Parallel Job Scheduling in Cloud with Lookahead and Workload Consolidation

Swathi S
MTech Student
TKM College of Engineering
Kollam, India

Thushara A
Associate Professor
TKM College of Engineering
Kollam, India

Shabi Kamal
Assistant Professor
TKM College of Engineering
Kollam, India

ABSTRACT
The cloud computing paradigm enables consumers to run their applications in remote data centers. Many of these applications may be complex which requires parallel processing capabilities. Parallel job scheduling techniques mainly focus on improving responsiveness and utilization. For a data center that deals with parallel jobs, it is important to devise an optimal schedule which results in maximal utilization of available node capacity. For that, this paper propose a parallel job scheduling technique which uses the key concepts such as workload consolidation through virtualization technologies and backfilling with look ahead mechanism. The proposed method is compared to scheduling using backfilling technique with workload consolidation. The results show that the proposed method with lookahead mechanism has shown better performance.

General Terms
Cloud computing, Parallel Job Scheduling, Backfilling, Resource Consolidation.

Keywords
Workload Consolidation, Backfilling with workload consolidation, Backfill with Lookahead.

1. INTRODUCTION
Parallel job scheduling in distributed environment[1] has been a major research area for several years. With the advancement in cloud computing paradigm, the high performance computing (HPC) applications can be executed in remote data centers. With Infrastructure as a Service (IaaS) cloud service delivery model, computational capacity is delivered over the internet to end users. The parallel computing applications which require simultaneous execution of its processes on different nodes can be executed in a remote data center through the cloud paradigm with this advance in technology. Scheduling of such parallel jobs in a computational cloud data center is the main focus of this paper. Many parallel job scheduling strategies have been proposed for scheduling parallel jobs in distributed environment. Most of them focus on reducing makespan and response time as well as improving the utilization of processors available in a distributed system. The main goal of this work is to propose a better scheduling technique in a computational cloud data center that improves the node utilization and reduces the make span by considering the virtualization technology.

A parallel job can be characterized by the number of nodes it requires for execution and the expected execution time. A process in a parallel job may wait for the data from other process, so communication and synchronization is there between processes of a parallel job. Thus, a parallel job requires simultaneous execution of its processes in a number of nodes. If the requirement of a parallel job cannot be satisfied with the available number of nodes, there is a chance that those available nodes may remain idle. This constraint leads to under utilization of nodes that run parallel jobs.

Many HPC applications require complex parallel computation. Several Massively Parallel Processors (MPPs) have been used for the computation of these HPC applications. All these systems had their own parallel job scheduling policies. With the development of cloud paradigm, these complex HPC applications can be executed in remote data centers. Concepts of parallel job scheduling techniques in MPPs can be effectively used in a cloud data center by applying some changes. In cloud efficiency can be further improved by exploiting the virtualization capabilities of cloud. The proposed model exploits the virtualization capability through workload consolidation and uses the FCFS with backfill technique for queue management. Also, the concept of lookahead is introduced to improve the packing of parallel jobs in the resultant schedule and thereby improving the utilization further.

The remainder of this paper is organized as follows: Section 2 discusses the related work on parallel job scheduling techniques. Details regarding the parallel job scheduling techniques studied and proposed are given in Section 3. Section 4 discusses the simulation details and the results obtained. The conclusion and future work are given in Section 5.

2. RELATED WORK
Many parallel job scheduling approaches were proposed in the literature for massively parallel processors. The basic parallel job scheduling technique in use was First Come First Serve (FCFS)[2]. In this approach, the parallel jobs are considered for execution in the order of their arrival. Whenever the request for the job at head of queue can be satisfied with the freely available number of nodes in the system, then it can be immediately dispatched for execution. If the request of head of queue job cannot be satisfied with the freely available nodes in the system, then those nodes remain idle which results in node fragmentation. Thus the main problem with FCFS is the under utilization of the system resources due to this node fragmentation. In order to overcome these problems with FCFS, techniques such as gang scheduling [3],[4],[5] and backfilling [5], [6] were proposed. Resource sharing among multiple parallel jobs is supported in gang scheduling. It follows an approach where the processes of jobs are shared by dividing the computing capacity of a node into different time slices. [7] Proposes another approach to gang scheduling in which the processes with complement resource needs are placed together to minimize the interference between them.

