EMC XTREMIO OPTIMIZED FLASH STORAGE FOR ORACLE DATABASES

ABSTRACT

The goal of the paper is to help the reader understand Oracle performance bottlenecks and how XtremIO can help address these performance issues.

June, 2014

EMC WHITE PAPER

EMC²

Part Number H13174

EXECUTIVE SUMMARY

Physical I/O patterns generated by Oracle Database workloads are well understood. The predictable nature of these I/O characteristics have historically enabled platform vendors to implement widely varying I/O acceleration technologies including prefetching, coalescing transfers, tiering, caching and even I/O elimination. However, the key presumption central to all of these acceleration technologies is that there is an identifiable active data set. While it is true that Oracle Database workloads generally settle on an active data set, the active data set for a workload is seldom static—it tends to move based on easily understood factors such as data aging or business workflow (e.g., "month-end processing") and even the data source itself. Identifying the current active data set and keeping up with movement of the active data set is complex and time consuming due to variability in workloads, workload types, and number of workloads. Storage administrators constantly chase the performance hotspots caused by the active dataset.

All-Flash Arrays (AFAs) based on flash media can completely eliminate the need to identify the active dataset because of the ability of flash to service any part of a larger data set equally. But not all AFAs are created equal.

Even though numerous AFAs have come to market, obtaining the best performance required by databases is challenging. The challenge isn't just limited to performance. Modern storage arrays offer a wide variety of features such as deduplication, snapshots, clones, thin provisioning, and replication. These features are built on top of the underlying disk management engine, and are based on the same rules and limitations favoring sequential I/O. Simply substituting flash for hard drives won't break these features, but neither will it enhance them.

EMC has developed a new class of enterprise data storage system, XtremIO flash array, which is based entirely on flash media. XtremIO's approach was not simply to substitute flash in an existing storage controller design or software stack, but rather to engineer an entirely new array from the ground-up to unlock flash's full performance potential and deliver array-based capabilities that are unprecidented in the context of current storage systems.

This paper will help the reader understand Oracle Database performance bottlenecks and how XtremIO AFAs can help address such bottlenecks with its unique capability to deal with constant variance in the IO profile and load levels. We demonstrate that it takes a highly flash-optimized architecture to ensure the best Oracle Database user experience.

TABLE OF CONTENTS

INTRODUCTION

One important trend in the IT industry is the move into the era of Low Latency Commercial Computing. For many years, technology improvements in CPU have greatly outpaced improvements in storage. This trend is not tenable in light of IT deployment models such as consolidation and virtualization. IT can no longer suffer high-latency, unpredictable storage performance. To that end, the storage market is quickly becoming saturated with All-Flash-Array (AFA) technology offerings. However, AFAs are not universally equal.

This document offers insights and case studies focused on OLTP performance characterization of EMC XtremIO. Readers will gain an understanding of the principles of Oracle Databases that impact overall application performance. It also discusses why AFA technology is important in solving Oracle performance issues. It will also touch upon different architectural approaches and trade-offs for AFAs that affect behavior under different IO patterns. Finally, a case study with test results showing how XtremIO delivers consistent Oracle Database performance under all conditions is presented.

CHASING ACTIVE DATA SET

Physical I/O patterns generated by Oracle Database workloads are well understood because they haven't changed for decades. The understandable, predictable nature of these I/O characteristics have historically enabled platform vendors to implement widely varying I/O acceleration technologies including prefetching, coalescing transfers, tiering, caching and even I/O elimination. However, to a certain extent, the key presumption central to all of these acceleration technologies is that there is an active data set and that it is identifiable.

It's true, Oracle Database workloads generally settle on an active data set. However, the active data set for a workload is seldom static—it tends to move based on easily understood factors such as data aging or business workflow (e.g. "month-end processing"). Identifying the current active data set—or even responding to movement of the active data set—is also a well-understood science and that is why historical platform acceleration approaches could be quite effective. However, would these techniques work in the era of consolidation?

THE BIRTH OF THE ALL-FLASH ARRAY

Modern Oracle Database deployments seldom dedicate a single storage device to a single database. The most common approach is to consolidate many database instances into as few, powerful servers as possible and use shared storage. There simply isn't enough physical space in IT datacenters to allow for any sort of one-to-one provisioning. This is not bad by definition, but the paradigm presents challenges—particularly when dealing with storage performance needs. This is nothing new. IT professionals have been identifying storage performance problems on shared storage and taking action for years. We all know the drill: identify the "hot" data (a.k.a. active data set) and take action.

It's simple to identify the active data set of a single Oracle Database, but what about 10 databases? What about 100 or 1000 databases? The answer is quite simple—there is no single active data set! For most Oracle Database shops, there are many active data sets—or aggregate group active data sets.

To ensure predictable low-latency, high-bandwidth storage for Oracle there is but one option—the AFA. With AFA technology one can realistically consider the entirety of their storage optimal for all data—not just the active data set of a bygone era.

But are all of the AFA products on the market created equal? No, they are not.

ALL-FLASH ARRAYS ARE NOT CREATED EQUAL

Enterprise storage arrays are highly sophisticated systems that have evolved over decades to provide levels of performance, reliability, and functionality unavailable in the underlying hard disk drives. A significant aspect of the engineering work to make this possible is designing the array hardware and software stack to extract the most capability out of hard drives.

All-Flash Arrays (AFAs) based on flash media, not bound by hard disk drive limitations, have come to market. However, obtaining superior performance from an SSD, much less in an enterprise storage array filled with them, is challenging. Existing storage arrays designed to 'sequentialize' workloads are simply optimized along a dimension that is no longer relevant because there is no "seek penalty" on NAND flash. The array controller designs must be entirely rethought to exploit the random access flash media, or they simply become the bottleneck. The challenge isn't just limited to performance. Modern storage arrays offer a wide variety of features such as deduplication, snapshots, clones, thin provisioning, and replication. These features are built on top of the underlying disk management engine, and are based on the same rules and limitations favoring sequential I/O. Simply substituting flash for hard drives won't break these features, but neither will it enhance them.

XtremIO has developed a new class of enterprise data storage system based entirely on flash media. XtremIO's approach was not simply to substitute flash in an existing storage controller design or software stack, but rather to engineer an entirely new array from the ground-up. This allows XtremIO to unlock flash's full performance potential and deliver array-based capabilities that are unprecidented in the context of current storage systems.

AFA vendors have adopted different architectures—scale up (scaling of performance and capacity by adding by adding disk shelves) and scale out (linear scaling of performance and capacity together using a unit containing both controllers and disk shelves). Different data protection schemes have been adopted—most re-used RAID systems developed for HDDs, while very few have created flash-optimized data protection schemes. These architectural choices impact not only data protection and effective capacity, but also sustainability of service times (latency) and throughput.

immediate

[ih-mee-dee-it] adjective

1. occurring or accomplished without delay; instant: an immediate reply

Indeed, application users expect information immediately and for that reason, time is all that matters from an application performance perspective. Skilled Oracle practitioners know this as the Oracle Database Performance Method based on Time Model statistics—or Time Model Method for brevity.

TIME MODEL

Mastering Oracle Database performance requires an understanding of the fundamentals of the Time Model Method:

- **DB Time**: The wall-clock time of database requests by active sessions attached to an Oracle Database instance (e.g., a PL/SQL package execution or simple SQL statements) is known as DB Time. DB Time consists of two time domains:
    - **DB CPU**: The wall-clock time in which Oracle Database processes are in a running state (from the operating system point of view) on behalf of active sessions.
    - **Wait Time**: The wall-clock time Oracle Database processes are not in a running state (from the operating system point of view) in spite of active sessions. An example of Wait Time is a session waiting for a random physical read from a data file.
- **Idle Time**: The wall-clock time in which Oracle Database has been booted up, but is not processing requests for active sessions is known as Idle Time.
- **Elapsed Time**: This is any period of wall clock time covered by a sampling of Oracle Database internal performance metrics (e.g., via Active Workload Repository scripts).

The Oracle Database is multiprocessing software—and runs on multiprocessor systems—therefore one should expect DB Time to be much larger than Elapsed Time on an active Oracle Database. For example, consider an Oracle Database instance with 400 active sessions that have each waited for 1 second for the results from a SQL statement. Such a situation would accumulate 400 seconds of DB Time to account for. To further the example, consider the same 400 seconds of DB Time on a server with 16 processors. In such a situation, one could expect to see DB Time broken into as much as 16 seconds of DB CPU and roughly 384 seconds of Wait Time per second.

Understanding of Time Model is key to optimizing application performance.

WAIT EVENT ANALYSIS TO CHOSE THE RIGHT STORAGE

Generally speaking, the goal of Time Model tuning is to identify and eliminate Wait Time. Once a workload consists purely of DB CPU there are no acceleration options other than provisioning more processor bandwidth or changing the work Oracle Database is being tasked with (e.g., SQL tuning, reducing calls to the database via application tier caching, etc.)

