Methodology for Performance Improvement of Future Request Predicting Disk Scheduler for Virtualization

Ashwini Meshram
Department of Computer Science and Engineering
G.H.Raisoni College of Engineering, Nagpur, India

Urmila Shrawankar
Department of Computer Science and Engineering
G.H.Raisoni College of Engineering, Nagpur, India

ABSTRACT

Virtualization enables to switch different operating systems without reboot. It enables live migration from one Operating System (OS) to another and results in proportional sharing of storage resources. Virtualization is gaining importance day by day in the fields of academics, industry and business. Performance is the major requirement to fulfill today's need. As far as, computer's workload is concerned, there is a need of high performance computing system. As the use of virtualization has increased tremendously there is much focus on optimizing the virtual machine performance. Disk scheduling within the virtual environment plays a key role in optimizing the overall system performance. Prior works on disk scheduling in virtual environment found it difficult to achieve system performance because of the high disk seek time. This paper presents an approach towards the performance improvement of disk scheduling in virtualized environment by future request arrival prediction. The basic idea is to examine whether the traditional High Throughput Token Bucket Disk Scheduling algorithm (HTBS) is still efficient for the performance improvement in virtualized environment.

General Terms

Virtualization, Operating System and Algorithms

Keywords

Disk Scheduling, Virtualization, Throughput, Seek Time, Future Request

1. INTRODUCTION

Disk Scheduler is an important component of operating system which aims to improve disk utilization and to achieve high throughput by proportionally sharing the storage resources. All the information produced in the world is stored on the disk [1]. For controlling and providing the memory to all the requests, operating system uses the concept of Disk Scheduling. Today's, modern computers are powerful enough to run entire operating systems within the main operating systems which is nothing, but operating system virtualization. Virtualization allows emulation of one operating system within another. Without rebooting from one OS to another, virtualization allows to run applications on different operating systems by allowing live migration.

The disk is a resource which is shared by different applications. Although there is more than one operating system, but the disk is a single resource which is shared by all. As there is sharing of disk, there must be a policy to use disk properly. Virtualization has brought several different challenges to disk scheduling as the disk technology advances. The disk scheduler within the Virtual Machine Monitor (VMM) plays a very important role in determining the overall fairness and performance characteristics of the virtualized system. As virtualization allows multiple operating systems on the same set up, these multiple system can be used by different users. Users requirement differ according to their need and time. From the users log it is helpful to schedule their recent requests using a better scheduling algorithm so that it improves the overall system throughput by minimizing seek time.

The latency of a disk plays an important role in the design of disk schedulers. But, disk latency characteristics in the virtual environment depends not only on the disk being used but the additional queuing and processing that happens in the virtualization layer. Furthermore, virtual machine is being provided with the virtual disk which shows the limitations of existing disk schedulers [2]. In order to improve the performance of the system, the HTBS algorithm is implemented in order to improve the seek time of the system. The disk scheduling algorithm HTBS has been successfully implemented for the traditional disk scheduling [3].

The paper is organized as follows: Section 2 describes the related work. Section 3 gives experimental set up. Section 4 and 5 discusses the complete results and conclusion.

2. RELATED WORK

The virtualization technology is basically used for storage consolidation which permits independent configuration of multiple operating systems, software, and device drivers [4][5]. Virtualization helps to achieve greater system utilization and it lowers the total cost of implementation. It is helpful as it responds more effectively to changing business conditions in enterprise, government and organizations [6].

Operating system virtualization is, conceptually, a virtualized operating environment [7]. Scheduling brings new issues in the virtual environment because virtual machine contains more properties which are used for scheduling [8]. In the virtual environment each guest operating system runs on top of the virtual machine monitor. The virtual machine shares all the physical devices available with the host operating system [9]. Each guest has virtual disk share of the single shared physical disk [10]. The aim for using disk scheduling algorithms is to minimize the seek time required to reach the particular memory block. In virtual environment there is a sharing of hard disk between the host OS and guest OS. Hence to minimize the total seek time the HTBS algorithm is being presented in the virtual environment.

