OCI File Storage Performance Characteristics

August 2023, version 2.0

Revision History

The following revisions have been made to this document since its initial publication.

DATE	REVISION
August 2023	Updated the title, contents, and test results
May 2022	Reviewed and verified as current
August 2021	Updated template and made minor edits
June 2019	Made minor updates and additions
September 2018	Initial publication

Table of Contents

Purpose

Oracle Cloud Infrastructure (OCI) File Storage is a versatile file storage service that can be used for a variety of purposes, including general-purpose file sharing and performance-intensive workloads such as media processing, artificial intelligence, and machine learning. Several performance characteristics are important to understand when deploying performance-intensive workloads in File Storage. This paper provides best practices to achieve optimal performance levels with File Storage, describes performance characteristics and expectations of File Storage, and demonstrates the results of various performance benchmarks conducted.

Architecture Overview

OCI File Storage is a fully managed, scalable, and secure file storage service that can be accessed by thousands of compute instances at the same time. You can scale up to exabytes without the need to pre-provision storage. It uses Network File System (NFSv3) protocol to provide Portable Operating System Interface (POSIX)-compliant file system access. This enables users and applications to access the file system as if it is a locally attached UNIX file system.

File System

File systems provide a single namespace to access your data. File Storage is a distributed file system with data spread across a large numbers of storage nodes in an availability domain. The file systems are exported through one or many mount targets.

Mount Targets

Mount targets are the network endpoints that you use to access the highly distributed file system. They have an IP address that allows compute instances to connect to the file system. The mapping of file systems to mount targets is flexible: one to one, one to many, or many to one, as shown in the following diagrams.

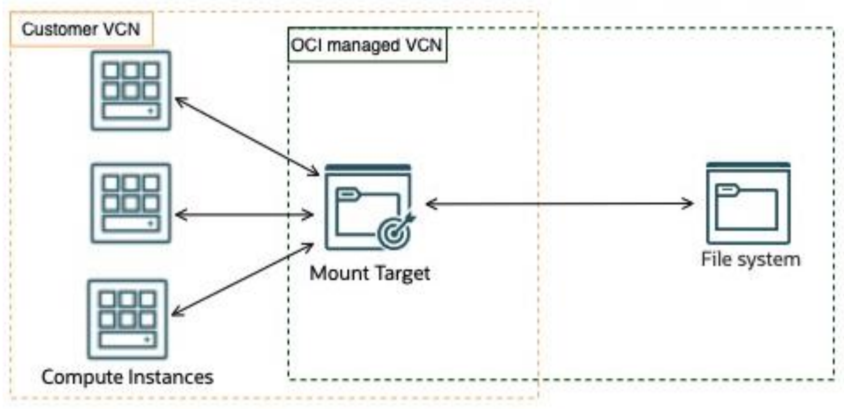


Figure 1. File System Access Through a Mount Target—One File System and One Mount Target

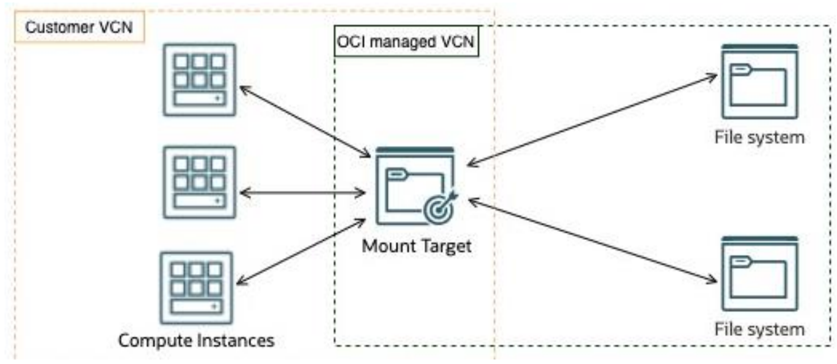


Figure 2. File System Access Through a Mount Target—Multiple File System and One Mount Target

Performance Characteristics

This section describes the performance-related aspects of File Storage in detail to help you to optimize and ensure that your workloads can perform at their best.

Throughput and IOPS at Scale: Horizontal Scaling of Mount Targets

The File Storage file system is designed to scale out without any performance limits. However, access to the file system requires a mount target in between compute instances and the file system. Mount targets are provisioned with the following capacity.

Table 1. Provisioned IOPS/Throughput for Mount Targets

TYPE	IOPS	BANDWIDTH	NUMBER OF CLIENTS	NUMBER OF CLIENTS (TLS)	COMMENT
Regular	50,000	1GB/s	100,000	64	These numbers are provisioned capacity and not guaranteed SLA.

