Validating Your IO Subsystem: Coming out of the Black Box

Michael R. Ault (with sections by Jamon Bowen, TMS)
Texas Memory Systems, Inc.
Houston, Texas, USA

Keywords:
IO Subsystem, validation, IOMeter, Orion, Oracle, dbms_resource_manager.calibrate_io, IOPS, throughput

Introduction

For most people the IO subsystem for their servers is a black box. You pick up the phone and call the storage people and tell them how much room you need and it mysteriously appears at the server interface ready for use. Unfortunately while the capacity (as defined in megabytes) is easy to confirm, the performance (defined by input and output operations per second (IOPS) and latency (milliseconds, ms) may be more difficult to determine.

The IO subsystem starts at the device, which actually is a software construct, inside the server. You can have a block device or a character device. A block device has its IO characteristics mapped by the application which accesses it while a character mode device is usually handled by the operating system. You may hear a block device called a RAW device. The device maps to a port on the computer bus and the port maps to a host bus adapter (HBA). The HBA will determine the type of protocol used by the server to address the downstream components. Usually a HBA will be Ethernet, Fibre Channel, iSCSI, Infiniband or some other protocol.

If your system is direct attached to the disk subsystem, then the next component will be the IO interface of the disk or storage device. If your system uses a storage area network (SAN) then it will connect to a switch or a hub. The difference between a switch and a hub is that a switch usually allows each connection the full bandwidth available from the protocol, while a hub will portion out the bandwidth based on the total number of connections.

The bandwidth of the IO subsystem is determined by the number ant type of connections it has between either the server and the IO subsystem or the switch and the server and IO subsystem. Bandwidth is usually stated as an amount of data transferred per second, for example 300 MB/s or 4gb/s. Be careful when looking at specifications as they can be stated in bits or bytes. For example a 4 gigabit per second bandwidth will give you 400 megabytes per second of transfer capability (approximately). Thus the number of approximate IOPS is determined by dividing the total bandwidth by the average transfer size. So, 400 mb/sec with a 128kb IO size would yield a maximum of 3200 IOPS per interface.

Inside of the IO subsystem you may have one or more storage processors and chains of disks or other devices that are actually using some form of SCSI (small computer systems interface) protocol, although some are now using SATA (serial ATA) or newer protocols. This can limit the total IOPS for a tray of disks inside a storage array to a maximum number of IOPS depending on the protocol used.

IO is not a Black Box

Too often as Oracle DBAs and professionals we are led to believe that the entire IO subsystem is out of our hands, that it is a black box technology that we don't need to know anything about other than that we have enough storage capacity for our database. Nothing could be further from the truth. The details of the IO subsystem will make or break your database performance. One of my colleagues has been known to say that eventually all performance issues are traced to IO

issues. I am not sure if that is 100% true but a good many of the performance issues I have come across in 20 years did either directly or indirectly relate to latency issues.

Oracle provides a great many IO related statistics that can either be used directly, or, used to calculate values that are useful for looking at the IO subsystem. At the operating system level, whether it be Perfmon on Windows or any of the many commands on Unix and Linux that can be utilized to obtain IO subsystem performance values, the data you need to verify and validate your IO subsystem are available.

Inside of Oracle views such as v$filestat and v$tempstat provide cumulative information about the number of IO operations since startup and the total time used by the IO operations. By doing some simple queries you can determine long term averages for IOPS and latency for the files in your system. To get more granular data tools such as Statspack and AWR must be used.

```
rem NAME: fileio.sql
rem
rem FUNCTION: Reports on the file io status of all of the
rem FUNCTION: datafiles in the database.
rem HISTORY:
rem WHO    WHAT    WHEN
rem Mike Ault    Created                  1/5/96
rem
column sum_io1 new_value st1 noprint
column sum_io2 new_value st2 noprint
column sum_io new_value divide_by noprint
column Percent format 999.999 heading 'Percent|Of IO'
column brratio format 999.99 heading 'Block|Read|Ratio'
column bwratio format 999.99 heading 'Block|Write|Ratio'
column phyrds heading 'Physical | Reads'
column phywrts heading 'Physical | Writes'
column phyblkrd heading 'Physical|Block|Reads'
column phyblkwrt heading 'Physical|Block|Writes'
column name format a45 heading 'File|Name'
column file# format 9999 heading 'File'
set feedback off verify off lines 132 pages 60 sqlbl on trims
on
rem
select
 nvl(sum(a.phyrds+a.phywrts),0) sum_io1
from
 sys.v_$filestat a;
select nvl(sum(b.phyrds+b.phywrts),0) sum_io2
from
        sys.v_$tempstat b;
select &st1+&st2 sum_io from dual;
rem
ttitle 'File IO Statistics Report'
spool fileio
select
 a.file#,b.name, a.phyrds, a.phywrts,
 (100*(a.phyrds+a.phywrts)/&divide_by) Percent,
```

```
 a.phyblkrd, a.phyblkwrt, (a.phyblkrd/greatest(a.phyrds,1))
brratio,
      (a.phyblkwrt/greatest(a.phywrts,1)) bwratio
from
 sys.v_$filestat a, sys.v_$dbfile b
where
 a.file#=b.file#
union
select
 c.file#,d.name, c.phyrds, c.phywrts,
 (100*(c.phyrds+c.phywrts)/&divide_by) Percent,
 c.phyblkrd, c.phyblkwrt,(c.phyblkrd/greatest(c.phyrds,1))
brratio,
      (c.phyblkwrt/greatest(c.phywrts,1)) bwratio
from
 sys.v_$tempstat c, sys.v_$tempfile d
where
 c.file#=d.file#
order by
 1
/
spool off
pause Press enter to continue
set feedback on verify on lines 80 pages 22
clear columns
ttitle off
```

Listing 1: Example SQLPLUS IO Report

Listing 1 provides a breakdown of IO by total IOs and what file is doing the most, in Listing 2 we see a report to breakdown IO timing.

```
col name format a58 heading 'Name'
col phywrts heading 'Phys. Writes'
col phyreads heading 'Phys. Reads'
col read_rat heading 'Avg. Read|Time'
col write_rat heading 'Avg. Write|Time'
set lines 132 pages 45
ttitle 'IO Timing Analysis'
spool rep_out\&db\io_time
select  f.FILE#
,d.name,PHYRDS,PHYWRTS,READTIM/greatest(PHYRDS,1)*10
read_rat,WRITETIM/greatest(PHYWRTS,1)*10 write_rat
from v$filestat f, v$datafile d
where f.file#=d.file#
union
select  f.FILE#
,d.name,PHYRDS,PHYWRTS,READTIM/greatest(PHYRDS,1)*10
read_rat,WRITETIM/greatest(PHYWRTS,1)*10 write_rat
from v$tempstat f, v$tempfile d
where f.file#=d.file#
order by 5 desc
```

```
/
spool off
ttitle off
clear columns
set lines 80 pages 22
```

Listing 2: IO Timings Report

What Every IO Subsystem Needs

Of course the IO subsystem is not a standalone system, but instead depends on many components to make sure that the IO gets from the storage to the place where it is needed for processing. Any well designed IO subsystem will have redundancy designed, or built into it.

A reliable IO subsystem will have multiple paths to get to the subsystem from the systems that it serves. Each system server should have multiple paths (either on the same HBA or through multiple HBAs) that connect it to the IO subsystem. In order to share the IO subsystem a switch or fabric needs to be utilized to allow multiple servers to access the same IO subsystem. An example is shown in Figure 1.

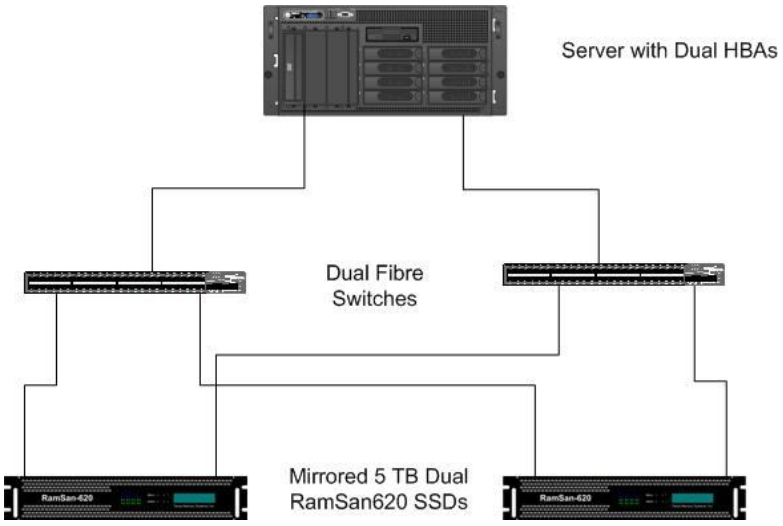

Figure 1: Example of Redundant Storage Network

At the IO subsystem level redundancy is needed at the storage level. The various types of RAID (RAID10, RAID5, RAID6, RAID-S, etc) provide for redundancy at the disk or storage unit level. Within the RAID stripe widths and depths should be aligned to the majority IO need. Most modern SAN systems have redundant controllers and power supplies as well as mundane items such as fans.

Of course from a purely logical viewpoint the IO subsystem needs to provide three things to the servers:

1. Adequate storage capacity (megabytes, gigabytes, terabytes, etc)
2. Adequate bandwidth (megabytes, gigabytes or terabytes per second)
3. Proper response time (low enough latency for performance) (millisecond or sub-millisecond response)

So in order to be a proper IO subsystem, the subsystem must provide reliability, redundancy and satisfy capacity, bandwidth and latency requirements. These are the items that must be validated for an IO subsystem, preferably before it is purchased or implemented.

