OCI File Storage Performance Characteristics

November 2024, Version 3.0

Public

ORACLE

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Revision History

The following revisions have been made to this document.

Date	Revision
November 2024	Updated with information about new File Storage high performance mount targets
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 is used for a variety of purposes, including general-purpose file sharing and performance-intensive workloads such as media processing, artificial intelligence, and machine learning. When you deploy performance-intensive workloads in File Storage, it's important to understand the workload and performance characteristics of the service. 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 in parallel. You can scale up to exabytes without the need to preprovision storage. It uses Network File System (NFSv3) protocol to provide Portable Operating System Interface (POSIX)-compliant file system access, which 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 name space for accessing 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 system is exported through one or many mount targets.

Mount Targets

Mount targets are the network endpoints that you use to access the 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.

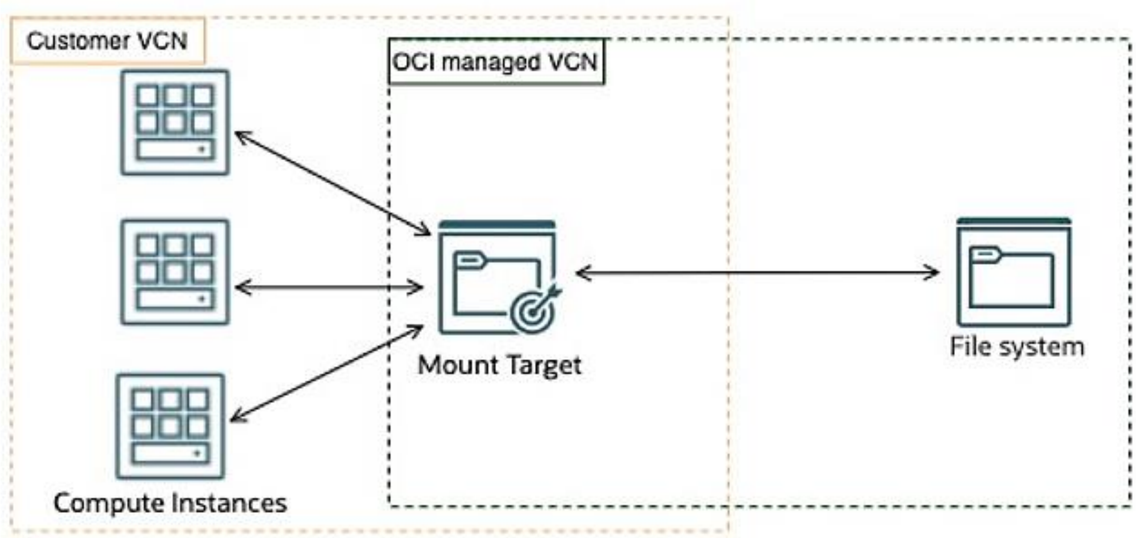

Figure 1. File System Access Through a Mount Target—One File System and One Mount Target