Unlike block volumes, I/O to a file system includes not only read and write operations, but also NFSv3 metadata operations such as CREATE, DELETE, GETATTR, SETATTR, MKDIR, RMDIR, ACCESS, and RENAME. These operations might involve one or many I/O operations at the File Storage backend. For example, a 32 KiB read is considered two I/O operations at the storage backend, while a GETATTR operation is just one I/O operation at the backend.

NFS Operations to IOPS Mapping

The following table maps NFSv3 operations to IOPS.

Table 2. NFS Operations to IOPS Mapping

NFS OPERATION	IOPS	COMMENT
ACCESS, COMMIT, FSINFO, FSSTAT, GETATTR, READLINK, and PATHCONF	1	Lightweight NFS operations
LOOKUP, READDIR, and SETATTR	2	Not applicable
CREATE, LINK, MKDIR, MKNOD, READDIRPLUS, RMDIR, and SYMLINK	4	Not applicable
REMOVE and RENAME	8	Not applicable

NFS OPERATION	IOPS	COMMENT
All NLM Operations	4	Except FREE_ALL
NLM FREE_ALL	8	Not applicable
READ 32 KiB	2	For large I/O size > 32 KiB, add 1 IOPS for each 32 KiB size increments. For example: read size > 32 KiB and size <= 64 KiB is 3 IOPS.
WRITE 32 KiB	4	For large I/O size > 32 KiB, add 2 IOPS for each 32 KiB size increments.

To achieve higher IOPS and throughput for your file system, mount targets can be scaled horizontally to achieve almost linear performance improvements. The following diagram is using two mount targets to double the file system throughput and IOPS. See the Performance Expectations section to see how to achieve higher throughput and IOPS by scaling out mount targets.

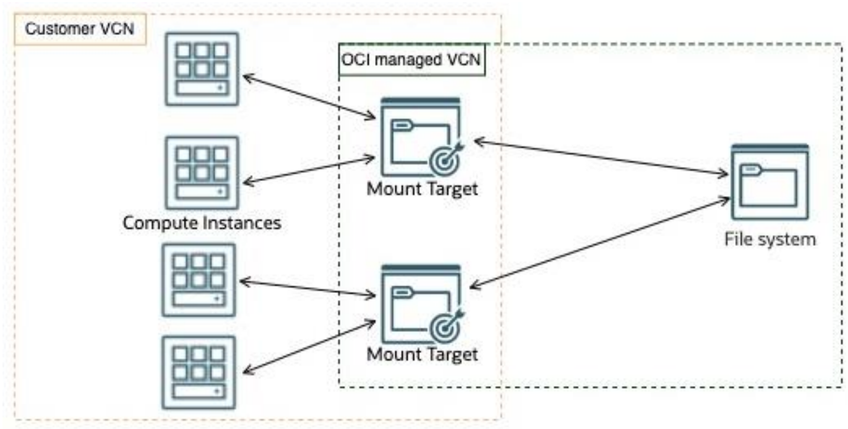


Figure 3. Mount Target Horizontal Scaling—Multiple Mount Target to Improve Throughput/IOPS to a File System

