

Multi-Dimensional Storage Virtualization *

Lan Huang[†]
650 Harry Rd.
IBM Almaden Research Center
San Jose, CA 95120, USA
lanhuang@us.ibm.com

Gang Peng, Tzi-cker Chiueh
Department of Computer Science
State University of New York
Stony Brook, NY 11790, USA
gpeng,chiueh@cs.sunysb.edu

ABSTRACT

Most state-of-the-art commercial storage virtualization systems focus only on one particular storage attribute, capacity. This paper describes the design, implementation and evaluation of a *multi-dimensional storage virtualization* system called Stonehenge, which is able to virtualize a cluster-based physical storage system along multiple dimensions, including bandwidth, capacity, and latency. As a result, Stonehenge is able to multiplex multiple virtual disks, each with a distinct bandwidth, capacity, and latency attribute, on a single physical storage system as if they are separate physical disks. A key enabling technology for Stonehenge is an efficiency-aware real-time disk scheduling algorithm called dual-queue disk scheduling, which maximizes disk utilization efficiency while providing Quality of Service (QoS) guarantees. To optimize disk utilization efficiency, Stonehenge exploits run-time measurements extensively, for admission control, computing latency-derived bandwidth requirement, and predicting disk service time.

Categories and Subject Descriptors

D.4.1 [Operating Systems]: Process Management; D.4.2 [Operating Systems]: Storage Management ; D.4.8 [Operating Systems]: Performance—*Measurements*

General Terms

Algorithms, Management, Design, Performance

*This research is supported by NSF awards ACI-0083497, CCF-0342556, ACI-0234281, SCI-0401777, USENIX student research grants, as well as fundings from Reuters Information Technology Inc., Computer Associates Inc., National Institute of Standards and Technologies, Siemens, and Rether Networks Inc.

[†]The author performed the work for this paper when she was a Ph.D student at SUNY Stony Brook.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMETRICS/Performance'04, June 12–16, 2004, New York, NY, USA.
Copyright 2004 ACM 1-58113-873-3/04/0006 ...\$5.00.

Keywords

Storage Virtualization, Quality of Service

1. INTRODUCTION

With the advent of storage area network (SAN) and network attached storage (NAS) technology, more enterprise servers are built based on the assumption that a separate storage server provides low-level disk access service. In large organizations, there are many different enterprise servers that support different business processes, such as Web servers, application servers, or database servers, which may have very different performance requirements on their backend storage server. While consolidating these backend storage servers into a single physical storage system that is under the control of one administrative team makes economic sense, sophisticated resource allocation and scheduling technology is required to effectively isolate these logical storage servers as if they are separate physical storage servers.

Storage virtualization refers to the technology that allows creation of a set of logical storage devices from a single physical storage resource. Each such logical or virtual storage device could then serve as a backend storage server for a separate enterprise server for a particular business function. Commercial storage virtualization systems are rather limited because they can virtualize storage capacity. They may also bundle multiple logical units for higher aggregated I/O rates. However, from the standpoint of storage clients or enterprise servers, the virtual storage devices are desired to be as tangible as physical disks. This requirement motivates the development of *multi-dimensional storage virtualization* technology. This paper describes the design, implementation and evaluation of a cluster-based multi-dimensional storage virtualization system called Stonehenge, which aims to virtualize efficiently any standard attribute associated with a physical disk, such as bandwidth, latency, availability, etc.

The key enabling technology for multi-dimensional storage virtualization is the disk bandwidth guarantee, because various disk performance requirements can be readily translated into bandwidth requirements. Although there is a large body of research on resource scheduling techniques for QoS guarantees, they cannot be directly applied to the storage virtualization problem because previous efforts focused mainly on streaming workloads, such as video playback. In contrast, the disk access workloads in enterprise environments are much more complicated. As a result, it becomes challenging to calculate a disk system's effective bandwidth

due to unpredictable seek and rotational overhead. Moreover, skews in disk access loads also complicate the calculation of effective disk bandwidth. First, a physical disk’s effective bandwidth is workload dependent, because different disk access sequences lead to different seek and rotational overhead. This very fact also complicates the support for certain QoS metrics such as delay bound. Second, physical disk resource is not necessarily additive, unlike CPUs or network links.

To address these issues, Stonehenge applies run-time measurements extensively throughout the entire disk resource allocation and scheduling process, including admission control, latency-derived bandwidth requirement computation, and disk service time prediction. In addition, Stonehenge features a novel two-level disk resource scheduler that couples a standard Virtual Clock (VC) request scheduler with a dual-queue physical disk scheduler. The resulting disk scheduler, called CSCAN-based Virtual Clock scheduler (CVC) guarantees the reserved disk bandwidth and latency at run time. In addition, the CVC scheduler is able to allocate spare bandwidth proportional to the reservations of the participating virtual disks. A major goal of this paper is to present the effectiveness of the CVC scheduler and the benefits of measurement-based disk resource allocation and scheduling, based on a trace-driven study on a working Stonehenge prototype.

The rest of the paper is organized as follows. Section 2 reviews previous research efforts. Section 3 presents the overall system architecture of the proposed multi-dimensional storage virtualization system, Stonehenge. Section 4 describes the details of the CVC disk scheduling algorithm and measurement-based admission control. Section 5 presents the results of a detailed performance evaluation study on the Stonehenge prototype and their analysis. Section 6 concludes the paper with its major contributions.

2. RELATED WORK

Storage management has attracted great research attention in recent years. Minerva [1] is an automated resource provisioning tool that takes as inputs declarative specifications of performance requirements, workload statistics and storage device characteristics, and generates storage system designs by solving the storage configuration problem as a multi-constraint optimization problem. Hippodrome [3, 4] can further migrate data between storage configurations determined at run time. All these storage management projects focus on storage system configuration and do not pay much attention to run-time disk scheduling. Their model takes I/O rate but not I/O latency as QoS specifications. One challenge that Stonehenge addresses is to convert I/O latency to I/O rates in disk scheduling domain.

Façade [13] has a similar design goal to Stonehenge. However, its underlying implementation is quite different from Stonehenge. First, because Façade uses EDF (earliest deadline first) as the disk scheduling algorithm, it is difficult to correlate delay bound guarantees with bandwidth resource requirements. As a result, it does not include an admission control algorithm. Second, the only mechanism in Façade to improve disk utilization efficiency is to adjust disk request queue size. It is less flexible and less accurate compared with Stonehenge’s CVC scheduler, which can make the best of the slacks available in QoS-driven scheduling while provably satisfying all the performance guarantees.

A fundamental problem in the design of QoS guaranteed systems is how to leverage dynamic workload patterns to reduce the amount of resource reservation while providing the QoS guarantees at a high probability. This problem is particularly acute when the guaranteed QoS attribute is delay bound. Breslau et al. [5] compared the effectiveness of measurement-based admission control algorithms [12] for wide-area network connections. The basic approach of these algorithms is to measure the system and load parameters that affect the packet delays, form a statistical model of these parameters dynamically, and eventually apply these models in the admission control process to avoid resource over-reservation. Urgaonkar et al. [17] studied resource over-booking in shared hosting platforms for CPU and network resources to achieve high resource utilization while guaranteeing QoS. To overbook resources in a controlled fashion, they do capacity profiling (either CPU or network bandwidth). In comparison, Stonehenge focuses more on request latency profiling, as we need to guarantee per request latency. Furthermore, Stonehenge profiles the *aggregate* of applications at run time, to better exploit the multiplexing effect of multiple streams, while [17] utilizes off-line profiling of each *individual* application. Online profiling is particularly important for storage systems, whose disk access time is workload dependent. During admission control, in [17], it multiplies the worst case resource requirement of an application by probability with which the resource requirement needs to be satisfied, and uses the result as an estimate for the long term resource requirement of the application. When the resource requirement deviation of an application is high, this may overestimate the resource requirement of the application. In Stonehenge, the admission control utilizes run-time measurement feedback to adjust resource reservations for virtual disks constantly, and therefore can correct the potential resource overestimate made in the previous rounds.