The Time Model Method can also be used to identify Oracle Database workloads that are good candidates for acceleration through the use of faster storage—such as EMC XtremIO. This can be done with Automatic Workload Repository (AWR) reporting or even Oracle Enterprise Manager (a.k.a. EM). For example, consider the graph in Figure 1, which shows an EM timeline that categorizes Time Model waits by active sessions into classes ranging from User I/O to Cluster class waits. The example shows the EM webpage has highlighted User I/O class waits in yellow. User I/O class waits are suffered by active sessions reading or writing data while executing SQL statements.

The workload depicted in Figure 1 shows a clear-cut case for faster storage since I/O waits ranked as the highest class of waits.

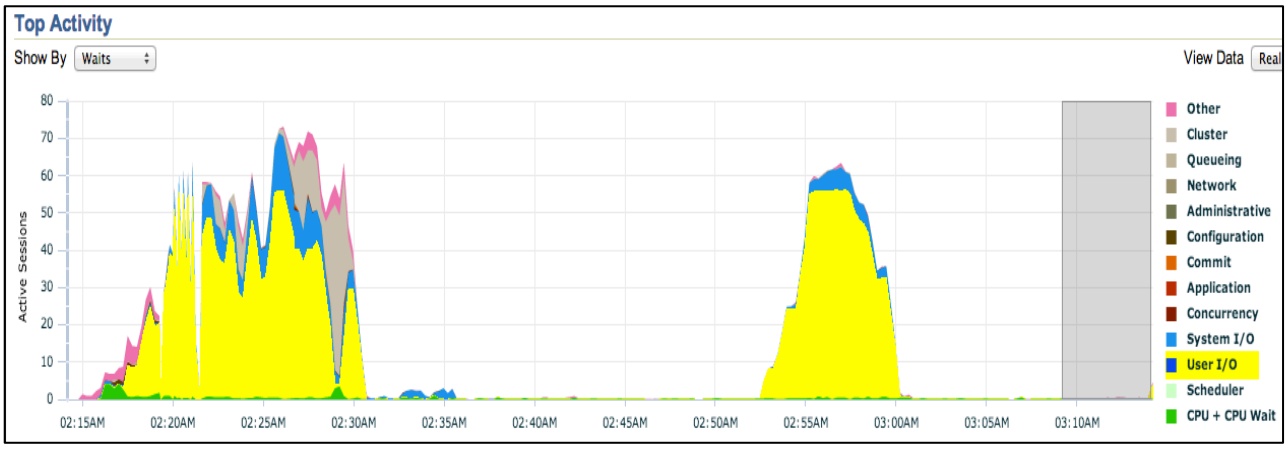

Figure 1: IO Intensive Oracle Database Workload

Details about specific types of User I/O waits can be seen in Automatic Workload Repository (AWR) reports. For instance, Figure 2 shows an example of a workload prime for acceleration from low-latency storage since roughly 41% (=14.68 + 13 + 8.33 + 4.96) of DB Time is languishing on physical I/O (User I/O class) with service times ranging from 4 to 17 milliseconds.

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		447		17.06	
db file sequential read	109,209	384	4	14.68	User I/O
direct path read	49,001	340	7	13.00	User I/O
db file scattered read	46,315	218	5	8.33	User I/O
direct path read temp	7,715	130	17	4.96	User I/O

Figure 2: Time Model Top 5 Timed Waits for an I/O Bound System

The Oracle Time Model Method also plays a crucial role in helping administrators determine when storage is not a significant performance problem. For example, Figure 3 shows an AWR snippet of an application that is both CPU-bound and suffering internal Oracle Database contention[1].

The workload profile shown in Figure 3 cannot be improved by accelerating storage.

[1] Concurrency Class waits are internal database server waits. Note, however, that CPU saturation is generally the root cause of these waits. http://docs.oracle.com/cd/E11882_01/server.112/e40402/waitevents001.htm#BGGHJGII.

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		6,905		89.86	
latch: row cache objects	3,016,874	454	0	5.91	Concurrency
library cache: mutex X	2,537,405	316	0	4.11	Concurrency
cursor: mutex X	30,373	17	1	0.23	Concurrency
cursor: mutex S	15,014	3	0	0.04	Concurrency

Figure 3: Time Model Top 5 Timed Waits for a CPU Bound System

OLTP SERVICE TIMES CHARACTERIZATION ON EMC XTREMIO

With EMC XtremIO, it is very easy to provide an example of what Time Model data should look like in the era of low-latency commercial computing. To illustrate the point, EMC engineers conducted a test with the following characteristics:

- Oracle Database 11g Release 2 was installed on a 2-socket Intel Xeon E5-2600 server[2] running Oracle Linux 6.
- The server was connected to a single X-Brick EMC XtremIO AFA via 4 paths of 8Gbps Fibre Channel.
- Four LUNs were provisioned from the array.
- The host was configured with Device Mapper in accordance with My Oracle Support Doc ID: 1594584.1. The LUNs were then added as disks in an Automatic Storage Management (ASM) disk group created with external redundancy.
- As pre-conditioning exercise for the array, a 5 terabyte tablespace—based on 8KB blocks—was created.
- Immediately after the tablespace was initialized, 1 terabyte of application data was loaded. No form of Oracle compression was used.
- After loading the data a 1-hour test of an OLTP-style SQL workload (SLOB[3]) was initiated with a SQL mix of 75% SELECT, 25% UPDATE. There were 96 dedicated connections to the instance each submitting SQL requests to the database server with zero think-time.

The primary goal of this testing was to increase physical I/O rates only to the point at which average service times remained at or below 1 millisecond as measured by the Oracle Database wait interface. The simplest way to show the runtime characteristics of the 1-hour test is to examine sections of the text AWR report. Figure 4 shows the AWR report.

DB Name	DB Id	Instance	Inst Num	Startup Time	Release	RAC
[redacted]	3564722959	[redacted]	1	30-Jan-14 00:30	11.2.0.3.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory(GB)
[redacted]	Linux x86 64-bit	16	16	2	31.42

	Snap Id	Snap Time	Sessions	Curs/Sess
Begin Snap:	81	30-Jan-14 00:31:05	131	.5
End Snap:	82	30-Jan-14 01:31:12	37	.6
Elapsed:		60.12 (mins)		
DB Time:		5,763.84 (mins)		

Cache Sizes	Begin	End		
Buffer Cache:	1,024M	1,024M	Std Block Size:	8K
Shared Pool Size:	8,192M	8,192M	Log Buffer:	151,696K

Load Profile	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	95.9	0.0	0.00	306.04
DB CPU(s):	9.3	0.0	0.00	29.72
Redo size:	21,834,010.5	3,010.2		
Logical reads:	168,425.0	23.2		
Block changes:	44,359.0	6.1		
Physical reads:	83,114.8	11.5		
Physical writes:	15,754.8	2.2		
User calls:	0.3	0.0		
Parses:	0.5	0.0		
Hard parses:	0.1	0.0		
W/A MB processed:	0.0	0.0		
Logons:	0.0	0.0		
Executes:	29,014.9	4.0		
Rollbacks:	0.0	0.0		
Transactions:	7,253.5			

Figure 4: AWR Load Profile Section - Single X-Brick XtremIO Array

[2] This particular lab system had Intel Simultaneous Multithreading (SMT) disabled. AWR reports will show 2 sockets, 16 cores and 16 CPUs.
[3] For the sake of openness the widely popular and easily obtained Silly Little Oracle Benchmark (SLOB) kit was used to test the array. For more information on SLOB please refer to https://community.emc.com/docs/DOC-33564.

The following observations can be made about runtime characteristics:

- The AWR sample period was 60.12 minutes (3,607 seconds). The Oracle Database block buffer cache was 1024MB (sufficient to cache only .09% of the active data set which was 1 terabyte)
- Time Model Method characteristics were as follows:
  - DB Time. 95.9 seconds which maps directly to the 96 zero think-time sessions
  - DB CPU. 9.3 seconds. The host had 16 CPUs, therefore the host processor utilization level was 58% busy (9.3/16)
- 29,014 SQL Executions issued to the database instance per second
- The database instance physical I/O characteristics were as follows:
  - Physical Reads. 83,114 physical read requests (on average), per second, to the EMC XtremIO array
  - Physical Writes. 15,754 physical write requests (on average), per second, to the EMC XtremIO array
- The transaction logging rate was 20.8 MiB (21,834,010 bytes) per second—or roughly 73 GB/h

Now that the runtime characteristics of the test are understood, let us look at Figure 5, which shows Top 5 Timed Wait Events (some components of the Time Model) section of the AWR report.

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	299,784,368	320,702	1	92.7	User I/O
DB CPU		33,584		9.7	
cursor: pin S	120,261	238	2	.1	Concurrenc
library cache lock	1,079	123	114	.0	Concurrenc
log file switch (private stran	2,282	24	11	.0	Configurat

Figure 5: AWR Timed Wait Events - Single X-Brick XtremIO Array

In addition to showing the ranking of crucial wait events, Figure 5 also shows wall-clock average wait times as well. Here are some important observations:

This OLTP-style workload encountered User I/O class waits accounting for 92.7% of all wait time—typical of an OLTP workload profile. It is very clear that this kind of workload simply needs faster storage.