Load Distribution Across Mount Targets

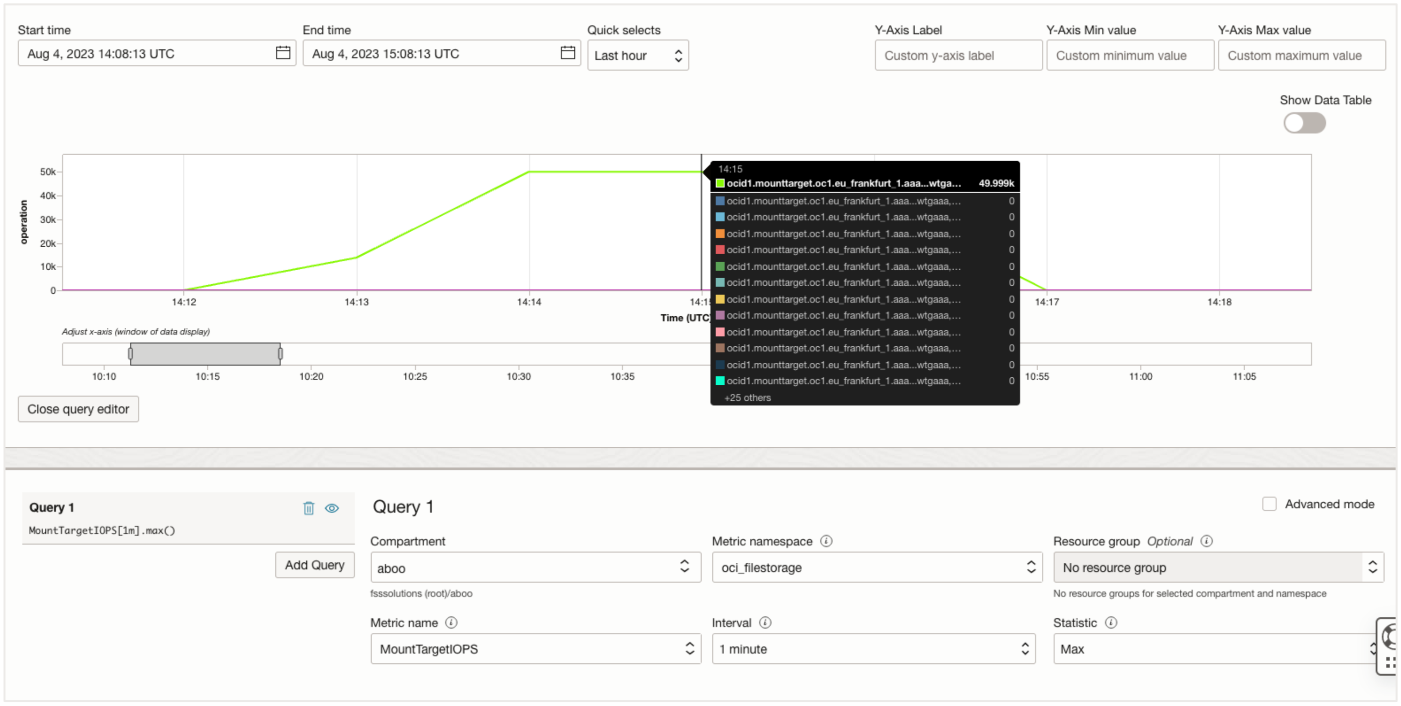
The load across multiple mount targets needs to be distributed evenly. This distribution can be done statically across your compute instances. For example, if you have 100 compute instances and 4 mount targets, 25 instances should use 1 mount target.

Monitoring Throughput and IOPS

File Storage provides metrics that you can use to monitor File Storage and take automated actions.

- Use the MountTargetIOPS metric to monitor the IOPS from a mount target.
- Use the MountTargetReadThroughout and MountTargetWriteThroughput metrics to monitor the throughput delivered from a mount target.
- Use the MetadataRequestAverageLatency, FileSystemReadAverageLatencybySize, and FileSystemWriteAverageLatencybySize metrics to monitor latency.

The following figure shows monitoring results of a mount target close to a 50K IOPS limit.



Start time: Aug 4, 2023 14:08:13 UTC
End time: Aug 4, 2023 15:08:13 UTC
Quick selects: Last hour
Y-Axis Label: Custom y-axis label
Y-Axis Min value: Custom minimum value
Y-Axis Max value: Custom maximum value

```
14:15
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtga...   49.999k
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
ocid1.mounttarget.oc1.eu_frankfurt_1.aaa...wtgaaa,...   0
+25 others
```

Query 1
MountTargetIOPS[1m].max()

