Optimizing OLTP Oracle Database Performance using Dell Express Flash PCIe SSDs

Author

Kai Yu

Global Solutions Engineering

This document is for informational purposes only and may contain typographical errors and technical inaccuracies. The content is provided as is, without express or implied warranties of any kind.

July 2012| Rev 1.0

Contents

Figures

Executive summary

The Dell PowerEdge Express Flash PCIe SSD is an enterprise class high performance storage device. It is built with SLC NAND flash and designed for applications that require low latency and high IOPs (IO Per Second) operation. OLTP database workloads which require very low storage IO latency with many small random read/write IO operations are the ideal cases to use Dell Flash PCIe SSDs to improve the performance. To understand how much PCIe SSDs can really improve OLTP database performance, several performance studies have been conducted on a single node Oracle 11g R2 database as well as a two node 11gR2 Oracle Real Application clusters (RAC) database running on Dell PowerEdge R720 servers with Oracle Enterprise Linux 6.2 platform. This article is to show these case studies and some preliminary performance comparison results.

Introduction to Dell Express Flash PCIe SSDs

The Dell PowerEdge Express Flash PCIe SSD is built with SLC NAND flash and can be used as an internal storage of Dell PowerEdge servers. This not only removes the performance bottleneck posted by the mechanical parts of conventional HDDs, also improves the storage IO performance by eliminating the latency and performance bottleneck between the server and the external storage. For example, a single Express Flash PCIe SSD drive response time can result in up to 26 ms, up to 1/4th lower than 4 SAS SSD drives and up to 1/10th the latency of 16 traditional HDDs. This makes PCIe SSDs the ideal storage for the applications that require low latency and high IOP (IO Per Second) operation. Figure 1 shows that the Dell 12 Generation PowerEdge server R720 can have up to 4 high performance internal PCIe SSDs drives in additional to the regular SAS or SATA drives which are usually used for OS and local file systems. The four PCE-e SSDs drives fit into driver carries and are front accessible and can be used for improving the database performance.

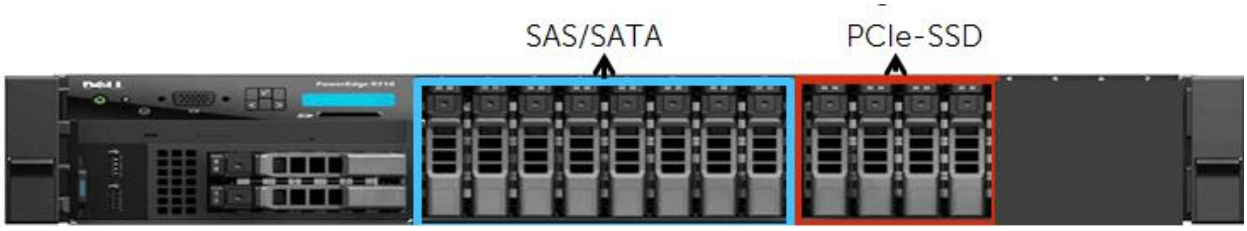

Figure 1: PCIe SSDs on Dell PowerEdge Server R720

To understand how PCIe SSDs improve Oracle database performance, we have established three use cases of PCIe SSDs for an Oracle database:

1. **Use PCIe SSDs to store the entire application database schema**. In this case, PCIe SSDs are used as the primary storage for all the application data. As the total capacity of four 350GB PCIe SSDs is 700GB for the Raid 10 disk array configuration, this user case only applies to a small to middle size of user database with the size of all tables plus indexes together equal to or less than the 700GB maximal capacity. Since the PCI-S SSDs are only accessible to the local server, this user case is limited to a single node Oracle database configuration.

2. **Use PCIe SSDs to store a portion of application database schema.** In this case, PCIe SSDs are used a part of tiered storage combining with conventional hard disk drives (HDDs). Only those most active data are stored in PCIe SSDs while the rest of the database is stored in the conventional hard disk drives. The size of the application schema is not limited by the maximal capacity of the PCIe SSDs. However the bigger the application schema is, the smaller portion of the user schema can be stored in the PCIe SSDs. Since the PCI-S SSDs are only accessible to local server, this user case is also limited to a single node Oracle database configuration.
3. **Use PCIe SSDs as Oracle database smart flash cache.** Oracle smart cache is an extension of Oracle database cache. Using PCIe SSDs as Oracle database smart flash cache improves the database performance by allowing some data read from the flash cache instead from disk storage**.** This case is suitable for multi-node Real Application Clusters database as the smart flash cache is only accessed by the local database node. And in this case there is no limitation on the size of the application database.

We used the Quest Benchmark software to create the TPC-C like workloads on the database and measure the performance characteristics such as TPS (Transaction Per Second) and the Average Response Time. The goal of this study was to measure the performance gain of using PCIe SSDs on these three cases. In order to quantify the performance gain, on each of these cases, we first measured the database performance without using PCIe SSD as the baseline, then compare the database performance of using PCIe SSDs with the baseline:

Conventional HDD Storage Configuration

A hard disk based conventional storage was configured to establish the performance baseline to compare the performance improvement of using PCIe SSDs. In this paper, a Dell MD3220 SAS storage with 24 hard disk drives (HDDS) was used. The configuration of this storage is shown in figure 2:

1) Five disk groups: DataG1, DataG2, DataG3, DataG4 and FRAG were created with four physical disks each in Raid 10. Five disk volumes Data1, data2, Data3, Data and FRA were created in these five disk groups respectively. Because of disk mirroring, IO operations on each of these disk volumes are stripped on 2 effective disk spindles
2) Two Raid 1 disk group RedoG1 and RedoG2 were created on two physical disks each. Two disk volumes Redo1 and Redo2 were created with these two disk groups respectively. Because of disk mirroring, each of volume only has 1 effective disk spindle.