Backfilling is a technique that allows jobs with lesser node number requirement to use the idle nodes available when the request for job at the head of queue cannot be satisfied. Many variations to traditional backfilling approach were also introduced to optimize the schedule. An integrated approach to parallel job scheduling using backfilling and gang scheduling has been introduced in [8]. An effort to introduce parallel job scheduling concept in cloud has been proposed in [9]. It uses the concept of backfilling along with migration to schedule parallel jobs.

3. PARALLEL JOB SCHEDULING TECHNIQUE

Parallel job requires the simultaneous execution of its processes in several nodes. Many parallel jobs may compete for resources in a data center to get executed. It is the duty of the scheduler to effectively allocate and manage resources available in the data center. Scheduler can be defined as the software component that effectively manages and allocates available resources to the competing parallel jobs which are placed in the queue according to the scheduling policy used in the system.

The proposed parallel job scheduling mechanism uses the FCFS with backfill policy for queue management. Inorder to improve utilization workload consolidation approach is introduced by exploiting the virtualization capabilities. Packing of jobs in the schedule is improved by applying lookahead strategy. Basic assumptions in the proposed method are that parallel jobs considered are of rigid jobs. i.e, once the requirement is specified it requires that number of nodes to complete their execution. Also, the model follows run-to-completion strategy. i.e, once the job is dispatched for execution, no other job is allowed to preempt it from execution.

3.1 Workload Consolidation Approach

Workload consolidation is applied in order to improve the utilization. As a parallel job with many processes is executed on many processors, the computing capacity of those processors is not fully utilized. In order to improve the utilization of a processor that runs parts of a parallel job, workload consolidation technique is introduced. In this method, the computing capacity of a processor is partitioned into two virtual machine tiers as Foreground Virtual Machine (FVM) tier and Background Virtual Machine (BVM) tier. The advantage of this approach is that each processor is capable of running job parts of two different parallel jobs simultaneously and hence improves the utilization. This approach allows the scheduler to dispatch more number of parallel jobs at a time and hence it also contributes to reduction of make span.

3.2 Backfilling with Workload Consolidation

When workload consolidation is there, we can run multiple jobs with the computing capacity of a single machine. In such a case, when the queue order follows a First come First Serve (FCFS) approach alone, then it will result in severe node fragmentation due to parallelization. When the number of nodes required for job at head of queue cannot be satisfied with the currently idle nodes, then according to FCFS, no jobs will be dispatched and those nodes remain idle until enough idle nodes become available to process the job at head of queue. This results in severe under utilization. In order to overcome this problem, the backfilling technique is introduced in which the jobs with less number of nodes required for execution and which arrived later than the job at head of queue are allowed to execute. Thus the under utilization can be reduced when there are backfill jobs available which can satisfy the current idle node capacity.

When backfilling is allowed in the case of workload consolidation concept, the computing capacity of machine is divided into two tiers, FVM and BVM, so we are able to run job parts of two different jobs on the same machine. In this case scheduling is done in such a way that:

When enough idle processors are available to process the job at the head of the queue, then it is immediately dispatched. While dispatching the capacity to process the job is also considered, when a job whose job parts have a lesser capacity requirement (size of job is less), then it is dispatched for execution in the BVM else if enough BVMs are not available, then it is allowed to execute in FVMs. So here a best fit policy is also adopted. This is required as there can be more than one job request that arrive at the same time. So FCFS policy alone is not enough. In such a case, in order to maximize utilization, best fit strategy is applied. Same jobs whose arrival time is same will be dispatched in a manner that will result in better utilization of available capacity. So, jobs which arrive at the same time are processed in such a way that it will improve overall system utilization. When jobs which arrived at the same time can't be dispatched because of its higher node number requirement, then backfilling will be initiated. Here, a job with later arrival time and placed towards the end of queue will be allowed to execute. Because its node number requirement is less.

3.3 Backfilling with Workload Consolidation and Lookahead Approach

In this case, a lookahead mechanism is introduced in the above method. It is to improve the utilization, reduce response time and hence to result in better makespan by finding the best victim node for backfilling.

Here the mechanism is that while initiating backfilling, in order to avoid starvation of higher priority jobs in the queue, expected allocation time of priority jobs will be calculated. Then backfilling is done in such a way that those backfill jobs whose execution time is less than the expected allocation time of higher priority jobs will be allowed to dispatch. Then the advantage is that there will be no starvation for higher priority jobs also the node utilization will be maximum as the backfill job selected can complete its execution when the higher priority job will get processors according to its requirement.

