Spatial-Locality-Aware Virtual Storage Devices with Tangible QoS Expressions

Pei Yan, Song Jiang
Dept. of Electrical & Computer Engineering
Wayne State University
Detroit, MI 48202, USA
pyan@wayne.edu, sjiang@eng.wayne.edu

Abstract

Consolidated storage service receives its momentum in the building of various IT system infrastructures because of its cost efficiency, reliability, and maintainability. Meanwhile, only when users who run their applications on the consolidated storage system have a similar or better performance experience compared to those using direct-attached storage on each server can the storage design really be widely accepted. We propose a framework in which performance of servicing requests from one user is well isolated from that for the others in such a way that each user can be regarded as being allocated an independent virtual disk (VD) of performance as specified in its service-level agreement (SLA). Three SLA expressions that we believe are most tangible to users are supported in the spatial-locality-aware framework, which we name as rent-a-disk, latency-conscious, and throughput-conscious. Our proposed I/O scheduling algorithm for implementing the three types of VDs provides strong isolation, low interference, and high fidelity of performance for each VD if SLA is not violated. Our algorithm also enables different types of VDs to cooperate with each other to improve performance for each VD as well as for the entire physical storage.

1. Introduction

While data increasingly becomes the center of today's computing, more and more applications and services rely on high-performance and reliable storage systems to deliver their promised service qualities. This technical trend has put a high demand on the hardware and management investments, which may be beyond the resources available to a single division or department. Therefore, an increasingly common practice is to consolidate the storage resources in a single data center to simultaneously provide storage services through the high-speed network to participating users. An example is Amazon.com's Internet-wide storage service, Simple Storage Service or S3 [2]. The shared I/O service provided through the consolidated storage system promises high performance of data access, low amortized cost, and high reliability. However, this promise can become reality only when a user receives his or her expected performance from the consolidated storage infrastructure. There are several issues to address in ensuring that a particular user receives his or her expected I/O performance.

The first issue is how to take disk layout of requested data into account. This is important because the hard disk is such a storage device that random access can deliver an I/O performance one order of magnitude worse than sequential access on the same disk. Therefore, to meet the same user-specified I/O performance requirements or service-level agreement (SLA), in terms of either latency or throughput, the resources consumed can be vastly different for different data request patterns (random or sequential). This is because both latency and throughput are measured in the amount of data accessed, disregarding spatial locality of the data. Existing performance isolation policies are mostly derived from the bandwidth allocation algorithms designed in the networking domain and thus do not explicitly consider the unique performance characteristic of the hard disk. The consequence is that resource cannot be well provisioned by only knowing latency and/or bandwidth requirements in the SLA. In our design, we use disk service time that is actually spent, rather than the amount of data that is accessed, to allocate resource to a particular VD for a stronger performance isolation among VDs. In this way, spatial locality of requested

1-4244-0328-6/06/$20.00 ©2006 IEEE.

data only affects the latency or throughput it receives, rather than the amount of resource available to others.

The second issue is how a customer should express the performance requirements on his/her VDs in SLA. Existing policies usually ask a user of a VD to provide both latency and bandwidth requirements. These SLA expressions commonly assumed in prior studies have two drawbacks. First, they are not closely relevant to users' experiences with disks. If users' intention of using a VD is to replace his/her local physical disk, they would expect the VDs to produce performance similar or better than the physical disk with the understanding that actual latency or bandwidth varies with data spatial locality. Second, using both latency and throughput at the same time to specify SLA is usually unnecessary for users and limits the space for performance optimization for storage system designers. In our design, we customize three types of VDs according to different users' performance needs, each with its SLA expressions. Among them, *rent-a-disk* is used to simulate the performance of a physical disk in a tangible manner, *latency-conscious* and *throughput-conscious* VDs are designed for those users who care only latencies or only throughputs of their applications, respectively.

The third issue is how to make co-existing VDs of different types to mutually benefit from each other. While existing policies allocate spare disk resource to requests' streams, the targets of the allocation are usually chosen simply according to the request deadlines. While throughput-conscious VDs do not have pre-set deadlines for each individual request, they artificially impose evenly spaced deadlines to requests and strictly observe them. This practically takes out the opportunity of exploiting latency-insensibility of the throughput-conscious VDs to improve service quality for the latency-conscious VDs. In our design, we exploit the opportunity by treating throughput-conscious VDs as a reservoir so that excess disk resource can be accumulated and released afterward, if needed, to keep requests to the latency-conscious VDs from missing their deadlines.