Vin et al. [18] discussed statistical admission control algorithms for media servers, and found that three times the number of streams can be admitted compared to the deterministic approach, if up to 3% of the playback cycles are allowed to overflow. The statistical measurement-based admission control algorithms for media servers rely on run-time statistics, such as mean, standard deviation, etc., because the assumption is that the workloads are largely predictable. In contrast, the measurement-based admission control algorithm used in Stonehenge is specifically designed for storage systems that service disk access workloads which do not have as predictable a pattern as media stream playback. As a result, the input loads are expected to be more fluctuating and irregular. Therefore a more detailed and accurate workload model is required to capture the statistical properties of these loads. Stonehenge maintains probability distribution functions to summarize the observed resource usage patterns in latency and throughput, and uses them to exploit statistical multiplexing and reduce the physical resource reservation requirement.

Another fundamental problem in QoS-guaranteed systems design is to how to balance the potentially contradicting goals of optimizing system resource utilization efficiency and satisfying QoS requirements. Cello [15] combines a class-independent scheduler with a set of class specific schedulers. Two time scales are considered in the two levels of the framework to allocate disk bandwidth: coarse-grain allocation of bandwidth to application classes and fine-grained band-

width allocation to optimize disk utilization efficiency. Wijayarathne and Reddy [19] present a similar two-level scheduler. Bruno et al. [6] described a QoS-aware disk scheduler to implement the weighted fair queuing (WFQ) algorithm [14], which optimizes disk resource utilization by servicing one batch of high-priority requests at a time. Compared to previous utilization efficiency-aware, QoS-guaranteed disk schedulers, the CVC scheduler can achieve better disk resource utilization efficiency without breaking QoS guarantees because it does not limit itself to a fixed batch and thus could exploit as much “slack” as is available in the QoS schedule.

3. THE STONEHENGE SYSTEM

3.1 Storage Virtualization Model

Given a physical storage resource, Stonehenge partitions it into multiple *virtual disks*, each with a unique set of disk performance attributes. The design goal of Stonehenge is to allow users to attach to a virtual disk any disk attributes that are traditionally associated with a physical disk, e.g., bandwidth, access delay, capacity, etc. Compared with existing storage virtualization systems, Stonehenge takes a step forward with this ability to virtualize a storage resource along multiple dimensions. In Stonehenge, a virtual disk can be specified in terms of the following attributes:

- Availability (A): The availability of a virtual disk is expressed in terms of the degree of replication. It can be remote or local replication.
- Bandwidth (B): The bandwidth of a virtual disk represents the number of disk access requests per unit time that the virtual disk can support.
- Capacity (C): The capacity of a virtual disk is the total amount of user data that could be put in the virtual disk.
- Delay (D): The delay of a virtual disk is the worst-case end-to-end delay that a disk access request targeted at the virtual disk could experience.
- Elasticity (E): The delay and bandwidth guarantees of a virtual disk could be probabilistic (specified with a probability) rather than deterministic.

Given a virtual disk specification quintuplet $\langle A, B, C, D, E \rangle$, Stonehenge first converts it to A non-replicating virtual disks, each with the specification $\langle 1, B, C, D, E \rangle$. Note that the prototype discussed in this paper does not consider A parameter and focuses on B, C, D, E . For a given QoS-aware disk request scheduler, there is a well-defined function that correlates a virtual disk’s bandwidth reservation and its access requests’ worst-case delay bound. Let this function be $F(\cdot)$, which computes the amount of bandwidth reservation required (B_{delay}) in order to achieve a certain delay bound D . Given $B_{delay} = F(D)$, Stonehenge can then further reduce each virtual disk specification to $\langle 1, \max(B, B_{delay}), C, \infty, E \rangle$. We will discuss about the correlation function F in section 4.2.

3.2 Virtual to Physical Disk Resource Mapping

Stonehenge maps a virtual disk $\langle B, C, D, E \rangle$ to one or a set of physical storage servers. A storage server contains a data processing unit, a disk controller and a disk

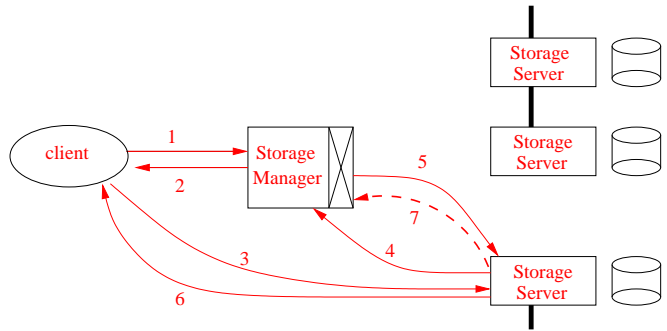


Figure 1: Data flow of the real-time scheduling in Stonehenge storage cluster. Stonehenge consists of a centralized management server and a set of storage servers that are connected through a Gigabit Ethernet network.

array. In the rest of the paper, we use disk array and storage server interchangeably. Stonehenge assumes that virtual disk reservations come in at different times. It is not necessary that all the virtual disks are mapped to the physical storage at once. The mapping procedure can be formulated as a multi-dimension bin packing problem, which is known to be NP complete. One heuristic algorithm to solve this problem is Toyoda weighted bin packing algorithm [16].

A virtual disk in Stonehenge can overflow the throughput or capacity capability in a single physical storage server. In these cases, it has to be assigned to multiple servers. A virtual disk with (B, C) requirements can be fulfilled by N striped partial virtual disks with $(f \times B/N, C/N)$, where f is a *leeway factor* to absorb the load imbalance. In the striped case, the total “read” and “write” bandwidth are the same. Each partial disk is admitted using the mapping heuristics and the virtual disk is admitted only if all the component partial disks can be admitted. It can also be fulfilled by mirrored partial virtual disks. We focus on the symmetric case of striped layout, and leave the mirrored case for future study.

3.3 System Architecture

Figure 1 shows the Stonehenge architecture. Stonehenge is a cluster-based iSCSI storage system that consists of a storage manager and a set of storage server nodes that are connected through a Gigabit Ethernet network. The central manager server takes a virtual disk specification as input and performs admission control to decide whether the virtual disk can be admitted, and if so, how to allocate physical disk resources to satisfy its QoS requirements. In the current Stonehenge implementation, the management server does not perform load balancing since we focus on striped mapping only. Potentially it can do so for a virtual disk with mirrored mapping. For scalability and fault tolerance, the management server itself may be implemented as a cluster. Clients are assumed to have the file layout information, and therefore decide which server has the target data without consulting with the central manager. Stonehenge chooses to centralize the load balancing and request deadline calculation function on a single management server because it simplifies the implementation, and because it allows each storage server node to be more modular. Otherwise metadata information will be spread on multi-parties in Stone-

hence, complicating management. It may also have negative impact on system performance. On the other hand, it is the storage server nodes that directly interact with Stonehenge clients through an iSCSI interface, thus freeing the central management server from the data path. However, there is a drawback of Stonehenge’s architecture: Each request needs to incur an additional round-trip delay for commands passing between a storage server node and the central manager. This is the price we have to pay to avoid data traffic going through one central point: the management server. Also, the central manager can potentially be a performance bottleneck. For our current testbed, it is not an issue. More details can be found in Section 5.1.

