Additional notes

Enterprise applications

NetApp
January 12, 2026

Table of Contents

Additional notes

Performance optimization and benchmarking

Accurate testing of database storage performance is an extremely complicated subject. It requires an understanding of the following issues:

- IOPS and throughput
- The difference between foreground and background I/O operations
- The effect of latency upon the database
- Numerous OS and network settings that also affect storage performance

In addition, there are nonstorage databases tasks to consider. There is a point at which optimizing storage performance yields no useful benefits because storage performance is no longer a limiting factor for performance.

A majority of database customers now select all-flash arrays, which creates some additional considerations. For example, consider performance testing on a two-node AFF A900 system:

- With a 80/20 read/write ratio, two A900 nodes can deliver over 1M random database IOPS before latency even crosses the 150µs mark. This is so far beyond the current performance demands of most databases that it is difficult to predict the expected improvement. Storage would be largely erased as a bottleneck.
- Network bandwidth is an increasingly common source of performance limitations. For example, spinning disk solutions are often bottlenecks for database performance because the I/O latency is very high. When latency limitations are removed by an all-flash array, the barrier frequently shifts to the network. This is especially notable with virtualized environments and blade systems where the true network connectivity is difficult to visualize. This can complicate performance testing if the storage system itself cannot be fully utilized due to bandwidth limitations.
- Comparing the performance of an all-flash array with an array containing spinning disks is generally not possible because of the dramatically improved latency of all-flash arrays. Test results are typically not meaningful.
- Comparing peak IOPS performance with an all-flash array is frequently not a useful test because databases are not limited by storage I/O. For example, assume one array can sustain 500K random IOPS, whereas another can sustain 300K. The difference is irrelevant in the real world if a database is spending 99% of its time on CPU processing. The workloads never utilize the full capabilities of the storage array. In contrast, peak IOPS capabilities might be critical in a consolidation platform in which the storage array is expected to be loaded to its peak capabilities.
- Always consider latency as well as IOPS in any storage test. Many storage arrays in the market make claims of extreme levels of IOPS, but the latency renders those IOPS useless at such levels. The typical target with all-flash arrays is the 1ms mark. A better approach to testing is not to measure the maximum possible IOPS, but to determine how many IOPS a storage array can sustain before average latency is greater than 1ms.

Oracle Automatic Workload Repository and benchmarking

The gold standard for Oracle performance comparisons is an Oracle Automatic Workload Repository (AWR) report.

There are multiple types of AWR reports. From a storage point of view, a report generated by running the

`awrrpt.sql` command is the most comprehensive and valuable because it targets a specific database instance and includes some detailed histograms that break down storage I/O events based on latency.

Comparing two performance arrays ideally involves running the same workload on each array and producing an AWR report that precisely targets the workload. In the case of a very long-running workload, a single AWR report with an elapsed time that encompasses the start and stop time can be used, but it is preferable to break out the AWR data as multiple reports. For example, if a batch job ran from midnight to 6 a.m., create a series of one-hour AWR reports from midnight–1 a.m., 1 a.m.–2 a.m., and so on.

In other cases, a very short query should be optimized. The best option is an AWR report based on an AWR snapshot created when the query begins and a second AWR snapshot created when the query ends. The database server should be otherwise quiet to minimize the background activity that would obscure the activity of the query under analysis.

Where AWR reports are not available, Oracle statspack reports are a good alternative. They contain most of the same I/O statistics as an AWR report.

Oracle AWR and troubleshooting

An AWR report is also the most important tool for analyzing a performance problem.

As with benchmarking, performance troubleshooting requires that you precisely measure a particular workload. When possible, provide AWR data when reporting a performance problem to the NetApp support center or when working with a NetApp or partner account team about a new solution.