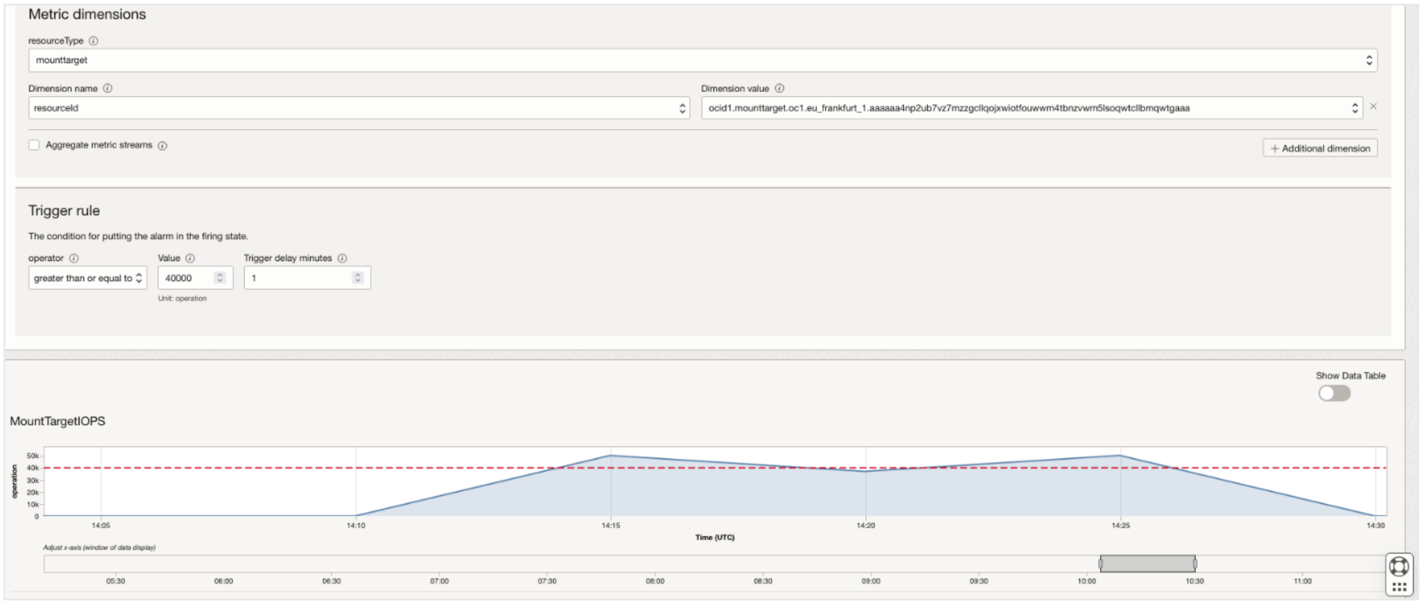
Query 1
Compartment: aboo
fsssolutions (root)/aboo
Metric namespace: oci_filestorage
Resource group *Optional*: No resource group
No resource groups for selected compartment and namespace
Metric name: MountTargetIOPS
Interval: 1 minute
Statistic: Max

Figure 4. Mount Target Close to 50K IOPS Limits

You can use OCI Monitoring to monitor and alarm on File Storage metrics. One aspect of monitoring File Storage is to get notification or take action when a mount target exceeds 80% of the maximum IOPS. When a mount target reaches this limit consistently, it's time add another mount target and distribute the load. The following alarm definitions can be used for this scenario. Similarly, you can configure alarms for throughput, latency, number of connections, and so on.

Alarm Configuration in Basic Mode

Alarms can be configured based on various File Storage metrics from the OCI Metric Explorer, as shown in the following screenshot.



Metric dimensions
resourceType: mounttarget
Dimension name: resourceId
Dimension value: ocid1.mounttarget.oc1.eu_frankfurt_1.aaaaaa4np2ub7vz7mzzgcllqojxwiotfouwwm4tbnzvwm5lsoqwtcllbmqwtgaaa
Aggregate metric streams

Trigger rule
The condition for putting the alarm in the firing state.
operator: greater than or equal to
Value: 40000
Unit: operation
Trigger delay minutes: 1

Figure 5. Alarm Configuration for 80% IOPS

Alarm Trigger in Advanced Mode

The following query is for setting alarms based on IOPS.

```
MountTargetIOPS[5m]{resourceType = "mounttarget", resourceId =
"ocid1.mounttarget.oc1.eu_frankfurt_1.aaaaaa4np2ub7vz7mzzgcllqojxwiotfouwwm4tbnzvwm5lsoqwtclxxxxxx"}.
max() >= 40000
```

Mount Target Size and Limits

To achieve the required IOPS and throughput for your workloads, we recommend scaling out the mount targets horizontally as previously explained. By default, you can have two mount targets per availability domain. The maximum possible throughput/IOPS per mount target is also limited. However, these limits can be increased based on your storage usage. If you have specific needs, such as more mount targets, higher throughput/IOPS per mount target than the default, or if your performance expectations are not met in your region, please contact us. We will evaluate your storage usage and performance requirements to provide you with the right number and type of mount targets for your tenancy.

I/O Latency

Throughput and IOPS aren't the only considerations for I/O-bound applications. I/O latency can also be a factor for performance, especially for applications that perform I/O operations in serial and single-threaded manner. The application tasks can take longer to complete because they're not able to take advantage of the higher queue depth and parallelism offered by File Storage. The parallelization can help the workloads to drive more IOPS and throughput from FSS without an increase in latency.