Stonehenge uses a two-level disk scheduler architecture. As shown in Figure 1, a Stonehenge client first contacts the storage manager for virtual disk layout information (step 1, 2). Then it sends a disk access request to a storage server node (step 3), which relays it to the central manager (step 4). Note that only the disk commands are relayed, not the actual read/write data. Hence the central manager is not in the data path. The central management server uses a virtual clock algorithm to compute a “deadline” for the request. This algorithm determines the order in which the requests associated with different virtual disks are processed in such a way that each virtual disk’s QoS requirement is satisfied. The tagged disk requests are sent to the storage server for service (step 5). The results are replied back to the clients directly (step 6). As disk access requests are dispatched and queued at individual storage server node, a separate CVC disk scheduler is used to decide the actual order in which requests are serviced by physical disks. Periodically, the workload statistics including disk service time are sent from the storage servers to the central manager (step 7) in a batched fashion.

4. DISK SERVICE QUALITY GUARANTEE

There are two aspects of our disk QoS guarantee mechanism: a disk request scheduling algorithm that enforces the QoS requirement of each virtual disk at run time, and an admission control algorithm that determines when to stop accepting new virtual disks.

4.1 Efficiency-Aware QoS-Guaranteed Disk Scheduling

4.1.1 Basic Algorithm

There are three requirements when we search for the ideal disk request scheduling algorithm for Stonehenge. First, the scheduler should be able to provide bandwidth guarantees. Second, the scheduler should be able to directly correlate a virtual disk’s delay bound with its bandwidth reservation. The existence of such correlation makes it possible to support delay guarantees using the bandwidth guarantee mechanism. Third, the scheduler should be able to attach a real-world deadline to each request. This requirement is essential for optimization of disk utilization efficiency, as we will see later. Because of the third requirement, the standard weighted fair queuing (WFQ) algorithm [14] used in network QoS is eliminated from consideration. Because of the second requirement, another well-known real-time request scheduling algorithm, delay-EDD [8], is also abandoned. In the end we adopt the Virtual Clock (VC) algorithm [22] because it satisfies all three requirements.

When the i th request associated with the j th virtual stream arrives, the VC algorithm first computes its finish time, $FT_{i,j}$:

$$FT_{i,j} = \max(AT_{i,j}, FT_{i-1,j}) + L_{i,j}/B_j \quad (1)$$

$$FT_{i,j} = \max(AT_{i,j}, FT_{i-1,j}) + (L_{i,j} + \delta)/B_j \quad (2)$$

$$\delta = avg_overhead \times T$$

where $L_{i,j}$ is the request size, $AT_{i,j}$ is the actual arrival time, T is the total available bandwidth of the underlying physical disks, and B_j is the bandwidth reservation of the j th virtual stream. Equation 1 is the FT formula in network resource scheduling. Equation 2 is for virtual disk scheduling. It accommodates the fact that each disk request has a fixed access overhead $avg_overhead$. Note it only includes the seek and rotational overhead, not the queuing delay. Then the VC algorithm sorts disk access requests in the system according to their finish time, and dispatches the one with the smallest finish time for service next. Intuitively, VC simulates the effect of applying a Time-Division Multiplexing scheme on the physical disk resource among multiple virtual disks according to their bandwidth reservation. Hence spare bandwidth is shared among backlogged virtual disks proportionally to their bandwidth reservations. Therefore, as long as the disk access requests are serviced before their finish time, the bandwidth share that the corresponding virtual disks receive will not be less than their bandwidth reservation. If we further play down the importance of disk request size in the finish time calculation, Equation 2 becomes

$$FT_{i,j} = \max(AT_{i,j}, FT_{i-1,j}) + 1/IOPS_j \quad (3)$$

where $IOPS_j$ is the j th virtual disk’s bandwidth reservation in number of I/Os per second. This modification is justified because the transfer time for most requests in real-world disk access workloads is negligible compared with seek delay and rotational latency, especially for modern disk drives (10000 RPM or faster). When the transfer time is not negligible compared to fixed access overhead, Equation 2 should be used.

From disk utilization efficiency standpoint, the disk request service order derived from the VC algorithm may not be optimal. To address this problem, Stonehenge uses an instance of dual-order disk scheduling algorithm [9] called CSCAN-based Virtual Clock (CVC) scheduler, which maintains two disk request queues, one (*QoS queue*) whose requests are ordered based on the VC algorithm’s finish time, and the other (*utilization queue*) ordered by the standard CSCAN order [7]. To pick the next disk request for service, the CVC scheduler checks whether servicing the head request in the utilization queue first will violate the finish time of the head request of the QoS queue, and dispatches the utilization queue’s head request if no finish time guarantee is violated and the QoS queue’s head request otherwise.

Note that the CVC scheduler only needs to check the head request of the QoS queue for finish time violation, but not subsequent ones. The reason is that Stonehenge uses the notion of *latest start time* as the basis of this check. In the QoS queue, the latest start time (LST_i) of the i th request is calculated as follows:

$$LST_i = \min(LST_{i+1}, FT_i) - 1/IOPS_{max} \quad (4)$$

$$1 \leq i < Q$$

$$LST_Q = FT_Q - 1/IOPS_{max}$$

where $IOPS_{max}$ is the maximum I/O rate of the underlying

physical disk resource, and Q is the size of the QoS queue. The Q th request is the tail of the QoS queue. Assume the head request of the utilization queue is Y . The finish time violation check in the CVC scheduler is

$$Service_Time(Y) + Current_Time \leq LST_1 \quad (5)$$

Note that every time a request from the QoS queue is serviced, the order of requests in the utilization queue may need to be recomputed.

4.1.2 Practical Issues

The service time depends on many factors such as sequentiality and read write ratio in a workload. The most accurate prediction is based on canonical disk service time model and details of physical disk parameters [20]. Our experience shows that while such an accurate disk service time prediction exists for SCSI disks, it is impossible for IDE disks. This is due to the IDE disk head instability [2]. Because Stonehenge is built on IDE hard drives, we choose to use run-time measurements for IDE disk service time prediction. We will show in Section 5.6 the impact of service time estimates.