However, what if the storage is already very fast? If storage is already very fast, and service level agreements are being met, one can simply claim success and move along to the next IT performance challenge they are facing. Of course the definition of "very fast" is relative, but in terms of contemporary flash storage, millisecond latency is generally considered the gold standard. In this case, as Figure 5 shows, there were 299,784,368 total waits for db file sequential read[4] which cost the active sessions 320,702 seconds of wait time. This works out to 1.07 millisecond average wait time for the top wait event (320,702/299,784,368).

Averages are not a sufficient metric of suitable low-latency storage for Oracle Database. One must also examine the AWR histogram data to see if there are significant outliers. Figure 6 shows that latency for both single-block random and multi-block noncontiguous reads fell

[4] Routinely misunderstood, the db file sequential read event accounts for single block random I/O calls to the operating system. The moniker includes the word sequential because these I/O calls are made one after the other—sequentially. They are, in fact, blocking single-block reads from random file offsets.

below 1 millisecond 65% and 60% of the time respectively—and over 90% of all such I/O completed in less than 2 milliseconds. These are key performance indicators because these I/O events fall within the transaction scope.

		% of Waits							
Event	Total Waits	<1ms	<2ms	<4ms	<8ms	<16ms	<32ms	<=1s	>1s
[...lines deleted...]									
db file parallel write	666K	98.1	1.6	.3	.0	.0	.0		
db file scattered read	6358	64.9	33.3	1.4	.5				
db file sequential read	299.3	60.3	32.6	6.5	.4	.1	.0	.0	
[...lines deleted...]									
direct path read	71	59.2	32.4	7.0		1.4			
direct path write	67	85.1	11.9	1.5	1.5				
[...lines deleted...]									
log file parallel write	2817.	56.7	37.4	5.5	.3	.0	.0		

Figure 6: Wait Event Histogram - Single X-Brick XtremIO Array

Transaction log (aka redo logging) writes are also a key performance indicator since these I/O operations are, of course, scoped within transactions. Figure 6 shows that more than 94% of all redo log writes completed in less than 2 milliseconds and only 0.3% of all of log writes took longer than 4 milliseconds.

SCALING ORACLE OLTP WORKLOADS WITH XTREMIO ARRAYS

The previous section covers typical EMC XtremIO performance characteristics with an Oracle OLTP-style I/O profile against a single X-Brick XtremIO array (the smallest system available). The amount of test data was 1 TB and the workload harness (SLOB) forces completely random access across all of the data loaded into the tablespaces. The I/O rate shown for the 75:25 read/write workload was 98,868 IOPS with very favorable service times measured by the Oracle wait interface. What would happen if the workload scaled up to achieve more I/O requests? In this section, we will examine how service levels can be kept within tolerable levels by scaling out the X-Brick cluster.

Current AFA technology offers various forms of scalability. Some manufacturers offer scaling from a single storage processor to two. EMC XtremIO, on the other hand, supports scaling from a single X-Brick to four X-Bricks at publication time of this document. The most important aspect of XtremIO scalability is the fact that it is active-active multi-controller scaling which means hosts with multipath I/O technology are able to issue I/O requests to the array in a manner that balances the requests across all of the controllers on all X-Bricks in the array. For example, an Oracle Database that fits in the usable capacity of a single X-Brick array can also be stored in an array with two X-Bricks for double the IOPS and double the I/O bandwidth.

SCALE FROM SINGLE X-BRICK TO DUAL X-BRICK ARRAY

To illustrate the scalable nature of XtremIO, EMC engineers created an Oracle Database 11g Real Application Clusters database on a two-node cluster attached to an XtremIO array with two X-Bricks. Next, the same schema and data discussed in the single X-Brick section above were loaded into the XtremIO array. Once the data was loaded, scalability was assessed via the SLOB workload test kit.

Figure 7 shows that the testing of the two X-Brick array was based on the same 60-minute test duration used in the single X-Brick array testing described in the previous section. Further, the Oracle Database block buffer cache was also configured with the same capacity on both Real Application Clusters nodes as the single X-Brick testing. The schemas, queries and data volumes were in fact identical between

the one X-Brick and two X-Brick testing.

Database Summary

Database Id	Name	RAC	Block Size	Snapshot Ids Begin	Snapshot Ids End	Number of Instances In Report	Number of Instances Total	Number of Hosts In Report	Number of Hosts Total	Report Total (minutes) DB time	Report Total (minutes) Elapsed time
4205402758	[redacted]	YES	8192	220	223	2	2	2	2	11,525.36	60.51

Database Instances Included In Report

-> Listed in order of instance number, I#

I#	Instance	Host	Startup	Begin Snap Time	End Snap Time	Release	Elapsed Time(min)	DB time(min)	Up Time(hrs)	Avg Active Sessions	Platform
1	[redacted]	xioraca	08-Mar-14 21:01	08-Mar-14 23:17	09-Mar-14 00:17	11.2.0.3.0	60.05	5,763.25	3.27	95.97	Linux x86 64-bi
2	[redacted]	xioracb	08-Mar-14 21:02	08-Mar-14 23:18	09-Mar-14 00:18	11.2.0.3.0	60.05	5,762.11	3.27	95.96	Linux x86 64-bi

Cache Sizes

-> All values are in Megabytes

-> Listed in order of instance number, I#

-> End values displayed only if different from Begin values

I#	Memory Target Begin	Memory Target End	Sga Target Begin	Sga Target End	DB Cache Begin	DB Cache End	Shared Pool Begin	Shared Pool End	Large Pool Begin	Large Pool End	Java Pool Begin	Java Pool End	Streams Pool Begin	Streams Pool End	PGA Target Begin	PGA Target End	Log Buffer
1					1,024		8,192				224				8,192		128.0
2					1,024		8,192				224				8,192		128.0
Avg					1,024		8,192				224				8,192		128.0
Min					1,024		8,192				224				8,192		128.0
Max					1,024		8,192				224				8,192		128.0

Figure 7: XtremIO Two-Node Oracle RAC Testing

Before we measure the effectiveness of the scaling, we need to make sure the workloads are similar in terms of IO behavior. We will use the ratio between SQL executions to database block changes and instance I/O events.

With an OLTP-style workload, like SLOB, physical accesses to blocks of disk are prefaced by an attempt to access the block in the SGA block buffer cache. These buffer accesses are called logical I/O. Workloads that exhibit a significant variance in logical I/O are not considered comparable.

Another important metric in comparing test results is the ratio of SQL executions to block changes. Since the SQL blend in both tests was 75% SELECT and 25% UPDATE, the ratio of SQL executions to block changes must show near parity.

Figure 8 validates that the two test configurations sustained nearly identical processing as per the key indicators below:

- The ratio of SQL executions to logical I/O was within 4%
- The ratio between SQL executions and physical I/O was within 2%
- The ratio between SQL executions and database block changes was within 1%

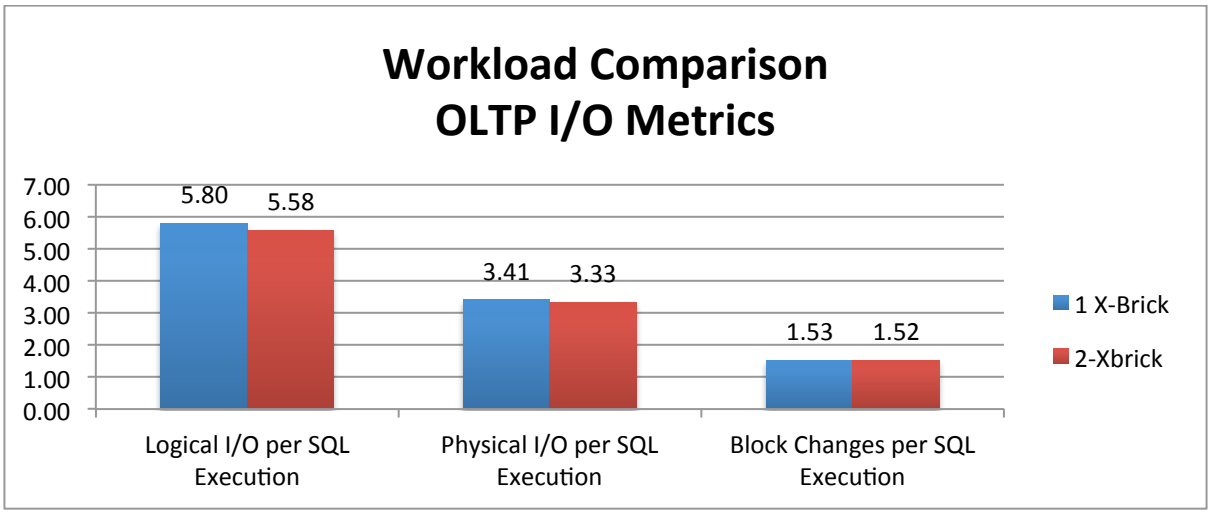

Figure 8: Validating Oracle Database OLTP Results - Single vs Dual X-Brick XtremIO Array

CONSISTENT AVERAGE WAIT TIMES

With the assurance of workload similarity in mind, we turn to the Time Model data from the 2 X-Brick array testing. Figure 9 shows that the AWR top wait event was single-block random read (User I/O class). The data shows how the *db file sequential read* wait events were satisfied, on average, in 961us (549,803.06 seconds /571,931,278 waits). Further, we see that both of the Real Application Clusters nodes experienced the same I/O latency because the maximum of averages on any host was 960us—as reported under the *Summary Avg Wait Time* heading. Those are, of course, random reads. What about writes?

