International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 5 – No. 9, July 2013 – www.ijais.org

Analysis and Performance Assessment of CPU Scheduling Algorithms in Cloud using Cloud Sim

Monica Gahlawat
L.J Institute of Computer Applications, Ahmadabad (Gujarat)

Priyanka Sharma
Institute of Science & Technology for advanced studies and Research, Anand (Gujarat)

ABSTRACT

Scheduling refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. CPU is by far the most important resource of the computer system. Recent advances in software and architecture of the system increased the complexity of the processing as the computing is now distributed and parallel. Job scheduling is complex in this environment. The VM (Virtual Machine) can use a distinctive VCPUs (Virtual CPU) running queue for each physical CPU, which is referred to Partition Queue Model (PQM). As a contrast, a Sharing Queue Model (SQM) of CPU scheduling algorithm can be used. This paper is analyzing and evaluating the performance of various CPU scheduling in cloud environment using CloudSim.

General Terms

CPU Scheduling algorithms, distributed and parallel computing, cloud computing.

Keywords

Virtual machine, virtual CPU, Cloud computing, CPU scheduling algorithms

1. INTRODUCTION

Scheduling refers to the set of policies to control the order of work to be performed by a computer system. Job scheduling is very challenging task in cloud computing because it is parallel and distributed architecture.

The job completion time determination is difficult in cloud because the job may be distributed between more than one Virtual machine. Virtual CPUs are assigned to each virtual machine. The virtual CPUs can use the shared running queue for each physical CPU or PQM can be used to use a separate queue for each physical CPU.

Before analysing and evaluating various CPU scheduling algorithms we first establish some terminology and classical ways of classifying the CPU schedulers. The CPU schedulers are basically classified as Proportional Schedulers and fair-share schedulers.

Proportional schedulers allocate CPU in proportion to the weights given to the virtual machines. This scheduler can not be viewed as a fair scheduler. A fair scheduler is that which allocates all the VM a time-averaged form of proportional sharing based on the actual use measured over long time periods. For example if two clients are sharing a system with equal CPU share then proportional scheduler will allocate the CPU to active clients according to their weights. It can be termed as space shared CPU allocation strategy. The clients cannot interfere into other clients CPU time even if the other client’s CPU is free to use. The counterpart of these kinds of algorithms is fare-share schedulers. In Fair share schedulers if one client is blocked and one is active we can assign more CPU time to active client and when the second one become active then the CPU is assigned to the second client to catch up with client1. The Schedulers can also be scheduled as time shared basis. In time-shared scheduling the equal time is provided to each client’s job. When client1 finishes the CPU can be allocated dedicatedly to client2

CPU schedulers can also be categorized preemptive and non-preemptive. In preemptive scheduling the algorithm is executed every time when a new task comes to the system. If the new task has the higher priority over the running task then the CPU will preempts the running task and executes the new task. In non preemptive scheduling the CPU allows every task to complete its CPU slice.

2. CPU SCHEDULING ALGORITHMS

CPU scheduling algorithms [8] decides how the CPU cycles should be allocated to the applications to achieve good performance. In a simple system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever. A scheduling system allows one process to use the CPU while another is waiting for I/O, thereby making full use of otherwise lost CPU cycles. In case of multiple processors, the scheduling gets more complicated, because now there is more than one CPU which must be kept busy and in effective use at all times. The scheduling algorithm should be designed such that it is capable of balancing the load between multiple processors

Multi-processor systems may be Heterogeneous. It is a set of cores which may differ in area, performance, power dissipated etc or Homogenous, where each core is same as the other. Even in the latter case there may be special scheduling constraints, such as devices which are connected via a private bus to only one of the CPUs.

The challenge is to make the overall system as "efficient" and "fair" as possible. Whenever the CPU becomes idle, it is the job of the CPU Scheduler (a.k.a. the short-term scheduler) to select another process from the ready queue to run next.

