

Scalable File Service

User Guide

Issue 02
Date 2023-07-27



Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Permissions Management.....	1
1.1 Creating a User and Granting SFS Permissions.....	1
2 File System Management.....	3
2.1 Viewing a File System.....	3
2.2 Deleting a File System.....	4
3 Capacity Expansion.....	5
4 Quotas.....	7
5 Encryption.....	8
6 Backup.....	9
7 Monitoring.....	11
7.1 SFS Turbo Metrics.....	11
8 Auditing.....	13
9 Typical Applications.....	15
9.1 Enterprise Website/App Background.....	15
9.2 Log Printing.....	16
10 Other Operations.....	18
10.1 Testing SFS Turbo Performance.....	18
10.2 Mounting a File System to a Linux ECS as a Non-root User.....	26
10.3 Mounting a Subdirectory of an NFS File System to ECSs (Linux).....	28
10.4 Data Migration.....	29
10.4.1 Migration Description.....	29
10.4.2 Using Direct Connect to Migrate Data.....	30
10.4.3 Using the Internet to Migrate Data.....	31
10.4.4 Migrating Data Between File Systems.....	34
A Change History.....	36

1 Permissions Management

1.1 Creating a User and Granting SFS Permissions

1.1 Creating a User and Granting SFS Permissions

This chapter describes how to use IAM to implement fine-grained permissions control for your SFS resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing SFS resources.
- Grant only the permissions required for users to perform a specific task.

If your Huawei Cloud account does not require individual IAM users, skip this section.

This section describes the procedure for granting permissions (see [Figure 1-1](#)).

Prerequisites

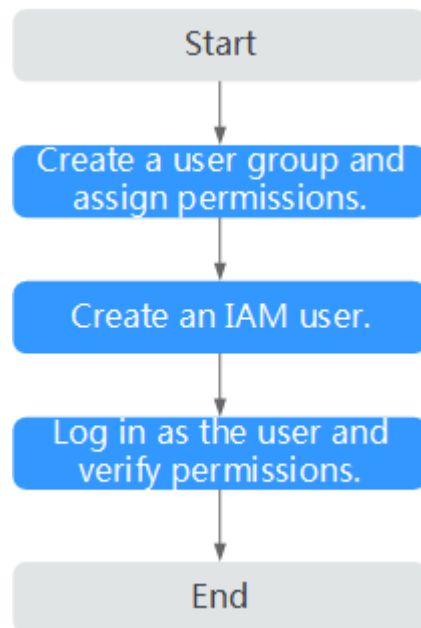
Learn about the permissions (see [System-defined roles and policies](#)) supported by SFS and choose policies or roles according to your requirements.

Restrictions

- Both system-defined policies and custom policies are supported in SFS Turbo file systems.

Process Flow

Figure 1-1 Process for granting SFS permissions



1. Create a user group and assign permissions to it.
Create a user group on the IAM console, and attach the **SFS Turbo ReadOnlyAccess** policy to the group.
2. Create a user and add it to a user group.
Create a user on the IAM console and add the user to the group created in **1**.
3. Log in and verify permissions.
Log in to SFS Console using the created user, and verify that the user only has read permissions for SFS.
 - Choose **Scalable File Service**. Click **Create File System** on SFS Console. If a message appears indicating that you have insufficient permissions to perform the operation, the **SFS Turbo ReadOnlyAccess** policy has already taken effect.
 - Choose any other service. If a message appears indicating that you have insufficient permissions to access the service, the **SFS Turbo ReadOnlyAccess** policy has already taken effect.

2 File System Management

[2.1 Viewing a File System](#)

[2.2 Deleting a File System](#)

2.1 Viewing a File System

You can search for file systems by file system name keyword or file system status, and view their basic information.

Procedure

Step 1 Log in to SFS Console.

Step 2 In the file system list, view the file systems you have created. [Table 2-1](#) describes the file system parameters.

Table 2-1 Parameter description

Parameter	Description
Name	Name of the file system, for example, sfs-name-001
Status	Possible values are Available, Unavailable, Frozen, Creating, Deleting.
Type	File system type
Protocol Type	File system protocol, which is NFS
Used Capacity (GB)	File system space already used for data storage NOTE This information is refreshed every 15 minutes.
Maximum Capacity (GB)	Maximum capacity of the file system
Encrypted	Encryption status of the file system. The value can be Yes or No.

Parameter	Description
Enterprise Project	Enterprise project to which the file system belongs
Mount Point	File system mount point, which is in the format of <i>File system IP address/</i>
Operation	For an SFS Turbo file system, valid operations include capacity expansion, deletion, monitoring metric viewing, subscription renewal, and unsubscription.

Step 3 (Optional) Search for file systems by file system name keyword or file system status.

----End

2.2 Deleting a File System

Data in a deleted file system cannot be restored. Ensure that files in a file system have been properly stored or backed up before you delete the file system.

Prerequisites

The file system to be deleted has been unmounted. For details about how to unmount the file system, see [Unmount a File System](#).

Procedure

Step 1 In the file system list, locate the file system you want to delete and click **Delete** in the **Operation** column.

Step 2 In the displayed dialog box, confirm the information and then click **Yes**. After clicking **Unsubscribe** for a yearly/monthly SFS Turbo file system, complete the unsubscription as prompted.

 **NOTE**

Only **Available** and **Unavailable** file systems can be deleted or unsubscribed from.

Step 3 Check that the file system disappears from the file system list.

----End

3 Capacity Expansion

Scenarios

You can expand the capacity of a file system when needed.

Constraints

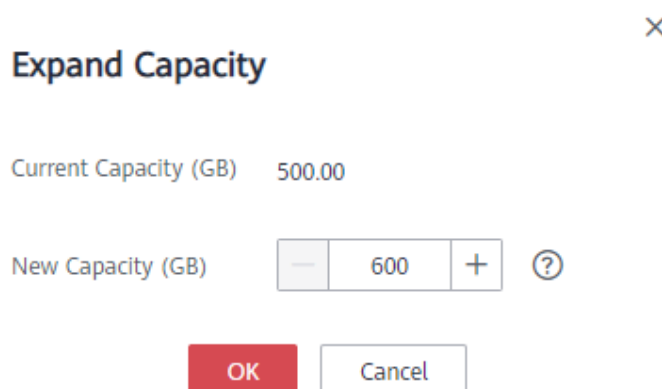
SFS Turbo file systems can only have their capacities expanded, not reduced. And only **In-use** file systems can be expanded.

Procedure

Step 1 Log in to SFS Console.

Step 2 In the file system list, click **Expand Capacity** in the row of the desired file system. The following dialog box is displayed. See [Figure 3-1](#).

Figure 3-1 Resizing a file system



Step 3 Enter a new maximum capacity of the file system based on service requirements, and click **OK**. [Table 3-1](#) describes the parameters.

Table 3-1 Parameter description

Parameter	Description
Used Capacity (GB)	Used capacity of the current file system
Maximum Capacity (GB)	Maximum capacity of the current file system
New Maximum Capacity (GB)	Target maximum capacity of the file system after expanding. The value ranges from 1 GB to 512,000 GB .

Step 4 In the displayed dialog box, confirm the information and click **OK**.

Step 5 In the file system list, check the capacity information after resizing.

----End


4 Quotas

What Is Quota?

Quotas can limit the number or amount of resources available to users, such as the maximum number of ECSs or EVS disks that can be created.

If the existing resource quota cannot meet your service requirements, you can apply for a higher quota.

How Do I View My Quotas?

1. Log in to the management console.
2. Click  in the upper left corner and select the desired region and project.
3. In the upper right corner of the page, choose **Resources > My Quotas**.
The **Service Quota** page is displayed.
4. View the used and total quota of each type of resources on the displayed page.
If a quota cannot meet service requirements, apply for a higher quota.

