

WEIGHTED FAIRNESS RESOURCE ALLOCATION OF DISKS IN XEN

¹Wang Xiaojing, ^{*2}Tong Wei, ¹Ren Jia, ¹Ding Linjie, ²Liu Jingning

¹ School of Computer Science, Huazhong University of Science and Technology

²Wuhan National Laboratory for Optoelectronics, Wuhan,China

*Corresponding Author: tongwei2006@gmail.com

ABSTRACT

Virtualization improves the utilization ratio of resource, saving considerable amount of equipment, energy, management, etc. However, the benefits of virtualization are at the penalty of fairness but inefficient resource sharing. Various tasks in virtual domains often have different demands of resources. Therefore, allocation of resource and scheduling function are particularly important. A weighted max-min fairness allocation of disk resources is proposed in virtual platform. We focus on properties of sharing incentive, initial irrelevant and Pareto efficiency while designing the allocation algorithm. This Allocator is implemented in Xen virtual block devices manager. Tests in this paper show that the weighted max-min Allocator has a better allocation of resources and guarantees the quality of services.

KEYWORDS

Virtualization, Xen, block device, resource allocation, Qos

1. INTRODUCTION

Virtual machine (VM) technology, which emphasizes ease of server managements and maximum utility of hardware resources, is drawing vast attention from both industry and academia. With virtualization, logical resources could be simulated upon one single physical node and be allocated to various virtual hosts. In traditional data center, taking security and reliability into consideration, each node possibly provides one service in one Operating System instance. VMs provide suitable resources and portable environments to meet the requirements of modern servers.

The benefits of virtualization are at the penalty of fairness but inappropriate resource allocation, while sharing the hardware resources between VMs. As currently more researches on absolutely fairness and a full utilization of resources [1], virtual machine technologies should focus on quality of services in every virtual domain. Hard drive resources are allocated to VMs and scheduled by the virtual machine monitor (or hypervisor). Since existing operating systems are typically not designed for a virtualized environment, the scheduler used at hypervisor level has exhibited several drawbacks [2]: disorder optimization, inability to apply arbitrary bandwidth, and poor anti-jamming ability. In other words, they cannot provide effective disk resources allocation either functionally or compatibly.

In this paper, we implement a new algorithm to allocate the block device for both fairness and quality of services in virtualization environment. In this new framework, we drop off the original allocation rules that distribute all the block device resources to virtual domains with full equality. Weighted max-min fairness is proposed in this algorithm. The intuition behind is to incentive VMs to take full use of disk resources [3] and ensure the quality of services of each VM. For such an algorithm, allocations would be sharing incentive, initial irrelevant and Pareto efficiency. The virtual machines are incentive to take full use of hardware resources and allocations would only be affected by the virtual domains' demand. Finally, Pareto efficiency is made to ensure the quality of services for each domain. We have implemented the weighted max-min fairness in Xen block device drivers, to manage and schedule the hard drive devices for the virtual domains. A comparison with fairness sharing scheme used before is made to verifying the new algorithm. With the test result, we believe that the new allocation scheme is more applicable for various services in virtual domains.

The rest of the paper is organized as follows. Section 2 motivates the problem of disk allocation in Xen. Section 3 introduces the weighted max-min fairness scheme we would implement in the system. Section 4 lists the mechanism in Xen to carry out the new rules. Section 5 presents the experimental results based on different allocation algorithms. We conclude the paper in Section 6.

2. MOTIVATION

While many of the researches focus on resource fairness and latencies in virtual environment, they are considering the network resources in VMs. Unlike these approaches we focus on the allocation of disk resource that is more complicated.

Para-virtualization in Xen [4] implements a front-end driver and a back-end driver to bridge the guest domain and privilege domain. As all the requests sent from the guest domain (unprivileged domain, or DomU) would be resorted by hypervisor in privilege domain (Dom0) [5], a common Noop scheduler is used to avoid meaningless sorting in DomU. Thus, a specific scheduler in Dom0 would be critical to performance of services in VMs. Noop, Deadline, Anticipatory and CFQ [6] are existing schedulers used in Dom0 to allocate the block devices for virtual domains. But none of them could allocate a specified resource of VM. This leads to sketchy allocation, a poor match for task demands and quality of services.

