DELL EMC VMAX ALL FLASH STORAGE FOR MISSION-CRITICAL ORACLE DATABASES

VMAX® Engineering White Paper

ABSTRACT

VMAX® All Flash array is a system designed and optimized for high-performance, while providing the ease-of-use, reliability, availability, security, and versatility of VMAX data services. This white paper explains and demonstrates why a VMAX All Flash array is an excellent platform for Oracle mission-critical databases.

H14557.2

May 2017
TABLE OF CONTENTS

EXECUTIVE SUMMARY ........................................................................................................ 5
AUDIENCE.......................................................................................................................... 5

INTRODUCTION TO VMAX ALL FLASH .............................................................................. 5
Overview ............................................................................................................................ 5
Storage design .................................................................................................................... 6
Adaptive Compression Engine (ACE) .................................................................................. 7

ORACLE AND VMAX ALL FLASH BEST PRACTICES ............................................................ 9
Host and storage connectivity ............................................................................................ 9
Number and size of host devices ....................................................................................... 10
Consistent device names across hosts for RAC ................................................................ 11
  Linux Device Mapper example ..................................................................................... 11
  PowerPath example ....................................................................................................... 11
  VMware native multipathing example .......................................................................... 12
Device permissions .......................................................................................................... 12
  Device Mapper Oracle user permissions example ....................................................... 12
  PowerPath Oracle user permission example ................................................................ 12
Partition alignment ......................................................................................................... 12
Oracle ASM best practices ............................................................................................. 13
  ASM disk groups .......................................................................................................... 13
  ASM striping .................................................................................................................. 14
VMAX masking views ...................................................................................................... 14
4 KB redo log sector size ................................................................................................. 16
Kernel I/O scheduler ........................................................................................................ 16
VMAX compression and Oracle Advanced Compression .................................................. 16
  Key points regarding VMAX Adaptive Compression Engine (ACE):......................... 16
  Key points regarding Oracle Advanced Row Compression ......................................... 17
  Additional considerations ............................................................................................ 18

TEST ENVIRONMENT ......................................................................................................... 19
Hardware and software configuration .............................................................................. 19
Tested workloads .............................................................................................................. 19

TEST CASES ........................................................................................................................ 20
Test 1: OLTP tests with negligible VMAX cache read-hit benefits .................................... 20
  Test Motivation ............................................................................................................. 20
Appendix II

Test 2: DSS test with focus on sequential read bandwidth

Test motivation

Test configuration

Test results

Test conclusion

Test 3: In-Memory OLTP tests (all reads are satisfied from database cache)

Test motivation

Test configuration

Test results

Test conclusion

Test 4: DSS test with focus on sequential read bandwidth

Test motivation

Test configuration

Test results

Test conclusion

CONCLUSION

REFERENCES

APPENDIX

Appendix I – Oracle Advanced Row Compression

Minimal performance overhead

Background compression

Appendix II – Oracle AWR analysis of storage-related metrics

AWR Load Profile

AWR Top Foreground Events

AWR data file read and write I/O metrics

AWR and redo logs

Redo log switches

Redo log file sync

iostat analysis

Host CPU analysis
EXECUTIVE SUMMARY

VMAX® All Flash array is a system designed and optimized for high performance, while providing the ease of use, reliability, availability, security, and versatility of VMAX data services. This white paper explains and demonstrates why a VMAX All Flash array is an excellent platform for Oracle mission-critical databases. It provides a brief overview of VMAX All Flash portfolio and design for scale-up and scale-out. It also explains some of the key architecture components and how they benefits Oracle database performance and availability.

The paper has an extensive best practices section covering topics from host and storage connectivity, number and size of devices, ASM striping and database aspects. These guidelines can help customers understand how to leverage VMAX features best, and how to deploy Oracle databases with VMAX All Flash. It also covers VMAX Adaptive Compression Engine, explains how it works, and how it contrasts with Oracle Advanced Row Compression.

Four test cases in the paper demonstrate VMAX versatility and performance. These include three different types of OLTP workloads: a corner case where the active data set is so large it does not benefit from VMAX cache, a more typical case where VMAX cache provides moderate benefit to the workload, and a case of an ‘in-memory’ database where all reads are serviced from the database cache. The fourth use case is for a data warehouse workload with sequential read query trying to achieve best read bandwidth.

Finally, the appendixes include discussions about Oracle Advanced Row Compression, and a high-level description of how to make use of Oracle AWR and host statistics to analyze performance bottlenecks related to I/O.

It is not possible to cover all the Oracle and VMAX best practices in a single paper. Therefore the references at the end include links to other material with best practices such as Oracle backup and recovery and business continuity best practices, and more.

AUDIENCE

This white paper is intended for database and system administrators, storage administrators, and system architects who are responsible for implementing, managing, and maintaining Oracle databases and VMAX storage systems. Readers should have some familiarity with Oracle and the VMAX family of storage arrays, and be interested in achieving higher database availability, performance, and ease of storage management.

INTRODUCTION TO VMAX ALL FLASH

Overview

The VMAX family of storage arrays is built on the strategy of simple, intelligent, modular storage. It incorporates a Dynamic Virtual Matrix interface that connects and shares resources across all VMAX engines, allowing the storage array to grow seamlessly from an entry-level configuration into the world’s largest storage array. It provides the highest levels of performance, scalability, and availability featuring advanced hardware and software capabilities.

Figure 1 VMAX All Flash storage arrays 950F (left), and 250F (right)
In 2016, Dell EMC announced new VMAX® All Flash products: VMAX 250F, VMAX 450F and VMAX 850F. In May 2017 Dell EMC announced VMAX 950F, which replaces the VMAX 450F and 850F by providing higher performance at a similar cost. The VMAX All Flash offers a combination of ease of use, scalability, high performance, and a robust set of data services that makes it an ideal choice for Oracle database deployments.

**Ease of use**—VMAX uses virtual provisioning to create new storage devices in seconds. All VMAX devices are thin, consuming only the storage capacity that is actually written to, which increases storage efficiency without compromising performance. VMAX devices are grouped into storage groups and managed as a unit, including device masking to hosts, performance monitoring, local and remote replications, compression, host I/O limits, and more. In addition, VMAX management can be done using Unisphere for VMAX, Solutions Enabler CLI, or REST APIs.

**High performance**—VMAX All Flash is designed for high performance and low latency. It scales from one up to eight engines (V-Bricks). Each engine consists of dual directors, each with 2-socket Intel CPUs, front-end and back-end connectivity, hardware compression module, Infiniband internal fabric, and a large mirrored and persistent cache.

All writes are acknowledged to the host as soon as they registered with VMAX cache and only later, perhaps after multiple updates, are written to flash. As a result, log writes, checkpoints, or batch processes are extremely fast. Reads also benefit from the VMAX large cache. When a read is requested for data that is not already in cache, FlashBoost technology delivers the I/O directly from the back-end (flash) to the front-end (host) and only later staged in the cache for possible future access. VMAX also excels in servicing high bandwidth sequential workloads leveraging pre-fetch algorithms, optimized writes, and fast front-end and back-end interfaces.

**Data services**—VMAX All Flash excels in data services. It natively protects all data with T10-DIF from the moment data enters the array until it leaves (including replications). With SnapVX™ and SRDF®, VMAX offers many topologies for consistent local and remote replications. VMAX offers optional Data at Rest Encryption (D@RE), integrations with Data Domain® such as ProtectPoint™, or cloud gateways with CloudArray™. Other data services, including Quality of Service (QoS)², Compression, “Call-Home” support feature, non-disruptive upgrades (NDU), non-disruptive migrations (NDM), and more. In virtual environments VMAX also offers support for VAAI primitives such as write-same, xcopy, and others.

While outside the scope of this paper, VMAX can also be purchased as part of a Converged Infrastructure (CI) called VxBlock™ System 740.

**Storage design**

VMAX All Flash offers greater simplicity for sizing, ordering, and managing storage systems. The VMAX deployment allows scale-out using V-Bricks. V-Bricks is comprise of a VMAX engine and a starter brick Flash Capacity Pack. V-Bricks can be further scaled-up with additional increment Capacity Packs, as shown in Figure 2. VMAX All Flash systems can be ordered with pre-packaged software bundles—the entry “F” package, or the more encompassing “FX” package. They also come standard with embedded Unisphere® for VMAX, a management and monitoring solution that provides a single management view across VMAX storage systems.

---

1 VMAX All Flash cache is large (from 512 GB-16 TB, based on configuration), mirrored, and persistent due to the vault module that protects the cache content in case of power failure, and restores it when the system comes back up.

2 Two separate features support VMAX QoS. The first relates to Host I/O limits that allow placing IOPS and/or bandwidth limits on “noisy neighbors” applications (set of devices) such as test/dev environments. The second relates to slowing down the copy rate for local or remote replications.

3 VMAX 450F, 850F, and 950F use starter Flash capacity Pack of 53 TBu and incremental Capacity Packs of 13 TBu. VMAX 250F uses 11 TBu starter and incremental Flash Capacity Packs.
The VMAX engine consists of two redundant directors. Each director includes mirrored cache, host ports, hardware compression module, and software emulations based on the services and functions the storage was configured to provide. For example, FC, iSCSI, eNAS, and SRDF. Each director also includes CPU cores that are pooled for each emulation and serve all its ports.

The VMAX All Flash array comes pre-configured with flash storage, cache, and connectivity options based on the parameters provided during the ordering process. The flash drives are spread across the backend, logically split into thin data devices, or TDATs. The TDATs are RAID protected and pooled together to create compressibility pools. The aggregation of the TDAT pools become a Storage Resource Pool (SRP), where VMAX systems typically ship with only one SRP.

When host-addressable thin devices (TDEVs) are created they appear to be normal LUNs to the host, in fact they are a set of pointers that do not consume any capacity. When the host starts writing to them, their capacity is consumed in the SRP and the pointers are updated accordingly. By separating the logical entity of the TDEV from its physical storage in the TDATs, VMAX can operate on the data at a sub-LUN granularity, moving extents as needed for the sake of replications and compression.

Because VMAX comes pre-configured, when it is powered up, activities can immediately focus on physical connectivity to hosts, zoning, and device creation for applications. VMAX All Flash deployment is fast, easy, and focused on the applications’ needs.

For more information on VMAX All Flash refer to the following note: https://www.emc.com/collateral/white-papers/h14920-intro-to-vmax-af-storage.pdf.

Adaptive Compression Engine (ACE)

VMAX All Flash uses a compression strategy that is targeted to provide best data reduction without compromising performance. The VMAX Adaptive Compression Engine (ACE) is the combination of core compression functions and components: Hardware Acceleration, Optimized Data Placement, Activity Based Compression (ABC), and Fine Grain Data Packing.

Hardware acceleration—Each VMAX engine is configured with two hardware compression modules (one per director) that handle the actual compression and decompression of data, reducing compression overhead from other system resources.