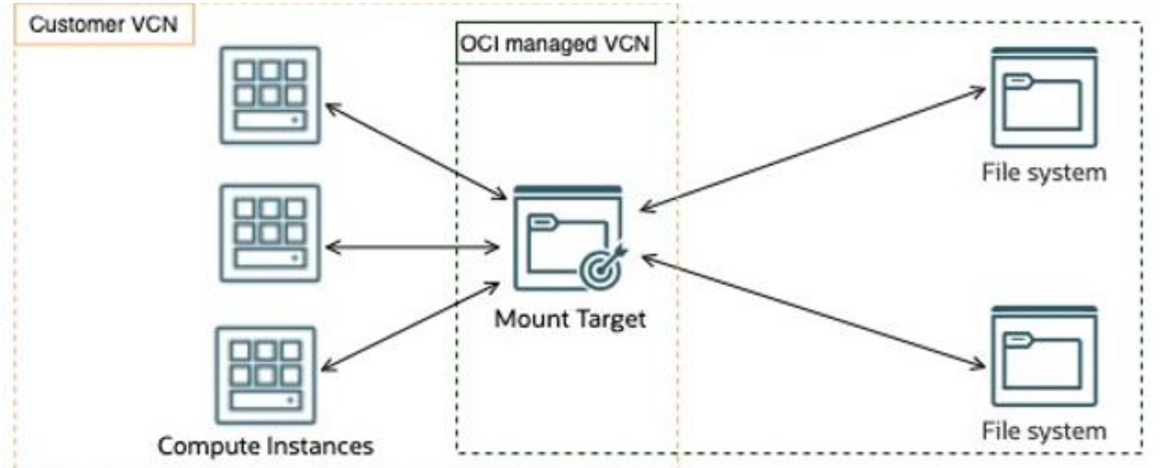

Figure 2. File System Access Through a Mount Target—Multiple File Systems and One Mount Target

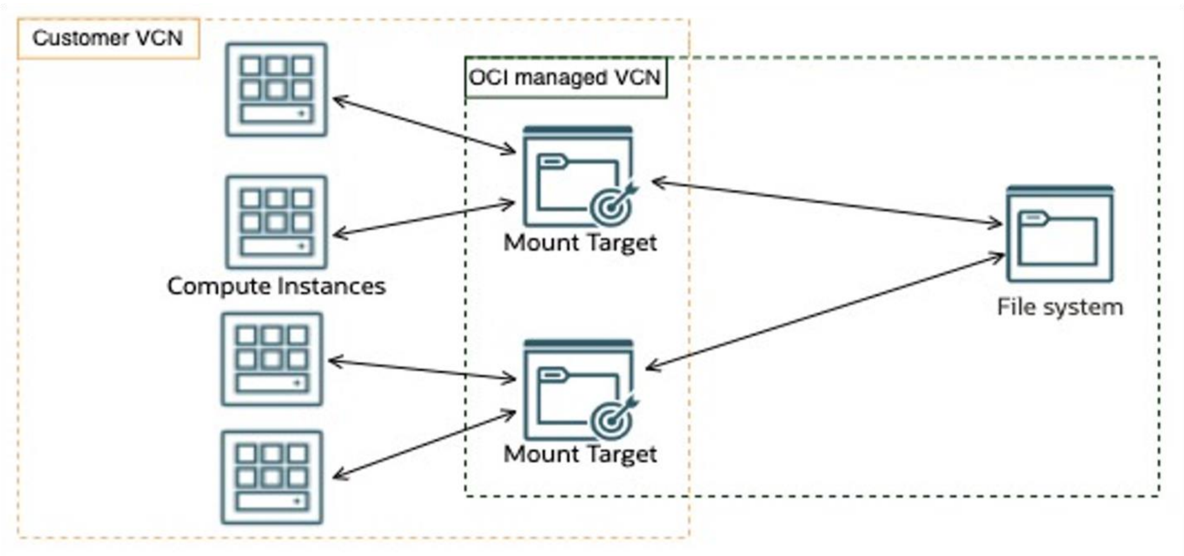

Figure 3. Mount Target Horizontal Scaling—Multiple Mount Target to Improve Throughput/IOPS to a File System

Performance Characteristics

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

File Storage is designed to scale without any limit. Performance can be scaled up in the following ways:

- **Scale up**: Change the mount target type to achieve 20 Gbps, 40 Gbps or 80 Gbps throughput. A single mount target can provide throughput up to 80 Gbps.
- **Scale out**: Linearly scale performance by adding additional mount targets for higher aggregate throughput. For example, use eight HPMT-80 high-performance mount targets to get an aggregate throughput of 640 Gbps to a file system. Multiple IP addresses are involved when multiple mount targets are used to access a file system. However, every mount point provides a consistent single view of the name space regardless of which IP address is in use. For details, see Scale Out OCI File Storage Performance for AI/ML and Data-Intensive Workloads.

Note: The number of mount targets that you can create in an availability domain is restricted by limits, which can be raised on request. There are separate limits for Standard, HPMT-20, HPMT-40, and HPMT-80 mount targets.

Mount Target Types and Performance Considerations

The mount target types are characterized by their provisioned read bandwidth without any service level agreement (SLA). The performance can further be scaled up linearly by adding more mount targets.