One must always bear in mind that the critical write operations in an OLTP workload are transaction log (a.k.a. redo) writes. To that end, Figure 9 shows that active sessions suffered *log file parallel write* wait events at the rate of roughly 1,400 per second (5,156,057/3600), yet this aggregate wait time ranked fifth in the list of waits events. This is because the service times for the redo log flush operations languished only 1.1ms on average. Given the sustained redo rate at this level (discussed more in the next section), 1.1ms waits for redo log flushing is very respectable.

	Wait		Event		Wait Time			Summary Avg Wait Time (ms)				
I#	Class	Event	Waits	%Timeouts	Total(s)	Avg(ms)	%DB time	Avg	Min	Max	Std Dev	Cnt
*	User I/O	db file sequential read	571,931,278	0.0	549,803.06	1.0	79.51	0.96	0.96	0.96	0.00	2
		DB CPU	N/A	N/A	101,026.05	N/A	14.61					2
	Cluster	gc cr grant 2-way	148,209,799	0.0	40,608.93	0.3	5.87	0.27	0.26	0.29	0.02	2
	Cluster	gc current grant 2-way	49,371,628	0.0	13,975.73	0.3	2.02	0.28	0.27	0.29	0.01	2
	System I/O	log file parallel write	5,156,057	0.0	5,850.22	1.1	0.85	1.13	1.13	1.14	0.01	2
	Cluster	gc buffer busy acquire	4,286,464	0.0	5,569.14	1.3	0.81	1.26	1.22	1.30	0.05	2
	User I/O	read by other session	4,370,203	0.0	3,257.01	0.7	0.47	0.57	0.39	0.75	0.25	2
	Cluster	gc cr grant congested	3,316,028	0.0	1,541.91	0.5	0.22	0.46	0.31	0.61	0.21	2
	Applicatio	enq: TX - row lock contention	808,738	0.0	1,186.04	1.5	0.17	1.40	1.33	1.47	0.10	2
	Concurrenc	cursor: pin S	279,302	0.0	991.90	3.6	0.14	3.55	3.31	3.79	0.34	2
1	User I/O	db file sequential read	279,720,137	0.0	268,801.80	1.0	77.73					
		DB CPU	N/A	N/A	51,308.83	N/A	14.84					
	Cluster	gc cr grant 2-way	71,923,693	0.0	20,515.78	0.3	5.93					
	Cluster	gc current grant 2-way	23,956,922	0.0	7,037.11	0.3	2.04					
	Cluster	gc buffer busy acquire	4,286,417	0.0	5,569.08	1.3	1.61					
	User I/O	read by other session	4,364,748	0.0	3,254.88	0.7	0.94					
	System I/O	log file parallel write	2,579,555	0.0	2,909.88	1.1	0.84					
	Applicatio	enq: TX - row lock contention	808,725	0.0	1,186.02	1.5	0.34					
	Cluster	gc cr grant congested	1,728,151	0.0	1,055.17	0.6	0.31					
	Concurrenc	cursor: pin S	137,914	0.0	456.52	3.3	0.13					
2	User I/O	db file sequential read	292,211,141	0.0	281,001.26	1.0	81.28					
		DB CPU	N/A	N/A	49,717.21	N/A	14.38					
	Cluster	gc cr grant 2-way	76,286,106	0.0	20,093.14	0.3	5.81					
	Cluster	gc current grant 2-way	25,414,706	0.0	6,938.63	0.3	2.01					

Figure 9: AWR Top Timed Events - XtremIO Dual X-Brick Array

Average wait time for db file sequential read wait events was 961µs. Further, both of the Real Application Clusters nodes experienced the

same I/O latency because the maximum of averages on any host was 960us—as reported under the Summary Avg Wait Time heading.

Transaction Logs (redo logs) are critical part of any OLTP workloads. It is clear from Figure 9 that although there were 1,400 per second log file parallel write wait events for redo log flushing, average service times were only 1.1 milliseconds. Given the sustained redo rate at this level (discussed more in the next section), the service times of 1.1 milliseconds waits for redo log flushing is very respectable.

SUSTAINED IO RATES (THROUGHPUT)

Next, we turn to the I/O rates section the AWR report. Since this was a test with 2 X-Bricks, we expect to see double the rates (throughput) measured in the single X-Brick testing (see Figure 4). Figure 10 shows:

- Physical reads were 158,740 per second
- Physical writes were 30,788 per second
- Redo logging was 41.4 MiB

System Statistics - Per Second DB/Inst: [redacted] Snaps: 220-223

I#	Logical Reads/s	Physical Reads/s	Physical Writes/s	Redo Size (k)/s	Block Changes/s	User Calls/s	Execs/s	Parses/s	Logons/s	Txns/s
1	157,406.46	77,636.6	15,064.3	20,987.0	42,774.4	3.5	28,173.3	1.5	0.17	7,041.2
2	160,347.79	81,103.2	15,724.0	21,449.4	43,726.7	2.5	28,795.1	2.3	0.21	7,195.9
Sum	317,754.25	158,739.8	30,788.3	42,436.5	86,501.1	6.0	56,968.3	3.8	0.39	14,237.1
Avg	158,877.13	79,369.9	15,394.1	21,218.2	43,250.6	3.0	28,484.2	1.9	0.19	7,118.6
Std	2,079.83	2,451.3	466.5	327.0	673.4	0.7	439.7	0.5	0.03	109.4

Figure 10: Oracle Database OLTP - XtremIO Dual X-Brick Array

Figure 11 offers a side-by-side comparison of the single and dual X-Brick XtremIO testing where we see:

- Random reads scaled at 96% of linear scalability
- Writes scaled at 98% of linear scalability
- Redo logging and SQL executions scaled linearly

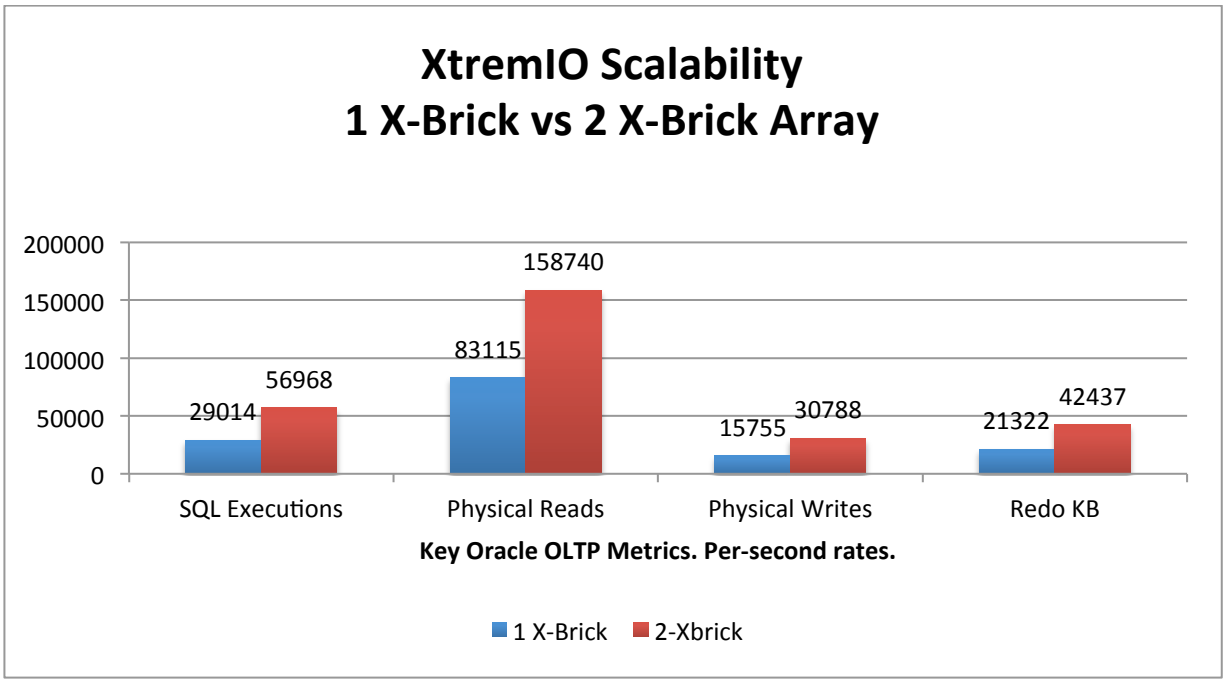

Figure 11: Comparison of Key Oracle Database OLTP Metrics - Linear Scalability of XtremIO Array

PREDICTABLE SERVICE TIMES ON XTREMIO

Service times with XtremIO do not increase unpredictably as I/O queues grow deeper.

To illustrate the predictability of service times as I/O rates increase, EMC engineers scaled up the SLOB testing against a dual X-Brick XtremIO array. The zero-think-time runs were gradually increased to 368 (184 dedicated connections per RAC host) from 192 (from previous tests). Note, each data point in Figure 12 represents a 60-minute test of zero think-time SQL processing with a SQL blend of 75% SELECT with 25% UPDATE.

