Storage Workload Characterization and Consolidation in Virtualized Environments

Ajay Gulati, Chethan Kumar, Irfan Ahmad
VMware Inc., Palo Alto, CA 94304
{*agulati, ckumar, irfan*}*@vmware.com*

Abstract

Server virtualization is accelerating the already existing need for shared storage area network (SAN) infrastructure for the associated benefits of migration, flexible management and economies of sharing. Understanding the characteristics of a workload in terms of access locality, IO sizes, read write ratio etc. is crucial for effective placement and consolidation. The lack of visibility into workload details and placement often leads to design inefficiencies and over-provisioning.

This paper presents workload characterization study of three top-tier enterprise applications using VMware ESX server hypervisor. We further separate out different components (for example data, index and redo log in a database) of these workloads to understand their behavior in isolation. We find that most workloads show highly random access patterns. Next, we study the impact of storage consolidation on workloads (both random and sequential) and their burstiness. Our experiments show that its beneficial to put random workloads together because it improves utilization and reduces latency without hurting performance. In case of consolidating random and sequential workloads, we see performance degradation for the sequential workload. Burst analysis reveals that most workloads are very bursty and the rate can change by as much as 50% within a few seconds. This makes it even more important to have the ability to absorb bursts and exploit benefits of statistical multiplexing by doing consolidation of workloads, whenever possible.

1 Introduction

Most organizations are dependent on multiple IO centric applications for their day-to-day business. Mail servers, transaction processing and decision support systems with tens of terabytes of data volumes are quite common. The space required to store the data associated with these applications and the IO performance required to transport the data to/from processing node (CPU) to the permanent storage (hard disks) has necessitated the need for a dedicated external storage infrastructure. But the cost and complexity of such an infrastructure prohibits having dedicated infrastructure for each of the applications. The best option in this situation is to share the storage infrastructure across multiple applications.

In spite of a lot of existing work on storage performance analysis, automatic data placement and storage allocation [6,7,9–11], most storage administrators rely on ad hoc techniques and rules of thumb for both configuration of storage and data layout on different volumes. For example, one of the common ways to place two top-tier workloads is to create separate RAID groups on disjoint sets of disks. Over provisioning is another technique commonly used to mitigate real or perceived performance issues. These techniques although helpful in some cases cannot be generalized for all. Since over provisioning and hard partitioning can be inefficient and expensive, the placement of workloads should be guided by detailed workload characterization and analysis. Multiplexing of various workloads in virtualized environments, is causing enormous pressure on SAN infrastructure due to a complex mix of IO workloads. Good understanding of IO characteristics of applications in shared environment and the effect of storage consolidation on their IO performance is paramount to designing an efficient storage infrastructure.

This paper takes a first step towards a comprehensive workload characetrization of three top-tier applications and studies the impact of storage consolidation on workloads. Investigated workloads include a large mail server, multiple OLTP loads, a non-comparable implementation of TPC-C business model and a decision support system. The disk IO for these top-tier enterprise applications can be broken into various components such as data access, index access, log writing and so on, which can be supported by different back-end storage devices. Existing studies have often looked at these in a more coarse grained manner, which hide some of the useful characteristics of these components that can help in improving performance by better placement. In this study, we capture their individual IO patterns.

To understand the effect of sharing IO devices we compare the two cases of running workloads in isolation and then running them simultaneously while combining the underlying devices. Such combined access patterns also represent a workload that an array may see from multiple applications running on various hosts (in a virtualized environment or otherwise). Finally, we analyze the degree of burstiness of workloads by looking at the arrivals with time using detailed tracing. Our preliminary results show that: (1) Different components of these well known workloads have very different IO characteristics and it helps to place them accordingly, (2) Workloads show high degrees of randomness and are bursty in nature (arrivals can change by almost 50% in few seconds) (3) Sharing underlying devices for random workloads is a good way to improve utilization and reduce latency especially in case of bursts.

In the rest of the paper, we first discuss our methodology in Section 2. Extensive workload characterization of our workloads is provided in Section 3. Section 4 presents the study of storage consolidation followed by burst characteristics of workloads in Section 5. Section 6 provides a survey of previous work related to workload analysis and finally we conclude with some directions for future work in Section 7.

2 Characterization Methodology

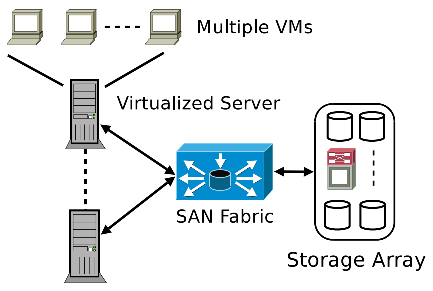


Figure 1. Shared storage access in virtualized environments

To characterize workloads we make use of virtualization technology by running workloads inside virtual machines (VMs) and observing the IO patterns from the hypervisor. Figure 1 shows an example setup which is quite common for virtualized data centers and which we used for this study. In this setup, the storage array is carved up into groups of disks in a RAID configuration. On each of the RAID groups, multiple logical units of storage (LUNs) are allocated, which are then assigned to the applications. Our characterization technique is from previous work [12] which uses a utility called *vscsiStats* in the VMware ESX Server hypervisor to efficiently track a comprehensive set of IO characteristics. The utility is very light-weight, transparent to the VMs running on top and generates histograms for the following IO parameters:

Seek distance: a measure of the spatial locality in the workload measured as the minimum distance in terms of sectors or LBN (logical block number) from among the last k number of IOs (k=1 in the data presented here). In the histograms, small distances signify high locality.

IO data length: in different bins of size 512 Bytes, 1KB, 2KB and so on.

Interarrival period: histogram of the time interval (microseconds) between subsequent IO requests.

Outstanding IOs: these denote the IO queue length that the hypervisor sees from a VM.

IO latency: measured for each IO from the time it gets issued by the VM till the VM is interrupted for its completion.

Read/Write: All of these data points are maintained for both reads and writes to clearly show any anomaly in the application's or device's behavior towards different request types.