Tools are available that can help to parallelize legacy single-threaded application tasks. You can use the File Storage parallel tools—partar, parcp, and parrm—to parallelize file operations with tar, copy, and remove workflows. A good example of how the parallel tools can be used is in speeding up the application patching process. This process often involves removing old files and extracting new files, which can be time-consuming. However, the parallel tools can parallelize these file operations, which can significantly reduce the amount of time it takes to patch the application.

The following table and figure illustrate the performance improvements from using higher queue depth and concurrency. The partar tools performs I/O operations in parallel to drive more IOPS to the mount target. In this example, the traditional tar takes about 39 minutes to extract files from a large tarball, while partar takes only 15 minutes.

Table 3. Tar and partar Completion Time and IOPS Difference

TOOL	COMPLETION TIME IN MINUTES	IOPS USED
tar	39	1300
partar	15	5100

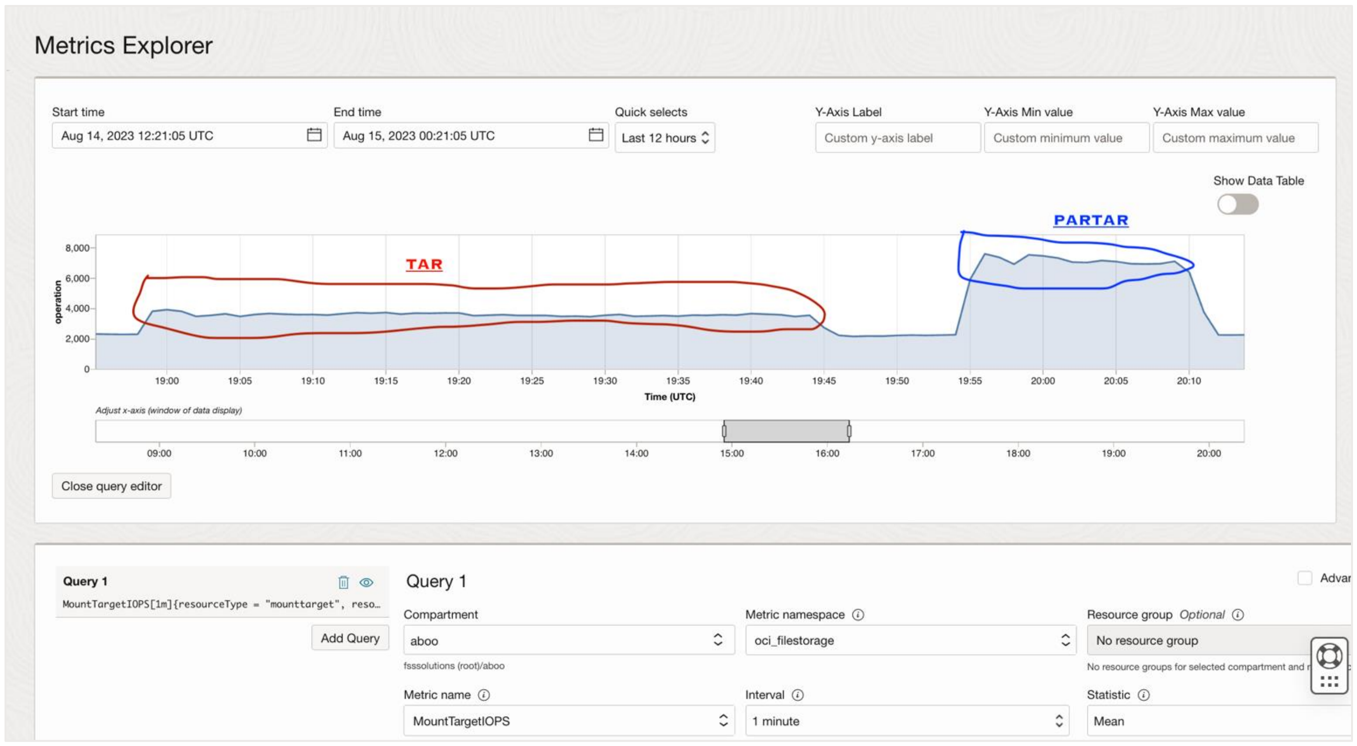


Figure 6. Tar and partar IOPS Difference

Network Latency

Higher network latency can lead to higher I/O latency. TCP window size and high network latency influence networking throughput because of the bandwidth delay product. For this reason, direct NFS mounts across OCI regions or across networks with high round trip time (RTT) is discouraged. We recommend having both compute instances and mount targets in the same availability domain for optimal performance.

