THE ALL FLASH DATABASE – IT'S NOT JUST CACHE ANYMORE

Randal Cook, Violin Memory

Abstract

In today's ever growing database environments and the age old plea for performance, the requirement for faster, denser storage is rapidly becoming necessity rather than luxury. This paper highlights the presence of NAND based flash storage media, how it addresses performance bottlenecks inside the Big Data environments, and educates the reader of the different types of flash in the market today and how they can be implemented. Furthermore, Oracle specific reports are discussed to demonstrate how to diagnose a system for IO bottlenecks.

Target Audience

All individuals assessing the relevance of flash based storage being introduced into their company infrastructure or desiring a further clarification as the nature and uses of NAND based flash storage media.

Executive Summary

Reader will be able to:

- Define the different types of NAND flash prevalent in the market today and their different characteristics.
- Summarize the performance indicators within an existing system that point towards an IO bottleneck that could be addressed by flash based storage.
- Discuss the key differences in an all-flash database implementation versus a rotating disk implantation.

Background

Since the inception of the computational world and especially in the database environment where accessing and presenting data is the name of the game, the speed at which data can be accessed has remained a primary objective in the overall design of a system. Server CPU has become faster and more powerful with eight cores being the norm while the internal DRAM has reached terabyte levels. Networks have also gone through innovative changes to incorporate the industry's gigabyte per second standard conversation. The third piece of the architectural puzzle however, has not been able to keep up pace. Because of the mechanical design of spinning disks, shear physics alone have dictated that the storage subsystem has remained the governor to just how fast systems can go.

By removing the last mechanical moving part from the database environment and replacing the outdated spinning drive storage with solid state NAND based storage the last piece of the puzzle will finally be able to keep pace with the innovations that have already introduced solid state into the server and networking environments.

Technical Discussions and Examples

NAND Flash

What is it?

Invented by Dr. Fujio Masouka in 1980 while working for Toshiba, NAND flash, often referred to as flash memory, is a non-volatile storage media that can be electrically erased and reprogrammed. Non-Volatile meaning that unlike DRAM the storage does not require electrical power to retain any data being stored on it. Its electrically erasable and reprogrammable characteristic define it as having no moving parts taking physical limitations out of the equation as to its speed and storage density. The term NAND refers to the type of logic gates comprising the storage, the second type being NOR flash type storage. NAND flash storage reads and writes in blocks of data where NOR flash reads

and writes in single bytes. To get into the more in-depth electrical properties, Wikipedia has a great discussion of flash at http://en.wikipedia.org/wiki/Flash_memory.

IOPS / Latency / Bandwidth – Learning the Nomenclature

IOPS – Input Output operations Per Second – The base of storage related metrics, this term refers to a common performance indicator for computer storage devices. The higher the number of IOPS, the faster the media is perceived to be; the more work it can do in any given amount of time.

Latency – The measurement of the time delay between the CPU requesting a block of data and the time the IO is serviced and received. In the physical hard drive world this measurement will contain several different parameters to the overall formula such as head assembly seek time and rotational disk latency which add to the overall storage latency. In a flash environment these mechanical aspects are removed from the equation resulting in a much lower overall latency. Typical HDD average rotational latencies range from greater than 7 milliseconds for 4200 RPM spinning drives to 2 milliseconds for the fastest 15K RPM drives. By comparison NAND flash is documented to have average latencies well below 500 microseconds with SLC numbers reaching 100 microseconds or 20x less than the fastest hard drive.

Bandwidth – The measure in bits or bytes per second of the transfer of data from a storage device to the calling server.

SLC versus MLC

Single-level cell (SLC) flash stores a single bit of information whereas in the multi-level cell (MLC) type of flash more than one bit of information can be stored per cell allowing for a denser level of storage. However, the density comes at a cost to both performance and longevity of the flash based on the characteristics of the flash. In the MLC model the multiple bits of information are achieved by choosing between multiple levels of electrical charge applied to the floating gate transistors making up each cell of the flash thus taking more time and wearing the cells at a faster rate. Therefore, when choosing between SLC and MLC flash the main question to ask is "do I need performance or is capacity the key decision factor?"

