A Proposal for Cache based Methodology and Parallel Precedence Consolidation for Similar Workloads in Cloud

Banoth Sreenivas[1], B.Narasimha[2], Janapati Venkata Krishna [3]

[1]*pursuing M.Tech (CSE), Holy Mary Institute of Technology and Science, Keesara, Affiliated to JNTU- Hyderabad, A.P, India*

[23]*working as Associate Professor (CSE Department) in Holy Mary Institute of Technology and Science, Keesara, Affiliated to JNTU- Hyderabad, A.P, India*

Abstract— **The complex applications are attracted by cloud computing is increased in day to day manner to run in remote data centers. Many applications needs parallel processing capabilities. The nature of parallel application is decrease the utilization of CPU resources as parallelism grows, because of the communication and synchronization between parallel processes. It challenging task but important for the data centers to reach a certain level of utilization of its nodes at the time of maintaining the level of responsiveness of parallel jobs. The existing parallel scheduling mechanisms take irresponsibleness as the top important and need nontrivial effort to make them work for the data centers in the cloud era. In this we introduced a parallel priority based technique to consolidate parallel workload in the cloud. We influence virtualization technology to partition the computing capacity of every node into two tiers, the fore virtual machine (VM) tier (with high CPU priority) and the background VM tier (with low CPU priority). They provided scheduling algorithms for parallel jobs to make effective utilization of the two tier VMs to improve the responsiveness of these jobs. Our wide range experiments display that our parallel scheduling algorithm expressively outperforms commonly used algorithms such as extensible Argonne scheduling system in a data center setting. This technique is practically and experimentally effective for consolidating parallel workload in data centers**.

Keywords— **cloud computing, consolidation, scheduling technique, parallel priority.**

1. INTRODUCTION

The cloud computing model promises a cost-effective resolution for running business type of applications with the use of virtualization technologies, highly accessible distributed computing, and data management methods as well as a pay as-you-go pricing models. In current years, it also provides high performance computing capacity for applications to resolve difficult problems. The improvement of resource utilization is important for reaching cost effectiveness. The low utilization is an issue in data centers. The servers available in typical data center are worked at 10 to 50 percent of their optimum utilization level. 10 to 20 percent of utilization is common in data centers. In a data center, or a subset of servers that mainly handles applications with highly-performance calculating needs and most of the time runs parallel jobs, the problem can be significant.

There are two things that may decrease the use of nodes that run parallel jobs:

1. A parallel job technique always requires a certain number of nodes to run the application. A set of nodes is likely to be disjointed by parallel jobs with different number of nodes requirement. If the number of available nodes cannot fulfill the requirement of an upcoming job, these nodes are may remain idle.
2. Typical parallel programming simulations, such as BSP often include calculating, communication, and synchronization period. A process in a parallel job may often wait for the data from some other processes. During waiting, the nodes of utilization are low.

The most initial but powerful batch scheduling algorithm for parallel jobs is first come first serve (FCFS). Each and every job specifies the number of nodes needed and the scheduler processes the jobs according to the order of their arrival. When there is a enough number of nodes to process the job at the head of the queue, the scheduler migrate the job to run on these nodes then, it waits till jobs now running finish and release enough nodes for the job. FCFS may cause node split and methods such as backfilling and Gang scheduling were suggested to increase it. However, they do not goal on the utilization degradation caused by parallelization itself.

In this paper we concentrate on increasing the utilization of data centers those run parallel jobs, particularly we mean to make use of the remaining computing capacity of data center nodes those run parallel processes using of low resource utilization to increase the performance of parallel job scheduling. The similar jobs we deal with have the following characteristics:

1. The time of job execution is unknown
2. With the supporting of check point saving and restoring the state job is very cheap.

3. The usage of CPU process of the job can be estimated during design phase.

Parallel discrete event simulation pertained to this category of jobs, and there are struggles, to run this type of jobs in the cloud. We proposed in this paper a priority-based consolidation method for scheduling these types of parallel jobs with the following aims: 1) it will improves the utilization of servers allocated to these jobs; 2) reserve the FCFS order of jobs when available resources fulfill the needs of these jobs. Our technique gives a methodical way to consolidate parallel workload. The elementary idea is to put a background virtual machine (VM) in every node so that the background virtual machine can use calculating the resources when the foreground virtual machine cannot fully use them. We will make the following contributions:

1. We will conduct extensive experimentations for workload consolidation. We establish that using virtualization technologies with suitable assignment of significances to virtual machines, we can effectively allow jobs collocated in a physical node to efficiently use the computing capacity without significant impact to the performance of the high-priority job.
2. Constructed on the above surveillance, we give a parallel priority based workload consolidation method with the support of primary VM collocation mechanism. We panel the calculating capacity of each physical node into two layers, namely foreground VM (with high CPU priority) and background VM (with low CPU priority) by pinning two VMs to the node. They can concurrently process different jobs. The contextual job can therefore use the underutilized calculating capability whenever the foreground job cannot fully use it. The proposed method supports backfilling in such a two-layer setting.

