IJCAT International Journal of Computing and Technology
ISSN : 2348 - 6090
www.IJCAT.org

Parallel Workload Scheduling in Cloud: A Survey

[1] Shahabanath K K, [2] Sreekesh Namboodiri T

[1] M.Tech student in Computer Science and Engineering, Calicut University,
Kerala, India

[2] Assistant Professor in Computer Science and Engineering, Calicut University,
Kerala, India

Abstract - The cloud computing paradigm is attracting an increased number of complex applications to run in remote data centers. Scheduling is an important issue in the cloud. The main goal of scheduling is distribute the load among processors and maximizing their utilization by minimizing the total task execution time and also maintaining the level of responsiveness of parallel jobs. Existing parallel scheduling mechanisms have some drawbacks such as context switching rate, large waiting times and large response time. The paper presents a comparative study on various scheduling algorithms used in the cloud. This papaer discusses three techniques backfilling, gang scheduling and migration and also propose a two tier architecture for workload consolidation.

Keywords **- Cloud computing, parallel job scheduling, FCFS, Gang scheduling, Backfilling, Migration, Aggressive backfilling, Conservative Backfilling.**

1. Introduction

Cloud computing is an on demand service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. Scheduling jobs is an important issue in the cloud. There are various scheduling algorithm exist in cloud computing environment. The main goal of scheduling algorithm is distribute the load among processors and maximizing their utilization while minimizing the total task execution time. The job scheduler is responsible for assigning preferred resources to a particular job so that the overall computing resources are utilized effectively. The application also has to make sure each job is given adequate amount of resources, or its fair share.

There two types of scheduling :

- Application scheduling
- Job scheduling

The process of scheduling parallel tasks determines the order of task execution and the processor to which each task is assigned. Typically, an optimal schedule is achieved by minimizing the completion time of the last task. Two types of scheduling strategies are space sharing and time sharing. Time sharing techniques virtualizes the physical machine by slicing the time axis into multiple virtual machines. Space sharing techniques runs the jobs side by side on different nodes of the machine at the same time.

2. Literature Survey

2.1 First-Come-First-Serve

The basic batch scheduling algorithm is First-Come-First-Serve (FCFS) [5]. Under this algorithm, jobs are considered in order of arrival. If there are enough processors are available to run a job, the processors are allocated and the job is started. Otherwise, the first job in the queue must wait for some currently running job to terminate. This may lead to a waste of processing power as processors sit idle waiting for enough of them to accumulate.

2.2 Backfilling

Quinn Snell et al. [5] proposed a backfilling scheme. Backfilling is an space sharing optimization that tries to balance between the goals of utilization and maintaining FCFS order. It allows small jobs to move ahead and run on processors that would otherwise remain idle. This is done to avoid situations in which the FCFS order is completely violated and some jobs are never run.

There are two types of backfilling algorithms:

- Conservative Backfilling
- Aggressive Backfilling

1) Conservative Backfilling: Ahuva et al. [2] proposed a conservative backfilling approach. In this scheme, jobs are scheduled according to the order of arrival time when there is enough number of processors.If not, another job with later arrival time and smaller jobs are scheduled to run. It provides reservation to all jobs and limits the slowdown.

2) EASY Backfilling: Ahuva et al. [2] also proposed an aggressive approach, provides a reservation to only the job at the head of the job queue and only allow job at the head of the queue can be pre-empt other jobs. It does not have a guaranteed response time of the user job at the time of job submission.

2.3 Gang Scheduling

Jonathan Weinberg [7] proposed a main alternative to batch scheduling is gang scheduling, that schedules related threads or processes to run simultaneously on different processors. It is a time sharing optimization technique. This scheduling is used if two or more threads or processes are communicate with each other. The problems with gang scheduling is that the requirement that all a job's processes always run together causes too much fragmentation and context switching overhead.

Gang scheduling is based on a data structure called ousterhout matrix. In this matrix each row represent a time slice, and each column represent a processor. The threads or processes of ajob are packed into a row of the matrix.

	P0	P1	P2	P3	P4	P5	P6	P7
Time-slice0	J1	J1	J1	J1	J1	J1	J1	J1
Time-slice1	J2	J2	J2	J2	J2	J2	J2	J2
Time-slice2	J3	J3	J3	J3	J4	J4	J5	J5
Time-slice3	J6	J6	J6	J6	J4	J4	J5	J5

Fig 2.1 Ousterhout Matrix

2.4 Backfilling Gang Scheduling

Moreiraz et al. [6] proposed a Backfilling gang-scheduling (BGS) method. It is an optimization technique which combines gang scheduling and backfilling scheduling. This scheduling can be done by treating each of the virtual machines created by gang-scheduling as a target for backfilling. Which produce better results than individual approaches gang scheduling or backfilling.

2.5 Migration Gang Scheduling

Moreiraz et al. [6] proposed a Migration Gang Scheduling method. The process of migration embodies moving a job to any row in which there are enough free processors to execute that job. There are two options for migrate a job from a source row to a target row.