When providing AWR data, consider the following requirements:

- Run the `awrrpt.sql` command to generate the report. The output can be either text or HTML.
- If Oracle Real Application Clusters (RACs) are used, generate AWR reports for each instance in the cluster.
- Target the specific time the problem existed. The maximum acceptable elapsed time of an AWR report is generally one hour. If a problem persists for multiple hours or involves a multihour operation such as a batch job, provide multiple one-hour AWR reports that cover the entire period to be analyzed.
- If possible, adjust the AWR snapshot interval to 15 minutes. This setting allows a more detailed analysis to be performed. This also requires additional executions of `awrrpt.sql` to provide a report for each 15-minute interval.
- If the problem is a very short running query, provide an AWR report based on an AWR snapshot created when the operation begins and a second AWR snapshot created when the operation ends. The database server should be otherwise quiet to minimize the background activity that would obscure the activity of the operation under analysis.
- If a performance problem is reported at certain times but not others, provide additional AWR data that demonstrates good performance for comparison.

calibrate_io

The `calibrate_io` command should never be used to test, compare, or benchmark storage systems. As stated in the Oracle documentation, this procedure calibrates the I/O capabilities of storage.

Calibration is not the same as benchmarking. The purpose of this command is to issue I/O to help calibrate database operations and improve their efficiency by optimizing the level of I/O issued to the host. Because the type of I/O performed by the `calibrate_io` operation does not represent actual database user I/O, the

results are not predictable and are frequently not even reproducible.

SLOB2

SLOB2, the Silly Little Oracle Benchmark, has become the preferred tool for evaluating database performance. It was developed by Kevin Closson and is available at https://kevinclosson.net/slob/. It takes minutes to install and configure, and it uses an actual Oracle database to generate I/O patterns on a user-definable tablespace. It is one of the few testing options available that can saturate an all-flash array with I/O. It is also useful for generating much lower levels of I/O to simulate storage workloads that are low IOPS but latency sensitive.

Swingbench

Swingbench can be useful for testing database performance, but it is extremely difficult to use Swingbench in a way that stresses storage. NetApp has not seen any tests from Swingbench that yielded enough I/O to be a significant load on any AFF array. In limited cases, the Order Entry Test (OET) can be used to evaluate storage from a latency point of view. This could be useful in situations where a database has a known latency dependency for particular queries. Care must be taken to make sure that the host and network are properly configured to realize the latency potentials of an all-flash array.

HammerDB

HammerDB is a database testing tool that simulates TPC-C and TPC-H benchmarks, among others. It can take a lot of time to construct a sufficiently large data set to properly execute a test, but it can be an effective tool for evaluating performance for OLTP and data warehouse applications.

Orion

The Oracle Orion tool was commonly used with Oracle 9, but it has not been maintained to ensure compatibility with changes in various host operation systems. It is rarely used with Oracle 10 or Oracle 11 due to incompatibilities with OS and storage configuration.

Oracle rewrote the tool, and it is installed by default with Oracle 12c. Although this product has been improved and uses many of the same calls that a real Oracle database uses, it does not use precisely the same code path or I/O behavior used by Oracle. For example, most Oracle I/Os are performed synchronously, meaning the database halts until the I/O is complete as the I/O operation completes in the foreground. Simply flooding a storage system with random I/Os is not a reproduction of real Oracle I/O and does not offer a direct method of comparing storage arrays or measuring the effect of configuration changes.

That said, there are some use cases for Orion, such as general measurement of the maximum possible performance of a particular host-network-storage configuration, or to gauge the health of a storage system. With careful testing, usable Orion tests could be devised to compare storage arrays or evaluate the effect of a configuration change so long as the parameters include consideration of IOPS, throughput, and latency and attempt to faithfully replicate a realistic workload.

Stale NFSv3 locks

If an Oracle database server crashes, it might have problems with stale NFS locks upon restart. This problem is avoidable by paying careful attention to the configuration of name resolution on the server.

This problem arises because creating a lock and clearing a lock use two slightly different methods of name resolution. Two processes are involved, the Network Lock Manager (NLM) and the NFS client. The NLM uses

`uname n` to determine the host name, while the `rpc.statd` process uses `gethostbyname()`. These host names must match for the OS to properly clear stale locks. For example, the host might be looking for locks owned by `dbserver5`, but the locks were registered by the host as `dbserver5.mydomain.org`. If `gethostbyname()` does not return the same value as `uname -a`, then the lock release process did not succeed.