How Do I Apply for a Higher Quota?

1. Log in to the management console.
2. In the upper right corner of the page, choose **Resources > My Quotas**.
The **Service Quota** page is displayed.
3. Click **Increase Quota** in the upper right corner of the page.
4. On the **Create Service Ticket** page, configure parameters as required.
In the **Problem Description** area, fill in the content and reason for adjustment.
5. After all necessary parameters are configured, select **I have read and agree to the Ticket Service Protocol and Privacy Statement** and click **Submit**.

5 Encryption

Creating an Encrypted File System

To use the file system encryption function, you can directly select the encryption function when creating an SFS Turbo file system without authorization. For details, see [File System Encryption](#).

You can create a file system that is encrypted or not, but you cannot change the encryption settings of an existing file system.

For details about how to create an encrypted file system, see [Create a File System](#).

Unmounting an Encrypted File System

If the custom key used by the encrypted file system is disabled or scheduled for deletion, the file system can only be used within a certain period of time (30s by default). Exercise caution in this case.

For details about how to unmount the file system, see [Unmount a File System](#).

6 Backup

SFS Turbo file systems can be backed up using CBR.

Scenarios

A backup is a complete copy of an SFS Turbo file system at a specific time and it records all configuration data and service data at that time.

For example, if a file system is faulty or encounters a logical error (for example, mis-deletion, hacker attacks, and virus infection), you can use data backups to restore data quickly.

Creating a File System Backup

Ensure that the target file system is available. Or, the backup task cannot start. This procedure describes how to manually create a file system backup.

NOTE

If any modification is made to a file system during the backup, inconsistencies may occur. For example, there may be duplicate or deleted data, or data discrepancies. Such a modification includes a write, rename, move or delete. To ensure backup data consistency, you are advised to stop the applications or programs that use the file system during the backup, or schedule the backup at off-peak hours.

Step 1 In the navigation pane on the left, choose **SFS Turbo Backups**.

Step 2 The system automatically backs up the file system.

You can view the backup creation status on the **Backups** tab page. When the **Status** of the backup changes to **Available**, the backup has been created.


Step 3 If the file system becomes faulty or an error occurred, you can restore the backup data to a new file system. For details, see [Using a Backup to Create a File System](#).

----End

Using a Backup to Create a File System

In case of a virus attack, accidental deletion, or software or hardware fault, you can use an SFS Turbo file system backup to create a new file system. Data on the new file system is the same as that in the backup.

Step 1 Log in to CBR Console.

1. Log in to the management console.
2. Click  in the upper left corner and select your desired region and project.
3. Choose **Storage > Cloud Backup and Recovery > SFS Turbo Backups**.

Step 2 Click the **Backups** tab and locate the desired backup.

Step 3 If the status of the target backup is **Available**, click **Create File System** in the **Operation** column of the backup.

Step 4 Set the file system parameters.

 **NOTE**

- For detailed parameter descriptions, see table "Parameter description" under [Create a File System](#).

Step 5 Click **Next**.

Step 6 Go back to the file system list and check whether the file system is successfully created.

You will see the file system status change as follows: **Creating, Available, Restoring, Available**. After the file system status has changed from **Creating** to **Available**, the file system is successfully created. After the status has changed from **Restoring** to **Available**, backup data has been successfully restored to the created file system.

----End

7 Monitoring

7.1 SFS Turbo Metrics

7.1 SFS Turbo Metrics

Function

This section describes metrics reported by SFS Turbo to Cloud Eye as well as their namespaces and dimensions. You can use the console or APIs provided by Cloud Eye to query the metrics generated for SFS Turbo.

Namespace

SYS.EFS

Metrics

Table 7-1 SFS Turbo metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
client_connections	Client Connections	Number of client connections	≥ 0	SFS Turbo file system	1 minute
data_read_io_bytes	Read Bandwidth	Data read I/O load Unit: byte/s	≥ 0 bytes/s	SFS Turbo file system	1 minute
data_write_io_bytes	Write Bandwidth	Data write I/O load Unit: byte/s	≥ 0 bytes/s	SFS Turbo file system	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
metadatas_io_bytes	Metadata Read and Write Bandwidth	Metadata read and write I/O load Unit: byte/s	≥ 0 bytes/s	SFS Turbo file system	1 minute
total_io_bytes	Total Bandwidth	Total I/O load Unit: byte/s	≥ 0 bytes/s	SFS Turbo file system	1 minute
iops	IOPS	I/O operations per unit time	≥ 0	SFS Turbo file system	1 minute
used_capacity	Used Capacity	Used capacity of a file system Unit: byte	≥ 0 bytes	SFS Turbo file system	1 minute
used_capacity_percent	Capacity Usage	Percentage of used capacity in the total capacity Unit: Percent	0% to 100%	SFS Turbo file system	1 minute

Dimension

Key	Value
efs_instance_id	Instance

Viewing Monitoring Statistics

Step 1 Log in to the management console.

Step 2 View the monitoring graphs using either of the following methods.

- Method 1: Choose **Service List > Storage > Scalable File Service**. In the file system list, click **View Metric** in the **Operation** column of the target file system.
- Method 2: Choose **Management & Deployment > Cloud Eye > Cloud Service Monitoring > SFS Turbo**. In the file system list, click **View Metric** in the **Operation** column of the target file system.

Step 3 View the SFS Turbo file system monitoring data by metric or monitored duration.

For more information about Cloud Eye, see the *Cloud Eye User Guide*.

----End

8 Auditing

Scenarios

Cloud Trace Service (CTS) records operations of SFS resources, facilitating query, audit, and backtracking.

Prerequisites

Operations

Table 8-1 SFS operations traced by CTS


Operation	Resource Type	Trace
Creating a shared file system	sfs	createShare
Modifying a shared file system	sfs	updateShareInfo
Deleting a shared file system	sfs	deleteShare
Adding a share access rule	sfs	addShareACL
Deleting a share access rule	sfs	deleteShareACL

Table 8-2 SFS Turbo operations traced by CTS

Operation	Resource Type	Trace
Creating a file system	sfs_turbo	createShare
Deleting a file system	sfs_turbo	deleteShare

Querying Traces

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Choose **Management & Deployment > Cloud Trace Service**.

The **Cloud Trace Service** page is displayed.

Step 4 In the navigation pane on the left, choose **Trace List**.

Step 5 On the trace list page, set **Trace Source**, **Resource Type**, and **Search By**, and click **Query** to query the specified traces.

For details about other operations, see section "Querying Real-Time Traces" in the *Cloud Trace Service User Guide*.

----End

Disabling or Enabling a Tracker

This section describes how to disable an existing tracker on the CTS console. After the tracker is disabled, the system will stop recording operations, but you can still view existing operation records.

Step 1 Log in to the management console.

Step 2 Choose **> Cloud Trace Service**.

The **Cloud Trace Service** page is displayed.

Step 3 Click **Trackers** in the left pane.

Step 4 Click **Disable** on the right of the tracker information.

Step 5 Click **Yes**.

Step 6 After the tracker is disabled, the available operation changes from **Disable** to **Enable**. To enable the tracker again, click **Enable** and then click **Yes**. The system will start recording operations again.

----End

9 Typical Applications

[9.1 Enterprise Website/App Background](#)

[9.2 Log Printing](#)

9.1 Enterprise Website/App Background

Context

For I/O-intensive website services, SFS Turbo can provide shared website source code directories and storage for multiple web servers, enabling low-latency and high-IOPS concurrent share access. Features of such services are as follows:

- A large number of small files: Static website files need to be stored, including HTML files, JSON files, and static images.
- Read I/O intensive: Scope of data reading is large, and data writing is relatively small.
- Multiple web servers access an SFS Turbo background to achieve high availability of website services.