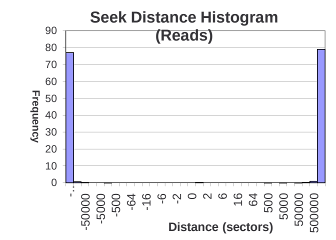
These parameters provide information in great detail to understand and reason about workload behavior. Furthermore, instead of computing averages over large intervals, histograms are kept so that a full distribution is available for later analysis. For example, knowing the average IO size of 23 KB for an application is not as useful as knowing that the application is doing 8 and 64KB IOs in certain proportion. The characterization tool can also collect a complete trace of IOs as well, keeping detailed information about arrival time, starting block, IO size, destination device, etc. for each individual IO. We use traces only to collect high resolution burst characteristics because, unlike online histograms, enabling traces has an observable performance effect. Examples in the next section will help in understanding the ease and power of this technique.

3 Workload Characterization

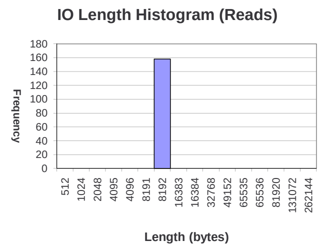
In this section, we introduce all the applications that we studied and their main characteristics. Our application benchmark workloads include Swingbench [4], DVD-Store [1], Microsoft Exchange [2] and a TPC-C like workload running on top of a top-tier commercial database (Oracle [3]). Now we will explain each of them in more detail.

3.1 DVDStore (Oracle)

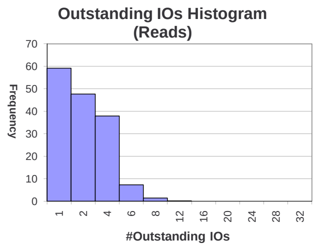
DVDStore version 2.0 is an online e-commerce test application with a backend database component, and a client program to generate workload. We used the largest dataset option for DVDStore (100 GB), which includes 200 million customers, 10 million orders/month and 1 million products. The server ran in a RHEL4-U4 64 bit VM with 4 CPUs, 32 GB of memory and a storage backend of 5 disk RAID 5 configuration. The client ran on a native Windows machine. We separated out the data, index and redo log portions of the database onto separate virtual disks. Figure 2 shows the results for DVDStore data virtual disk. We observe that most



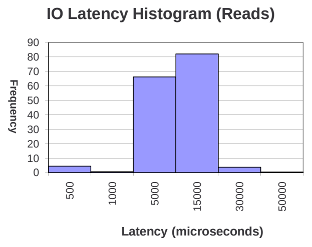
(a)



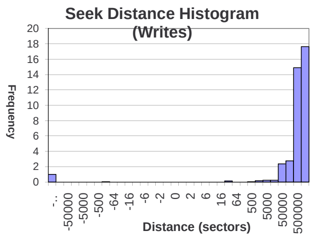
(c)



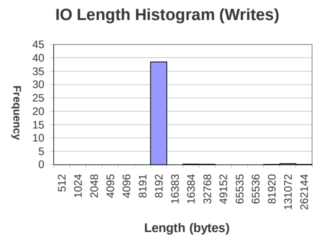
(e)



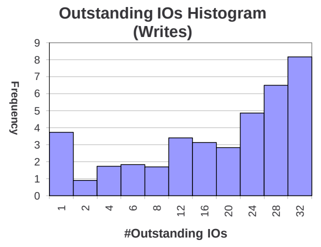
(g)



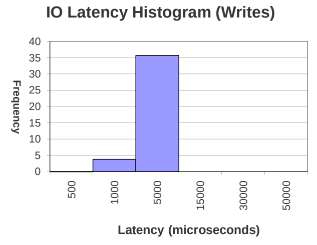
(b)



(d)



(f)



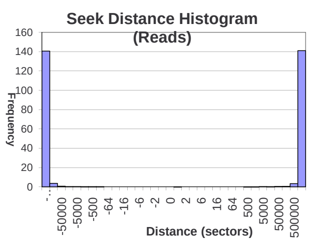
(h)

Figure 2. DVDStore Data disk characteristics

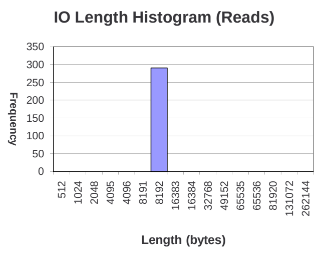
IOs are random (Figure 2a, 2b), 8KB in size (Figure 2c, 2d) with ≤15ms latency (Figure 2g, 2h). Most of the time, the number of outstanding IOs that are reads is less than 5 (Figure 2e) whereas many more writes (Figure 2f) are outstanding at any given time (up to 32). An interesting oddity is that the seek distances observed for writes are heavily biased in the positive direction. This is probably due to the buffering of asynchronous writes and periodic flushing which allows IO scheduling of writes by an elevator or SCAN based algorithm. This doesn't happen for reads because they need to be issued right away. Read latencies are less than 15 ms, and write latencies are ≤5 ms (due to write caching at the storage array). The index LUN (Figure 3) has largely similar characteristics as the data LUN. Redo logs (Figure 4) show very sequential writes with size varying from 0.5 to 64 KB with peak around 16KB and the IO latencies are also very small (≤ 1ms) in all cases. Also note that log disk always has only one IO outstanding in comparison to multiple outstanding IOs for data and index disks.