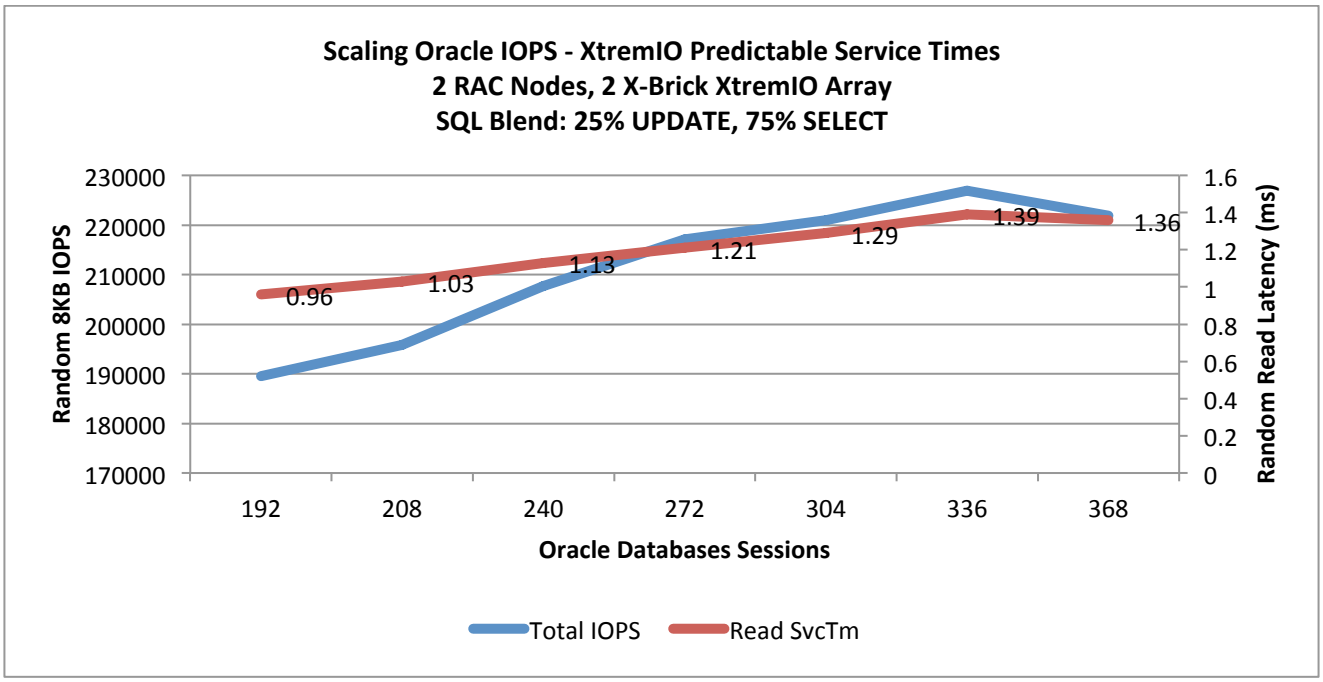

Figure 12: Predictable Service Times Measured with Oracle Database OLTP - XtremIO Dual X-Brick Array

Figure 12 shows that IOPS increased from 189,525 to 221,906—an increase of 17% without any sharp spike in average service times

when number of sessions were increased from 192 to 368. Even at the 368 sessions, service times were still averaging 1.36 milliseconds.

It is clear from Figure 12 that XtremIO keeps the service times within predictable range—unlike logarithmic increases seen with other flash array products—even when the IOPS serviced increases 2X.

Please refer to Appendix A for more information on how the data presented in Figure 12 was collected.

ARCHITECTURES IMPACT SUSTAINABILITY OF SERVICE TIMES

AFA vendors have adopted different architectures—scale up (two controllers and many disk shelves) and scale out (linear scaling of performance and capacity). Different data protection schemes have also been used—most of them have adopted RAID systems that were developed for HDDs and very few have created data protection from the ground up. The architectural choices impact not only data protection and effective capacity but also the sustainability of service times (latency) and throughput. The architectural weaknesses are usually exposed by sustained heavy workloads—exactly the kind of workloads for which flash arrays are typically deployed. For example, the design approach for handling intrinsic NAND Flash garbage collection and write amplification for a particular array might favor workloads that are either entirely write-centric or read-centric but exhibit pathological behavior with a mixed I/O profile. Design choices also dictate how arrays perform when the array capacity is written close to full (space exhaustion).

ORACLE DATABASE TABLESPACES ARE GENERALLY PRE-ALLOCATED

Oracle Database stores tables and indexes in tablespaces that are comprised of one or more datafiles. Datafiles are initialized when added to the database. Database Administrators (DBA) must chose when space is added to tablespaces—pay now or pay later. The DBA can, for instance, chose to add large datafiles to the database before application data is loaded or opt instead to add auto-extending files to the database. The first option takes some processing time while the space is overwritten with large sequential writes. On the other hand, the latter approach taxes active sessions by triggering auto-extend operations when the database needs more space. An auto-extend operation consists of initializing each block and writing them to the extending datafile, which can impact application SLAs. For this reason, DBAs generally size the application and pre-allocate space. Often times, the DBAs will also leave the tablespace in auto-extend mode just to ensure any miscalculation in pre-sizing doesn't result in application failure.

AFAs are not equally capable of handling pre-allocation, especially if it causes space exhaustion. So what happens if a DBA decides to create tablespaces that fill up the user-capacity of an AFA? It depends.

In the next sections, we will investigate how XtremIO and a competing product behave under space exhaustion. EMC engineers did a side-by-side comparison to find overall array behavior over an 8-hour time period and also looked for any trends on a much longer period.

TEST DESCRIPTION

Each array was attached with identical storage networking connectivity to servers configured with two Intel E5-2600 processors. Each server ran Oracle Linux with the Unbreakable Enterprise Kernel. Oracle Database 11g Release 2 was installed and configured to use Automatic Storage Management on both servers as well.

The following sequence of tasks was performed on both configurations:

1. Baseline preparation
    a. A tablespace of 1100MB was created in which to load the OLTP-style test data. Once again, SLOB was used for this testing.
    b. The tablespace was then loaded with 1TB of SLOB test data using the single-schema feature.
    c. Baseline tests were conducted at varying read:write ratios to ensure the health of the 1TB scale SLOB schemas.

2. Create large Oracle Tablespace sized to array's usable capacity
    a. EMC engineers created a large tablespace in order to drive space utilization beyond roughly ninety percent of the formatted user-addressable capacity of each array. The arrays offered vastly different gross capacities, so the size of this tablespace was relative to the user-addressable space supported by the array.
        i. The tablespace created in the competitor's array was 12TB.
        ii. The tablespace created in the single X-Brick XtremIO array was a little over 4TB.

3. Long-Duration OLTP-Style Testing

    a. An 8-hour SLOB test with 25% SQL Update and 96 sessions was started, targeting the 1TB data set loaded in step 1.

8-HOUR WAIT EVENTS ANALYSIS

Figure 13 and Figure 14 compare the Top 5 wait events between the competitor's AFA and XtremIO flash array for the 8-hour run.

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	1.846120E+09	2,568,641	1	92.9	User I/O
DB CPU		196,798		7.1	
read by other session	35,431,056	50,065	1	1.8	User I/O
enq: TX - row lock contention	3,567,648	5,382	2	.2	Applicatio
cursor: pin S	1,018,643	2,214	2	.1	Concurrenc

Figure 13: 8-Hour OLTP Testing Top 5 Timed Events - Competitor's array

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	2.459788E+09	2,565,411	1	92.4	User I/O
DB CPU		269,784		9.7	
cursor: pin S	1,095,231	2,444	2	.1	Concurrenc
db file scattered read	391,172	1,860	5	.1	User I/O
latch: cache buffers chains	321,661	170	1	.0	Concurrenc

Figure 14: 8-Hour OLTP Testing Top 5 Timed Events - XtremIO Single X-Brick Array

Key observations from Figure 13 and Figure 14:

1. Both the competitor's and XtremIO array showed a workload with healthy wait profiles for 25% SQL Update OLTP-style testing since random single block reads ranked at the top.
2. Single X-Brick serviced 2.5 billion 8KB random read IOs at average latency of 1.04ms whereas the competitor's array needed 1.39ms.
3. XtremIO serviced 33% more random reads during the 8-hour test as a result of 33% faster average service times.

8-HOUR THROUGHPUT ANALYSIS

Generally, averages over longer duration tend to "smooth over" the pathologies. Given that the testing was conducted for an 8-hour period, it was decided to scrutinize the results with vmstat(8) data to unearth any trends in the result.

The SLOB test kit produces vmstat data based on 3-second interval samples, which amounted to 9,600 3-second samples. Figure 15 shows the resulting graph which shows throughput (MB/s) sustained by the single X-Brick XtremIO array compared to the competitor's array.