Where are Storage Systems Going?
Disk based systems have gone from 5000 RPM and 30 or less megabytes to 15K RPM and terabytes in size in the last 20 years. The first Winchester technology drive I came in contact with had a 90 megabyte capacity (9 times the capacity that the 12 inch platters I was used to had) and was rack mounted, since it weighed over 100 pounds! Now we have 3 terabyte drives in a 31/2 inch form factor. However as information density increased, the bandwidth of information transfer didn't keep up at the hard drive level. Most modern disk drives can only accomplish 2 to 3 times the transfer rate of their early predecessors, why is this?

The speed at which a disk drive can access information is limited by 3 things:

1. The number of independent heads
2. The speed of the disk actuator mechanism
3. The speed of the disks rotation

While most disks have multiple platters and multiple read/write heads, the read/write heads are mounted to a single actuator mechanism. By mounting the heads on a single actuator mechanism you may increase the amount of data capable of being read/written at the same time, but you do not increase the maximum random IOPS. Because of these physical limitations most hard drives can only deliver 2-5 millisecond latency and 200-300 random IOPS.

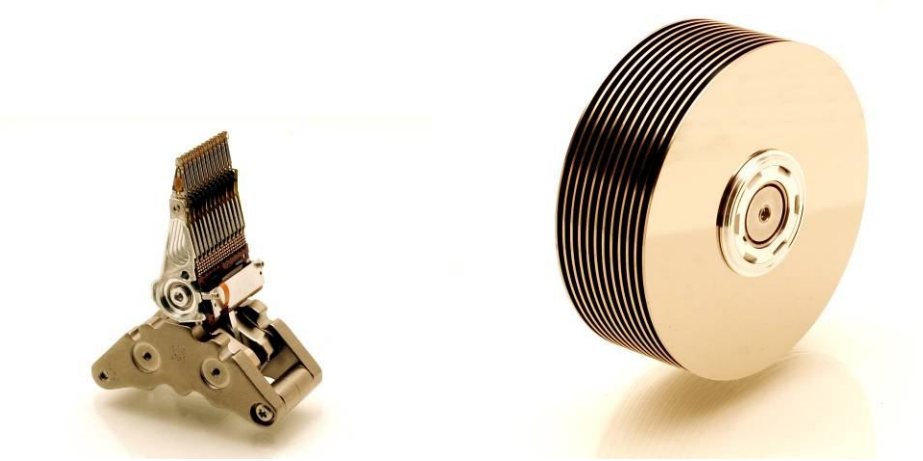

Figure 2: HDD Armature and Disks

Modern SAN and NAS system are capable of delivering hundreds of thousands of IOPS, however, you must provide enough disk spindles in the array to allow this. To get 100,000 IOPS assuming 300 IOPS per disk you would need 334 disk drives at a minimum, more if you want to serve that 100,000 IOPS to multiple servers and users. Of course, the IOPS latency would still be from 2-5 milliseconds or greater. The only way to reduce latency to nearer to the 2 millisecond level is to do what is known as short-stroking.

Short stroking means only utilizing the outer, faster, edges of the disk, in fact usually less than 30% of the total disk capacity. That 333 disks just became 1000 or more to give 100,000 IOPS at 2 millisecond latency.

Disks have fallen dramatically in cost per gigabyte. However their cost per IOPS has remained the same or risen. The major cost factors in a disk construction are the disk motor/actuator and technology to create the high density disks. This means that as disk technology ages, without major enhancements to the technology, their price will eventually stall at around 10 times the cost of the raw materials to manufacture them.

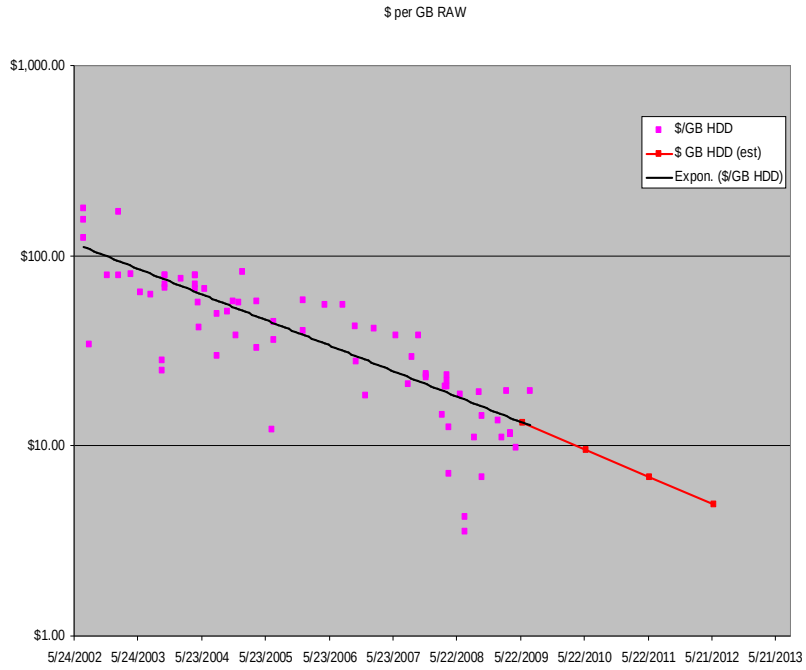

Figure 3: Disk Cost Per GB RAW

So where does all this leave us? SSD technology is the new kid on the block (well, actually they have been around since the first memory chips) now that costs have fallen to the point where SSD storage using Flash technology is nearly on par with enterprise disk costs with SSD cost per gigabyte falling below $40/gb.

SSD technology using Flash memory utilizing SLC based chips provides reliable, permanent, and relatively inexpensive storage alternatives to traditional hard drives. Since each SSD doesn't require its own motor, actuator and spinning disks, their prices will continue to fall as manufacturing technology and supply-and-demand allows. This leads to a closing of the gap between HDD and SDD usage modes as shown in Figure 4.

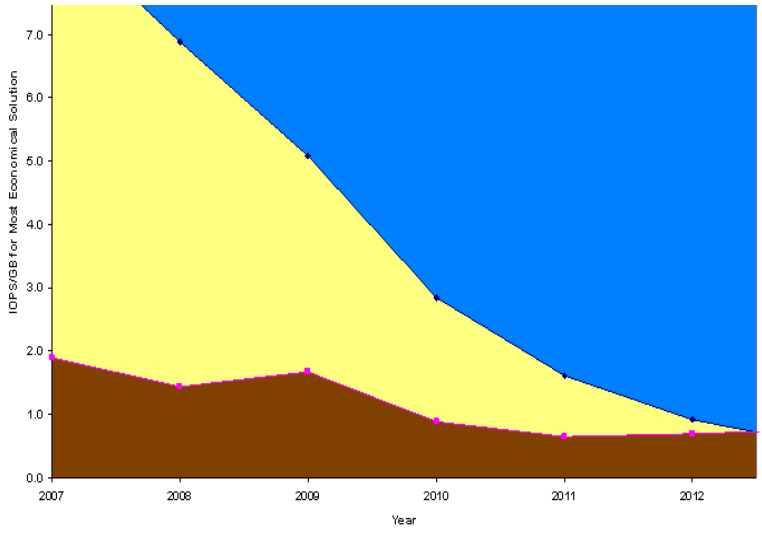

Figure 4: Usage Mode Changes for HDD and SSD

In addition to price to purchase, operational costs (electric, cooling) for SSD technology is lower than the costs for hard disk based technology. Combine that with the smaller footprint per usable capacity and you have a combination that sounds a death knoll for disk in the high performance end of the storage spectrum. By the year 2012 or sooner, SSDs will be less expensive then enterprise level disks at dollars (or Euros) per gigabyte. When a single 3-U chassis full of SSDs can replace over 1000 disks and costs are nearing parity, it doesn't take a genius to see that SSDs will be taking over storage.

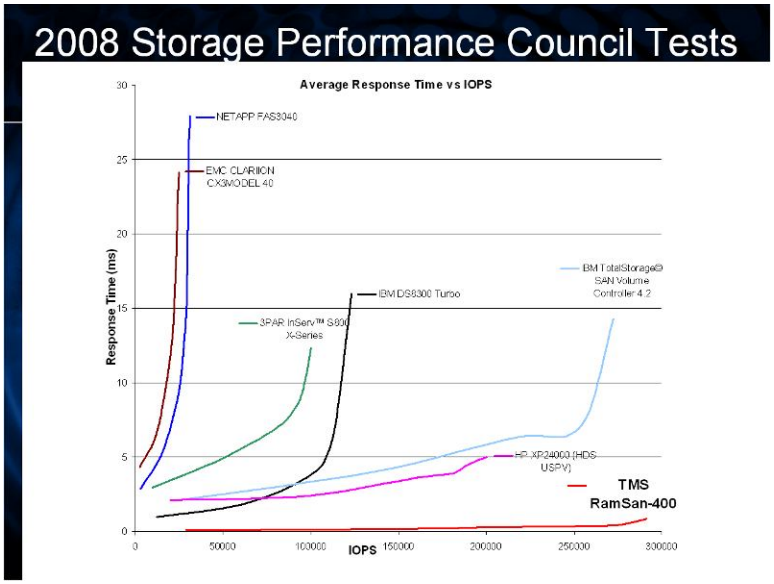

Figure 5: Comparison of IOPS verses Latency

An SSDs full capacity can be used for storage, there is no need to "short-stroke" them for performance. This means that rather than buying 100 terabytes to get 33 terabytes you buy 33 terabytes. SSDs also deliver this storage capacity with IOPS numbers of 200,000 or greater and latencies of less then 0.5 milliseconds. With the various SSD options available the needed IO characteristics for any system can be met as is shown in Figure 6.

