Volume 8, No. 4, May 2017 (Special Issue)
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
REVIEW ARTICLE
Available Online at www.ijarcs.info

The Survey of Enhanced Min-Max Approach For Resource Aware Job Scheduling In Cloud Computing

Harpreet Kaur Sachdeva
Computer Science Department
Chandigarh Engineering College, Landran
Punjab, India
cse11309.sbit@gmail.com

Anurag Jain
Computer Science Department
Chandigarh Engineering College, Landran
Punjab ,India
er.anuragjain@gmail.com

Abstract:***The idea of cloud computing encourages the working of a simple computer to work together and form super computer. The scheduling of jobs in cloud environment is done by choosing the finest and most appropriate resource available for completion of jobs or to distribute computer equipments to jobs in such a manner that the achievement time is reduced as possible. Job scheduling has become an active research area. A set of priority of different jobs, based on various parameters is formed. Jobs are then decided ranked and allocated to available processors and computer machines which satisfy a predefined objective function. This paper includes the study of different job scheduling algorithms in cloud environment. The main is to reduce the turnaround time and to improve the resource utilization.***

Keywords:cloud computing, job scheduling, Heterogeneous Earliest Finish Time, Enhanced Min-Max Algorithm

I. INTRODUCTION

Cloud computing could be an innovation that uses the web and local remote [1] servers to keep data and perform various functions. Distributed, cluster and grid computing are the foundation stones of this evolving technology called cloud computing. Cloud computing licenses buyers and in addition, organizations to utilize functions while not establishment and contact their own documents at any PC by web contact. This innovation licenses for a lot of capable [2] computing by integrative data stockpiling, procedure and measure data.

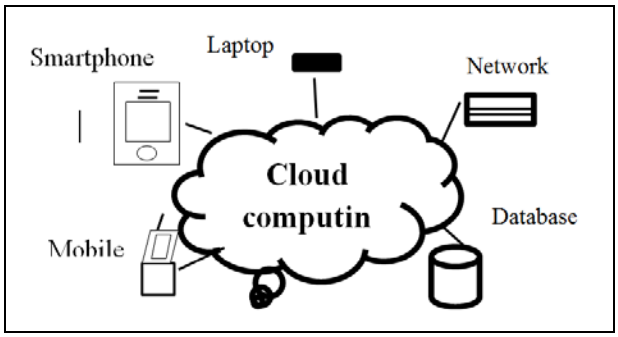


Figure 1. Cloud Computing Environment.

The most apt example of cloud computing is Yahoo email, Gmail, or Hotmail and so on. All is necessary to have essentially a web alliance and having the capacity to start sending electronic mail. The principle server and in addition, electronic mail administration, programming, framework is all in the cloud (web) and is totally overseen by the cloud administration dealer Yahoo, Google and the purchaser gets the chance to utilize the bundle alone and delight in the preferences.

Cloud computing could be a general term at any cost that includes delivering hosted services over the web. Administrations are extensively talking separated into three classes:

- Infrastructure-as-a-Service (IaaS),
- Platform-as-a-Service (PaaS) and
- Software-as-a-Service (SaaS).

II. JOB SCHEDULING

Job scheduling algorithm is a method by which jobs are matched, or allocated to data center resources. Due to contradictory scheduling objectives normally no absolutely perfect scheduling algorithm exists. A good scheduler implements a suitable compromise or applies a combination of scheduling algorithms according to different functions. A problem can be solved in seconds, hours or even years, depending on the algorithm applied. The efficiency of an algorithm is evaluated by the amount of time necessary to execute it. The execution time of an algorithm is stated as a time complexity function related to the input. There are several kinds of time complexity algorithms that appear in the literature. If 20 problems have a polynomial time algorithm, the problem is tractable, feasible, efficient or fast enough to be executed on a computational machine. In computational complexity theory, the set of problems can be treated as complexity class based on a certain resource.