Our appraisal results show that our consolidation based algorithm (Aggressive Migration and Consolidation supported Back Filling (AMCBF)) expressively outperforms FCFS and Extensible Argonne Scheduling Ystem (EASY) (accurate job execution time is available for EASY in our experiment) on famous traces. In adding, our method outperforms EASY even when it only knows the information of the jobs' node number needed. Finally, our algorithm can accomplish two commonly conflicting goals in parallel job scheduling: improving the system use and the job reaction.

2. RELATED WORK

The scheduling appliances for parallel jobs in clusters have been several efforts. FCFS is the elementary but commonly used batch scheduling algorithm. Backfilling, which was developed as the EASY for IBM SP1, it is a mechanism that allows short or small jobs to use indolent nodes while the job at the head of the queue do not have much number of nodes to run. Backfilling can increase node consumption, but it requires each job to specify its maximum execution time so that only jobs that will not delay the start of the job at the head of the queue are backfilled. Additionally, a preempted job is oalways given a reservation for a future time to run. Different techniques of handover reservations differentiate several variances of backfilling techniques. Backfilling techniques had shown the low-utilization problem reason by different node number requirements of parallel jobs. However, due to parallel jobs the backfilling does not deal with low resource utilization themselves.

Gang scheduling permits resource sharing from multiple parallel jobs. The calculating capacity of a node is bifurcation into time slices for the purpose of sharing among the processes of jobs. The gang scheduling algorithm will manages for making all the processes of a job progress together so that even one process will not be in sleeping state when the remaining process needs to communicate with it. The allocating time slices of different nodes to parallel processes are synchronized, which OS needs support. Some of gang scheduling algorithms, such as paired gang scheduling inspect how to place processes with accompaniment resource needs composed to minimize their interference, e.g., when a process do some I/O works and leaves CPU without any work, the paired gang scheduling algorithm can choose a procedure to use the idle CPU resources. A same approach is used in cloud resource consolidation with correlation investigation of resource use surrounded by VMs. Processes of parallel jobs share the calculation capacity of a node equally in common scheduling algorithms. This method can improve the optimum usage to a convinced degree, but is likely to expanse the execution time of separate jobs. There is attempt to incorporate backfilling and gang scheduling [22], but it only results in a comparable performance to that of the simple backfilling algorithm [23].

Both backfilling and gang scheduling intend to improve utilization caused by node destruction. They will not aim on the utilization degradation caused by parallelization itself.

3 WORKLOAD CONSOLIDATION METHOD

To a parallel application with the dependency from its parallel processes, reaching optimum usage on the nodes on which these processes run is regularly difficult. For a cloud service provider to runs this kind of applications, how to report this issue is important for its effectiveness in the market. We will do two workload consolidation experimentations in endeavoring to improve node utilization and inspect the effect to the execution time of parallel jobs.

In the first research, we calculate two VMs in each physical node and give these VMs having the same priority, i.e., each VM is assigned a weight of 256. In the second test, the calculated two VMs having different priorities, in which one is allocated a weight of 10,000 and the other is allocated a weight of 1. We call the foreground VM is high-priority and the background VM one is low-priority. In this setting, the background VM only runs when the foreground VM is unemployed.

Throughout the experimentations, we had been made the following observations:

1. Priority-based VM apposition experiences trivial performance effect on jobs running in high-priority VMs. The medium performance loss of jobs running in the front layer is between 0.0 and 3.7 percent matching to those running in the nodes entirely. We simply model the loss as a constant spreading.

2. When a job is running by foreground VM with CPU usage more than 96 percent, apposition a VM to run in background does not get any benefit either the foreground or the background job because of that context switching incurs overhead and the background VM having very small chances to get physical resource to run.