Figure 1 introduces the structure of Xen virtual block devices. Requests are initiated by applications in file systems in guest domains that would be passed to the privilege domain by the front driver. Back-end driver calls the file system and physical device driver to respond after receiving the requests. For a better performance, Dom0 chooses CFQ as a default I/O scheduler for fairly allocation of bandwidth resources to all VMs. In that case, as some of the DomUs demand a higher bandwidth, the only approach is to increase the priority of the corresponding tapdisk process by system administrator. These existing fair schedulers ignore the various demands of users, and take an average resources allocation to guest domains. This leads to inefficient allocation and a poor match for the user demands.

To meet stringent quality of service requirement, several researches are made in virtual environment. In [7], Credit scheduling is used to count the utility of CPU resources, while an allocation of credits could only be made when virtual domains initiate. After the start of virtual domains, allocation is fixed and leads to inefficient allocation. [8] used time stamp to limit the process of network requests while this method could not be applied to block devices as the sequences of requests affect the disk performance a lot. Paper [9] introduced an allocation of multiple resource types including CPUs and Memories. The DRF did not take disk resource into consideration but gave us a hint. Max-min fairness is first put forward in economics to get a maximum profit by allocation of limited resources. Several papers were focused on resource allocation on CPU [10,11,12], link bandwidth in wireless network [13,14] and storage [15,16]. CloudScale [17] applies fair allocation of CPU and memory resources when there is contention for resources on a host with Xen platform. Zhang et al. [18] implemented an approach to quickly reassign of the CPU and memory resources for a sudden increase in application load in VMware virtual machines. These works [8,17,18] concentrate on CPU and memory resources of VMs, while we focused on disk resource allocation in virtualization platform. A modification of max-min fairness algorithm could be applied to the virtual block device allocation.

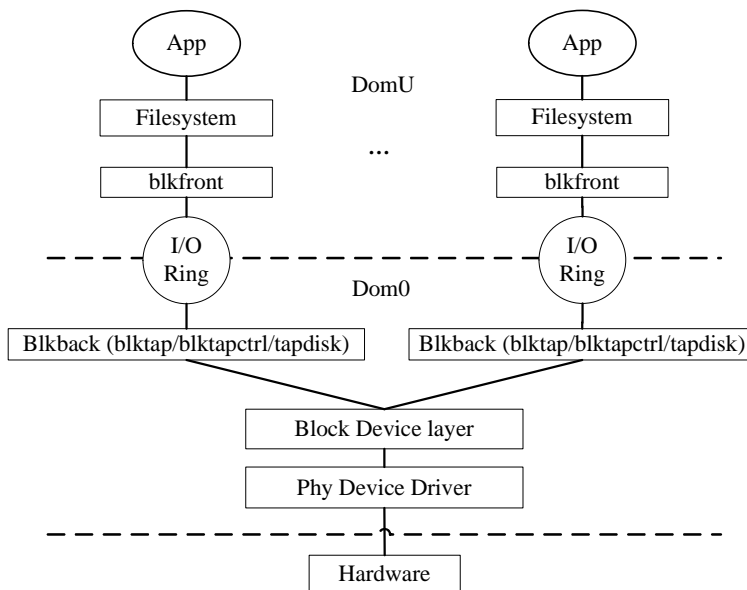


Figure 1 The structure of Xen virtual block device

3. WEIGHTED MAX-MIN FAIRNESS

To address the problem above, we focus on designing a max-min fairness allocation policy for virtual block devices that could also be implemented on virtual disk. In section 2, properties of sharing incentive, initial irrelevant and Pareto efficiency are described as the key features of the allocation algorithm. Performance of applications in virtual domains should not be less than physical servers so that users need not to concern about the performance. Resource allocation is made according to the actual usage dynamically and limited by weighted value but not the initial demands. After each allocation, applications could get

enough resources to process the requests or get proper resources to maintain the services while the physical resource is not enough.

The algorithm 1 shows the procedure of weighted fairness allocation. Δa indicates the change of demands by applications that would be introduced in section 4. A_i is adjusted by Δa in unit time and the algorithm would check if the modification exceeds the limitation of the virtual domain. To these who decrease the demands of resources, allocation would be reduced. But for those who increased the demands, they would distribute the residual of the resources by their weight. And the most important thing is that, a reservation of each domain is kept to ensure the performance of the domain. So, the system would check if sum of all the reservations exceeds the physical resource.