Configuration Process

1. Sort out the website files.
2. Log in to SFS Console. Create an SFS Turbo file system to store the website files.
3. Log in to the server that functions as the compute node and mount the file system.
4. On the head node, upload the files to the file system.
5. Start the web server.

Prerequisites

- A VPC has been created.
- Servers that function as head nodes and compute nodes have been created, and have been assigned to the VPC.

- SFS has been enabled.

Example Configuration

Step 1 Log in to SFS Console.

Step 2 On the **Create File System** page, set parameters as instructed.

Step 3 To mount a file system to Linux ECSs, see [Mounting an NFS File System to ECSs \(Linux\)](#). To mount a file system to Windows ECSs, see [Mounting an NFS File System to ECSs \(Windows\)](#).

Step 4 Log in to the head node and upload the files to the file system.

Step 5 Start the web server.

----End

9.2 Log Printing

Context

SFS Turbo can provide multiple service nodes for shared log output directories, facilitating log collection and management of distributed applications. Features of such services are as follows:

- A shared file system is mounted to multiple service hosts and logs are printed concurrently.
- Large file size and small I/O: The size of a single log file is large, but the I/O of each log writing is small.
- Write I/O intensive: Write I/O of small blocks is the major service.

Configuration Process

1. Log in to SFS Console. Create an SFS Turbo file system to store the log files.
2. Log in to the server that functions as the compute node and mount the file system.
3. Configure the log directory to the shared file system. It is recommended that each host use different log files.
4. Start applications.

Prerequisites

- A VPC has been created.
- Servers that function as head nodes and compute nodes have been created, and have been assigned to the VPC.
- SFS has been enabled.

Example Configuration

Step 1 Log in to SFS Console.

- Step 2** On the **Create File System** page, set parameters as instructed.
- Step 3** To mount a file system to Linux ECSs, see [Mounting an NFS File System to ECSs \(Linux\)](#). To mount a file system to Windows ECSs, see [Mounting an NFS File System to ECSs \(Windows\)](#).
- Step 4** Configure the log directory to the shared file system. It is recommended that each host use different log files.
- Step 5** Start applications.
- End

10 Other Operations

[10.1 Testing SFS Turbo Performance](#)

[10.2 Mounting a File System to a Linux ECS as a Non-root User](#)

[10.3 Mounting a Subdirectory of an NFS File System to ECSs \(Linux\)](#)

[10.4 Data Migration](#)

10.1 Testing SFS Turbo Performance

fio is an open-source I/O pressure testing tool. You can use fio to test the throughput and IOPS of SFS.

Prerequisites

fio has been installed on the ECS. It can be downloaded from [the official website](#) or from [GitHub](#).

Note and Description

The test performance depends on the network bandwidth between the client and server, as well as the capacity of the file system.

Installing fio

The following uses a Linux CentOS system as an example:

1. Download fio.
yum install fio
2. Install the libaio engine.
yum install libaio-devel
3. Check the fio version.
fio --version

File System Performance Data

The performance metrics of SFS Turbo file systems include IOPS and throughput. For details, see [Table 10-1](#).

Table 10-1 File system performance data

	General		HPC	
	SFS Turbo Standard	SFS Turbo Performance	125 MB/s/TiB	250 MB/s/TiB
Maximum capacity	32 TB	32 TB	1 PB	1 PB
Maximum IOPS	5,000	20,000	1 million	1 million
Maximum throughput	150 MB/s	350 MB/s	20 GB/s	20 GB/s
Formula used to calculate the IOPS	IOPS = Min. [5,000, (1,200 + 6 x Capacity)] Unit: GB	IOPS = Min. [20,000, (1,500 + 20 x Capacity)] Unit: GB	IOPS = Min. (1,000,000, 6,000 x Capacity) Unit: TB	IOPS = Min. (1,000,000, 12,500 x Capacity) Unit: TB

IOPS Calculation Formula

- IOPS of a single file system = Min. [Maximum IOPS, (Baseline IOPS + IOPS per GB x Capacity)]
For an SFS Turbo Performance file system:
 - If the file system capacity is 500 GB: IOPS = Min. [20,000, (1,500 + 20 x 500)] = 11,500
 - If the file system capacity is 1,000 GB: IOPS = Min. [20,000, (1,500 + 20 x 1,000)] = 20,000
- No performance calculation formula is available for the SFS Turbo Standard - Enhanced and SFS Turbo Performance - Enhanced file systems. The IOPS of an SFS Turbo Standard - Enhanced file system is 15,000, and that of an SFS Turbo Performance - Enhanced file system is 100,000.

Common Test Configuration Example

NOTE

The following estimated values are obtained from the test on a single ECS. You are advised to use multiple ECSs to test the performance of [SFS](#).

In the following examples, SFS Turbo Performance and ECSs with the following specifications are used for illustration.

Specifications: General computing-plus | c3.xlarge.4 | 4 vCPUs | 16 GB

Image: CentOS 7.5 64-bit

Mixed read/write with a read/write ratio of 7:3

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/nfs/test_fio --bs=4k --iodepth=128 --
size=10240M --readwrite=rw --rwmixwrite=30 --fallocate=none
```

NOTE

`/mnt/nfs/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/nfs` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (groupid=0, jobs=1): err=0: pid=10110: Mon Jun 0 11:40:57 2020
read: IOPS=7423, BW=28.0MiB/s (30.4MB/s)(7167MiB/247160msec)
slat (msec): min=1234, max=997477, avg=4145.45, stdev=3344.40
clat (msec): min=245, max=133325, avg=11162.10, stdev=12136.31
lat (msec): min=252, max=133330, avg=11166.32, stdev=12136.34
clat percentiles (msec):
| 1.00th=[ 2245], 5.00th=[ 2540], 10.00th=[ 2671], 20.00th=[ 2900],
| 30.00th=[ 3130], 40.00th=[ 3450], 50.00th=[ 4293], 60.00th=[ 7032],
| 70.00th=[13173], 80.00th=[19792], 90.00th=[20443], 95.00th=[36439],
| 99.00th=[53216], 99.50th=[60031], 99.90th=[79160], 99.95th=[85459],
| 99.99th=[90042]
bw ( KIB/s): min=16600, max=45560, per=100.00%, avg=29696.00, stdev=5544.46, samples=494
iops      : min= 4150, max=11390, avg=7424.01, stdev=1386.11, samples=494
write: IOPS=3182, BW=12.4MiB/s (13.0MB/s)(3073MiB/247160msec)
slat (msec): min=1488, max=302730, avg=4613.59, stdev=3359.60
clat (msec): min=1447, max=140666, avg=14166.05, stdev=13373.72
lat (msec): min=1457, max=140671, avg=14170.73, stdev=13373.74
clat percentiles (msec):
| 1.00th=[  41], 5.00th=[  41], 10.00th=[  41], 20.00th=[  51],
| 30.00th=[  51], 40.00th=[  61], 50.00th=[  81], 60.00th=[ 141],
| 70.00th=[ 101], 80.00th=[ 241], 90.00th=[ 331], 95.00th=[ 421],
| 99.00th=[ 591], 99.50th=[ 671], 99.90th=[ 871], 99.95th=[ 941],
| 99.99th=[ 1221]
bw ( KIB/s): min= 7144, max=19600, per=100.00%, avg=12730.90, stdev=2395.77, samples=494
iops      : min= 1706, max= 4900, avg=3102.70, stdev=590.96, samples=494
lat (msec) : 250=0.01%, 500=0.01%, 750=0.01%, 1000=0.01%
lat (msec) : 2=0.20%, 4=39.15%, 10=21.01%, 20=17.92%, 50=20.06%
lat (msec) : 100=1.62%, 250=0.02%
cpu       : usr=1.35%, sys=6.43%, ctx=1072910, majf=0, minf=30
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
submit    : 0=0.0%, 4=100.0%, 0=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete : 0=0.0%, 4=100.0%, 0=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued rwts: total=1034036,706604,0,0 short=0,0,0,0 dropped=0,0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=120

Run status group 0 (all jobs):
  READ: bw=28.0MiB/s (30.4MB/s), 28.0MiB/s-28.0MiB/s (30.4MB/s-30.4MB/s), io=7167MiB (7515MB), run=247160-247160msec
  WRITE: bw=12.4MiB/s (13.0MB/s), 12.4MiB/s-12.4MiB/s (13.0MB/s-13.0MB/s), io=3073MiB (3222MB), run=247160-247160msec
```

