ISSN:0975-9646



# An Algorithm to Optimize the Traditional Backfill Algorithm Using Priority of Jobs for Task Scheduling Problems in Cloud Computing

Lal ShriVratt Singh[1], Jawed Ahmed[2], Asif Khan[3]

[1]*Research Scholor, M. Tech. (CS), Jamia Hamdard, New Delhi, India*

[2]*Assistant Professor, Deptt. of CS, Jamia Hamdard, New Delhi, India*

[3]*Assistant Professor, Deptt. of CS, GNIOT, Greater Noida, Uttar Pradesh, India*

***Abstract*— The concept of cloud computing came out after a great development over the distributed computing, grid computing, cluster computing and the parallel computing. It is a research field of one of the newest computing technologies. Now a days the IT market is adopting the cloud based technologies over its traditional computing technologies. The problems of job scheduling and resource allocation have been emerging as important and challenging tasks since the starting of this concept. The execution of each job needs the efficient and proper utilization of cloud resources so that the optimum performance of the cloud system may be achieved.**

**This paper proposes an efficient algorithm 'P-Backfill' (say) which is based on the traditional Backfill algorithm using prioritization of jobs for achieving the optimality of scheduling in cloud systems. In this paper the comparisons between the P-Backfill Scheduling (PBS) algorithm and other various scheduling algorithms (SJS, LJF, Round-Robin, FCFS and traditional Backfill) are also shown.**

***Keywords*— Cloud Computing, Multiple Jobs, Job Scheduling (JS), First Come First Serves (FCFS), Traditional Backfill, Longest Job First (LJF), Shortest Job First (SJF), Round Robin (RR), P-Backfill (Priority Based Backfill).**

## I. INTRODUCTION

The concept of cloud computing integrates the concepts of large-scale distributed computing, grid computing, cluster computing and the parallel computing after a great enhancement and development of many years. It came as a revolutionary concept that has totally changed the working style of computing environment and introduced the latest trends in the IT market. Over a communication medium such as internet, the applications which are delivered as the services and the system software/hardware provided by a datacenter which provide those services are referred by cloud computing.

The most challenging task in the cloud systems is the 'Cost of job scheduling' which plays a major role in order to decide the performance of cloud systems. The Job Scheduling (JS) strategies of cloud systems also anticipate profit for the quality of services (QoS) of the cloud system. The schedulers within the cloud systems can be categorized into two categories- local schedulers and global metaschedulars. The local schedulers deal with a single CPU and their residing processes. They also focus on the allocation and execution of those processes. The metaschedulars receive the jobs, submitted by the users. They access the system information for the allocation of processes among the different clusters. The time of job execution can't be predicted in cloud systems so the schedulers must work in dynamic nature [1].