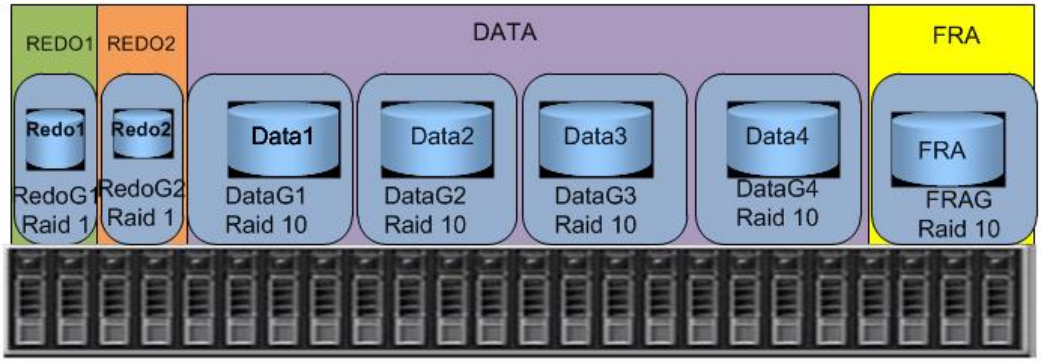

Figure 2: Configuration of Dell MD3220 storage

These disk volumes were used to create the following ASM diskgroups:

1) ASM diskgroup DATA was stripped on four volumes: Data1, Data2, data3, Data4. It has 4 X 2 =8 effective spindles. DATA diskgroup were used to store the tablespace for tables and indexes
2) FRA diskgroup had only disk volume FRA with 2 spindles. FRA diskgroup were used for fresh recovery area and archived logs
3) REDO1 diskgroup had disk volume Redo1 with 1 spindle. This diskgroup was used for online redo logs.
4) REDO2 diskgroup had disk volume Redo1 with 1 spindle. This diskgroup were used for online redo logs.

Use Case 1: use PCIe SSDs to store the entire application schema

In this use case, four PCIe SSDs were used as the primary storage all the application database objects. This configuration only works for a single node database configuration and not for multi-node Oracle Real application Clusters database configuration as the PCIe SSDs drives are the internal storage that cannot be shared by another server. The test environment is showed in figure 3:

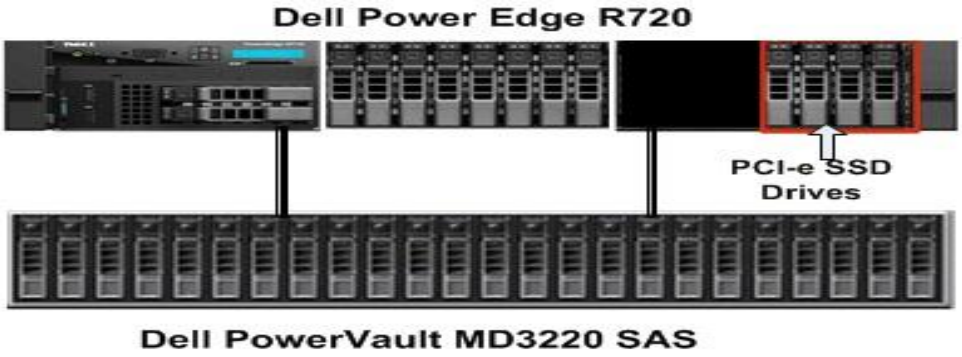

Figure 3: Single Node database configuration

In addition to the DATA diskgroup on the MD3220 storage, four internal PCIe SSD drives PCIe1, PCIe2, PCIe3, PCIe4 are plugged-in to the server as shown in figure3. Before the server can see the PCIe SSD drives, we loaded the PCIE SSD SLA Drivers. The current latest version of the driver for Oracle Enterprise Linux 6.2 (Red Hat Enterprise Linux 6.2 compatible) is 1NY9K (file name: Express_Flash_PCIe-SSD_DRVR_RHEL6.2_1NY9K_A00_1.2.32-1.tar.gz).

To load this drive, first unzip the downloaded file on the server:

 # gunzip Express_Flash_PCIe-SSD_DRVR_RHEL6.2_1NY9K_A00_1.2.32-1.tar.gz
 # tar xvf Express_Flash_PCIe-SSD_DRVR_RHEL6.2_1NY9K_A00_1.2.32-1.tar

kmod-mtip32xx-1.2.32-1.el6.x86_64_rhel6u2.rpm is the driver rpm from this tar file.