Algorithm 1 Weighted Max-min Fairness

```

R = <r1, r2, ..., rn> // reservations of each domain
W = <w1, w2, ..., wn> // weights of each domain
L = <l1, l2, ..., ln> // limitations of each domain according to weights

Ai = Ait + Δ ai // adjustment of Ai by the step of Δ ai

If Ai > li then // allocation exceeds the limitation
    Ai = li
For each Aj that Aj < rj || Δ aj < 0 // total consumption of reservations and decreases
    T0 = Σj=0m Aj
For each Aj that Aj > rj && Δ aj > 0 //distribute the rest resources by weights
    Δ aj = (T - T0 - Σi=1t rj) ·  $\frac{w_j}{\sum_{i=1}^t w_i}$ 
    Aj = rj + Δ aj
Return
    
```

The basic idea of this algorithm is to first allocate the reservation resources for these contending domains. If a guest domain could not utilize the allocated resources in last unit time, algorithm would decrease the resource allocation by a . The residual resource is distributed among those domains whose demand increases. In this case, no domain is penalized for the lack of resource and a certain quality of services is guaranteed by the constraint of reservation. And the guest domains would get enough allocations once there is sufficient resource. The adjustment is implemented dynamically according to the usage of allocated resource in last circle so that virtual domain could take full use of physical resources and avoid an inefficient allocation.

4. ALLOCATION IMPLEMENTATION

The allocation manager consists of three modules: Controller, Monitor and Allocator. Figure 2 shows the modules in a system with two guest domains. Controller and monitor are placed

in the tapdisk of each domain to implement the resource usage control and feedback the status of each domain to Allocator. Allocator has only one instance in blkTapCtrl and decides the resource allocation according to information from different guest domains. Controller and Monitor are running in the same process, sharing data via global variables; Allocator, however, runs in another process and communicates with tapdisk via message queue.

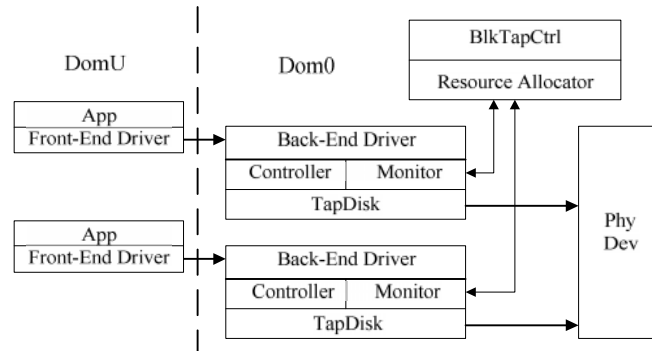


Figure 2 The backend driver with bandwidth manager

4.1. Controller

Bandwidth Controller records and limits the throughput of tapdisk. The workflow is described in Figure 3. `request_limit` is the amount of requests allowed per unit time and is the maximum bandwidth limit. The value of `request_limit` is determined by Allocator and is updated by Controller. When `request_limit` is a positive value, `request_limit` is the amount of requests that allowed to be handled per unit time. Whenever a request is handled, `request_limit` is subtracted by the size of the request. When the size of all handled requests is larger than `request_limit`, it means the traffic is saturated and the data access has to be blocked until next unit time. The size of all handled requests per unit time is recorded in order to judge whether tapdisk consumes minimum bandwidth it requested, incurring feedback that changes the status of SA.

```
//start of the bandwidth controller
if(request_limit != 0) // Bandwidth is limited of this domain
{
    if(request_size < request_limit) // enough resource for the request
        request_limit = request_limit - request_size;
    else
        Hang on the request and wait for the next resource allocation;
}

Call file system to answer this request;

request_sum += request_size; // to record the total size of requests in unit time
//end of one request process
```

Figure 3 The workflow of bandwidth controller

4.2. Monitor

Status Monitor supervises the status of tapdisk and sends bandwidth usage information to Allocator. The type of the status is determined by three of factors: 1) minimum bandwidth specified in the configuration file; 2) actual bandwidth usage recorded by Controller; 3) current status. First, Monitor attempts to retrieve instructions of Allocator from the message queue. If there were new instructions, it would change the status of tapdisk and set the value of request_limit in Controller accordingly. If there were no new instructions, it would compare the request_limit with the actual bandwidth usage. Then, it will send information to Allocator according to current status.

