Priority Based Job Scheduling For Heterogeneous Cloud Environment

S.Rekha[1]and R.Santhosh Kumar[2]

[1]Information Technology, Sri Venkateswara College of Engineering, Sriperumbudur, Tamil Nadu, India

[2]Information Technology, Sri Venkateswara College of Engineering, Sriperumbudur, Tamil Nadu, India

Abstract

Cloud computing is a form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large task. It has emerged as a strong domain in the field of networking primarily due to the ability of running an application or program simultaneously on multiple nodes that are connected through a network. Hence, it involves sharing of resource or computational information amongst the nodes. A proper job-scheduling algorithm is required for the efficient functioning of the cloud environment. The proposed priority based scheduling algorithm for cloud computing is based on factors that govern the functioning of a job.

Keywords: *Cloud Computing, Job Scheduling, Priority, Computational Complexity and Level of Parallelism.*

1. Introduction

In cloud computing, multiple nodes process large amount of data and perform complex computations. Hence jobs arriving to be executed must be scheduled effectively since job delays or data loss is not acceptable in a highly clustered network. The main aim of the job-scheduling algorithm is to increase the throughput of the system and improve the performance. The existing Batch mode heuristic scheduling algorithms (BMHA) are: First Come First Served scheduling algorithm (FCFS), Shortest Job Fastest Resource (SJFR),Longest Job Fastest Resource (LJFR),Min–Min algorithm and Max–Min algorithm. These algorithms consider all jobs with equal importance. This cannot be the scenario when some jobs have to be executed prior to others. The proposed Priority Based Scheduling algorithm resolves this issue. This algorithm proves to be efficient as it considers the computational complexity, level of parallelism, no of resources available and so on. The paper is organized as follows: section 2 deals with the cloud framework, section 3 describes the proposed algorithm, section 4 provides a picture of performance analysis and section 5 concludes the paper.

2. Cloud Framework

The cloud framework or architecture comprises of the cloud components that communicative with each other and deliver the output. These components are classified into two categories: front end platforms such as mobile devices or any client and back end platform : servers, storage and a network. This is shown in fig 1.

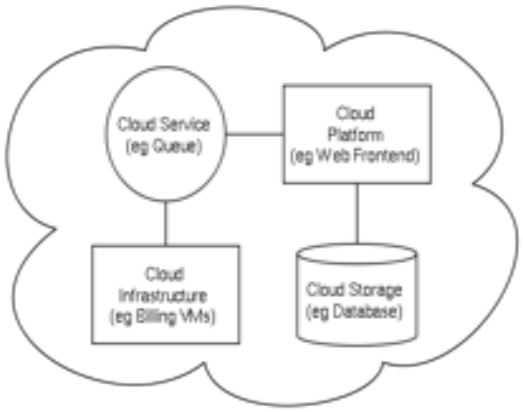

Fig 1. Cloud Computing Architecture

Jobs arrive at the scheduler to be executed along with request for cloud resources. The function of the scheduler is to select how several incoming jobs have to be processed and allocate resources wisely. The overall performance and throughput of the system depends on how the scheduler works. The scenario is depicted in fig 2.

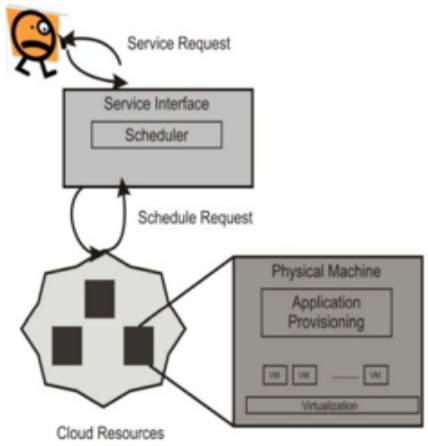

Fig 2 : Cloud job scheduler.