The CVC scheduler makes the most of the “slack” in the request scheduling order imposed by the QoS requirement to optimize the disk resource utilization efficiency. However, because the CVC scheduler does not service disk requests strictly according to the “finish time” order, the bandwidth allocation to individual virtual disks is not completely the same as the original VC algorithm. This may lead to short-term deviation in bandwidth allocation from the virtual disks’ bandwidth reservation, as well as short-term skew in spare bandwidth allocation. However, this does not create fairness problems because long-term bandwidth allocation is still proportional to bandwidth reservation as long as each disk request is serviced before its associated finish time. To limit the extent of short-term unfairness in bandwidth allocation in CVC scheduler, one can impose an upper bound on the number of requests in the utilization queue that are considered as candidates for dispatching. Assume this bound is M . This CVC variant guarantees that at most M consecutive requests in the utilization queue can be serviced in a row. That is, the QoS queue should be visited at least once every M requests serviced. As M increases, the CVC scheduler focuses more on maximizing disk bandwidth utilization at the expense of short-term fairness in bandwidth allocation, and vice versa. Stonehenge sets M to one since it focuses only on long-term fairness. Modern disk drives and controllers can queue disk requests and service them in an order that maximizes throughput. The CVC scheduler needs to have full control over the request servicing order, even if this may lead to under-utilization of these queuing capabilities. To take advantage of command queuing in hardware and provide QoS guarantee at the same time, the CVC scheduler dispatches up to K ($K \geq 1$) requests to the disk. Typical command queuing depth is 64. The scheduler keeps on dispatching requests to the device as long as the total pending requests in device is less than or equal to K and serving them doesn’t violate the deadline of the first request in the QoS queue. In our current prototype, we use IDE drives only. Since IDE drives mostly don’t support command queuing, we set K to one in our prototype.

4.2 Measurement-Based Admission Control (MBAC)

Besides capacity, Stonehenge uses bandwidth as the criterion for admission control. Therefore, each of the other QoS requirements needs to be converted to an equivalent bandwidth requirement. The final bandwidth reservation for a virtual disk is the maximum of its inherent bandwidth requirements and the equivalent bandwidth requirements derived from other QoS attributes. To avoid wasting resource due to over-provisioning, Stonehenge exploits the actual workload information collected at run time to more accurately derive the equivalent bandwidth requirements.

4.2.1 Converting Delay Bound to Bandwidth Reservation

Since CVC uses the same formula as VC to compute the finish time for each request, and all requests are guaranteed to be serviced before their finish time, the requests serviced under CVC have the same delay bound as those under VC [14, 22]. The delay bound D for the i th virtual stream with a bandwidth reservation B_i is

$$D \leq (\sigma + L_{max})/B_i + L_{max}/T \quad (6)$$

where σ is the maximum burst in the incoming request sequence in terms of bytes, L_{max} is the maximum request size, and T is the total bandwidth of the underlying system [21]. This equation is appropriate for network resource only. The intuition behind this formula is that the worst case delay happens for a new request is when another request with size L_{max} is being serviced, and the bucket with depth σ is full. An easy way to convert a network latency bound Formula 6 to a disk latency bound formula is to replace the bandwidth parameters with *effective* disk bandwidth (Formula 7). The notion of effective disk bandwidth takes disk service overhead into account.

$$D \leq (\sigma + L_{max})/B_{i_effective} + L_{max}/T_{effective} \quad (7)$$

$$L/B_{i_effective} = (L + avg_overhead \times T)/B_i \quad (8)$$

Formula 8 converts a disk data transfer bandwidth (B) to an effective bandwidth $B_{effective}$ by expanding the request size by ($overhead \times T$) bytes. L is a disk request size variable. It is less than or equal to L_{max} . Plug in this conversion into Formula 7, the resulting disk delay bound formula becomes

$$D \leq (overhead \times T + \delta + L_{max})/B_i + (L_{max} + avg_overhead \times T)/T \quad (9)$$

$$overhead = avg_overhead \times (max_pending_reqs + 1)$$

$$\delta = max_pending_reqs \times avg_req_size$$

where $max_pending_reqs$ is the maximum number of requests the queue can hold for a given request size. The $avg_overhead$ is the average disk access overhead, and is measured and computed at run time. Compared with the original delay bound formula, we add an additional overhead item to account for the access overhead for each request. By multiplying the measured average disk access with T , we translate it to the number of bytes that could be transferred during the overhead time.

Because seek delay and rotational latency play an increasingly more significant part in disk service time, disk request size becomes relatively unimportant, especially when most requests are small. Therefore, we further simplify the latency bound formula to the following:

$$D \leq (\text{max_pending_reqs} + 1)/IOPS_i + 1/IOPS_{max} \quad (10)$$

where $IOPS_i$ is the reserved throughput for the i th virtual disk and $IOPS_{max}$ is the maximum throughput the physical storage system can support. If the assumption about disk request size is invalid in some cases, one can always use Formula 9.

4.2.2 Exploiting Load Information in Admission Control

Like physical disks, the utilization of a virtual disk is mostly under 100%. In addition, it is unlikely that all the virtual disks multiplexed on a physical storage system need their full bandwidth requirements simultaneously. Stonehenge exploits these two facts to increase the total number of virtual disks that can be admitted into a physical storage system.

Formula 10 converts a delay bound to its equivalent throughput requirement based on the worst-case delay formula associated with the VC scheduler. In practice, this has proved to be too conservative because not every disk request experiences the worst-case delay. Therefore Stonehenge also measures $P_{service}^N()$ curve based on the delay experienced by each disk access request in a system with N virtual disks. This curve represents the cumulative probability distribution of the ratio between the actual delay experienced by a request and the worst-case delay of the virtual disk with which the request is associated. $P_{service}^N()$ depends on the number of virtual disks in the system because the delay a request actually experiences depends on the actual load in the system, which is correlated with the number of virtual disks. Fortunately, $P_{service}^N$ typically can serve as a pretty good predictor for $P_{service}^{N+1}$, when N is much smaller than the maximum number of virtual disks that can be admitted into the system. With $P_{service}^N$, the delay bound formula used to decide whether to admit the $N + 1$ th virtual disk becomes

$$D \leq ((\text{max_pending_reqs} + 1)/IOPS_{N+1} + 1/IOPS_{max}) * (Q_{service}^N(E_{N+1}) + s) \quad (11)$$

where $Q_{service}^N$ is the inverse function of $P_{service}^N$, E_{N+1} is the probability that the $(N+1)$ th virtual disk needs its delay bound to be met and s is an adjustment factor that accounts for the impact of the new virtual disk on the delay behavior of existing virtual disks. When the system is lightly loaded or N is small, $Q_{service}^N(E)$, with E equal to 0.95 for example, can be as low as 10%, which means 95% of the requests experience a delay that is smaller than 10% of the worst-case delay. In contrast, a deterministic admission control algorithm will assume 100% of the requests experience the full worst-case delay. For a given E , $Q_{service}^N(E)$ grows closer to 1 with increasing N . A typical value for s in Stonehenge is 0.2. s is largely workload dependent. However, if the system is stable enough, the measurement-based feedback is able to detect a relatively stable s value. In this case, Equation 11 can be used. Otherwise, Equation 10 should be used if the workload is highly unpredictable.

We classify the VDs into two types: latency-bound or throughput-bound. Given a VD with requirement $(IOPS'_i, C_i, D_i, E)$, if its corresponding latency related throughput $IOPS''_i$ is larger than the virtual disk's general throughput $IOPS'_i$, this virtual disk is a latency-bound virtual disk.