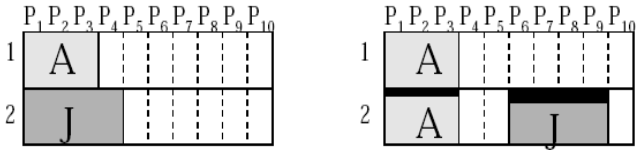

Fig 2.2 Migration option 1

In the above figure, job A resides in the first row and job J in the second row occupy the same columns as job A in first row. Job J migrate to other columns in the same row and job A is replicated to second row.

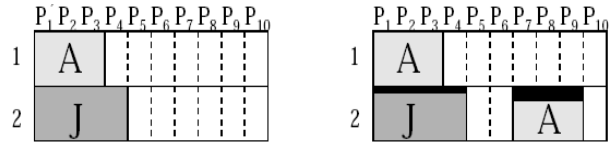

Fig 2.3 Migration option 2

In migration option 2, job A in the first row can be directly migrate to second row.

2.6 Migration Backfilling Gang Scheduling

Moreiraz et al. [6] proposed a method that the migration embodies moving a job to any row in which there are enough free processors to execute that job. If we cannot replicate a job in a different row because its set of processors are busy with another job, attempt to move the blocking job to a different set of processors. A job can appear in multiple rows of the matrix, but it must occupy the same set of processors in all the rows. This rule prevents the ping-pong of jobs.

2.7 Job kill based EASY Backfilling (KEASY)

Xiaogang Qiu et al. [1] proposed a scheduling scheme Job kill based EASY backfilling (KEASY). It is capable of dispatching a job to run in background VMs while it is not qualified for backfilling according to EASY. There is a chance that the corresponding foreground VMs of those background VMs are idle during the jobs lifetime, which leads to performance improvement.

2.8 Reservation based EASY Backfilling (REASY)

Xiaogang Qiu et al. [1] proposed a Reservation based EASY backfilling (REASY) scheme. In this scheme job kill is not allowed in the scheduling; once a job is deployed onto background VMs of a set of pro- cessors, its run is pinned onto this set of processors. Only all the foreground VMs of this set of processors are available can this job run in the foreground. For the reservation making, if a reservation is being made for a job running in the background tier, the shadow time is the last termination time of the jobs running in its foreground VMs; the extra foreground VMs are the ones now idle and no process of the job is running in their background VMs.

2.9 Conservative Migration Supported Backfilling

Xiaocheng Liu et al. [4] proposed a Conservative Migration Supported Backfilling (CMBF) , which is same as Conservative backfilling. Only the difference is, the scheduler is able to suspend a job and resume it on other nodes in a later time. This algorithm avoid starving a pre-empted job. When the number of jobs in the queue is large, the cost can be high because CMBF requires tracking backfilling jobs for each job in the queue.

2.10 Aggressive Migration Supported Backfilling

Xiaocheng Liu et al. [4] proposed an An alternative to CMBF is Aggressive Migration Supported Backfilling (AMBF). It only tracks backfilling jobs for the job at the head of the queue and allows the head-of queue job to pre-empt other jobs. The rest of jobs in the queue are not allowed to pre- empt jobs.

2.11 Priority-based Consolidation Method

In priority-based method to consolidate parallel workloads in the cloud, dividing computing capacity into two tiers : foreground tier and background tier. The VM running in foreground is assigned a high CPU priority and the VM running in background is assigned a low CPU priority. There are two priority-based methods to consolidate parallel workloads in the cloud: Conservative migration and consolidation supported backfilling (CMCBF) and Conservative migration and consolidation supported backfilling (AMCBF).

1) Conservative Migration and Consolidation supported BackFilling: Xiaogang Qiu et al. [4] proposed a priority based consolidation method, Conservative Migration and Consolidation supported BackFilling (CMCBF). That allows jobs to run in background VMs simultaneously with those foreground VMs to improve node utilization. It ensures that a job is dispatched to foreground VMs whenever the foreground VMs are idle or that job satisfies the node requirement. It allows jobs to run in background VMs simultaneously with those foreground VMs to improve node utilization. Compared to CMBF, CMCBF also deals with how to ensure that the background workload does not affect the foreground job. CMCBF only dispatches a job to run in background VMs when the corresponding foreground VMs have a utilization lower than a given threshold. The foreground VM utilization can be obtained from the profile of foreground jobs, or from the runtime monitoring data.

2) Aggressive Migration and Consolidation supported BackFilling: Xiaogang Qiu et al. [4] proposed another priority based consolidation method, CMCBF faces similar problem as CMBF when tracking backfilling jobs for each job. To reduce the cost, new modified algorithm is Aggressive Migration and Consolidation supported BackFilling (AMCBF) [4]. Only the job in head-of queue can preempt other jobs in AMCBF.