Table 1. Mount Target Types and Performance

Type	Maximum Read Bandwidth	Maximum Write Bandwidth	Expected Maximum IOPS	Capacity Requirement per Mount Target	Number of Clients
Standard	1 Gb/s	500 Mb/s	5K	Not applicable	100,000
HPMT-20	20 Gb/s	10 Gb/s	100K	20,000 GB	100,000
HPMT-40	40 Gb/s	20 Gb/s	200K	40,000 GB	100,000
HPMT-80	80 Gb/s	40 Gb/s	400K	80,000 GB	100,000

Consider the following information about the bandwidth and IOPS of the mount target types shown in the preceding table:

- The mount target performance is characterized by read throughput with 1-MB I/O size.
- The IOPS values aren't front-end IOPS. 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, one 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. This mapping is provided in table 2 in the next section. The MountTargetIOPS metric and mount target performance levels are tracked using backend IOPS.
- The bandwidth and IOPS values were the expected performance levels when this paper was published. Based on the region size and use in the fleet, you might observe higher performance than mentioned in the table.
- The capacity requirement per mount target type is the optimal size requirement to guarantee sufficient distribution at the storage backend to achieve expected performance. This requirement is also reflected in the customer monthly commitment of the high performance mount targets. File Storage optimally distributes the storage across as many storage servers as possible, which makes it possible to get higher performance levels even with a much smaller file system in most OCI commercial regions. However, this capacity requirement becomes an important matter when the number of mount targets increases as you scale out for performance with smaller file systems.

NFS Operations to IOPS Mapping

The following table maps front-end 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	None
CREATE, LINK, MKDIR, MKNOD, READDIRPLUS, RMDIR, and SYMLINK	4	None
REMOVE and RENAME	8	None
All NLM operations except FREE_ALL	4	None
NLM FREE_ALL	8	None
READ 32 KiB	2	For large I/O size greater than 32 KiB, add 1 IOPS for each 32 KiB size increment. For example: `read size > 32 KiB` and `size <= 64 KiB` is `3 IOPS`.
WRITE 32 KiB	4	For large I/O size greater than 32 KiB, add 2 IOPS for each 32 KiB size increment.

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.

You can use OCI Monitoring to monitor and alarm on File Storage metrics. One aspect of monitoring File Storage is to get notifications when a mount target exceeds 80% of its capacity for a period of time. When a mount target reaches this limit consistently, it's time add another mount target to distribute the load. The following screenshot shows an alarm definition that triggers when the mount target exceeds 40K IOPS consistently for a minute. Similarly, you can configure alarms for throughput, latency, number of connections, and so on.