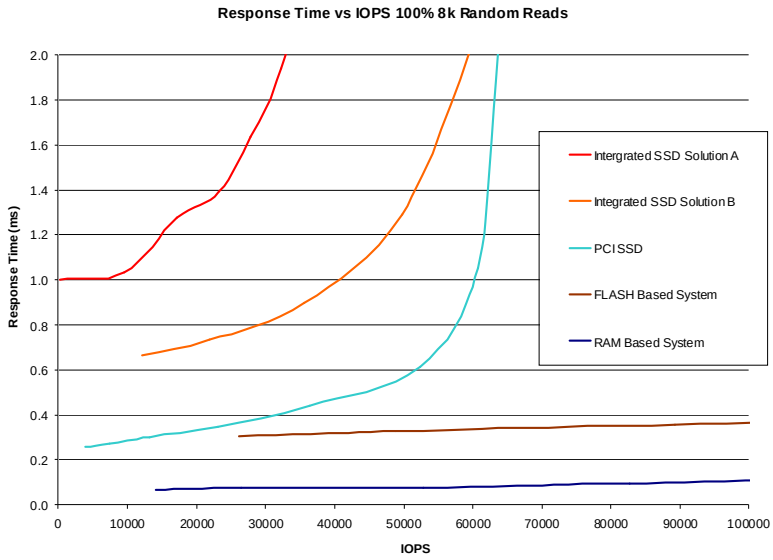

Figure 6: Response and IOPS for Various SSD Solutions

These facts are leading to tiering of storage solutions with high performance SSDs at the peak, or tier zero, and disks at the bottom as archival devices or for storage of non-critical data.

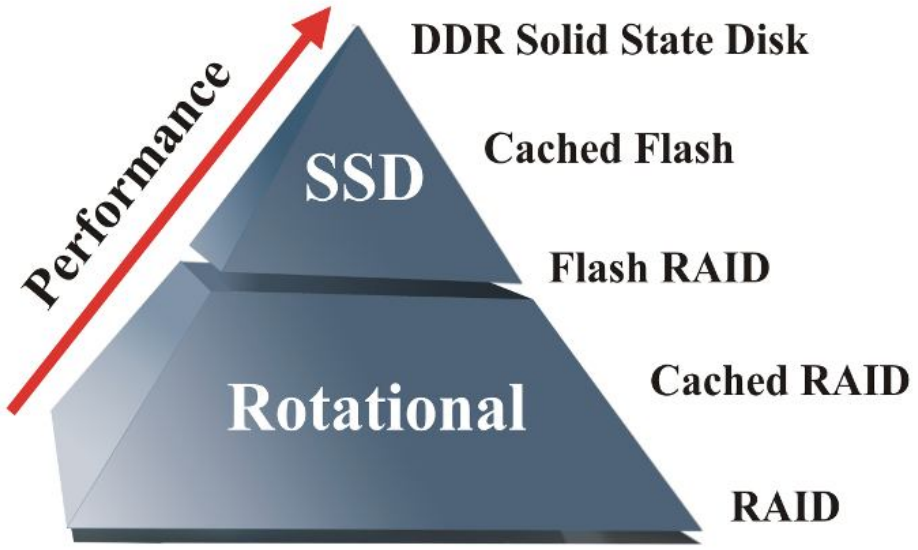

Figure 7: Storage Pyramid

Types of IO Subsystems
There are essentially three types of IO subsystems, these are:

1. SAN – Storage area network
2. NAS – Network attached storage
3. DAS – Direct attached storage

Each of the types of IO subsystems have their benefits and drawbacks, the most used is probably the SAN however with advances in bandwidth and reduction in latency, the NAS is now moving up form being relegated as backup storage only to mainstream, online storage. DAS is limited to a single server, generally speaking, unless it is then mounted as a network block device and shared similar to a NAS.

A SAN is usually fibre attached to the host via a switch or multiple switches for redundancy. Infiniband is now becoming more popular as its increased bandwidth is becoming needed to see the best performance. A SAN is usually shared between multiple servers. A NAS is usually connected via a dedicated Ethernet line through Ethernet switches, although iSCSI is showing up more and more in the industry. Traditionally NAS suffered from latency limitations due to the bandwidth limitations of Ethernet but now that 10, 100 and 1000 gigabit Ethernet is available these limitations are being removed. DAS is moving from SCSI to SATA for many types of DAS systems. However, DAS is usually reserved for disks or form factor SSDs. A new option for SSDs is the PCIe interface which also provides the greatest bandwidth and lowest latency.

The basic interfaces in the various storage technologies are:

- SAN
    - SCSI Fibre channel/Infiniband
    - SATA Fibre Channel/Infiniband

    - SSD Fibre Channel/Infiniband
- NAS
    - Ethernet
    - iSCSI
    - SATA
- DAS
    - ATA
    - SATA
    - SCSI
    - PCIe

This can lead to a confusing array of options when choosing your IO subsystem. Just remember to always plan for peak load and make sure the delivered IOPS and latency for your IO subsystem will allow you to meet any expected performance or service level agreements. This is why validating your IO subsystem, before you put your application onto it is critical!

IO System Issues
The main issues with the IO subsystem fall into the three things we have discussed:

- Capacity
- Bandwidth
- Latency

Let's look at each of these issues.

Capacity
Capacity is probably the easiest issue to deal with. Depending on performance you decide how much of each disk can be used, subtract the rest and then divide the needed capacity by the amount available on each disk and do the proper multiplier for concurrency and RAID level to determine the number of physical platters required. The number of disks needed for concurrency of access depends on queuing theory. Of course, should you need to short-stroke disks to get performance, you must take the short-stroke factor in account as well.

Bandwidth
In one of my other presentations I show how the most bandwidth can be achieved by loading a van with backup tapes and driving it to a destination. Bandwidth is fairly easy to determine if you have an idea of the maximum IOPS needed for your application and the size of each IO, of course I would add a fudge factor to the final value, like say at least 25-50 percent to allow for peaking loads. Once you know how many gigabyte or megabytes per second are needed then the decision comes down to the number and type of host bus adapters (HBA) needed to meet the bandwidth requirement.

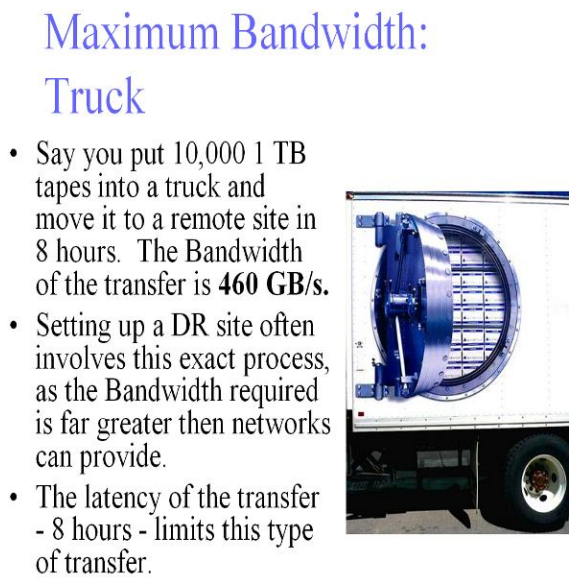

Figure 8: Example of maximum Bandwidth

Generally speaking the type of HBAs are going to be driven by what the IO subsystem supports, if it is a SAN then your choices will be between the various fibre channel HBAs (1gb, 4gb, 8gb, etc) and the use of Infiniband (up to 96 gbit/sec). On a NAS your choices are for network interface cards (NICs) usually you will need a NIC with a TCP offload engine (TOE) to get the maximum speeds. The other NAS choice is to use normal Ethernet or iSCSI. It has been my experience to expect at least 10-15 millisecond latencies with NAS, sometimes even higher.

DAS connections will be SCSI, SATA or PCIe. Most modern systems will be using SATA. Between SATA and PCIe, PCIe is the faster, but least flexible of the connection types as it will usually involve a proprietary card based interface. PCIe is usually reserved for SSD solutions.

Latency

Many times companies and individuals will try to tie latency and IOPS together. However, IOPS are easy to get even with horrendous latency. In fact, with some disk based systems as latency increases, so will IOPS until bandwidth is saturated (see Figure 14).

Don't get the wrong impression, lower latency will result in higher IOPS, but low latency is not required to get higher IOPS. As long as bandwidth is available you can add disks to a SAN and increase IOPS, however, each IO will still be between 2-5 ms or worse in latency!

Low latency's biggest benefit is in increased performance. The lower the latency the better chance that a transaction will take a shorter period of time to complete, thus response time is reduced and performance increases.

In a heavily cached SAN you can see 1 millisecond or better response times, until the cache is saturated (see Figure 2). Once the cache is saturated performance returns to disk based latency values. In most cases 1 millisecond is about the best latency, even from a heavily cached system, that you can expect in a disk based SAN. To break the 1 ms barrier you need to look at either DDR or Flash based SSD solutions. In a DDR based SSD latencies will be less than 20 microseconds (.02 ms) and in a Flash based SSD the latency should be between .080-.250 ms. Most enterprise level SSD systems will use a combination of DDR buffers and Flash for permanent storage.

Benchmarking Your Current Application IO

In order to accurately model databases to allow us to predict future needs based on current trends we must be able to control two specific parts of the database environment:

- User load
- Transaction mix

If you can not control the users or the transactions impacting the database then it becomes impossible to accurately predict the trends. However, if we are concerned with average loads and normal data growth then we must ensure that the transaction mix and user load when we do our measurements is "normal" if we don't know what a normal load is our efforts will probably result in inaccurate forecasts based on biased trends.