Most of the work on traditional disk scheduler concentrates on workloads running on native operating systems [11, 12]. As the storage system has evolved to new technologies, the traditional disk scheduling algorithms which are used in virtual environment have become obsolete.

In the recent studies by Boutcher et.al. [13] about the disk scheduling in virtual environment, suggests the right combination of schedulers to maximize throughput and fairness between VMs. Seelam and Teller proposed the

Virtual I/O scheduler [14]. The scheduler is work –conserving scheduler which in the presence of multiple VM's provides fairness among them. Gulati, Ahmad, and Waldspurger suggested the PARDA system which uses proportional allocation of a distributed storage resource among virtualized environments [15]. It employs a global scheduler that enforces proportionality across hosts and a local scheduler for VMs running on a single host. In [16], the authors studies the impact of virtualization and shared disk usage in virtualized environments on the guest VM-level I/O scheduler, and its ability to continue to enforce isolation and fair utilization of the VM's. VM shares I/O resources among applications and application components deployed within the VM.

All the prior work carried out in disk scheduling in virtual environment uses work conserving algorithms which does not predict the arrival of future request. The paper presents an algorithm for disk scheduling is non-work conserving in nature. It introduces the concept of prediction of future request [17]. The concept is used in the case of virtual environment to improve the system performance.

3. EXPERIMENTAL SETUP

VM used in this experiment is VMware player 5.0. It requires minimum of 6 GB of free disk for guest OS to run on the host OS. Each guest operating system is viewed as a single process of the host operating system that runs in the user space. The virtual machine is configured with 40 GB of hard disk space and 512 MB of RAM. The host operating system used in this experiment is Windows 7 Ultimate. Through the VMware player the guest operating system that is installed is Window XP.

The space required for guest OS is greater than or equal to the sum of space required by the guest's raw image files, the space required by the host operating system and the swap space that guest OS will require and can be expressed as:

$$Total\ Required\ Space\ for\ Guest = Images + Host + Swap\ space \text{----- (1)}$$

Using swap space can provide additional memory beyond the available physical memory.

Figure 1 shows the complete virtualization scenario. The host OS is the main operating system and the guest OS is the one which is installed through virtual machine monitor (VMM).The VMM acts as an abstraction layer between host and guest OS. The task of VMM is to allocate the virtual resources to the guest OS. The resources are shared among host and guest operating system through the VMM.

The High Throughput Token Bucket (HTBS) scheduling algorithm uses the concept of future request prediction. The algorithm dispatches the disk requests so that spatial locality is maintained, which improves the overall system performance in terms of seek time. In this experiment, the same algorithm is used in the virtual environment to examine its working whether it improves the system performance in terms of seek time. To compare the results existing disk scheduling algorithms are implemented which are the NOOP scheduler (No Operation) and CFQ (Commonly Fair Queuing) scheduler. NOOP follows simple uses the FIFO policy where the files are read or written according to their order of arrival. CFQ being fair allocates equal time slice to each request. The flowchart of the HTBS algorithm which is going to be used in the virtual environment is as shown in figure 2.

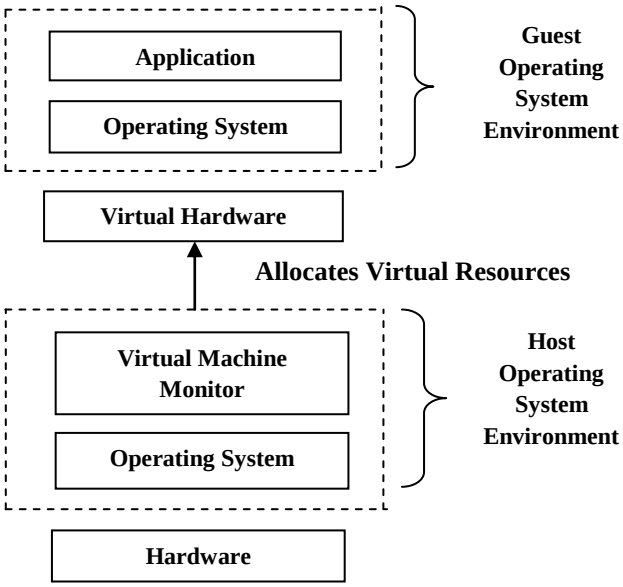

Fig 1: Virtual Environment

The stepwise description of the HTBS disk scheduling algorithm is as follows:

Step1: Maintain a queue to hold at max 10 file read requests.

Step2: Add file read/write requests in the queue.

Step3: If number of files exceeds 10

Then display message "Number of files should be less than 10"

Step4: Call Get_Volume_Map () to calculate the start address and end address of files

Step5: Select the request from the user's log with maximum frequency of occurrences

Step6: Apply loop from 1 to number of files

Step7: Dispatch the request having smallest Start LBN

Step8: Note the Start Time and End time of each file read request.

Step9: Calculate:

$$Disk\ Read/Write\ Rate = Block\ Read/Written\ perSecond * Block\ Size$$

$$Seek\_Time = Seek\_Factor * [abs(Block\ Access - Current\ Head\ Position)]$$

$$Average\ Execution\ Time = Total\ Ellapsed\ Time/No.of\ Files$$

The system is provided with the user login. The user can read /write up to 10 files at a time and every time, the user data is maintained in a separate log file. From the respective user's log file the future request is predicted. To search for the arrival of future request, the disk must wait for a specific time period. This time period is known as Twait. This factor limits the time the disk can be kept idle. At a time, limited number of consecutive requests can be issued which is being controlled by the factor Bmax. Greater the number of requests with locality grater is the performance of the system. The seek_time, required to the disk head from current position to the to the target track, is given by:

$Seek\_Time = Seek\_Factor * [abs(Block\ Access - Current\ Head\ Position)]$ ----- (2) [18]

Where, Block access is the block location on the disk. Seek_factor will be assumed as 0.3.

A drive of size 63.8 GB is being shared among host OS and guest OS. Different files of variable sizes are maintained in the shared drive. The scheduler is provided with the file read write operation from the shared drive. At a time, different files of varying sizes are given as input to the scheduler. All the files residing on the disk has a physical address in the form of Cylinder Head and Sector. This physical address is converted into Logical Block Numbers (LBN) using following formula.

$$LBN = ((C * HPC) + H) * SPT + S - 1 \text{ ----- (3)}$$

Where, HPC: Heads per Cylinder, SPT: Sectors per Track, C: Cylinder, S: Sector

4. RESULTS AND DISSCUSION

When the file read/write requests are given to the scheduler their starting LBNs are calculated. The scheduler dispatches the requests with strongest spatial locality on the basis of LBNs. As the algorithm being used is non-work conserving, it will predict the arrival of future request i.e. instead of considering the requests from the pending queue only; the scheduler will also consider the future request. If the incoming future request is nearer than the request in the pending queue, the scheduler will dispatch the future request else the request from the pending queue.

Table 1 and figure 3 show the result of the HTBS scheduler implemented in host and in guest OS. Table 1 gives the numerical data and figure 3 shows the graphical results for the same. The scheduler has been tested in the host as well as in the guest. 12 files with different sizes has been given as input to the scheduler and time required for reading those files in the host as well as in the guest has been noted. In addition with the HTBS, NOOP and CFQ has also been implemented in the guest OS.

The results show that, the HTBS algorithm improves the execution time as compared with the CFQ and NOOP. When the file size is small the execution time required by all the three algorithms is same. But, as the file size grows gradually HTBS requires minimum execution time. From the table, it is also observed that, in spite of having the sharing of disk there is a slight difference between the execution time in host and in guest OS by HTBS algorithm.

