling techniques under heuristic mapping for both immediate as well as batch mode in the area of Cloud and Grid environment. Heuristic techniques are usually faster than Meta-heuristic techniques and the generated solutions by heuristic techniques are usually optimal for problem that is not large enough as per experiments performed. During literature survey, we have found that most of the researchers have focused on makespan, average resource utilization rate of resources and QoS parameters. To achieve all the required factors in a single technique is yet another challenging task to be resolved in future. In this paper we have discussed techniques so that one can select any with respect to their speciality and their pitfalls (if any) accordingly.

7. REFERENCES

[1] Mell P and Grance T. The NIST definition of cloud computing. US department of Commerce. 2011, Sep.

[2] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems. 2009 Jun 30;25(6):599-616.

[3] Desai T, Prajapati J. A survey of various load balancing techniques and challenges in cloud computing. International Journal of Scientific & Technology Research. 2013 Nov 25;2(11):158-61.

[4] Kokilavani T, Amalarethinam DD. Load balanced min-min algorithm for static meta-task scheduling in grid

computing. International Journal of Computer Applications. 2011 Apr;20(2):43-9.

[5] Henzinger TA, Singh AV, Singh V, Wies T, Zufferey D. Static scheduling in clouds. memory. 2011 Jun 14;200(o1):i1.

[6] Xhafa F, Abraham A. Computational models and heuristic methods for Grid scheduling problems. Future generation computer systems. 2010 Apr 30;26(4):608-21.

[7] Maheswaran M, Ali S, Siegal HJ, Hensgen D, Freund RF. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. InHeterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth 1999 (pp. 30-44). IEEE.

[8] What are the differences between heuristics and metaheuristics? https://www.researchgate.net/post/What_are_the_differences_between_heuristics_and_metaheuristics.

[9] Annette J R, Banu W A, Shriram S. A taxonomy and survey of scheduling algorithms in cloud: based on task dependency. International Journal of Computer Applications. 2013 Nov;82(15):20-6.

[10] Mangla N, Singh M. Workflow Scheduling In Grid Environment, IJERA, 2014.

[11] Braun TD, Siegel HJ, Beck N, Bölöni LL, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B, Hensgen D, Freund RF. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing. 2001 Jun 30;61(6):810-37.

[12] Sharma G, Banga P. Classifier MCT for immediate mode independent task scheduling in Computational Grid. International Journal of Engineering Trends and Techonology.;1(4):2722-6.

[13] Gupta K, Singh M. Heuristic based task scheduling in Grid. International Journal of Engineering and Technology. 2012 Aug;4(4):254-60.

[14] Parsa S, Entezari-Maleki R. RASA: A new task scheduling algorithm in grid environment. World Applied sciences journal. 2009;7:152-60.

[15] Sharma G, Banga P. Task aware switcher scheduling for batch mode mapping in computational grid environment. International Journal of Advanced Research in Computer Science and Software Engineering. 2013 Jun;3.

[16] He X, Sun X, Von Laszewski G. QoS guided min-min heuristic for grid task scheduling. Journal of Computer Science and Technology. 2003 Jul 1;18(4):442-51.

[17] Dong F, Luo J, Gao L, Ge L. A grid task scheduling algorithm based on QoS priority grouping. In2006 Fifth International Conference on Grid and Cooperative Computing (GCC'06) 2006 Oct (pp. 58-61). IEEE.

[18] Keat NW, Fong AT, Chaw LT, Sun LC. Scheduling framework for bandwidth-aware job grouping-based scheduling in grid computing. Malaysian Journal of Computer Science. 2006;19(2):117-26.

[19] Munir EU, Li J, Shi S. QoS sufferage heuristic for independent task scheduling in grid. Information Technology Journal. 2007 Aug;6(8):1166-70.

[20] Etminani K, Naghibzadeh M. A min-min max-min selective algorihtm for grid task scheduling. InInternet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on 2007 Sep 26 (pp. 1-7). IEEE.

[21] Kokilavani T, Amalarethinam DD. Load balanced min-min algorithm for static meta-task scheduling in grid computing. International Journal of Computer Applications. 2011 Apr;20(2):43-9.

[22] Elzeki OM, Reshad MZ, Elsoud MA. Improved max-min algorithm in Cloud computing. International Journal of Computer Applications. 2012 Jan 1;50(12).

[23] Laxmi V, Kaur N. Batch Mode Scheduling-Mid_Max Algorithm. International Journal of Computer Applications. 2012 Jan 1;49(15).

[24] Suresh P, Balasubramanie P. Grouping based user demand aware job scheduling approach for computational grid. International Journal of Engineering Science and Technology. 2012 Dec;4(12):4922-8.

[25] Meraji S, Salehnamadi MR. A batch mode scheduling algorithm for grid computing. Journal of Basic and Applied Scientific Research. 2013;3(4):173-81.

[26] Chen H, Wang F, Helian N, Akanmu G. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. InParallel Computing Technologies (PARCOMPTECH), 2013 National Conference on 2013 Feb 21 (pp. 1-8). IEEE.

[27] Bhoi U, Ramanuj PN. Enhanced max-min task scheduling algorithm in cloud computing. International Journal of Application or Innovation in Engineering and Management. 2013 Apr;2(4):259-64.

[28] Patel G, Mehta R, Bhoi U. Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing. Procedia Computer Science. 2015 Dec 31;57:545-53.

[29] Thomas A, Krishnalal G, Raj VJ. Credit based scheduling algorithm in cloud computing environment. Procedia Computer Science. 2015 Dec 31;46:913-20.

[30] Sharma A, Sharma S. Credit based scheduling using deadline in Cloud Computing environment. 2016 Feb: 4(2): 1588-1594.