One way to completely control both the users and the transaction load is to utilize benchmarking tools. In the next section of this paper we look at the benchmark tools you can utilize to perform accurate trend analysis and prediction.

Trend identification with benchmark tools

If all that benchmarking tools did were standard canned benchmarks, they would be of limited use for trend analysis. Top of the line tools such as Benchmark Factory from Quest provide utilities that allow the reading of trace files from databases such as Oracle and the entry of SQL statements to test transactions for other database systems. In addition the tools should allow for the specification of multiple user scenarios so that insert, update, delete as well as select transactions can be modeled. Other tools such as Loadrunner and Mercury capture keystrokes and then use multiple users to provide for stress testing.

Performance review using Oracle

Oracle has provided SQL Replay in Oracle10g and Real application testing or database replay in Oracle11g. SQL Replay allows the capture and replay of a single set of SQL statements and these can then be used for a one-time run to perform regression testing in a before and after changes scenario. The database replay in Oracle11g allows capturing all, or portions of a running databases transactions and then replaying those transactions in another system. Both of the Oracle based tools do not allow for stress testing, they only do regression tests.

Of course there are older tools form Oracle such as Statspack and the automated workload repository (AWR.) these are more performance monitoring tools but can be used to perform before and after checks on a system providing the identical workload can be run at will.

Profiling an Oracle Storage Workload

Oracle keeps excellent storage statistics that are easily accessible from an AWR or statspack report.

Finding the three storage characteristics (bandwidth, latency, queue) and the two workload characteristics (average transfer size, application queue) is a straightforward exercise. The website www.statspackanalyzer.com automates this process and outputs the storage profile for an AWR/statspack report. It can be used to quickly performance these calculations.

IOPS, Bandwidth, and Average IO Size - Oracle 10g

Oracle 10g has a few new counters in AWR reports that simplify determining the IOPS and bandwidth that a database workload generates on a storage device. Simply collect an AWR report for a busy period for the database and go to the Instance Activity Stats and look for the per second values for the following counters:

```
physical read total IO requests – Total read (input) requests
per second
physical read total bytes – Read Bandwidth
```

```
physical write total IO requests – Total write (output)
requests per second
physical write total bytes – Write bandwidth
```

An example of this section of an Oracle 10g AWR report is shown below:

```
Instance Activity Stats  DB/Inst: RAMSAN/ramsan  Snaps: 22-23

Statistic                                     Total     per Second   per Trans
--------------------------------- ------------------ -------------- -----------
physical read IO requests                   302,759        4,805.7    20,183.9
physical read bytes                  35,364,380,672  561,339,375.8 ############
physical read total IO requests             302,945        4,808.7    20,196.3
physical read total bytes            35,367,449,600  561,388,088.9 ############
physical read total multi block r            292,958        4,650.1    19,530.5
physical reads                             4,316,960       68,523.2   287,797.3
physical reads cache                       4,316,941       68,522.9   287,796.1
physical reads cache prefetch              4,014,197       63,717.4   267,613.1
physical reads direct                              0            0.0         0.0
physical reads direct temporary t                  0            0.0         0.0
physical reads prefetch warmup                     0            0.0         0.0
physical write IO requests                       484            7.7        32.3
physical write bytes                       5,398,528       85,690.9   359,901.9
physical write total IO requests                 615            9.8        41.0
physical write total bytes                 7,723,520      122,595.6   514,901.3
physical write total multi block                 124            2.0         8.3
physical writes                                  659           10.5        43.9
physical writes direct                             0            0.0         0.0
physical writes from cache                       659           10.5        43.9
physical writes non checkpoint                   582            9.2        38.8
```

From this example you can see that the IO traffic is almost exclusively reads: ~560 MB/s bandwidth and ~4,800 IOPS. Using a variant of one of the formulas presented above, the average size per IO can be calculated as 116 KB per IO. The queue depth for this traffic can't be deduced from these measurements alone, because it is dependent on the response time of the storage device (where device latency, device bandwidth, and the maximum device queue are factors). Response time is available from another section of the report which is unchanged from earlier Oracle 8 and 9 statspack reports.

Bandwidth, IOPS, and Average IO Size

A single statspack report from an Oracle 9 database will be used for the remainder of this chapter so data from various sections can be compared, statspack and AWR reports from verions 9, 10 and 11 are similar and this information will apply to all versions.

Bandwidth

The bandwidth for a particular database can be found by looking at the physical reads and physical writes reported in statspack reports during a busy period of an Oracle database. For example, the load profile section near the top of a statspack report contains this data

```
Load Profile
~~~~~~~~~~~~                            Per Second       Per Transaction
                                   ---------------       ---------------
                  Redo size:           17,007.41             16,619.62
              Logical reads:          351,501.17            343,486.49
              Block changes:              125.08                122.23
             Physical reads:           11,140.07             10,886.06
            Physical writes:            1,309.27              1,279.41
                 User calls:            7,665.49              7,490.70
                     Parses:               14.34                 14.02
                Hard parses:                4.36                  4.26
                      Sorts:                2.85                  2.78
                     Logons:                0.17                  0.17
```

```
            Executes:              22.41                21.90
       Transactions:               1.02
```

It is important to note, however, that these are recorded as database blocks per second rather than IOs per second. This means that a single large read of 16 sequential database blocks has the same count as 16 single block reads of random locations. Since the storage system will see this as one large IO request in the first case and 16 IO requests in the second, the IOPS can't be determined from this data.

To determine the bandwidth the database block size needs to be known. The standard database block size is recorded in the header for the statspack.

```
STATSPACK report for

DB Name         DB Id    Instance     Inst Num Release     Cluster Host
------------ ----------- ------------ -------- ----------- ------- ----------
MTR           3056795493 MTR                 1 9.2.0.6.0   NO      Oraprod

            Snap Id     Snap Time      Sessions Curs/Sess Comment
            ------- ------------------ -------- --------- -------------------
Begin Snap:   25867 13-Dec-06 11:45:01       31        .9
  End Snap:   25868 13-Dec-06 12:00:01      127       7.5
   Elapsed:               15.00 (mins)

Cache Sizes (end)
~~~~~~~~~~~~~~~~~
               Buffer Cache:     7,168M      Std Block Size:         8K
           Shared Pool Size:       400M          Log Buffer:     2,048K
```

The IO profile for this database is mainly reads, and is ~12,500 database blocks per second. For this database the block size is 8 KB. The bandwidth of the database is the physical reads and physical writes multiplied by the bandwidth, in this case about 100 MB/s (90 MB/s reads and 10 MB/s writes).

IOPS and Response Time

The IOPS, bandwidth, and response time can be found from the Tablespace IO Statistics section if the database block size is known[1].

```
Tablespace IO Stats for DB: MTR  Instance: MTR  Snaps: 25867 -25868
->ordered by IOs (Reads + Writes) desc

Tablespace
------------------------------
                 Av      Av     Av                    Av        Buffer Av Buf
         Reads Reads/s Rd(ms) Blks/Rd       Writes Writes/s      Waits Wt(ms)
-------------- ------- ------ ------- ------------ -------- ---------- ------
MSERVERTAB
     7,861,586   8,735    5.5     1.0       59,214       66     45,542   28.9
MSERVERIND
       884,275     983    5.0     1.0       24,261       27        925   19.1
TEMP
       122,465     136    7.7     8.9      121,028      134          0    0.0
TOOLS
         1,166       1    1.3     1.5          452        1          0    0.0
UNDOTBS1
            66       0    5.6     1.0          353        0          2    0.0
SYSTEM
            51       0    9.0     1.0            9        0          0    0.0
          -------------------------------------------------------------
```

[1] The tablespace IO statistics omits the IO for a few database components (redo logs, LOB reads and writes, etc), but covers the vast majority of IO for database where storage performance is a concern.

If you have more than one tier of Storage available you can use the Tablespace IO Stats to determine the table spaces that can benefit from faster storage the most. To do this take the Av Reads/s multiplied by the Av Rd (ms) and save this data. This is the amount of time spent waiting on IO for a tablespace. Next find the capacity of each tablespace. The tablespaces with the highest amount of wait time per capacity are the top candidates for higher performance storage.

The Av Reads/s and the Av Writes/s for each tablespace are shown; these can be summed across all tablespaces to get the IOPS for the database. In this case the IOPS is about 10,000 IOPS and predominantly reads. The average read response time is presented as Av Rd(ms) for each tablespace. To find the overall Read response time, take the weighted average response time across all table spaces using the Av reads/s for the weighting. For this database it is ~5.5 milliseconds. The write response time for the tablespaces isn't directly measureable because the DBWR process is a background process. For simplicity just take the read response time to be the storage response time because writes in Oracle are either storage array cache friendly (logs) or background processes that don't have a big impact on database performance[2].

The storage workload for this database is now fully defined:

- Predominantly reads
- 100 MB/s
- 10,000 IOPS
- 5.5 ms response time
- The average request size and queue for the profile can be found using the formulas provided earlier:
- Application queue depth: ~55
- Average request size: 10 KB

Benchmarking

For many large Oracle databases, building test environments is next to impossible. Many incorrect conclusions can be drawn by a test environment with a load that doesn't compare to production. Even if the expensive production hardware is available for testing, the ability to generate a heavy user work load isn't a possibility. If an application-level test is performed, it is critically important to measure the storage workload of the test environment and compare it to what was measured in production before beginning any software or hardware optimization changes. If there is a big difference in the storage profile, then the test environment is not a valid proxy for production.