3.2 Microsoft Exchange Server

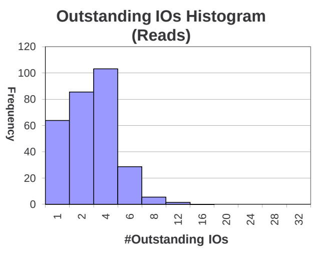
Microsoft Exchange Server is a scalable and popular commercial mail server supporting thousands of users per instance. It is also one of the popular applications used in virtualized data centers. As such, it is highly valuable to determine its IO characteristics. We picked a standard enterprise configuration of Exchange 2007 SP1 and ran it on Windows Server 2008 in a VM on top of VMware ESX server. Loadgen, which is a well known Exchange workload generator, was used to simulate Exchange workload. The VM was configured with 1 vCPU and 7 GB of memory. Loadgen was configured to create 1200 heavy users in two databases (160 GB, 600 users each). The data LUN was put on RAID 0 with 13 disks and log LUN was put on RAID 0 with 6 disks. The CPU was not fully utilized on the ESX server running the Windows VM. See Figures 5, 6 for histograms for the Microsoft Exchange workload. In the workload configuration, we placed log files in a separate virtual disk. The data disk saw a lot of random reads but they exhibit a bimodal behavior in their spatial locality: read IOs arrive either at large random offset from previous reads or have a 32KB offset from previous IOs (20% within 32KB). This is indicative of some sort of a stride pattern. Data writes on the other hand show a wide range of seek distance patterns (16KB and above). Again writes are mostly in the forward direction likely because of buffering and IO scheduling. Data reads are always 8KB while writes vary from 8 to 32KB, with almost 75% at 8KB. The read and write histograms clearly show significant burstiness in this workload (Figure 5e-f). Writes undergo more bursts presumably due to periodic flushing of buffers. Also the IO latencies move in line with the burst sizes (compare Figure 5f and 5h), which shows that it is quite important to understand and handle bursty behavior while doing scheduling in shared environments. This shows that sharing underlying disks will absorb bursts better due to statistical multiplexing of IO request patterns. As expected, the log disk doesn't receive any reads. For writes, log IO is completely sequential (Figure 6). Log writes sizes are predominantly 512 Bytes.



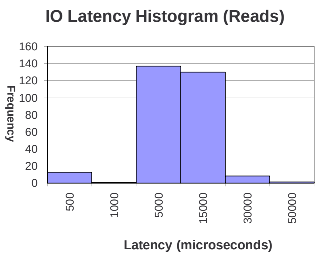
(a)



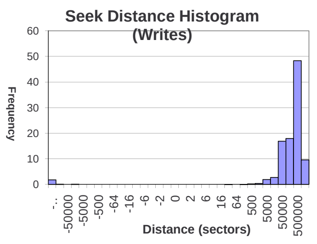
(c)



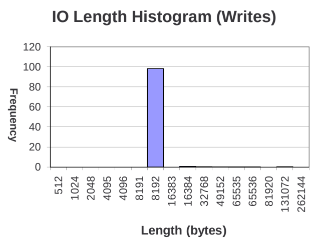
(e)



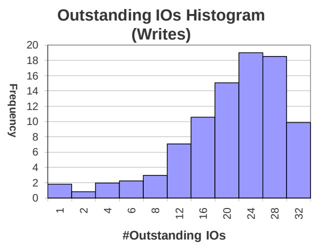
(g)



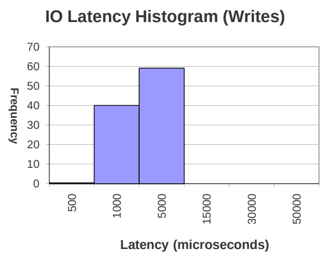
(b)



(d)

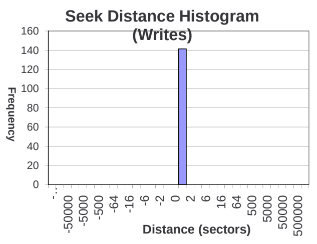


(f)

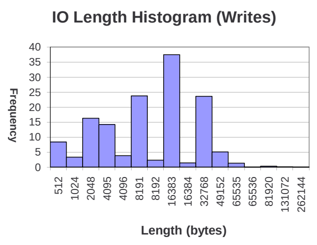


(h)

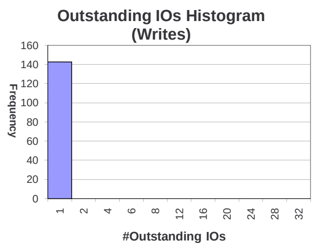
Figure 3. DVDStore Index disk characteristics



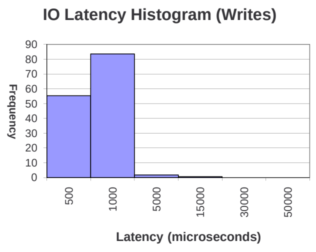
(a)



(b)



(c)



(d)

Figure 4. DVDStore Redo Log disk characteristics

3.3 TPC-C (Oracle)

This workload is a non-comparable implementation of TPC-C business model. It is run in a closed loop, where a small number of processes generate requests with zero think time between the completion of a request and issue of next request. Also the setup is a two tier environment, where load is generated on the first tier (client) and submitted to the back end database. These runs are against 2500 warehouse database, with 80 users. The number of users is large enough to keep the CPU fully utilized, and adding more users won't increase the performance.

Characteristics were collected for one data and one log LUN, each on 3 disk RAID 0 configuration. There were 14 data LUNs in the experiment but we report numbers for only one because they are mostly identical. The seek distance histogram (Figure 7) shows that data LUN has mostly random accesses and log LUN (Figure 8) is mostly sequential. Between reads and writes, its interesting to note that reads are done both in forward and backward direction but write are again mostly done to higher block numbers. Data LUN has both reads and writes in ratio 2:1 and the sizes are mostly 2KB per IO (consistent with the value set in application). In terms of IO latency, reads see a higher IO latency of ~15 ms whereas writes only see a latency of around 5ms. We believe this is due to the caching of writes in the storage controller's cache. Concurrency is relatively high for this workload, getting tens of reads outstanding per-LUN (more than 100 overall). Log LUN (Figure 8) shows very sequential writes with zero reads. The IO sizes are quite variable and the most common sizes are 16 and 32KB. The latency of these writes is around 5 ms, which is most likely due to bigger IO sizes.