Latency is generally a limitation with NFS and TCP connections across networks with large RTT. Increasing the number of TCP connections from the client can be helpful to alleviate the effect of high RTT. The number of connections per client can also increase the number of threads serving the I/O at the mount target to achieve higher throughput and IOPS. See the `nconnect` mount option in the next section.

For data migrations or data movement across networks with high RTT, use instance streaming rather than directly cross-mounting File Storage by using NFS.

NFS Mount Options

In modern Linux operating systems such as Oracle Linux 8, using the `nconnect=16` parameter can improve the performance. With this parameter, a Linux client can maintain up to 16 TCP connections and multiplex the I/O requests across these TCP connections.

For best performance, unless required by the application, don't set the `rsize` and `wsize` options when mounting the file system. In the absence of these options, the system automatically negotiates optimal read and write sizes. Larger read and write sizes are better for higher throughput.

Mount Target Caching

Caching at the mount target improves the latency of NFS operations. Single-threaded latency-sensitive workloads can get a significant performance boost from caching because caching reduces latency. Caching is always enabled unless the file system is exported across multiple mount targets. For this reason, horizontal scaling of mount targets can't use caching because the file system needs to be exported through multiple mount targets.

Directory Size

In File Storage, the total number of files in the entire file system can scale to billions of files. Although no limits imposed are by File Storage in a single flat directory, we recommend keeping the directories to less than 10,000 files. Large directories can slow down applications that rely on reading directories often. This isn't an architecture limitation of File Storage but a general situation with large directories when accessed through NFS.

Instance Capacity

The available network bandwidth of an instance has impact on I/O performance. In OCI, larger instances (more CPUs) are entitled to more network bandwidth. File Storage performance is best with OCI Compute bare metal instances or large VM shapes.

Performance Expectations

The highest levels of performance assume concurrent access and can be achieved only by using multiple clients, multiple threads, and multiple mount targets. The actual throughput and IOPS that can be achieved using a single mount target depend on many factors, such as the type and size of the I/O, the capacity of the instance, and I/O patterns. For this reason, this paper takes a practical approach to demonstrating mount target scaling and helping you get an expectation of IOPS and throughput for your workload.

The tests were conducted with up to eight mount targets attached to a single file system to show that the horizontal scaling of mount targets can scale linearly. The IOPS mentioned in this section are read or write NFS operations. They translate to IOPS at the mount target. It can be calculated by using the IOPS for read size and write size as described in the NFS Operations to IOPS Mapping section.

Test Environment

Region: eu-frankfurt-1/AD3

File system dataset size: 4 TiB

Mount targets: Eight mount targets exporting the same file system, one mount target mapped to each instance

IO type: Random reads and writes

INSTANCE	SHAPE	SIZE	MEMORY	NETWORK	OS
OCI VM x 8	VM.Standard2	16	240 GB	16 Gbps	Oracle Linux 8