For example, if a dedicated enterprise storage array with a large cache is available in a test environment, the response time could be quite good if the test dataset fits mainly within the cache. However, if in production the array supports a wide number of applications and the array cache can be considered a dedicated resource, response time of the array will be wildly different. Applications that perform wonderfully in test may fail miserably in production.

---

[2] If a database is experiencing a high level buffer waits this can be a symptom of a storage write performance issue as dirty blocks aren't getting flushed out to disk fast enough to free the buffer cache. However, if the issue is storage performance related, it will be because the maximum queuing of the storage device is being exceeded. These will manifest itself in high read response times due to the queuing wait time component of the read response time. So even in this case the read response time can be used.

Using the storage profile measured for an application can be used to create a synthetic environment to compare how various storage devices would compare to the production workload. This can be helpful to evaluate different storage devices after storage has been found to be the bottleneck for a database.

Storage Benchmarking Tools

There are lots of tools available to generate storage workloads. Two of these will be examined in detail, IOMeter and Oracle's IO benchmarking utility: ORION. These benchmarking tools don't actually process any of the data that is sent or received to a storage array so the processing power requirements of the server running the test are slight. As many storage vendors will provide equipment free of charge for a demo or trial period for prospective customers, the costs associated with running these tests are minimal.

IOMETER

IOmeter is a free tool available at www.iometer.org that can be used to create virtually any I/O workload desired. It is a very powerful tool that can be a bit intimidating on the first use but is easy to use once the main control points are identified. It is easiest to use on a Windows platform.

Setup and Use

Launching IOmeter will start a control GUI as well as a separate process that actually submits the I/O. The important sections of the GUI to control an I/O profile will be highlighted and then the process to run a storage workload will be described. The control GUI is illustrated in Figure 9.

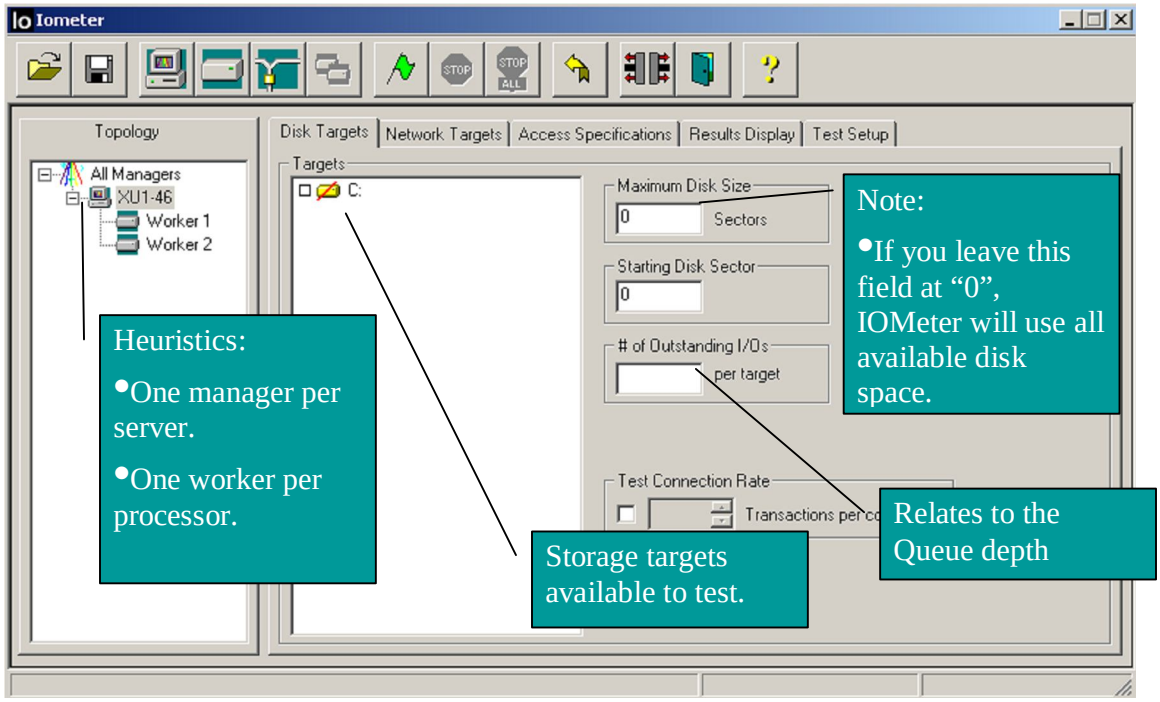

Figure 9: IOMeter Control GUI

On the left hand side under the "All Managers" tree item, the computer name is listed with a number of workers. By default, IOmeter will create one worker per CPU core (including hyper-threading cores). A separate I/O load can be created for each worker for more advanced application simulation. The first area where the I/O load is specified is under the disk targets tab. This lists the disk targets available for testing, allows specifying a capacity range for tests, and

allows setting the "# of Outstanding I/Os." All of these values can be set on a per worker basis; however, to simplify the testing a single worker can be used. The "# of Outstanding I/Os" is the same as the application queue depth in a single worker case, and with multiple workers the application queue depth is just the sum of all of the workers' outstanding I/Os.

The easiest drives to test haven't been formatted with a file system yet. If a file system has been created, a forward slash will appear through the picture of the target with the volume label ("C:" in the example above). These can still be tested, but a test file has to be created on the drive. By default IOmeter will create a test file that uses all free capacity (for large drives this could take a long time); the "Maximum Disk Size" setting allows limiting the size to the sectors specified (there are 2 sectors per KB). To ensure valid test results, be sure to set the test file size to be much larger than any cache on the storage device.

The next major configuration section is on the access specification tab, this is shown in Figure 10.

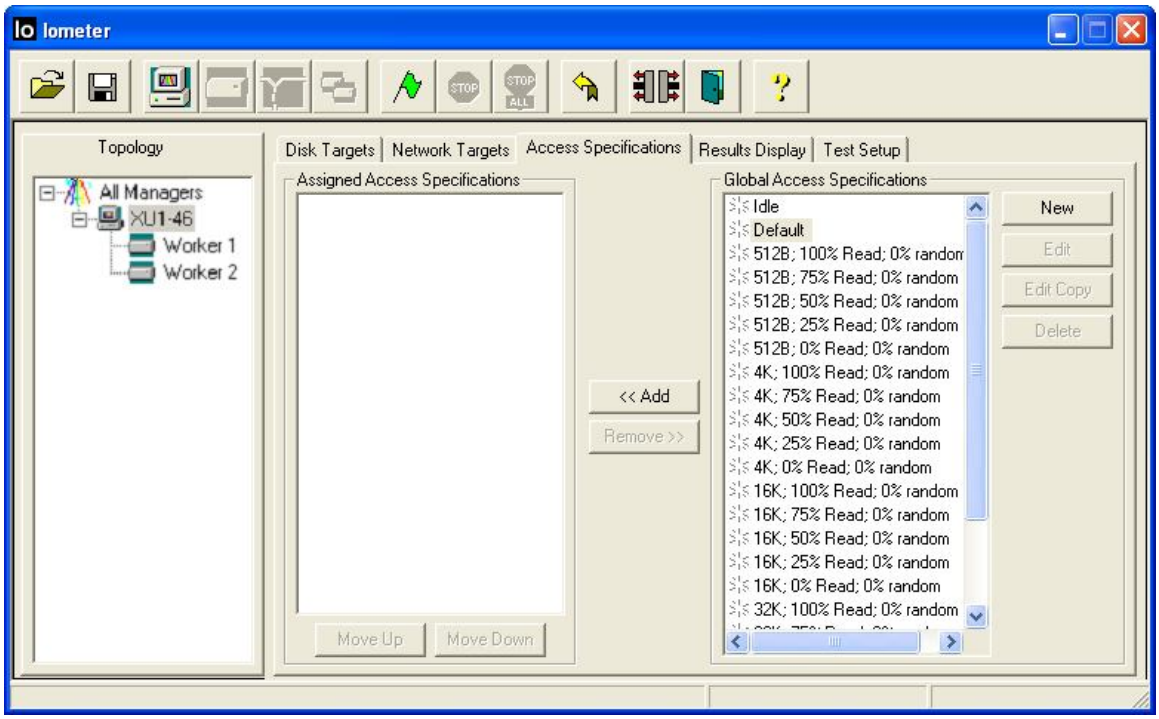

Figure 10: IOMeter Configuration Screen

With the GUI in Figure 3 the specifics of the workload can be created and assigned to a worker. Selecting the "Default" access specification and clicking the "Add" button will assign this access specification. After it is assigned, it can be edited by clicking the edit button and making changes in the pop-up window that opens, this window is shown in Figure 11.