Otherwise, this virtual disk is a throughput-bound virtual disk. For the set of throughput-bound VDs, we measure the combined I/O rate $Q_{I/O \text{ Rate}}(x)$ of these VDs at run time. $Q_{I/O \text{ Rate}}(x)$ represents for x percent of the periods, the I/O rate is less than or equal to $Q_{I/O \text{ Rate}}(x)$. Then $Q_{spare}(x) = IOPS_{max} - Q_{I/O \text{ Rate}}(x)$. For the latency-bound VDs, we measure the $P_{service}^M$, assuming we have M latency-bound VDs in the system. We only measure the I/O rate for the throughput-bound VDs, as we don't want to repetitively apply the measurement-based scaling down on the latency-bound VDs. The latency-bound VD's resource requirement is scaled down using Equation 11. Figures for the distribution curves can be found in [11]. The measurement-based admission control algorithm on one storage server for the i th virtual disk $(IOPS'_i, C_i, D_i, E)$ works as follows:

1. For all the j th VDs, ($1 \leq j \leq i$), convert the delay bound D_j to its equivalent bandwidth requirement $IOPS''_j$ using Equation 11.
2. Let $IOPS_j = \max(IOPS'_j, IOPS''_j)$, ($1 \leq j \leq i$).
3. Group the i VDs into two classes: latency-bound VDs and throughput-bound VDs.
4. Compute $Q_{spare}(x)$ from the throughput-bound VDs' measured total I/O rate $Q_{I/O \text{ Rate}}(x)$.
5. If $\sum IOPS_j \leq Q_{spare}(E)$, where j is in latency-bound VD set, admit the i th virtual disk; otherwise, reject it.

An important design decision in taking measurements is the choice of measurement window size. Previous workload studies show that disk I/O traffic shows stable daily cycles, and the I/O rate pattern is self-similar [10]. For these reasons, we choose 24 hours for the measurement window size in Stonehenge. In Stonehenge, we assume that virtual disk reservations come at different times. Each new virtual disk may change the measured distributions for existing workloads. Hence measurements must be done constantly at system run-time and the results are used in making admission control decisions for future virtual disk reservations. The measurements based on last N virtual disks need to be stabilized before they can be used for MBAC.

5. PERFORMANCE EVALUATION

5.1 Testbed and Methodology

We carried out the performance evaluation on the following Stonehenge testbed. The storage server nodes are Pentium-III 1.0-GHZ machines with 512 MBytes of memory, 64-bit 66-MHz PCI bus, two Promise 66-MHz dual-channel IDE controllers, an Intel Gigabit network card (Intel Pro-1000 XT), and a RAID0 array of four IDE drives (IBM DTLA-307075) connected to the IDE controllers. Each disk array has a total capacity of 300 GBytes, and uses a 64-KByte stripe unit. The client and central manager machines have the same hardware configuration except the disk array. The operating system is RedHat Linux with kernel version 2.4.18. We turned on the read-ahead cache on the disk and turned off the write-back cache.

The average round-trip time of sending an iSCSI command between a storage server node and the central manager node is around 50 μ sec. It includes step 4 and 5 in Figure 1 and the deadline computation on the manager node. Compared with the disk service time, which is 8 msec on the average, this round trip time is negligible. These two

Virtual Disk (VD)	Trace	Capacity (GB)	Throughput (IOPS)	Latency	I/O Rate Speed-up
0	TPC-C	9	N/A	N/A	N/A
1	Financial	15	128	120 msec	1.25
2	Web Search	96	675	N/A	2

Table 1: Specifications of the three virtual disks in Stonehenge. The performance requirements of Virtual disk 0 (best-effort) are not guaranteed. The delay bound and throughput requirements of Virtual disk 1 are guaranteed, and the throughput of Virtual disk 2 is guaranteed. The I/O Rate Speed-up column shows the speed-up of the corresponding trace when it is run. This speed-up ensures that the system is fully loaded.

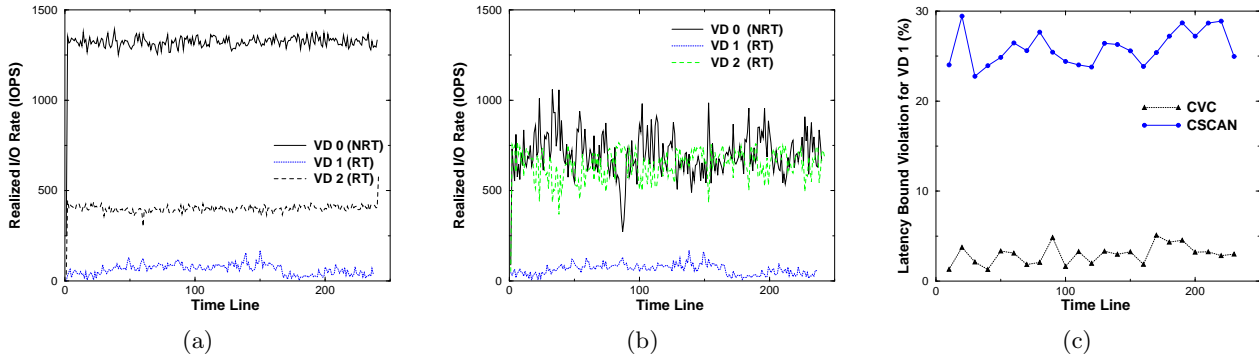


Figure 2: Throughput and latency guarantee in Stonehenge. Time line is in terms of seconds. (a) Throughput of each virtual disk over time when CSCAN is used. (b) Throughput of each virtual disk over time when CVC is used. (c) Running percentage of accesses to VD 1 that violate their delay bound over time for CVC and CSCAN.

numbers (50 μ sec and 8 msec) also suggest that a central manager node can handle at least 160 storage server nodes without becoming a bottleneck. However, in practice, before this limit is reached, the network link would already have been saturated.

Test programs running on the clients read requests from trace files and send them to the storage servers through iSCSI protocol. We used four traces in this study: the TPC-C trace is a transaction processing trace with 14 warehouses, the video stream trace is a trace that sequentially accesses data with a 64-KByte request size, the web search trace is a disk access trace collected from a commercial web search engine, and the financial trace is a trace collected from an anonymous enterprise financial application. Table 1 shows the specifications of the three virtual disks used in this study. The actual I/O rate is after speed-up for a full load on the testbed. Unless otherwise stated, each of these virtual disks is served by three identical storage server nodes with a striped mapping. Multiple virtual disks co-exist on this three-node storage cluster.

5.2 Effectiveness of QoS Guarantee

The major goal of Stonehenge is to ensure the performance specifications of each of the virtual disks sharing a physical storage system are satisfied at all times. To verify whether Stonehenge achieves this goal, we set up the three virtual disks (VD) in Table 1 on the Stonehenge testbed. Against each VD, we ran a separate disk access trace that is appropriately scaled to ensure that the input I/O rate exceeds the throughput requirements and system capacity. Figure 2(a) and Figure 2(b) show the throughput each virtual disk receives over time, under the CSCAN scheduler and the CVC scheduler, respectively. Because CVC takes into account the throughput requirement of VD 2, it is able to meet VD

2’s throughput requirement whereas CSCAN cannot. Moreover, the curves in Figure 2(b) are fluctuating more dramatically than those in Figure 2(a), because the disk requests in the input traces tend to be bursty, and CVC is more likely to reflect the input burstiness in the virtual disks’ throughputs than CSCAN, because CVC needs to be more responsive to the disk access requests to guarantee their associated virtual disks’ throughput or latency requirements. Figure 2(c) shows the latency bound violation percentage over time for VD 1 in the same experiment. Overall, around 97% of requests to VD 1 were serviced within their delay bound under CVC while only 75% under CSCAN, which demonstrates that CVC can indeed meet the delay bound requirement of a VD quite well. The main reason that there are still delay bound violations under CVC is the leeway factor f for VD 1 is set to 1.5 in this experiment, which is not high enough to absorb the load skew among the three disks underlying VD 1.