Optimized data placement—Based on the compressibility of the data, it is allocated in different compression pools that provide compression ratio (CR) from 1:1 (128 KB pool) up to 16:1 (8 KB pool) and are spread across the VMAX backend for best performance. The pools are dynamically added or deleted based on need, and if not enough capacity is available in a high CR pool, data may temporarily reside in a lower CR pool and automatically move to its right location a short time later when it is created. That means that storage group CR may improve after a few hours when this happens, even if the data has not changed.

Activity Based Compression (ABC) —Typically, the most recent data is the most active data, creating an access skew. ABC relies on that skew to prevent constant compression and decompression of data that is hot or frequently accessed. The ABC function marks the 20 percent busiest data in the SRP to skip the compression flow, regardless of the related storage group compression setting. This means that the portion of the data that is highly active will remain uncompressed, even if its storage group as a whole has compression enabled. As the data matures and become less active, it will automatically compresses and newly active data will be part of the 20 percent data set that remains uncompressed (as long as storage capacity is available in the SRP).

Note: Due to Activity Based Compression, storage group compression ratio may be lower than its potential. This behavior is more emphasized with newly deployed but active applications, or in performance test environments, until an access, skew is created over time and the dormant data gets compressed.

Fine Grain Data Packing—When VMAX compresses data, each 128K track is split into four 32K buffers. Each buffer is compressed individually in parallel, maximizing the efficiency of the compression I/O module. The total of the four buffers result in the final compressed size and determines in which pool the data is allocated. Included in this process is a zero reclaim function that prevents the allocation of buffers with all zeros or no actual data. In the event of a partial write updates or read I/O, if only one or two of the sections need to be updated or read, only that data is uncompressed.

When using Unisphere, VMAX compression is enabled by default when creating new storage groups, and can be disabled by unchecking the compression checkbox. Unisphere also includes views and metrics that show the compression ratio of compressed storage groups, potential compressibility of uncompressed storage groups, and more.

The following section demonstrate some of the actions and reports using Solutions Enabler CLI.

---

4 The activity measurement is done by VMAX FAST algorithms that operate at small sub-LUN granularity. They respond very fast to new activity and slower to reduced activity (to prevent data from being considered “inactive” just because it is a weekend, for example).
When using Solutions Enabler CLI, to enable compression the storage group has to be associated with the SRP, such as in the following example:

```
# symsg -sg data_sg set -srp SRP_1 -compression
```

In a similar way, to disable compression on a storage group where compression is enabled:

```
# symsg -sg data_sg set -srp SRP_1 -nocompression
```

To display the compression ratio of a storage group (add: '-detail' to include the compression pools, where).

```
# symcfg list -tdev -sg data_sg -gb
```

To display the estimated compression ratio of storage groups, including SG's with compression disabled:

```
# symcfg list -sg_compression -by_compressibility -all
```

To display overall system efficiency:

```
# symcfg list -efficiency -detail
```

For more information on the VMAX Adaptive Compression Engine refer to the following white paper: VMAX All Flash with the Adaptive Compression Engine.
Host and storage connectivity

HBA ports (initiators) and storage ports (targets) are connected to a fibre channel (FC) or gigabit Ethernet (GbE) switch. FC connectivity requires zone sets on the switch that define which initiator has access to which target. The physical connectivity, together with zones creates the I/O paths between the host and storage. The number of paths can strongly affect performance aspects of the host and database. When planning for host and storage connectivity please consider the following guidelines.

- When zoning host initiators to storage target ports, ensure that each pair is on the same switch. Performance bottlenecks are often created when I/Os need to travel through ISL (paths between switches) because those are a shared resource and may not support the full server-to-storage bandwidth, as shown in Figure 3.

![Figure 3 ISL potential bottleneck](image)

- Use at least two HBAs for each database server to provide better availability and scale. Use multipathing software like EMC PowerPath® or Linux Device-Mapper to load-balance and to automatically failover paths. Use commands like `powermt display paths` or `multipath -l` to check that all paths are visible and active. Run it from each database server.

- Consider the number of paths that will be available to the database storage devices. Each path between the host and storage creates additional SCSI representation for the database devices on the host (/dev/sd*). In addition, the multipathing software creates one more ‘pseudo’ device (for example, /dev/emcpowera, or /dev/dm-1).

![Figure 4 host/storage connectivity](image)

  - While more paths add I/O queues and the potential for more concurrency and performance, consider that server boot time is affected by the number of SCSI devices discovered (one for each path combination x number of devices, plus one pseudo device name). Also, once connectivity needs are satisfied for performance and availability, additional paths do not add more value, though they do add to the boot and discovery time.

  - In general, for best availability and performance we recommend having each HBA port zoned-masked to two VMAX ports, as shown in Figure 4. The VMAX ports should be on different engines and directors when possible.

- Consider the number and speed of ports when planning bandwidth requirements. For example, a 8 Gb FC port can deliver up to approximately 800 MB/sec of bandwidth. Therefore, a server with 4 x 8 GB ports can deliver roughly 3 GB/sec bandwidth. Also consider that between the host initiators, storage ports, and switch, the speed of the slowest component will be negotiated and used for that path.
Number and size of host devices

VMAX uses thin devices exclusively. That means that unless devices are specifically requested on creation to fully allocate their capacity in the SRP, they only consume as much capacity as the application actually writes to the thin device.

This approach allows for capacity savings based on thin-devices, as storage is only consumed with actual demand. For example, a newly created 2 TB device does not consume any storage capacity, yet looks to the host just like a normal 2 TB LUN.

New devices can be easily created using Unisphere for VMAX, or using a command-line interface (CLI) like this one:

```
# symdev create -tdev -cap 500 -captype gb -N 4 -v # create 4 x 500GB thin devices
```

VMAX host devices can be sized from a few megabytes to multiple terabytes. Therefore, the user may be tempted to create only a few very large host devices. Consider the following:

- When Oracle ASM is used, devices (members) of an ASM disk group should be of similar capacity. If devices are sized very large from start, each increment will also be very large, perhaps creating waste if that capacity is never used.

- Oracle ASM best practice is to add multiple devices together to increase disk group capacity, rather than adding one at a time. This spreads ASM extents during rebalance in a way that avoids hot spots. Size the database devices so that a few devices can be added with each increment.

- As explained earlier, each path to a device creates a SCSI representation on the host. Each representation provides a host I/O queue for that path. Each queue can hold a limited (default of 32 on RHEL 7) number of I/Os simultaneously. Provide enough database devices for concurrency (multiple I/O queues), but not so many that it creates management overhead. Commands such as `iostat -xtzm` can show host queue sizes per device. Refer to the appendix for an example on how to monitor host queue lengths.

- Finally, another benefit of using multiple host devices is that internally the storage array can use more parallelism for operations such as data movement and local or remote replications, thus shortening the time the operation takes.

While there is no one size that fits all for the size and number of host devices, Dell EMC recommends a low number that offers enough concurrency. Each device sized to provide adequate building block for capacity increments when additional storage is needed, and does not become too large to manage.

As an example, 8-16 data devices and 4-8 log devices are often sufficient for high performance database of up to 32 TB (Oracle ASM limits devices up to 2 TB in size). With larger databases or higher demand for concurrency, more devices will be needed.
Consistent device names across hosts for RAC

When Oracle RAC is used, the same storage devices are shared across the cluster nodes. ASM places its own labels on the devices in the ASM disk group; therefore, matching their host device presentation is not important to Oracle. However, to the user it often makes storage management operations simpler. This section describes naming devices with Linux Device Mapper (DM), Dell EMC PowerPath multipathing software, and VMware multiplathing.

Linux Device Mapper example

By default, Device Mapper (DM) uses WWID to identify devices uniquely and consistently across hosts. While this is sufficient, often the user prefers “user-friendly” names, for example, /dev/dm-1, /dev/dm-2, or even aliases, for example, /dev/mapper/ora_data1, /dev/mapper/ora_data2.

This is an example for setting up /etc/multipath.conf with aliases. To find the device WWN use the Linux command scsi_id -g /dev/sdXX.

```
# /usr/lib/udev/scsi_id -g /dev/sdb
360000970000198700067533030314633

# vi /etc/multipath.conf
multipaths {
    multipath {
        wwid 360000970000198700067533030314633
        alias ora_data1
    }
    multipath {
        wwid 360000970000198700067533030314634
        alias ora_data2
    }
}
```

To match user-friendly names or aliases across cluster nodes, follow these steps:

1. Set up all multipath devices on one host. Then disable multipathing on the other hosts. For example:

```
# service multipathd stop
# service multipath -F
```

2. Copy the multipath configuration file from the first host to all other hosts. If user friendly names need to be consistent then copy the /etc/multipath/bindings file from the first host to all others. If aliases need to be consistent (aliases are setup in multipath.conf), copy the /etc/multipath.conf file from the first host to all others.

3. Restart multipath on the other hosts. For example:

```
# service multipathd start
```

4. Repeat this process if new devices are added.

PowerPath example

To match PowerPath pseudo device names between cluster nodes, follow these steps:

1. Use the `emcpadm export_mappings` command on the first host to create an XML file with the PowrPath configuration. For example:

```
# emcpadm export_mappings -f /tmp/emcp_mappings.xml
```

2. Copy the file to the other nodes.

3. On the other nodes import the mapping. For example:

```
# emcpadm import_mappings -f /tmp/emcp_mappings.xml
```
Note: The PowerPath database is kept in two files: /etc/emcp_devicesDB.idx and /etc/emcp_devicesDB.dat. These can be copied from one of the servers to the others, followed by reboot. The emcpadm export/import method is the recommended way to match PowerPath device names across hosts, where the file copy is a shortcut that will over-write existing powerpath mapping on the other hosts.

VMware native multipathing example
When running Oracle from a VMware VM the native ESX multipathing is in effect and the VM devices appear as /dev/sdXX. One way to match device names across hosts and maintain their permissions is to use ASMLib. However, when ASMLib is not used, udev rules can be used to create device aliases and maintain device permissions. The following is an example for the device: /dev/sdb1 giving it an alias of /dev/ora-data1. The ASM disk string will therefore be: /dev/ora* and the alias will create a symbolic link like: /dev/ora-data1 --> /dev/sdb1.

Note: In RHEL 7 the scsi_id command is located in /lib/udev/scsi_id. In previous releases it was located in /sbin/scsi_id.

```
# /lib/udev/scsi_id -g -u -d /dev/sdb
36000c29127c3ae0670242b058e863393

# cd /etc/udev/rules.d/
# vi 99-oracle-asmdevices.rules
KERNEL=="sd*1", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d /dev/$parent", RESULT=="360000970000196800312533030344131", SYMLINK=="ora-data1", OWNER="oracle", GROUP="dba", MODE="0660"
```

Device permissions
As of Oracle or RedHat Linux 6, device permissions are determined using udev rules in /etc/udev/rules.d/.