Table 2 and figure 5 shows the average disk read/write rate for the three algorithms. Table 2 shows the numerical data and figure 5 shows the graphical results for the same. From the graph, it is observed that the disk read write rate using HTBS is high as compared to NOOP and CFQ. CFQ gives the lowest disk read/write rate.

Table 3 shows the seek time required by the three schedulers. Figure 6 shows the graphical results for the average seek time by three algorithms.

The maximum queue limit of these schedulers is 10. The number of files is varied and for each one the three schedulers are tested.

Table 1. Request Execution Time in Guest OS

Sr. No.	File Size	Execution time in Guest OS (Sec)NOOP	Execution time in Guest OS (Sec)CFQ	Execution time in Guest OS (Sec)HTBS	Execution time in Host OS (Sec) HTBS
1	577	0.6567	0.6412	1.3901	1.3416
2	2389	2.7192	2.6551	2.5773	2.4648
3	2481	2.8239	2.7573	2.7959	2.6208
4	2929	3.3338	3.2552	3.2489	3.1117
5	4033	4.5904	4.4822	4.4517	3.1264
6	4961	5.6467	5.5135	5.5763	5.1792
7	5441	6.1932	6.047	5.2876	5.231
8	5457	6.2113	6.0648	6.0761	5.7408
9	8593	9.7801	9.5501	5.5619	4.1652
10	9185	10.4549	10.208	10	9.516
11	9201	10.4728	10.2258	9.5125	8.9076
12	9777	11.1284	10.866	10.8715	10.218
Avg.Execution Time(Sec)		8.2568	8.0621	7.3122	5.1352	8.2568

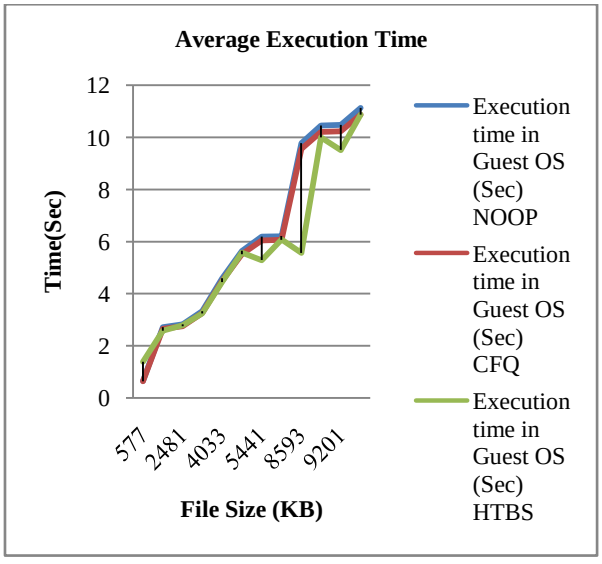

Fig3. Execution Time (Sec) in Guest OS by NOOP, CFQ and HTBS

CFQ, as being fair, allocates equal time slice to each of the file and executes each file, until its time slice is expired in a round robin fashion. From the graph, it is observed that the average seek time using CFQ is maximum as compared to NOOP. NOOP uses the FIFO policy, and executes the requests in the order of their arrival. So in the case of this scheduler the average seeks time is found to be minimum as compared to CFQ. HTBS dispatches the request in order to maintain spatial locality, so in the case of HTBS, average seek time is lowest as compared to both NOOP and CFQ.

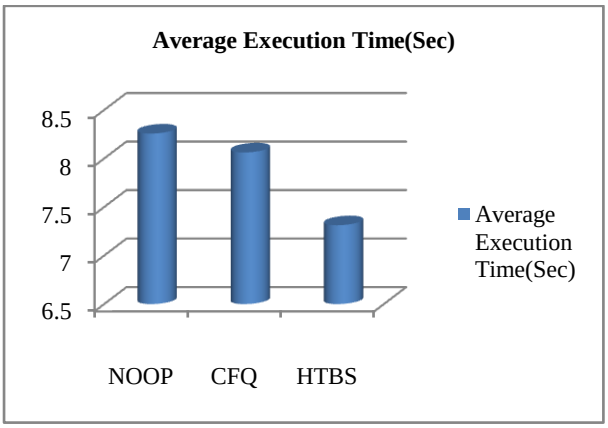

Fig4. Average Execution Time (Sec) in Guest OS by NOOP, CFQ

Table 2. Disk Read/Write Rate (MB/Sec)