3.4 Swingbench DSS (Oracle)

Swingbench is a free workload generator that can produce transaction processing and decision support type of workloads. It is designed to stress an underlying Oracle database. Swingbench DSS represents a decision support system type

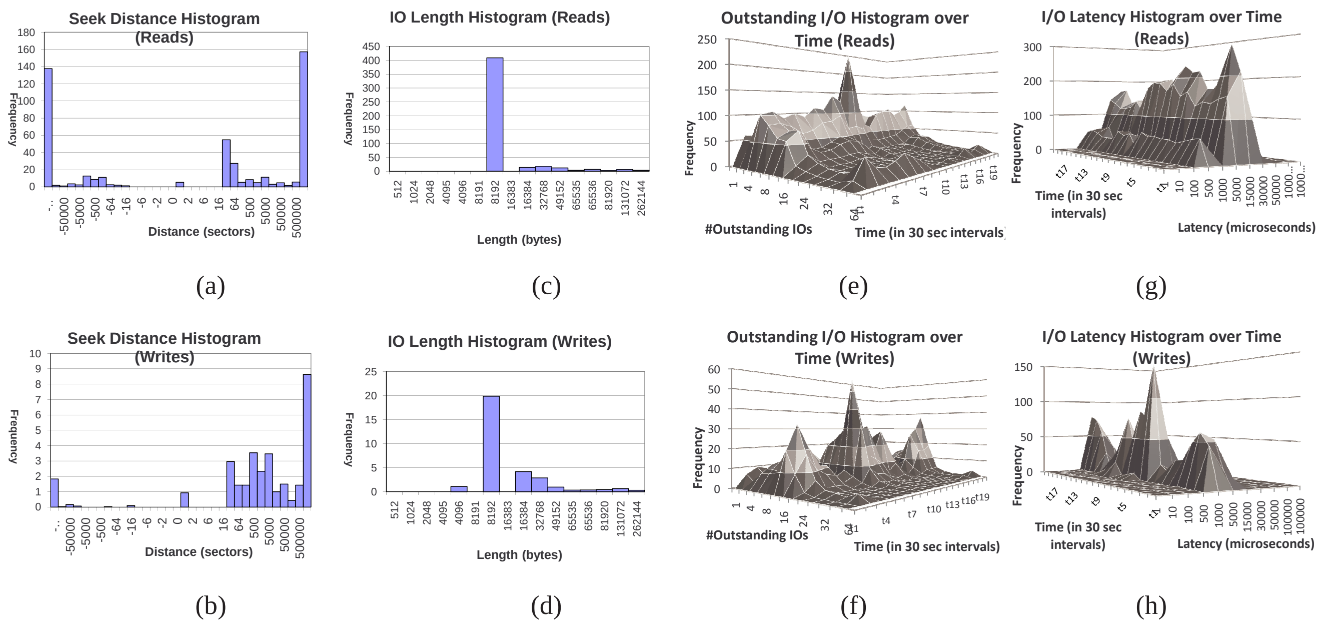


Figure 5. Microsoft Exchange Data disk characteristics. OIO and I/O latency histograms are shown over time to illustrate the variability in the workload over time.

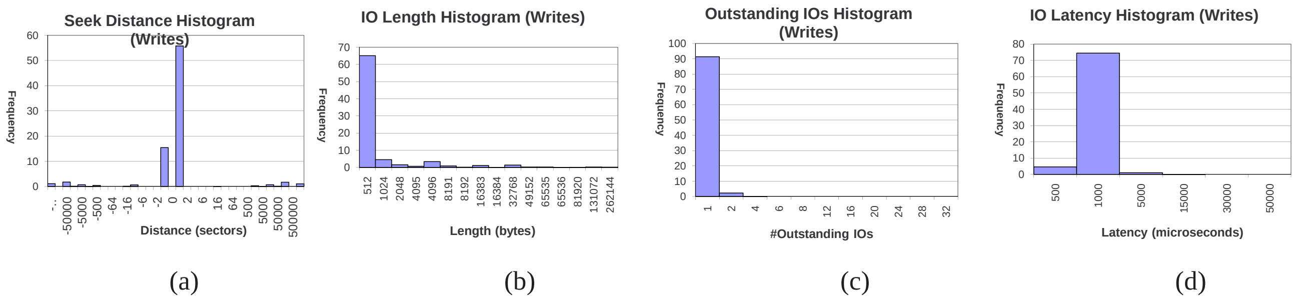


Figure 6. Microsoft Exchange Redo Log disk characteristics

of application. A typical decision support application usually processes large amounts of data in a batch in order to produce nightly, weekly (i.e. periodic) reports. The application usually generates a lot of reads that are sequential in nature. This is one of the main reasons for us to choose this application, so that we can compare it with others with random access pattern. The benchmark takes number of users, minimum and maximum delay between transactions, and percentage of each type of transactions as input parameters. For this testing we chose a set of six queries (i.e. sales roll up by month, top sales by quarter, period to period sales comparison) with equal percentage. The database was divided into data and index LUNs. The data and index were placed on separate 5 disk RAID 5 LUNs with capacity 75 and 20GB respectively. Figure 9 shows the results for data LUN for this workload. The data seeks show an interesting bi-modal distribution, where they are either sequential or are 1-2 MB apart. There are very few seeks that are further away. This is expected since the workload tries to go through complete tables instead of a set of rows. Writes are almost non-existent in the workload. The IO sizes are 8KB and latency is mostly 500 microseconds. Few requests see a high latency of 50-100 ms. Index LUN (not shown) interestingly doesn't exhibit much IO activity. This is again because most queries require full table reads and don't need to use index for specific data location. Redo logs again generated sequential writes of size from 512 bytes to 16KB and very few large writes of size $>$ 16KB. The redo log IO latency is mainly 0.5 to 1 ms with worst case latency of 100 ms.