2 Related Work

Many of existing QoS-based resource allocation for storage services are derived from algorithms for allocating bandwidth and latency in the networking area. To allocate bandwidth among traffic flows, these algorithms assign tags to requests in each flow based on their respective claimed bandwidths using either real time, such as virtual clock [15], or virtual time, such as WFQ [5], WF^2Q [3], SFQ [7], SCFQ [6].

Following the practice in the networking domain, some I/O scheduling strategies for storage service quality guarantee tag requests from flows of different QoS requirements with their deadlines (or called finish times), which are calculated from users-specified bandwidth and latency [9, 10, 14, 8]. In Stonehenge [9], each request flow is directed to a virtual disk (VD) and both required bandwidth (or throughput) and latency need to be specified for a VD. However, these two metrics are correlated. At a particular time, a VD is either throughput-bound or latency-bound. A feedback control mechanism is introduced to determine whether the physical storage is overloaded and is able to accept additional VDs by monitoring how the latency requirement is statistically met. In pClock [8], an additional metric is introduced to characterize a flow – burstiness – and the scheduling algorithm can guarantee that the latency requirement is always met if its request burstiness (or the number of pending in the bucket in the leaky bucket model) is within the burstiness bound and request arrival rate is less than a pre-set bound. In some other I/O scheduling schemes such as Fasade [11], users provide a curve that describes the required maximum latencies under different request arrival rates. In all of these approaches, users need to specify performance bounds using *effective* throughput and latency, as they do in the networking service. However, effective throughput and latency for a disk-based storage system are not only affected by arrival rate, which users can be well aware of, but also determined by spatial locality of the requested data, which is related to the data on-disk layout and users are hard, if not impossible, to know. In addition, as we know that a user can be interested in only one of the performance metrics for a particular application, leaving out one of the two constraints would bring more opportunities for performance optimization.

While hosting multiple virtual disks on one physical disk, the disk head has to move among disk regions belonging to different VDs. The movements represent the overhead associated with hosting multiple VDs in the same physical system. Argon [13] uses data prefetch and write-back to increase the granularity of service provided to the competing VDs so that the overhead can be reduced. Argon uses explicit disk time quanta to allocate disk resource to VDs. Our design takes a similar approach. Different from Argon, which focuses on the reduction of interference among multiple VDs, our work is concerned with how to provide conditioned performance guarantees to different request flows. Furthermore, Argon allocates sufficient amount buffer cache to a VD, if available, to support targeted disk efficiency.

3 Performance Guarantee for Virtual Disks using Spatial-locality-aware I/O Scheduling

To meet performance requirements, the system capacity demand associated with the requirements of a VD must be evaluated before a decision is made on the acceptance of the VDs. However, spatial locality of requested data, which is usually hard to predict beforehand, affects both effective throughput and effective latency, and consequently makes performance requirements presented in these metrics hard to be associated to their commensurate system capacity demands. Therefore, we use peak bandwidth, which is delivered under the fully sequential access pattern, to express performance requirement or to cap system capacity demands of effective latency/throughput requirements.

3.1 Three Types of VDs

We propose three types of VDs, named as *rent-a-disk*, *latency-conscious*, and *throughput-conscious*. Each type of disk provides a convenient method for users to specify tangible QoS requirements that correlate to their experienced/expected performance of dedicated DAS disks.

The rent-a-disk VD is used to simulate a physical disk whose peak bandwidth is specified. Once a rent-a-disk VD with a required peak bandwidth is accepted, we guarantee that it has an effective bandwidth that is commensurate to the bandwidth available on a dedicated physical disk with the same required peak bandwidth. We call the peak bandwidth peak equivalent of the effective bandwidth, which allows effective bandwidths associated with different spatial localities to be comparable. For users who replace their local disks with shared storage services, such as outsourcing I/O services to the third-party storage service providers, to run their programs, the rent-a-disk VD aims to retain performance characteristics of their dedicated disks.

The latency-conscious VD is designed for transaction-oriented applications, in which a latency value is specified to bound the duration from a request's arrival to its completion. While the latency requirement itself is hard to characterize system capacity demand, users need to specify a peak bandwidth to cap the VD's capacity demand. The latency bound is guaranteed only if a dedicated disk with the peak bandwidth does not violate the bound. In this way, a latency-conscious VD looks like a rent-a-disk VD whose request latencies are bounded. A difference from the rent-a-disk VD is that the latency-conscious VD provides additional opportunity for its performance optimization, through using spare system capacity or through the cooperation with the throughput-conscious VD.