The following sample script verifies whether name resolution is fully consistent:

```
#! /usr/bin/perl
$uname=`uname -n`;
chomp($uname);
($name, $aliases, $addrtype, $length, @addrs) = gethostbyname $uname;
print "uname -n yields: $uname\n";
print "gethostbyname yields: $name\n";
```

If `gethostbyname` does not match `uname`, stale locks are likely. For example, this result reveals a potential problem:

```
uname -n yields: dbserver5
gethostbyname yields: dbserver5.mydomain.org
```

The solution is usually found by changing the order in which hosts appear in `/etc/hosts`. For example, assume that the hosts file includes this entry:

```
10.156.110.201  dbserver5.mydomain.org dbserver5 loghost
```

To resolve this issue, change the order in which the fully qualified domain name and the short host name appear:

```
10.156.110.201  dbserver5 dbserver5.mydomain.org loghost
```

`gethostbyname()` now returns the short `dbserver5` host name, which matches the output of `uname`. Locks are thus cleared automatically after a server crash.

WAFL alignment verification

Correct WAFL alignment is critical for good performance. Although ONTAP manages blocks in 4KB units, this fact does not mean that ONTAP performs all operations in 4KB units. In fact, ONTAP supports block operations of different sizes, but the underlying accounting is managed by WAFL in 4KB units.

The term “alignment” refers to how Oracle I/O corresponds to these 4KB units. Optimum performance requires an Oracle 8KB block to reside on two 4KB WAFL physical blocks on a drive. If a block is offset by 2KB, this block resides on half of one 4KB block, a separate full 4KB block, and then half of a third 4KB block. This arrangement causes performance degradation.

Alignment is not a concern with NAS file systems. Oracle datafiles are aligned to the start of the file based on the size of the Oracle block. Therefore, block sizes of 8KB, 16KB, and 32KB are always aligned. All block operations are offset from the start of the file in units of 4 kilobytes.

LUNs, in contrast, generally contain some kind of driver header or file system metadata at their start that creates an offset. Alignment is rarely a problem in modern OSs because these OSs are designed for physical drives that might use a native 4KB sector, which also requires I/O to be aligned to 4KB boundaries for optimum performance.

There are, however, some exceptions. A database might have been migrated from an older OS that was not optimized for 4KB I/O, or user error during partition creation might have led to an offset that is not in units of 4KB in size.

The following examples are Linux-specific, but the procedure can be adapted for any OS.

Aligned

The following example shows an alignment check on a single LUN with a single partition.

First, create the partition that uses all partitions available on the drive.

```
[root@host0 iscsi]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel
Building a new DOS disklabel with disk identifier 0xb97f94c1.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
The device presents a logical sector size that is smaller than
the physical sector size. Aligning to a physical sector (or optimal
I/O) size boundary is recommended, or performance may be impacted.
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-10240, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-10240, default 10240):
Using default value 10240
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
[root@host0 iscsi]#
```

The alignment can be checked mathematically with the following command:

```
[root@host0 iscsi]# fdisk -u -l /dev/sdb
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
64 heads, 32 sectors/track, 10240 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 65536 bytes
Disk identifier: 0xb97f94c1
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1              32    20971519    10485744   83  Linux
```

The output shows that the units are 512 bytes, and the start of the partition is 32 units. This is a total of 32 x 512 = 16,834 bytes, which is a whole multiple of 4KB WAFL blocks. This partition is correctly aligned.