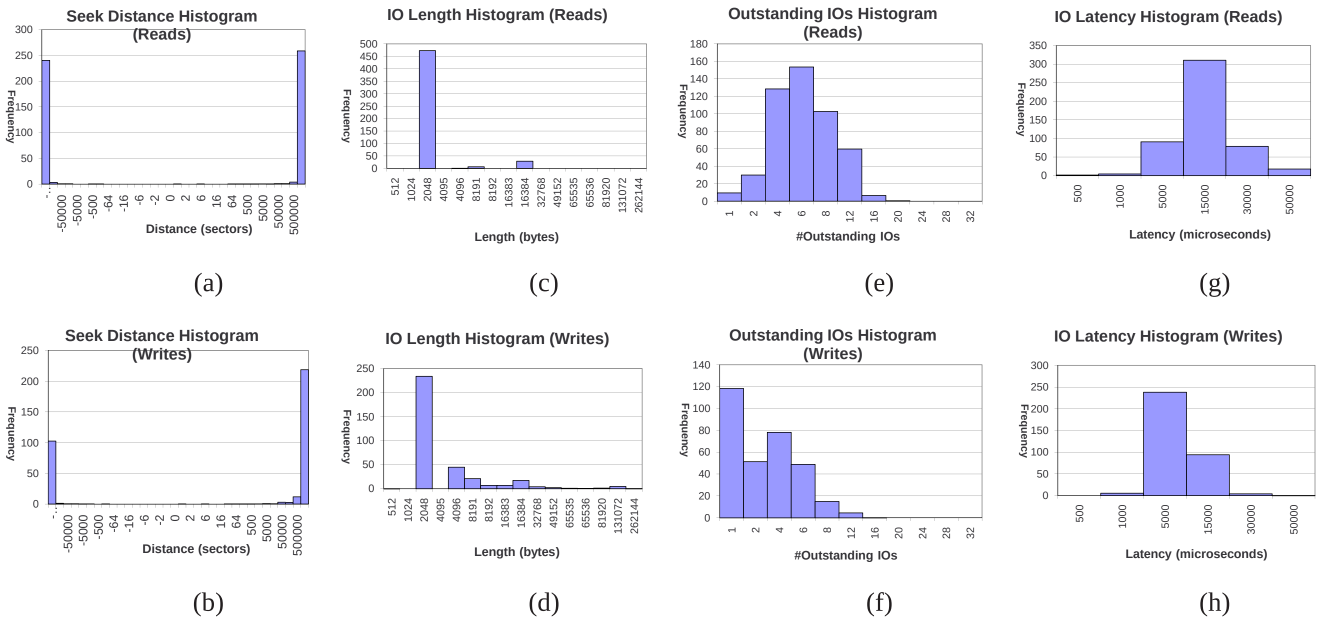


Figure 7. TPC-C (Oracle RDBMS) Data disk characteristics

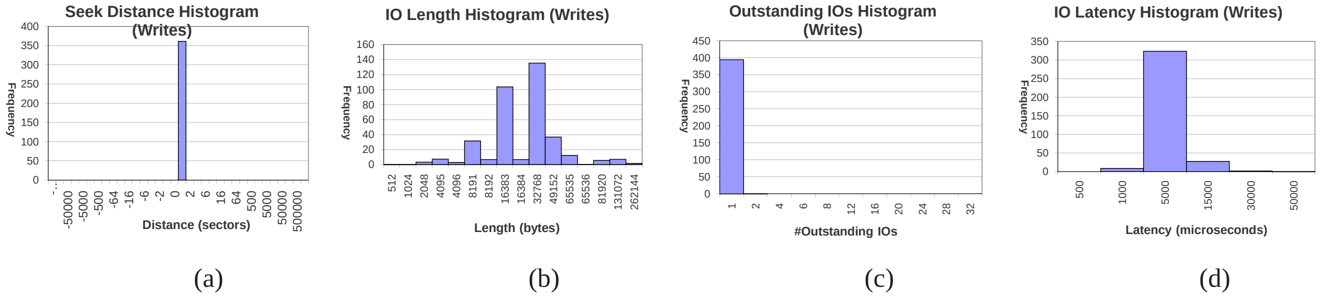


Figure 8. TPC-C (Oracle RDBMS) Redo Log disk characteristics

3.5 Swingbench OLTP (Oracle)

Swingbench OLTP (officially called order entry workload) represents an online transaction processing type of application. It takes number of users, think time between transactions, and a set of transactions as input to generate a workload. The transaction set can be selected from a pool of pre-defined types and a percentage can be attached to each of them. This allows us to vary the workload with different degree of CPU and IO needs. For this workload, we used 30 users, zero think time between requests and we used all five transaction types (i.e. new customer registration, browse products, order products, process orders and browse orders with variable percentages). Each user issues a transaction in a closed loop. The timeout for a transaction is set to 60 seconds. We noticed that 30 users was sufficient to keep the CPU fully utilized. Adding more users just increased the overall latency without further improving the overall transactions per second. The data files related to the workload are divided into two components - data and index. The database itself has system and redo logs. In this study, we focus on data, index and redo logs only because accesses to system log are very few and intermittent. Figures 10, 11 show the results for the data and index components. The seek distance histograms again showed mostly random access with some locality in writes. IO sizes are always 8KB and latency of reads vary from 5 to 15 ms, whereas the latency of writes varies from 1 to 5 ms. Index LUN shows the similar characteristics with even higher degree of randomness in both reads and writes. This is expected by the very nature of index lookups. IO sizes are again 8KB and latency shows a very distinct tri-modal distribution: 33%

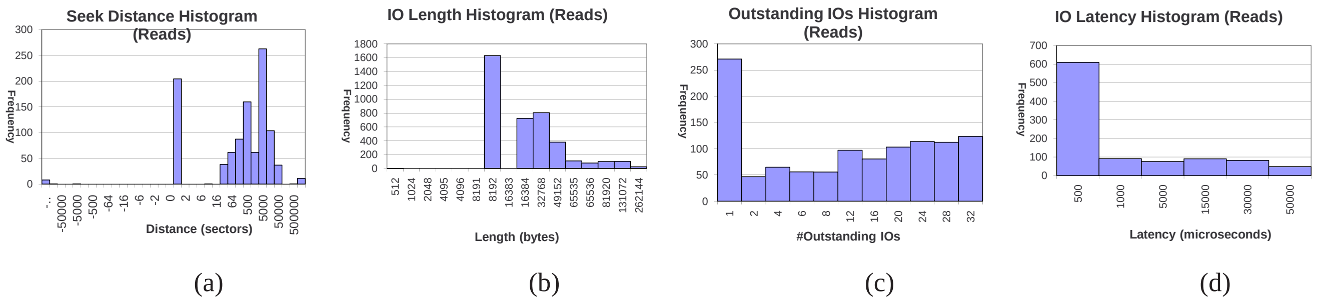


Figure 9. Swingbench Decision Support System (DSS) Benchmark Data disk characteristics