The storage structure for the ready queue and the algorithm used to select the next process are not necessarily a FIFO queue. There are several alternatives to choose from, as well as numerous adjustable parameters for each algorithm

2.1.1 Goals of CPU Scheduling Algorithm

Fairness is important under all circumstances. A scheduler must make sure that each process gets its fair share of the CPU and no process can suffer starvation

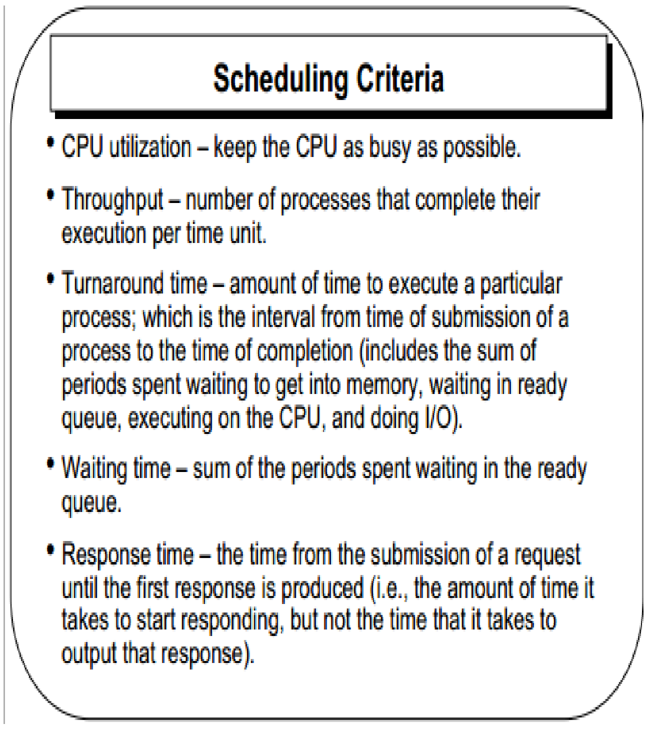
Efficiency: Scheduler should keep the system (or in particular CPU) busy cent percent of the time when possible. If the CPU and all the Input/output devices can be kept running all the time, more work gets done per second than if some components are idle.

Response Time: A scheduler should minimize the response time for real time applications.

Turnaround: A scheduler should minimize the time batch users must wait for an output.

Throughput: A scheduler should maximize the number of jobs processed per unit time.

A little thought will show that some of these goals are contradictory. It can be shown that any scheduling algorithm that favours some class of jobs hurts another class of jobs. The amount of CPU time available is finite, after all. So a good scheduling algorithm can be decided by focusing on the scheduling criteria explained in the figure 1.1.



Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible.
- Throughput – number of processes that complete their execution per time unit.
- Turnaround time – amount of time to execute a particular process; which is the interval from time of submission of a process to the time of completion (includes the sum of periods spent waiting to get into memory, waiting in ready queue, executing on the CPU, and doing I/O).
- Waiting time – sum of the periods spent waiting in the ready queue.
- Response time – the time from the submission of a request until the first response is produced (i.e., the amount of time it takes to start responding, but not the time that it takes to output that response).

Figure 1.1 Scheduling Criteria

3. EXISTING CPU SCHEDULING ALGORITHMS

Conventional operating systems typically employ a simple notion of First Come First Serve or priority for process scheduling. After advances in hardware technologies (Virtualization, multi-core CPUs), the scheduling scenario also changed and various sophisticated scheduling algorithms come in the existence.

Paper [1] presents the difference between the three CPU schedulers (Borrowed Virtual Time (BVT), Simple Earliest Deadline First (SEDF) and Credit Scheduler) available in Xen also presents the analysis of the performance of the schedulers in different workloads.

The BVT algorithm works on the virtual time. The algorithm dispatches the runnable VM with smallest virtual time first. This algorithm is a better choice for real time latency sensitive applications. The latency sensitive clients can get the priority over other by distorting the virtual time. The term “warp” is used for this process. The client affectively “borrows” the virtual time from its future CPU allocations because of this reason only the algorithm is named as “borrowed virtual time”.