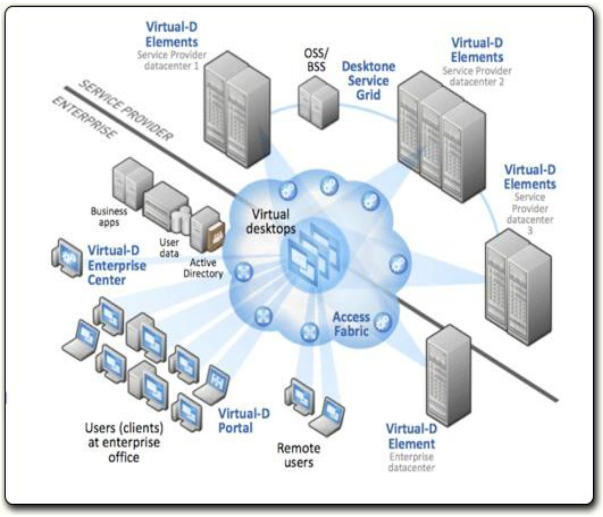


Fig 1: Cloud Data Centers

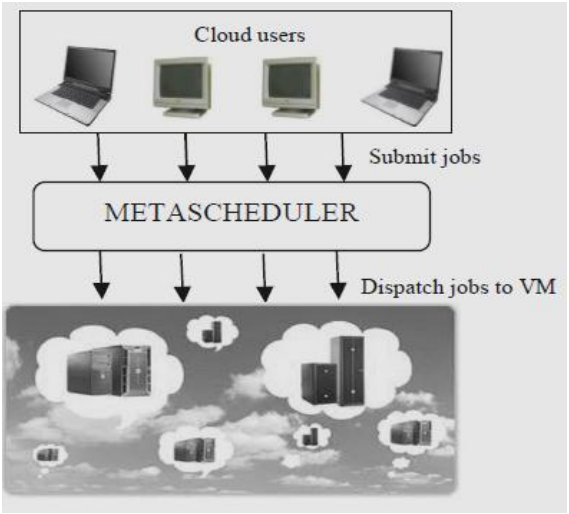


Fig 2: Cloud Metaschedular

## II. Related Works And Their Limitations

Many researchers and IT experts began to start their work on studying the quality of services (QoS) of the job scheduling systems in recent years [2], [3], [4], [5], [6]. It had been expected in 2008 that the cloud computing technology would rise more than 2000 billion in the IT market and its 32% global market share would adopt cloud computing technology till 2013 [7]. A lot of work on scheduling algorithms for their randomness, complexity, dynamic characteristics and various scheduling problems, has been carried out in past years. A systematic comparative analysis of various job scheduling algorithms has been discussed in [8]. The ultimate goal of the cloud service providers is to use the cloud resources in optimized manner using efficient job scheduling strategies. Various requirements of quality of services for the cloud resources have been analyzed and differentiated in [9]. Some basic scheduling algorithms such as FCFS, LJF, SJF, RR and traditional Backfill, which are used in job scheduling in cloud systems are discussed below:-

### *A. FCFS Algorithm*

The FCFS (first come first served) algorithm is widely used scheduling algorithm in cloud computing but it doesn't remove some major performance issues such as high waiting time and less throughput. The utilization of system resources is not optimized by using FCFS scheduling method. If there is a job which has a bigger time of execution, may use the resources of the system for a long time. In this condition the other resources would be idle and would not be used by other jobs. It would make the system delayed. A parallel approach for the waiting job queue in FCFS is proposed in [10].

### *B. LJF and SJF Algorithms*

The other scheduling strategies such as Longest Job First (LJF) and Shortest Job First (SJF) are also not able to fill the resource-gaps. In order to overcome these shortcomings of above mentioned algorithms, the traditional Backfill algorithm was proposed in [11], [12], [13], [14] and [15].

### *C. Traditional Backfill Algorithm*

The traditional Backfill algorithm [11] to [15], works in a different manner, it picks the first coming job and proceeds by taking next job which is having smallest time and so on. It uses the pipelining execution form so that the multiple jobs may be executed at the same time. The main disadvantage of Backfill algorithm is starvation of bigger jobs due to waiting for a long time. Sometimes the execution of these bigger jobs is needed at a high priority but this Backfill algorithm doesn't give any provision to do it. Hence this Backfill algorithm needs to be modified to incorporate the prioritization of arriving jobs using a queue. This work gives a P-Backfill algorithm which will add the priority factor of the arriving jobs with the traditional Backfill algorithm.