PCI Cards

Host based flash storage physically installed within the server sharing the server resources.

Pros:

- Entry level flash solution. As long as the server has a PCI slot available this is an easy install.
- Smaller increments to buy into starting at 100s of GB upwards to 10 - 20TB.

Cons:

- Each card is installed within a server creating islands of non-sharable flash.
- Consumes server resources
- Can produce a single point of failure

NAND Flash Storage Arrays

Through the aggregation of several modules of NAND flash, these external devices provide a standalone storage solution not reliant upon the calling server for any resources. Because of the external nature of the array, these devices can often be added to an existing storage area network (SAN) to be utilized as primary storage for several different servers accessing the network allowing for a much more diverse utilization of the NAND flash storage versus dedicating the flash to a single server and purpose. Standard networking protocols allow for the connectivity of the array to the storage infrastructure.

Pros:

- A self-contained system. Containing their own power supplies, cooling systems, processors, and memory structures that do not rely on the server utilizing the storage for any resources.
- Sharable storage. Connected to the SAN, the flash storage can be utilized by multiple servers and systems.

- Vendor specific models provide a very highly available storage solution for the tier one enterprise solution. Redundant, hot swappable components provide a solution that does not take the system down when a component fails.
- The external nature of the array also provides for a completely scalable solution. Unlike the internal PCI card solution constrained by the number of slots within the server, arrays can be aggregated within the SAN to provide for unlimited NAND flash storage.
- Because the array is a storage system, certain vendors are able to provide functionality for the storage that has commonly been adopted by the industry as a norm such as de-duplication, compression, encryption, cloning, and replication.

Cons:

- Due to the highly available and storage subsystem natures of the array, as well as the additional capacity provided by an array often totaling 50 – 100 terabytes per unit, the price point is usually higher at the entry level.

IO Latency

While the extreme density and power reduction attributes are often the compelling reasons why some companies turn to flash as their ongoing storage solutions, the extreme performance of flash storage when compared to spinning drive systems is the true driving force for this game changing technology. Diagram 1 below illustrates the latency found in typical storage platforms in the industry today. With an average latency in a standard 15K RPM disk array it can be seen that a typical IO will take 20 times longer to return than the slower MLC flash and over 50 times longer than SLC flash. In diagram 2 below this concept is reiterated by illustrating the additional amount of work a server is able to accomplish as a result of the drastically reduced latency found in NAND flash.