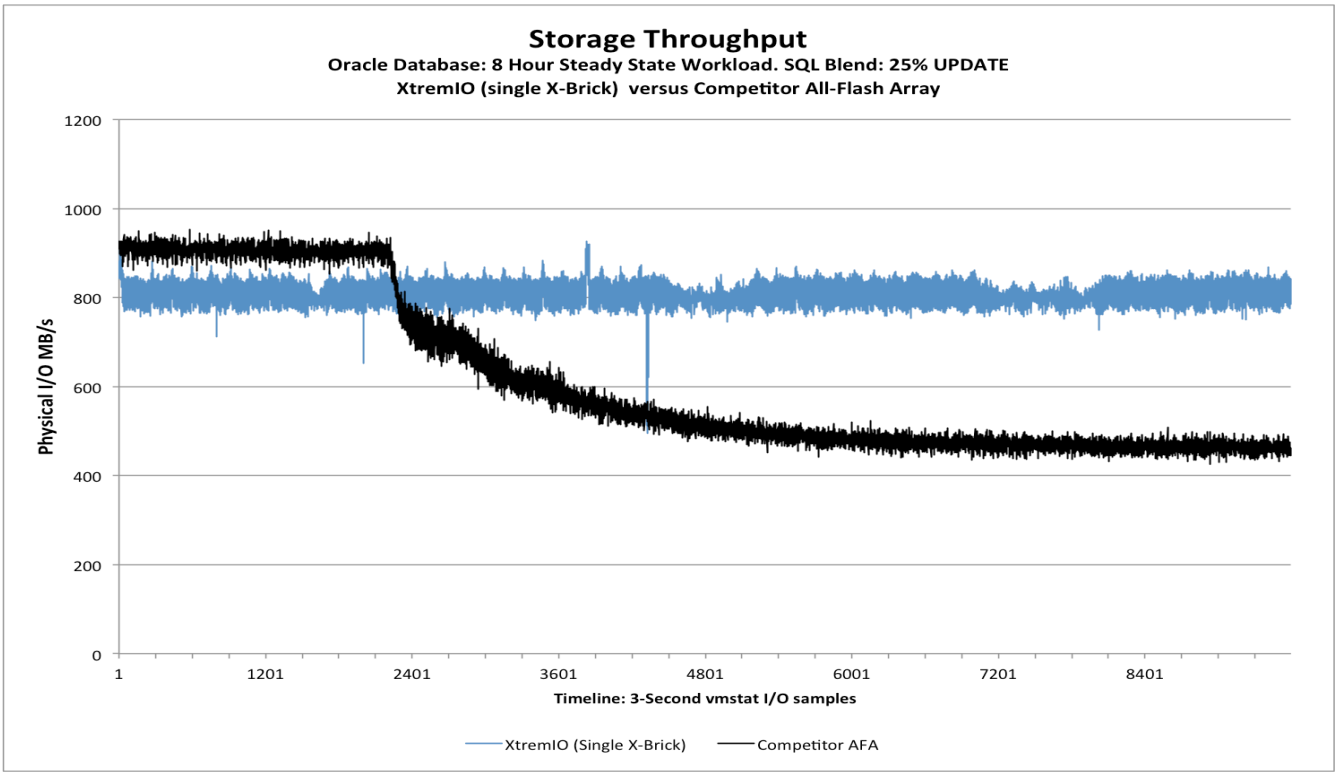

Figure 15: 8-Hour OLTP Workload Comparison - XtremIO Single X-Brick array vs. Competitor's Array

Figure 15 reveals the following trends:

1. The competitor's array was roughly 17% faster than the single X-Brick XtremIO array for about the first 2 hours of the test cycle.
2. The competitor's array entered into a pathological state where the throughput started dropping at roughly two-hours into the test and settled to about 50% drop in few hours. It did not return to the throughput levels demonstrated at the beginning.
3. The XtremIO array sustained the workload with predictable performance from beginning to end.

It was concluded that the competitor's array was experiencing larger latency around 2-hours into the test due to garbage collection of unused blocks and hence was directly impacting the throughput of the Oracle OLTP workload. Latency in physical I/O directly limits the amount of concurrent I/O and, thus, storage throughput as depicted in Figure 15 suffers. In other words, I/O service times are the only performance trait that can affect throughput with this workload.

Notably, throughput slope trend attained a new steady state around 5 hours. The key question was whether the array could revert to the initial level of throughput, if it was given enough time to correct the pathological state. During the eight hours of testing, we had no evidence suggesting it could.

2-DAY TEST CYCLE

To explore how the competitor's array behaved with a much longer test cycle and whether idle time helps with garbage collection, EMC engineers changed the parameters of the test slightly by replacing the existing 12TB with a new 12TB tablespace. The test was conducted for 215 iterations (or 53 hours and 45 minutes to be precise) in the following order:

1. Run 300-second 1TB-scale SLOB test at 96 sessions, 25% SQL Update
2. Sleep 600 seconds

The competitor's array remained idle for 66% of the entire test due to the ten-minute pause between each invocation workload.

At the end of the test cycle, EMC engineers had 215 AWR reports from which to extract I/O performance characterization data. On analysis of this huge data, it was concluded that in spite of altering the test to include 66% idle time, the competitor's array demonstrated the same behavior during the 53 hour 45 minute test as during the 8-hour single test suite (Figure 15).

It appeared that more "rest" was needed for the array to go back to the initial state. It was decided to rest the array for two days before performing any further tests.

TWO DAYS OF REST FOR THE COMPETITOR'S ARRAY

After the 2-day idle period, EMC engineers aimed to see whether the array had "healed itself." To that end, the engineers executed a 5-minute SLOB test with 96 sessions, 25% SQL Update and 1TB active data set. It turns out the 2-day idle period was beneficial since the 5-minute SLOB test showed performance had returned to pre-pathological levels. In fact, performance returned to a level roughly 6% better than the best pre-pathological test results. Figure 16 shows aggregate IOPS graphed on a timeline that includes the first 215 iterations of the work/rest testing followed by 2 days of idle time followed by the final, 5-minute test.

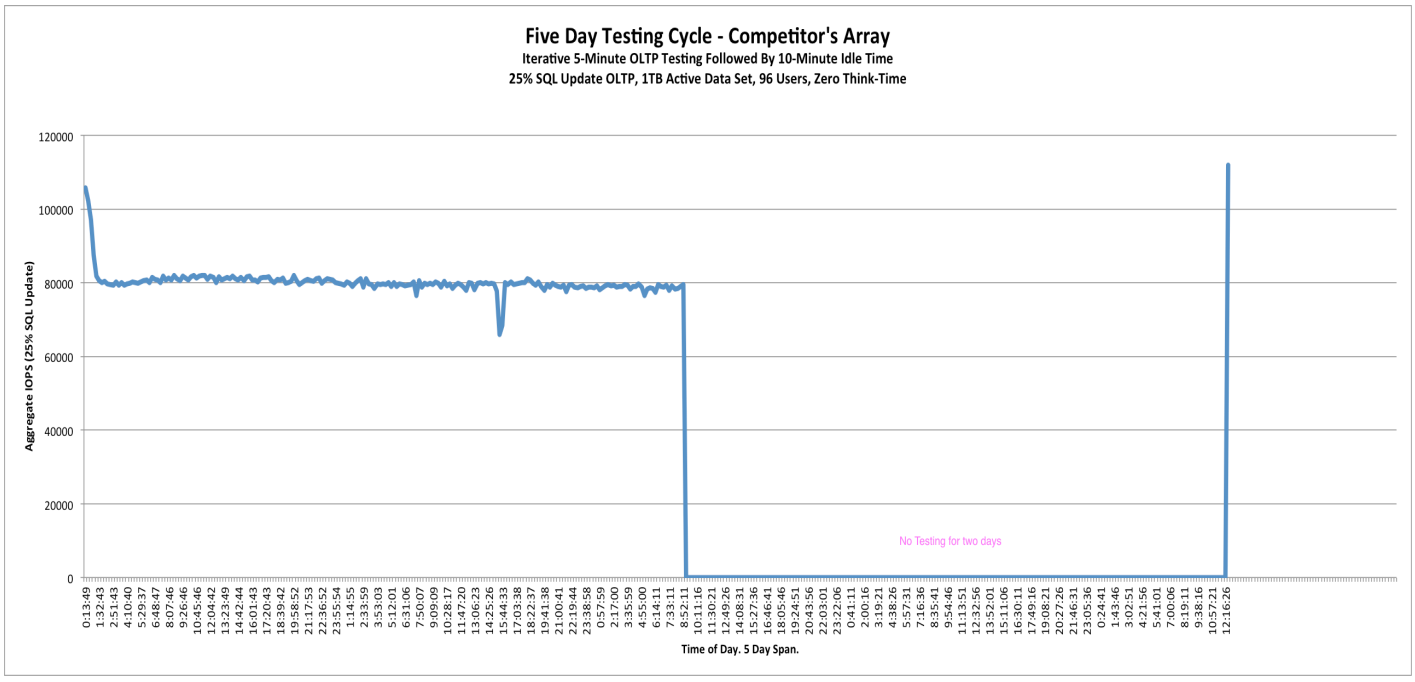

Figure 16: Garbage Collection Pathology Timeline - Competitor's Array

EMC engineers began to question whether the test methodology was simply too aggressive—or too far outside the design center of the competitor's array. After all, most enterprise-grade IT products have both strengths and weaknesses.