The throughput-conscious VD is designed for applications that are only interested in average I/O throughput they receive or aggregate request service times, rather than instantaneous bandwidth or latency. Example applications include transferring of a large amount of data and background jobs such as file backup and data recovery. In the throughput-conscious VD, a throughput value is specified to bound the effective throughput averaged over the duration from the beginning of the flow to the current time. In addition, like in the latency-conscious VD, users need to specify a peak bandwidth to cap the VD's capacity demand. The effective throughput bound is guaranteed only if the peak equivalent of the effective throughput is not greater than the peak bandwidth.

3.2 The Scheduling Framework

In our proposed framework, all the three types of VDs are supported. In the design, there are two critical questions to answer, both of which are related to the spatial locality of requested data. First, what is the metric that should be used to quantify the allocations of storage system capacity to each VDs? Some existing strategies use number of requests serviced or number of bytes accessed as the metric. As an example, an allocation policy may ensure that 20% of the requests that are serviced are from the flow to a specific VD, or 20% bytes of data that are accessed in a time unit are from or to a specific VD. However, because the spatial locality of requested data can be non-uniform across VDs, this 20%s may lead to an allocation of system capacity to the VD that is way away from the intended percentage. We use peak bandwidth as the metric, which is independent of spatial locality. Therefore, if we allocation 20% of system's peak bandwidth to a VD, the VD will receive one fifth of the system's capacity. Second, what is the metric that should be used for accounting system resource a VD receives? For the same reason as mentioned before, if we use number of requests or bytes a VD has serviced for the accounting, a VD that is randomly accessed would obtain much more than its fair share of system capacity. Therefore, we use disk time that is consumed for servicing a request flow to its VD as the metric. While we know that disk time for servicing the random data is much larger than that for servicing the same amount of sequential data, the spatial locality is reflected in the metric. Therefore, a VD that receives 20% of total disk time receives 20% of system capacity, regardless how spatial locality varies

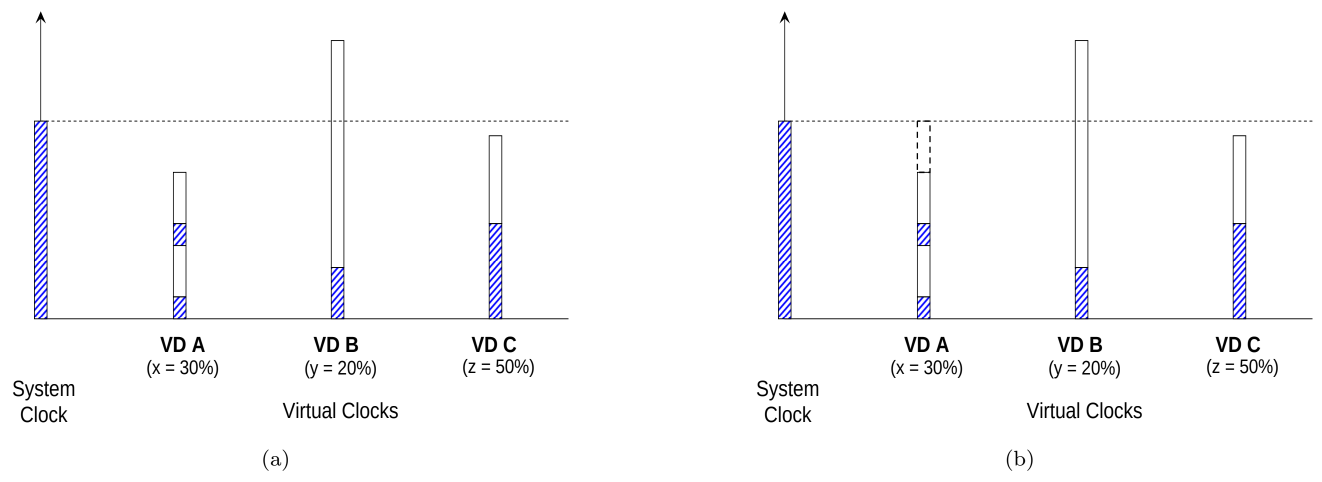

Figure 1. The I/O request scheduling in a storage system hosting three virtual disks (VDs). The three VDs, A, B, and C, are of rent-a-disk, latency-conscious, and throughput-conscious types. In the figure the dashed line shows the current system clock time. The top of the bars for virtual clocks shows the current virtual clock times. The speed differences between the system clock and virtual clocks reveal how performance requirements of a VD is met in the shared storage system. Thus, the differences serve as the basis of request scheduling. In each of the bars, the shaded areas represent the real disk times that are spent, and the unshaded areas exist due to the contractual peak bandwidth less than 100% of system peak bandwidth. Figure (b) shows that (1) virtual time of VD A is increased to the current system time when it is selected for scheduling but does not have pending requests. (2) VD B has requests that have missed or are missing their latency bounds. In the case where VD C's current average throughput is larger than its throughput bound, disk times are allocated to VD B even if VD C's virtual clock time is less than the system time.

across the VDs.