3. Proposed algorithm - Priority Scheduling in Cloud Computing

In the proposed priority scheduling algorithm in cloud computing, a parameter named "priority" has been introduced. The jobs are classified into high, medium and low based on the priority. The priority is assigned based on the computational complexity of the job and level of parallelism of the resources. The level of parallelism of a resource and computational power of a job is decided by considering the job parallelism, resource parallelism and job's computational complexity respectively. In this algorithm, a higher priority is assigned to job of higher computational complexity and the resource exhibiting higher level of parallelism .The fastest resource available is assigned to the job of high priority. This priority algorithm optimizes the computational speed of the cloud and reduces the usage of nodes and also shows a consistent performance during execution of the assigned jobs.

3.1. Computational Complexity

Task partitioning algorithm takes care of efficiently dividing a given job into subtasks of appropriate grain size and an abstract model of such a partitioned application is

represented by a Directed Acyclic Graph (DAG). Each task can be executed on a processor and the directed arc shows transfer of relevant data from one processor to another. Each node in DAG represents sequence of operations. .Each task of a DAG corresponds to a sequence of operations and a directed arc represents the precedence constraints between the tasks. All the operations are represented in terms of additions. The amount of computations involved in a particular node is represented by node weight.

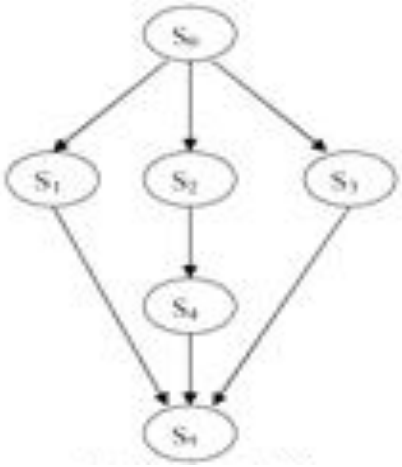

Fig 3 : Directed Acyclic Graph (DAG)

This graph needs to be traversed to find out the longest path. The total sum of the amount of computations involved in each node through which the traversal has been performed leads to computational complexity of the application.

3.2. Level Of Parallelism

Generally, the amount of parallelism exhibited by a job is computed and analysed by analysing it's layered DAG representation. The width of the DAG is equal to the number of sub tasks which are executed through parallelism. The maximum number of independent instructions getting executed in a unit time (in one clock cycle) is equal to the width of the DAG that gives the amount of parallelism exhibited by the job. The amount of parallelism exhibited by a resource is computed by considering the number of operations per cycle per processor, number of processors per node and number of nodes in a system. The amount of parallelism exhibited by each free resource available in the cloud is computed. The amounts of parallelism exhibited by all the available free resources in the cloud are fixed by analysing the max, min and mid ranges. The value for the level of parallelism is

assigned by comparing the amount of parallelism exhibited by each job with the max, min and mid ranges.

3.3. Priority Assignment

When it comes to Priority based job scheduling in cloud higher priority is assigned generally to a job which needs high computational power and which exhibits high parallelism. A job, which exhibits low parallelism and needs low computational power for execution is given a low priority. A job, which exhibits a medium level of parallelism and needs medium computational power, is given a medium priority. The fastest free resource available in the cloud is allocated to the job which has high priority. The procedure is given below :

Amount of Parallelism = *OC* PN*NS*
Where OC= No. of operations per cycle per processor
PN= No.of processors per node
NS=No. of nodes in a cloud.

Let m represent number of free resources available in the cloud and n represent the number of jobs present in the queue. The worst case time complexity of the algorithm is O(n logn) ,when m <= n and O(m logm) when m > n.