EMC engineers had already eliminated test-duration as a possible flaw in the test methodology by extending the testing cycle to several days (Figure 16). Further, it was obvious that going directly from a large tablespace initialization operation into an OLTP-style workload exposed either a severe design or implementation weakness in the competitor's array.

LARGE ACTIVE DATA SET TESTING

EMC engineers hypothesized that perhaps the 1TB active data set in the previous testing may have resulted in an unfavorable hot-spot on the competitor's array. To investigate this hypothesis, EMC engineers decided to change several test parameters and reengage both the competitor's array and the two X-Brick XtremIO array under the following altered test conditions:

- **Large Active Data Set:** The SLOB test database was loaded with an 11TB active data set—11X the active data set of the previous tests. Loading SLOB data is not a high-bandwidth workload, per se. From an AFA perspective, loading SLOB data is actually a fairly low-volume stream of write I/O. In this particular case, it took roughly 16 hours to load (roughly 200 MB/s). This loading time was consistent across both the competitor's array and the XtremIO array.

- **SQL Workload Mix:** The SLOB SQL Update percent variable was set to 100. Oracle Database cannot process an UPDATE SQL statement on a block of data unless it is the SGA buffer pool. Because the active data set was so much larger than the buffer pool, the processing sequence was essentially read/modify/write in nature. To put it another way, the block must be read into the SGA (the read), the SQL processing updates the block (the modify) and the transaction commit causes a write to the online redo log (the write).

- **Larger SGA Buffer Pool:** The Oracle SGA buffer pool size was increased from 1 gigabyte to 32 gigabytes. The idea in such a change is not to improve Oracle SGA hit ratio though. Indeed, considering the size of the active data set of this workload (11TB), this buffer pool capacity only has room for .03% of the active data set. Instead, this type of tuning is intended to affect how the Oracle Database Writer (DBWR) processes behave. In general, larger SGA buffer pool sizes change the batching-nature of modified page flushing. In other words, it's a different I/O profile—a bit more burst-like in nature.

- **Test Duration:** The test duration was 24 hours of zero think-time SQL processing with 96 Oracle Database sessions ("users").

AWR LOAD PROFILE ANALYSIS

Once the testing was complete, EMC engineers analyzed AWR and system-level performance data. As always, the most important data for analysis comes from AWR. First, the EMC engineers compared the Load Profile data from the AWR reports. Figures 17 and 18 show:

1. Both arrays sustained an Oracle Database workload with 6.04 average block changes per SQL execution. This establishes parity between the measured results.
2. Both results show 96 seconds of DB Time per second. This maps directly to the fact that the SLOB test was executed with 96 zero think-time users.
3. The XtremIO array was able to sustain 76,289 physical IOPS compared to the competitor's array, which delivered 39% less 46,457 IOPS.

```
Cache Sizes                       Begin        End
~~~~~~~~~~~~~~~                ---------- ----------
               Buffer Cache:    32,768M    32,768M  Std Block Size:          8K
           Shared Pool Size:     8,192M     8,192M      Log Buffer:    199,676K

Load Profile              Per Second    Per Transaction   Per Exec   Per Call
~~~~~~~~~~~~~~         ---------------    --------------- ---------- ----------
      DB Time(s):               96.0               0.0       0.01     383.47
       DB CPU(s):                3.7               0.0       0.00      14.91
       Redo size:       17,560,184.1         1,819.6
   Logical reads:           87,770.1               9.1
   Block changes:           58,373.6               6.1
  Physical reads:           26,407.2               2.7
 Physical writes:           20,050.0               2.1
      User calls:                0.3               0.0
          Parses:                0.9               0.0
     Hard parses:                0.0               0.0
W/A MB processed:                0.0               0.0
          Logons:                0.1               0.0
        Executes:            9,654.3               1.0
       Rollbacks:                0.0               0.0
    Transactions:            9,650.3
```

Figure 17: 24-Hour Large Active Data (100% SQL Update) Testing - Competitor's Array

```
Cache Sizes                       Begin        End
~~~~~~~~~~~~~~~                ---------- ----------
               Buffer Cache:    32,768M    32,768M  Std Block Size:          8K
           Shared Pool Size:     8,192M     8,192M      Log Buffer:    199,676K

Load Profile              Per Second    Per Transaction   Per Exec   Per Call
~~~~~~~~~~~~~~         ---------------    --------------- ---------- ----------
      DB Time(s):               96.0               0.0       0.01      22.37
       DB CPU(s):                6.9               0.0       0.00       1.61
       Redo size:       30,230,760.6         1,875.9
   Logical reads:          132,317.9               8.2
   Block changes:           97,506.9               6.1
  Physical reads:           42,917.5               2.7
 Physical writes:           33,372.5               2.1
      User calls:                4.3               0.0
          Parses:                1.8               0.0
     Hard parses:                0.0               0.0
W/A MB processed:                0.1               0.0
          Logons:                0.1               0.0
        Executes:           16,137.9               1.0
       Rollbacks:                0.0               0.0
    Transactions:           16,115.4
```

Figure 18: 24-Hour Large Active Data Testing - XtremIO Dual X-Brick Array

AWR WAIT EVENT ANALYSIS

With the AWR Load Profile results in mind, EMC engineers turned their focus to the Wait Event data. After all, with a zero think-time workload one expects to be CPU-bound. Since that was not the case, one must account for waits. Figure 19 and 20 show:

1. **General Workload Health**: Both the XtremIO and competitor arrays exhibited a healthy wait profile. As explained earlier in this document the workload consists of a read/modify/write cycle and thus the top wait event of db file sequential read is to be expected.
2. **Key Performance Indicator**:
   a. The competitor's array serviced the critical db file sequential read I/O with 3.49ms average service times (7961122/ 2.2809*10^9).
   b. The XtremIO array serviced db file sequential read I/O with 1.07ms service times (3981226/ 3.7089*10^9).

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	2.280979E+09	7,961,122	3	96.0	User I/O
DB CPU		322,635		3.9	
read by other session	15,147,394	53,350	4	.6	User I/O
enq: TX - row lock contention	8,590,078	31,063	4	.4	Applicatio
cursor: pin S	726,126	2,398	3	.0	Concurrenc

Figure 19: Time Model Events for 24-Hour Large Active Data Testing - Competitor's array

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	3.708963E+09	3,981,226	1	48.0	User I/O
free buffer waits	257,599,808	3,292,653	13	39.7	Configurat
DB CPU		598,201		7.2	
enq: HW - contention	609,306	327,519	538	3.9	Configurat
log file switch (checkpoint in	91,740	142,198	1550	1.7	Configurat

Figure 20: Time Model Events for 24-Hour Large Active Data Testing - XtremIO Dual X-Brick Array

After accounting for waits and service times, EMC engineers turned their focus on the SQL Statistics portion of the AWR reports—specifically the SQL ordered by User I/O Time section. This section of AWR proves a) the same SQL code was executed and b) each execution of the SQL statement accesses the same number of rows in the table. Figures 21 and 22 show:

1. **Test Parity**:
   a. The identical SQL statement ranked number one in I/O-generating SQL in testing of both arrays as per the universally unique SQL Id value.
   b. The Per Execution section proves each SQL execution modified 2 rows in the table.
2. **Performance**: The XtremIO array sustained 67% more executions of the top I/O-generating SQL than the competitor's array (1,390,465,627 versus 831,656,705).

SQL ordered by User I/O Time (Global)

- Captured SQL account for 99.7% of Total User I/O Wait Time (s): 8,015,147
- Captured PL/SQL account for 99.2% of Total User I/O Wait Time (s): 8,015,147

	Total								Per Execution						
SQL Id	IOWait (s)	Elapsed (s)	CPU (s)	Gets	Reads	Rows	Cluster (s)	Execs	IOWait (s)	Elapsed (s)	CPU (s)	Gets	Reads	Rows	Cluster (s)
fmrxq2k0nbp4b	7,992,984.45	8,195,642.24	236,071.39	6,464,229,445	2,274,241,246	1,662,997,730	0.00	831,656,705	0.01	0.01	0.00	7.77	2.73	2.00	0.00

Figure 21: Top SQL Ordered by User I/O Wait Time for Large Active Data Testing – Competitor's array

SQL ordered by User I/O Time (Global)

- Captured SQL account for 99.8% of Total User I/O Wait Time (s): 4,001,584
- Captured PL/SQL account for 99.0% of Total User I/O Wait Time (s): 4,001,584

	Total								Per Execution						
SQL Id	IOWait (s)	Elapsed (s)	CPU (s)	Gets	Reads	Rows	Cluster (s)	Execs	IOWait (s)	Elapsed (s)	CPU (s)	Gets	Reads	Rows	Cluster (s)
fmrxq2k0nbp4b	3,992,706.65	8,125,593.96	455,324.78	8,760,367,137	3,702,262,981	2,780,748,054	0.00	1,390,465,627	0.00	0.01	0.00	6.30	2.66	2.00	0.00

Figure 22: Top SQL Ordered by User I/O Wait Times for Large Active Data Testing - XtremIO Dual X-Brick array

I/O HISTOGRAM ANALYSIS—AVERAGES HIDE PATHOLOGY

As pointed out throughout this document, averages are not an acceptable metric for I/O performance in the era of low-latency commercial computing. It is important to ensure averages are not obscuring latency outliers.