Load this driver rpm: *# rpm -ivh kmod-mtip32xx-1.2.32-1.el6.x86_64_rhel6u2.rpm*
Then you will see the PCIe SSD devices:

```
# more /proc/partitions
major minor  #blocks  name
```

252 0 341873784 rssda →PCIe1
252 256 341873784 rssdb → PCIe2
252 256 341873784 rssdc → PCIe3
252 256 341873784 rssdd → PCIe4

Partition these four devices with fdisk utility to create four partitions:
/dev/rssda1 , /dev/rssdb1, /dev/rssdc1 and /dev/rssdd1
Then create the Raid 1 software array /dev/md0 by mirroring /dev/rssda1 and /dev/rssdb1 and the Raid 1 software array /dev/md1 by mirroring /dev/rssdc1 and /dev/rssdd1:

mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/rssda1 /dev/rssdb1
mdadm --create --verbose /dev/md1 --level=1 --raid-devices=2 /dev/rssdc1 /dev/rssdd1

By default, both devices /dev/md0 and /dev/md1 are owned by root. For Oracle Enterprise Linux 6.2, we created a udev rules file 97-oracle-asmdevices.rules in /etc/udev/rules.d to set the proper ownership and permissions for /dev/md0 and /dev/md1 devices so that Oracle ASM instance can create ASM diskgroup on them. This udev rules file has the two lines:

KERNEL=="md0",OWNER="grid", GROUP="asmadmin", MODE="0775"
KERNEL=="md1",OWNER="grid", GROUP="asmadmin", MODE="0775"

The Linux user 'grid' with the Linux group 'asmadmin' is the owner of the Oracle ASM instance. All the devices for Oracle ASM diskgroups need to be owned by this user and the permission needs to be set "0775"

ASM diskgroup PCIED was built by stripping on two disks array MD01 and MD02 using the external redundancy setting (as shown in figure 4). Since both md0 and md1 are 350GB. PCIED diskgroup has 700GB capacity and is used as the primary storage to store database objects such as tables and indexes.

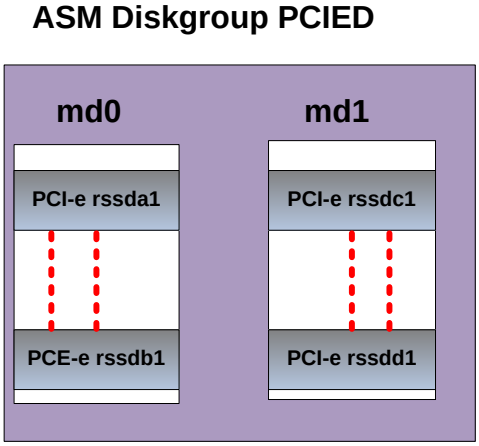

Figure 4: ASM Diskgroup built on PCIe SSD Drives

As a performance comparison, we ran the TPC-C like performance tests on a database with 500GB data (300GB data and 200GB indexes) with two different storage configurations:

1) Baseline of HDD only configuration. All the tables and indexes of the test schema are stored in DATA diskgroup in the hard disk drives of the MD storage
2) PCIE configuration: all the tables and indexes of the test schema were stored in PCIED diskgroup in the PCIe SSD drives

We used the Transaction Per Second (TPS) of baseline (HDDS only configuration) as the performance baseline(100%) and calculated the performance improvement percentage ratio :

TPS percentage = TPS from the test case/TPs of baseline.

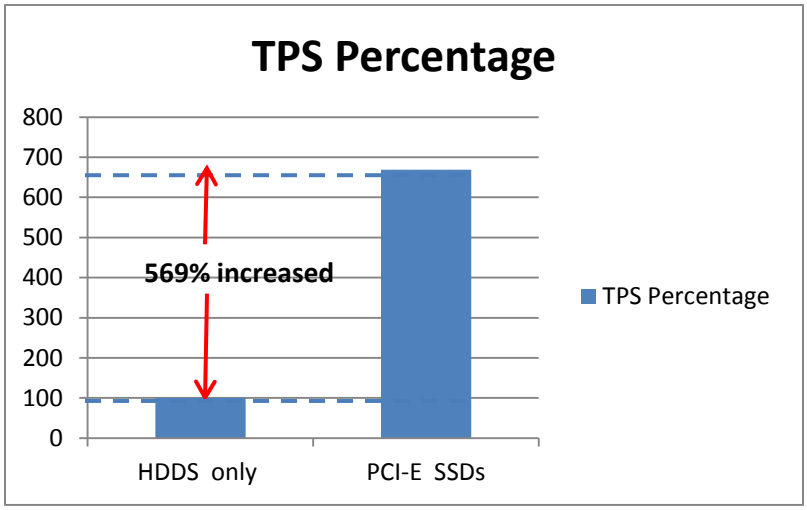

Figure 5: TPS comparison of HDD only configuration and PCIE configuration

The comparison of average response time vs number of user loads in two configurations is showed in figure 6:

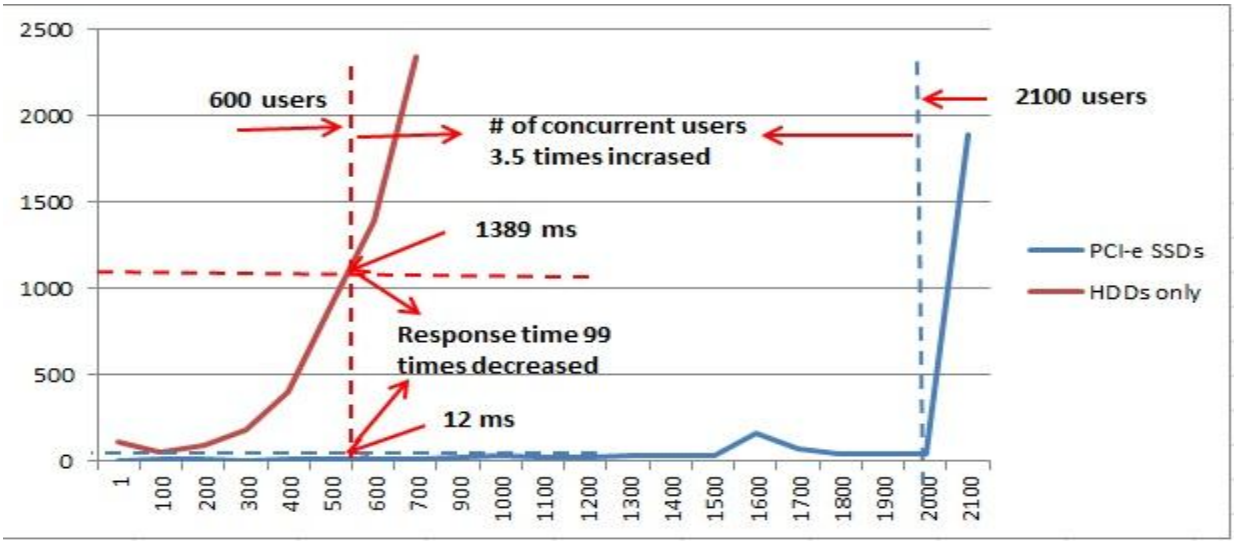

Figure 6: Response Time Comparison

As shown in Figure 6 above, we used 1 second = 1000ms (milliseconds) response time as the valid benchmark threshold. With this threshold, the HDDS only configuration carried 600 users, while at the same 600 user load the response time of the PCIE configuration is 12ms. The response time for PCIE SSD is 99 times less than the one for the HDDs configuration. While for the same 1000 ms (=1 second) mark, the PCIE SSDs configuration can handle 2100 users while the HDDs can only handle 600 users. It is 3.5 times increase of the user loads.

Use case 2: use PCIe SSDs to store a portion of an application schema

In user case 1, we compared the database performance differences between two extreme scenarios: all application database objects stored in HDDs vs all database objects stored in PCIe SSDs. In real life, it is very common to have two tiers of storage: HDDs for less active database objects (cold data) and PCIe SSDs for more active database objects(hot data). Using the same test environment configuration as use case 1, use case 2 was designed to measure the database performance differences while relocating some active database objects to PCEe SSDs. This test case consists of four configurations for a 800 GB database (500GB data plus 300 GB indexes).

1) Config1: all the database objects stored in HDDs. This is the baseline configuration
2) Config2: all indexes stored in PCIe SSD Drives and the rest stored in HDDS
3) Config3: all indexes plus one active table (c_stock table) stored in PCIe SSD drives while the rest stored in HDDS
4) Config4: all indexes plus four active tables stored in PCIe SSD Drives while the rest stored in HDDS

As shown in figure 7, by comparing to the baseline configuration TPS(100%) where all database objects are stored in HDDs,

1) Config2 (storing all indexes to PCI-SSD) increases TPS by 14% from the baseline TPS.
2) Config3 (storing all indexes and one most active table) increased TPS by 190% from the baseline TPS
3) Config4 (storing all indexes and four most active table) increases TPS by 310% from the baseline TPS

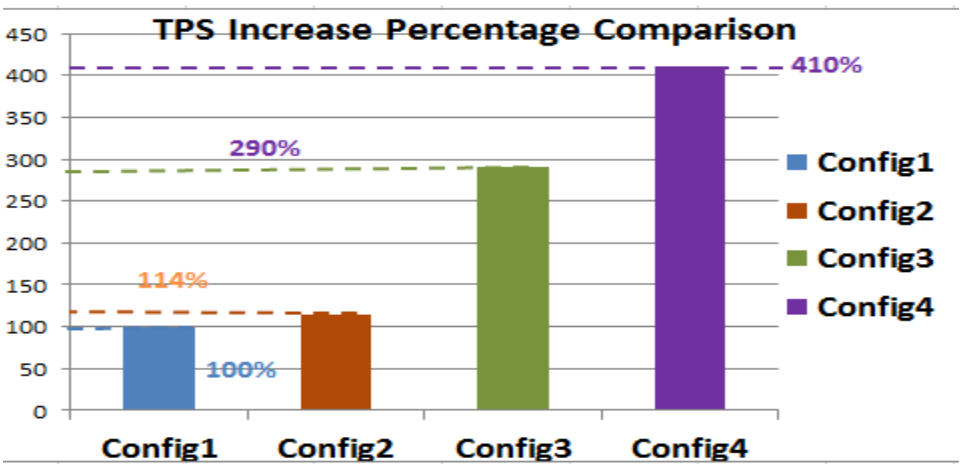

Figure 7. TPS Comparison of the four Configurations

Figure 8 shows the response time comparison of these four configurations. We used 1 second (1000 ms) response time as the valid benchmark threshold. Config1, the baseline configuration HDDS only configuration carried 800 users , while at the same 800 user loads, the response times of config2, config3 and config4 were 587 ms, 50ms, and 63 ms respectively. They were 41.3% less, 20 times less and 16.6 times less than the baseline response time. For the same 1000 milliseconds (=1 second) mark, config2, config3 and config4 can handle 900 users, 2000 users and 2400 users respectively. These were 12.5 % more, 150% more and 200% more than the baseline 800 users.