3.4. Proposed Algorithm

```
AssignLevelofParallelism( ResourceList Rs_List)
While(Rs_List!=NULL)
For each resource
/*OC = No. of operations per cycle per processor
PN = No. of processors per node
NS = No. of nodes in a coud*/
/* LL_List contains the amount of parallelism
Exhibited by each resource */
LL_List[i] = OC*PN*NS
End While
Find the Max, Min and Mid values in PR_List
/* LJ_List contains the amount of parallelism exhibited
by each job */
For each job in LJ_List
If LJ_List[i] >= Maximum
LP_List[i] = High //LP_List contains the level of
parallelism value
Else If LJ_List[i] >= Middle
LP_List[i] = Medium
Else LP_List[i] = Low
EndIf
End AssignLevelofParallelism
Assign Priority Procedure
AssignPriority ( CloudList CL_List)
While( CL_List !=NULL)
For each job
/* CompC_List contains the Computational Complexity of
jobs */
If (CompC_List[i] =High AND LP_List[i] = High)
Priority[i] = 1
Else If (CompC_List[i] = High AND LP_List[i] =
Medium)
Priority[i] = 2
Else If (CompC_List[i] = High AND LP_List[i] =Low)
Priority[i] = 3
Else If (CompC_List[i] = Medium AND LP_List[i] =
High)
Priority[i] = 4
Else If (CompC_List[i] = Medium AND LP_List[i] =
Medium)
Priority[i] = 5
Else If (CompC_List[i] = Medium AND LP_List[i] =
Low)
Priority[i] = 6
Else If (CompC_List[i] = Low AND LP_List[i] = High)
Priority[i] = 7
Else If (CompC_List[i] = Low AND LP_List[i] =
Medium)
Priority[i] = 8
ElseIf (CompC_List[i] = Low AND LP_List[i] = Low)
Priority[i] = 9
EndIf
End AssignPriority
```

4. PERFORMANCE STUDY

Here we compare the perfomance of our priority Based scheduling algorithm for resources in cloud with the existing job scheduling algorithms in cloud First Come First Serve, Shortest Job Fastest Resource, Longest Job Fastest Resource and Min Min algorithm and Max Min algorithm.

4.1. First Come First Serve (FCFS)

This algorithm schedules the job to the available resources on the cloud on the “First Come First Serve” basis .From the Fig 4, we get that FCFS algorithm is basic and does not consider the factors like computational complexity and level of parallelism during scheduling .It shows very low

computation results compared to other scheduling algorithms.

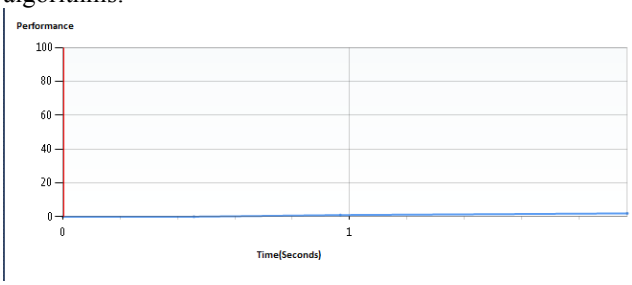

Fig 4:Implementation of FCFS

4.2. Shortest Job Fastest Resource (SJFR)

Shortest Job Fastest Resource is a scheduling algorithm, assigns the job with very low turnaround time to the fastest resources in the cloud . From the Fig 3 we can decipher that SJFR is more stable in handling jobs and hence outperforms FCFS scheduling algorithm.

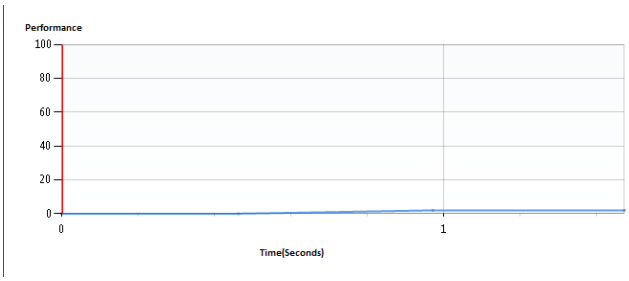

Fig 5:Implementation of SJFR

4.3. Longest Job Fastest Resource (LJFR)

Longest Job Fastest Resource is a scheduling algorithm that assigns the complex job to a big efficiency resource . It tries to reduce the overall execution time of the jobs. From the Fig 3 of the LJFR algorithm we can infer that LJFR outperforms FCFS and the SJFR as the jobs of high computational complexity are assigned to faster resources in the cloud which leads to shorter execution time.