Suppose that we have three VDs: a rent-a-disk VD A, a latency-conscious VD B, and a throughput-conscious VD C. The latency bound of VD B is b seconds, and the throughput bound of VD C is c IO requests per second (if each I/O request is for approximately same amount of data). Moreover, we assume that the required peak bandwidths of there VDs account for x, y, and z (all in percentage) of the system capacity T, also in terms of peak bandwidth. To quantify the consumption of a VD's allocated capacity, we introduce the concept of virtual time of a VD, which is t/f, where t is disk time received by the VD, and f is x, y, or z for the three types of VDs, respectively. If the entire system is dedicated for providing services to a VD for disk time t, equivalently the VD has used virtual time t/f to service its requests. Accordingly, in the next t/f - t time, no requests from the VD need to be serviced in order to keep their performance requirements for any of there types of disks. This is because performance requirements are defined against a dedicated disk with the fT peak bandwidth.

We set up a system clock for the entire system, which generally advances as disk times are spent. Each VD also has its own clock, called virtual clock. If a VD's current virtual clock time is not greater than the current system time, which indicates that the VD does not exceed its contractual system capacity and has the right to receive disk time servicing its requests. Accordingly, the request scheduler selects requests from VDs whose virtual clock times are less than the system clock times.[1] Once such a request is serviced, the disk service time, t, is added to the system time to advance the clock, and t/f is added to virtual time of the request's VD to advance its virtual clock. However, if the VD that is selected does not have pending requests, we proceed to choose next qualified VD. Meanwhile, we advance the VD's virtual clock to the current system time to ensure that unused allocation of disk time cannot be saved as credit for future uses. Otherwise, the saved credits would allow the VD to claim more than its contractual system capacity. If virtual clock times of all VDs are larger than the system clock time, which indicates that the system has spare capacity, we uses our coordinated system capacity allocation policy, which will be described later, to choose a VD that has pending requests, to the spare disk time. Note that the disk time does not add to the system time or the VD's virtual time, because a VD's reception of bonus system capacity should not make it less likely to receive future disk times or make other VDs more favorable in receiving future disk times.

[1]The priority of the selection is given to those VDs that have large gaps between the two clock times.

Figure 1 illustrates a storage system hosting the three types of VDs, A, B, and C, whose required peak bandwidth percentages, x, y, and z, are 30%, 20%, 50%, respectively.

While users of the latency-conscious disk usually have difficulty in deciding an appropriate peak bandwidth, we use a coordinated system capacity allocation policy to assist the latency-conscious VDs to fulfill its latency bound. When the system has spare capacity and there are latency-conscious VDs, we allocate the disk times to the VDs, with a priority to the VDs that have the larger number of requests that have missed or are missing their latency bounds. If there are non-latency-conscious VDs, or latency-conscious VDs do not have pending requests, the spare system capacity goes to the throughput-conscious VDs, instead of the rent-a-disk VDs. This is because extra services that throughput-conscious VDs receive can be accumulated into their performance metric – current average throughput, and be released to assist latency-conscious VDs afterward, if there is such a need, without compromising their performance requirements. Furthermore, we may consider to overdraw a small percentage of contractual system capacity belonging to throughput-conscious VDs if we find that requests of latency-conscious VDs are missing their bounds. This percentage is called throughput deviation tolerance, denoted as *tolerance*. In the overdrawing operation, we make sure that at any time current average throughout is not less than $(1 - tolerance)$ of the contractual throughout. In this way, throughput-conscious VDs actually provide a cushion in the allocation of system capacity to enable coordination for a better service quality.