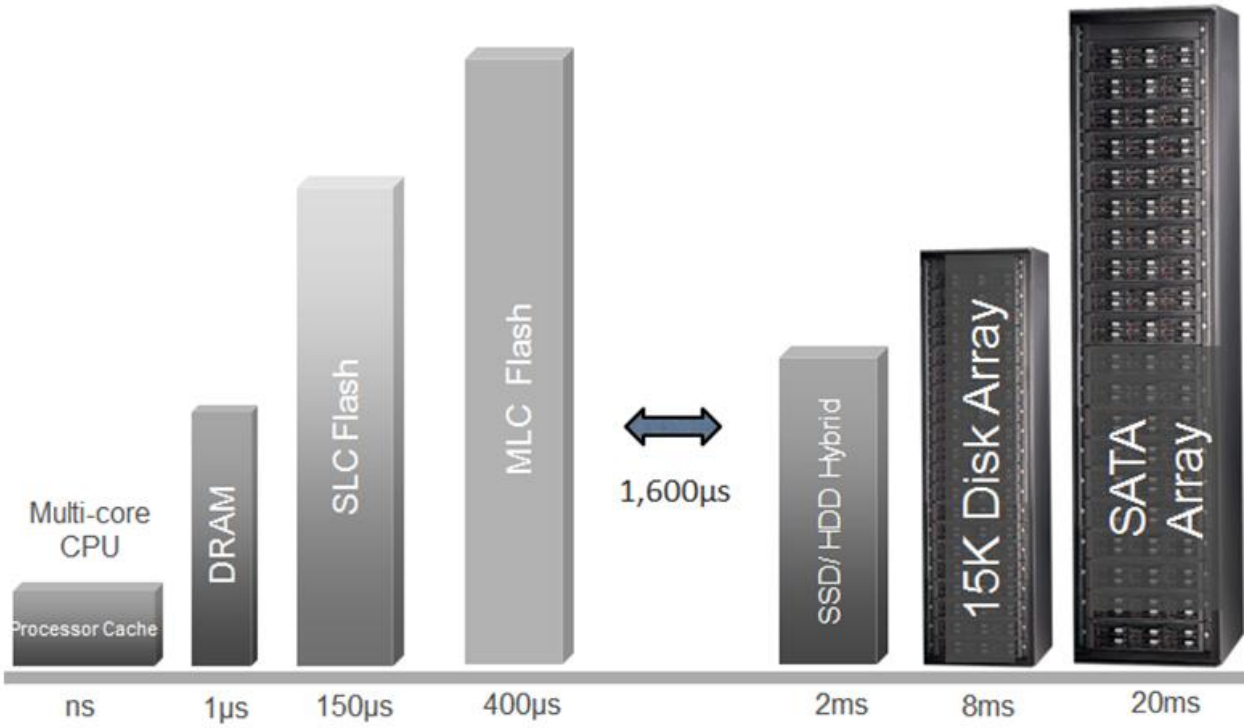


Diagram 1

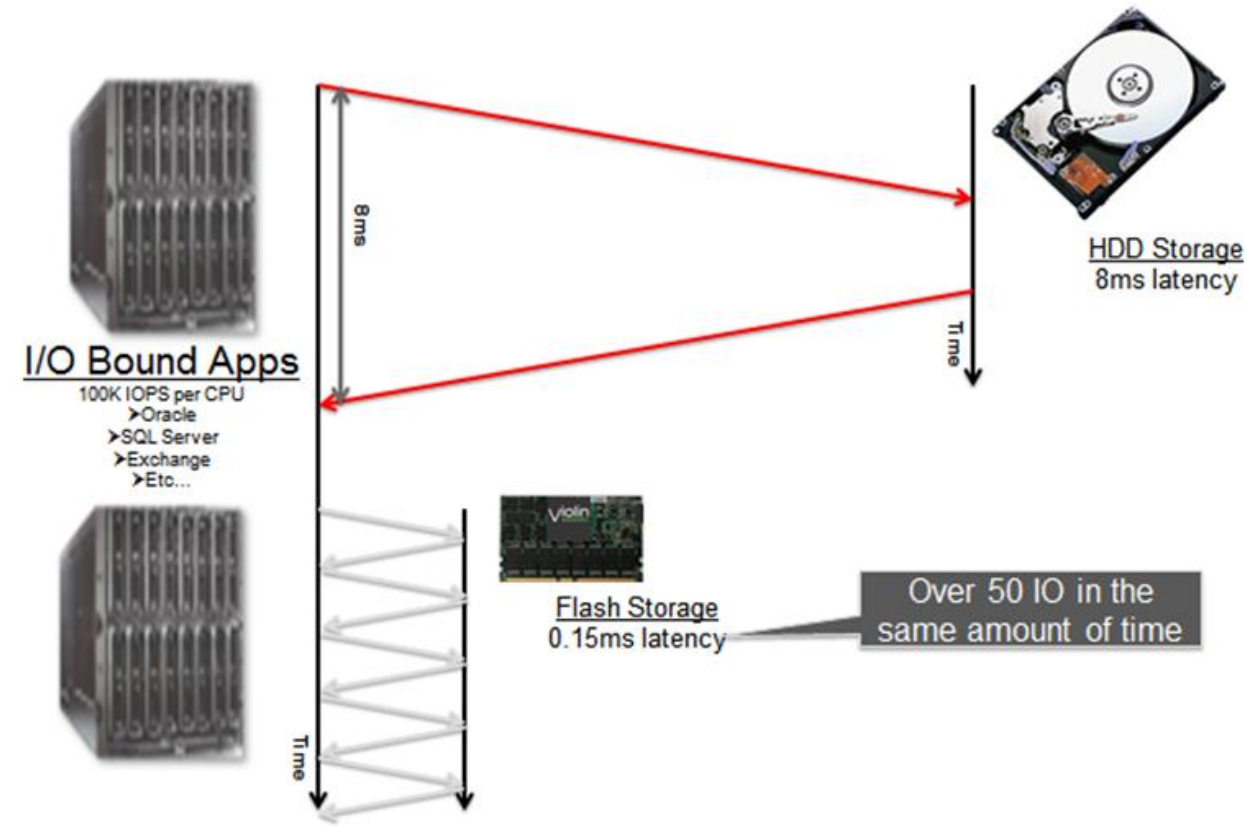


Diagram 2

Identifying the Bottleneck

CPU or IO Bound. What does SAR say?

Fortunately there is a wonderful utility that accompanies most UNIX and Linux systems called SAR that when executed with the –u parameter produces a very easy to read IO report. This utility runs in the background capturing several levels and types of data but for the purpose of this paper only the CPU characteristics of IO will be discussed. Here is a sample of a Linux system's SAR report for an hour long segment. This particular system is utilizing a large 3Par disk array.

```
Linux   2.6.18-128.el5  (ora04) "       11/14/2010"

12:00:01AM      CPU     %user   %nice   %system %iowait %steal  %idle
12:05:01AM      all     26.62   0       7.21    52.67   0       13.5
12:10:01AM      all     27.19   0       7.34    53.08   0       12.39
12:15:01AM      all     26.24   0       7.24    54.12   0       12.4
12:20:01AM      all     21.02   0       7.12    55.03   0       16.83
12:25:01AM      all     21.42   0       6.68    53.67   0       18.22
12:30:01AM      all     21.1    0       6.63    54.46   0       17.81
12:35:01AM      all     20.45   0       6.52    52.04   0       20.99
12:40:01AM      all     21.58   0       6.93    54.18   0       17.31
12:45:01AM      all     19.62   0       6.5     52.58   0       21.3
12:50:01AM      all     19.77   0       6.42    52.53   0       21.28
12:55:01AM      all     21.99   0       8.12    54.03   0       15.87
1:00:01 AM      all     20.87   0       6.97    52.01   0       20.16
```

This eight column report provides a very clear window into what the CPU are busy doing in each five minute increment of the report. For example, at 12:15:01 in the morning this particular system's CPU were 87.6% busy performing work (26.24 + 7.24 + 54.12). 26.24% of that work fell under %user which would be the application using the CPU such as an Oracle database executing a query. 7.24% of that 87.6% busy CPU was servicing the system calls and 54.12% of the time the CPU were considered busy they were waiting on IO. Let's restate that. Over half of the time the CPU appeared busy they were actually waiting on IO. If a top command were to be executed during this time