Device Mapper Oracle user permissions example
In the example below, the devices with aliases starting with "ora_" and using partition number 1 (p1) will be updated to oracle user with group dba. If a more specific list of devices is required, replace the device description: "ora_*p1" appropriately, or use a separate line for each device.

```
# vi /etc/udev/rules.d/12-dm-permissions.rules
ENV{DM_NAME}=="ora_*p1", OWNER=="oracle", GROUP=="dba", MODE=="660"
```

PowerPath Oracle user permission example
In the example below, all the PowerPath devices with partition 1 will be updated to oracle user with group dba. If a more specific list of devices is required, replace the device description: "emcpower*1" appropriately, or use a separate line for each device.

```
# vi /etc/udev/rules.d/85-oracle.rules
ACTION=="add|change", KERNEL=="emcpower*1", OWNER=="oracle", GROUP=="dba", MODE=="660"
```

Partition alignment
Oracle recommends creating one partition on each ASM device. By default, Oracle or RedHat Enterprise Linux release 6 and earlier use a partition offset of 63 blocks, or 63 x 512 bytes = 31.5 KB. Because VMAX uses a 128 KB track size, the default offset can create performance overhead. A 1 MB partition alignment (offset) for Oracle ASM deployments with VMAX is strongly recommended.

Note: Oracle or RedHat Linux release 7 use a default of 1 MB offset when the first partition is created and therefore no manual partition alignment is required. Also, although Oracle recommends creating a partition, ASM doesn’t actually require one. If ASM is given the full device (for example, ASM disk_string points to /dev/emcpower*) then the beginning of a device is Logical Block Address (LBA) 0, which is aligned by default.

One way to align partitions is to use the fdisk command to create a single primary partition on each device. Use “x” to enter fdisk expert mode, use “b” to change the partition offset. Enter: 2,048 for a 1 MB offset (2,048 x 512 bytes).

A simpler way is to use the Linux parted command, as shown in the following example:
for i in {a..h}; do
  parted -s /dev/emcpower$i mklabel msdos
  parted -s /dev/emcpower$i mkpart primary 2048s 100%
  chown oracle.dba /dev/emcpower$i
done
fdisk -lu # this command lists all host devices and shows partitions offset

When RAC is used, other nodes will not be aware of the new partitions. This can be resolved by simply reading and writing the partition table from all other nodes, without any changes to it. For example:

for i in {a..h}; do
  fdisk /dev/emcpower$i << EOF
  w
  EOF
  chown oracle.dba /dev/emcpower$i
done
fdisk -lu # this command lists all host devices and shows the partitions' offset

Oracle ASM best practices

ASM disk groups

In general, for mission critical databases deployed on VMAX, Dell EMC recommends that the following data types are separated to a distinct ASM disk groups. Use external redundancy (no ASM mirroring) for all ASM disk groups except +GRID.

- **+GRID**: If Oracle RAC is used, it keeps the cluster configuration and quorum files in the initial ASM disk group. Only for this initial disk group Dell EMC recommends using Normal Redundancy (ASM dual mirroring). The reason is that this disk group should not contain any user data and therefore remain very small. When set to normal redundancy, Oracle creates three quorum files rather than a single one (such as if external redundancy was selected instead). Having three quorum files helps in avoiding delays while nodes try to register with the quorum during high database activity.

  **Note**: Do not store database files in +GRID disk group. When the SRDF or SnapVX replications are used, the ASM disk group with database files can be replicated to any other cluster environment, unclustered environment, or restored back to the original environment without overwriting the cluster information residing within the +GRID disk group.

- **+DATA**: A minimum of one disk group for data and control files. Large databases may use more disk groups. A minimum of eight devices are recommended for performance and scale.

- **+REDO**: Online redo logs. A single ASM disk group dedicated for the online logs to allow separation of data from logs, primarily for performance and monitoring reasons, but also to allow the use of storage snapshots to offload backups, without overwriting the online logs during a restore.

  **Note**: While it is the choice of the DBA, we do not believe that there is a reason to multiplex the redo logs, as it double the log writes. Redo logs are extremely resilient for corruption by utilizing a very small block size (512 bytes default on Linux).

- **+TEMP** (optional): Temp files can typically reside with data files. However, when TEMP is very active and large, the DBA may decide to separate TEMP to its own ASM disk group. Also, when SRDF is used, if TEMP files are separated, there is no need to replicate them, saving bandwidth.

  **Note**: If Temp is very active in reading and writing to storage, it is recommended that it uses fine-grain ASM striping by modifying its ASM template, regardless if it is separated to its own ASM disk group. Refer to the next section.

- **+FRA**: Typically used for archive and/or flashback logs. If flashback logs are used and consume a lot of space, the DBA may decide to separate them to their own ASM disk group.
ASM striping

By default, ASM uses an Allocation Unit (AU) size of 1 MB and stripes the data across the whole disk group using the AU as its stripe-depth. This default ASM striping method is called: Coarse Striping, and is optimal for OLTP type applications as they generate small random I/O and ASM spreads them across the disk group devices. The DBA may decide to increase the size of the AU though there is not a clear benefit for doing so in an OLTP environment due to the random I/O behavior.

ASM has an alternative striping method called Fine-Grain Striping. With Fine-Grain Striping ASM selects eight devices in the disk group (if available), allocates an AU on each, and further stripes (divide) the AU to pieces of 128 KB in size. It then allocates data in a round-robin fashion across the eight devices using the 128 KB stripe-depth. Once all 8 AUs are full, it repeats the process by selecting again a set of eight devices and so on.

Fine-Grain striping is the VMAX recommended striping method for Oracle objects with primarily sequential writes. Because sequential reads and writes tend to be large I/Os, by breaking them to 128 KB stripes, latencies improve, and VMAX processing is optimal (as VMAX track size is also 128 KB). In low-activity databases this will not be noticeable, and therefore default coarse-striping can be used, in highly-active databases it is recommended to use Fine-Grain striping for the online logs. This is especially applicable to In-Memory databases where transactions are very fast and redo load can get very heavy, or batch data loads that tax the redo logs heavily. In Data Warehouses where Oracle Temp files can get very busy with I/Os, Temp files can also benefit from Fine-Grain Striping.

The type of striping for each Oracle data type is kept in ASM template. Each ASM disk group has its own set of templates so modifications only apply to the appropriate disk group. Furthermore, existing ASM allocations (extents) are not affected by template changes and therefore it is best to set the ASM templates correctly as soon as the ASM disk group is created, or recreate the object, such as redo logs or temp files after the change (easy operation for these objects).

To inspect the ASM templates execute the following command from the ASM instance:

```sql
SQL> select DG.name Disk_Group, TMP.name Template, TMP.stripe from v$asm_diskgroup DG, v$asm_template TMP where DG.group_number=TMP.group_number order by DG.name;
```

To change the database redo logs template in the +REDO ASM disk group:

```sql
SQL> ALTER DISKGROUP REDO ALTER TEMPLATE onlinolog ATTRIBUTES (FINE);
```

To change the temp files template in the +TEMP ASM disk group:

```sql
SQL> ALTER DISKGROUP TEMP ALTER TEMPLATE tempfile ATTRIBUTES (FINE);
```

VMAX masking views

VMAX uses masking views to determine which devices are visible to the hosts. A masking view contains a Storage Group, Port Group, and Initiator Group. When a masking view is created, the set of devices in the storage group are made visible to the appropriate hosts via their initiators with access to storage via the ports in the port group. When changes are made to any of the masking view components, they automatically propagated, such as adding or removing devices from the storage group, adding initiators (hosts), or storage ports.

The following sections show an example of creating storage groups, port groups, initiator groups and masking views using Solutions Enabler CLI.

**Note:** Dell EMC recommends separating data files from log files at both the ASM disk group level as well as the SG level. For that reason in the following example we create data and log SG’s, and then use a parent SG that contains both. In that way, when using SnapVX, a snapshot of the parent creates a restartable copy of the database, while for backup/recovery purposes, just the data SG can be restored without overwriting the logs.

- **Device creation:** when using Unisphere, storage provisioning creates devices and SG’s together. When using CLI, we first create devices, then we group them in SGs. The following is an example for creating new devices. Use `-v` option to have the command output the new device ID’s.

```
# symdev create -tdev -cap 500 -captype gb -N 4 -v # create 4 x 500GB thin devices
```
• **Storage Group (SG):** A group of devices (TDEVs) that can be made visible to hosts. Furthermore, storage groups can be cascaded, such as a parent SG contains a group of child SGs and each child SG contains a group of devices. The parent SG is used to mask all the database devices as a unit to hosts, or for replications of the whole database. The child SGs can be used for backup/recovery solution (such as snapshot of the data files alone) or for performance monitoring at a SG level.

```bash
set -x
export SYMCLI_SID=<SID>
export SYMCLI_NOPROMPT=1

# Create SGs
symsg create grid_sg  # Stand-alone SG for Grid infrastructure devices
symsg create fra_sg   # Stand-alone SG for archive logs
symsg create data_sg  # Child SG for data and control file devices
symsg create redo_sg  # Child SG for redo log devices
symsg create db_sg    # Parent SG for database (data+redo) devices

# Add appropriate devices to each SG
symsg -sg grid_sg addall -devs 64:66
symsg -sg data_sg addall -devs 67:76
symsg -sg redo_sg addall -devs 77:7E
symsg -sg fra_sg addall -devs 7F:82

# Add the child SGs to the parent
symsg -sg db_sg add sg data_sg,redo_sg
```

• **Port Group (PG):** A group of storage front-end ports for host connectivity. Ensure the ports are zoned correctly to the switch. The following example creates a port group with 16 ports.

```bash
symaccess -type port -name 048_pg create
symaccess -type port -name 048_pg add -dirport 1D:4,1D:5,1D:6,1D:7
symaccess -type port -name 048_pg add -dirport 2D:4,2D:5,2D:6,2D:7
symaccess -type port -name 048_pg add -dirport 1D:8,1D:9,1D:10,1D:11
symaccess -type port -name 048_pg add -dirport 2D:8,2D:9,2D:10,2D:11

Note: Often the port group includes all designated ports for the database, while the zone-sets determine the exact pathing strategy (for example, each HBA initiator zoned to two FA ports).

• **Initiator Group (IG):** A group of host initiators (HBA ports). Dell EMC recommends assigning each database server its own initiator group for ease of management. In a cluster, these IGs can be cascaded into a parent IG. If nodes are added or deleted over time, the parent IG simply adds or removes the child SGs that are based on hosts. The following example creates a child IG for each server and a parent IG.