There are 3 possible status of tapdisk to indicate the usage of allocated resource: IDLE, UNDER and OVER. The meaning of each status is shown in Table 1. These 3 statuses are decided by the instructions from Controller. When there is no data access, the status of Monitor will be IDLE. When the demand of data access exceeds the allocation, the status would be changed to OVER, which means the virtual domain needs more resource to complete the workload.

Table 1 The status of tapdisk

Status	Describe
IDLE	No block device access
UNDER	Demand of accessing is less than allocated resource.
OVER	Allocated resource has been run over.

The request sent to Monitor by Controller includes the type field and the data field which has 5 types. The type NOWORK indicates no block devices access from tapdisk, thus it does not participate in bandwidth allocation and it leads to the status of IDLE of tapdisk. The NONE type illustrates that tapdisk is allocated with enough bandwidth and is working normally. The type RESERVE means that current bandwidth cannot meet requirement. The domain runs out the allocated resource and requests more. The type UNRESERVE, which is used when tapdisk devices access completed, means that bandwidth is no longer required. The type RELEASE indicates that bandwidth reservation surpasses actual bandwidth usage. As a result, guest domain can release the residual part.

4.3. Allocator

Bandwidth Allocator collects the status of tapdisks, respond to requests and allocate bandwidth globally. Whenever a tapdisk is created, the process id of new tapdisk along with status information will be appended to the list. Accordingly, tapdisk will notify Allocator to remove corresponding entry when it is about to exit. Allocator fetches messages, which are used to update the status of tapdisks, from message queue per second. Table 2 shows the status of tapdisk in Monitor. Compared to Monitor, the status change of tapdisk in Allocator is more complicated.

Table 2 the status of tapdisk in the bandwidth Allocator

STATUS	Description
IDLE	No block device access
RESERVED	Resource allocation is under reservation.
UNDER LIMITATION	Demand of allocation is over reservation and getting less.
OVER RESERVATION	Demand of allocation is over reservation and getting more.

After received requests, Allocator updates the status of corresponding tapdisk in the list and computes Δa_i and A_i . For UNDER domains in Monitor, their Δa_i would be min us. And the value is 20% of A_i . For OVER domains, their Δa_i would be 20% of A_i . Then, new A_i would be renewed by the algorithm in Section 3. Allocator initiates global allocation and dispatches instructions to Controller.

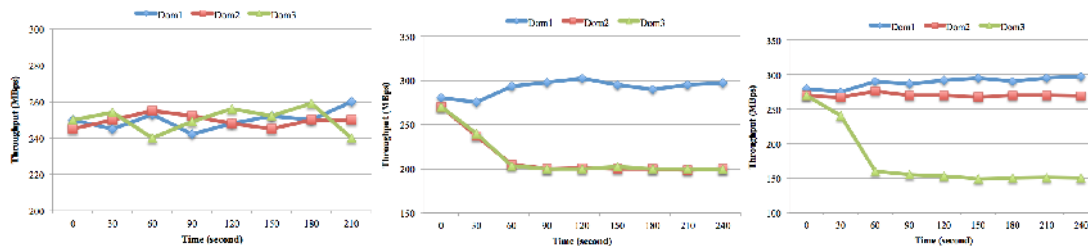
5. TEST RESULTS

In this section, we evaluate the allocation of resources in Xen through benchmarks. To avoid the effect by disks, we first did the evaluation in Ramdisk and then hard disk. In Section 5.1 and Section 5.2, the function of allocation is tested compared to the origin allocator in Xen. We implemented a prototype on Xen 3.4 running in Linux 2.6.26. We have preformed the fio storage benchmark on the system with Intel Xeon CPU E5540 (2.63 GHz *8), 16GB DDR3 memory, unmodified guest of Linux 2.6.26, and Xen 3.4.

5.1. Bandwidth Allocation

We test minimum bandwidth protection on three VMs visiting virtual block devices. The result is shown in Figure 4.

The effect of no bandwidth protection is shown as (a), that throughput resource is distributed to virtual domains according to the accessing request. All three VMs have almost identical bandwidth usage at approximately 250MBps. This results from the strategy of First-in-First-Serve, which is applied in Xen by default.