it would appear that the CPU were being heavily utilized at 87% leaving very little room for growth on this particular system without adding more CPU.

IO Indicators inside the AWR Report

The Oracle AWR report is an excellent tool to utilize when looking for key indicators of a database struggling with IO latency issues. These reports are very long and daunting, but by looking in a few key places it is usually relatively obvious if IO is an issue. Here are the steps to deciphering the AWR report looking for IO problems.

1. First of all always try to create a one hour long AWR report that spans a known window of significant load. A report spanning too long of a window runs the risk of averaging light IO times into the heavier ones thus skewing the averages low. Glance at the header paying attention to the version of the database and if it is a RAC environment or not. 10G and 11G reports have subtle little differences in structure but nothing that will get in the way of this exercise. If it is a RAC environment make sure to gather a report for each instance in the RAC for the exact same timeframe. There may be something occurring on one node and not the others.

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
MMDB01	2415615897	MMDB01	1	10.2.0.4.0	NO	ora13

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	19452	20-Oct-10 09:00:42	616	59.1
End Snap:	19453	20-Oct-10 10:00:44	615	59.2
Elapsed:		60.03 (mins)		
DB Time:		7,693.26 (mins)		

2. Next jump down to the Load Profile section. No need to spend a lot of time digging through all this excellent data. Remember, we're looking for physical IO which can be found in the two rows labeled Physical reads and Physical writes. Those numbers in the Per Second column can be looked at as the IOPS but the key to looking at this section is to make sure there IS physical IO going on and not all IO is happening in memory which would show up as Logical reads in the table. In my experience, significant IO numbers in this table are anything above 500 per second.