```bash
# cat /sys/class/fc_host/host*/port_name # Provides a list of ports WWN’s
symaccess -type initiator -name dsib0144_ig create
symaccess -type initiator -name dsib0144_ig add -wwn 10000090faa910b2
symaccess -type initiator -name dsib0144_ig add -wwn 10000090faa910b3
symaccess -type initiator -name dsib0144_ig add -wwn 10000090faa90f86
symaccess -type initiator -name dsib0144_ig add -wwn 10000090faa90f87

symaccess -type initiator -name dsib0146_ig create
symaccess -type initiator -name dsib0146_ig add -wwn 10000090faa910aa
symaccess -type initiator -name dsib0146_ig add -wwn 10000090faa910ab
symaccess -type initiator -name dsib0146_ig add -wwn 10000090faa910ae
symaccess -type initiator -name dsib0146_ig add -wwn 10000090faa910af

symaccess -type initiator -name db_ig create
symaccess -type initiator -name db_ig add -ig dsib0144_ig
symaccess -type initiator -name db_ig add -ig dsib0146_ig
```
Key points regarding VMAX Adaptive Compression Engine (ACE):

- VMAX thin provisioning ensures that regardless of the LUN size, only the capacity that was written to is allocated in the storage, and VMAX compression works on the allocated capacity to further reduce its footprint. Together, VMAX thin provisioning and compression provide a powerful data reduction solution.

- VMAX ACE compresses all the allocated capacity – be it user data, system tables, undo, archive logs, empty data files, and so on. As a result it has more capacity to work with and can often result in very good CR. In comparison, Oracle Advanced Row Compression only works on database tables and indexes.

 symaccess create view -name db_mv -pg 048_pg -ig db_ig -sg db_sg
 symaccess create view -name fra_mv -pg 048_pg -ig db_ig -sg fra_sg
 symaccess create view -name grid_mv -pg 048_pg -ig db_ig -sg grid_sg

4 KB redo log sector size

Starting with Oracle 11gR2 Oracle introduced the ability to change the redo log block size from its 512 bytes per sector to 4 KB. The reason being that certain drives used 4 KB as their native block size (for example, SSD drives). Another reason was to reduce the metadata overhead associated with very high density drives by increasing the block size from its legacy 512 bytes to 4 KB.

There are two key reasons why when using VMAX all flash storage the redo log block size should not be changed from the default 512 bytes per sector. The first is that the flash drives are never written-to directly by the database. Instead, all writes to VMAX arrive at the VMAX cache, where they can be aggregated to provide optimized writes to the flash media at a later time. Therefore such a change will not benefit the drives directly. The second reason is the increase in redo wastage, which is often very significant. When the Oracle database commits frequently, it requires an immediate write of the log buffer. With 4 KB blocks they are often mostly empty, therefore creating unnecessary write overhead and redo wastage.

Kernel I/O scheduler

It is important to note the effect of the Operating System kernel I/O scheduler on database performance. According to RedHat, the preferred scheduler for databases is Deadline. The default scheduler however was only changed to Deadline with RHEL 7, and was CFQ with prior releases. It is highly recommended to make sure that the IO scheduler is set to Deadline for the databases devices.

VMAX compression and Oracle Advanced Compression

As described in the introduction above, VMAX Adaptive Compression Engine (ACE) uses a versatile architecture to achieve high storage compression ratio while taking in account data activity. When considering Oracle Advanced Compression and/or VMAX compression the following points provide insight.

- In general, compression ratio (CR) always depends on the actual data, that is, data that is completely random without repetition or pattern (for example, encrypted data) will not compress as well as an empty data file or a table with low data cardinality.

- Because each database is different, it is hard to generalize CR expectations. However, based on a research by Oracle, an E-Business Suite 12.1.3 database compressed from 1,200 GB to 500 GB, which is a CR of 2.4:1.

- Based on studies with VMAX, a typical CR is 2:1 and better. Again, this is just to provide general idea for CR expectations. Activity Based Compression may reduce the effective CR based on the amount of active data in the storage group.

- To demonstrate some of the points in the following sections, we used a SLOB database with 48 user schemas and scale of 200 GB. We set PCT FREE to default of 20 percent and so each user schema consumed a 100 GB for a total of 48 x 100 GB = 4.8 TB. ASM capacity included the system, sysaux, undo, and SLOB tablespaces, for a total of 5,727 GB. To test database compression we used table move on the SLOB uncompressed tables. To test performance we ran the SLOB benchmark with 25 percent update and lite redo stress.

Note: SLOB ‘data’ is a repeating string of characters and therefore compresses extremely well. The CRs achieved in the following sections are likely higher than CR of a real customer data.

Key points regarding VMAX Adaptive Compression Engine (ACE):

- VMAX thin provisioning ensures that regardless of the LUN size, only the capacity that was written to is allocated in the storage, and VMAX compression works on the allocated capacity to further reduce its footprint. Together, VMAX thin provisioning and compression provide a powerful data reduction solution.

- VMAX ACE compresses all the allocated capacity – be it user data, system tables, undo, archive logs, empty data files, and so on. As a result it has more capacity to work with and can often result in very good CR. In comparison, Oracle Advanced Row Compression only works on database tables and indexes.
• With VMAX compression database activity may temporarily reduce the CR, but when the data is no longer active it gets compressed automatically in the background. Oracle can accomplish a similar behavior using ADO policies, but this involves much higher complexity level, management overhead, and the capacity freed is within the database tables, but not ASM or storage.

• As shown in Figure 5, we compared the storage groups’ allocations of SLOB data and Oracle system tables before and after VMAX compression. Prior to enabling VMAX compression, the allocated capacity was 5,727 GB (matching perfectly with ASM, as shown in Figure 7 later). After VMAX compression was enabled (and without activity, avoiding ABC effect), the allocated capacity was reduced to 721 GB, a CR of 7.9:1 and data reduction of 5 TB.

![VMAX Storage Group capacity (GB)](image)

**Figure 5** VMAX Advanced Compression Engine test – data reduction

• Figure 6 shows the percent overhead of CPU and IOPS we experienced in our test environment, using SLOB benchmark before and after VMAX compression was enabled on the database system and SLOB storage groups. The IOPS numbers are based on AWR physical reads and writes, and CPU overhead is the difference AWR shows for % DB CPU used before and after VMAX compression was enabled. We can see that after VMAX compression the database CPU utilization was the same and IOPS were reduced by 1%.

![IOPS and CPU After VMAX Compression](image)

**Figure 6** VMAX Advanced Compression Engine test – CPU and performance

**Key points regarding Oracle Advanced Row Compression**

• Oracle Advanced Row Compression is a licensed database option which only operates on table or index data. It does not free storage capacity and only makes more room in the tables or indexes for additional data. That means that allocated storage capacity, such as data files, tablespaces, and ASM remain allocated, even if these are empty files or unused extents (e.g. deleted files).

• Oracle uses a background compression that requires a trigger, such as moving a table or partition (heavy read/write I/O operation), or data updates that fill the data blocks beyond a compression threshold. As a result if a table was compressed (e.g. using table move), OLTP activity on compressed data tends to reduce its CR.
As since in Figure 7, the database extents of the SLOB tables consumed 4,806 GB before Oracle compression was enabled. After enabling Oracle compression AND performing a ‘table move’ on all SLOB tables (heavy I/O and CPU activity), SLOB tables consumed 195 GB. However, after running SLOB workload, the consumption increased to 692 GB. Therefore, the effective CR was 6.9:1 and although no actual storage was freed (as can be seen in green, looking at ASM allocations), 4.1 TB of data was made available inside the SLOB tables due to the database compression.

**Figure 7 Oracle Advanced Compression test – data reduction**

Consider using ADO (Automatic Data Optimization), a 12c feature that allows compressing tables or partitions only after their data is idle for a period of time, therefore, avoiding DML reduction of CR.

Since Oracle compression is done by software, there is an added CPU overhead associated with Oracle compression that can affect performance. Figure 8 shows the percent overhead of CPU and IOPS we experienced in our test environment using SLOB benchmark before and after Oracle compression was enabled on SLOB tables. The IOPS numbers are based on AWR physical reads and writes, and CPU overhead is the difference AWR shows for % DB CPU used before and after Oracle compression was enabled. We can see that after Oracle compression was enabled, the database CPU utilization increased by 7%, and IOPS were reduced by 11%.

**Figure 8 Oracle Advanced Compression test – CPU and performance**

**Additional considerations**

- When combined with Oracle Transparent Database Encryption (TDE), Oracle compression can operate on the pre-encrypted data and therefore can achieve higher CR than storage compression operating on the encrypted data.

- Oracle Advanced Row Compression allows each data block to contain more data and therefore each I/O carries a bigger payload. This is an advantage for data warehouses as the same number I/Os bring more data.

- Another potential advantage for using Oracle Advanced Row Compression is cache efficiency. By using Oracle compressed data the active data set can fit better in Oracle buffer cache and/or VMAX cache providing a potential performance benefits.

- When comparing Engineered Systems compression, such as Exadata Hybrid Columnar Compression (HCC), it is important to note that Exadata is commonly deployed with ASM High Redundancy ( triple-mirroring). While HCC provides exclusive advantage to Exadata, it has to offset the triple-mirroring capacity overhead. In comparison, VMAX uses External Redundancy (no ASM mirroring) and as mentioned above, its compression operates on the whole storage group and not just table data.
TEST ENVIRONMENT

Hardware and software configuration