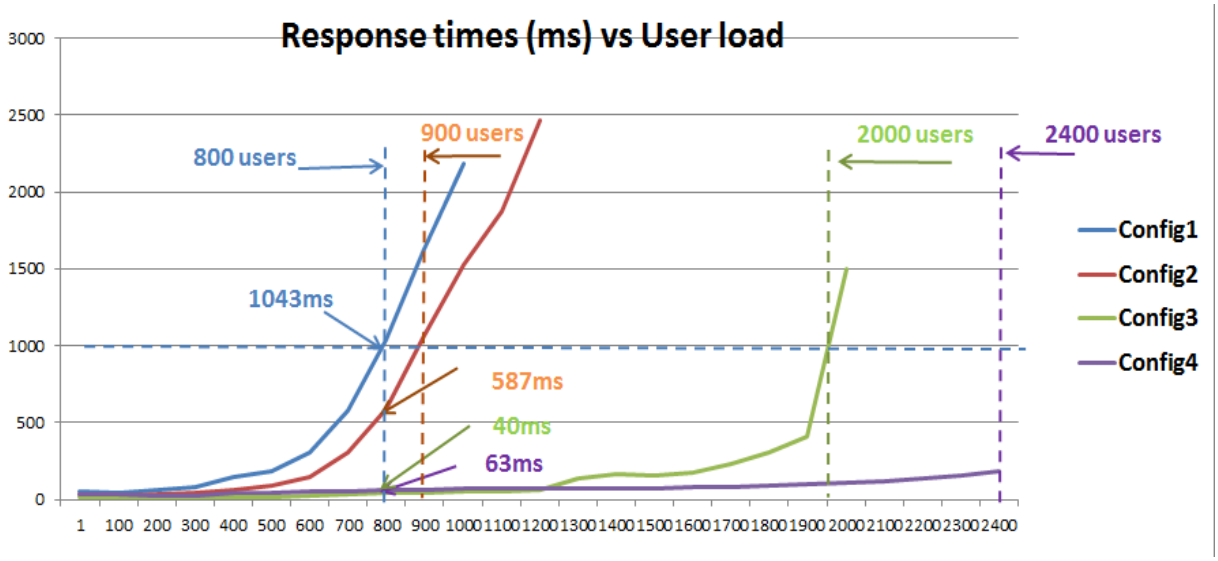

Figure 8: Response Time Comparison of the Four Configurations

Use Case 3: use PCIe SSDs as Oracle Database Smart Flash Cache

Oracle 11gR2 database introduced the new smart flash cache feature. This feature allows creating a transparent extension of Oracle database buffer cache using SSD (Solid State Device). The SSD is used as a level 1 cache to the level 1 cache of the database buffer cache.

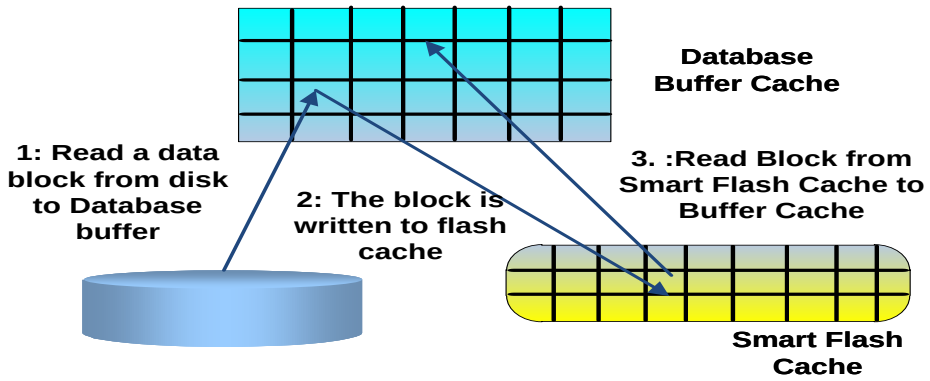

Figure 9: How the Oracle Database smart flash cache works

Figure 9 shows how the Oracle smart flash cache works: Initially a data block is read from the disk to the database buffer. When this data block is aged out from the database buffer, it is written to the flash cache. Next time when this block is needed, it will be read from the flash cache instead from the disk. Since the flash cache is based on SSD technology, the read operation from the flash cache is much

faster than the read from the disk. This feature can significantly improve the overall database IO performance.

This new feature also allows us to use leverage PCIe SSDs on a multi-node Oracle Real Application Clusters (RAC) database environment as this flash cache is local to each node and is not shared by other node like the primary database storage. Figure 10 shows such a configuration. According to the recommendation [1] from Oracle, the size of the flash cache should be 2-10 times of the database SGA_target size. In this test, the SGA_traget size was set 8GB, the flash_cache size is set between 20GB to 80GB. Therefore we only need one PCIe SSD drive on each database server as shown in Figure 9. We created a 500 GB test database (300GB data and 200GB indexes OLTP database) in DATA ASM diskgroup on MD3220 shared storage. Notice that we only use one PCIe SSD per server without any disk Raid configuration as the SSD does not store the data permanently.