4 Performance Evaluation

4.1 Experimental Settings

We built a trace-driven simulator to extensively evaluate the performance of our scheduling policy. In the simulator, we use DiskSim3.0 [1] to simulate the disk systems, and other simulator components are responsible to receive, enqueue, and dispatch requests from multiple flows to their respective VDs in an order that is determined by our scheduling policy. Because we are mainly concerned with performance implication of I/O scheduling, we do not consider the effect of caching or prefetching. Alternatively, we can view the simulation is carried out at the I/O driver. DiskSim is a disk simulator that faithfully captures many details of a disk system and has been validated to be highly accurate. In the evaluation, we use two disk models that have been validated: SEAGATE_ST32171W with 7200 RPMs and its average seek time of 1.943 ms, which is referred as slow disk in the rest of the paper, and SEAGATE_ST3450IN with 10033 RPMs and its average seek time of 0.636 ms, referred as fast disk. In our experiments, both are used as single-disk system. In the synthetic traces, the request size is 512Bytes. In the generation of traces of different spatial locality, we use the probability of adjacency of the data requested in any two consecutive requests in the trace to quantify spatial locality. Thus, a 0% spatial locality means no any two consecutive requests are for adjacent data, and 100% spatial locality means the entire trace is a sequence of requests for fully sequential data.

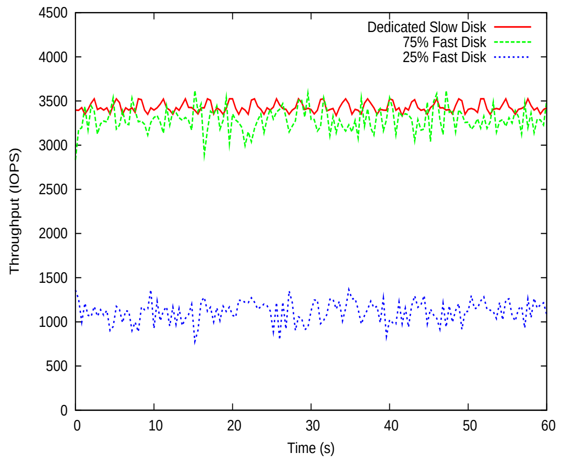

Figure 2. Throughputs of sequential flows to two rent-a-disk VDs.

4.2 Performance Guarantee of rent-a-disk VDs

Let us first examine performance guarantee of rent-a-disk VDs. We use two single disks in the experiment. Because the peak bandwidth of the slow and fast are around 3410 and 4500 sequential IO operations per second, or IOPS, respectively, i.e., the former is about 75% of the latter. We create two rent-a-disk VDs on the fast disk. one has 75% of peak bandwidth of the fast disk, and the other has 25% of the peak bandwidth. They are referred as 75% VD and 25% VD thereafter, respectively. The 75% VD is supposed to emulate the performance of a dedicated physical disk (the slow disk) on its host physical disk (the fast disk).

We feed simultaneously two sequential requests flows into the system, each to one VD. Both flows have sufficiently high arrival rates, so that each VD is saturated. Figure 2 shows the throughputs of the two flows, or the bandwidths of the two VDs as the flows of requests are serviced on the fast disk. As we can see that, the 75% VD has a bandwidth very close to that of the slow disk. Meanwhile, the 25% VD has an IOPS of about

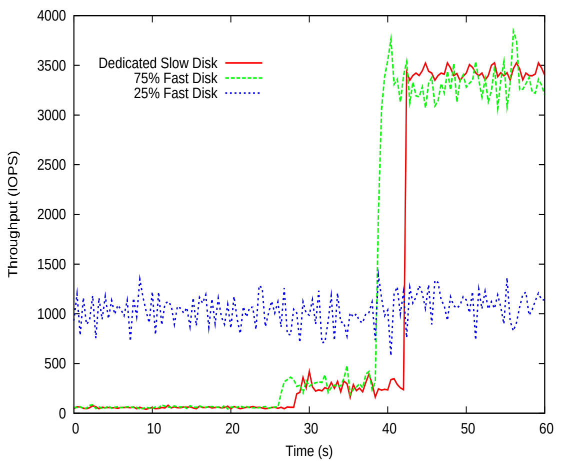

Figure 3. Throughputs of two request flows to two rent-a-disk VDs, one is of varying spatial locality and the other is fully sequential.