In job scheduling many advantages like Current use of all AIS resources, Increased Throughput or Accuracy, Less improvement period, customer limits meet, consumer made answerable for provided that input on agenda, superior infrastructures with customer, Prevention of congestion and underuse of capitals, work setback more willingly clear, Reduced misunderstanding within the AIS facility, Better use of multi-programming competences [3], Predictability of the effects of an increased assignment Some drawback of Job scheduling likework Quality, Reduced Absenteeism and turnover, Inequity, Less Teamwork.

III. RELATED WORK

Kumar et.al, 2016 [4] demonstrated a two-level load balancer approach by combining join idle queue and join shortest queue

approach. Authors have used cloud analyst stimulant to test proposed two-level load balancer approaches. The results are analyzed and compared with the existing algorithms and as observed, the proposed work is one step ahead of existing techniques. **Hasan Mahmud et.al; 2016 [5]** considered a practical hybrid data center infrastructure (including both self-managed and colocation data centers) and suggest a novel resource management algorithm based on alternating direction method of multipliers, called CAGE (Carbon and Cost-Aware Geographical Job Scheduling) to decrease carbon footprints. CAGE dynamically distributes incoming workloads to Geo-distributed data centers based on local renewable availability, carbon efficiency, electricity price, and also the energy usage of other tenants that share the colocation data centers. . **Dr. Rajneesh Ahmad Abba Haunt et.al;2015 [6]** describeda Hybrid Integrated Thermal-Aware Scheduling Algorithms based on baseline approaches. The aim of this paper is to minimize cooling energy consumption in data center labs when assigning jobs for computation. These algorithms avoid high thermal stress situations such as large hot spots and thermal violations events. **Anurag Jain et.al 2014 [7]** defined "cloud computing"for the beginnersand give the details about its taxonomy. Also, this paper provides an idea of the design challenges of cloud computing and helps in identifying important research directions in this area. In another paper [8], author does a detailed discussion about a two level load balancer approach by using join idle queue and join shortest queue approach together.**Hung-Jui Chang et.al; 2009 [8]** proposes scheduling algorithms for assigning jobs with different release time and execution time, to machines with heterogeneous processor cardinality. The author shows that these scheduling problems is NP-complete, and propose a dynamic programming to find the optimal schedules. Since the dynamic programming was time-consuming the proposed techniques that improve the efficiency of the dynamic programming. And also propose heuristic algorithms for this scheduling problem. Experimental results suggest that particular of the heuristics not only compute the answer efficiently, but also provide a good solution.

Table 1. Technique Used

Author Name	**Year**	**Techniques used**	**Parameter**
Dr. Rajneesh Kumar et.al	2016	Join the idle queue, join the shortest queue	Data Processing Time, Response Time, Cost
Hasan Mahmud et.al;	2016	CAGE (Carbon and Cost-Aware Geographical Job Scheduling)	Network and processing delays, Reduction compared to PerfMax, Total carbon Emission, CostMin, PerfMax
Ahmad Abba Haunt et.al;	2015	First Come First Serve and Round Robin	Electricity Consumption and power usage effectiveness
Anurag Jain et.al	2014	-	-
Hung-Jui Chang et.al;	2009	Enhanced Min-Max algorithm	Resources, Bandwidth and Processing Speed