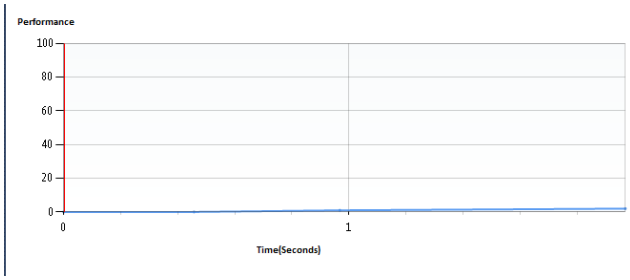

Fig 6:Implementation of LJFR

4.4. Min-Min Algorithm

The Min-Min algorithm schedules the less complex jobs to high performance resources for execution. It is similar to the Shortest Job Fastest Resource (SJFR) algorithm. From the Fig 3 we can observe that outperforms FCFS but shows low performance comparing the other algorithms due to the delay caused in execution of complex jobs.

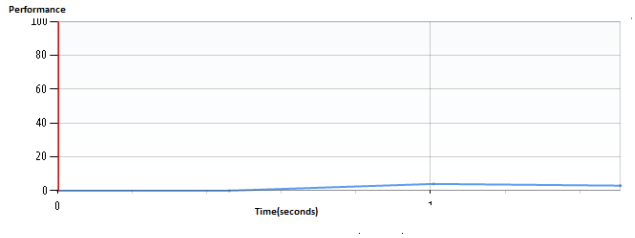

Fig 7: Implementation of Min-Min

4.5 Max-Min Algorithm

The complex job is scheduled first to high performance resources in the cloud and leads to the long delay in the execution of less complex jobs. This is similar to the Longest Job Fastest Resource (LJFR) algorithm. Fig 3 shows the performance of the Max-Min algorithm where it outperforms FCFS, SJFR, MIN-MIN algorithms.

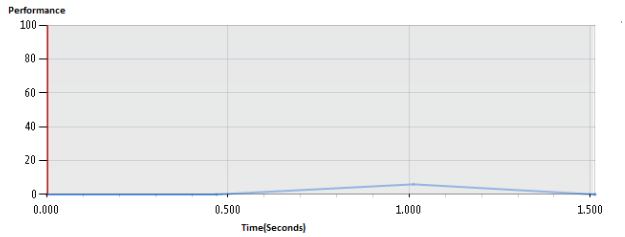

Fig 8:Implementation of Max-Min

4.6. Priority Based Algorithm

The Priority based Algorithm generally is based on a new concept “priority”. In this generally a job which needs high computational power and which exhibits high parallelism is given a high priority. A job, which exhibits low parallelism and needs low computational power, is given a low priority. A job, which exhibits a medium level of parallelism and needs medium computational power, is given a medium priority. The fastest free resource available in the cloud is allocated to the job which has high priority. The job with medium computational complexity and medium level of parallelism are given a first priority. This method of prioritizing enhances the rate of completion of jobs with a greater accuracy and with a proper usage of resources. From Fig 3 of the Priority based algorithm, we can state that Priority based outperforms FCFS, SJFR , LJFR,MIN-MIN and MAX-MIN due to its enhanced usage of resources.

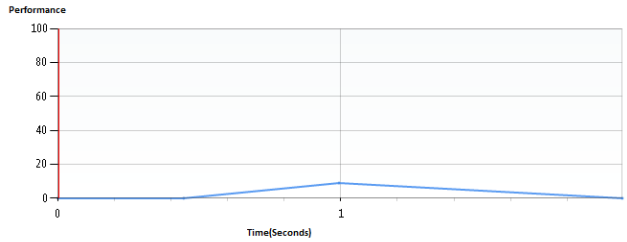

Fig 9: Implementation of Priority

4.7. Comparative Study Between Priority Based Cloud Scheduling Algorithm And Other Algorithms

Priority based algorithm provides Maximum economic utilization of resources that are provided for Low Computational Complexity, exhibiting High Parallelism and High Complexity, exhibiting Low Parallelism compared to other task scheduling algorithms like FCFS, LJFR, LJFR, MIN-MIN and MAX-MIN . This gives the best execution results among all the algorithms. It helps in scheduling tasks that exhibit medium computational complexity, medium parallelism. Priority based algorithm also optimizes the allocation of resources for completion of complex tasks with comparatively higher performance than LJFR, SJFR, MAX-MIN and MIN-MIN. Based on Fig 4 we get a clear idea about the performance of various global scheduling algorithms. Among all the above listed algorithms Priority based algorithm as seen from the graph is the best algorithm that provides efficient load balancing ,and better computation with efficient usage of resources in scheduling job in a cloud with heterogeneous resources. This is shown in fig10 .