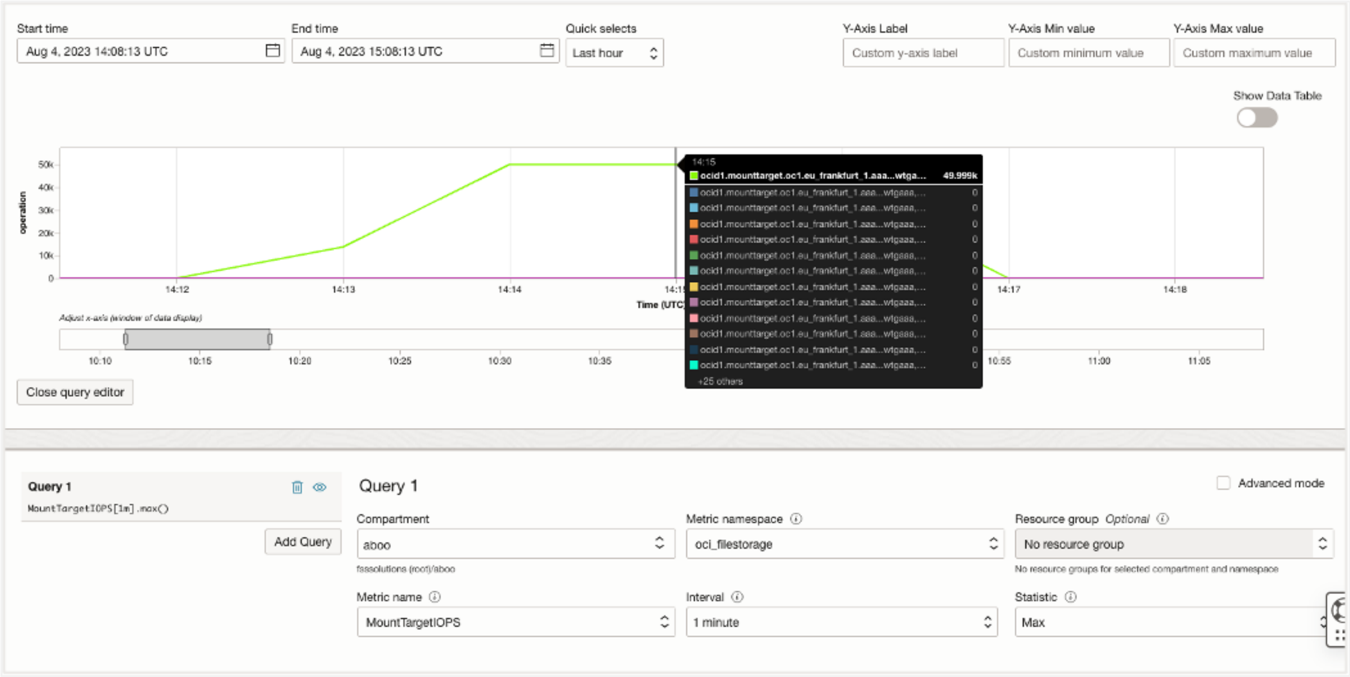

Figure 4. Mount Target IOPS

Alarm Configuration in Basic Mode

You can configure alarms based on various File Storage metrics from the OCI Metric Explorer.

Figure 5. Alarm Configuration for a Particular IOPS

Alarm Trigger in Advanced Mode

You can use queries to set alarms. The following query sets alarms based on IOPS. Similarly, you can set alarms based on other metrics for throughput and latency.

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

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. File Storage can deliver latency for most NFS operations under one millisecond. See the "Metadata Caching" section later in this paper for additional details.

The parallelization can help the workloads to drive more IOPS and throughput from File Storage 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 show 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