IV. VARIOUS ALGORITHM IN JOB SCHEDULING

A. Resource Aware scheduling algorithm

The resource-aware scheduling algorithm called RASA, which searches and deletes redundant task duplications dynamically in the process of scheduling. A further optimizing scheme is designed for the schedules generated by our algorithm, which can further reduce resource consumption without degrading the makespan. Experiments are conducted to verify that both the proposed algorithm and the optimizing scheme can achieve good performance in terms of makespan and resource efficiency. The factors affecting the performance of our algorithm are analyzed. Introduce the resource-aware job schedulers to the MapReduce framework. However, these schedulers specify a fixed size for each task in terms of required resources (e. g. CPU & memory), thus pretentious the run-time resource consumption of the task is stable over its lifetime. In particular, it has been reported that the execution of each MapReduce task can be divided into multiple phases of data transfer, processing and storage. A task is divided into small unequal sizes called phases. The phases involved in the same task can have dissimilar resource demand in terms of CPU, memory, disk & network usage. Therefore, scheduled tasks based on fixed resource requirements over their durations will often cause either excessive resource contention by scheduling too many simultaneous tasks on a machine.
SaeedParsa and Reza Endear Maleki proposed RASA a new task scheduling algorithm, it is composed of Max-min and Min -min the two traditional scheduling algorithms[11]. It considers the advantages of Max-min and Min-min algorithms and covers the disadvantages.

B. Heterogeneous Earliest Finish Time Algorithm

Heterogeneous Earliest Finish Time (or HEFT) is a heuristic method to schedule a set of dependent tasks onto a network of heterogeneous workers taking communication time into account. HEFT takes a set of tasks represented as a directed acyclic graph, a set of workers, the times to execute each task on each worker, and the times to communicate the results from every job to every of its children between every pair of workers. It descends fromthe list of scheduling algorithms.
The HEFT algorithm is highly competitive in that it generates a schedule length compared to the schedule lengths of other scheduling algorithms with a lower time complexity. The HEFT algorithm has two phases: a task prioritizing and a processor selection phase. In the first phase task, priorities are defined as rank u. rank u represent the length of the longest path from task ni to the exit node, including the computational cost of ni , and is given by rank u(ni) = wi+maxnj∈succ(ni){ci,j +rank u(nj)}. For the exit task, rank u(nexit) = wexit. The task list is ordered by decreasing value of rank u. In the processor selection phase, the task on top of the task list is assigned to the processor pj that allows for the

EFT (Earliest Finish Time) of task ni . However, the HEFT algorithm uses an insertion policy that tries to insert a task in at the earliest idle time between two already scheduled tasks on a processor, if the slot is large enough to accommodate the task.

The aim of efficient scheduling is to map the tasks onto processors and execution order is set onto the processors and execution order is set so that task precedence requirements are satisfied and minimum schedule length is given. HEFT algorithm calculates average execution time first for each task and average communication time between resources of two immediate successive tasks. Then tasks in the workflow are ordered (not - increasing) based on a rank function. Higher priority is given to the task with higher rank value. The tasks are executed in the order of their priorities and each task is mapped to the resource that earns complete the task at the earliest time in the in the resource selection phase.

C. Enhanced Max-min Algorithm

When a larger job is assigned to the slower resource, it results in increase in the make span. Sometimes jobs in a meta-task largest task are too large compared to the other tasks. Max,Min and RASA algorithm suffer from this limitation. Here the new approach is used for assigning the task to the resource and the proposed solution is, first by selecting the job just greater than the average execution time and mapping to the slowest resource [9].

Sometimes the largest task is too large compared to other tasks in Mata-task, in that kind of case overall makespan is increased because too large task is executed by slowest resource first while other tasks are executed by quicker resource OR when there is major variance among slowest and fastest resource in the context of processing speed or bandwidth in that case largest task is executed by slowest resource cause increasing in Makespan and load imbalance across resources. Therefore, instead of selecting largest task if the select Average or nearest greater than the average task, then overall makespan is reduced and also balance load across resources [10].

Pseudo code Enhanced Max-min Algorithm

1. For all submitted tasks in Meta-task; Ti
   For all resources; RJ
   Cij = Eiji + RJ
2. Find task Tk costs Average or nearest Greater than Average execution time.
3. Assign task Tk to resource R which gives t minimum completion time (Slowest resource).
4. Remove task Tk from Meta-tasks set.
5. Update RJ to selected RJ.
6. Update Cij for all j.
7. While Meta-task not Empty
   a) Find task Tk costs maximum completion time.
   b) Assign task Tk to resource R which gives minimum execution time (Faster Resource).
   c) Remove Task Tk form Meta-tasks set.
   d) Update rj for Selected Rj .
   e) Update Cij for all j.