No. of Files	Disk Read/Write Rate(MB/Sec)		
	NOOP	CFQ	HTBS
5	1.555	1.5601	1.5601
7	2.8146	2.8167	2.8276
10	3.1889	3.2013	3.1988
Avg. Disk Read/Write Rate (MB/Sec)	2.5195	2.5260333	2.5288333

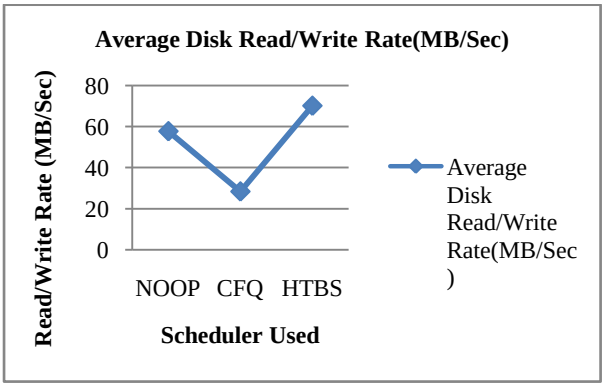

Fig 5. Average Disk Read/ Write Rate (MB/Sec) in Guest OS by NOOP, CFQ and HTBS

Table 3. Seek time (Millisecond)

No. of files	Seek time (Millisecond)		
	NOOP	CFQ	HTBS
5	390	734	437
7	421	620	375
10	984	625	515
Avg. Seek Time (Millisecond)	598.3333	659.6667	442.3333

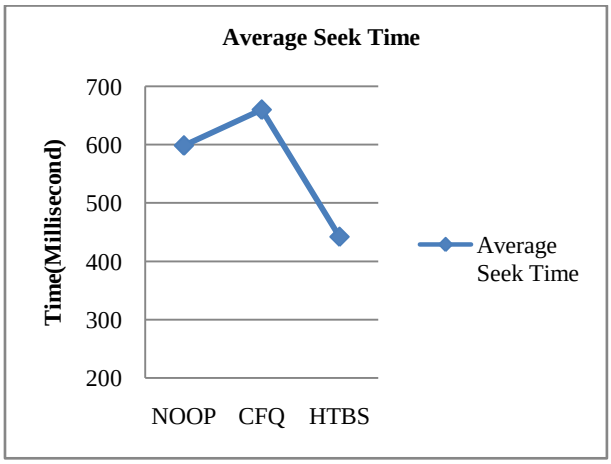

Fig 6. Average Seek Time (Sec) in Guest OS by NOOP, CFQ and HTBS

5. CONCLUSION

The work presented in this paper implements HTBS disk scheduling algorithm in the virtual environment. The algorithm is working fine for the traditional disk scheduling. As, the algorithm is non-work conserving, it predicts the arrival of future requests. It dispatches the request with the strongest spatial locality. While dispatching the requests, it not only considers the requests from the pending queue but the future request as well which is predicted on the basis of concurrency of occurrences. This reduces the seek time overhead. After comparing the results with the existing virtual disk scheduling algorithm i.e. NOOP and CFQ, it is observed that HTBS improves the overall system performance by minimizing execution time and the seek time as well as maximizing the disk read write rate. Ultimately, this will result in the better utilization of system resources even if they are in sharing mode. Therefore, it can be concluded that the same algorithm can be used for disk scheduling in the virtual environment where there is a competition of resources to increase the performance of the system.