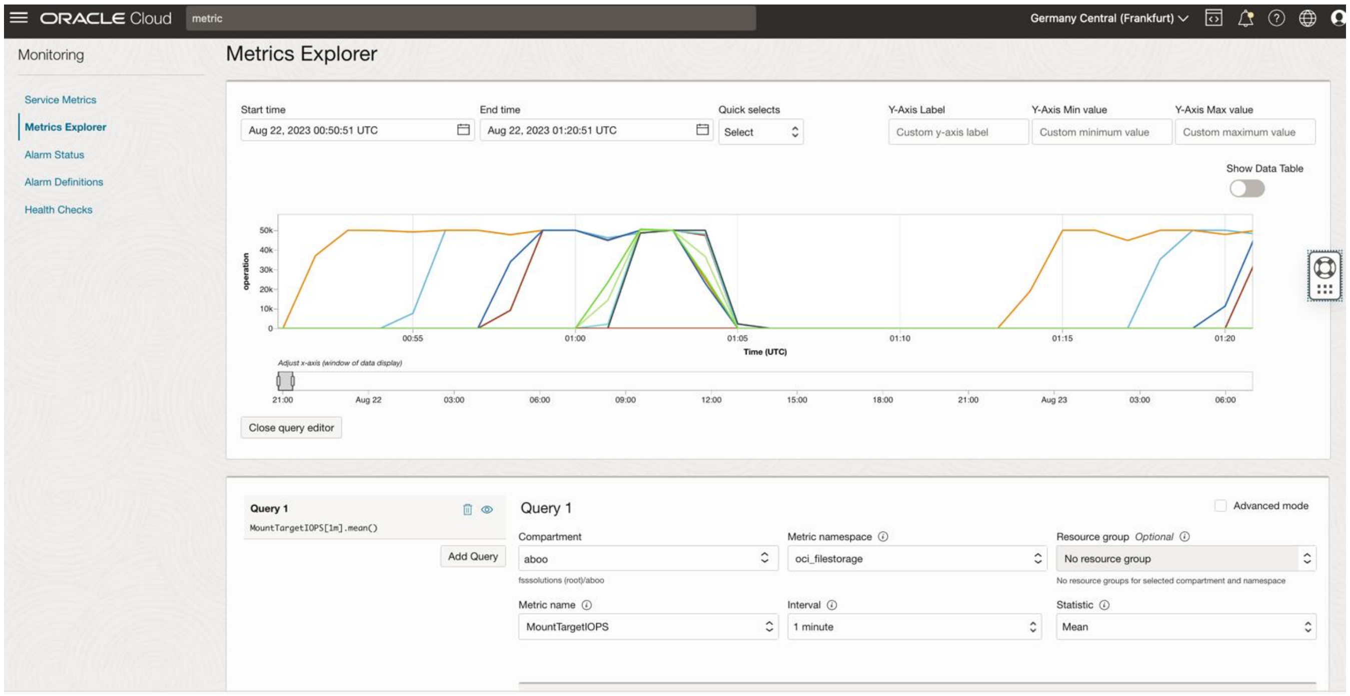
Test method: FIO

```
$ fio --name=read_throughput --directory=/fss --numjobs=8 -size=8G --time_based --runtime=180 --
ioengine=libaio --direct=1 --verify=0 --bs=$IOSIZE --iodepth=32 --rw=randread --group_reporting=1

$ fio --name=write_throughput --directory=/fss --numjobs=8 -size=8G --time_based --runtime=180 --
ioengine=libaio --direct=1 --verify=0 --bs=$IOSIZE --iodepth=32 --rw=randwrite --group_reporting=1
```

Test Results for Read: IOPS Optimized

I/O SIZE	NUMBER OF MOUNT TARGETS	AVERAGE READ IOPS PER MOUNT TARGET	TOTAL FS READ IOPS	AVERAGE THROUGHPUT PER MOUNT TARGET (MB/S)	TOTAL FS THROUGHPUT (MB/S)
32 KiB	1	25018	25018	819	819
32 KiB	2	25057	50115	820	1640
32 KiB	4	25056	100226	818	3279
32 KiB	8	24444	195554	800	6402



Start time: Aug 22, 2023 00:50:51 UTC
End time: Aug 22, 2023 01:20:51 UTC
Quick selects: Select
Y-Axis Label: Custom y-axis label
Y-Axis Min value: Custom minimum value
Y-Axis Max value: Custom maximum value

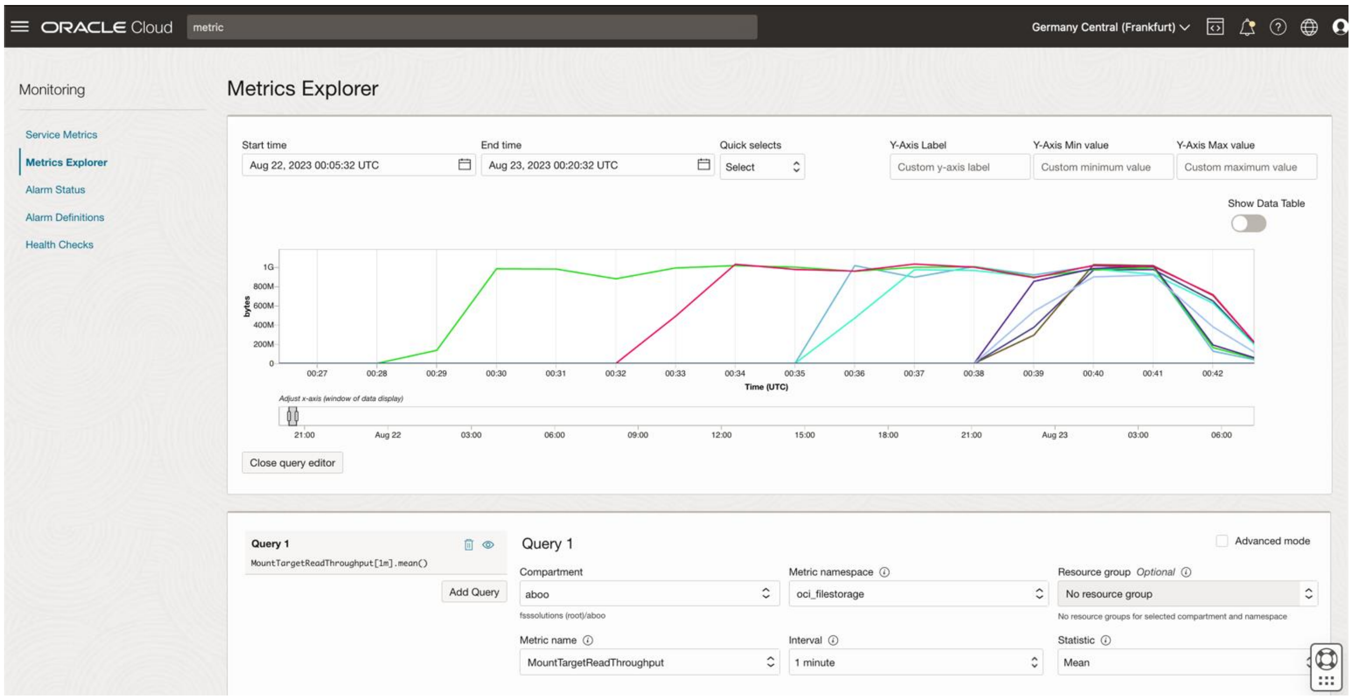
Query 1
MountTargetIOPS[1m].mean()