The test environment is shown in Figure 9. It consisted of two Dell R730 servers, each with two Intel Xeon E5-2690v4 2.6GHz, and 28-cores. Each host utilized two dual-port host bus adapters (HBAs) for a total of four initiators per server connected to the SAN switches. The servers used two networks; the first was public for user connectivity and management, and the second was private for Oracle RAC. The servers were installed with RedHat 7.2, and Oracle 12.1 RAC database and Grid Infrastructure with Automatic Storage Management (ASM). ASM used external redundancy for all disk groups except the +GRID disk group, which is a small disk group created during Grid Infrastructure installation and did not contain user data. The database used an 8 KB block size and a very small buffer cache in order to generate as many I/Os as possible.

The servers used two networks; the first was public for user connectivity and management, and the second was private for Oracle RAC. The servers were installed with RedHat 7.2, and Oracle 12.1 RAC database and Grid Infrastructure with Automatic Storage Management (ASM). ASM used external redundancy for all disk groups except the +GRID disk group, which is a small disk group created during Grid Infrastructure installation and did not contain user data. The database used an 8 KB block size and a very small buffer cache in order to generate as many I/Os as possible.

The storage system was a VMAX 950FX, 1 V-Brick, with 32 x SSD drives in a RAID5 configuration. The VMAX was running HYPERMAX OS Q2 2017 release. Table 1 shows a list of the hardware and software components used.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>TYPE</th>
<th>QUANTITY/SIZE</th>
<th>VERSION/RELEASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage system</td>
<td>EMC VMAX 950F</td>
<td>1 x V-Brick, 1 TB raw cache, 32 x SSD drives in RAID5</td>
<td>HYPERMAX OS 5977.1124 based on Q2 2017 release</td>
</tr>
<tr>
<td>Database servers</td>
<td>Dell R730</td>
<td>2 servers clustered with: 28 cores 2.6 GHz, 128 GB RAM, 4 x FC ports</td>
<td>RedHat 7.2, PowerPath 6.1</td>
</tr>
<tr>
<td>Oracle Database</td>
<td>Oracle Database and Grid Infrastructure 12c, with ASM</td>
<td>two-node Oracle RAC</td>
<td>Oracle Database and Grid Infrastructure 12.1</td>
</tr>
</tbody>
</table>

Tested workloads

The purpose of the tests was to demonstrate the capabilities of the system using mixed workloads such as OLTP and DSS.

For the OLTP tests, we used SLOB 2.3. SLOB is an easy-to-use Oracle database benchmark tool. In our tests, we used 48 users (workload processes) with 2 threads each, and a 1.2 TB SLOB data set (48 users x 25 GB scale). We used a 25 percent update rate to simulate an OLTP-type workload. In some test cases, we ran against the whole database (to reduce storage cache read-hit benefits) and in others we used the SLOB locality feature to create a more cache-friendly workload and gain a higher read-hit rate with the VMAX cache. For test cases 1 and 2 we kept the Oracle buffer cache very small to help with I/O generation. In test case 3 (in-memory database), we increased the buffer cache size and as a result the data got cached in the database and the storage received only writes.

For DSS tests, we used Lineitem table, created by dbgen tool. Lineitem table was partitioned by date as a primary partition and a secondary hash partition. We generated near 1 TB of data and forced the Oracle SQL query to perform full-table scans via SQL hint, so we could measure the storage bandwidth and therefore the speed of query execution. We ran a series of queries to ensure the storage bandwidth was consistent and sustainable.
Test 1: OLTP tests with negligible VMAX cache read-hit benefits

Test Motivation
The purpose of this test case was to simulate a situation where the active data set is so large, that it does not get any read-hit benefits from the VMAX cache. As a result all reads are “misses” and have to be satisfied from the flash media instead of VMAX cache.

This is not a common scenario because even in very large databases, users typically focus on a small portion of the data (often the most recent data). This allows the VMAX cache algorithms to keep frequently accessed data in cache and satisfies a large portion of the read I/O requests without ever going to the flash drives.

Test configuration
We used a 2-node RAC to run SLOB 2.3 benchmark with a scale of 25 GB and 48 users for a total of 1.2 TB data set. SLOB was set for a 25 percent update rate, 2 threads per user process, and with light redo log stress. This test case resulted in just a 4 percent read-hit rate in the VMAX cache.

Test results
Based on the 30-minute AWR report from the run, even with as little as a 4 percent read-hit rate, the sum of “physical read total I/O requests” and “physical write total I/O requests” shows 216,255 + 59,371 = 275,626, a total of approximately 275K IOPS, as shown in Figure 10.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Second</th>
<th>per Trans</th>
<th>Average</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical read IO requests</td>
<td>388,951,508</td>
<td>216,244.79</td>
<td>262.20</td>
<td>108,122.40</td>
<td>164.19</td>
<td>108,006.30</td>
<td>108,238.50</td>
</tr>
<tr>
<td>physical read bytes</td>
<td>3,190,188,015,616</td>
<td>1,773,644,107.81</td>
<td>2,151,061.72</td>
<td>886,822,053.90</td>
<td>2,875,893.05</td>
<td>884,786,490.43</td>
<td>888,855,617.38</td>
</tr>
<tr>
<td>physical read total IO requests</td>
<td>386,970,083</td>
<td>216,255.12</td>
<td>262.27</td>
<td>108,127.56</td>
<td>166.98</td>
<td>108,009.49</td>
<td>108,245.63</td>
</tr>
<tr>
<td>physical read total bytes</td>
<td>3,190,908,731,392</td>
<td>1,774,044,803.75</td>
<td>2,151,547.68</td>
<td>887,022,401.87</td>
<td>3,087,022,401.87</td>
<td>889,204,117.23</td>
<td></td>
</tr>
<tr>
<td>physical read total multi block requests</td>
<td>27,230</td>
<td>15.14</td>
<td>0.02</td>
<td>7.57</td>
<td>10.70</td>
<td>0.00</td>
<td>15.14</td>
</tr>
<tr>
<td>physical reads cache</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>physical reads cache prefetch</td>
<td>471,346</td>
<td>262.05</td>
<td>0.32</td>
<td>185.14</td>
<td>261.94</td>
<td>0.11</td>
<td>261.94</td>
</tr>
<tr>
<td>physical reads direct</td>
<td>4,797</td>
<td>2.67</td>
<td>0.00</td>
<td>1.66</td>
<td>2.66</td>
<td>0.00</td>
<td>2.66</td>
</tr>
<tr>
<td>physical reads direct (lob)</td>
<td>11</td>
<td>0.01</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>physical reads direct temporary tablespace</td>
<td>4,780</td>
<td>2.66</td>
<td>0.00</td>
<td>2.66</td>
<td>2.66</td>
<td>2.66</td>
<td>2.66</td>
</tr>
<tr>
<td>physical write IO requests</td>
<td>103,315,882</td>
<td>57,440.39</td>
<td>69.66</td>
<td>28,720.19</td>
<td>2,586.51</td>
<td>26,891.26</td>
<td>30,549.13</td>
</tr>
<tr>
<td>physical write bytes</td>
<td>880,342,016,000</td>
<td>489,442,510.93</td>
<td>593,591.98</td>
<td>244,721,255.46</td>
<td>21,041,644.58</td>
<td>229,842,565.89</td>
<td>259,599,945.03</td>
</tr>
<tr>
<td>physical write total IO requests</td>
<td>106,789,450</td>
<td>59,371.58</td>
<td>72.01</td>
<td>29,685.79</td>
<td>2,661.14</td>
<td>27,804.08</td>
<td>31,567.50</td>
</tr>
<tr>
<td>physical write total bytes</td>
<td>964,984,926,720</td>
<td>536,501,304.22</td>
<td>650,664.52</td>
<td>268,250,652.11</td>
<td>20,970,243.91</td>
<td>253,422,450.43</td>
<td>283,079,853.78</td>
</tr>
<tr>
<td>physical write total multi block requests</td>
<td>11,933</td>
<td>6.63</td>
<td>0.01</td>
<td>3.32</td>
<td>1.43</td>
<td>2.31</td>
<td>4.32</td>
</tr>
<tr>
<td>physical writes</td>
<td>107,463,625</td>
<td>59,746.40</td>
<td>72.46</td>
<td>29,873.20</td>
<td>2,568.56</td>
<td>28,056.95</td>
<td>31,689.45</td>
</tr>
</tbody>
</table>

Figure 10 Test 1 – RAC AWR system Statistics (Global)
The latency for the data file random read\(^5\) I/O was about 0.9 ms, as shown in the AWR report in Figure 11. Note that Log Writer latency was only 0.4 ms (reported as ‘log file parallel write’ event), benefiting from writing to VMAX persistent cache.

<table>
<thead>
<tr>
<th>Class</th>
<th>Event</th>
<th>Wait Time</th>
<th>Summary Avg Wait Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User I/O</td>
<td>db file sequential read</td>
<td>388,957,940</td>
<td>335,531.17, 0.86, 97.47</td>
</tr>
<tr>
<td></td>
<td>DB CPU</td>
<td>18,141.91</td>
<td>5.27</td>
</tr>
<tr>
<td>System I/O</td>
<td>log file parallel write</td>
<td>1,759,348</td>
<td>712.60, 0.41</td>
</tr>
<tr>
<td>Cluster</td>
<td>gc cr grant 2-way</td>
<td>1,863,000</td>
<td>209.44, 0.11, 0.06</td>
</tr>
<tr>
<td>System I/O</td>
<td>db file parallel write</td>
<td>2,387,838</td>
<td>113.48, 0.03, 0.05</td>
</tr>
<tr>
<td>Cluster</td>
<td>gc cr disk read</td>
<td>1,300,187</td>
<td>105.81, 0.08</td>
</tr>
<tr>
<td>Cluster</td>
<td>gc current grant 2-way</td>
<td>626,122</td>
<td>76.49, 0.12</td>
</tr>
<tr>
<td>User I/O</td>
<td>db file scattered read</td>
<td>26,639</td>
<td>23.78, 0.89</td>
</tr>
<tr>
<td>User I/O</td>
<td>read by other session</td>
<td>24,248</td>
<td>4.81, 0.08</td>
</tr>
<tr>
<td>Cluster</td>
<td>gc remaster</td>
<td>3</td>
<td>6.00, 1999.44</td>
</tr>
</tbody>
</table>

Figure 11 Test 1 – RAC AWR TopTimed Events

We confirmed the numbers by comparing them to the VMAX storage statistics in Unisphere, as shown in Figure 12. You can see 275K IOPS by the data_sg with 0.8 ms read latency and 0.3 ms write latency. You can also see 1K IOPS by the redo_sg, with 0.3 ms write latency.

Figure 12 Test 1 – Unisphere DATA and REDO storage groups performance

Test conclusion

Even in such an uncommon scenario (worst-case scenario) as a workload without VMAX read cache benefits, writes continued to be served at sub-millisecond latency from the VMAX cache, and the total IOPS remained very high (over 270K) with <1 ms read latency.

---

\(^5\) Oracle "db file sequential read" event is in fact the host random read I/O. The event name can be confusing.
Test 2: OLTP tests with medium VMAX cache read-hit benefits

Test motivation
In this test case, the VMAX cache provides only medium benefits for database reads. The goal was to satisfy about 50 percent of the database read I/O from the VMAX cache, allowing the flash drives to service the rest.

A secondary goal was to look for latency outliers: when some reads were satisfied from the cache and some from flash drives, how much variation appeared in the database I/O latency?

Test configuration
As in the first use case, we used a 2-node RAC to run SLOB 2.3 benchmark with a scale of 25 GB and 48 users for a total of 1.2 TB data set. SLOB was set for a 25 percent update rate, two threads per user process, and with light redo log stress. However, now we have added a SLOB "hot-spot" (creating workload locality) to achieve a 50 percent read-hit rate in the VMAX cache. We ran the test for 30 minutes in a steady state (ignoring ramp-up and ramp-down periods).

Test results
Based on the 30-minute AWR report from the run, with VMAX read-hit rate of 50 percent, the sum of "physical read total IO requests" and "physical write total IO requests" shows 316,358 + 91,329 = 407,687: a total of about 400K IOPS, as shown in Figure 13.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Second</th>
<th>per Trans</th>