Mixed read/write with a read/write ratio of 3:7

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/nfs/test_fio --bs=4k --iodepth=128 --
size=10240M --readwrite=rw --rwmixwrite=70 --fallocate=none
```

NOTE

`/mnt/nfs/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/nfs` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (groupid=0, jobs=1): err= 0: pid=28358: Mon Jun 8 11:57:14 2028
read: IOPS=5865, BW=19.8MiB/s (20.7MB/s)(3873MiB/155288msec)
slat (usec): min=1271, max=269588, avg=4873.51, stdev=3848.12
clat (usec): min=226, max=88185, avg=5711.35, stdev=7879.46
lat (usec): min=232, max=88187, avg=5715.49, stdev=7879.48
clat percentiles (usec):
| 1.00th=[ 1221], 5.00th=[ 1958], 10.00th=[ 2188], 20.00th=[ 2442],
| 30.00th=[ 2686], 40.00th=[ 2882], 50.00th=[ 2999], 60.00th=[ 3228],
| 70.00th=[ 3687], 80.00th=[ 5684], 90.00th=[14222], 95.00th=[21898],
| 99.00th=[35914], 99.50th=[48633], 99.90th=[51643], 99.95th=[55837],
| 99.99th=[66847]
bw ( KIB/s): min=13368, max=28848, per=99.99%, avg=28257.97, stdev=2913.85, samples=318
iops      : min= 3348, max= 7212, avg=5864.48, stdev=728.27, samples=318
write: IOPS=11.8k, BW=46.2MiB/s (48.4MB/s)(7167MiB/155288msec)
slat (usec): min=1396, max=398684, avg=4485.68, stdev=3891.75
clat (usec): min=857, max=148259, avg=8377.47, stdev=8488.15
lat (usec): min=867, max=148264, avg=8382.82, stdev=8488.16
clat percentiles (msec):
| 1.00th=[  31], 5.00th=[  41], 10.00th=[  41], 20.00th=[  41],
| 30.00th=[  51], 40.00th=[  51], 50.00th=[  51], 60.00th=[  61],
| 70.00th=[  71], 80.00th=[ 131], 90.00th=[ 211], 95.00th=[ 281],
| 99.00th=[ 421], 99.50th=[ 471], 99.90th=[ 681], 99.95th=[ 681],
| 99.99th=[ 1281]
bw ( KIB/s): min=32224, max=67456, per=99.98%, avg=47254.23, stdev=6792.41, samples=318
iops      : min= 8856, max=16864, avg=11813.55, stdev=1698.11, samples=318
lat (usec) : 250=0.81%, 500=0.84%, 750=0.87%, 1000=0.89%
lat (msec) : 2=1.53%, 4=36.85%, 10=41.27%, 20=11.38%, 50=0.61%
lat (msec) : 100=0.23%, 250=0.81%
cpu       : usr=2.13%, sys=9.98%, ctx=925778, majf=0, minf=31
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued rwts: total=786597,1834843,0,0 short=0,0,0,0 dropped=0,0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
READ: bw=19.8MiB/s (20.7MB/s), 19.8MiB/s-19.8MiB/s (20.7MB/s-20.7MB/s), io=3873MiB (3222MB), run=155288-155288msec
WRITE: bw=46.2MiB/s (48.4MB/s), 46.2MiB/s-46.2MiB/s (48.4MB/s-48.4MB/s), io=7167MiB (7516MB), run=155288-155288msec
```

Sequential read IOPS

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=4k --iodepth=128 --
size=10240M --readwrite=read --fallocate=none
```

NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (groupid=0, jobs=1): err= 0: pid=28459: Mon Jun 8 12:28:18 2028
read: IOPS=9654, BW=37.7MiB/s (39.5MB/s)(18.86GiB/271519msec)
slat (usec): min=1233, max=662168, avg=4118.17, stdev=4773.23
clat (usec): min=365, max=131116, avg=13253.18, stdev=13958.89
lat (usec): min=371, max=131118, avg=13257.29, stdev=13958.89
clat percentiles (usec):
| 1.00th=[ 1762], 5.00th=[ 1991], 10.00th=[ 2147], 20.00th=[ 2376],
| 30.00th=[ 2704], 40.00th=[ 3621], 50.00th=[ 7767], 60.00th=[ 11994],
| 70.00th=[ 16989], 80.00th=[ 23462], 90.00th=[ 33162], 95.00th=[ 41681],
| 99.00th=[ 59587], 99.50th=[ 66847], 99.90th=[ 83362], 99.95th=[ 98782],
| 99.99th=[183285]
bw ( KIB/s): min=18656, max=61576, per=99.99%, avg=38615.41, stdev=7783.32, samples=543
iops      : min= 4664, max=15394, avg=9653.82, stdev=1925.83, samples=543
lat (usec) : 500=0.81%, 750=0.81%, 1000=0.82%
lat (msec) : 2=5.25%, 4=36.35%, 10=12.76%, 20=20.56%, 50=22.62%
lat (msec) : 100=2.42%, 250=0.82%
cpu       : usr=1.84%, sys=5.35%, ctx=913138, majf=0, minf=159
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued rwts: total=2621448,0,0,0 short=0,0,0,0 dropped=0,0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
READ: bw=37.7MiB/s (39.5MB/s), 37.7MiB/s-37.7MiB/s (39.5MB/s-39.5MB/s), io=18.86GiB (18.76B), run=2
```

Random read IOPS

- fio command:


```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --  
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=4k --iodepth=128 --  
size=10240M --readwrite=randread --fallocate=none
```

NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128  
fio-2.1.10  
Starting 1 process  
Jobs: 1 (f=1): [r] [100.0% done] [17824KB/0KB/0KB /s] [4456/0/0 iops] [eta 00m:00s]  
test: (groupid=0, jobs=1): err= 0: pid=20755: Tue Dec 28 09:41:43 2021  
read : io=10240MB, bw=18597KB/s, iops=4649, runt=563832msec  
slat (usec): min=1, max=375, avg= 2.64, stdev= 2.52  
clat (usec): min=715, max=755902, avg=27527.31, stdev=106233.39  
lat (usec): min=718, max=755903, avg=27530.03, stdev=106233.39  
clat percentiles (msec):  
| 1.00th=[ 3], 5.00th=[ 5], 10.00th=[ 6], 20.00th=[ 6],  
| 30.00th=[ 7], 40.00th=[ 7], 50.00th=[ 8], 60.00th=[ 9],  
| 70.00th=[ 11], 80.00th=[ 15], 90.00th=[ 21], 95.00th=[ 28],  
| 99.00th=[ 676], 99.50th=[ 693], 99.90th=[ 725], 99.95th=[ 734],  
| 99.99th=[ 750]  
bw (KB /s): min= 1896, max=35752, per=100.00%, avg=18605.56, stdev=1980.86  
lat (usec): 750=0.01%, 1000=0.01%  
lat (msec): 2=0.32%, 4=3.28%, 10=63.65%, 20=22.42%, 50=7.50%  
lat (msec): 100=0.07%, 250=0.01%, 500=0.03%, 750=2.72%, 1000=0.01%  
cpu : usr=0.82%, sys=2.41%, ctx=1231561, majf=0, minf=155  
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%  
submit : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%  
complete : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%  
issued : total=r=2621440/w=0/d=0, short=r=0/w=0/d=0  
latency : target=0, window=0, percentile=100.00%, depth=128  
  