3. Perfomance Analysis

FCFS compared with other algorithms, then it has high response time and high waiting time. Migration Backfilling Gang Scheduling achieves better response time and waiting time than Gang Scheduling, Backfilling Gang Scheduling and Migration Gang Scheduling.Response time is higher in Reservation based EASY backfilling. In terms of waiting time, Key based EASY backfilling has lower waiting time than EASY but the waiting time is higher in Reservation based EASY backfilling.

CMCBF method requires tracking backfilling jobs for each job in the queue when making pre-emption decisions. When the number of jobs in the queue is large, the cost can be high. Simplify this algorithm called AMCBF to address this problem. In AMCBF, only tracks backfilling jobs for the job at the head of the queue and allows the head-of-queue job to pre-empt other jobs. The rest of jobs in the queue are not allowed to pre-empt jobs.

In AMCBF, there is a delay in the execution of jobs other than first job in the queue. This algorithm challenging to achieve responsiveness of parallel jobs and high processor utilization in the cloud. Another issue in a large data center is the communication cost is high because the processes of a job to be allocated to nodes may not be close to each other.

Table 3.1 Perfomance Analysis

Algorithms	Response time	Waiting time	Cost
FCFS	High	High	Low
Conservative BF	High	Low	High
Aggressive BF	Low	High	Low
Gang Scheduling	Low	High	Low
Backfilling GS	Low	Low	Low
Migration GS	High	High	High
KEASY BF	High	Low	High
REASY BF	High	High	High
CMBF	High	Low	High
AMBF	High	High	Low
CMCBF	Low	Low	High
AMCBF	Low	Low	Low

4. Conclusions

REASY produces the worst performance among our algorithms. The performance is achieved by MEASY better than KEASY, REASY and EASY. BGS is always better than MGS . MBGS which combines all techniques gang-scheduling, backfilling, and migration, provides the best results. In par- ticular, it can drive utilization higher than MGS, and achieves better slow down and wait times than BGS. AMBF achieves better performance than CMBF. CMCBF and AMCBF, significantly outperform FCFS, CMBF, AMBF, and EASY on response time and bounded slowdown. The performance of AMCBF degrades as the migration cost increases, but AMCBF bears high migration cost.AMCBF and CMCBF lead to better node utilization compared to other algorithms.AMCBF also shows slightly better performance than CMCBF.

In future work, will exploit a mechanism for further improve the node utilization and minimize the delays in the execution of the jobs in the cloud.

References

[1] Xiaogang Qiu, Ying Cai, Xiaocheng Liu, Bin Chen and Kedi Huang, ”scheduling parallel jobs using migration and consolidation in the cloud”, Mathematical Problems in Engineering, July 2012.

[2] Ahuva W. Mu’alem and IEEE Dror G. Feitelson, Senior Member, ”utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling”, IEEE Transactions on Parallel and Distributed system, June 2001.

[3] Xiaogang Qiu, Bing Bing Zhou, Bin Chen, Xiaocheng Liu, ChenWang and Albert Y. Zomaya, ”backfilling under two-tier virtual machines”, IEEE International Conference on Cluster Com- puting, 2012.

[4] Bing Bing Zhou Junliang Chen Ting Yang Xi- aocheng Liu, Chen Wang and IEEE Albert Y. Zomaya, Fellow, ””priority-based consolidation of parallel workloads in the cloud”, IEEE Transactions on Parallel and Distributed system, September 2013.

[5] Quinn Snell, David Jackson and Mark Clement, ”core algorithms of the maui scheduler”, Work- shop Job Scheduling Strategies for Parallel Processing, 2001.

[6] J. E. Moreiraz ,A Sivasubramaniamy, Y. Zhangy and H. Frankez, ”an integrated approach to parallel scheduling using gang-scheduling, backfilling and migration” IEEE Transactions on Parallel and Distributed Systems, March 2003.

[7] Jonathan Weinberg, ””job scheduling on parallel systems”, University of California, San Diego, 2001.

[8] Helen D, Karatza Ioannis, A. Moschakis.”performance and cost evaluation of gang scheduling in a cloud computing system with job migrations and starvation handling”, IEEE, 2011.

[9] Y. Etsion D. Tsafrir and D. Feitelson ”backfilling using system- generated predictions rather than user runtime estimates”, volume 18. IEEE Transactions on Parallel and Distributed Sys tems, 2007.

[10] Mohammad B. Dadfar Hassan Rajaei ””job scheduling in cluster computing: A student project”, Proceedings of the 2005 American Society for Engineering Education Annual Confer- ence Exposition, 2005.

Shahabanath K K received the bachelor’s degree in Information Technology from the Calicut University, Kerala in 2012. Presently she is pursuing her M.Tech in the department of Computer Science and Engineering from Calicut University, Kerala. Her research interests include Load balancing in cloud, Scheduling in cloud etc.

Sreekesh Namboodiri T received bachelor’s degree in Computer Science and Engineering from calicut university and master’s degree in Computer Aided Design from TamilNadu. Currently working as an Assistant Professor in Computer Science and Engineering Department, MES College of Engineering, Under the Calicut University, Kerala.