5.3 Efficiency of MBAC

To demonstrate the efficiency of measurement-based admission control (MBAC) algorithm, we run three experiments each using a distinct sequence of VD requests as input. We show the maximum number of VDs accepted in Table 2 under three different admission control schemes: deterministic, MBAC and the Oracle scheme. The Oracle scheme assumes no limit of system resources and keeps admitting VD requests until some admitted VDs’ QoS guarantees are violated. The number of VDs admitted by the Oracle scheme represents the upper bound of number of VDs a system can sustain while still satisfying all VDs’ QoS requirements. The table shows that MBAC can double or even triple the maximal number of VDs that can be admitted by the deterministic approach. In most cases, MBAC admits

	VD Type	Probability	Deterministic	MBAC	Oracle
Run 1	Financial	95%	7	20	22
Run 2	Mixed	95%	7	14	14
Run 3	Mixed	85%	7	17	17

Table 2: A comparison of the maximum number of virtual disks that deterministic or measurement-based admission controller can accept with the same system resource. In the case of mixed VD type, 50% of the VDs are running financial trace and 50% are running web search trace.

Number of VDs	7	8	9	10	11	12	13	14	15
$Q_{service}(0.95)$	11%	14%	15%	19%	24%	30%	37%	49%	-
MBAC Resource Reservation	N/A	34%	38%	43%	47%	51%	55%	67%	95%
Deterministic Resource Reservation	90%	-	-	-	-	-	-	-	-

Table 3: Resource reservation of MBAC and deterministic admission control algorithms when admitting a sequence of VDs with mixed types. It also shows the evolution of the parameter $Q_{service}()$ used in Equation 11. The QoS guarantee probability is 0.95 in this case.

almost as many VDs as the Oracle scheme can. Due to the conservative value of s we used in run 1, MBAC performs slightly worse than Oracle. Also, as the VDs in run 2 have more stringent QoS guarantee probability, fewer VDs are admitted in run 2 compared to run 3.

To understand why MBAC can admit more VDs than the deterministic scheme, Table 3 compares the resource reservations that the MBAC and deterministic approach actually make as the number of virtual disk requests increases. It shows that one can significantly reduce the throughput requirement for a given delay bound when using Equation 11 and thus increases the number of VDs to be admitted. For example, when there are seven VDs, the deterministic admission control already reserves close to 90% of the disk throughput while MBAC reserves less than 40% of resource. Despite this advantage, as $P_{service}^N$ and thus $Q_{service}^N$ already accurately capture the run-time load and dynamic disk access pattern, MBAC’s decision to admit more VDs rarely leads to violation of the QoS guarantees of existing VDs. Table 3 also shows that $Q_{service}(E_i = 0.95)$ grows faster than linearly with the increasing N . Adding one more VD usually increases $Q_{service}^N(E)$ by 0.2. As a result, we choose 0.2 as the conservative factor s in Equation 11 to cover the potential gap between $P_{service}^N$ and $P_{service}^{N+1}$.

Table 4 shows the effect of QoS guarantee probability on the number of virtual disks that can be admitted. In this test, the VDs are latency-bound and are either running financial trace or web search trace. However, the capacity, latency, and throughput requirements have been scaled down to allow more VDs to be accepted in Stonehenge. As expected, the number of virtual disks admitted increases with the decrease in QoS guarantee probability. The improvement of MBAC over deterministic approach increases from 1.4 fold when E is 95% to 1.9 fold when E is 80%.

Probability	95%	90%	85%	80%
MBAC	17	18	19	20
Deterministic	7	7	7	7
Oracle	17	20	21	22

Table 4: The impact of QoS guarantee probability E on the admission efficiency of MBAC.

5.4 Efficiency of CVC Scheduler

To determine the improvement in disk utilization efficiency of CVC over VC, we map the three virtual disks on a single disk array, and ensure each test run touches 40% of the disk array. To examine the impact of system spare bandwidth on CVC’s efficiency, we varied the resource reservation scaling factor (S). The throughput reservations for each of the three co-existing VDs are: 2S, 5S and 14S IOPS. The larger the scaling factor is, the less spare bandwidth is left in the system. A scaling factor of 10 corresponds to 0% spare bandwidth. Figure 3(a) shows the average disk bandwidth utilization of VC, CVC and CSCAN schedulers, as normalized against CSCAN’s disk bandwidth utilization efficiency. CVC is 25% better than VC and 11% worse than CSCAN in bandwidth utilization efficiency when there is plenty of spare bandwidth, i.e, when the scaling factor is 2. As spare bandwidth decreases, the disk utilization efficiency of CVC decreases, because there is less room for optimization. On the other hand, the disk utilization efficiency of VC remains largely unaffected by the amount of spare bandwidth because its schedule is load-independent. When the system is fully booked or over-booked, the disk utilization efficiency improvement of CVC over VC is within 5%. To examine the performance impact of CVC for traces with more sequential access patterns, we run video stream trace on two VDs and present the results in Figure 3(c). It shows similar trend of disk utilization efficiency of CVC as in Figure 3(a) with increasing resource reservation. However, as VC tends to reorder the requests to catch the request deadline and thus destroy the sequential access pattern, the performance gap between VC and CVC (CSCAN) is much larger than what we saw in the previous experiment. Under this trace setup, when the system is relatively lightly loaded (scaling factor ≤ 3), CVC performs about 50% to 80% better than VC.

Figure 3(b) presents the deadline violation percentage of VC, CVC, and CSCAN with varying spare system bandwidth in the experiment with three VDs. As expected, VC has the lowest delay bound violation percentage, followed by CVC and then CSCAN. Before the system is close to fully booked (scaling factor=10), the gap in delay bound violation percentage between CSCAN and CVC is growing because as system spare bandwidth decreases, it becomes

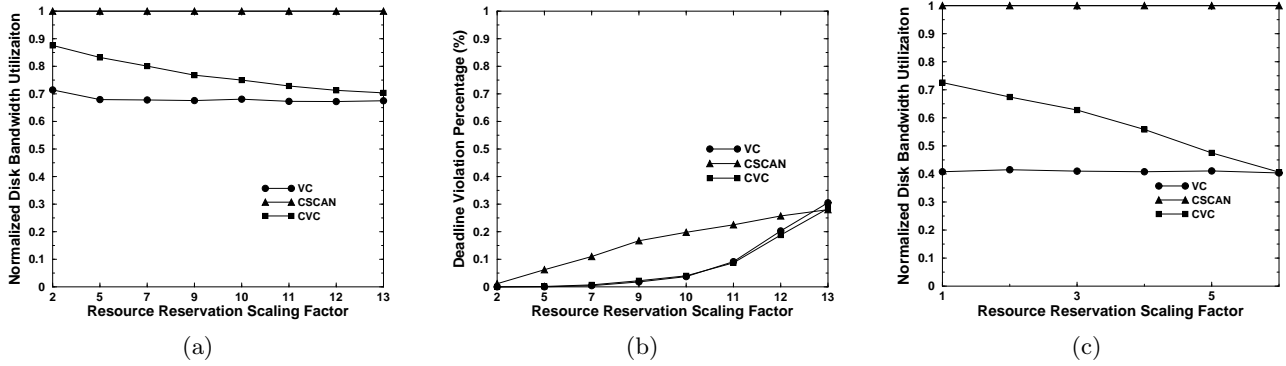


Figure 3: The impact of spare system throughput (controlled by the resource reservation scaling factor) on CVC, VC and CSCAN schedulers. The experiment of the first two figures run three VD's each hosting one trace listed in Table 1 respectively. (a) The normalized disk utilization efficiency under various resource reservation scaling factors. (b) The request deadline violation percentage various resource reservation scaling factors. (c) The normalized disk utilization efficiency under various resource reservation scaling factors when running two video streaming traces.