To verify correct alignment, complete the following steps:

1. Identify the universally unique identifier (UUID) of the LUN.

```
FAS8040SAP::> lun show -v /vol/jfs_luns/lun0
              Vserver Name: jfs
                  LUN UUID: ed95d953-1560-4f74-9006-85b352f58fcd
                    Mapped: mapped`                `
```

2. Enter the node shell on the ONTAP controller.

```
FAS8040SAP::> node run -node FAS8040SAP-02
Type 'exit' or 'Ctrl-D' to return to the CLI
FAS8040SAP-02> set advanced
set not found.  Type '?' for a list of commands
FAS8040SAP-02> priv set advanced
Warning: These advanced commands are potentially dangerous; use
         them only when directed to do so by NetApp
         personnel.
```

3. Start statistical collections on the target UUID identified in the first step.

```
FAS8040SAP-02*> stats start lun:ed95d953-1560-4f74-9006-85b352f58fcd
Stats identifier name is 'Ind0xffffff08b9536188'
FAS8040SAP-02*>
```

4. Perform some I/O. It is important to use the `iflag` argument to make sure that I/O is synchronous and not buffered.

Be very careful with this command. Reversing the `if` and `of` arguments destroys data.

```
[root@host0 iscsi]# dd if=/dev/sdb1 of=/dev/null iflag=dsync count=1000
bs=4096
1000+0 records in
1000+0 records out
4096000 bytes (4.1 MB) copied, 0.0186706 s, 219 MB/s
```

5. Stop the stats and view the alignment histogram. All I/O should be in the `.0` bucket, which indicates I/O that is aligned to a 4KB block boundary.

```
FAS8040SAP-02*> stats stop
StatisticsID: Ind0xffffff08b9536188
lun:ed95d953-1560-4f74-9006-85b352f58fcd:instance_uuid:ed95d953-1560-
4f74-9006-85b352f58fcd
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.0:186%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.1:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.2:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.3:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.4:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.5:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.6:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.7:0%
```

Misaligned

The following example shows misaligned I/O:

1. Create a partition that does not align to a 4KB boundary. This is not default behavior on modern OSs.

```
[root@host0 iscsi]# fdisk -u /dev/sdb
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (32-20971519, default 32): 33
Last sector, +sectors or +size{K,M,G} (33-20971519, default 20971519):
Using default value 20971519
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

2. The partition has been created with a 33-sector offset instead of the default 32. Repeat the procedure outlined in Aligned. The histogram appears as follows:

```
FAS8040SAP-02*> stats stop
StatisticsID: Ind0xffffff0468242e78
lun:ed95d953-1560-4f74-9006-85b352f58fcd:instance_uuid:ed95d953-1560-
4f74-9006-85b352f58fcd
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.0:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.1:136%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.2:4%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.3:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.4:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.5:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.6:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_align_histo.7:0%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:read_partial_blocks:31%
```

The misalignment is clear. The I/O mostly falls into the* *`.1` bucket, which matches the expected offset. When the partition was created, it was moved 512 bytes further into the device than the optimized default, which means that the histogram is offset by 512 bytes.

Additionally, the `read_partial_blocks` statistic is nonzero, which means I/O was performed that did not fill up an entire 4KB block.

Redo logging

The procedures explained here are applicable to datafiles. Oracle redo logs and archive logs have different I/O patterns. For example, redo logging is a circular overwrite of a single file. If the default 512-byte block size is used, the write statistics look something like this:

```
FAS8040SAP-02*> stats stop
StatisticsID: Ind0xffffff0468242e78
lun:ed95d953-1560-4f74-9006-85b352f58fcd:instance_uuid:ed95d953-1560-4f74-
9006-85b352f58fcd
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.0:12%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.1:8%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.2:4%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.3:10%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.4:13%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.5:6%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.6:8%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_align_histo.7:10%
lun:ed95d953-1560-4f74-9006-85b352f58fcd:write_partial_blocks:85%
```

The I/O would be distributed across all histogram buckets, but this is not a performance concern. Extremely high redo-logging rates might, however, benefit from the use of a 4KB block size. In this case, it is desirable to

make sure that the redo-logging LUNs are properly aligned. However, this is not as critical to good performance as datafile alignment.

Copyright information

Trademark information