Run status group 0 (all jobs):  
READ: io=10240MB, aggrb=18597KB/s, minb=18597KB/s, maxb=18597KB/s, mint=563832msec, maxt=563832msec
```

Sequential write IOPS

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --  
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=4k --iodepth=128 --  
size=10240M --readwrite=write --fallocate=none
```

NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```

test: (groupid=0, jobs=1): err= 0: pid=20874: Mon Jun  8 14:23:09 2020
write: IOPS=11.0k, BW=43.1MiB/s (45.2MB/s)(10.0GiB/237436msec)
slat (nsec): min=1483, max=368726, avg=4388.87, stdev=3688.87
clat (usec): min=1953, max=106548, avg=11588.61, stdev=5876.84
lat (usec): min=1959, max=106552, avg=11593.86, stdev=5876.86
clat percentiles (usec):
| 1.00th=[ 4015], 5.00th=[ 5932], 10.00th=[ 6652], 20.00th=[ 7439],
| 30.00th=[ 8029], 40.00th=[ 8848], 50.00th=[ 9634], 60.00th=[10814],
| 70.00th=[12518], 80.00th=[15533], 90.00th=[19268], 95.00th=[22676],
| 99.00th=[32637], 99.50th=[37487], 99.90th=[49821], 99.95th=[53748],
| 99.99th=[69731]
bw ( Kib/s): min=31712, max=52431, per=99.99%, avg=44158.04, stdev=3987.31, samples=474
iops      : min= 7928, max=13187, avg=11839.58, stdev=996.83, samples=474
lat (msec) : 2=0.01%, 4=1.00%, 10=51.94%, 20=38.58%, 50=8.39%
lat (msec) : 100=0.00%, 250=0.01%
cpu       : usr=1.33%, sys=5.47%, ctx=392117, majf=0, minf=27
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued rwts: total=8,2621440,0,0 short=0,0,0 dropped=0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
WRITE: bw=43.1MiB/s (45.2MB/s), 43.1MiB/s-43.1MiB/s (45.2MB/s-45.2MB/s), io=10.0GiB (10.7GB), run=

```

Random write IOPS

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=4k --iodepth=128 --
size=10240M --readwrite=randwrite --fallocate=none
```

NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```

test: (g=0): rw=randwrite, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=128
fio-2.1.10
Starting 1 process

test: (groupid=0, jobs=1): err= 0: pid=16622: Thu Jan 13 10:13:22 2022
write: io=10240MB, bw=18463KB/s, iops=4615, runt=567947msec
slat (usec): min=1, max=356, avg= 3.21, stdev= 2.04
clat (usec): min=890, max=815560, avg=27727.54, stdev=101207.14
lat (usec): min=893, max=815564, avg=27730.83, stdev=101207.14
clat percentiles (msec):
| 1.00th=[  4], 5.00th=[  6], 10.00th=[  6], 20.00th=[  7],
| 30.00th=[  7], 40.00th=[  8], 50.00th=[  8], 60.00th=[ 10],
| 70.00th=[ 13], 80.00th=[ 16], 90.00th=[ 23], 95.00th=[ 30],
| 99.00th=[ 644], 99.50th=[ 668], 99.90th=[ 701], 99.95th=[ 709],
| 99.99th=[ 734]
bw (KB /s): min= 1064, max=36589, per=100.00%, avg=18469.11, stdev=3769.64
lat (usec) : 1000=0.01%
lat (msec) : 2=0.20%, 4=1.85%, 10=60.93%, 20=24.30%, 50=9.85%
lat (msec) : 100=0.09%, 250=0.01%, 500=0.08%, 750=2.68%, 1000=0.01%
cpu       : usr=0.98%, sys=2.90%, ctx=1552744, majf=0, minf=27
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued   : total=r=0/w=2621440/d=0, short=r=0/w=0/d=0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
WRITE: io=10240MB, aggrb=18462KB/s, minb=18462KB/s, maxb=18462KB/s, mint=567947msec, maxt=567947msec

```

Sequential read bandwidth

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=1M --iodepth=128 --
size=10240M --readwrite=read --fallocate=none
```

 NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (groupid=0, jobs=1): err= 0: pid=28962: Mon Jun 8 14:37:48 2020
read: IOPS=390, BW=391MiB/s (409MB/s)(10.06GiB/26221msec)
slat (usec): min=76, max=595, avg=99.58, stdev=39.09
clat (msec): min=35, max=544, avg=327.38, stdev=99.64
lat (msec): min=36, max=545, avg=327.48, stdev=99.63
clat percentiles (msec):
| 1.00th=[ 155], 5.00th=[ 161], 10.00th=[ 167], 20.00th=[ 180],
| 30.00th=[ 368], 40.00th=[ 372], 50.00th=[ 380], 60.00th=[ 384],
| 70.00th=[ 388], 80.00th=[ 393], 90.00th=[ 401], 95.00th=[ 414],
| 99.00th=[ 472], 99.50th=[ 506], 99.90th=[ 535], 99.95th=[ 542],
| 99.99th=[ 542]
bw ( KiB/s): min=381856, max=768000, per=99.52%, avg=397987.65, stdev=81583.56, samples=52
iops      : min= 294, max= 750, avg=388.65, stdev=79.67, samples=52
lat (msec): 50=0.17%, 100=0.28%, 250=27.61%, 500=71.37%, 750=0.58%
cpu       : usr=0.00%, sys=4.21%, ctx=10395, majf=0, minf=97
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.2%, 32=0.3%, >=64=99.4%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued rwts: total=10240,0,0,0 short=0,0,0,0 dropped=0,0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
  READ: bw=391MiB/s (409MB/s), 391MiB/s-391MiB/s (409MB/s-409MB/s), io=10.06GiB (10.7GB), run=26221-26221msec
```

Random read bandwidth

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=1M --iodepth=128 --
size=10240M --readwrite=randread --fallocate=none
```

 NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (g=0): rw=randread, bs=1M-1M/1M-1M/1M-1M, ioengine=libaio, iodepth=128
fio-2.1.10
Starting 1 process
test: (groupid=0, jobs=1): err= 0: pid=14261: Tue Dec 28 09:18:04 2021
read: io=10240MB, bw=154130KB/s, iops=150, runt= 68032msec
slat (usec): min=61, max=8550, avg=142.99, stdev=187.96
clat (msec): min=12, max=2002, avg=849.91, stdev=347.27
lat (msec): min=12, max=2003, avg=850.05, stdev=347.26
clat percentiles (msec):
| 1.00th=[ 47], 5.00th=[ 84], 10.00th=[ 105], 20.00th=[ 914],
| 30.00th=[ 947], 40.00th=[ 963], 50.00th=[ 971], 60.00th=[ 988],
| 70.00th=[ 996], 80.00th=[ 1012], 90.00th=[ 1037], 95.00th=[ 1057],
| 99.00th=[ 1876], 99.50th=[ 1926], 99.90th=[ 1975], 99.95th=[ 1975],
| 99.99th=[ 2008]
bw (KB /s): min=69974, max=167768, per=98.85%, avg=152360.15, stdev=10783.47
lat (msec): 20=0.33%, 50=0.80%, 100=7.02%, 250=7.95%, 1000=55.30%
lat (msec): 2000=28.57%, >=2000=0.02%
cpu       : usr=0.02%, sys=1.93%, ctx=4399, majf=0, minf=602
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.2%, 32=0.3%, >=64=99.4%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued   : total=r=10240/w=0/d=0, short=r=0/w=0/d=0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
  READ: io=10240MB, aggrb=154129KB/s, minb=154129KB/s, maxb=154129KB/s, mint=68032msec, max
t=68032msec
```

Sequential write bandwidth

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=1M --iodepth=128 --
size=10240M --readwrite=write --fallocate=none
```

 NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (groupid=0, jobs=1): err= 0: pid=21009: Mon Jun 8 14:53:44 2020
write: IOPS=243, bw=244MiB/s (255MB/s)(10.0GiB/42048msec)
slat (usec): min=103, max=504, avg=190.30, stdev=29.47
clat (msec): min=18, max=1104, avg=525.23, stdev=253.35
lat (msec): min=18, max=1104, avg=525.42, stdev=253.35
clat percentiles (msec):
| 1.00th=[ 51], 5.00th=[ 108], 10.00th=[ 167], 20.00th=[ 292],
| 30.00th=[ 422], 40.00th=[ 468], 50.00th=[ 506], 60.00th=[ 550],
| 70.00th=[ 625], 80.00th=[ 768], 90.00th=[ 902], 95.00th=[ 970],
| 99.00th=[ 1036], 99.50th=[ 1045], 99.90th=[ 1070], 99.95th=[ 1099],
| 99.99th=[ 1099]
bw ( KiB/s): min= 4096, max=468992, per=100.00%, avg=249500.99, stdev=147656.62, samples=83
iops      : min=   4, max=  458, avg=243.63, stdev=144.22, samples=83
lat (msec): 20=0.03%, 50=0.96%, 100=3.36%, 250=12.55%, 500=31.63%
lat (msec): 750=30.07%, 1000=18.96%
cpu       : usr=2.28%, sys=2.50%, ctx=3972, majf=0, minf=27
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.2%, 32=0.3%, >=64=99.4%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued rwts: total=0,10240,0,0 short=0,0,0,0 dropped=0,0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
WRITE: bw=244MiB/s (255MB/s), 244MiB/s-244MiB/s (255MB/s-255MB/s), io=10.0GiB (10.7GB), run=42048-42048msec
```

Random write bandwidth

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/sfs-turbo/test_fio --bs=1M --iodepth=128 --
size=10240M --readwrite=randwrite --fallocate=none
```

 NOTE

`/mnt/sfs-turbo/test_fio` indicates the location of the file to be tested. The location must be specific to the file name, which is the `test_fio` file in the `/mnt/sfs-turbo` directory in this example. Set it based on the site requirements.

- fio result:

```
test: (g=0): rw=randwrite, bs=1M-1M/1M-1M/1M-1M, ioengine=libaio, iodepth=128
fio-2.1.10
Starting 1 process

test: (groupid=0, jobs=1): err= 0: pid=16370: Tue Dec 28 09:22:59 2021
write: io=10240MB, bw=15600KB/s, iops=152, runt= 67216msec
slat (usec): min=93, max=349, avg=156.14, stdev=22.29
clat (msec): min=17, max=1964, avg=839.92, stdev=345.94
lat (msec): min=17, max=1964, avg=840.08, stdev=345.94
clat percentiles (msec):
| 1.00th=[ 30], 5.00th=[ 37], 10.00th=[ 42], 20.00th=[ 97],
| 30.00th=[ 97], 40.00th=[ 98], 50.00th=[ 98], 60.00th=[ 99],
| 70.00th=[ 99], 80.00th=[ 100], 90.00th=[ 100], 95.00th=[ 101],
| 99.00th=[ 102], 99.50th=[ 102], 99.90th=[ 103], 99.95th=[ 104],
| 99.99th=[ 105]
bw (KB /s): min=150104, max=180654, per=98.76%, avg=154058.04, stdev=3404.48
lat (msec): 20=0.04%, 50=13.44%, 100=1.04%, 250=0.73%, 500=1.05%
lat (msec): 750=0.04%, 1000=60.69%, 2000=22.97%
cpu       : usr=0.91%, sys=1.52%, ctx=2011, majf=0, minf=28
IO depths : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.2%, 32=0.3%, >=64=99.4%
submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
issued r/w: total=r=0/w=10240/d=0, short=r=0/w=0/d=0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
WRITE: io=10240MB, aggrbw=15600KB/s, minbw=15600KB/s, maxbw=15600KB/s, mint=67216msec, maxt=67216msec
```

10.2 Mounting a File System to a Linux ECS as a Non-root User

Scenarios

By default, a Linux ECS allows only the **root** user to run the **mount** command for mounting a file system. However, if the permissions of user **root** are assigned to other common users, such users can also run the **mount** command for file system mounting. The following describes how to mount a file system to a Linux ECS as a common user. The EulerOS is used as an example.

Prerequisites

- A non-**root** user has been created on the ECS.
- A file system has been created and can be mounted to the ECS by the **root** user.
- You have obtained the mount point of the file system.

Procedure

Step 1 Log in to the ECS as user **root**.

Step 2 Assign the permissions of user **root** to the non-**root** user.

1. Run the **chmod 777 /etc/sudoers** command to change the **sudoers** file to be editable.
2. Use the **which** command to view the **mount** and **umount** command paths.

Figure 10-1 Viewing command paths

```
root@ecs-os-45df /]#  
root@ecs-os-45df /]#  
root@ecs-os-45df /]#  
root@ecs-os-45df /]#  
root@ecs-os-45df /]# which mount  
/usr/bin/mount  
root@ecs-os-45df /]# which umount  
/usr/bin/umount  
root@ecs-os-45df /]#
```

3. Run the **vi /etc/resolv.conf** command to edit the **sudoers** file.
4. Add a common user under the **root** account. In the following figure, user **Mike** is added.

Figure 10-2 Adding a user

```
# Defaults    env_keep += "HOME"

Defaults    secure_path = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##     user    MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
mike    ALL=(ALL)    NOPASSWD: /usr/bin/mount
mike    ALL=(ALL)    NOPASSWD: /usr/bin/umount

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)    ALL

## Same thing without a password
# %wheel    ALL=(ALL)    NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users   ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users   localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
```

5. Press **Esc**, input **:wq**, and press **Enter** to save and exit.
6. Run the **chmod 440 /etc/sudoers** command to change the **sudoers** file to be read-only.

Step 3 Log in to the ECS as user **Mike**.

Step 4 Run the following command to mount the file system. For details about the mounting parameters, see [Table 10-2](#).

sudo mount -t nfs -o vers=3,timeo=600,noresvport,nolock *Mount point Local path*

Table 10-2 Parameter description

Parameter	Description
<i>Mount point</i>	The format of an SFS Turbo file system is <i>File system IP address./</i> , for example, 192.168.0.0:/ . NOTE x is a digit or letter. If the mount point is too long to display completely, you can adjust the column width.
<i>Local path</i>	Local path on the ECS, used to mount the file system, for example, /local_path .

Step 5 Run the following command to view the mounted file system:

mount -l

If the command output contains the following information, the file system has been mounted.


```
example.com:/share-xxx on /local_path type nfs (rw,vers=3,timeo=600,nolock,addr=)
```

----End

10.3 Mounting a Subdirectory of an NFS File System to ECSs (Linux)

This section describes how to mount a subdirectory of an NFS file system to Linux ECSs.

Prerequisites

You have mounted a file system to Linux ECSs by referring to [Mounting an NFS File System to ECSs \(Linux\)](#).

Procedure

Step 1 Run the following command to create a subdirectory in the local path:

```
mkdir Local_path/Subdirectory
```

NOTE

Variable *Local_path* is an ECS local directory where the file system will be mounted on, for example, **/local_path**. Specify the local path used for mounting the root directory.