AWR reports provide histogram data, however, EMC engineers chose to query the Oracle Database internal performance view v$file_histogram directly in order to further characterize the performance differences between the competitor's array and the XtremIO array. Figure 23 clearly illustrates how the competitor's array averaged roughly 3.5 milliseconds over the billions of I/O serviced during the 24-hour test. Although the average was 3.5 milliseconds, the histogram clearly shows that nearly 1 in 10 reads serviced by the competitor's array fell between 8 and 16 millisecond latency. This may not be the best way to think about this aspect of performance. It's important to normalize these I/O stalls to wall-clock time.

Figure 23 shows that 26,998,287 read operations suffered latencies between 16 and 32 milliseconds with the competitor's array. The test duration was 86,400 seconds so, on average, the competitor's array penalized some 313 SQL executions per second with I/O service times between 16 and 32 milliseconds. Bear in mind there were only 96 users connected to the database.

```
SQL> col tot_single_blk_reads format 999,999,999,999
SQL> select sum(SINGLEBLKRDS) tot_single_blk_reads from v$file_histogram;

TOT_SINGLE_BLK_READS
--------------------
       2,291,834,272

SQL> @file_histogram.sql

Read Time
(ms)                     Reads
------------- ---------------- ----------------------------------------------------
<1                 201,956,935  *********
1<2                486,707,158  **********************
2<4              1,052,712,503  ************************************************
4<8                408,278,563  ******************
8<16               112,545,345  ****
16<32               26,998,287
32<64                2,514,932
64<128                 120,537
128<256                     12

9 rows selected.

SQL>
```

Figure 23: Random Read Service Time Histogram for Large Active Data Testing - Competitor's Array

Finally, Figure 24 shows that the XtremIO array serviced over 93% of all 3.7 billion read operations in less than 2 milliseconds with the vast majority less than 1 millisecond. The distribution of outliers was negligible compared to the competitor's array. Consider the 16 to 32 millisecond service time bucket in the XtremIO case. Figure 24 shows XtremIO dipped to that level 95% less frequently than the competitor's array. It's fair to say the rest of the data represented in Figures 23 and 24 round out the picture quite well.

```
SQL> col tot_single_blk_reads format 999,999,999,999
SQL> select sum(SINGLEBLKRDS) tot_single_blk_reads from v$file_histogram;

TOT_SINGLE_BLK_READS
--------------------
       3,715,077,288

SQL> @file_histogram.sql

Read Time
(ms)                     Reads
------------- ---------------- ----------------------------------------------------
<1               2,532,793,306  **************************************************
1<2                936,941,653  ******************
2<4                223,308,601  ***
4<8                 14,528,299
8<16                 5,954,255
16<32                1,551,008
32<64                      111
64<128                      55

8 rows selected.

SQL>
```

Figure 24: Random Read Service Time Histogram for Large Active Data Testing - XtremIO Dual X-Brick Array

SUMMARY

The case studies and analysis offered in this paper conclusively demonstrate that XtremIO's clean sheet design uniquely exploits the characteristics of flash. The time model analysis and performance scaling exercise clearly demonstrates XtremIO's scalable and sustained performance traits for predictable latency and IO throughput.

XtremIO delivered more IOPS, lower latency, and a vastly superior latency histogram through a wide range of conditions from short to long duration and small to large data sets as well as varying modify-intensity of the SQL workloads. XtremIO's scale-out architecture also supported linear scaling of performance with the test data showing that XtremIO scales with 96% linearity from a single to a dual X-Brick array. Mixed 8 KB IOPS increased from 99,000 with one X-Brick to 190,000—with 1-millisecond service times (or less)—with two X-Bricks. XtremIO performed better than the competitor's array in longer duration and large scale SQL UPDATE testing that needs more sustained performance when the array gets filled. On the other hand, a competitor's array exhibited pathological unpredictable results due to non-optimized handling of intrinsic NAND attributes such as garbage collection.

It can be concluded that the EMC XtremIO flash arrays are well suited for meeting most demanding performance requirements of Oracle Database OLTP workloads.

APPENDIX A

Using Oracle Database to test storage for long periods of time produces volumes of rich Time Model data. For example, in the above section about scaling users to increase physical I/O demand, EMC engineers conducted 12 hour-long SLOB tests with varying user counts. Each SLOB test produces AWR reports. Figure 25 shows the AWR reports created for each user count identified by the numeric suffix of each file name.

As Figure 25 shows, the rac_stats.sh script was used to extract the germane data regarding physical I/O for the test period. For example, Figure 25 shows the AWR report collected at 192 users sustained an average aggregate IOPS rate of 189,525 consisting of 158,737 read IOPS and 30,788 write IOPS. The service times (reads) for the 192-user test averaged 0.96 milliseconds and the redo payload was 41 megabytes per second. At the top end of the test cycle we see how the 368 sessions issued an average of 189,020 random 8KB read IOPS and 36,886 writes per second—for an entire hour. Further, at 368 users the service times for reads was 1.36ms and the redo payload was 47MB/s.

```
$ ls -l awr_rac.txt*
-rw-r--r--  1 clossk  staff  194737 Mar 11 16:55 awr_rac.txt.192
-rw-r--r--  1 clossk  staff  193756 Mar 11 16:31 awr_rac.txt.208
-rw-r--r--  1 clossk  staff  192156 Mar 11 16:31 awr_rac.txt.224
-rw-r--r--  1 clossk  staff  192601 Mar 11 16:31 awr_rac.txt.240
-rw-r--r--  1 clossk  staff  192206 Mar 11 16:31 awr_rac.txt.256
-rw-r--r--  1 clossk  staff  192730 Mar 11 16:31 awr_rac.txt.272
-rw-r--r--  1 clossk  staff  192905 Mar 11 16:31 awr_rac.txt.288
-rw-r--r--  1 clossk  staff  189797 Mar 11 16:31 awr_rac.txt.304
-rw-r--r--  1 clossk  staff  192861 Mar 11 16:31 awr_rac.txt.320
-rw-r--r--  1 clossk  staff  192695 Mar 11 16:31 awr_rac.txt.336
-rw-r--r--  1 clossk  staff  189638 Mar 11 16:31 awr_rac.txt.352
-rw-r--r--  1 clossk  staff  192446 Mar 11 16:31 awr_rac.txt.368
$ sh ./rac_stats.sh awr_rac.txt.*
FILE|Total IO|Reads|Read Latency|Writes|Redo MB/s|
awr_rac.txt.192|189525|158737|   0.96|30788|41|
awr_rac.txt.208|195826|163565|   1.03|32261|42|
awr_rac.txt.224|202285|168941|   1.08|33344|43|
awr_rac.txt.240|207639|173389|   1.13|34250|45|
awr_rac.txt.256|212438|177351|   1.17|35087|46|
awr_rac.txt.272|217136|181237|   1.21|35899|47|
awr_rac.txt.288|211609|176560|   1.20|35049|46|
awr_rac.txt.304|220937|184298|   1.29|36639|47|
awr_rac.txt.320|226116|188593|   1.36|37523|49|
awr_rac.txt.336|226923|189227|   1.39|37696|49|
awr_rac.txt.352|227001|189268|   1.41|37733|49|
awr_rac.txt.368|221906|185020|   1.36|36886|47|
$
```

Figure 25: Extraction of details from large number of AWR reports using scripts

Figure 26 shows the source code listing for the simple script used to extract data from the AWR reports.

```
#!/bin/bash

function dprint(){
field=$1 ; awk -v f=$field '{ printf("%d\n",$f) }'
}

function top_line_no_commas() {
head -1 | sed 's/,//g'
}

function get_reads(){
f=$1 ; grep '^physical read [^a-z]' $f | top_line_no_commas | dprint 6
}

function get_writes(){
f=$1 ; grep '^physical writes [^a-z]' $f | top_line_no_commas | dprint 4
}

function get_redo(){
f=$1
grep '^redo size [^a-z]' $f | top_line_no_commas| awk '{ printf("%d\n", $4 / 1024 / 1024  ) }'
}

function get_read_latency() {
f=$1
grep 'db file sequential read' $f | sed 's/^.*User/User/g' | top_line_no_commas \
| awk '{ printf("%7.2lf\n",  ($9 / $7) * 1000 ) }'
}

# Program Body

exec 4>/dev/null

echo "FILE|Total IO|Reads|Read Latency|Writes|Redo MB/s|"

for file in $*
do
        reads=`get_reads $file 2>&4`
        writes=`get_writes $file 2>&4`
        redo=`get_redo $file 2>&4 `
        read_latency=`get_read_latency $file 2>&4 `

        echo "$file|$(( reads + writes ))|$reads|$read_latency|$writes|$redo|"
done
```

Figure 26: Script to extract detailed stats from AWR reports

CONTACT US

To learn more about how EMC products, services, and solutions can help solve your business and IT challenges, **contact** your local representative or authorized reseller—or visit us at www.EMC.com.

EMC²