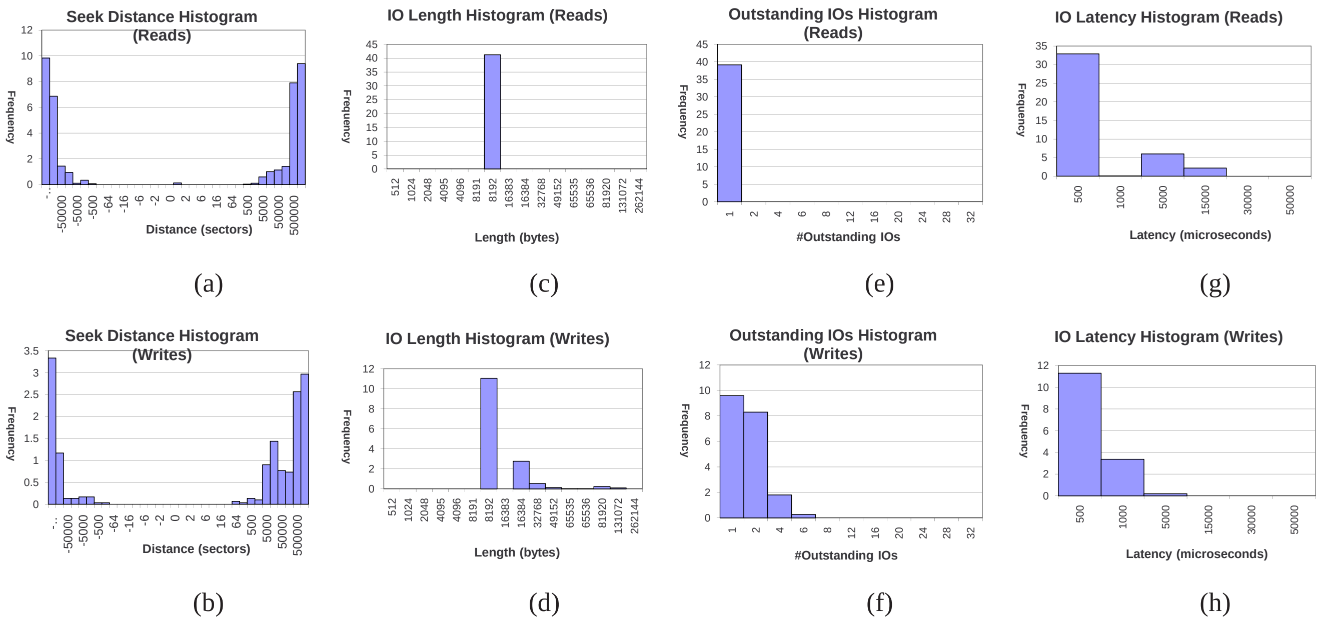


Figure 10. Swingbench OLTP Data disk characteristics

reads finish within 500 microseconds, 33% within 5 ms and remaining within 15 ms. IO latency for writes mostly varies from 1 to 5 ms.

Redo log data (not shown) has the familiar pattern for variable sized (512 bytes to 48 KB) sequential writes with average latency of 0.5 to 1 ms.

4 Impact of Storage Consolidation

In order to evaluate the impact of sharing devices on different applications, we conducted experiments with a combination of random and sequential workloads sharing the same IO devices. These workloads were run in separate VMs on top of VMware ESX Server 3.5 [21].

We ran two workloads each on an isolated 3-disk RAID 5 group. Then we ran them together on 6-disk RAID 5 group to evaluate the gain or loss from sharing. This comparison keeps the total number of disks identical which ensure that the baseline available performance from sheer spindle count, cost, power and reliability profiles don't change dramatically.

First we tested two workloads that both exhibit random IO seek patterns: DVDStore and Swingbench OLTP. To evaluate the effect of sharing, we ran them in isolation and then together. Table 1 shows the comparison in terms of average IOPS and 90th percentile latency for the two cases. The IOPS achieved by each applicaton remains largely the same in the isolated and shared cases (some variance is within the margin of experimental error). The application metric of transactions per minute (TPM) shows similar be-