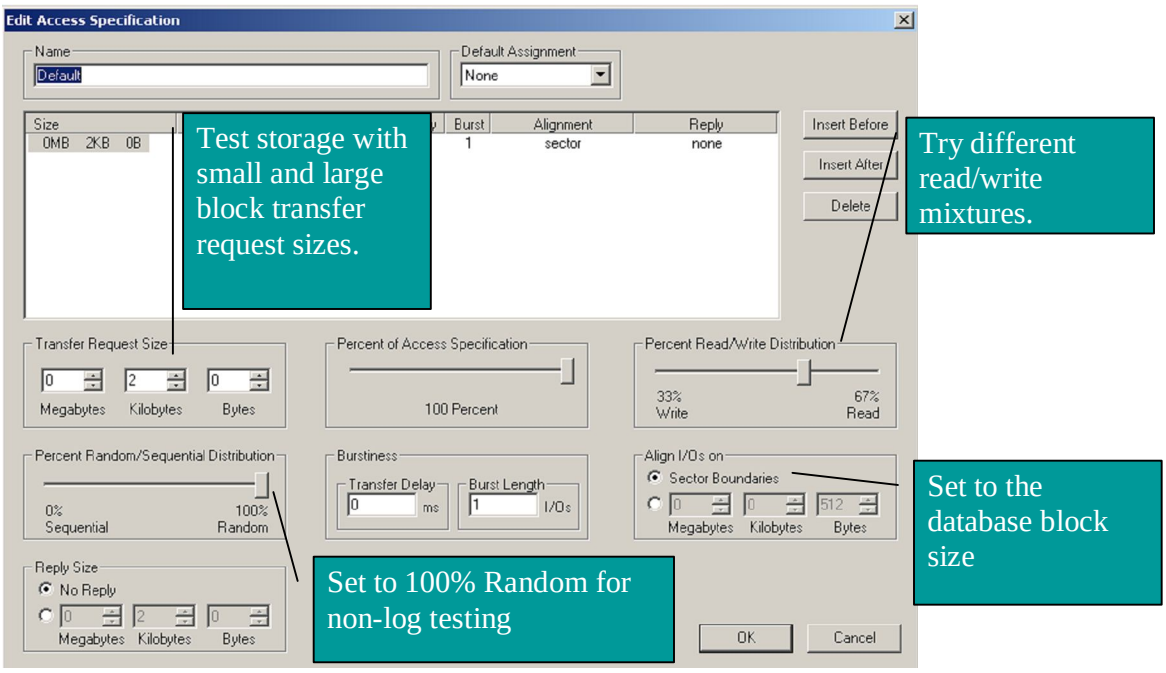

Figure 11: IOMeter Configuration Pop-Up

The Access specification can be changed to match a particular storage profile. The transfer request size, read vs. write ratio, percent random, and access alignment can all be easily set from this window.

Simulating a storage profile

A basic simulation of a storage profile that was determined from AWR/statspack analysis is very straightforward with IOmeter. Simply set a single worker to test a target storage device and put the measured storage profile metrics in the following configuration locations:

- **Measured Application Queue = # of oustanding I/Os**
- **Average Request Size = Transfer Request size.**
- **Set the measured read/write distribution.**
- **Set the test to 100% random**
- **Align I/Os by the *Std Block Size*.**

Example Runs

Once the storage workload information is entered into IOmeter, clicking the green flag will start the test. The "Results Display" tab will show the performance of the storage under the load.

To illustrate the differences between different storage devices, a test storage workload with the following parameters was used:

8 KB blocksize, 100% read access, with a queue depth of 4.

The first run was on a RamSan-300 RAM-based SSD, the output screen from the test is shown in Figure 12.

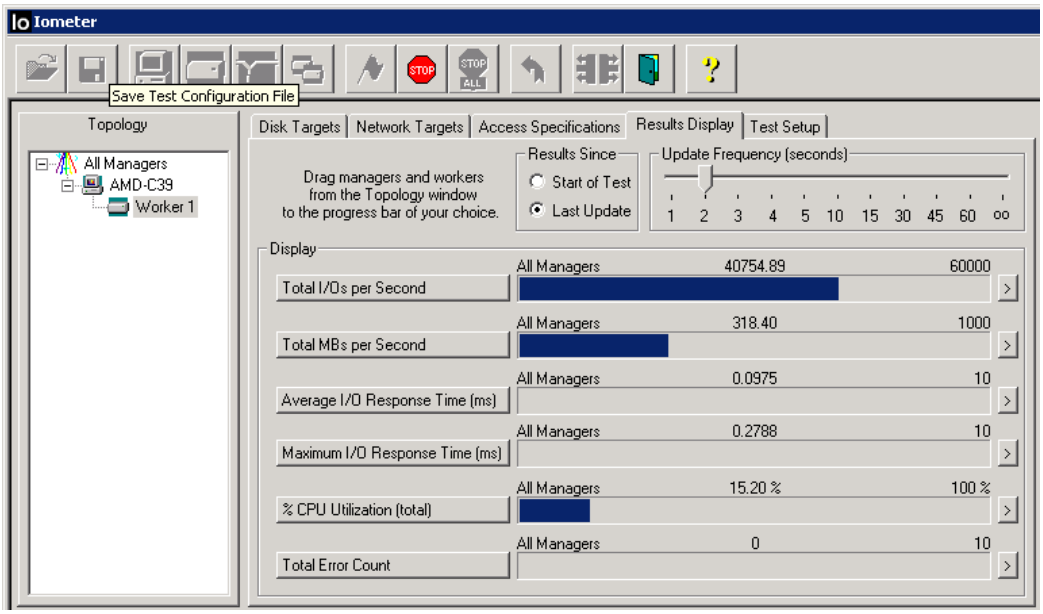

Figure 12: IOMeter output screen for the RamSan300

Recall that Little's Law relates response time, queue depth, and IOPS. Here IOPS (40755) * response time (0.0000975 seconds) = 4 as expected.

The IOMeter output screen for the same storage profile run on an HDD-based array with over 90 drives is shown in Figure 13.

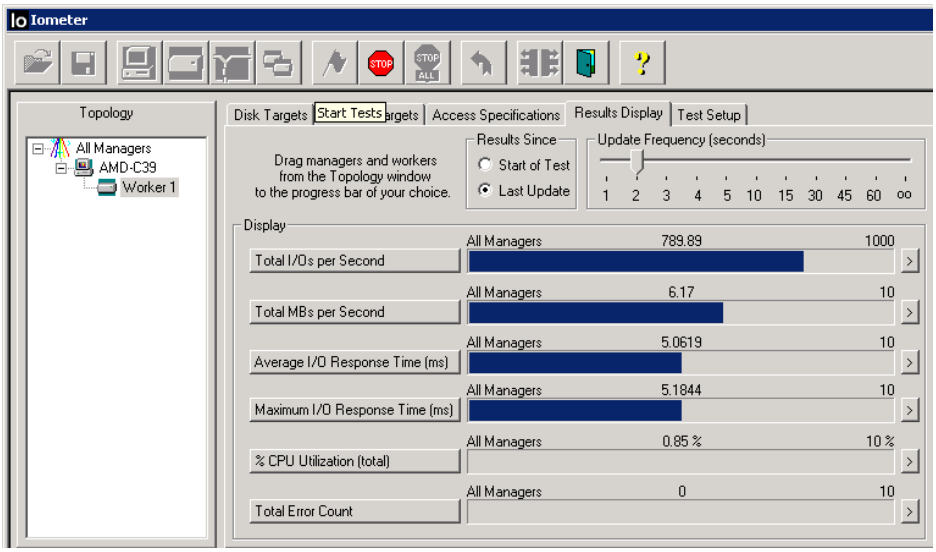

Figure 13: IOMeter output screen for 90 disk hard drive test.

Notice that Littles Law is still maintained: IOPS (790) *response time (0.00506 seconds) = 4, the workload's queue depth. The dramatic difference is that IOPS under these identical storage workloads is driven by the much lower response time of the RAM-based SSD.

Oracle's Orion

ORION (**ORacle I/O Numbers**) is a tool designed by Oracle to predict the performance of an Oracle database without having to install Oracle or create a database. It expressly simulates Oracle I/O by using the same software stack and can simulate the effect of disk striping on

performance done by ASM. It has a wide range of parameters and can be used to fully profile a storage device's performance or run a particular I/O workload against a storage array. ORION is available from the following URL (Requires free Oracle Technology Network login): http://www.oracle.com/technology/software/tech/orion/

Binaries are available for:

- **Linux, EM64 Linux**
- **Solaris**
- **Windows**
- **AIX**

ORION is a command line driven utility; however, a configuration file is also required. An example ORION command would be:

```
EX: orion –run simple –testname <Configuation File> -num_disks 8
```

The configuration file contains the path to the physical drives being tested.

For example, under Windows "\\.\e:" would be the only line in the file for testing only the E: drive. Under Linux /dev/sda specifies testing SDA. Multiple drives can be listed in the configuration file to simulate ASM.

Example Runs
The easiest way to use ORION is to use its default simple storage test to profile a storage device, then use this profile as a lookup table for a particular workload. The user's guide that comes with ORION can be referenced for more advanced testing.

To run the simple test, create a configuration file for the drive(s) that you want to test and then run the command "orion –run simple –testname <Configuation File> -num_disks 8". The simple test only performs reads to prevent overwriting data that is on the drive(s). The test will run for approximately 30 minutes and will output several files; three are needed to profile the storage:

```
<Configuation File>_mbps.csv
<Configuation File>_iops.csv
<Configuation File>_lat.csv
```

These files are formatted to open in a spreadsheet with the columns corresponding to the number of small outstanding I/O requests and the rows corresponding to the number of large I/O requests. In the simple mode of the test the small I/O size is set to 8 KB and the large I/O is set to 1 MB. This is meant to simulate single block reads and full table scans. In the simple test, large and small I/O requests aren't mixed, so the latency and IOPS data is captured by using only small I/Os and the bandwidth is captured using only large I/Os. The outstanding I/Os are linearly increased for both the large and small I/O case to provide a profile of the storage device's performance. An example of the latency and IOPS output from the simple test on the RamSan-300 is shown in Figure 14.