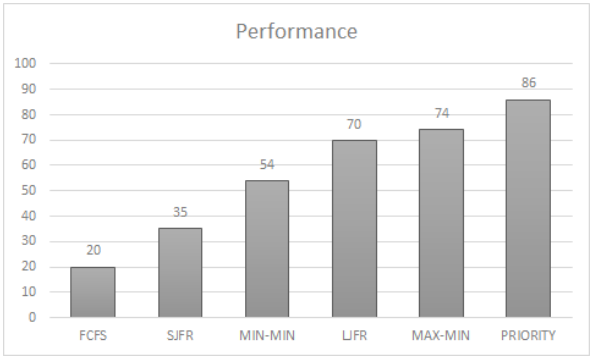

Fig. 10: Performance Chart

5.CONCLUSION

The cloud is a heterogeneous environment and designing a scheduling algorithm with an aim to perform scheduling a job to a resource in a optimized way has been a complex task.The fundamental algorithms (FCFS, SJFR,LJFR,MIN-MIN,and MAX-MIN) schedule the jobs based on computational complexity of the jobs and the speed of the resources. In the Priority based algorithm parameter named "priority" is used in the analysis and the jobs are classified into high, medium and low categories .In the Priority Based algorithm the jobs that possess a high computational complexity and the nodes that exhibits high level of parallelism is given a high priority. This method of prioritizing the jobs leads to completion of the job with high efficiency, lesser execution time with the usage of lesser number of resources and also shows consistency during the execution of the assigned tasks. The effectiveness of our algorithm is evaluated through simulation results and its superiority over other known algorithms is demonstrated.

6.REFERENCES

[1]. Dr.G.Sumathi,R.Santhosh Kumar,and S.Sathyanarayanan, "MidSFN Local Scheduling Algorithm for Heterogeneous Grid Environment", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012

[2]. Shamsollah Ghanbari, Mohamed Othman, "A Priority based Job Scheduling Algorithm in Cloud Computing", Procedia Engineering 50 (2012) 778 – 785.

[3]. Stelios Sotiriadis, Nik Bessis, Nick Antonopoulos, "Towards inter-cloud schedulers:A survey of meta-scheduling approaches" ,2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing

[4]. Mladen A. Vouk, "Cloud Computing – Issues, Research and Implementations",Journal of Computing and Information Technology - CIT 16,2008,4,235246doi:10.2498/cit.1001391

[5]. Anand, L., Ghose, D., and Mani, V., ELISA: an estimated load information scheduling algorithm for distributed computing systems. Computers & Mathematics with Applications, 37(8):57-85, 1999.

[6]. Yun-Han Lee et al, Improving Job Scheduling Algorithms in a Grid Environment, Future Generation Computer Systems, 27(2011) 991–998

[7]. Wei Wang, Cloud-DLS: Dynamic Trusted Scheduling for Cloud Computing, Expert Systems with Applications 39 (2012) 2321–2329.

[8]. Tai-Lung Chen et al, Scheduling of Job Combination and Dispatching Strategy for Grid and Cloud System, GPC,(2010) 612–621.

[9]. Domagoj Jakobovi´c et al, Evolving priority scheduling heuristics with genetic programming, Applied Soft Computing 12 (2012)2781–2789.

[10]. Monir Abdullah, Mohamed Othman et al, Optimal Workload Allocation Model for Scheduling Divisible Data Grid Applications,Future Generation Computer Systems 26 (2010) 971-978.

[11]. Amin Shokripour , Mohamed Othman et al, New Method for Scheduling Heterogeneous Multi-Installment Systems, Future Generation Computer Systems 28 (2012) 1205–1216.

S.Rekha is currently pursuing final year of B.Tech Information Technology from Sri Venkateswara College of Engineering. She is a Certified Windows Phone Developer and also a member of IEEE. Her area of research includes grid computing and cloud computing.

R.Santhosh Kumar has completed B.Tech Information Technology from Sri Venkateswara College of Engineering. He is a Microsoft Student Partner. He has published five research papers in international journals. His research area includes networking, big data, grid computing and cloud computing.