<th>Average</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical read IO requests</td>
<td>568,834,317</td>
<td>316,351.86</td>
<td>257.65</td>
<td>158,175.93</td>
<td>4,836.22</td>
<td>154,756.21</td>
<td>161,595.65</td>
</tr>
<tr>
<td>physical read bytes</td>
<td>4,659,901,874,176</td>
<td>2,591,560,599.00</td>
<td>2,110,643.47</td>
<td>1,295,780,299.50</td>
<td>39,615,733.09</td>
<td>1,267,767,745.99</td>
<td>1,323,792,853.01</td>
</tr>
<tr>
<td>physical read total IO requests</td>
<td>568,846,410</td>
<td>316,358.58</td>
<td>257.65</td>
<td>158,179.29</td>
<td>4,836.22</td>
<td>154,759.57</td>
<td>161,599.01</td>
</tr>
<tr>
<td>physical read total bytes</td>
<td>4,660,098,621,440</td>
<td>2,591,670,018.14</td>
<td>2,110,732.59</td>
<td>1,295,835,009.07</td>
<td>39,615,675.98</td>
<td>1,267,822,495.95</td>
<td>1,323,847,522.19</td>
</tr>
<tr>
<td>physical read total multi block requests</td>
<td>33</td>
<td>0.02</td>
<td>0.00</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>physical reads</td>
<td>568,835,681</td>
<td>316,352.60</td>
<td>257.65</td>
<td>158,176.30</td>
<td>4,835.91</td>
<td>154,756.80</td>
<td>161,595.81</td>
</tr>
<tr>
<td>physical reads cache</td>
<td>568,835,655</td>
<td>316,352.60</td>
<td>257.65</td>
<td>158,176.30</td>
<td>4,835.91</td>
<td>154,756.79</td>
<td>161,595.81</td>
</tr>
<tr>
<td>physical reads direct</td>
<td>1,399</td>
<td>0.78</td>
<td>0.00</td>
<td>0.39</td>
<td>0.32</td>
<td>0.16</td>
<td>0.62</td>
</tr>
<tr>
<td>physical reads direct (lob)</td>
<td>25</td>
<td>0.01</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>physical writes IO requests</td>
<td>159,218,208</td>
<td>88,547.71</td>
<td>72.12</td>
<td>44,273.88</td>
<td>4,800.54</td>
<td>40,879.37</td>
<td>47,668.35</td>
</tr>
<tr>
<td>physical write bytes</td>
<td>1,354,801,438,720</td>
<td>753,460,110.89</td>
<td>613,640.13</td>
<td>376,730,055.44</td>
<td>39,662,162.47</td>
<td>348,677,600.34</td>
<td>404,782,510.55</td>
</tr>
<tr>
<td>physical write total IO requests</td>
<td>164,220,913</td>
<td>91,329.92</td>
<td>74.38</td>
<td>45,664.96</td>
<td>4,882.23</td>
<td>42,212.70</td>
<td>49,117.22</td>
</tr>
<tr>
<td>physical write total bytes</td>
<td>1,481,358,051,328</td>
<td>823,843,380.78</td>
<td>670,962.35</td>
<td>411,921,690.39</td>
<td>40,842,421.29</td>
<td>383,041,737.36</td>
<td>440,801,643.42</td>
</tr>
</tbody>
</table>

Figure 13 Test 2 – RAC AWR system Statistics (Global)
The latency for the data file random read I/O was averaged at 0.58 ms. The minimum and maximum difference was 0.03 ms, as shown in Figure 14. Note that Log Writer latency was only 0.5 ms (reported as 'log file parallel write' event), benefiting from writing to VMAX persistent cache.

<table>
<thead>
<tr>
<th>Wait</th>
<th>Event</th>
<th>#</th>
<th>Class</th>
<th>Event</th>
<th>#</th>
<th>Wait Time</th>
<th>Summary Avg Wait Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>User I/O</td>
<td>db file sequential read</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB CPU</td>
<td></td>
<td></td>
<td>Avg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System I/O</td>
<td>log file parallel write</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Configuration</td>
<td>free buffer waits</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System I/O</td>
<td>db file parallel write</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>User I/O</td>
<td>read by other session</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Application</td>
<td>enq: TX - row lock contention</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Concurrency</td>
<td>latch: cache buffers chains</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Configuration</td>
<td>write complete waits</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other</td>
<td>LGWR wait for redo copy</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>User I/O</td>
<td>db file sequential read</td>
<td></td>
<td>Wait</td>
<td>Avg</td>
</tr>
</tbody>
</table>

Figure 14 Test 2 – RAC AWR Top Timed Events

We confirmed the numbers comparing them to the VMAX storage statistics in Unisphere, as shown in Figure 15. You can see 407K IOPS by the data_sg with 0.5 ms read latency and 0.4 ms write latency. You can also see 1K IOPS by the redo_sg, with 0.3 ms write latency.

Figure 15 Test 2 – Unisphere DATA and REDO storage groups performance

Test conclusion

This test showed a single V-Brick VMAX All-Flash performance capabilities under more 'normal' conditions with 50 percent read-hit rate. It achieved over 400K IOPS with sub-millisecond read and write latencies.

In addition, even as the workload was serviced from both the VMAX cache and the flash drives, Oracle AWR showed that the difference between the minimum and maximum data file read latency was only 0.03 ms, which is negligible.
Test 3: In-Memory OLTP tests (all reads are satisfied from database cache)

Test motivation
In recent years the cost of server memory has decreased dramatically and it became more common to see servers with hundreds of GB of RAM. As a result some databases benefit from being fully cached in the database buffer cache. Another trend is for customers using the Oracle 12c In-Memory database option. In either case, the outcome is that all database read requests are served from the database cache and the storage only experiences high write workloads, both for data updates as well as redo logs.

In this test case, we wanted to see the VMAX ability to support a write-intensive workload such as in-memory database. The goal was to keep an eye on redo log bandwidth and latency, as well as database writes latencies.

Test configuration
As in the previous use cases, we used a 2-node RAC to run SLOB 2.3 benchmark with a scale of 25 GB and 48 users for a total of 1.2 TB data set. SLOB was set for a 25 percent update rate, two threads per user process. However, we now increased the database buffer cache and set SLOB with “heavy” redo stress. In addition, we set the redo log ASM template to fine grain striping based on our best practice for high write-load. We ran the test for 30 minutes in a steady state (ignoring ramp-up and ramp-down periods).

Test results
Based on the 30-minute AWR report from the run, we can see that the redo logs were working at a rate of about 2.4 GB/sec, as shown in Figure 16.

<table>
<thead>
<tr>
<th>#</th>
<th>Logical Reads/s</th>
<th>Physical Reads/s</th>
<th>Physical Writes/s</th>
<th>Redo Size (k)'s</th>
<th>Block Changes/s</th>
<th>User Calls/s</th>
<th>Execs/s</th>
<th>Parses/s</th>
<th>Logons/s</th>
<th>Txns/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,221,487.49</td>
<td>1.04</td>
<td>96,432.78</td>
<td>1,272,444.91</td>
<td>592,778.78</td>
<td>1.45</td>
<td>15,479.64</td>
<td>132.33</td>
<td>0.31</td>
<td>3,836.97</td>
</tr>
<tr>
<td>2</td>
<td>1,152,119.28</td>
<td>0.77</td>
<td>90,545.86</td>
<td>1,206,728.99</td>
<td>559,702.66</td>
<td>1.44</td>
<td>14,666.29</td>
<td>95.20</td>
<td>0.31</td>
<td>3,642.96</td>
</tr>
<tr>
<td>Sum</td>
<td>2,373,606.77</td>
<td>1.81</td>
<td>186,978.64</td>
<td>2,479,173.90</td>
<td>1,152,481.44</td>
<td>2.89</td>
<td>30,145.92</td>
<td>227.53</td>
<td>0.62</td>
<td>7,479.94</td>
</tr>
<tr>
<td>Avg</td>
<td>1,186,803.38</td>
<td>0.91</td>
<td>93,489.32</td>
<td>1,239,586.95</td>
<td>576,240.72</td>
<td>1.44</td>
<td>15,072.96</td>
<td>113.77</td>
<td>0.31</td>
<td>3,739.97</td>
</tr>
<tr>
<td>Std</td>
<td>49,050.73</td>
<td>0.20</td>
<td>4,162.68</td>
<td>46,468.18</td>
<td>23,388.35</td>
<td>0.00</td>
<td>575.12</td>
<td>26.25</td>
<td>0.00</td>
<td>137.18</td>
</tr>
</tbody>
</table>

Figure 16 Test 3 – RAC AWR System Statistics – Per Second

Even at this rate the redo log latency was kept at 1 ms, as shown in Figure 17.

<table>
<thead>
<tr>
<th>#</th>
<th>Class</th>
<th>Event</th>
<th>Wait</th>
<th>Event</th>
<th>Wait Time</th>
<th>Summary Avg Wait Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Waits</td>
<td>%Timeouts</td>
<td>Total(s)</td>
<td>Avg(ms)</td>
</tr>
<tr>
<td>1</td>
<td>DB CPU</td>
<td></td>
<td>24,003.98</td>
<td>94.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>System I/O</td>
<td>log file parallel write</td>
<td>3,188,732</td>
<td>0.00</td>
<td>3,179.30</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Concurrency buffer busy waits</td>
<td>815,326</td>
<td>0.00</td>
<td>663.18</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration undo segment extension</td>
<td>32,635</td>
<td>74.42</td>
<td>294.61</td>
<td>9.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration log buffer space</td>
<td>10,018</td>
<td>0.00</td>
<td>198.49</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>reliable message</td>
<td>437,616</td>
<td>0.00</td>
<td>164.19</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration enq: HW - contention</td>
<td>6,622</td>
<td>0.00</td>
<td>134.35</td>
<td>20.29</td>
</tr>
<tr>
<td></td>
<td>System I/O</td>
<td>db file parallel write</td>
<td>1,212,491</td>
<td>0.00</td>
<td>97.28</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>enq: US - contention</td>
<td>844,526</td>
<td>0.00</td>
<td>43.96</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>LGWR wait for redo copy</td>
<td>2,057,159</td>
<td>0.00</td>
<td>23.07</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Figure 17 Test 3 – RAC AWR Top Timed Events
We confirmed the numbers comparing them to the VMAX storage statistics in Unisphere, as shown in Figure 18. You can see the data_sg receiving a 100 percent write workload and servicing it at 0.3 ms write latency. The redo_sg bandwidth is 2.4 GB/sec at 0.6 ms write latency.

![Figure 18 Test 3 – Unisphere DATA and REDO storage groups performance](image)

**Test conclusion**
This test showed that VMAX All Flash is well suitable to service in-memory databases, maintaining high-performance.

**Test 4: DSS test with focus on sequential read bandwidth**

**Test motivation**
This test case shows VMAX capabilities in servicing sequential reads, similar to those used by Oracle during data warehouse type queries. Unlike OLTP tests, the focus of this test was on bandwidth (GB/s). The higher the bandwidth, the faster the report execution.

**Test configuration**
We used the DBGEN toolkit to generate near 1 TB of data for the line item table, which was set with a primary partition by date, and a secondary hash partition. For sequential reads over large, partitioned, fact table, we used parallel query. In order to force full-table scan we used a hint in the SQL query and reviewed the execution plan. To ensure the query ran long enough to verify the storage array's ability to maintain steady-state bandwidth, we ran the query a few times in a loop without delay.

We ran the test for 30 minutes in a steady state (ignoring ramp-up and ramp-down periods).
Test results

Based on the 30-minute AWR report from the run, VMAX was able to deliver a consistent bandwidth of close to 11 GB/s.

The primary wait event was “direct path read”, indicating Oracle parallel query execution with large I/O. When Oracle issues large I/O (based on the “db_file_multiblock_read_count” parameter value), I/O latency was no longer the focus, and we looked for bandwidth instead.