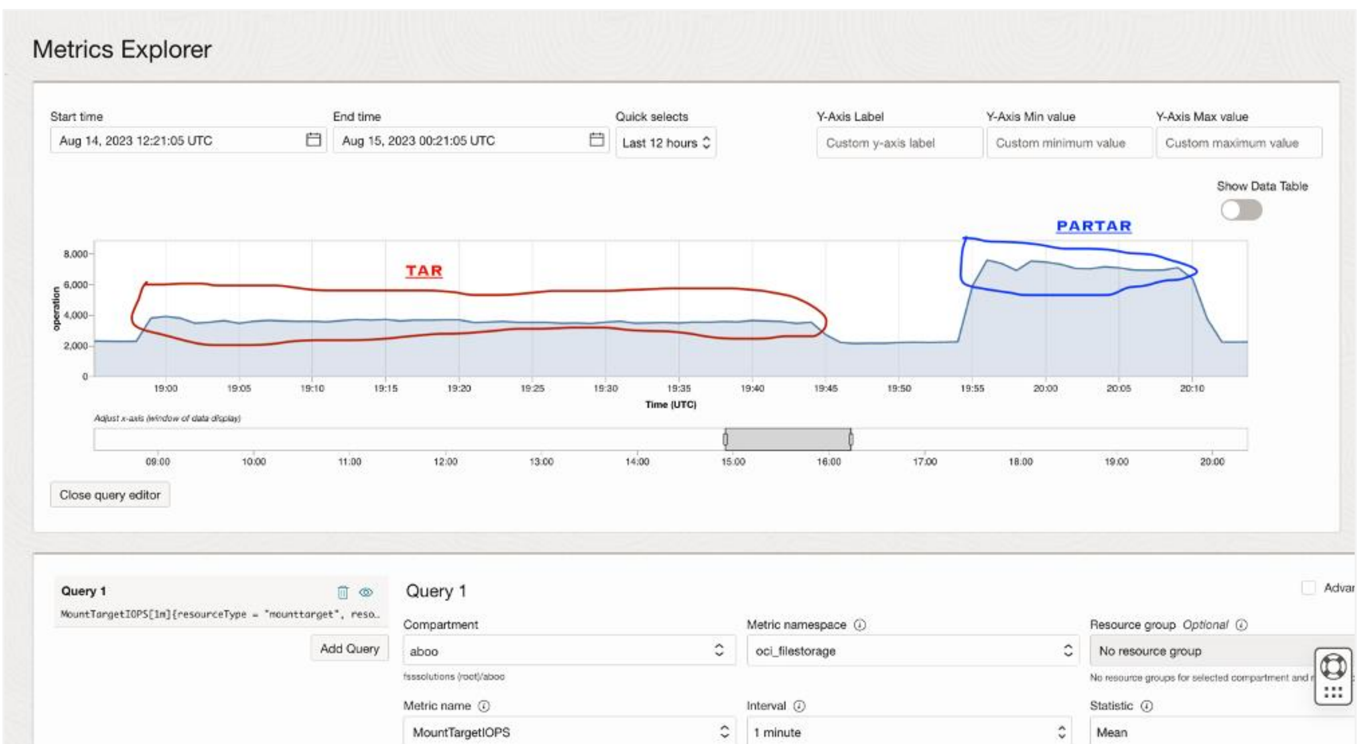

Figure 6. Tar and partar IOPS Difference

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, having 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 information about 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 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.

Metadata Caching

Metadata caching is an optimization that 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 benefit from 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 impose limits in a single flat directory, we recommend having fewer than 25,000 files in each directory. 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 by using a single mount target depends on many factors, such as the type of mount target and size of the I/O, the capacity of the instance, and I/O patterns. For this reason, this paper uses a practical approach to demonstrate mount target scaling using HPMT-80 mount targets. The tests described in this section were conducted with up to four mount targets attached to a single file system.

Test Environment

Region: eu-frankfurt-1/AD3

File system dataset size: 4 TiB

Mount targets: Four mount targets exporting the same file system (single name space), each distributed equally to the number of instances

I/O type: Random reads and writes

Instance	Shape	OCPU	Memory	NIC	OS	Mount Target Type
OCI VM	VM.Standard.E5.Flex	16	16 GB	16 Gbps	Oracle Linux 8	HPMT-80

Test method: FIO

```
$ sudo fio --name=fss-perf --directory=/fss --numjobs=32 -size=1G --time_based --runtime=600 --
ioengine=libaio --direct=1 --verify=0 --bs=1m –iodepth=64 –rw=randread --group_reporting=1
```

Note: Use the `-o nconnect=16` option when mounting a file system.

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

The test results confirm that performance scales linearly with the addition of mount targets. Like throughput, IOPS can also be scaled out, enabling File Storage to achieve millions of IOPS or more

I/O Size	Number of Mount Targets	Instances	Average Read IOPS per Mount Target	Total FS Read IOPS	Average Throughput per Mount Target (GB/s)	Total FS Throughput (GB/s)
1 MiB	1	8	391K	391K	12.1	12
1 MiB	2	16	390K	780K	12.1	24
1 MiB	4	24	391K	1173K	12.1	36
1 MiB	4	32	391K	1564K	12.1	48

Note: This test exceeded expectations, reaching over 12 GB/s per mount target. The maximum expected performance is 10 GB/s (80Gbps) for HPMT-80.

Figure 7. Scaling Out Mount Targets from 1 to 4—Throughput to Individual Mount Targets