Step 2 Run the following command to mount the subdirectory to the ECSs that are in the same VPC as the file system: (Currently, the file system can be mounted to Linux ECSs using NFS v3 only.)

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock Domain name or IP  
address of the file system:/Subdirectory Local_path
```

NOTE

- *Domain name or IP address of the file system*: You can obtain it in the file system list from the console.
 - SFS Turbo: *xx.xx.xx.xx:/subdirectory*
- *Subdirectory*: Specify the subdirectory created in the previous step.
- *Local_path*: An ECS local directory where the file system is mounted, for example, **/local_path**. Specify the local path used for mounting the root directory.

Step 3 Run the following command to view the mounted file system:

```
mount -l
```

If the command output contains the following information, the file system has been mounted.

```
Mount point on /local_path type nfs (rw,vers=3,timeo=600,nolock,addr=)
```

Step 4 After the subdirectory has been mounted, you can access it from the server, and read or write data.

----End

Troubleshooting

If a subdirectory is not created before mounting, the mounting will fail.

Figure 10-3 Mounting without a subdirectory created

```
[root@ecs-eos-0891 workstation]# mount -t nfs -o nolock,vers=3 [redacted] -vvv
mount.nfs: timeout set for Sun Oct 24 20:44:13 2021
mount.nfs: trying text-based options 'noLOCK,vers=3,addr=[redacted]'
mount.nfs: prog 100003, trying vers=3, prot=6
mount.nfs: trying [redacted] prog 100003 vers 3 prot TCP port 2049
mount.nfs: prog 100005, trying vers=3, prot=17
mount.nfs: trying [redacted] prog 100005 vers 3 prot UDP port 20048
mount.nfs: mount(2): Permission denied
mount.nfs: access denied by server while mounting [redacted] :/subdir
```

In the preceding figure, the root directory does not have the **subdir** subdirectory created so that the mounting fails. In this case, error message "Permission denied" is reported.

To troubleshoot this issue, mount the root directory, create a subdirectory, and then mount the subdirectory.

Figure 10-4 Mounting subdirectory

```
[root@ecs-eos-0891 workstation]# mount -t nfs -o noLOCK,vers=3 [redacted] .82:/mnt/sfsturbo -vvv
mount.nfs: timeout set for Sun Oct 24 20:47:26 2021
mount.nfs: trying text-based options 'noLOCK,vers=3,addr=[redacted].82' Mount the root directory.
mount.nfs: prog 100003, trying vers=3, prot=6
mount.nfs: trying [redacted].82 prog 100003 vers 3 prot TCP port 2049
mount.nfs: prog 100005, trying vers=3, prot=17
mount.nfs: trying [redacted].82 prog 100005 vers 3 prot UDP port 20048
[root@ecs-eos-0891 workstation]# mkdir /mnt/sfsturbo/subdir Create a subdirectory.
[root@ecs-eos-0891 workstation]# umount /mnt/sfsturbo
[root@ecs-eos-0891 workstation]# mount -t nfs -o noLOCK,vers=3 [redacted] .82:/subdir /mnt/sfsturbo -vvv
mount.nfs: timeout set for Sun Oct 24 20:47:50 2021
mount.nfs: trying text-based options 'noLOCK,vers=3,addr=[redacted].82' Mount the subdirectory.
mount.nfs: prog 100003, trying vers=3, prot=6
mount.nfs: trying [redacted].82 prog 100003 vers 3 prot TCP port 2049
mount.nfs: prog 100005, trying vers=3, prot=17
mount.nfs: trying [redacted].82 prog 100005 vers 3 prot UDP port 20048
[root@ecs-eos-0891 workstation]#
```

10.4 Data Migration

10.4.1 Migration Description

By default, an SFS Turbo file system can only be accessed by ECSs or CCE instances that reside in the same VPC as the file system. To access an SFS Turbo file system from an on-premises data center or a different VPC, you need to establish network connections by using Direct Connect, VPN, or VPC peering connections.

- Access from on premises or another cloud: Use Direct Connect or VPN.
- On-cloud, cross-VPC access using the same account in a given region: Use VPC peering.
- On-cloud, cross-account access in a given region: Use VPC peering.
- On-cloud, cross-region access: Use Cloud Connect.

You can migrate data to SFS Turbo using an ECS that can access the Internet.

- Mount the SFS Turbo file system to the ECS and migrate data from the local NAS storage to the SFS Turbo file system.

[10.4.2 Using Direct Connect to Migrate Data](#)

- If communication cannot be enabled through file system mounting, migrate data using the Huawei Cloud ECS via the Internet.

[10.4.3 Using the Internet to Migrate Data](#)

10.4.2 Using Direct Connect to Migrate Data

Context

You can migrate data from a local NAS to SFS Turbo using Direct Connect.

In this solution, a Linux ECS is created to connect the local NAS and SFS Turbo, and data is migrated to the cloud using an ECS.

You can also refer to this solution to migrate data from an on-cloud NAS to SFS Turbo. For details, see [Migrating Data from On-cloud NAS to SFS](#).

Limitations and Constraints

- Only Linux ECSs can be used to migrate data.
- The UID and GID of your file will no longer be consistent after data migration.
- The file access modes will no longer be consistent after data migration.
- Incremental migration is supported, so that only changed data is migrated.

Prerequisites

- You have enabled and configured Direct Connect. For details, see *Direct Connect User Guide*.
- You have created a Linux ECS.
- You have created an SFS Turbo file system and have obtained the mount point of the file system.
- You have obtained the mount point of the local NAS.

Procedure

Step 1 Log in to the ECS console.

Step 2 Log in to the created Linux ECS to access the local NAS and SFS Turbo file system.

Step 3 Run the following mount command to access the local NAS:

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock Mount point of the local NAS /mnt/src
```

Step 4 Run the following mount command to access the file system:

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock Mount point of the file system /mnt/dst
```

Step 5 Run the following commands on the Linux ECS to install the rclone tool:

```
wget https://downloads.rclone.org/v1.53.4/rclone-v1.53.4-linux-amd64.zip --no-check-certificate
unzip rclone-v1.53.4-linux-amd64.zip
chmod 0755 ./rclone-*/rclone
cp ./rclone-*/rclone /usr/bin/
rm -rf ./rclone-*
```

Step 6 Run the following command to synchronize data:

```
rclone copy /mnt/src /mnt/dst -P --transfers 32 --checkers 64 --links --create-empty-src-dirs
```

 **NOTE**

Set **transfers** and **checkers** based on the system specifications. The parameters are described as follows:

- **--transfers**: number of files that can be transferred concurrently
- **--checkers**: number of local files that can be scanned concurrently
- **-P**: data copy progress
- **--links**: replicates the soft links from the source. They are saved as soft links in the destination.
--copy-links: replicates the content of files to which the soft links point. They are saved as files rather than soft links in the destination.
- **--create-empty-src-dirs**: replicates the empty directories from the source to the destination.

After data synchronization is complete, go to the target file system to check whether data is migrated.

----End

Migrating Data from On-cloud NAS to SFS

To migrate data from an on-cloud NAS to your SFS Turbo file system, ensure that the NAS and file system are in the same VPC, or you can use Cloud Connect to migrate data.

For details about how to configure Cloud Connect, see *Direct Connect User Guide*.

10.4.3 Using the Internet to Migrate Data

Context

You can migrate data from a local NAS to SFS Turbo using the Internet.

In this solution, to migrate data from the local NAS to the cloud, a Linux server is created both on the cloud and on-premises. Inbound and outbound traffic is allowed on port 22 of these two servers. The on-premises server is used to access the local NAS, and the ECS is used to access SFS Turbo.

You can also refer to this solution to migrate data from an on-cloud NAS to SFS Turbo.

Limitations and Constraints

- Data cannot be migrated from the local NAS to SFS Capacity-Oriented using the Internet.
- Only Linux ECSs can be used to migrate data.
- The UID and GID of your file will no longer be consistent after data migration.
- The file access modes will no longer be consistent after data migration.
- Inbound and outbound traffic must be allowed on port 22.
- Incremental migration is supported, so that only changed data is migrated.

Prerequisites

- A Linux server has been created on the cloud and on-premises respectively.
- EIPs have been configured for the servers to ensure that the two servers can communicate with each other.
- You have created an SFS Turbo file system and have obtained the mount point of the file system.
- You have obtained the mount point of the local NAS.

Procedure

Step 1 Log in to the ECS console.

Step 2 Log in to the created on-premises server **client1** and run the following command to access the local NAS:

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock Mount point of the local NAS /mnt/src
```