***Algorithm***

```
START
STEP 1: Declare the variables.
STEP 2: Input Process name & burst time.
STEP 3: Set cycle=pt [0]
        Start Loop
        For i=1 to n
        Do
        Start if cycle < pt[i]
        Then set cycle = pt[i]
        Stop Loop
        Print Highest processing time
STEP 4: Set pipe [0] =p[0], o[0]=pt[0], status[0]=1;
        Start for i=0 to n
        Do
        Status[i] = 0
        Stop for
STEP 5: Set c=0
        Start for I = 0 to n
        Do
        Set o[i] =0
        Start for j =0 to n
        Do
        Start if status[j] ==0) && (pt[j] <=cycle-o[i]
        Then set pipe[c++] =j, o[i] = o[i] + pt[j], status[j] =1;
        Print Current Pipeline in Use
        End if
        End For
        End For
  STEP 6: Set wt [0] = 0
        Start for i=0 to n
        Do
        Set wt[i] = pt[pipe[i-1]] + wt[i-1];
        Print Average waiting time for process
  STEP 7: Set sum =0
        Start for i=0 to n
        Do
        Sum + =wt[i]
        End Loop
        Set awt = sum/n
        Print Avg Waiting time
        STOP
```

## III. Proposed P-Backfill Scheduling Algorithm

The proposed model for P-Backfill algorithm makes four classes 'A', 'B', 'C' & 'D' of arriving jobs based on their introduced priorities and frequencies of occurrence. Further this model states that the jobs of type 'A' have the highest priority and they have less frequency of occurrence. The jobs of type 'B' have highest priorities (but less than A) and have more frequency of occurrence. Similarly the jobs of type 'C' have the least priority (but more than D) and have less frequency of occurrence and jobs of type 'D' have least priority and have more frequency of occurrence. P-Backfill starts the execution of the jobs according to their priority status. It also uses the pipelining mechanism in order to execute multiple jobs at a time. The execution of the jobs is done from each and every queue based on the pipelining criteria which is prepared by looking the traffic of jobs in a particular channel over a long period and according to it queue responding of scheduler is decided.

***Algorithm***

```
START
STEP 1: Declare The Variables.
STEP 2: Input The process ID, burst time & their type.
STEP 3: Set cycle=pt [0]
STEP 4: Start Loop
        For i=1 to n
        Start if cycle < pt [i]
        Then cycle = pt [i]
        Stop Loop
        Print Job with the highest processing time.
STEP 5: Set c=o=m=0
STEP 6: Start Loop
        For k= „A" to „D"
        Start Loop
        For j=0 to n
        Start if ptype [j] == k
        Start if (o+pt[j]) <=cycle
        Print Current Pipeline in Use
        Set o= o+pt[j], order [m++] =j
        End if
        Start Else
        Set c++, o=0;
        Print Current Pipeline in Use
        o= o+pt[j], order[m++]=j; End
        For
        End For
STEP 7: Set wt [0] =0
        Print Waiting time for process
        Start Loop
        For i=1 to n
        Set wt[i] = pt [order [i-1]] + wt [i-1];
        Print Waiting time for process with Id
        End for Loop
STEP 8: Set sum=0
        Start Loop
        For i=0 to n
        Do
        Sum+ = wt[i]
        End For
        Set awt = (float) sum/n;
        Print Avg. Waiting time
        STOP
```

## IV. COMPARISIONS OF ALGORITHMS

The comparative analysis of various scheduling algorithms is discussed in this section practically by considering their waiting times. The waiting time of various processes is shown below in figure 3. The 'Type' attribute of the given figure is applicable for only two algorithms- Backfill and P-Backfill. In this figure three attributes of tasks 'Process ID', 'Burst Time' and 'Type' i.e. priority level are shown.

| PROCESS NAME / ID | BURST TIME | TYPE |
|---|---|---|
| A | 5 | B |
| B | 6 | C |
| C | 2 | A |
| D | 4 | D |
| E | 9 | C |
| F | 1 | A |
| G | 7 | B |
| H | 3 | D |
| I | 10 | C |
| J | 1 | A |

Fig 3: Process Name/ID, Burst Time & their Type

The 'Process ID' and 'Burst Time' refer to the name of the process and time required for its execution. For example 5 ms are required for the execution of process A. The 'Type' attribute of the processes refers the priority levels of arriving processes. Type 'A' denotes the highest priority level followed by lesser priority levels such as 'B', 'C' and 'D', which is the lowest priority level.