So in Enhanced Max-min, task selection scenario is changed, it is stated as "Select task with Average or nearest greater than average execution time (Average or Nearest greater than average task) then assign to be executed by a resource with minimum completion time (Slowest resource)".

4.4 **Min-Max Algorithm**

The Max-min Algorithm is based on "select a task with maximum execution time and assign to the resource with minimum execution time"Minimax (sometimes Min Max or MM) is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. Originally formulated for two-player zero-sum game theory, covering together the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision-making in the presence of uncertainty.

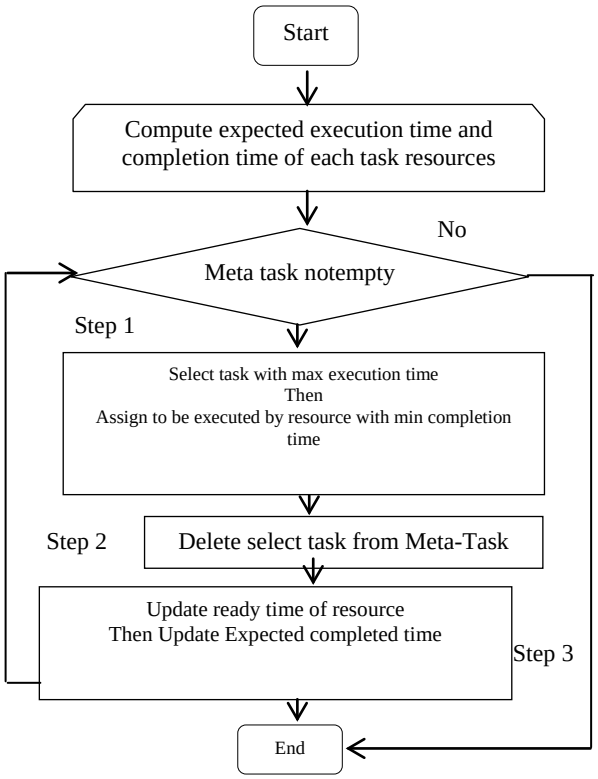
Max-min algorithm allocates task Ti on the resource Rj Where large tasks have the highest priority rather than smaller tasks. For example, if one long task, the Max-min could execute many short tasks concurrently while executing large one. The total makes span, in this case, is determined by the execution of the long task. But if meta-tasks comprises tasks have relatively dissimilar completion time and execution time, the makespan is not determined by one of the submitted tasks. It would be similar to the Min-min makespan. For these cases, original Max-min algorithm losses some of its major advantages as load balance between available resources in small distributed system configuration and small total completion time for all submitted tasks in large-scale distributed environment. Author can't use the Max-min and wait submitted tasks to decide what would be the allocation map, make span, load balance, etc. Author try to minimize waiting time of short jobs through assigning large tasks to be executed by slower resources. On the other hand, execute small tasks concurrently on a fastest resource to finish a large number of tasks during finalizing at least one large task on a slower resource. Based on these cases, where meta-tasks contain homogeneous tasks of their completion and execution time, a substantial improvement of the Max-min algorithm that leads to increasing of Max-min efficiency. Proposed improvement increases the opportunity of concurrent execution of tasks on resources. Author focus on the Max-min to derive improved Max-min because of its advantages as load balance that is desired in small distributed system rather than larger and small make span in large distributed system rather than small. True, load balance enhances performance in distributed systems but doesn't necessarily result in shorting makespan. The algorithm calculates the probable completion time of the submitted tasks on every resource. Then the task with the overall maximum expected execution time is assigned to a resource that has the smallest overall completion time. Lastly, this scheduled tasks removed from meta-tasks and all calculated times are updated and the processing is repeated until all submitted tasks are executed. The algorithm focuses on minimizing the total makespan which is the total complete time in large distributed environment, for example, cloud computing environment also, and executing tasks concurrently on available resources achieving load balance in small distributed system. The proposed algorithm produces mapping schema similar to RASA in such concurrency executing tasks and minimization of total completion time required to finish all tasks. Selecting task with maximum execution time leads to choosing largest task should be executed. While selecting resource overwhelming minimum completion time means choosing slowest resource in the obtainable resources. So