Step 3 Log in to the created Linux ECS **client2** and run the following command to access the SFS Turbo file system:

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock Mount point of the SFS Turbo file system /mnt/dst
```

Step 4 Run the following commands on **client1** to install the rclone tool:

```
wget https://downloads.rclone.org/v1.53.4/rclone-v1.53.4-linux-amd64.zip --no-check-certificate
unzip rclone-v1.53.4-linux-amd64.zip
chmod 0755 ./rclone-*/rclone
cp ./rclone-*/rclone /usr/bin/
rm -rf ./rclone-*
```

Step 5 Run the following commands on **client1** to configure the environment:

```
rclone config
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> remote name (New name)
Type of storage to configure.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
24 / SSH/SFTP Connection
 \ "sftp"
Storage> 24 (Select the SSH/SFTP number)
SSH host to connect to
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
1 / Connect to example.com
 \ "example.com"
host> ip address (IP address of client2)
SSH username, leave blank for current username, root
Enter a string value. Press Enter for the default ("").
user> user name (Username of client2)
SSH port, leave blank to use default (22)
Enter a string value. Press Enter for the default ("").
port> 22
SSH password, leave blank to use ssh-agent.
y) Yes type in my own password
g) Generate random password
n) No leave this optional password blank
y/g/n> y
Enter the password:
password: (Password for logging in to client2)
Confirm the password:
password: (Confirm the password for logging in to client2)
```

```

Path to PEM-encoded private key file, leave blank or set key-use-agent to use ssh-agent.
Enter a string value. Press Enter for the default ("").
key_file> (Press Enter)
The passphrase to decrypt the PEM-encoded private key file.

Only PEM encrypted key files (old OpenSSH format) are supported. Encrypted keys
in the new OpenSSH format can't be used.
y) Yes type in my own password
g) Generate random password
n) No leave this optional password blank
y/g/n> n
When set forces the usage of the ssh-agent.
When key-file is also set, the ".pub" file of the specified key-file is read and only the associated key is
requested from the ssh-agent. This allows to avoid `Too many authentication failures for *username*` errors
when the ssh-agent contains many keys.
Enter a boolean value (true or false). Press Enter for the default ("false").
key_use_agent> (Press Enter)
Enable the use of the aes128-cbc cipher. This cipher is insecure and may allow plaintext data to be
recovered by an attacker.
Enter a boolean value (true or false). Press Enter for the default ("false").
Choose a number from below, or type in your own value
 1 / Use default Cipher list.
  \ "false"
 2 / Enables the use of the aes128-cbc cipher.
  \ "true"
use_insecure_cipher> (Press Enter)
Disable the execution of SSH commands to determine if remote file hashing is available.
Leave blank or set to false to enable hashing (recommended), set to true to disable hashing.
Enter a boolean value (true or false). Press Enter for the default ("false").
disable_hashcheck>
Edit advanced config? (y/n)
y) Yes
n) No
y/n> n
Remote config
-----
[remote_name]
type = sftp
host=(client2 ip)
user=(client2 user name)
port = 22
pass = *** ENCRYPTED ***
key_file_pass = *** ENCRYPTED ***
-----
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
Current remotes:

Name          Type
====          ====
remote_name    sftp

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> q

```

Step 6 Run the following command to view the **rclone.conf** file in **/root/.config/rclone/rclone.conf**:

```

cat /root/.config/rclone/rclone.conf
[remote_name]
type = sftp
host=(client2 ip)

```

```
user=(client2 user name)
port = 22
pass = ***
key_file_pass = ***
```

Step 7 Run the following command on **client1** to synchronize data:

```
rclone copy /mnt/src remote_name:/mnt/dst -P --transfers 32 --checkers 64
```

NOTE

- Replace *remote_name* in the command with the remote name in the environment.
- Set **transfers** and **checkers** based on the system specifications. The parameters are described as follows:
 - **transfers**: number of files that can be transferred concurrently
 - **checkers**: number of local files that can be scanned concurrently
 - **P**: data copy progress

After data synchronization is complete, go to the SFS Turbo file system to check whether data is migrated.

----End

10.4.4 Migrating Data Between File Systems

Solution Overview

You can migrate data from an SFS Capacity-Oriented file system to an SFS Turbo file system or the other way around.

This solution creates a Linux ECS to connect an SFS Capacity-Oriented file system with an SFS Turbo file system.

Limitations and Constraints

- Only Linux ECSs can be used to migrate data.
- The Linux ECS, SFS Capacity-Oriented file system, and SFS Turbo file system must be in the same VPC.
- Incremental migration is supported, so that only changed data is migrated.

Prerequisites

- You have created a Linux ECS.
- You have created an SFS Capacity-Oriented file system and an SFS Turbo file system and have obtained their mount points.

Procedure

Step 1 Log in to the ECS console.

Step 2 Log in to the created Linux ECS that can access SFS Capacity-Oriented and SFS Turbo file systems.

Step 3 Run the following command to mount file system 1 (either the SFS Capacity-Oriented or SFS Turbo file system). After that, you can access file system 1 on the Linux ECS.

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock [Mount point of file system 1] /mnt/src
```

Step 4 Run the following command to mount file system 2 (the other file system that you have not mounted in the previous step). After that, you can access file system 2 on the Linux ECS.

```
mount -t nfs -o vers=3,timeo=600,noresvport,nolock [Mount point of file system 2] /mnt/dst
```

Step 5 Run the following commands on the Linux ECS to install the rclone tool:

```
wget https://downloads.rclone.org/v1.53.4/rclone-v1.53.4-linux-amd64.zip --no-check-certificate
unzip rclone-v1.53.4-linux-amd64.zip
chmod 0755 ./rclone-*/rclone
cp ./rclone-*/rclone /usr/bin/
rm -rf ./rclone-*
```

Step 6 Run the following command to synchronize data:

```
rclone copy /mnt/src /mnt/dst -P --transfers 32 --checkers 64 --links --create-empty-src-dirs
```

NOTE

Set **transfers** and **checkers** based on the system specifications. The parameters are described as follows:

- **--transfers**: number of files that can be transferred concurrently
- **--checkers**: number of local files that can be scanned concurrently
- **-P**: data copy progress
- **--links**: replicates the soft links from the source. They are saved as soft links in the destination.
--copy-links: replicates the content of files to which the soft links point. They are saved as files rather than soft links in the destination.
- **--create-empty-src-dirs**: replicates the empty directories from the source to the destination.

After data synchronization is complete, go to the target file system to check whether data is migrated.

----End

Verification

Step 1 Log in to the created Linux ECS.

Step 2 Run the following commands on the destination server to verify file synchronization:

```
cd /mnt/dst
ls | wc -l
```

Step 3 If the data volume is the same as that on the source server, the data is migrated successfully.

----End

A Change History

Released On	Description
2023-07-27	This issue is the second official release, which incorporates the following change: Updated and added constraints in sections 3 Capacity Expansion and 6 Backup .
2022-09-30	This issue is the first official release.