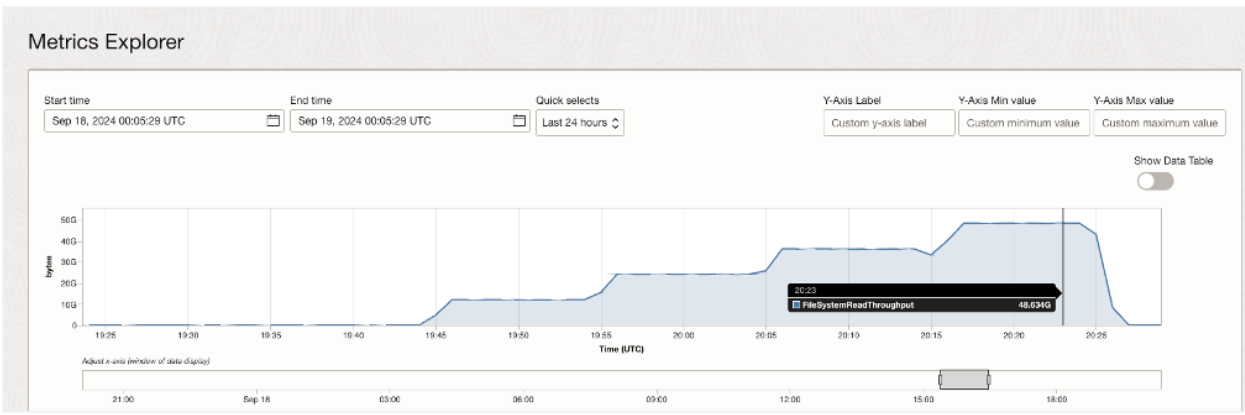

Figure 8. Scaling Out Mount Targets from 1 to 4—Throughput to File System Reaching 48 GB/s

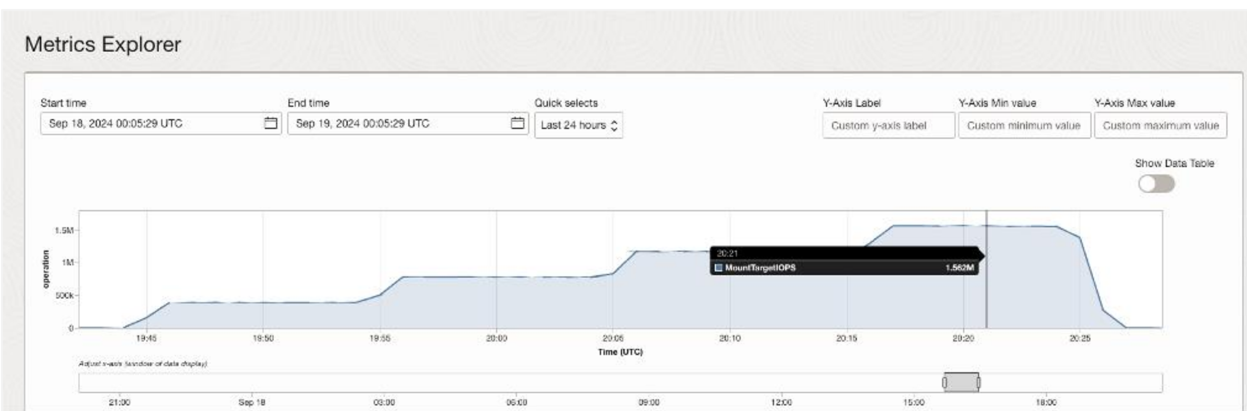

Figure 9. Scaling out Mount Targets from 1 to 4—IOPS Reaching 1.6 million IOPS to the File System

Test Results for Read: Small I/O Size (4K)

This test demonstrates that the throughput and IOPS achieved are lower with small I/O size. However, they scale out linearly with the addition of mount targets.

I/O Size	Number of Mount Targets	Instances	Average Read IOPS per Mount Target	Total FS Read IOPS	Average Throughput per Mount Target (MB/s)	Total FS Throughput (MB/s)
4 KiB	1	8	122k	122K	260	260
4 KiB	2	16	122k	244K	259	517
4 KiB	4	24	122k	366K	258	774
4 KiB	4	32	120K	483K	255	1022

Figure 10. Scaling Out Mount Targets from 1 to 4—Throughput to Individual Mount Targets