The bandwidth during the run (“physical read total bytes”) was 10,932,040,316 bytes/sec, which is ~11 GB/s, as shown in Figure 19.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Second</th>
<th>per Trans</th>
<th>Average</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical read IO requests</td>
<td>144,532,591</td>
<td>83,619.38</td>
<td>675,385.94</td>
<td>41,809.69</td>
<td>530.96</td>
<td>41,434.23</td>
<td>42,185.15</td>
</tr>
<tr>
<td>physical read bytes</td>
<td>18,895,386,132,480</td>
<td>10,932,040,316.48</td>
<td>88,296,196,880.75</td>
<td>5,466,020,158.24</td>
<td>5,416,866,529.12</td>
<td>5,515,065,997.25</td>
<td></td>
</tr>
<tr>
<td>physical read total bytes</td>
<td>18,895,572,443,136</td>
<td>10,932,040,316.48</td>
<td>88,297,067,491.29</td>
<td>5,466,200,158.24</td>
<td>5,416,918,028.33</td>
<td>5,515,122,288.14</td>
<td></td>
</tr>
<tr>
<td>physical read total multi block requests</td>
<td>143,147,940</td>
<td>82,818.29</td>
<td>668,915.61</td>
<td>41,409.15</td>
<td>528.41</td>
<td>41,035.50</td>
<td>41,782.79</td>
</tr>
<tr>
<td>physical reads</td>
<td>2,306,565,690</td>
<td>1,334,464.42</td>
<td>10,778,344.35</td>
<td>667,232.21</td>
<td>8,476.26</td>
<td>661,238.59</td>
<td>673,225.83</td>
</tr>
<tr>
<td>physical reads cache</td>
<td>7,890</td>
<td>4.56</td>
<td>36.87</td>
<td>2.28</td>
<td>3.17</td>
<td>0.04</td>
<td>4.53</td>
</tr>
<tr>
<td>physical reads cache prefetch</td>
<td>6,803</td>
<td>3.94</td>
<td>31.79</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
<td></td>
</tr>
<tr>
<td>physical reads direct</td>
<td>2,306,557,800</td>
<td>1,334,459.85</td>
<td>10,778,307.48</td>
<td>667,229.93</td>
<td>8,473.08</td>
<td>661,238.55</td>
<td>673,221.30</td>
</tr>
<tr>
<td>physical write IO requests</td>
<td>1,380</td>
<td>0.80</td>
<td>6.45</td>
<td>0.40</td>
<td>0.15</td>
<td>0.28</td>
<td>0.50</td>
</tr>
<tr>
<td>physical write bytes</td>
<td>52,797,440</td>
<td>30,545.99</td>
<td>246,717.01</td>
<td>15,272.99</td>
<td>13,341.62</td>
<td>5,839.04</td>
<td>24,706.95</td>
</tr>
<tr>
<td>physical write total IO requests</td>
<td>3,387</td>
<td>1.96</td>
<td>15.83</td>
<td>0.98</td>
<td>0.41</td>
<td>0.69</td>
<td>1.27</td>
</tr>
<tr>
<td>physical write total bytes</td>
<td>89,334,784</td>
<td>51,684.68</td>
<td>417,452.26</td>
<td>25,842.34</td>
<td>14,212.56</td>
<td>15,792.55</td>
<td>35,892.14</td>
</tr>
<tr>
<td>physical write total multi block requests</td>
<td>100</td>
<td>0.06</td>
<td>0.47</td>
<td>0.03</td>
<td>0.01</td>
<td>0.02</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Figure 19 Test 4 – RAC AWR system Statistics (Global)

We confirmed the numbers comparing them to the VMAX storage statistics in Unisphere, as shown in Figure 20. You can see the tpch_sg is receiving executing 10,602 MB/sec. Also the average read I/O size shows as 128 KB, based on our best practice to set db_file_multiblock_read_count.

![Figure 20 Test 3 – Unisphere tpch_sg storage groups performance](image)

Test conclusion

This test showed that even a single V-Brick VMAX All Flash is able to achieve extremely high bandwidth suitable for Oracle reports, BI, and data warehousing (DW) workloads. Also, not only are flash drives capable of maintaining high small-block requests, they also have far better large-block bandwidth than conventional disks. In this test just 32 physical disks were able to sustain almost 11 GB/sec which is over 300 MB/sec per physical disk.
CONCLUSION

VMAX All Flash is well suited to run mission-critical and high-performance Oracle workloads while leveraging VMAX data services including data protection, data replications, compression, resiliency, and availability. VMAX All Flash provides outstanding performance in all workloads, OLTP low hit rate, OLTP moderate hit rate, in-memory and high bandwidth.

The tests described in this white paper demonstrate the ability of a single V-Brick VMAX 950FX with 32 x flash drives to deliver performance values as shown in Table 2.

Table 2 Test result summary

<table>
<thead>
<tr>
<th>TEST NUM</th>
<th>TEST CASE</th>
<th>DB IOPS</th>
<th>DB READ LATENCY (MS)</th>
<th>VMAX CACHE READ-HIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>OLTP with negligible VMAX cache read-hit benefits</td>
<td>275K</td>
<td>0.9</td>
<td>4%</td>
</tr>
<tr>
<td>Test 2</td>
<td>OLTP with moderate VMAX cache read-hit benefits</td>
<td>400K</td>
<td>0.6</td>
<td>50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST NUM</th>
<th>TEST CASE</th>
<th>REDO LOG WRITE BANDWIDTH (GB/S)</th>
<th>REDO LOG WRITE LATENCY (MS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 3</td>
<td>OLTP in-memory test (data is fully cached in the database)</td>
<td>2.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Test 4</td>
<td>DSS (sequential read test)</td>
<td>~11</td>
<td></td>
</tr>
</tbody>
</table>

REFERENCES

The VMAX All Flash Storage Family: A Detailed Overview
VMAX All Flash with the Adaptive Compression Engine
Dell EMC VxBlock System 740 and Vblock System 740
Oracle Database Backup and Recovery with VMAX3
ProtectPoint 3.1 File System Agent with VMAX – Backup & Recovery Best Practice for Oracle on ASM
Appendix I – Oracle Advanced Row Compression


Oracle Advanced Row Compression is part of Oracle Advanced Compression Option (ACO), a licensed feature that works within the database blocks to eliminate repeating data values and provide efficient compression for OLTP and DSS databases. Advanced Row Compression minimizes the overhead of write operations on compressed data, making it suitable for transactional and OLTP environments as well as data warehouses, extending the benefits of compression to all application workloads.

The benefits of Advanced Row Compression go beyond on-disk storage savings. One significant advantage is Oracle’s ability to read compressed blocks (data and indexes) directly, in memory, without uncompressing the blocks. This helps improve performance due to the reduction in I/O, and the reduction in system calls related to the I/O operations. In addition, the buffer cache becomes more efficient by storing more data without having to add memory.

Minimal performance overhead

Oracle Database compresses blocks in batch mode rather than compressing data every time a write operation takes place. A newly initialized block remains uncompressed until data in the block reaches an internally controlled threshold. When a transaction causes the data in the block to reach this threshold, all contents of the block are compressed. Subsequently, as more data is added to the block and the threshold is again reached, the entire block is recompressed to achieve the highest level of compression. This process repeats until Oracle determines that the block can no longer benefit from further compression. Only the transaction that performs the compression of the block will experience the slight compression overhead—the majority of DML transactions on compressed blocks will have the same performance as they would with uncompressed blocks.

Background compression

Often database blocks are compressed as a result of the block fills above a specific threshold in an insert or update operation. Alternatively, background compression can be triggered by an operation such as a table move (redefinition), or by using ADO (Automatic Data Optimization). ADO provides activity-based policies that can trigger a background move and compression of the Oracle data.
Appendix II – Oracle AWR analysis of storage-related metrics

AWR reports should be collected over peak workload periods to identify any potential bottlenecks. AWR averages all metrics over the duration of the report; therefore a 24-hour report is generally not going to be useful. A good AWR report should be produced for a short period of time, for example, 15 minutes, 30 minutes, or 1 hour, where the workload is stable and heavy.

When using Oracle RAC, AWR reports can be produced for each instance separately, and for the cluster as a whole. Dell EMC recommends that you collect both, because they each include slightly different information and present it differently. The Instance AWR metrics only represent the workload on that specific host. The RAC AWR metrics represent the workload from the entire cluster. In the following examples, we show both types to demonstrate different aspects of performance metrics.

AWR Load Profile

The Load Profile area in an Instance AWR report includes “Physical reads”, “Physical writes”, and “Logical reads” metrics, as shown in Figure 21. The units for these metrics are database blocks. Block units cannot be directly translated to I/O metrics. Blocks do, however, provide an indication of the type of workload the database is doing, such as reads versus writes ratio, and how many of the reads are satisfied from the buffer cache (logical reads). Typically, you should expect a much higher number of the reads in an OLTP workload to be logical reads. In the following example, they almost match the physical reads. This is because, in our tests, we limited the size of the buffer cache to force reads to storage. Note that in an Oracle 12c AWR the load profile also provides actual I/O metrics (not just block metrics) for the instance.

<table>
<thead>
<tr>
<th></th>
<th>Per Second</th>
<th>Per Transaction</th>
<th>Per Exec</th>
<th>Per Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Time(s):</td>
<td>95.7</td>
<td>0.2</td>
<td>0.04</td>
<td>54.57</td>
</tr>
<tr>
<td>DB CPU(s):</td>
<td>7.1</td>
<td>0.0</td>
<td>0.00</td>
<td>4.04</td>
</tr>
<tr>
<td>Background CPU(s):</td>
<td>1.3</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Redo size (bytes):</td>
<td>34,513,768.1</td>
<td>55,076.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical read (blocks)</td>
<td>175,519.2</td>
<td>280.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block changes:</td>
<td>85,056.4</td>
<td>135.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical read (blocks)</td>
<td>161,595.8</td>
<td>257.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical write (blocks)</td>
<td>49,411.9</td>
<td>78.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read IO requests:</td>
<td>161,595.7</td>
<td>257.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write IO requests:</td>
<td>47,668.4</td>
<td>76.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read IO (MB):</td>
<td>1,262.5</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write IO (MB):</td>
<td>386.0</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IM scan rows:</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 21 Load Profile table in Instance AWR report

AWR Top Foreground Events

Ideally, the database is waiting most of the time for CPU and I/O. That is an indication that the system is working at its physical limit. Ensure that the “db file sequential read” field of the Instance AWR report (which actually means random-read) has an average wait time appropriate to the storage type and application needs. For example, 0.57 ms I/O latency, as shown in Figure 22 from the Instance AWR report for Test Case 2.