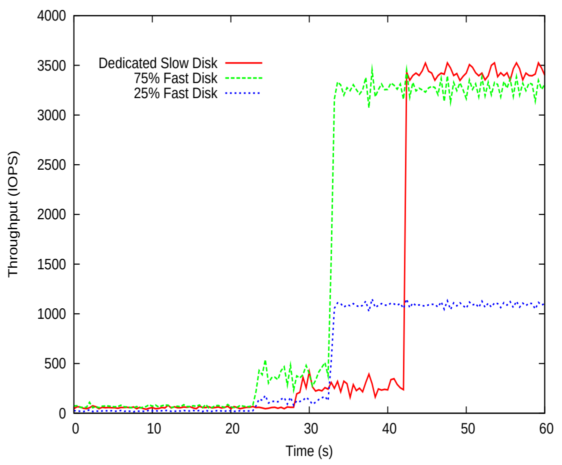

Figure 4. Experimenting with the I/O scheduler that is based on number of requests.

1100, or 24% of the peak bandwidth of the fast disk, which is an accurate emulation of a 25% VD. We have two additional observations. One is that both VDs have bandwidths that are a little bit lower than their required ones. This is due to the interference among multiple VDs on a physical disk. The other observation is that the variations of the VDs' bandwidth are larger than that of the dedicated slow disk. This is also due to the co-existence of multiple VDs on one physical disk and only one VD is serviced at a time. When one VD is serviced, it gets the disk's full bandwidth and the other receives zero. For the same reason, we see that the instantaneous bandwidths of the two VDs are complemented with each other.

To observe how spatial locality impacts the allocation of system capacity, we change the spatial locality of the 75% VD from 35% to 85% at 0.4 second, and then to 100% at 1.33 second. The results are shown in Figure 3. Apparently increasing spatial locality significantly improves effective bandwidth of the 75% disk. There are delays between the change of locality and increase of effective bandwidth, because newly arriving requests are enqueued at the queue tail, it takes time for the requests to move to the queue head and be scheduled for disk services when there is a long queue. We see that the curve for 75% disk matches that for the dedicated slow disk receiving the same flow except that the increase of bandwidth for the 75% VD happens earlier than that for the slow disk. The delay is caused by the smaller seek times for the fast disk, which makes the 75% VD have smaller average latencies and thus more efficient in servicing random requests than the slow disks. As the figure shows, the effective bandwidths of the 25% VD are almost not affected by the changing spatial locality for the 75% VD, which suggests that the two VDs are well isolated.

Figure 4 shows the results of an experiment that is set up in the same way as that with Figure 3, except that the I/O scheduler chooses requests for service according to number of requests serviced by each VD, rather than the disk times used. Therefore, approximately 75% of all the requests that have been serviced are to the 75% VD. From Figure 4 we can see that the bandwidth of the 25% VD is heavily influenced by the changing spatial locality at the 75% VD. When the locality is weak, or the data access is random, the same number of requests serviced actually consume more disk capacity, and leave less capacity for the 25% VD. As the figure shows, the 25% VD can receive its contractual 25% peak bandwidth of the system only when the spatial locality of the 75% VD is 100%. In the other time, its effective bandwidths are significantly lower and the performance agreement is seriously violated.

We further examine the performance of rend-a-disk VDs using a real-world trace *openmail* and the results are shown in Figure 5. *openmail* was collected on a production e-mail system running the HP OpenMail application for 25,700 users, 9,800 of whom were active during the hour-long trace. The system has 6 HP 9000 K580 servers running HP-UX 10.20. The size of the data set accessed by all six clients was 18.6G. In this experiment, we randomly select an *openmail* client to feed into the 75% fast disk while supporting another fully sequential trace on the 25% fast disk. In Figure 5, it can be seen that the throughput of the 75% fast disk approximates very well the throughput received when running the same *openmail* trace alone on the dedicated slow disk. Running this *openmail* trace does

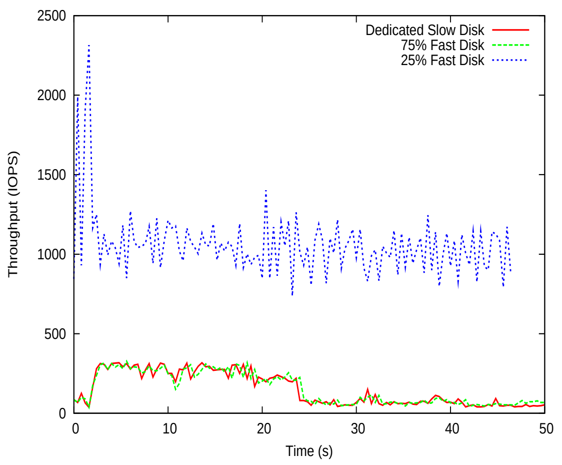

Figure 5. Performance of the rent-a-disk VDs using *openmail*.