SEDF is a dynamic algorithm used for real time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs i.e. task finishes, new task released, etc., the queue will be searched for the process closest to its deadline. This process is the next to be scheduled for execution. This algorithm in an extension of Shortest job first.

Credit Scheduler is the best fair share algorithm than BVT and SEDF. This algorithm is best fit for real time multiprocessor environment. In this algorithm each CPU has a priority queue of runnable VCPUs (Virtual CPUs). As VCPUs runs it consumes credits. Negative credit means the priority is over. This algorithm is also good for load balancing. When the credit of the VCPU becomes negative other VCPUs will get the priority over it. Active VMs earn credits every 30ms according to their weights, and burn credits as they run. Active VMs can be in either priority UNDER, meaning they have positive credit, or OVER, meaning negative. VCPUs in UNDER will always run ahead of VCPUs in priority OVER. Scheduling within a priority is round-robin. These scheduling algorithms works on VM level

[2] Presented the analysis of the performance of Berger Model and proposed model for job scheduling in cloud environment using CloudSim. This algorithm works at one upper level of the CPU scheduling algorithms and implemented in Data centre Broker so that Broker can decide which job should be bind to which VM. This algorithm is also works on VM level Scheduling

[5] Proposed a shared queue model for CPU scheduling algorithm and compared it with Partition queue algorithm that is used in both Xen and VMvare. The results shows that the shared queue model is more efficient than the partition queue.

[7] Explains the CPU scheduling policies in virtual environment. First-Come, First-Served (FCFS) is a traditional allocation policy that assigns one task per resource in the order in which the tasks have been submitted to the system. FCFS algorithm has the biggest drawback that the processes those are coming after other processes have to wait for the processes in the queue to complete. In cloud environment if any client want priority and is ready to pay more for immediate processing of his application then FCFS algorithm is not a good choice for the scenario presented above.

FCFS-NoWait (FCFS-NW) [7] is an extension to FCFS where jobs are not queued when no available VMs exist. Instead, this policy assigns jobs to VMs that are already running other jobs, round robin. This policy eliminates the wait time, but may introduce bottlenecks in the execution of jobs.

FCFS-MultiQueue is an extension to FCFS where several FCFS queues, one for each range of job durations, are maintained. The simplest case considered here, FCFS-2Q, has two queues, one for short jobs and another for long jobs. Although an estimation of the runtime is necessary for this policy, it is enough to have partial knowledge of it to classify jobs [7].

[9] Describes various algorithms like SJF (Shortest Job First). The algorithm is based on the priority. The process which is small has the higher priority over other processes. The algorithm can be pre-emptive or non-preemptive. If pre-emptive, the disadvantage with this algorithm is starvation. Round robin algorithm assigns CPU to each process a fixed time slice. All the processes have equal priority.

[7] Describes a CPU scheduling algorithm SRT (Shortest Remaining time). The ready queue will be on the basis of the process which is going to be completed recently. Priority based algorithm can be used to provide customers priority

based scheduling. This algorithm is best suited for the cloud environment. The cloud providers can give cost based scheduling to the customers. The customer who wants to complete the jobs first should have to pay more for the CPU cycles.

4. EXPERIMENTAL SETUP

The CloudSim 3.0[6] toolkit is used to simulate cloud environment. The experiments are performed with Sequential assignment which is default in CloudSim and other two very common CPU scheduling algorithms i.e FCFC, SJF and priority scheduling. The less number is termed as the higher Priority . The jobs arrival is Uniformly Randomly Distributed to get generalized scenario. The configuration of the cloud includes 2 datacenters, 2 VMs and 3 hosts. We have implemented the algorithm using 4 cloudlets for simplicity and also analyzed the algorithms for more cloudlets .As the cloudlets (applications) are submitted by the user it is the task of the cloud broker (Cloud broker works on behalf of client and finds out the best VM to run the application, the VM is decided by looking at different parameters like size, bandwidth, cost of VM) to assign those tasks to the VM and then Virtual Machine Manager (VMM) decides the host on which this VM should be allocated based on the VM Allocation policy. After VM is assigned to the host the VM starts running the cloudlets i.e. applications. Here CPU scheduling algorithms come into existence. Every VM has a virtual CPU called PE in CloudSim. The VM can have one PE or more which simulates the original multi-core CPUs. We have analyzed the two very common and basic CPU scheduling algorithms i.e. first come First Serve and Shortest Job First Algorithm. The Analysis in the table 1 shows that the cloudlets are sorted based on the size of the Cloudlet in shortest job first algorithm which results in less turnaround time and waiting time. The throughput will be more in case of SJF (Shortest Job First).