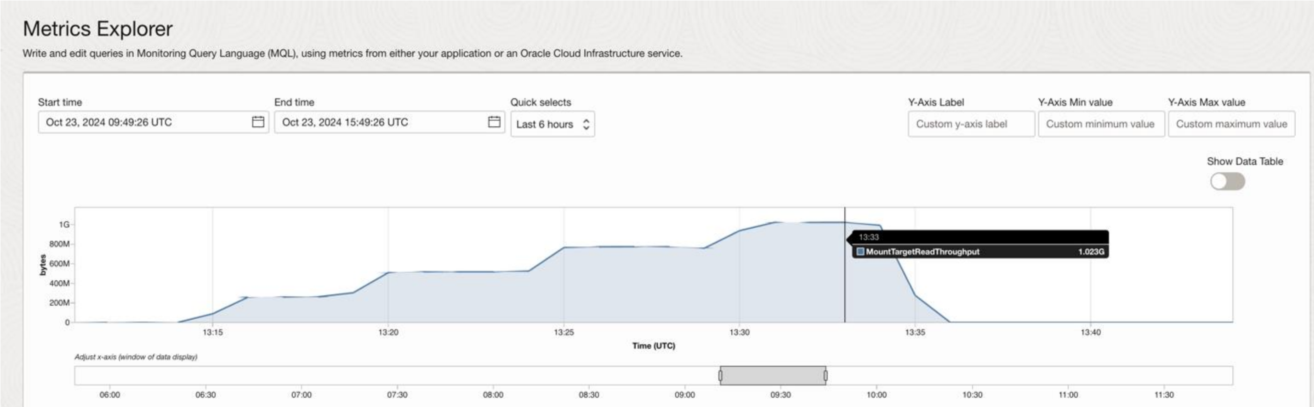

Figure 11. Scaling Out Mount Targets from 1 to 4—Throughput to File System

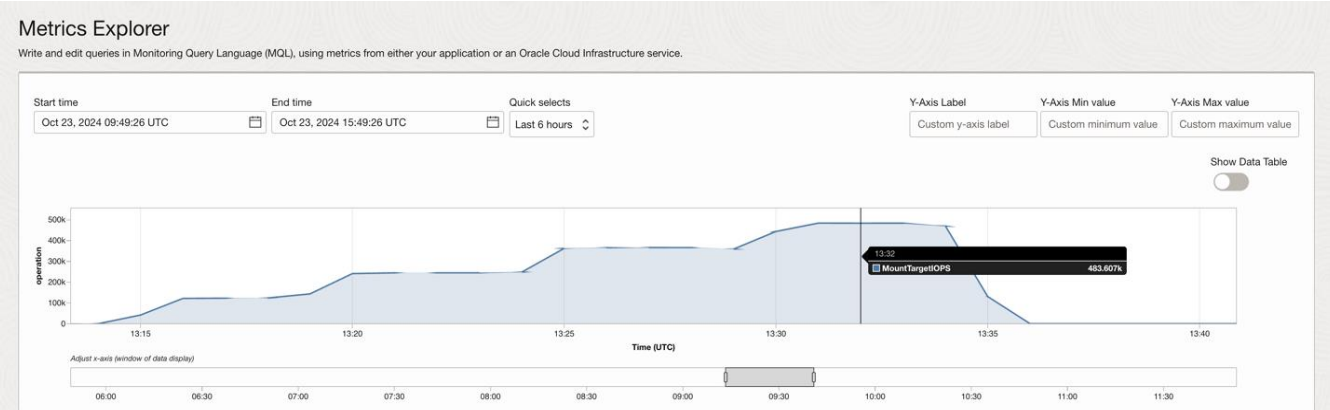

Figure 12. Scaling Out Mount Targets from 1 to 4—IOPS to File System

Conclusion

OCI File Storage file systems can scale to meet your requirements without the need to preprovision storage size or performance. File Storage is well-suited for workloads requiring high throughput and IOPS. To fine-tune performance, choose the appropriate mount target type: HPMT-20, HPMT-40, or HPMT-80. For I/O-intensive applications, leverage the power of scaling out by adding multiple mount targets to achieve linear performance increases. See our documentation for comprehensive information on scaling OCI File Storage for AI/ML and data-intensive workloads. If you have questions or specific performance requirements, contact us by email or through OCI help and support.

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at **oracle.com/contact**.

blogs.oracle.com   facebook.com/oracle   twitter.com/oracle