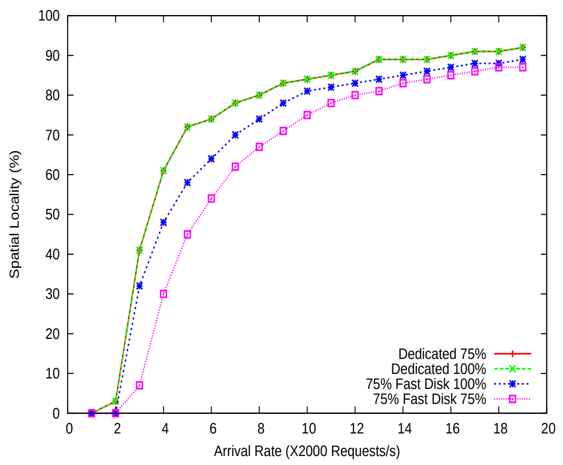

Figure 6. The iso-percentage-latency curves with different arrival rates and spatial localities. Note that the two curves for dedicated disks are overlapped in the figure. The last percentage value in each legend indicates the percentage of requests that have latencies no more than the given value.

not affect the performance of the 25% fast disk — the bandwidth of the 25% fast disk remains at around 1000 IOPS (about 25% of the overall bandwidth of the fast disk). The random access pattern of *openmail* only makes its own throughput low due to the performance isolation mechanism provided by our IO scheduler.

4.3 The Latencies of VDs with Varying Arrival Rates and Spatial Locality

While we have shown that in our design a VD can provide an effective bandwidth that is close to the required bandwidth, we now show how latencies of VDs are compared with those in the dedicated disk when we vary arrival rate and spatial locality of flows to VDs. The latency, or response time, refers to the time from the arrival of a request to the completion of service of the request, including queuing time and disk service time. When we increase arrival rate of a flow, we must improve its spatial locality so as to keep the latency constant. Alternatively, we must reduce arrival rate if spatial locality is reduced to keep the latency constant. To characterize the relationship, we propose the iso-percentage-latency curve that shows pairs of arrival rate and spatial locality values that keep a given percentage of requests experience latencies that are equal to or less than a given latency. Figure 6 shows the iso-percentage-latency curves with the given percentage of 75% and 100% respectively, and the given latency of 0.3 second on the dedicated slow disk and on the 75% VD (the other VD is the 25% VD). While the area above a curve represents the combinations of arrival rate and spatial locality that produce a latency less than 0.3 second, the figure clearly shows that the 75% fast disk provides a much larger space to have smaller disk seek time. This is because the fast disk has a smaller latency. When requests from a VD are serviced with a shorter time, all the requests receive smaller latencies.

4.4 Cooperation between Latency-Conscious VD and Throughput-Conscious VD

To investigate the impact of the cooperation of the latency-conscious VD and the throughput-conscious VD on the improvement of latencies of the latency-conscious VD, we choose to run a flow whose arrival rates are shown in Figure 7 on the 75% VD. On the 25% VD we test three cases, in which a rent-a-disk VD, a throughput-conscious VD of 0% throughput deviation tolerance, and a throughput-conscious VD of 5% deviation tolerance are used, respectively. Figure 8 shows the cumulative distribution curves (CDF) of the latencies of requests to the latency-conscious VD. The curve with the rent-a-disk VD exhibits the worst latency distribution, because the VD is not allowed to release the extra system capacity it receives when the latency-conscious VD has an arrival rate of 0. Comparatively, the throughput-conscious VD provides the opportunity for the latency-conscious VD to save its unused system capacity through the throughput-conscious disk when its arrival rate is 0 and the saved capacity is released from the throughput-conscious disk for its use when its arrival rate is surged again. It is understandable that a throughput-conscious VD with a deviation tolerance can provide an even larger cushioning space to accommodate a surged arrival rate of the flow and keep low latencies.

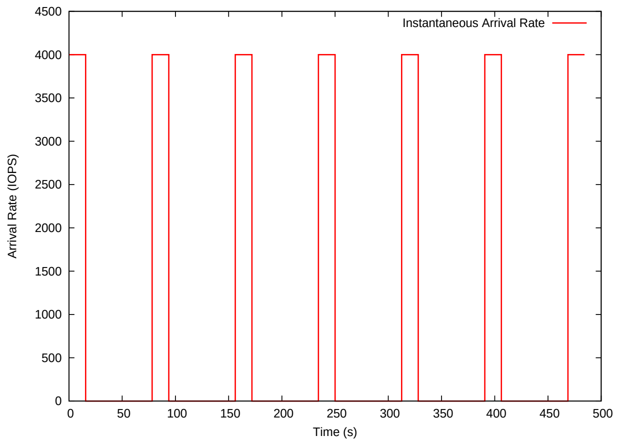

Figure 7. Varying arrival rates of the flow to the 75% VD.