3. Whenever a foreground Virtual Machine will runs a job with low CPU utilization, to run the job running in the collocated background VM can get significant share of physical resources. The utilization of the idle CPU cycles is between 80 and 100 percent for a single-process background job and simply follows uniform distribution; for a multi-processes background job, the value should be between 19.8 and 76.6 percent and it can be modeled by a normal dissemination with _ ¼ 0:428 and _ ¼ 0:144. By these observations, we would discuss our scheduling algorithms in the following section.

4 SCHEDULING ALGORITHMS

In this session we are going to discuss about our scheduling algorithms that parallel priority based workload consolidation. Prior to discuss our consolidation strategies based algorithms we will discuss about the basic scheduling algorithms.

4.1 Basic Algorithms

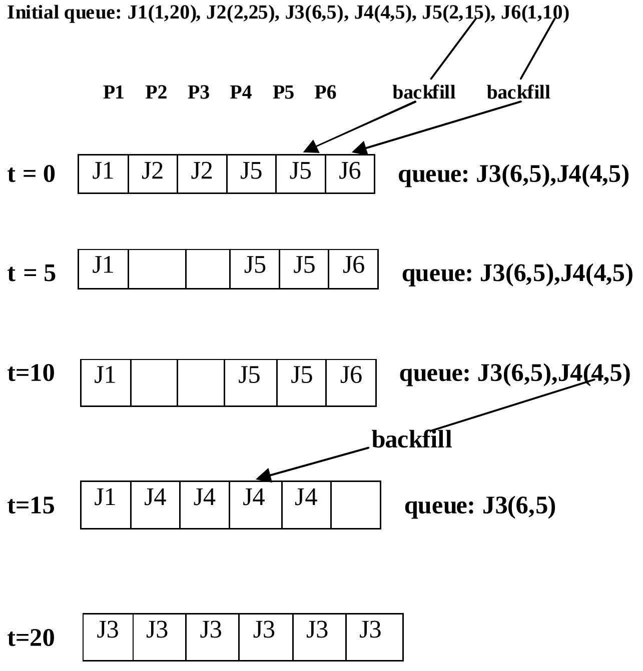
Our prior scheduling algorithm, Conservative Migration supported Back Filling (CMBF) it is backfill based. This algorithm imagining that the position of job could be saved restored; so, this scheduler is capable to stop a job and then it will resume on the remaining nodes in later time.

4.2 Scheduling with workload consolidation

In the above we discussed the prior algorithm only considers the mapping one parallel process to one node. As we discussed in section 1 and 3 that node utilization can be very for these nodes because of that high efficiency in parallel computing is always difficult to reach. In this session, we elaborate the basic algorithms to be node usage awareness in

Algorithm 1:

Initial queue: J1(1,20), J2(2,25), J3(6,5), J4(4,5), J5(2,15), J6(1,10)



improving the node utilization in the cloud.

Algorithm 2: CMCBF – On the departure of foreground job

```
input : Q: the queue for incoming jobs;
        M: a map between jobs and nodes;
Output : M': the updated allocation map;
begin
    j ← get the first job from Q;
    while j $\neq$ null do
        $N_j$ ← the number of nodes required by j;
        $N_{idle}$ ← the number of idle nodes;
        if $N_j \leq N_{idle}$ then
            remove j from Q and dispatch it any $N_j$ idle nodes;
            update M accordingly;
            if j is not the head of Q then
                insert j into $Q_{backfull}$ ;
        else
            $N_{backfill}$ ← the number of nodes running Jobs arriving later than j;
            if $N_j \leq (N_{backfill} + N_{idle})$ then
                suspend jobs in $Q_{backfill}$ that arrive later than j and move them back to Q according to descending order of their arrival time until the number of idle node is greater than $N_j$;
                remove j from Q and dispatch it to $N_j$ idle nodes;
                update M;
        j ← get the next job from Q;
```

Based on our notice in Section 3, we will bifurcate the calculating capacity of a physical node into two layers, named as foreground and background. We imagine that a physical node can be run at most two VMs one in the foreground and another one in the background. While running in foreground the Virtual Machine is assigned a high CPU priority and while running in background the VM is assigned as a low CPU priority. In the following paragraph, we will give a scheduling algorithm to handle two types of VM resources.