Load Profile

	Per Second	Per Transaction
Redo size:	2,963,178.53	1,017.42
Logical reads:	75,484.91	25.92
Block changes:	19,722.21	6.77
Physical reads:	4,799.55	1.65
Physical writes:	2,737.06	0.94
User calls:	4,635.09	1.59
Parses:	5,756.72	1.98
Hard parses:	0.00	0.00
Sorts:	2,229.37	0.77
Logons:	0.04	0.00
Executes:	9,019.64	3.10
Transactions:	2,912.45	

3. So we know what the environment is and that there is significant load on the database. Let's jump down to the Top 5 Timed Events section (Top 5 Timed Foreground Events in 11G). This is another section that is a quick glance indicator. The column Wait Class calls out exactly what type of wait events are causing the database the most problems during this AWR report window. If there are any IO type wait events, glance over to the Avg Wait(ms) column and take note of the value for each IO wait event. This is the latency the database is witnessing

for that specific wait event during this report time period. If these numbers are elevated above 5ms per wait event and are occurring often, the DB most likely has an IO issue.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file sequential read	17,292,809	358,703	21	77.7	User I/O
log file sync	10,485,848	90,996	9	19.7	Commit
CPU time		32,220		7.0	
db file parallel write	4,181,646	9,475	2	2.1	System I/O
log file parallel write	1,337,917	3,284	2	.7	System I/O

4. The last section of discussion in the report is the IO Stats portion. Paying particular attention to the Tablespace and File IO Stats reports (there are others in the 11G version), the Av Rd(ms) column is reporting the latency of each of the tablespaces and data files within the database. In the following example, the database is seeing a steady 20ms read latency throughout the database. Keep in mind flash averages .15 - .5 ms read latency.

Tablespace IO Stats

- ordered by IOs (Reads + Writes) desc

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
P1TBS	548,436	152	20.77	1.00	248,729	69	185	17.89
P14TBS	551,286	153	20.74	1.00	239,582	67	251	17.45
P5TBS	545,707	152	20.77	1.00	241,888	67	531	18.32
P10TBS	545,718	152	20.78	1.00	241,141	67	444	18.72
P9TBS	547,710	152	20.70	1.00	239,081	66	274	18.10

File IO Stats

- ordered by Tablespace, File

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
P0TBS	+DATAMMDB01/mmdb01/datafile/datatbs2.533.708266061	188,678	52	20.62	1.00	72,667	20	77	20.78
P0TBS	+DATAMMDB01/mmdb01/datafile/p0tbs.441.717171495	163,711	45	20.75	1.00	77,456	22	52	20.19
P0TBS	+DATAMMDB01/mmdb01/datafile/p0tbs.557.711215913	183,214	51	20.73	1.00	76,230	21	65	28.00
P10TBS	+DATAMMDB01/mmdb01/datafile/p10tbs.345.731971089	24,437	7	20.89	1.00	26,052	7	12	10.00
P10TBS	+DATAMMDB01/mmdb01/datafile/p10tbs.474.716437083	177,098	49	20.86	1.00	80,847	22	141	20.78

What Should these Reports Look Like with Flash?

Now let's take a look at the same reports for the same system after migrating its data onto flash storage arrays.

```
Linux   2.6.18-128.el5  (ora14) "       12/07/2010"

12:00:01AM      CPU     %user   %nice   %system %iowait %steal  %idle
12:10:01AM      all     18.48   0       5.55    9.14    0       66.84
12:20:01AM      all     21.09   0       6.6     9.56    0       62.75
12:30:01AM      all     20.94   0       6.37    12.16   0       60.54
12:40:01AM      all     16.71   0       5.55    8.82    0       68.92
12:50:01AM      all     17.32   0       5.66    10.47   0       66.56
1:00:01 AM      all     17.91   0       5.9     9.94    0       66.25
```

First the sar -u report. Notice by simply removing the %iowait for the CPU they instantly went from averaging less than 21% idle to over 60% idle. Over 40% of the CPU available to the sytem were freed up by eliminating the IO bottleneck. This allowed for a drastic reduction in Oracle database server hardware thus reducing the necessary licenses for the environment.