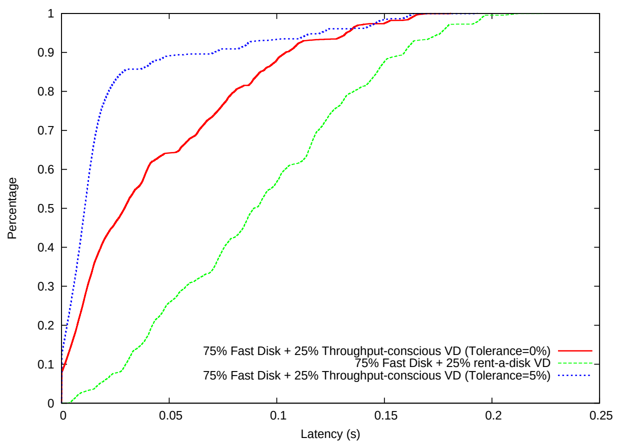

Figure 8. CDF curves for latencies of the latency-conscious VD that is coordinated with three different VDs.

5 Conclusions

In the paper we identify three tangible expressions for I/O performance requirements to allow users of a consolidated storage system to conveniently specify their performance requirements. Our proposed I/O scheduling framework can provide a strong performance isolation and low interference among VDs hosted on the same physical disk systems, which is demonstrated in our experiments.

In the future, we would like to address several issues that are not well covered in the paper. First, the interference among multiple VDs can be further reduced by grouping more requests in the same flow to service together. The negative impact of the optimization is that latency can become more dynamic. The trade-off between these two effects should be studied. Second, we plan to configure more complex storage system consisting of different disk arrays to evaluate our design. Third, we will collect and run real-world traces for a more comprehensive evaluation. We will also implement the framework in a representative storage system so that more technical issues can be addressed.

References

[1] John S. Bucy, Gregory R. Ganger The DiskSim Simulation Environment Version 3.0 Reference Manual http://reports-archive.adm.cs.cmu.edu/anon/2003/CMU-CS-03-102.pdf.

[2] Amazon Simple Storage Service (Amazon S3) http://www.amazon.com/gp/browse.html?node=16427261.

[3] J. C. R. Bennett and H. Zhang. Wf^2q: Worst-case fair weighted fair queueing. In *Proc. of INFOCOM*, 1996.

[4] R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communications*, 13(6), 1995.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. *Journal of Internetworking Research and Experience*, 1(1), 1990.

[6] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proc. of INFOCOM*, 1994.

[7] P. Goyal, H. M. Vin, and H. Cheng. Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Trans. Netw.*, 5(5), 1997.

[8] A. Gulati, A. Merchant, and P. J. Varman. pclock: an arrival curve based approach for qos guarantees in shared storage systems. In *Proc. of 2007 ACM SIGMETRICS Conference*, 2007.

[9] L. Huang, G. Peng, and T. Chiueh. Mutli-dimensional storage virtualization. In *Proc. of 2004 ACM SIGMETRICS Conference*, 2004.

[10] W. Jin, J. S. Chase, and J. Kaur. Interposed proportional sharing for a storage service utility. In *Proc. of 2007 ACM SIGMETRICS Conference*, 2004.

[11] C. Lumb, A. Merchant, and G. Alvarezg. Facade: virtual storage devices with performance guarantees. In *Proc. of the 2003 USENIX Annual Technical Conference*, 2003.

[12] H. Sariowan, R. L. Cruz, and G. Polyzos. Scheduling for quality of service guarantees via service curves. In *Proc. of the 4th International Conference on Computer Communications and Networks*, 1995.

[13] M. Wachs, M. Abd-El-Malek, E. Thereska, and G. R. Ganger. Argon: performance insulation for shared storage servers. In *Proc. of FAST '07*, 2007.

[14] J. Zhang, A. Sivasubramaniam, Q. Wang, A. Riska, and E. Riedel. Storage performance virtualization via throughput and latency control. In *Proc. of MASCOTS'05*, 2005.

[15] L. Zhang. Virtualclock: a new traffic control algorithm for packet-switched networks. *ACM Trans. Comput. Syst.*, 9(2), 1991.