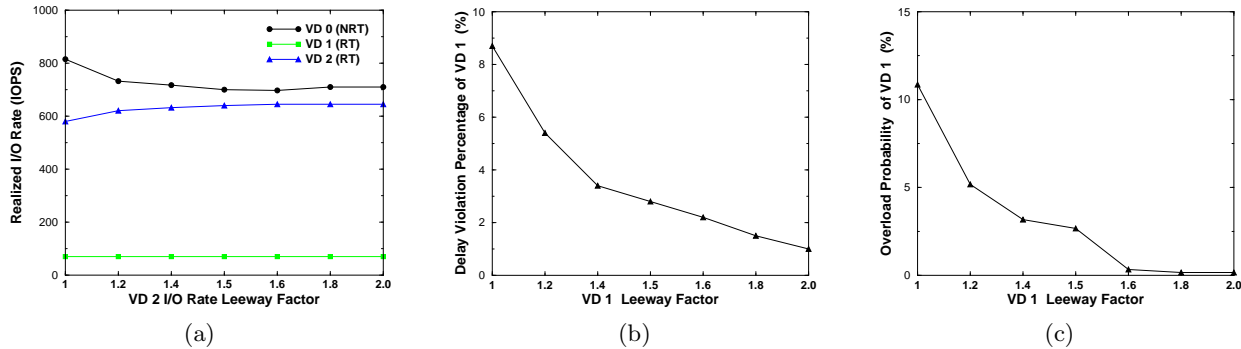


Figure 4: Impact of leeway factor on QoS guarantee. (a) We vary the estimate of leeway factor f of VD 2 and examine the realized throughput for each VD. (b) We vary the estimate of leeway factor f of VD 1 and examine the delay bound violation percentage for VD 1. (c) The probability distribution that the actual I/O rate on any component drives of VD 1 exceeds its reservation under varying leeway factor f of VD 1. We call this probability the overload probability. The reserved rate is scaled by f .

more important to take into account deadlines when scheduling disk requests in order to meet their delay requirements. The same reasoning explains the small gap between CVC and VC. However, after the system is fully booked, the gap in request deadline violation percentage between CSCAN and CVC is shrinking, because as the system becomes more overloaded, deadline-driven disk schedulers such as CVC can meet fewer requests' deadlines. When the scaling factor is 13, the deadline violation percentage of CSCAN is actually slightly smaller than that of VC and CVC. The reason is that at this input load, the improvement in disk utilization efficiency of CSCAN over CVC (or VC) effectively increases the system spare bandwidth, and is enough to offset the increase in deadline violation percentage due to its disregard of deadlines.

Because of traffic skew (reflected in f), even disk access requests under VC experience delay bound violation, although the percentage is much lower than that associated with CSCAN. In addition to traffic skew, the disk service time estimate used in CVC may be less than its actual service time occasionally, thus causing CVC to violate delay bounds slightly more frequently than VC.

5.5 Impact of Leeway Factor

When a virtual disk is mapped to multiple disk arrays, either by striping or mirroring, each component disk array of a virtual disk may not receive the same amount of disk access load within a short period of time or in the long run. This traffic skew is fairly common in real disk traces, and can also happen at the level of component disks within a disk array. To address the load imbalance due to traffic skew, Stonehenge reserves f ($f \geq 1$) times the throughput requirement for each virtual disk, where f is called the leeway factor. To evaluate the effect of the leeway factor, we use the same set-up as in Section 5.1 to run the following experiments. By varying the leeway factor of VD 2 and VD 1 separately, we get the results in Figure 4(a) and (b). As the leeway factor increases, the quality of the throughput and latency guarantee that Stonehenge can provide is improved. Note that with the same increment in the leeway factor, the throughput increase in VD 2 is not as significant as the delay bound violation percentage reduction in VD 1. Because traffic skew is an instantaneous phenomenon, it impacts delay bound violation, which is also more likely to be effected by short-term events, more than received throughput, which is relatively more long-term. As a result, any technique to

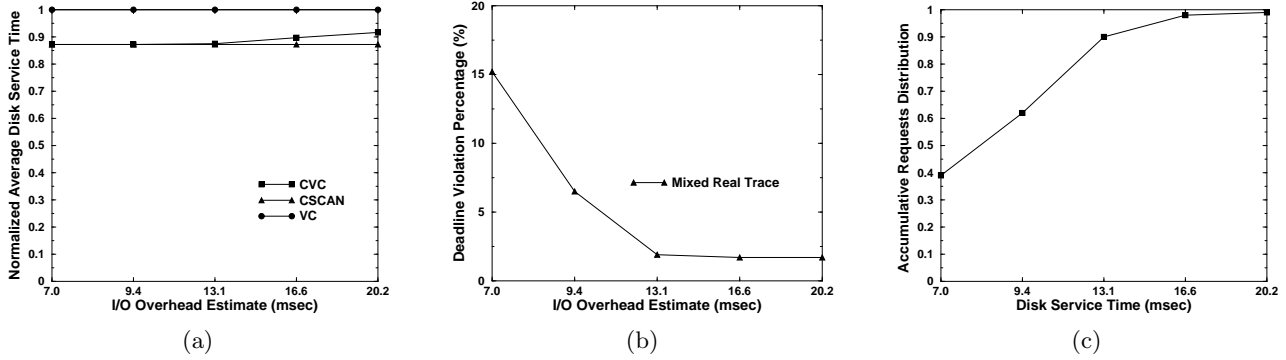


Figure 5: Impact of disk service time estimate. The system hosts three virtual disks each running the TPC-C trace, financial trace and web search trace respectively. (a) Measured average disk service time normalized over VC’s measured disk service time under different disk service time estimates used in CVC. (b) The deadline violation percentage of CVC under different disk service time estimates. (c) The measured cumulative distribution of disk service time.

mitigate the impact of traffic skew benefits delay bound violation percentage more than received throughput. To confirm that the degradation in QoS guarantee is indeed due to disk access skew, we analyze the disk request trace used in the experiment that results in Figure 4(b) to generate Figure 4(c). The Y-axis of Figure 4(c) is the probability that the input I/O rate to any component array of VD 1 exceeds its throughput reservation and the X-axis is the leeway factor of VD 1. Together with Figure 4(b), this figure clearly shows that adjusting the leeway factor properly can effectively eliminate the effect of traffic skew on QoS guarantee. The reason that CVC still sees delay bound violation after the leeway factor is set to 2.0 is mainly due to incorrect disk service time estimate, whose effect is discussed in the next subsection. The leeway factor is more suitable for offline analysis, since we don’t want to adjust the resource allocation per storage server online.