The key sections of the AWR report show just as drastic of a change for the good.

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
MMDB02	2111504056	MMDB02_STDBY	1	10.2.0.4.0	NO	ora14

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	19459	20-Oct-10 09:00:15	587	61.9
End Snap:	19460	20-Oct-10 10:00:16	585	62.1
Elapsed:		60.02 (mins)		
DB Time:		440.17 (mins)		

This is actually a Data Guard copy of the production database MMDB01 that was analyzed above.

Load Profile

	Per Second	Per Transaction
Redo size:	2,785,931.11	966.96
Logical reads:	77,542.73	26.91
Block changes:	19,583.60	6.80
Physical reads:	4,716.76	1.64
Physical writes:	2,790.56	0.97

With a very similar load profile.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		23,035		87.2	
log file sync	10,337,686	9,372	1	35.5	Commit
db file sequential read	16,980,465	9,124	1	34.5	User I/O
log file parallel write	5,403,199	2,543	0	9.6	System I/O
log file sequential read	11,899	155	13	.6	System I/O

But notice how the IO wait events have dropped to the bottom of the top 5 and are now only showing a single millisecond versus 21 milliseconds in the previous reports. Now the CPU are the main resources the database is waiting on. Perfect !!

Tablespace IO Stats

- ordered by IOs (Reads + Writes) desc

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
P33TBS	544,355	151	0.54	1.00	250,565	70	31	0.97
P62TBS	539,821	150	0.53	1.00	249,834	69	46	0.65
P61TBS	540,757	150	0.54	1.00	246,349	68	152	0.33
P39TBS	537,915	149	0.54	1.00	248,560	69	10	0.00
P40TBS	533,505	148	0.54	1.00	252,641	70	3	0.00

File IO Stats

- ordered by Tablespace, File

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
P32TBS	+DATAMMDB02/mmdb02/datafile/p32tbs.400.717902943	164,647	46	0.54	1.00	76,355	21	6	0.00
P32TBS	+DATAMMDB02/mmdb02/datafile/p32tbs.510.710643793	167,579	47	0.52	1.00	73,431	20	14	0.71
P32TBS	+DATAMMDB02/mmdb02/datafile/p32tbs.511.710644043	203,248	56	0.54	1.00	95,498	27	20	0.50
P33TBS	+DATAMMDB02/mmdb02/datafile/p33tbs.416.732796149	3,108	1	0.48	1.00	5,286	1	1	0.00
P33TBS	+DATAMMDB02/mmdb02/datafile/p33tbs.445.717197343	192,211	53	0.54	1.00	81,878	23	6	3.33

And the read latency has been reduced to sub millisecond.

Adopting a New Paradigm

Paradigm shift – Thomas Kuhn described this in 1962 as "a change in the basic assumptions within the ruling theory of a topic." He used the picture below to demonstrate the way in which a paradigm shift could cause one to see the same information in an entirely different way. If you focus on the slow HDD on the left you probably see a duck. If you shift your vision to the NAND flash on the right your paradigm may shift to a rabbit.

Systems have been developed with the latency of spinning hard drives since the beginning of time and the Oracle database is no exception. Since its earliest versions painstaking efforts have been made to ensure that the physical storage aspect of the database was designed out of the equation when accessing data. It is these design methodologies that have differentiated the Oracle database from its competitors when speaking of large enterprise level databases. First of all there is the method that Oracle utilizes memory on the database server to, in no better terms, cache data and perform all the behind the scenes activities it needs to accomplish extremely fast. I'm speaking of course of the SGA and all the memory pools of the Oracle instance. Oracle very ingeniously uses the DRAM of the server to mask the painfully slow HDD subsystem by bringing data into memory and caching it as effectively as possible to reduce the level of physical IO to as little as possible. It has been a tuning joke around the water cooler forever that the fastest database is the one that fits completely into memory. There are countless other hoops the DBA has had to jump through over the years to tune around the slow spinning drive. Physical database layouts, complex partitioning schemes, extreme SQL tuning activities, materialized views with magical query rewrite, and endless indexing to name a few. Enter the all NAND flash Oracle database. Let's talk about the paradigm of database storage approaching the speed of DRAM memory. No longer will it be necessary to make sure the database data files are evenly spread out across physical drives either by complex RAID based disk arrays or by creating the files across countless mount points. When millions of IOPS are suddenly available to the database multi-million record tables begin to perform full table scans in seconds versus minutes. The most expensive and valuable resource within the server, the CPU, is released from waiting for an IO request to return and is able to perform those millions of operations per second they are rated at and purchased for. That multi-terabyte database that was growing at an un-scalable rate once again becomes manageable. It is safe to say that a majority of the day to day tuning activities of the DBA become a thing of the past.