Here, a job which can be executed with lesser machine capacity will be dispatched to BVMs which are slow processors compared to FVMs, FVMs whose machine capacity is more can be utilized for long jobs which requires more processing power. In such a way we can optimize the schedule generated.

4. SIMULATION RESULTS AND DISCUSSION

The proposed system is simulated in cloudsim [10]. Both the algorithms proposed in the previous session are implemented and a result analysis is performed. The effect of schedule generated as per the proposed algorithm with lookahead mechanism is compared with the schedule generated by the backfilling with workload consolidation technique. How both schedules vary in utilization, makespan and response time are studied by calculating each of these factors for the schedules generated by both the algorithms. The results are studied and how it gets affected by the system load is plotted graphically.

Table 1 and Figure 1 show the effect of makespan in both the schedules. Makespan of a schedule is the length of schedule and it can be obtained by calculating the completion time of all the jobs in the schedule. The makespan metric analysis indicates that the schedule generated by the proposed algorithm resulted in better makespan.

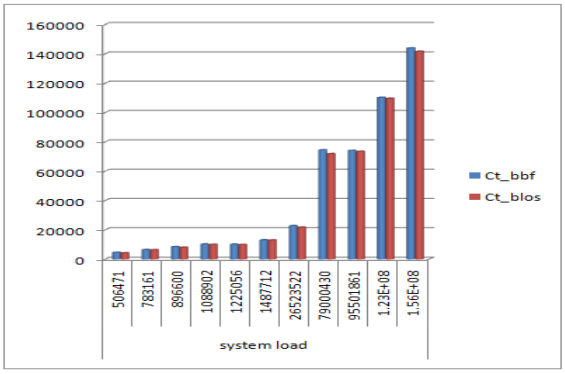


Figure 1. System Load vs Make Span

Table 1. System load vs Makespan

SYSTEM LOAD	MAKESPAN	
	Ct_bbf	***Ct_blos***
506471	4540.957	4230.12
783161	6490.147	6379.438
896600	8448.161	8059.593
1088902	10232.42	9988.987
1225056	10149.16	9836.741
1487712	13107.52	13014.59
26523522	22792.39	21782.03
31075787	27308.01	27558.97
31769829	25948.43	25615.24
48406823	45683.68	43698.42
50940030	37208.09	36724.73
79000430	74445.58	71889.57
95501861	74025.65	73413.56
98248549	91857.23	91660.42
1.23E+08	110254.2	109587
1.56E+08	143939	141646.6

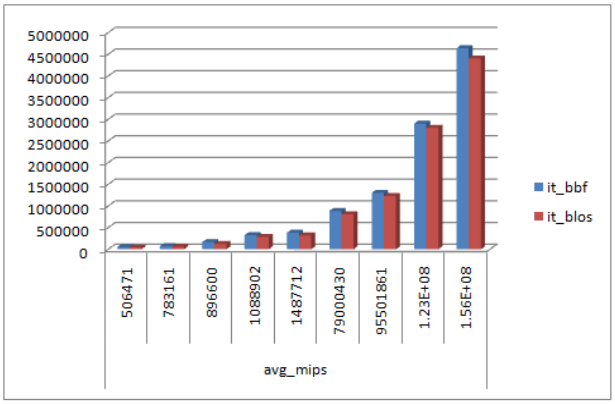
Table 2 and Figure 2 show the effect of utilization of nodes in the data center by both the schedules. Utilization of nodes in the system can be calculated from the idle time. Idle time of a node is the time during which the nodes in the system remain idle. A short idle time indicates a better utilization of available system capacity. The proposed scheduling policy results in a better utilization compared to that of the basic backfilling with workload consolidation approach.

Table 2. System Load vs Utilization

SYSTEM LOAD	UTILIZATION	
	it_bbf	***it_blos***
506471	50584	42147
783161	71386	58998
896600	164078	122958
1088902	325056	285408
1225056	174226	142049
1487712	383451	322415
9971343	137998	112040
31075787	71574	45094
31769829	127086	97263
48406823	255390	222923
50940030	197596	154486
79000430	886909	808897
95501861	1302376	1229721
98248549	2903336	2900247
1.23E+08	2901491	2803748
1.56E+08	4641509	4401180

Figure 2. System Load vs Utilization

Table 3 and Figure 3 depict the effect on response time by both schedules. Response time of a job is the time duration from which the job is submitted to the system until it gets completed. Response time of a schedule is calculated by



summing up the response times of all the jobs in the system. By analysing the response time metrics on both schedules, it is found that the proposed scheduling policy has shown a short response time compared to that of the basic backfilling scheduling policy.