5.6 Impact of Disk Service Time Estimate

In CVC scheduling, the scheduler decides whether to schedule the next request from the utilization or QoS queue based on the estimate of disk service time of the head request in the utilization queue. On one hand, the more aggressive the disk service time estimate is, the more likely the CVC scheduler will follow the scheduling order of CSCAN and achieve higher disk utilization efficiency, at the expense of higher deadline violation percentage. On the other hand, a more conservative disk service time estimate makes it more likely to meet the deadline of the disk requests, but leading to lower disk utilization efficiency. To investigate empirically the impact of disk service time estimate on the disk utilization efficiency of the CVC scheduler, we vary the disk service time estimate from 7 msec to 20.2 msec. Figure 5(a) shows that when the disk service time estimate used is no greater than 13.1 msec, CVC’s average request service time is almost identical to CSCAN’s, indicating that CVC’s disk utilization efficiency is as good as CSCAN. Beyond 13.1 msec, the disk utilization efficiency of CVC drops gradually. The impact of disk service time estimate on CVC’s deadline violation percentage is shown in Figure 5(b). As expected, higher or more conservative disk service time estimates result in lower deadline violation percentages, which essentially remains unchanged when the disk service time estimate is greater than 13.1 msec.

Instead of using a fixed magic value such as 13.1 msec, Stonehenge dynamically measures a disk service time distribution, $P_{I/O_Time}(x)$, which represents the probability that a request’s disk service time is less than or equal to x msec. Stonehenge chooses to use the value $P_{I/O_Time}^{-1}(0.90)$ as the disk service time estimate. To verify this design, we ran three VDs as in Table 1 on a single server node and measure $P_{I/O_Time}(x)$, which is shown in Figure 5(c). It shows that 90% of the requests in this run experience a disk service time less than 13.1 msec. This means that using 13.1 msec as the disk service time estimate for the head request of the utilization queue in CVC scheduling is reasonably accurate, and should not cause substantial increase in deadline violation percentage, as is the case in Figure 5(b).

6. CONCLUSION

This paper presents the design, implementation and evaluation of a cluster-based multi-dimensional storage virtualization system called Stonehenge. Stonehenge can virtualize a physical disk storage system along three dimensions: capacity, throughput and latency. That is, Stonehenge can multiplex multiple virtual disks, each with a specific combination of capacity, throughput, and latency characteristics, on a single physical disk storage system, and guarantee each virtual disk’s performance specification regardless of the input loads. The key research contributions of this work include:

- A dual-queue real-time disk scheduler called CVC that is able to maximize the disk utilization efficiency by making the best of the slack available in satisfying the performance guarantees of virtual disks. CVC achieves 25% to 80% better disk utilization efficiency than generic real-time disk schedulers based on Virtual Clock scheduling algorithm. CVC provides delay bound, fair bandwidth (including spare bandwidth) allocation proportional to reservations.
- An innovative measurement-based admission control (MBAC) algorithm that is able to exploit the statistical multiplexing nature of the input loads to improve the number of virtual disks that can be admitted by a factor of 2 to 3, while supporting both deterministic and probabilistic guarantees on bandwidth and delay bound. MBAC features a statistical admission control

algorithm that can also provide delay guarantees for storage systems.

- A general introspective framework that incorporates run-time measurements to improve the accuracy of disk service time estimate, the degree of statistical multiplexing, and the conversion of delay to bandwidth guarantee, and indirectly optimize the overall system efficiency.

As a result of these innovations, the level of performance guarantee in Stonehenge is more rigorous than most if not all existing systems, and yet its system utilization efficiency is also among the best, to the best of our knowledge. In summary, the multi-dimensional storage virtualization technology in Stonehenge produces virtual disks that are as tangible as physical disks and yet much more manageable and flexible, while keeping the virtualization cost to a minimum.

7. REFERENCES

- [1] G. Alvarez, E. Borowsky, S. Go, T. Romer, R. BeckerSzendy, R. Golding, A. Merchant, M. Spasojevic, A. Veitch, and J. Wilkes. Minerva: an automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems*, 2001.
- [2] D. Anderson, J. Dykes, and E. Riedel. More than an interface: Scsi vs. ata. In *Proceeding of File System and Storage Technology Conference (FAST 03)*, Jan 2003.
- [3] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch. Hippodrome: running circles around storage administration. In *Conference on File and Storage Technology (FAST'02)*, pages 175–188, Monterey, CA, January 2002.
- [4] E. Anderson, M. Kallahalla, S. Spence, R. Swaminathan, and Q. Wang. Ergastulum: an approach to solving the workload and device configuration problem. Technical Report HPL-SSP-2001-05, HP Laboratories, 2001.
- [5] L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control. In *Proceedings of IEEE Infocomm 2000*, Tel Aviv, Israel, March 2000.
- [6] J. L. Bruno, J. C. Brustoloni, E. Gabber, B. Ozden, and A. Silberschatz. Disk scheduling with quality of service guarantees. In *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS'99)*, pages 400–405, 1999.
- [7] S. Chen, J. A. Stankovic, J. F. Kurose, and D. Towsley. Performance evaluation of two new disk scheduling algorithms for real-time systems. Technical Report UM-CS-1990-077, 1990.
- [8] D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, 1990.
- [9] K. Gopalan and T. Chiueh. Real-time disk scheduling using deadline sensitive scan. Technical Report ECSL-TR-92, SUNY Stony Brook, January 2001.
- [10] W. Hsu. Dynamic locality improvement techniques for increasing effective storage performance. Technical Report UCB/CSD-03-1223, Computer Science Division, University of California at Berkeley, Jan, 2003.
- [11] L. Huang. Stonehenge: A high performance virtualized storage cluster with qos guarantee. Technical Report TR-138, ECSL, Computer Science Department, SUNY Stony Brook, 2003.
- [12] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 2–13, 1995.
- [13] C. R. Lumb, A. Merchant, and G. A. Alvarez. Façade: Virtual storage devices with performance guarantees. In *Proceedings of the 2nd USENIX conference on File and Storage technologies*, pages 131–144, San Francisco, CA, April 2003.
- [14] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.
- [15] P. Shenoy and H. M. Vin. Cello: A disk scheduling framework for next generation operating systems. In *Proceedings of ACM SIGMETRICS Conference, Madison, WI*, pages 44–55, June 1998.
- [16] Y. Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one integer programming. *Management Science*, 21:1417–1427, 1975.
- [17] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource overbooking and application profiling in shared hosting platforms. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, pages 239–254, Boston, MA, December 2002.
- [18] H. M. Vin, P. Goyal, and A. Goyal. A statistical admission control algorithm for multimedia servers. In *ACM Multimedia*, pages 33–40, 1994.
- [19] R. Wijayarathne and A. Reddy. Integrated qos management for disk I/O. In *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS'99)*, Florence, Italy, June 1999.
- [20] B. Worthington, G. R. Ganger, Y. N. Patt, and J. Wilkes. On-line extraction of SCSI disk drive parameters. In *Performance Evaluation Review*, volume 23, pages 146–56, May 1995.
- [21] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. In *Proc. of the IEEE, Vol. 83, no. 10*, pages 1374–1396, Oct 1995.
- [22] L. Zhang. VirtualClock: A new traffic control algorithm for packet-switched networks. *ACM Transactions on Computer Systems*, 9(2):101–124, 1991.