| ALGORITHM | ARRIVAL SEQUENCE OF JOBS | AVERAGE WAITING TIME |
|---|---|---|
| FCFS | A,B,C,D,E,F,G,H,I,J | 22 ms |
| SJF | F,I,C,H,D,A,B,G,E,I | 19 ms |
| ROUND ROBIN | A,B,C,D,E,F,G,H,I,J,A,B,E,G,I,E,I | 25 ms |
| BACKFILL | A,C,F,J,B,D,E,G,H,I | 16.8 ms |
| P-BACKFILL | C,G,J,A,H,B,E,I,D,H | 17.2 ms |

Fig 4: Arrival sequence of jobs & average waiting time

The figure 4 shows the arrival sequence and average waiting time of various jobs according to different algorithms. In [4], [16] and [17], the time slices of Round Robin algorithm are discussed. It can be observed that Round Robin algorithm has the highest waiting time while Backfill has the least. In these both algorithms the tasks of highest priority have not been served earlier but P-Backfill gave the better results in order to fill this gap. However it has slightly more waiting time with respect to traditional Backfill but it servers the jobs of highest priority first that makes the algorithm better than others [3].

The results of FCFS, traditional Backfill and P-Backfill prove that the resources are not being fully utilized in FCFS and also produce high waiting time. Backfill tries to fill this gap but doesn't solve priority jobs' problem. Finally P-

Backfill gives a better solution of above mentioned problems. It reduces the waiting time of jobs and also enables the fully utilization of computing resources which improve the performance and throughput of the overall cloud system. Figure 5 shows the comparative analysis of FCFS, traditional Backfilling and P-Backfilling mechanisms in graphical format for the sample size of 5, 10 & 15 jobs respectively. It can be observed from following graph that the ratio of the job completion is very less in case of FCFS and very high in P-Backfill.

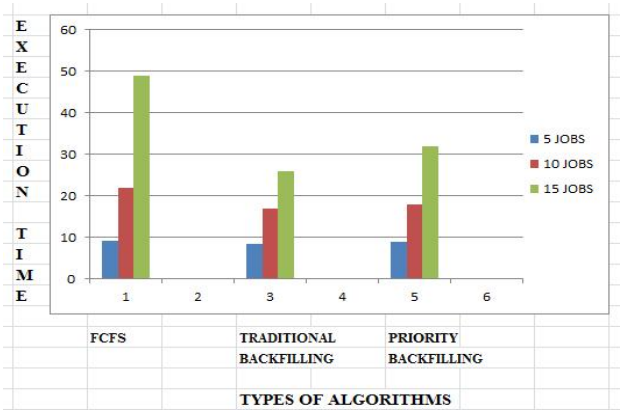


Fig 5: Performance of all Algorithms

## V. ANALYSIS OF P-BACKFILL ALGORITHM

After the comparison of P-Backfill algorithm with other four scheduling algorithms, it has been proved that the P-Backfill scheduling algorithm gives the better results. It has less waiting time of bigger jobs based on their priority level and highest utilization of computing resources in comparison to other scheduling algorithms. The average waiting time of P-Backfill algorithm for 10 jobs is shown in figure 6:-