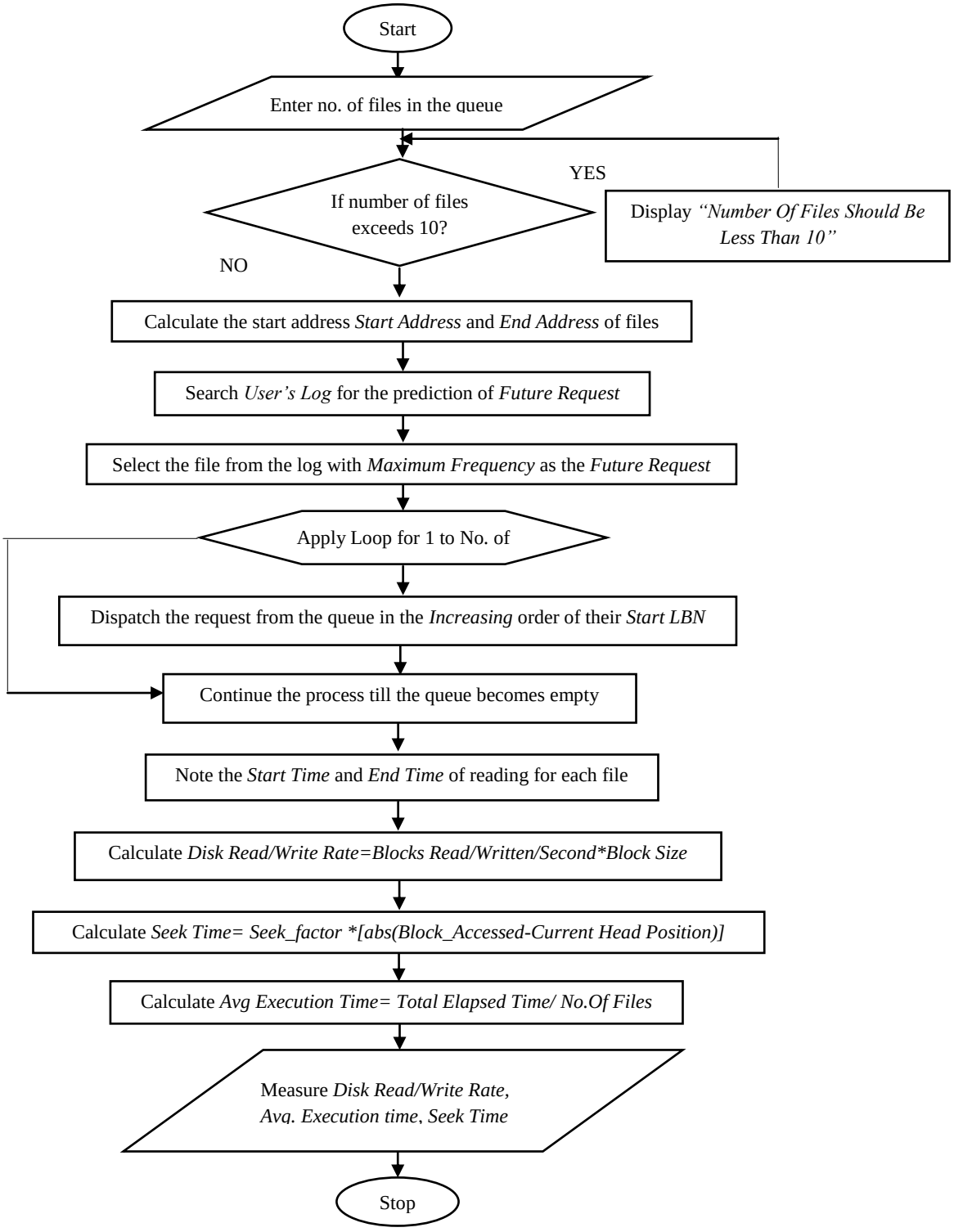

Fig 1: HTBS Algorithm

6. REFERENCES

[1] George Amvrosiadis, Alina Oprea, Bianca Schroeder "Practical Scrubbing: Getting to the bad sector at the right time", in the Proceedings of 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2012, USA,pp. 1-12