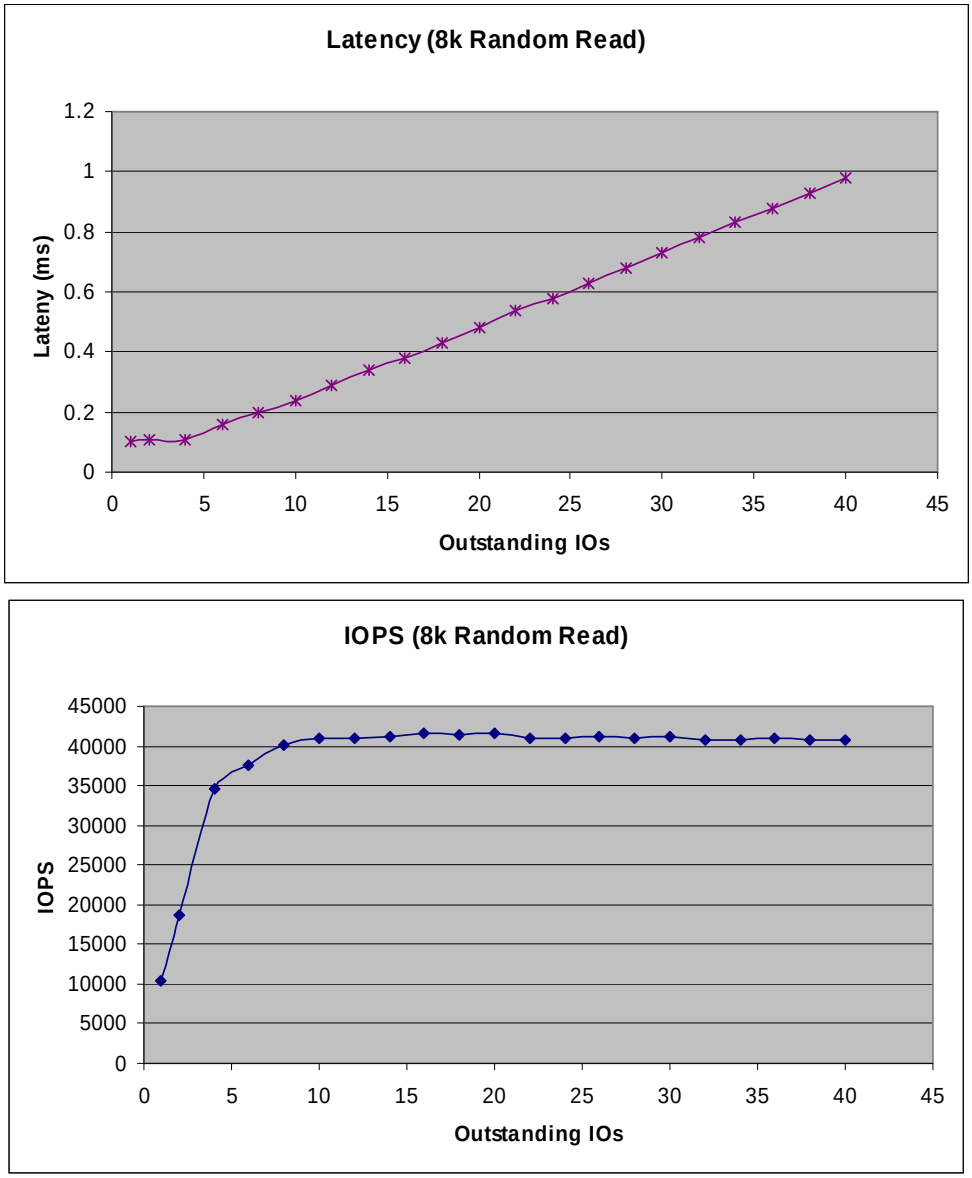

Figure 14: Output Data from ORION plotted in Excel

The output data from ORION can be used with the application storage profile to compare different storage device's behavior under a particular workload. First run a "long" version of the test to get the performance of mixed small and large I/Os for a full storage profile. Next, take the measured average request size to determine the right ratio of small and large I/O requests. The measured queue depth is the sum of the large and small I/O requests, and combined with the small/large ratio will identify which data point in the output files from ORION is closest to the measured storage profile. The profiles of different storage devices can be compared at this point to determine if they would be better for this workload.

Using DBMS_RESOURCE_MANAGER.CALIBRATE_IO

Oracle11g provides the DBMS_RESOURCE_MANAGER.CALIBRATE_IO PL/SQL procedure. This procedure generates read activity against all data files either at the database block size or at 1 megabyte IO sizes. Within the procedures limitations it can be used to do a read profile of your existing IO subsystem. The calibration status is shown in the V$IO_CALIBRATION_STATUS

view and results from the last successful calibration are located in the DBA_RSRC_IO_CALIBRATE table.

Running CALIBRATE_IO

To perform a calibration using CALIBRATE_IO you must have the SYSDBA privilege and the database must use async IO: *filsystemio_options* set to ASYNC or SETALL, with *timed_statistics* set to TRUE. Only One calibration run is allowed at a time. In a RAC environment the IO is spread across all nodes.

To run the CALIBRATE_IO procedure, provide the maximum number of disks and the maximum acceptable latency (between 10-100 milliseconds) and the CALIBRATE_IO package returns the maximum sustained IOs per second (IOPS) (db block size random reads), throughput in megabytes per second (1 megabyte random reads) and maximum throughput for any one process in process megabytes per second. The table in Figure 15 shows these input and output variables.

PARAMETER	DESCRIPTION
num_physical_disks	Approximate number of physical disks in the database storage (input)
max_latency	Maximum tolerable latency in milliseconds for database-block-sized IO requests (input)
max_iops	Maximum number of I/O requests per second that can be sustained. The I/O requests are randomly-distributed, database-block-sized reads.
max_mbps	Maximum throughput of I/O that can be sustained, expressed in megabytes per second. The I/O requests are randomly-distributed, 1 megabyte reads.
max_pmbps	Maximum megabytes per second of large I/O requests that can be sustained by a single process
actual_latency	Average latency of database-block-sized I/O requests at max_iops rate, expressed in milliseconds

Figure 15: Table of CALIBRATE_IO Variables

Oddly, when testing using ASM I had the filesystemio_options set to none (the default) and it worked.

Using CALIBRATE_IO

The tests in this tip were run using a dual node RAC cluster with a NexStor18F-18 disk array and a NexStor8F-8 disk array for storage, 2 of the disks were used for OCFS leaving 24 disks for use in ASM. The disk arrays utilized a 1GB fibre channel switch via dual paths and RedHat 4 mpio software. The system ran on Oracle11g with the 11.1.0.7 release of the software.

Finger Printing Your IO Subsystem

Use the following methods to fingerprint the maximum expected IOPS and ideal number of disks for your IO subsystem.

Keeping Latency Constant and With Disk Count Increasing

The first test is for various numbers of disks (1 to n, I used 1-30 for the 24 disk subsystem) and a fixed allowable latency (for disks: 30 ms). Use a PL/SQL procedure that runs the CALIBRATE_IO routine and stores the results in a permanent table called CAL_IO, which is created using a CTAS from the DBA_RSRC_IO_CALIBRATE table. The PL/SQL to use is show Listing 3.

```
set serveroutput on
Declare
     v_max_iops          PLS_INTEGER:=1;
     v_max_mbps          PLS_INTEGER:=1;
     v_actual_latency    PLS_INTEGER:=1;
      i integer;
begin
for i in 1..30 loop
     dbms_resource_manager.calibrate_io(i,30,
          max_iops=>v_max_iops,
          max_mbps=>v_max_mbps,
          actual_latency=>v_actual_latency);
     dbms_output.put_line('Results follow: ');
     dbms_output.put_line('Max IOPS: '||v_max_iops);
     dbms_output.put_line('Max MBPS: '||v_max_mbps);
     dbms_output.put_line('Actual Latency: '||v_actual_latency);
insert into io_cal select * from dba_rsrc_io_calibrate;
commit;
end loop;
end;
/
```

Listing 3: MAX_IOPS.SQL PL/SQL Procedure

The results for your first test will resemble the output shown in Figure 16.

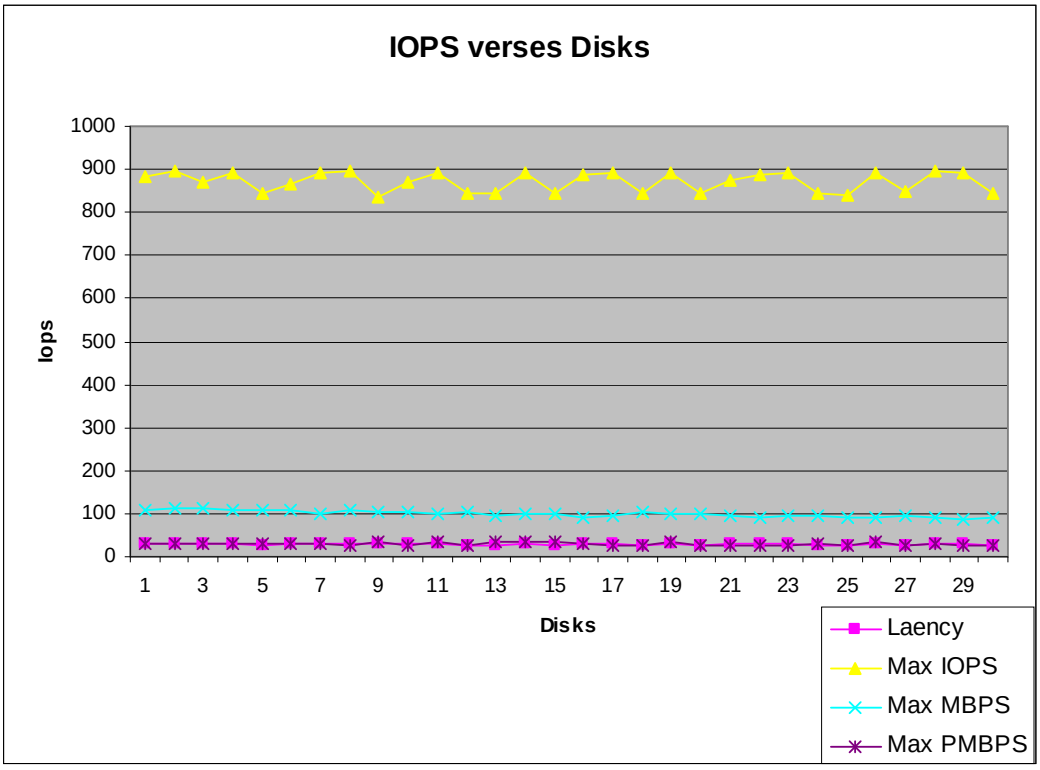

Figure 16: IOPS verses Disks with a Fixed Latency

As you can see from figure 16 the IOPS stayed at around 850 regardless of the number of disks specified at a maximum latency of 30 milliseconds. How the number of disks input is actually being used is not certain, but it didn't make any difference to the calculation of the various results.