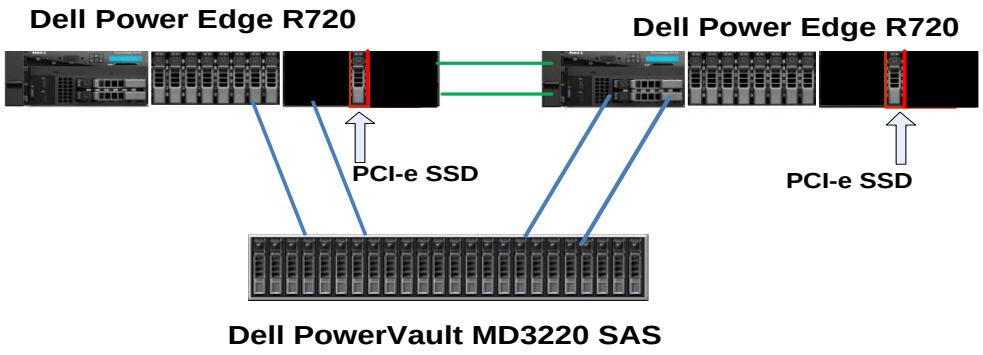

Figure 10 : Use PCE-e SSDs as flash cache in RAC database

To configure a PCE-e SSD drive as the flash cache, we first set the following udev rule to setup the proper ownership and permissions of this PCIe SSD device on all the RAC nodes:

KERNEL=="rssda1",OWNER="oracle", GROUP="dba", MODE="0775".

Notice that here the owner here is oracle user, the owner of the Oracle database. When the PCIe SSD device was used for build ASM diskgroup, the owner was set the grid user, the owner of the ASM instance.

It is required to enable the smart flash feature on all the nodes together. After executing the following SQL statements, restart the database on all the RAC nodes:

alter system set db_flash_cache_file='/dev/rssda1' sid='' scope=spfile*

*alter system set db_flash_*cache*_size=20G sid='*' scope=spfile;*

Due to an Oracle bug with Oracle database 11.2.0.3 with Oracle Enterprise Linux 6(Bug 12949806 - FLASH CACHE CHECK IS AGAINST ENTERPRISE-RELEASE), you will experience the following error when you try to start the database with the flash cache settings mentioned above:

SQL> startup
ORA-00439: feature not enabled: Server Flash Cache

The solution to this issue is to apply Patch 12949806. See reference [2] for details.

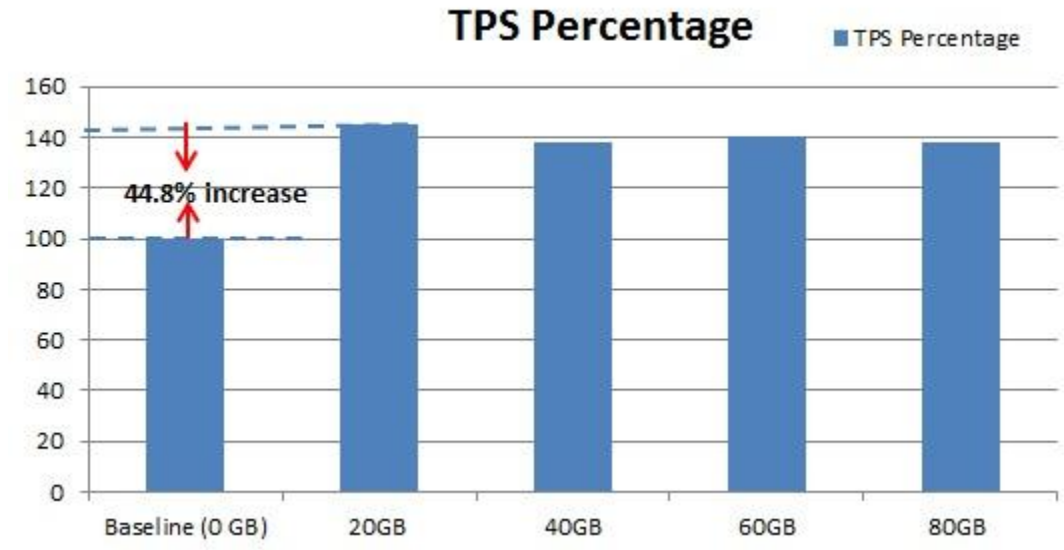

Figure 11: TPS Comparison of the four different flash cache size setting:

Figure 11 shows the 5.8 time reduction of the average query time by using PCE-e SSDs as flash cache compared to the baseline.