Forcing Oracle to Parallelize to Drive IOPS

Parallel processing within Oracle has always been a two headed monster. When implemented correctly it allows for today's multi-core servers to blaze through large table scans turning hour long reports into minutes, sometimes seconds. However, if too many child processes are turned loose on a query and HDD latency gets in the way, performance can plummet often bringing a server to a screeching halt. This is caused by as an avalanche effect taking place when dozens of child processes become stalled waiting on other child processes to complete which are waiting on IO causing the dreaded PX wait events. Oracle 11g has come a long ways with phenomenal algorithms in controlling the cascading deadlock effect of over parallelizing a system. When utilizing auto degrees of parallelism in conjunction with setting the PARALLEL_ADAPTIVE_MULTI_USER parameter to TRUE and PARALLEL_MAX_SERVERS and PARALLEL_THREADS_PER_CPU to appropriate values, it is very difficult to over burden a powerful server. But this safety has come at a price to unbridled performance because in doing so it has also put a form of governor on the database.

Now we enter into the all flash database paradigm once again. Latency is practically nonexistent and the database has the ability to perform millions of IOPS. It is imperative that the CPU of the server are turned loose to do as much

work as they possible can and parallelism is a major engine for driving the load. However, by design the database is throttled back because of the latency of HDD systems. In order to get past this the database must be tricked. Here is how this is accomplished. When the DBMS_RESOURCE_MANAGER.CALIBRATE_IO internal packaged procedure is executed a very key value is set within one of the interior $ tables that directly effects how the optimizer will choose to spawn parallel processes. This value is the per process maximum throughput (MAX_PMBPS) value of the resource_io_calibrate$ table. By default this value populates with the actual discovered throughput of the system during the calibrate_io procedure execution. When this value is large enough the optimizer determines that the need to spawn parallel processes is not as necessary because of the old way of thinking with HDD storage. So by manually altering the data in this table, auto degrees of parallelism can often be enhanced for an all flash database.

```
SQL> desc resource_io_calibrate$;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 START_TIME                                         TIMESTAMP(6)
 END_TIME                                           TIMESTAMP(6)
 MAX_IOPS                                           NUMBER
 MAX_MBPS                                           NUMBER
 MAX_PMBPS                                          NUMBER
 LATENCY                                            NUMBER
 NUM_DISKS                                          NUMBER
```

```
SQL> delete from resource_io_calibrate$;
SQL> insert into resource_io_calibrate$
        values(current_timestamp, current_timestamp, 0, 0, 200, 0, 0);
SQL> commit;
```

This tells the database a single process can drive at most 200 MB/s from the storage system. If you want more parallelism, tell Oracle each process drives less IO, and the database suggests creating more processes to go after the data. In a recent testing effort, this value was determined to be at a level of 330MB/s which was hindering parallelism efforts in the test database. Even reducing the value to the 200 MB/s setting drove no more parallelism in our test. The value was dropped to 50 MB/s and finally parallelism picked itself up off the floor, suggesting a degree of 8. When the value was reduced to 5 MB/s, suddenly Oracle wanted to throw all 80 cores at the test query. Success!!. We fell back to 50 MB/s, and ran the tests again. We hit a record time of 49 seconds on a report that took 21 minutes on spinning media; 28 minutes went to 59 seconds for another report; 22 minutes went to 44 seconds on yet another. And lastly a report that wasn't finishing in 5 hours completed in 28 minutes. A special thanks to Nathan Fuzi of Violin Memory for supplying this test data.