(a) Without bandwidth protection (b) Protect one block device (c) Protect two block devices

Figure 4 Bandwidth allocation test

When protecting the bandwidth of a VM, we set the minimum bandwidth of Dom1 to 280MBps, and then left Dom2 and Dom3 unreserved. At the beginning, three VMs fairly

shared bandwidth at approximately 260MBps. Then Allocator limited the bandwidth of Dom2 and Dom3 according to the limitation strategy, making more bandwidth available for Dom1. After that, the bandwidth of Dom1 is maintained at around 290MBps, whereas Dom1 and Dom2 fairly shared the residual bandwidth at about 200MBps. When protecting the bandwidth of multiple VMs, we set the minimum bandwidth of Dom1 to 290MBps, Dom2 to 270MBps and left Dom3 unprotected. At the beginning, Dom1 and Dom2 cannot fulfill their minimum bandwidth requirements. Thus, Allocator limited the bandwidth of Dom3 according to the strategy of reservation. Then, the demands of Dom2 and Dom3 would be met after the limitation of Dom3. At the same time, Allocator limited the bandwidth of Dom2 and increased the bandwidth of Dom1. As a result, the bandwidth of Dom1 is maintained at around 290MBps, Dom2 at 270MBps and Dom3 at 150MBps.

5.2. Weighted allocation

We also test the most important scenario, when virtual domains are running with different weights. Then a new VM is added, causing the fluctuation of available situation. The test result is demonstrated in Figure 5. In test 5(a), all the virtual domains have the same weight and have no reservation protections. In test 5(b), Dom2 and Dom3 have a same weight and Dom1 has a reservation of 300Mbps throughput.

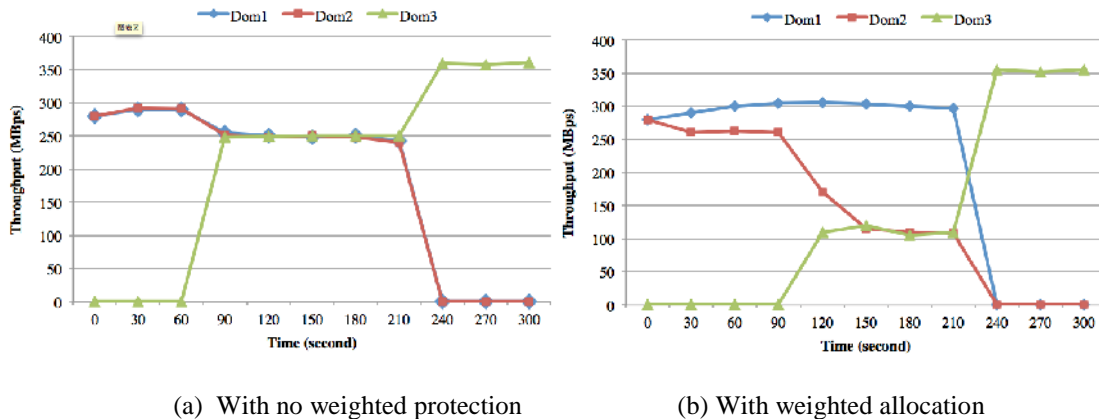


Figure 5 Weighted allocations in Allocator

Initially, we launched Dom1 and Dom2, then Dom3 after 90 seconds. After 2 minutes, virtual block device access of Dom1 and Dom2 ended and Dom3 continues running. In (a) before implemented bandwidth controller, Dom1 and Dom2 share equal bandwidth at 290MBps. After Dom3 was added, three VMs share the bandwidth, and each of them uses 250MBps. At the fourth minute, Dom1 and Dom2 finished their tasks and stopped block device accesses. Then Dom3 exclusively possess all the bandwidth resource and its bandwidth exceeds 350MBps.

When Allocator was applied in (b), we set the minimum bandwidth of Dom1 to 300MBps. As a result, Dom1 maintained its bandwidth at 300MBps at the first 90 seconds, and Allocator limited the bandwidth of Dom2 at 260MBps. After 90 seconds, Dom3 was added, leading to the decline of the other two VMs' bandwidth, resulting in the bandwidth of Dom1 below the

minimum bandwidth. Meanwhile, Allocator further allocated the bandwidth to meet the requirement of reservation. We can see that the bandwidth of Dom2 and Dom3 dropped to 110MBps and Dom1's bandwidth restored to above 300MBps. After the fourth minute, Dom1 and Dom2 stopped accessing virtual block devices, and Allocator cancelled the limitation. This leads to the rise of Dom3's bandwidth, exceeding 350MBps.