Table 3. System Load vs Response Time

SYSTEM LOAD	RESPONSE TIME	
	Ct_bbf	***Ct_blos***
506471	4109358	3921719
783161	9875227	9163678
896600	22531528	20715116
1088902	51943989	47648963
1487712	88536415	81731363
79000430	1.1E+08	1.07E+08
95501861	1.7E+08	1.68E+08
98248549	4.52E+08	4.48E+08
1.23E+08	4.79E+08	4.8E+08
1.56E+08	1.09E+09	1.08E+09

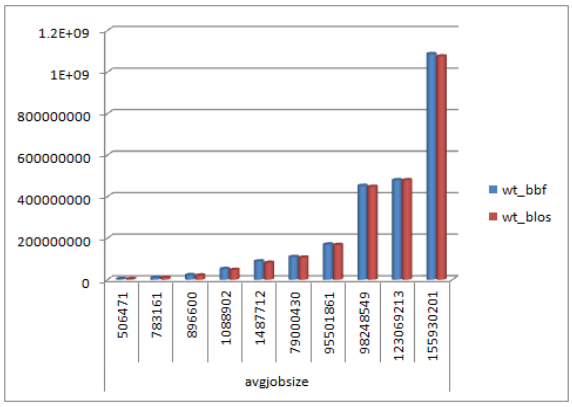


Figure 3. System Load vs Response time

5. CONCLUSION AND FUTURE WORK

Parallel job scheduling technique proposed in the work has been implemented in cloudsim and the result obtained has been analysed. From the result obtained, backfilling with workload consolidation and lookahead strategy has shown a better result in terms of utilization, response time and make span of schedule obtained when compared with the corresponding metrics of the schedule generated by backfilling with workload consolidation strategy.

The proposed work focused on scheduling of parallel jobs which follow a non preemptive strategy. Migration of backfill jobs can be performed in order to give priority to jobs that arrived earlier. So on the proposed algorithms, the effect of applying migration to parallel jobs on the schedule can be studied.

6. REFERENCES

[1] D. Feitelson, L.Rudolph, U. Schwiegelshohn, K.Sevcik, and P.Wong, "Theory and Practice in Parallel Job Scheduling," Proc.Workshop Job Scheduling Strategies for Parallel Processing, pp. 1-34,1997

[2] U. Schwiegelshohn and R. Yahyapour, "Analysis of First-ComeFirst-Serve Parallel Job Scheduling,"Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms,pp. 629-638, 1998.

[3] D. Feitelson and M. Jettee, "Improved Utilization and Responsiveness with Gang Scheduling," Proc. Workshop Job Scheduling Strategies for Parallel Processing,pp. 238-261, 1997.

[4] J. K. Ousterhout, "Scheduling techniques for concurrent systems," Proceedings of Third International Conference on Distributed Computing Systems, May 1982, pp.20-30.

[5] A. Mu'alem and D. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM sp2 with Backfilling,"IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 6, pp. 529-543, June 2001.

[6] Edi Shmueli, Dror G. Feitelson, "Backfilling with lookahead to optimize the packing of parallel jobs," Elsevier ScienceDirect Journal of Parallel and Distributed Computing. July 2005

[7] Y. Wiseman and D. Feitelson, "Paired Gang Scheduling,"IEEE Trans. Parallel and Distributed Systems,vol. 14, no. 6, pp. 581-592, June 2003.

[8] Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam, "An Integrated Approach to Parallel Scheduling Using Gang-Scheduling, Backfilling, and Migration," IEEE Trans. Parallel and Distributed Systems,vol. 14, no. 3, pp. 236-247, Mar. 2003.

[9] Marco Xiaocheng Liu, Chen Wang, Bing Bing Zhou, Junliang Chen, Ting Yang, and Albert Y. Zomaya, Fellow, "Priority-Based Consolidation of Parallel Workloads in the Cloud", IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 9, September 2013.

[10] Rajkumar Buyya, Rajiv Ranjan and Rodrigo N. Calheiros, "Modeling and Simulation of Scalable Cloud ComputingEnvironments and the CloudSim Toolkit: Challenges and Opportunities," 2009.