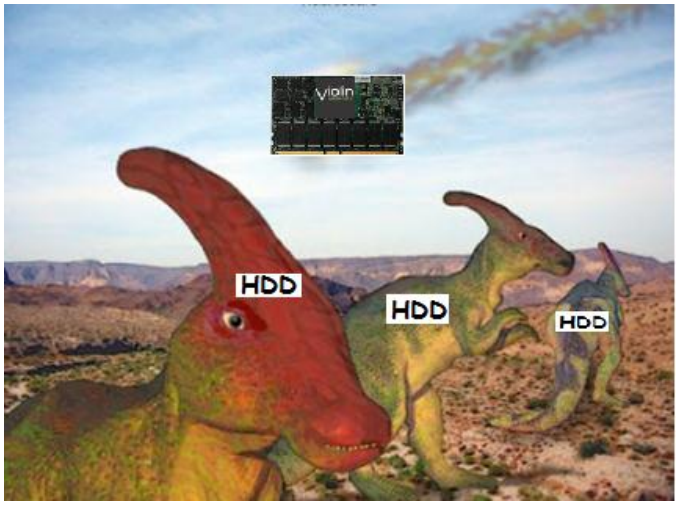
Oracle Installation – Things to be Aware Of

NAND flash is by nature referred to as native 4k block storage meaning that it functions optimally when all IO written to and read from the flash is in 4k block iterative segments. We all know that the Oracle database is natively 4k aligned data having default database block sizes being 4k, 8k, 16k, and 32k block sizes, all iterations of 4k. However, it was not until NAND flash was housing not just data files and indexes but the entire database that it was not discovered that the online redo logs of a database are NOT set to the db_block_size of a database but are aligned to the block size of the operating system. In all UNIX and Linux environments this block size is 512 bytes which in turn creates unaligned IO to a NAND flash device drastically reducing the performance of the storage. This issue has been addressed and corrected in databases of release 11.2.0.2 and higher with the implementation of the BLOCKSIZE parameter in the "create database" and "alter database add logfile" DDL statements. Here is an example.

```
CREATE DATABASE "POCDB"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 1000
```

```
DATAFILE
        SIZE 700M AUTOEXTEND ON NEXT  10240K MAXSIZE UNLIMITED
        EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE
        SIZE 600M AUTOEXTEND ON NEXT  10240K MAXSIZE UNLIMITED
BIGFILE DEFAULT TEMPORARY TABLESPACE TEMP
        TEMPFILE SIZE 20G AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
SMALLFILE UNDO TABLESPACE "UNDOTBS1" DATAFILE
        SIZE 200M AUTOEXTEND ON NEXT  5120K MAXSIZE UNLIMITED
CHARACTER SET WE8MSWIN1252
NATIONAL CHARACTER SET AL16UTF16
LOGFILE
GROUP 1  SIZE 10240M blocksize 4k,
GROUP 2  SIZE 10240M blocksize 4k,
GROUP 3  SIZE 10240M blocksize 4k,
GROUP 4  SIZE 10240M blocksize 4k,
GROUP 5  SIZE 10240M blocksize 4k,
GROUP 6  SIZE 10240M blocksize 4k
USER SYS IDENTIFIED BY oracle USER SYSTEM IDENTIFIED BY oracle;
```

In conclusion, the age of NAND flash as the primary storage media of the data center is rapidly approaching. With its characteristics of being hundreds of times faster, denser, and requiring far less power than the racks and racks of spinning drives in use today, many companies are already pursuing the "all flash data center" mindset. Like the CD was to the cassette tape and the IPOD has been to the CD, NAND flash will be to the spinning hard drive one day driving it into extinction.



References

Thomas Kuhn *The Structure of Scientific Revolutions* (1962)

Wikipedia - http://en.wikipedia.org/wiki/Flash_memory

Nathan Fuzi, Violin Memory, How to trick the optimizer into higher DOP.