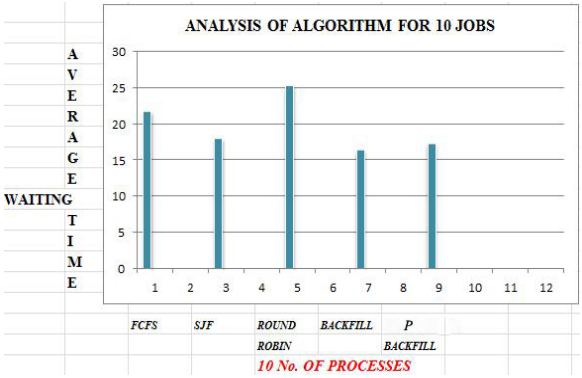


Fig 6: Analysis of P-Backfill algorithm

## VI. CONCLUSION

This paper has presented that how the dynamic metaschedular will deploy the arriving jobs using P-Backfill algorithm to utilize the cloud resources efficiently with less waiting time. This algorithm selects the jobs according to their priority levels whereas the other traditional algorithms such as traditional Backfill, FCFS, SJF, LJF and Round Robin algorithms do not handle the priority jobs so that they are delayed for execution.

## VII. FUTURE SCOPE

The scheduling strategies are always needed for enhancing the system performance and the benefits of the service providers and customers both. Each new policy anticipates the profit into the account of its concerns research field and gives the better solution in comparison to previous ones. As far as the future scope of this research work is concerned, if the priority concepts are added with traditional computing algorithms, they may evolve some new computing approaches which will work better than previous ones.

## REFERENCES

[1] Mayon L.M. Peixato et al, -A metascheduler architecture to provide qos on the cloud computing, 17th International Conference on Telecommunication, 2009.

[2] D. Goldbe, Genetic Algorithms in Search Optimization and Machine Learning & Reading, MA Addison Wesley, International Conference, 1989.

[3] Weidong, H., Y. Yang, and L. Chuang, Qos Performance Analysis for Grid Services Dynamic Scheduling System in Wireless Communications, International Conference on Networking and Mobile Computing, 2007.

[4] Afzal, A, A.S. McGough, and J. Darlington, Capacity planning and scheduling in Grid computing environments. Future Generation Computer Systems, vol 24, pp. 404-414, 2008.

[5] Kiran, M., et al. A prediction module to optimize scheduling in a grid computing environment, International Conference in Computer and Communication Engineering, ICCCE 2008.

[6] Rasooli, A., M. Mirza-Aghatabar, and S. Khorsandi, Introduction of Novel Rule Based Algorithms for Scheduling in Grid Computing Systems, Second Asia International Conference on Modeming & Simulation, 2008.

[7] Yuan-Shun, D., X. Min, and P. Kim-Leng, Availability modelling and Cost Optimization for the Grid Resource Management System, Systems, , Part A, IEEE Transactions on Man and Cybernetics.38(1), 2008.

[8] A Comparative Analysis of Job Scheduling Algorithm, Yarong CHEN, MSIE 2011, IEEE 2011.

[9] HUANG Qi-yi, HUANG Ting-lei, An Optimistic Job Scheduling Strategy based on QoS for Cloud Computing, 2010.

[10] F. Bonomi, "On job assignment for a parallel system of processor sharing queues," IEEE Transactions on Computers,39(7), pp. 858-869, July 1990.

[11] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Selective reservation strategies for backfill job scheduling. Lecture Notes in Computer Science – Job Scheduling Strategies for Parallel Processing, Springer, No. 2537, pages 55-77, 2002.

[12] Bo Li, Dongfeng Zhao, Performance impact of advance reservations from the grid on backfill algorithms, Sixth International Conference on Grid and Cooperative Computing (GCC 2007).

[13] Portable Batch System. URL: http://www.openpbs.org/ last access: March, 2007.

[14] Suresh. A et al, Improving scheduling of backfill algorithms using balanced spiral method for cloud Metascheduler, International Conference on Recent Trends in Information Technology, IEEE, 2011.

[15] Sun Microsystems Inc. Sun Grid Engine. URL:http://gridengine.sunsource.net/, March, 2007.

[16] R.F. Van der Wijngaart. The NAS Parallel Benchmarks 2.4. NAS Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA, 1995.

[17] A. Ganapathi, Y. Chen, A. Fox, R. Katz, D. Patterson, ―Statistics-driven workload modeling for the cloud, University of California, Berkeley, Tech. Rep, Nov 2009.