<table>
<thead>
<tr>
<th>Event</th>
<th>Waits</th>
<th>Total Wait Time (sec)</th>
<th>Wait Avg(ms)</th>
<th>% DB time</th>
<th>Wait Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>db file sequential read</td>
<td>2.9E+08</td>
<td>165.9</td>
<td>0.57</td>
<td>96.4</td>
<td>User I/O</td>
</tr>
<tr>
<td>DB CPU</td>
<td></td>
<td>12.7</td>
<td>3.10</td>
<td>7.4</td>
<td></td>
</tr>
<tr>
<td>free buffer waits</td>
<td>155,815</td>
<td>482.7</td>
<td></td>
<td></td>
<td>Configuration</td>
</tr>
</tbody>
</table>

Figure 22 Instance AWR Top Foreground Events
You can view similar information at a cluster-wide level in the RAC AWR report, as shown in Figure 23. If the wait time is considered too high, use the Linux "iostat" command to investigate if the delays are host, storage, or connectivity related as explained in a later section. Note also that "log file parallel write" shows the write latency of the redo log writer.

<table>
<thead>
<tr>
<th>ID</th>
<th>Class</th>
<th>Event</th>
<th>Wait</th>
<th>Event</th>
<th>Wait Time</th>
<th>Summary Avg Wait Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total(s)</td>
<td>Avg(ms)</td>
<td>%DB time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Avg</td>
</tr>
<tr>
<td>1</td>
<td>User I/O</td>
<td>db file sequential read</td>
<td>568,877,798</td>
<td>0.00</td>
<td>322,609.74</td>
<td>0.58</td>
</tr>
<tr>
<td>2</td>
<td>System I/O</td>
<td>log file parallel write</td>
<td>2,515,637</td>
<td>0.00</td>
<td>1,255.68</td>
<td>0.50</td>
</tr>
<tr>
<td>3</td>
<td>Configuration</td>
<td>free buffer waits</td>
<td>156,019</td>
<td>0.00</td>
<td>483.47</td>
<td>3.10</td>
</tr>
<tr>
<td>4</td>
<td>System I/O</td>
<td>db file parallel write</td>
<td>5,117,109</td>
<td>0.00</td>
<td>243.04</td>
<td>0.05</td>
</tr>
<tr>
<td>5</td>
<td>User I/O</td>
<td>read by other session</td>
<td>321,751</td>
<td>0.00</td>
<td>176.20</td>
<td>0.55</td>
</tr>
<tr>
<td>6</td>
<td>Application</td>
<td>enq: TX - row lock contention</td>
<td>1,174</td>
<td>0.00</td>
<td>23.58</td>
<td>20.07</td>
</tr>
<tr>
<td>7</td>
<td>Concurrency</td>
<td>latch: cache buffers chains</td>
<td>195,314</td>
<td>0.00</td>
<td>16.44</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>Configuration</td>
<td>write complete waits</td>
<td>443</td>
<td>0.00</td>
<td>7.39</td>
<td>16.68</td>
</tr>
<tr>
<td>9</td>
<td>Other</td>
<td>LGWR wait for redo copy</td>
<td>387,333</td>
<td>0.00</td>
<td>7.25</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Figure 23 RAC AWR top foreground wait events

AWR data file read and write I/O metrics
To find I/O-related metrics in the AWR report, look for the "...total I/O requests" and "...total bytes" metrics, as shown in Figure 24.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Second</th>
<th>per Trans</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical read total IO requests</td>
<td>568,846,410</td>
<td>316,358,58</td>
<td>257.65</td>
</tr>
<tr>
<td>physical read total bytes</td>
<td>4,660,098,621,440</td>
<td>2,591,670,018.14</td>
<td>2,110,732.59</td>
</tr>
<tr>
<td>physical write total IO requests</td>
<td>159,218,208</td>
<td>88,547,71</td>
<td>72.12</td>
</tr>
<tr>
<td>physical write bytes</td>
<td>1,354,801,438,720</td>
<td>753,460,110.89</td>
<td>613,640.13</td>
</tr>
<tr>
<td>physical write total IO requests</td>
<td>1,481,358,051,328</td>
<td>823,843,380.78</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 24 RAC AWR data file read and write I/O metrics

To find IOPS, add the “physical read total IO requests” and “physical write total IO requests” metrics using the “per second” column. For example, using the RAC AWR report in Figure 24: 316,358 + 91,329 = 407,687 IOPS.

To find bandwidth, use the “...total bytes” metrics using the "per second" column. For example, using the same report: 2,591,670,018 + 823,843,380 = 3,415,513,398 bytes/sec = 3.2 GB/s.

AWR and redo logs
Redo logs are key to Oracle database resiliency and performance. Oracle write size to the logs ranges from 512 bytes up to 1MB. Once a redo log file fills up, Oracle switches to the next log and starts archiving the previous log.

Redo log switches
Configure redo logs size so they only switch a small number of times per hour. Ensure there are enough log files so that they never wait for the archiving processes to complete.

Log switch count can be found in the Instance AWR report, as shown in Figure 25.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>log switches (derived)</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 25 RAC AWR log file switch per hour metrics
Redo log file sync

Log file sync time appears in the Instance AWR report, as shown in Figure 26. The report indicates log write wait events due to commits. If a large portion of database time is spent on log file sync (% DB time) you should investigate the cause. Such delays are not always storage related. They could be caused by high CPU utilization, low memory, excessive commits, and so on. Correct issues such as memory or CPU starvation.

<table>
<thead>
<tr>
<th>Event</th>
<th>Waits</th>
<th>%Time -outs</th>
<th>Total Wait Time(s)</th>
<th>Avg wait (ms)</th>
<th>Waits /txn</th>
<th>% DB time</th>
</tr>
</thead>
<tbody>
<tr>
<td>log file sync</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0.26</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 26 Instance AWR log file sync

If you suspect that storage contributes significantly to high log file sync latencies, check also 'log file parallel write' latency, which relates directly to the log writer process. Also use the "iostat" command, but focus on a log device instead of a data file device. More often than not, log performance issues have to do with the redo log files size, number of files, or queuing and paths issues. Ensure the logs have enough I/O paths, LUNs, and storage connectivity.

With VMAX, due to the large cache, redo log writes generally feature very good write latencies (typically < 1 ms).

iostat analysis

If read I/O latencies are higher than expected it is a good practice to run the Linux command: `iostat -xtz <interval> <iterations>` and capture the output to a file. After a few iterations stop the command and inspect the file. Ignore the first interval and focus on interval 2 and higher. Because the file can be quite large it is a good idea to find the pseudo name of one of the Oracle data files and focus on that single device6, as shown in Figure 27. Note how many I/Os are queued to the device (avgqu-sz), the amount of time I/Os take to be serviced, including queue time (await), and the amount of time I/Os take to be serviced once they leave the queue (svctm). If the await is long but svctm is short, the host suffers from queuing issue, and requires more LUNs (more I/O queues), and perhaps more paths to the storage. If svctm is high, it could point to a storage performance bottleneck.

<table>
<thead>
<tr>
<th>Device:</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>r/s</th>
<th>w/s</th>
<th>rsec/s</th>
<th>wsec/s</th>
<th>avgrq-sz</th>
<th>avgqu-sz</th>
<th>await</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>dm-395</td>
<td>0.00</td>
<td>0.00</td>
<td>2145.33</td>
<td>937.33</td>
<td>16.76</td>
<td>7.51</td>
<td>16.13</td>
<td>2.99</td>
<td>0.97</td>
<td>0.28</td>
<td>87.77</td>
</tr>
</tbody>
</table>

Figure 27 iostat output analysis

---

6 With host striping, such as Oracle ASM provides, the workload should be distributed across all devices evenly.
Table 3 summarizes iostat metrics and provides advice on how to use them.

Table 3: Linux iostat with flags xtz - metrics summary

<table>
<thead>
<tr>
<th>METRIC</th>
<th>DESCRIPTION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>The device name as listed in the /dev directory</td>
<td>When multipathing is used each device has a pseudo name (such as dm-xxx, or emcpowerxxx) and each path has a device name (such as /dev/sdxxx). Use the pseudo name to inspect the aggregated metrics across all paths.</td>
</tr>
<tr>
<td>r/s, w/s</td>
<td>The number of read or write requests that were issued to the device per second.</td>
<td>r/s + w/s provides the host IOPS requests for the device. The ratio between these metrics provides read to write ratio.</td>
</tr>
<tr>
<td>rsec/s, wsec/s</td>
<td>The number of sectors read or written from/to the device per second (512 bytes per sector)</td>
<td>Can be used to calculate KB/sec read or written by using the formula: &lt;rsec/s&gt; * 512 / 1024, to inspect KB/sec read, for example. Alternatively iostat such as –d provide KB/sec metrics directly. Note that by dividing the average KB/sec writes by w/s the average write size can be found, and in a similar way the average read size can be found by dividing KB/sec read by r/s.</td>
</tr>
<tr>
<td>avgrq-sz</td>
<td>The average size (in sectors) of the requests that were issued to the device</td>
<td></td>
</tr>
<tr>
<td>avgqu-sz</td>
<td>The average queue length of the requests that were issued to the device</td>
<td>The number of requests are queued to the device. With VMAX storage each host-device has many storage devices aggregated behind it but the queue size is still limited (often tunable). If not enough devices are configured, the queue on each device may be too large. Large queues can slow the I/O service time.</td>
</tr>
<tr>
<td>await</td>
<td>The average time (in milliseconds) for I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them</td>
<td>If &lt;await&gt;, which includes the queuing time, is much larger than &lt;svctm&gt;, it could point to a host queuing issue. Consider adding more host devices.</td>
</tr>
<tr>
<td>svctm</td>
<td>The average service time (in milliseconds) for I/O requests that were issued to the device</td>
<td>For active devices, the await time should be within the expected service level time (for example &lt;=1 ms for flash storage, ~6 ms for 15k rpm drives, etc.</td>
</tr>
</tbody>
</table>

Host CPU analysis

Another area in the AWR report to monitor is CPU utilization, as shown in Figure 28. High CPU utilization (for example, above 70 percent) indicates that the system is at its limit, and CPU utilization issues may start causing delays. You can see a more detailed view of the CPU core behavior by using the Linux command: mpstat –P ALL <interval>, pipe it to a file, and inspect it later. The workload should be balanced across all cores.

<table>
<thead>
<tr>
<th>CPUs</th>
<th>Cores</th>
<th>Sockets</th>
<th>Load Average Begin</th>
<th>Load Average End</th>
<th>%User</th>
<th>%System</th>
<th>%WIO</th>
<th>%Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>0.08</td>
<td>42.43</td>
<td>13.5</td>
<td>6.3</td>
<td>69.7</td>
<td>79.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%Total CPU</th>
<th>%Busy CPU</th>
<th>%DB time waiting for CPU (Resource Manager)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.3</td>
<td>102.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>