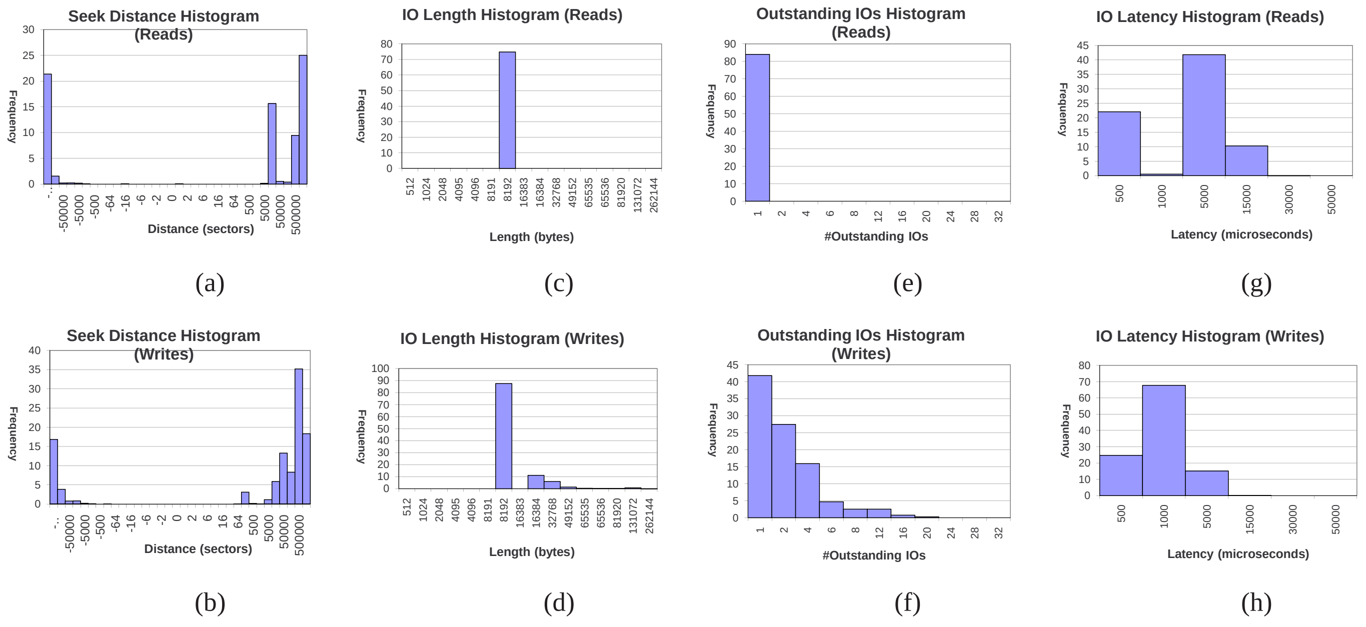


Figure 11. Swingbench OLTP Index disk characteristics

havior. *Note that latency is reduced in case of sharing because there are higher number of disks to absorb the bursts*.

Random workloads are additive in their load and do not destructively interfere with each other. This data clearly shows that as long as the underlying performance from spindle count is scaled up to match, overall achieved performance will also scale as workloads are added. The improvement in latency alone makes the case for consolidating random IO streams onto unified sets of spindles. Another significant benefit of moving away from hard partitioning is work conservation: if one workload is experiencing a transient burst of IO, it is able to expand its footprint out to a much larger number of disks thus allowing absorbtion of spare capacity from other workloads resulting in much higher burst scaling.

Next we tested two workloads one with random and another with sequential IO access pattern: DVDStore and Swingbench Decision support system. To evaluate the effect of sharing, we first ran them with separate 3 disk LUNs. Then we put them together on 6 disk LUNs to see the change in performance. As shown in Table 2, latency improved for both workloads because of higher number of disks. Whereas the throughput of the random workload is improved, the sequential workload suffers a 30% degradation. (Note that the DSS workload transactions are very large and therefore the number of completed ones in our 10 minute test period is small.) In case of purely sequential IO streams smaller IO requests from the applications are coalesced into fewer larger IO requests (>256K) that are serviced by underlying storage. This results in higher throughput with higher access latencies. In case of interleaved random and sequential IO access smaller sequential IO requests are not coalesced (remain 8K in our case) as the IO access pattern as seen by storage is no longer purely sequential. Thus even though the acces latencies are smaller in this case, throughput drops. This data suggests that mixing sequential workloads with random ones needs to be evaluated carefully. In this respect, consider that most realistic workloads are actually random and that our characterization technique can help identify the sequential ones that might need different treatment. Furthermore, we believe that an IO scheduler can be employed to restore the sequential workload to its isolated performance thus eliminating this limitation. Finally, the improved latency can help handle bursts for either storage workload.

5 Burst Analysis

In this section we look at arrival pattern and burst characteristics of workloads at a fine grained level. This data is lost from previous results due to aggregation over longer time intervals. We collected the complete trace of IO requests coming from a VM in ESX server using *vscsiStats*. The data consists of IO length, location, arrival time and device Id. Figure 12 shows the arrivals in every second for a 300 second window within the runs of DSS, OLTP and DVD store workloads. This shows that all workloads are very bursty and the rate can change by as much as 50%

Workload	LUN configuration	IOPS	IO latency	Application Metric
DVDStore	2+1	130	100	6132 TPM
OLTP	2+1	141	30	5723 TPM
DVDStore (Shared)	5+1	144	30	7630 TPM
OLTP (Shared)	5+1	135	30	5718 TPM

Table 1. Comparison of DVDStore and OLTP when run in isolation and shared mode

Workload	LUN configuration	Throughput	95% tile latency	Application Metric
DVDStore	2+1	130 IOPS	100	6132 TPM
DSS	2+1	44 MB/s	30	6 completed transactions
DVDStore (Shared)	5+1	164 IOPS	15	7630 TPM
DSS (Shared)	5+1	31 MB/s	1	3 completed transactions

Table 2. Comparison of DVDStore and DSS when run in isolation and shared mode

within a few seconds. This observation reinforces the case for consolidation: previous data has shown that having a larger number of disks available lowers IO latencies thus creating some extra headroom for the bursty workloads.

6 Related Work

The work related to workload characterization and placement falls into two broad categories: first are the tools for automation and reconfiguration of storage infrastructure and second are the studies on workload characterization, storage array modeling.

Hippodrome [7] tries to automate storage system configuration by iterating over three stages: analyze workload, design system and implement design. Similarly Minerva [6] uses a declarative specification of application requirements and device capabilities to solve a constraint based optimization problem for storage system design. The main issue with these is the lack of good models for storage systems that can be easily used for various optimizations. Also they look at the overall workload and capacity issues, whereas we try to provide more fine grained information about the various components of the workloads. We believe that such tools are complimentary to our work and would work better with the knowledge gained by our workload characterizations.

Many previous studies have looked into file system access patterns [5, 15, 18] and IO patterns at disks [16], mostly in context of user workloads in an organization. Although these are quite useful for file system level analysis, we mainly focus on real applications that are heavily used in businesses today for day to day operations. Also many commercial workloads bypass some of the file system level functionality and do their own cache management. Other studies have looked into IO access patterns for specific supercomputing applications and specific machines [13, 14, 19].