6. CONCLUSION

Allocation of resource is of great importance on the virtualization platform, which means proper allocation and scheduling can significantly improve the performance of the virtualized system. Although Xen has function of resources allocation, it cannot effectively make full use of resources and keep the quality of services in virtual domains. Therefore, we put forward a strategy of weighted max-min fairness allocation of virtual block devices to conduct the throughput allocation. Sharing incentive, initial irrelevant and Pareto efficiency are achieved to improve the performance of virtual machines. To implement the new algorithm in Xen, three modules as Controller, Monitor and Allocator are used on the visit path of Xen virtual block device. We have evaluated the algorithm and shown that it can lead to better allocation performance than the scheduler in use before.

ACKNOWLEDGEMENTS

This work is supported by the National Basic Research 973 Program of China under Grant No. 2011CB302301, 863 project 2009AA01A402, NSFC No.61025008, 60933002, 60873028, Changjiang innovative group of Education of China No. IRT0725.

REFERENCES

- [1] P. Barham, B. Dragovic, K. Fraser, etc. Xen and the art of virtualization. In Proceedings of 19th ACM Symposium on Operating Systems Principles, pp. 164-177, 2003.
- [2] Seetharami R. Seelam, Patricia J. Teller. Virtual I/O Scheduler: A Scheduler of Schedulers for Performance Virtualization. In Proceedings of ACM Virtual Execution Environments, pp. 105~115, 2007.
- [3] IBM Unstructured Information Management Architecture. <http://www.research.ibm.com/UIMA/>
- [4] INTEL CORPORATION. Intel Virtualization Technology Specification for the IA-32 Intel Architecture, April 2005.
- [5] A. Menon, A. Cox, and W. Zwaenepoel. Optimizing network virtualization in Xen. In Proceedings of the 2006 USENIX Annual Technical Conference. pp.15~28. 2006.
- [6] Kesavan, M., A. Gavrilovska and K. Schwan. On disk I/O scheduling in virtual machines. in Proceedings of the 2nd conference on I/O virtualization. 2010: USENIX Association.
- [7] Ongaro, D., A.L. Cox and S. Rixner. Scheduling I/O in virtual machine monitors. in Usenix International Conference On Virtual Execution Environments. 2008. Seattle,WA,USA: ACM.
- [8] Gulati, A., A. Merchant and P.J. Vaman, mclock:Handling Throughput Variability for Hypervisor IO Scheduling, in Proceedings of Operation System Designing and Implementation 2010. 2010.
- [9] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. In Proceedings of the 8th USENIX conference on Networked Systems Design and Implementation. USENIX Association, Berkeley, CA, USA, 2011.
- [10] Cherkasova, L., D. Gupta and A. Vahdat. When virtual is harder than real: resource allocation challenges in virtual machine based it environment. HP Laboratories: Palo Alto, CA, USA. 2007.

- [11] I.Stoica, H.Abdel-Wahab, K.Jeffay, S.Baruah,J. Gehrke, and G. Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In IEEE RTSS 96, 1996.
- [12] B.Caprita,W.C.Chan,J.Nieh,C.Stein,andH.Zheng.Group ratio round-robin: O(1) proportional share scheduling for uniprocessor and multiprocessor systems. In Proceedings of USENIX Annual Technical Conference, 2005.
- [13] P. Goyal, H. Vin, and H. Cheng. Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks. IEEE/ACM Transactions on Networking, 5(5):690–704, Oct. 1997.
- [14] Leandros Tassiulas, Saswati Sarkar. Maxmin Fair Scheduling in Wireless Ad Hoc Networks. In Proceedings of IEEE Journal on Selected Areas in Communications. VOL. 23, NO. 1, Jan 2005.
- [15] J. Axboe. Linux Block IO - Present and Future (Completely Fair Queueing). In Ottawa Linux Symposium 2004, pages 51-61, 2004.
- [16] Tin Tin Yee,Thinn Thu Naing. Pc-cluster based storage system architecture for cloud storage. International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.1, No.3, November 2011. pp115-126.
- [17] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. In proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC 2011. Association for Computing Machinery, USA. Oct 2011.
- [18] Wei Zhang, Hangwei Qian, Craig E. Wills et al. Agile resource management in a virtualized data center. In proceedings of the first joint WOSP/SIPEW international conference on Performance engineering. ACM New York, NY, USA. pp 129-140. 2010.