Conservative Migration and Consolidation supported Backfilling (CMCBF), as display in the three parts as Algorithms is based whatever policy is used in CMBF. It confirm that the is dispatched to foreground for running whenever the most number of foreground VMs are idle or having more nodes than its capacity arriving later than to satisfy the node requirement. Meanwhile it will run the background VM by allowing jobs simultaneously by this foreground VM will improve its node utilization. Compare with CMBF, CMCBF also be deals with whatever the work will not effected in the background job. CMCBF will dispatches a job to run in background VMs whenever the corresponding foreground VM is busy means lower utilization of threshold. The foreground VM utilization also get the details from its profile of job or from the runtime displaying data.

We are using the example to demonstrate the proposed algorithm. Here we take 5 nodes (P1-P2) for a job queue it has job j1 to j10 at the time consideration. Each node having two-tier computing capacity specified as f_g and b_g for simplicity of describing the example.

At time of 0, job J1, J2, and J3 are allocated for five nodes and run in foreground VMs according to Algorithm. As J1 has a single-process job, therefore P1 cannot provide allocation to another VM running in background. However, J4 and J5 can run in background VMs at node P2-P5. How to collocate a background VM which a foreground VM is determined through a simple process

5. CONCLUSION

For computing the number of complex applications the in the cloud for parallel computing the computing power should be efficiency to manage the compute resource utilization along with increase of parallelism. For scheduling parallel jobs for both efficient resource utilization and job responsiveness is very important.

Workload consolidation is supported by virtualization technology it's commonly used for improving utilization in the data centers to make the resource utilization is very efficiency. Our method divided the node's computing capacity into foreground VM (High priority) and background VM (Low priority) tier. The performance of job running inside foreground VMs is almost near to the jobs which running inside dedicated nodes. Then idle CPU cycle also can be well used by the jobs running in the background VMs. The proposed algorithm is the combination of Backfilling and Migration to make effective use of two types of VMs.

REFERENCES

[1] D. Feitelson, A Survey of Scheduling in Multiprogrammed Parallel Systems. IBM TJ Watson Research Center, 1994.

[2] J. Hamilton, "Cloud Computing Economies of Scale," Proc. AWS Genomics Cloud Computing Workshop, http://www.mvdirona. com/jrh/Talk sandpapers/JamesHamilton_GenomicsCloud 20100608.pdf, 2010.

[3] L. Barroso and U. Holzle, "The Case for Energy-Proportional Computing," Computer, vol. 40, no. 12, pp. 33-37, Dec. 2007.

[4] "High Performance Computing (HPC) on AWS,"Amazon Inc., http://aws.amazon.com/hpc-applications/, 2011.

[5] J. Jones and B. Nitzberg, "Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization," Proc. Workshop Job Scheduling Strategies for Parallel Processing, pp. 1-16, 1999.

[6] D. Feitelson and B. Nitzberg, "Job Characteristics of a Production Parallel Scientific Workload on the Nasa Ames ipsc /860," Proc. Workshop Job Scheduling Strategies for Parallel Processing, pp. 337- 360, 1995.

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

[8] L.G. Valiant, "A Bridging Model for Parallel Computation," Comm. ACM, vol. 33, no. 8, pp. 103-111, 1990.

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

[10] D. Lifka, "The Anl/Ibm SP Scheduling System," Proc. Workshop Job Scheduling Strategies for Parallel Processing, pp. 295-303, 1995.

[11] Y. Lin, "Parallelism Analyzers for Parallel Discrete Event Simulation," ACM Trans. Modeling and Computer Simulation, vol. 2, no. 3, pp. 239-264, 1992.

[12] R. Fujimoto, "Parallel and Distributed Simulation," Proc. 31st Conf. Winter Simulation: Simulation—A Bridge to the Future, vol. 1, pp. 122-131, 1999.

[13] Z. Juhasz, S. Turner, K. Kuntner, and M. Gerzson, "A Performance Analyser and Prediction Tool for Parallel Discrete Event Simulation," J. Simulation, vol. 4, no. 1, pp. 7-22, 2003.

BANOTH SREENIVAS, pursuing M.Tech (CSE) from Holy Mary Institute of Technology and Science, Keesara, Ranga Reddy Dist., Affiliated to JNTU-HYDERABAD.

B. NARSIMHA, Associate Professor (CSE Department), Holy Mary Institute Of Technology and Science, Keesara, Ranga Reddy Dist., Affiliated to JNTU-HYDERABAD.

Janapati Venkata Krishna, Associate Professor & H O D (CSE Department), Holy Mary Institute Of Technology and Science, Keesara, Ranga Reddy Dist., Affiliated to JNTU-HYDERABAD.