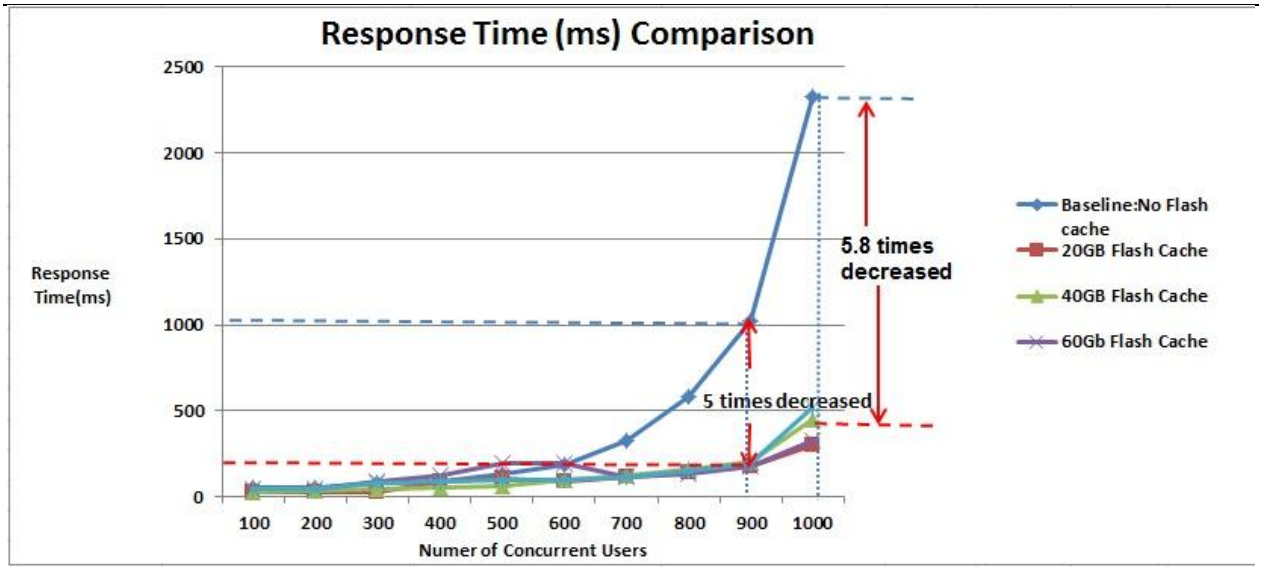

Figure 12: Response Time Comparison of four different flash cache size settings

These performance improvements are contributed by the significant data block reads from the PCIe SSD based database flash cache instead from physical disks. The following query shows the information about data block reads from buffer cache (consistent gets), flash cache (physical read flash cache hits) and physical disks (physical reads) during the test. It also shows the number of flash cache inserts

which stands for times of inserting the data aged out from the buffer cache into the flash cache for future queries.

```
SELECT name, value FROM v$sysstat
WHERE name IN ('physical read flash cache hits', 'physical reads', 'consistent gets', 'db block gets', 'flash cache inserts');

NAME                                                             VALUE
---------------------------------------------------------------- ----------
db block gets                                                    10912661
consistent gets                                                  194736483
physical reads                                                    4514407
physical read flash cache hits                                   2047180
flash cache inserts                                              2464691
```

However both figure 11 and figure 12 indicated that the increasing flash cache size from 20GB to 80GB didn't translate the further performance improvement. We also observed the different database wait pattern when the database reached the top TPS in the tests with flash cache. Figure 13 showed this wait event called 'Configuration' (in brown color) which we didn't see in the previous no flash cache tests.

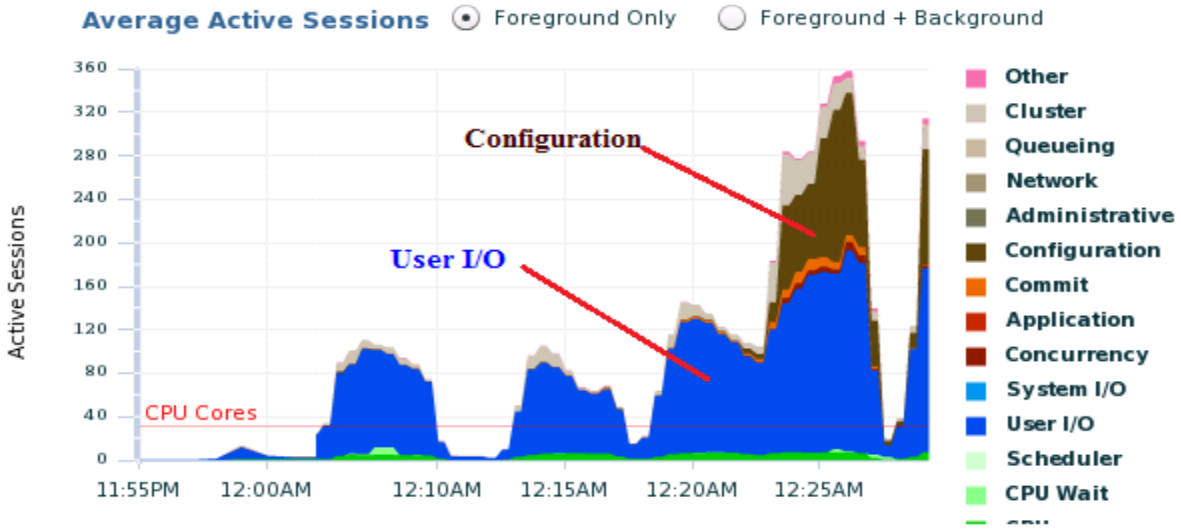

Figure 13: Database Wait Event Observed in flash cache test

We traced the 'configuration wait event' and found more details about this 'configuration' wait event in figure 14:

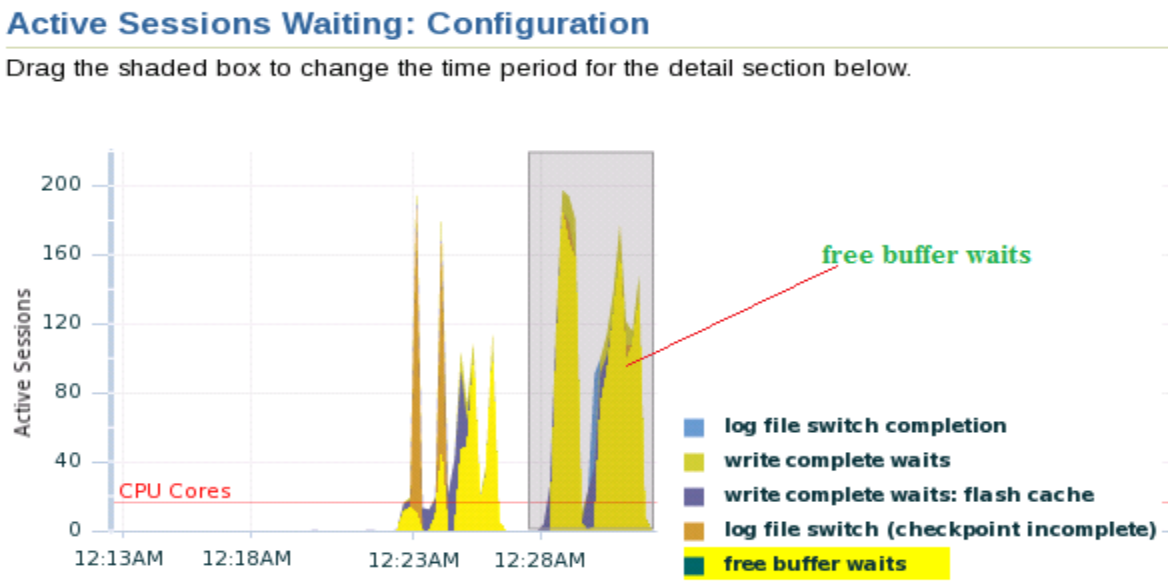

Figure 14: New wait event 'free buffer waits'

This wait event is caused by the difference between the fast data read from the smart flash cache to the buffer cache and the slow process of writing dirty data out from the buffer cache. As shown in figure 14, if there is not free slot in buffer cache that can used to store the block read from smart cache Oracle has to free a slot first. If the slot has the dirty block, this dirty block needs to be written to disk as a part of the process to free the slot. Since it is much slower to write the dirty block data to disk storage (operation B in figure 15) than to read the data from the flash cache (operation A in figure 15), very often the process to read data from the flash cache to the buffer cache has to wait for a free slot in the buffer cache. That is the 'free buffer waits' event shown figure 14.

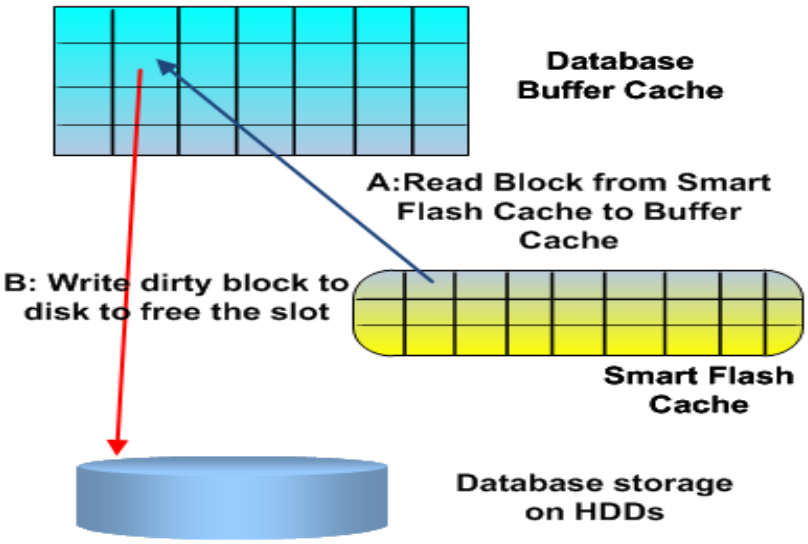

Figure 15: "Free buffer waits event" delays reading blocks from flash cache

So whenever this wait occurs, the advantage of fast read from smart flash cache really doesn't bring the further database performance improvement. When we increase the flash cache size, there may be more frequent reads from the flash cache, therefore this wait event may be more frequently occurred, and potentially this will slow down the data block read and slow down the overall database performance. That is why we didn't see the additional significant database performance improvement by increasing the size of the flash cache as indicated in figure 11, 12.

Summary

In this article we have explored three user cases of Dell Express flash PCIe for Oracle database:
1) Use PCIe SSDs to store the entire application database schema;
2) Use PCIe SSDs to store portion of application database schema.
3) Use PCIe SSDs as Oracle database smart flash cache

By storing the entire application database on PCIe SSDs, use case 1 has the best performance improvement: 569% TPS increase and 99 times response time reduction. However in the case, the size of the application database objects is limited by the 700GB total maximal capacity of PCE-e SSDs and it only works for a single node database.
By storing some portion of the application database on PCIe SSDs, User case 2 can have some different level of performance gains: 14% to 410% TSP increases and 41.3% to 20 times query response time reductions. Although the size of the application database is not limited by the maximal capacity of the PCIe SSDs, the bigger the application database is, the smaller portion of the user schema can be stored in the PCIe SSDs, therefor the less the performance improvement will have. And this case also only works for a single node database.
By using PCIe SSD as the Oracle smart flash cache, user case 3 achieved 44.5% TPS increase and 5.8 times response time reduction. In this use case the size of the application database is not limited by the maximal capacity of the PCI-D SSD and only one PCIe SSD is needed per server. And this is the only option that works with multi-node RAC database.
By comparing these three options, it is advised that we select an option that fit the application environment most by considering the factors such as the size of database , architecture of the database (single node or multi-node RAC database) and the most importantly the performance requirement of database, especially whether or not the storage IO is the major performance bottleneck .

References:

[1] Oracle Database Administrator's Guild 11gRelease 2 (11.2) Part number E25494-02

[2] Oracle metalink note [ID 1473149.1] ORA-00439 during Startup of Database with Oracle Flash Cache