Table1: Analysis of FCFS and SJF and Priority Scheduling in Cloud Environment

Cloudlet ID	**Cloudlet Length**	**FCFS**			**SJF**			**Priority Scheduling (PS)**		
		Seq. No.	Turnaround Time	Waiting Time	Seq. No.	Turnaround Time	Waiting Time	Seq. No	Turnaround Time	Waiting Time
0	1000	0	1	0	0	1	0	0	1	0
1	2000	1	3	1	3	4.5	2.5	2	3.5	1
2	3000	2	6	3	1	7.5	4.5	1	5.5	3.5
3	1500	3	7.5	6	2	2.5	1	3	7	5
			4.375	**2. 5**		**3.875**	**2**		**4.25**	**2.3**

Table 2: Turnaround time by increasing the cloudlets

No of Cloudlets	**FCFS**	**SJF**	**PS**
5	4.5285	4.1290	4.2569
10	9.057	8.258	8.5138
15	18.114	16.516	17.0276
20	36.228	33.032	34.0552
25	72.456	66.064	68.1104

5. CONCLUSIONS

The results show that shortest job first and priority scheduling algorithms are beneficial for the real time applications. Because of these algorithms the clients can get precedence over other clients in cloud environment. The cloud providers with these algorithms can decide their cost model. If any client will get priority over the other the client has to pay more money .In this paper we have analyzed only three very common algorithms. In future we will simulate other adaptive and dynamic algorithms suited the virtual environment of cloud.

6. REFERENCES

[1] Ludmila Cherkasova, and I.N. Sneddon, “Comparison of the Three CPU Schedulers in Xen,” published in ACM SIGMETRICS Performance Evaluation Review archive , September 2007Pages 42-51

[2] V. Venkatesa Kumar and K. Dinesh, “Job Scheduling Using Fuzzy Neural Network Algorithm in Cloud Environment” Bonfring International Journal of Man Machine Interface, Vol. 2, No. 1, March 2012

[3] Monika Choudhary, Sateesh Kumar Peddoju , “A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment” International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 2, Issue 3, May-Jun 2012, pp.2564-2568

[4] David Villegas, Athanasios Antoniou,Seyed Masoud Sadjadi, and Alexandru Iosup.” An Analysis of Provisioning and Allocation Policies for Infrastructure-as-a-Service Clouds: Extended Results” Draft, Dec 2011.report number PDS-2011-009, ISSN 1387-2109, Accessed on 10 Mar 2012.

[5] Xindong You; Xianghua Xu; Jian Wan; Congfeng Jiang, "Analysis and Evaluation of the Scheduling Algorithms in Virtual Environment," Embedded Software and Systems, 2009. ICESS '09. International Conference on , vol., no., pp.291,296, 25-27 May 2009

[6] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE), Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011.

[7] Maria Abu, Aminu Mohammed Sani Danjuma and Saleh Abdullahi.”A Critical Simulation of CPU Scheduling Algorithm using Exponential Distribution” IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 2, November 2011

[8] Salot, Pinal. "A survey of various CPU scheduling algorithms in clouds.”, International Jounal of Research in Engineering and Technology ISSN: 2319 – 1163, pp 131-135

[9] Oyetunji, E. O., and A. E. Oluleye. "Performance Assessment of Some CPU Scheduling Algorithms." Research Journal of Information and Technology 1.1 (2009): 22-26.

[10] Lingyun Yang, Jennifer M. Schopf, and Ian Foster. 2003. Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing* (SC '03). ACM, New York, NY, USA, 31-. DOI=10.1145/1048935.1050182 2