Compartment: aboo
fsssolutions (root)/aboo
Metric namespace: oci_filestorage
Resource group *Optional*: No resource group
No resource groups for selected compartment and namespace
Metric name: MountTargetIOPS
Interval: 1 minute
Statistic: Mean

Figure 7. Read Using 1-8 Mount Targets with 32-KiB IO Size

Test Results for Read: Throughput Optimized with Large I/O Size

I/O SIZE	NUMBER OF MOUNT TARGETS	AVERAGE READ IOPS PER MOUNT TARGET	TOTAL FS READ IOPS	AVERAGE THROUGHPUT PER MOUNT TARGET (MB/S)	TOTAL FS THROUGHPUT (MB/S)
1 MiB	1	937	937	983	983
1 MiB	2	964	1929	1008	2016
1 MiB	4	955	3923	990	3960
1 MiB	8	949	7594	990	7918



Start time: Aug 22, 2023 00:05:32 UTC
End time: Aug 23, 2023 00:20:32 UTC
Quick selects: Select
Y-Axis Label: Custom y-axis label
Y-Axis Min value: Custom minimum value
Y-Axis Max value: Custom maximum value

Query 1
```
MountTargetReadThroughput[1m].mean()
```

Query 1
Compartment: aboo
fsssolutions (root)/aboo
Metric namespace: oci_filestorage
Resource group *Optional*: No resource group
No resource groups for selected compartment and namespace
Metric name: MountTargetReadThroughput
Interval: 1 minute
Statistic: Mean

Figure 8. Read Using 1-8 Mount Targets with 1MiB IO Size

Test Results for Write: IOPS Optimized

I/O SIZE	NUMBER OF MOUNT TARGETS	AVERAGE WRITE IOPS PER MOUNT TARGET	TOTAL FS WRITE IOPS	AVERAGE THROUGHPUT PER MOUNT TARGET (MB/S)	TOTAL FS THROUGHPUT (MB/S)
32 KiB	1	12580	12580	412	412
32 KiB	2	12575	25151	412	824
32 KiB	4	12580	50321	412	1648
32 KiB	8	12574	100599	411	3295

Test Results for Write: Throughput Optimized with Large I/O Size

I/O SIZE	NUMBER OF MOUNTS	AVERAGE WRITE IOPS PER MOUNT TARGET	TOTAL FS WRITE IOPS	AVERAGE THROUGHPUT PER MOUNT TARGET (MB/S)	TOTAL FS THROUGHPUT (MB/S)
1 MiB	1	761	761	799	799
1 MiB	2	761	1523	799	1598
1 MiB	4	761	3046	799	3196
1 MiB	8	761	6093	799	6392

Mixed Read and Write Scenario

The performance of mixed read and write operations, whether sequential or random, is similar. As mentioned before, NFS read and write operations are translated into mount target IOPS based on the size of the I/O. The achievable performance is limited by the IOPS and bandwidth available to the mount target.

Conclusion

OCI File Storage file systems can scale to meet your requirements without the need to pre-provision storage size or performance. File Storage can be used for workloads that demand high IOPS and throughput. You can achieve the performance levels that you need by scaling out to multiple mount targets as demonstrated. If you have any questions or specific performance requirements, contact us by email or through OCI help and support.