The maximum throughput hovered around 100 megabytes and the process maximum throughput hovered around 30 megabytes per second. However, there was a downward trend in the max throughput number as disk count increases, notice how the blue line (second form the bottom) in the graph in Figure 16 starts mostly above 100 and ends mostly below 100?

Keeping Disk Count Constant and Increasing Latency - HDD

In the second test hold the number of disks constant at what level you deem best (I choose 6) and vary the acceptable latency. The PL/SQL code to use is shown in Listing 4.

```
set serveroutput on
Declare
     v_max_iops          PLS_INTEGER:=1;
     v_max_mbps          PLS_INTEGER:=1;
     v_actual_latency    PLS_INTEGER:=1;
      i integer;
begin
for i in 10..100 loop
if (mod(i,5):=0) then
     dbms_resource_manager.calibrate_io(I,30,
          max_iops=>v_max_iops,
          max_mbps=>v_max_mbps,
          actual_latency=>v_actual_latency);
     dbms_output.put_line('Results follow: ');
     dbms_output.put_line('Max IOPS: '||v_max_iops);
     dbms_output.put_line('Max MBPS: '||v_max_mbps);
     dbms_output.put_line('Actual Latency: '||v_actual_latency);
  insert into io_cal select * from dba_rsrc_io_calibrate;
  commit;
end if;
end loop;
end;
/
```

Listing 4: MAX_IOPS.SQL used for varying the Latency

The results from varying the max allowed latency between 10 and 100 ms, in increments of 5 ms, are shown in Figure 17.

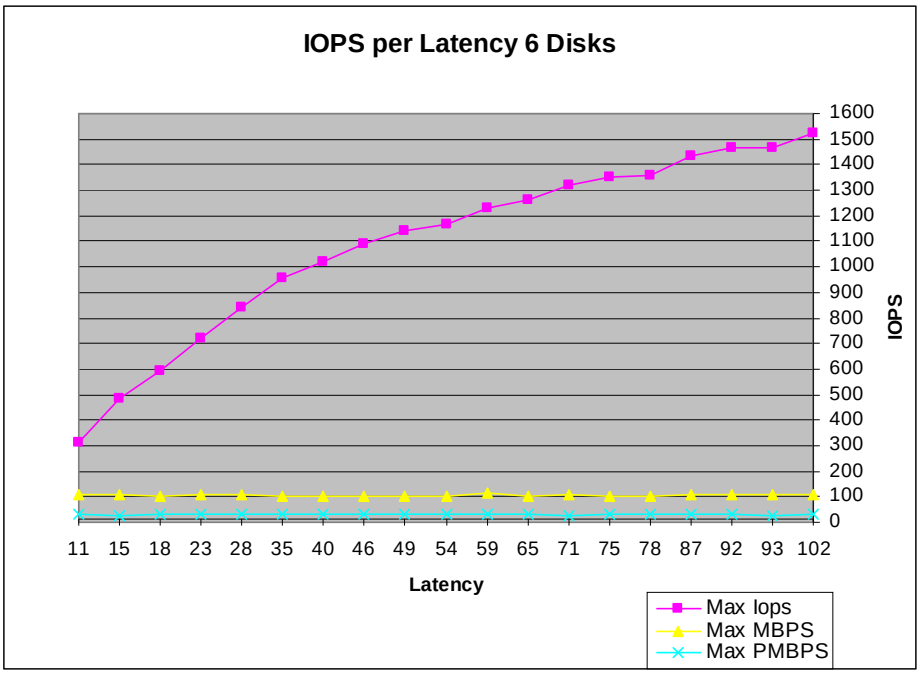

Figure 17: IOPS verses Max Allowed Latency

This is an unexpected result based on a derivation of Little's law (IOPS=QUEUE/Response time) which seems to indicate that as the response time (latency) is allowed to increase with a fixed queue size (6 disks) the IOPS should decrease. It can only be assumed that the number of disks is not being used as a queue size determiner in the procedure. What this does show as that if we allow latency to creep up, as long as we don't exceed the queue capability of the IO subsystem, we will get increasing IOPS.

As in the first test the maximum throughput hovers around 100, but, there is no over all downward trend, seemingly to indicate a relationship to the number of disks input is causing the downward motion of the values. The process maximum through put hovered around 30, just like in the previous test.

During my tests the average queue according to iostat was at 16 with jumps to 32 on some disks but not all. The utilization for each disk was at about 40-50 with peaks as high as 90 depending on the test. I suggest monitoring queue depth with OS monitoring tools during the test.

Based on the two result sets you should be able to determine the best final run settings to load the DBA_RSRC_IO_CALIBRATE table with your best expected results.

How About With SSDs?

But what about using this for SSDs? Well, remember what I said above about the minimum and maximum for the max allowed latency? Yep, it starts at 10 milliseconds. That alone makes it useless for profiling SSD based components because even the worse performing SSDs achieve max latencies of less than 2 milliseconds with most being less than 1 millisecond. So whoever the developer was for this package they obviously have never heard of SSDs, has never profiled SSDs and wasn't addressing this for use with SSDs. Rather like coming out with a new and improved buggy whip in 1903 (the year the Ford Motor Company was created) an interesting product with a limited lifetime. The package should be re-written to allow starting at 100 microseconds and ramping up at 100 microsecond intervals to 2000 microseconds to be useful for SSD profiling and testing. Take a look at this result testing SSDs for varying latency in Figure 18.

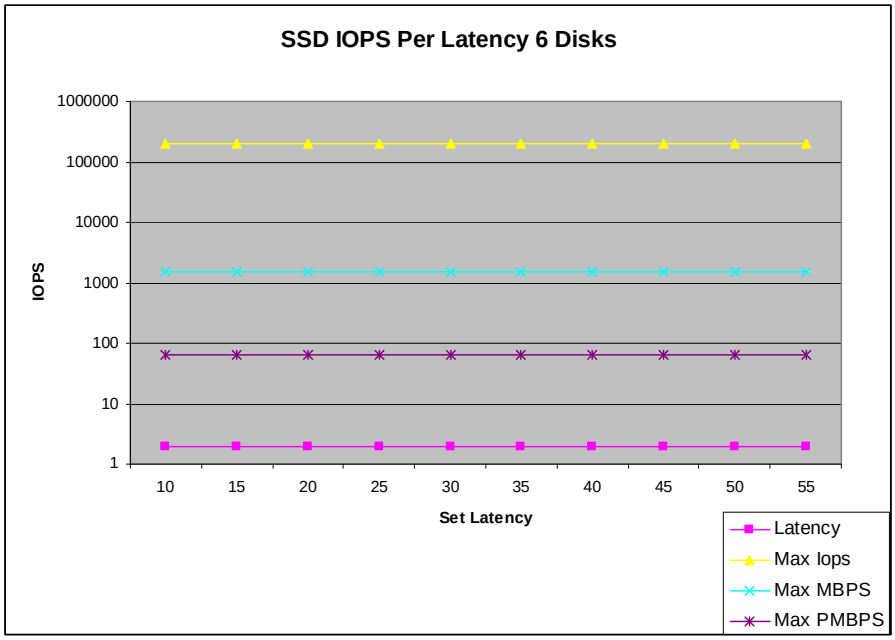

Figure 18: Set number of disks, varying latency, SSD

As you can see, at even the lowest allowed latency, 10ms, the IOPS for the SSD are maxed out since in this particular case the maximum latency is actually less than 1 millisecond.

One last problem, the procedure only tests the "I" part of IO. The procedure does no write testing, only read testing. I guess this is part of Oracle's overall blindness to write times. I agree that with the lazy write algorithms (delayed block cleanout for you old timers) that writes aren't generally important in the context of Oracle performance, except when those writes interfere with reads, which for disk systems is generally always. So again the procedure comes up a little short in true functionality.

So, while the CALIBRATE_IO procedure has limited usability for disk based systems (most of whom have less than 10 ms latency) it will have less and less usefulness as SSD's push their way into the data center.

Conclusions:

Understanding storage's behavior can be a very powerful tool. The mathematical relationships can seem a bit unwieldy but are actually quite simple and easy to master. Oracle provides very comprehensive data in AWR or statspack reports that include a wealth of data about what load the application is creating on the storage device and how the storage is handling it. Using this data, various storage devices can quickly be tested in a simple benchmark environment that accurately simulates a large production workload.

Michael Ault
Texas Memory Systems, Inc
5035 Forest Run Trace
30022, Johns Creek, Georgia, USA

Phone: +01(1) 770-754-9056
Fax: +01(1) 770-754-9949
Email mike.a@ramsan.com
Internet : http:/www.ramsan.com/