allocation of the slowest resource to longest task allows availability of high-speed resources for finishing other small tasks concurrently. Also, achieve the shortest make span of submitted tasks on available resources besides concurrently. Not as original Max-min which recommended to be used if and only if submitted tasks are heterogeneous in their completion time and execution time, by means, there are clearly large tasks and small tasks.

Pseudo code of Min-Max Algorithm

1. for all submitted tasks in meta-task; Ti
2. for all resources; RJ
3. Cij = Eiji + RJ
4. While meta-task is not empty
5. Find task Tk costs maximum execution time.
6. Assign Tk to the resource RJ which gives minimum completion time.
7. Remove Tk from meta-tasks set
8. Update for selected RJ
9. Update Cij for all j

Figure2. Improved Min-Max Algorithm



V. CONCLUSION

In Max-min Algorithm the largest task is assigned to the best available resource (fastest). One of the limitations of this algorithm is when the larger job assigned to the slower resource results with the increase in the Make span. In the proposed algorithm the probability of assigning the larger jobs to the slower resource is minimized. Here the resources are divided into two sets, first set with slow resources and the second set of fast resources. Based on the available fast resource task has been selected. If the selected resource belongs to the first set then the task with a length just greater than the average is selected else task with maximum length is selected.

REFERENCES

[1] VasiliosAndrikopoulos, Zhe Song, Frank Leymann, "Supporting the Migration of functions to the Cloud through a Decision Support System", Institute of Architecture of function Systems, IEEE, pp. 565-672, 2013.

[2] Haitao Li, LiliZhong, Jiangchuan Li, Bo Li, KeXu, " Cost-effective Partial Migration of VoD Services toContentClouds", 2011 IEEE 4th International Conference on Cloud Computing, pp. 203-110, 2011.

[3] Monjur Ahmed and Mohammad Ashraf Hossain, "CLOUD COMPUTING and SECURITY ISSUES IN THE CLOUD" International Journal of Network Security and Its functions (IJNSA), Vol.6, No.1, January 2014.

[4] Jain, Anurag, and Rajneesh Kumar. "A multi-stage load balancing technique for a cloud environment." In *Information Communication and Embedded Systems (ICICLES), 2016 International Conference on*, pp. 1-7. IEEE, 2016.

[5] Mahmud, A. Hasan, and S. S. Iyengar. "A Distributed Framework for Carbon and Cost-Aware Geographical Job Scheduling in a Hybrid Data Center Infrastructure." In *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pp. 75-84. IEEE, 2016.

[6]Haruna, Ahmad Abba, Low T. Jung, and NordinZakaria. "Design and Development of Hybrid Integrated Thermal-Aware Job Scheduling on Computational Grid Environment.",2014

[7] Jain, Anurag, and Rajneesh Kumar. "A Taxonomy of Cloud Computing." *International Journal of Scientific and Research Publications* 4, no. 7 (2014): 1-5.

[8] Chang, Hung-Jui, Jan-Jan Wu, and Pangfeng Liu. "Job scheduling techniques for distributed systems with heterogeneous processor cardinality." In *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pp. 57-62. IEEE, 2009.

[10] Bhoi, Upendra, and Purvi N. Ramanuj. "Enhanced max-min task scheduling algorithm in cloud computing." *International Journal of Application or Innovation in Engineering and Management* 2, no. 4 (2013): 259-64.

[11] Parsa, Saeed, and Reza Entezari-Maleki. "RASA: A new task scheduling algorithm in the grid environment." *World Applied sciences journal* 7 (2009): 152-160.