[2] Y.Zhang, B.Bhargava "Self Learning disk Scheduling" , in IEEE Transaactions on Knowledge and Data Engineering, vol. 21, IEEE computer Society,Jan 2009, pp. 50-65

[3] P.E.Rocha,Luis C. E. Bona "A QoS Aware Non-work-conserving Disk Scheduler" in the 28th symposium on Mass Storage System And Technologies(MSST) 2012, IEEE, San Diegao, CA,pp.1-5

[4] Ignacio M. Llorente "An Introduction To Virtualiaztion And Cloud Technologies To Support Grid Computing" , New Paradigms: Clouds, Virtualization and Co. EGEE08, Istanbul, September 25, 2008

[5] Ahmed Elnably, Kai Du, Peter Varman, "Reward Scheduling for QoS in Cloud Applications" in the proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), USA,pp. 98-106

[6] Yiduo Mei , Ling Liu and Xing Pu "Performance measurement and alalysis of network io applications in virtualized cloud" 3rd IEEE International Conference on Cloud computing, USA,2010,pp.59-66

[7] Mendel Rosenblum , Tal Garfinkel "Virtual Machine Monitors: Current Technology and Future Trends" in the proceedings of IEEE Computer Society, Los Alamitos, CA, USA, May 2005, pp. 39-47

[8] Hsu Mon Kyi and Thinn Thu Naing "An Efficient Approach for Virtual Machine Scheduling on a Private Cloud Environment",4th IEEE International Conference on Broadcast Network and Multimedia Technology (IC-BNMT), Myanmar,2011,pp.365-369

[9] Kuan-Rong Lee, Meng-Hsuan Fu, Yau-Hwang Kuo "A Hierarchical Scheduling Strategy for the Composition

[10] Tim Kaldewey, Theodore M. Wong, Richard Golding, Anna Povzner, Scott Brandt, Carlos Maltzahn, "Virtualizing Disk Performance" in the proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium, 2008, USA,pp. 319-330

[11] S. Pratt and D. Heger. Workload dependent performance evaluation of the linux 2.6 i/o schedulers. In Proceedings o the Linux Symposium, volume 2. Ottawa Linux Symposium, 2004.

[12] S. Seelam, R. Romero, and P. Teller. Enhancements to linux i/o scheduling. In Proceedings of the Linux Symposium Volume Two, pages 175{192. Ottawa Linux Symposium, July 2005.

[13] D.Bouthcher and A.Chandra "Does Virtualization Make Disk Scheduling Passe?" in Proceedings of ACM SIGPOS Operating System Review Volume 44, January 2010, New York, USA,pp 20-24

[14] S. R. Seelam and P. J. Teller "Virtual i/o scheduler: a scheduler of schedulers for performance virtualization" In VEE '07: Proceedings of the 3rd international conference on Virtual execution environments,2007,USA,pp.105-115

[15] A. Gulati, I. Ahmad, and C. A. Waldspurger "Parda: proportional allocation of resources for distributed storage access" In FAST '09: Proccedings of the 7th conference on File and storage technologies. USENIX Association, 2009.

[16] Mukil Kesavan, Ada Gavrilovska, Karsten Schwan "On Disk I/O Scheduling in Virtual Machines" in the Second Workshop on I/O Virtualization (WIOV '10), March 13, 2010, Pittsburgh, PA, USA.

[17] Ashwini Meshram, Urmila Shrawankar "Future Request Predicting Disk Scheduler For Virtualization" in the proceedings of Journal of Computer Science and Engineering, Vol.16, Issue 2, Dec.2012.

[18] S. Y. Amdani M. S. Ali S. M. Mundada "Mathematical Model for Real Time Disk Scheduling Problem" in the special issue on Emerging Trends in Computer Science and Information Technology-2012(ETCSIT2012) of International Journal of Computer Applications