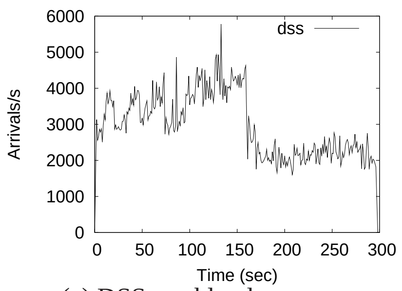
Studying RAID performance for different workloads has been an active area of research. Chen et. al. [8–10] have shown that stripe size, IO sizes and RAID level can have a wide impact on overall performance. They also showed that applications with large accesses do better with RAID 5 and application using large number of smaller accesses such as OLTP, perform better using mirrored RAID. We instead study the workload characteristics and the impact of device sharing for various applications, which is not considered in previous studies.

Researchers have also tried to model disk drives [17] and storage arrays [11, 20] to automate the analysis and prediction of workload behaviors. This is especially hard given the device characteristics of disks and all the complexity built into an array in terms of service processors, buses, controller caches etc. Getting better device models and workload charactistics should work in conjunction with doing better capacity planning.

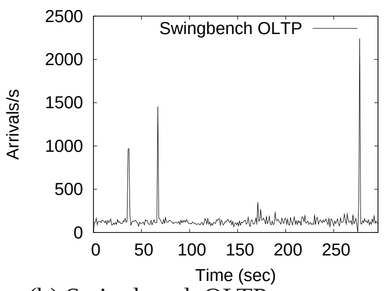
Most exisitng commercial products such as EMC CLARiiON, HP SureStore, IBM Tivoli, Equallogic PS series provide a layer of abstraction for storage devices and management tools to partition and configure devices as needed. Some of them also do automatic load balancing by moving data within or across arrays. However, configuring storage arrays using these tools is still a difficult task. We provide some insights and guidelines in this paper to configure various workloads, so as to acheive better performance and utilization of resources.

7 Conclusions and Future Work

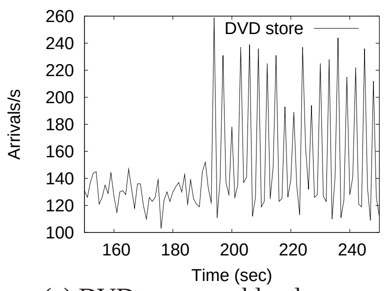
This paper presented a survey of IO characerisitcs of real application workloads: OLTP, DSS and mail server. We studied the different components (i.e. data, index and logs) of these workloads separately, and collected detailed infor-



(a) DSS workload



(b) Swingbench OLTP



(c) DVD store workload

Figure 12. Arrivals/sec observed by various workloads showing the burstiness of them. The arrivals change by almost 50% over a period of few seconds.

mation about their seek pattern, IO sizes, burstiness, IO latency and so on. Such a study is very helpful in enabling users to generate realistic impressions of real workloads for their research. We also studied the impact of storage consolidation on workloads and showed that putting random workloads together can help improve overall utilization and reduce latency. Consolidation provides benefits of statistical multiplexing and helps in absorbing bursts in these cases. However, putting a random and a sequential workload together can lead to performance degradation for the sequential workload. Based on these findings, we are looking at automating this task of consolidation and workload placement across LUNs for better load balancing and hotspot avoidance.

References

[1] Dvd store. http://www.delltechcenter.com/page/DVD+store.

[2] Microsoft exchange server 2007. http://www.microsoft.com/exchange/default.mspx.

[3] Oracle inc. www.oracle.com.

[4] Swingbench. http://www.dominicgiles.com/swingbench.html.

[5] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. *Proceedings of the 5th Conference on File and Storage Technologies (FAST '07)*, Feb 2007.

[6] G. A. Alvarez and et al. Minerva: an automated resource provisioning tool for large-scale storage systems. In *ACM Transactions on Computer Systems*, pages 483–518, November 2001.

[7] E. Anderson and et al. Hippodrome: running circles around storage administration. In *Proc. of Conf. on File and Storage Technology (FAST'02)*, pages 175–188, January 2002.

[8] P. M. Chen, G. A. Gibson, R. H. Katz, and D. A. Patterson. An evaluation of redundant arrays of disks using an amdahl 5890. *SIGMETRICS Perform. Eval. Rev.*, 18(1):74–85, 1990.

[9] P. M. Chen and E. K. Lee. Striping in a raid level 5 disk array. In *SIGMETRICS '95/PERFORMANCE '95: Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 136–145, New York, NY, USA, 1995. ACM.

[10] P. M. Chen and D. A. Patterson. Maximizing performance in a striped disk array. *SIGARCH Comput. Archit. News*, 18(3a):322–331, 1990.

[11] T. Denehy, J. Bent, F. Popovici, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. Deconstructing storage arrays, 2004.

[12] I. Ahmad et. al. An Analysis of Disk Performance in VMware ESX Server Virtual Machines. In *IEEE Int. Workshop on Workload Characterization (WWC-6)*, Oct 2003.

[13] E. L. Miller and R. H. Katz. Analyzing the I/O behavior of supercomputer applications. *Eleventh IEEE Symp. on Mass Storage Sys.*, page 51, 7-10 1991.

[14] J. Oly and D. Reed. Markov model prediction of I/O request for scientific application. In *Proc. of the 2002 International Conference on Supercomputing*, Jun 2002.

[15] J. K. Ousterhout, H. D. Costa, D. Harrison, J. A. Kunze, M. Kupfer, and J. G. Thompson. A trace-driven analysis of the unix 4.2 bsd file system. *SIGOPS Oper. Syst. Rev.*, 19(5):15–24, 1985.

[16] C. Ruemmler and J. Wilkes. UNIX disk access patterns. In *Usenix Conference*, pages 405–420, Winter 1993.

[17] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, 1994.

[18] M. Satyanarayanan. A study of file sizes and functional lifetimes. *SIGOPS Oper. Syst. Rev.*, 15(5):96–108, 1981.

[19] M. Uysal, A. Acharya, and J. Saltz. Requirements of I/O systems for parallel machines: An application-driven study. Technical Report CS-TR-3802, 1997.

[20] M. Uysal, G. Alvarez, and A. Merchant. A modular, analytical throughput model for modern disk arrays, August 2001.

[21] VMware, Inc. *Introduction to VMware Infrastructure*. 2007. http://www.vmware.com/support/pubs/.