

Scalable File Service

Best Practices

Issue 01
Date 2024-11-29



Copyright © Huawei Technologies Co., Ltd. 2024. 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.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:

<https://www.huawei.com/en/psirt/vul-response-process>

For vulnerability information, enterprise customers can visit the following web page:

<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

Contents

1 Migrating Data to SFS.....	1
1.1 Solution Overview.....	1
1.2 Using Direct Connect to Migrate Data (rclone).....	1
1.3 Cross-server Migration (rclone).....	3
1.4 Using Direct Connect to Migrate Data (rsync).....	7
1.5 Migrating Data Between File Systems.....	10
1.6 Migrating Data from SFS Capacity-Oriented to Another Type of File Systems.....	12
2 Testing SFS Turbo Performance.....	14
3 Testing SFS Turbo Latency.....	22
4 Creating a Readable and Writable Subdirectory on the File System for a Common User.....	25
4.1 Solution Overview.....	25
4.2 Resource and Cost Planning.....	26
4.3 Implementation Procedure.....	27
4.3.1 Creating a Local Directory for a File System.....	27
4.3.2 Creating a Readable and Writable Subdirectory on the File System for Each User.....	30

1 Migrating Data to SFS

1.1 Solution Overview

By default, an SFS Turbo file system can only be accessed by ECSs or CCE containers 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.
- Access from a different VPC under the same account and in the same region: Use VPC peering.
- Access from a different account in the same region: Use VPC peering.
- Access from a different region: Use Cloud Connect.

Data can be migrated to SFS Turbo by using an ECS that can access the Internet.

- Mount an SFS Turbo file system to an ECS and migrate data from the local Network Attached Storage (NAS) to the SFS Turbo file system.
 - [Using Direct Connect to Migrate Data \(rclone\)](#)
 - [Using Direct Connect to Migrate Data \(rsync\)](#)
- If communication cannot be enabled through file system mounting, migrate data using the ECS via the Internet.
[Cross-server Migration \(rclone\)](#)

1.2 Using Direct Connect to Migrate Data (rclone)

Solution Overview

You can migrate data from a local NAS to SFS Turbo using Direct Connect and the rclone tool.

In this solution, a Linux ECS is created to connect the local NAS and SFS Turbo, and data is migrated to the cloud using this 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.
- If data is written to the file system after you have run the rclone command to migrate data, data inconsistency may occur.

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.

Resource Planning

[Table 1-1](#) describes the resource planning in this solution.

Table 1-1 Resource planning

Resource	Example Configuration	Description
ECS	Specifications: 8 vCPUs 16 GB c7.2xlarge.2 OS: Linux Region: CN-Hong Kong VPC: VPC1	Ensure that the <code>/mnt/src</code> and <code>/mnt/dst</code> directories have been created.

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 SFS Turbo 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**

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

- **--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 SFS Turbo 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

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 [Cloud Connect User Guide](#).

1.3 Cross-server Migration (rclone)

Solution Overview

You can use rclone to migrate data from a local NAS to SFS Turbo over the Internet or private network.

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 over the Internet or private network.

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.
- If data is written to the file system after you have run the rclone command to migrate data, data inconsistency may occur.

Prerequisites

- A Linux server has been created on the cloud and on-premises respectively.
- Elastic IP addresses (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.

Resource Planning

[Table 1-2](#) describes the resource planning in this solution.

Table 1-2 Resource planning

Resource	Example Configuration	Description
ECS	Specifications: 8 vCPUs 16 GB c7.2xlarge.2 OS: Linux Region: CN-Hong Kong VPC: VPC1 Enabled port: 22 EIP: xxx.xxx.xxx.xxx	Ensure that the /mnt/dst directory has been created.

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)
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
```

NOTE

The IP address of **client2** is a public IP address.

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

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

1.4 Using Direct Connect to Migrate Data (rsync)

Solution Overview

You can migrate data from a local NAS to SFS Turbo using Direct Connect and the rsync tool.

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

You can also refer to this solution to migrate data from an on-cloud NAS to SFS Turbo using the Internet. Ensure that the on-cloud NAS and SFS Turbo belong to the same VPC.

Limitations and Constraints

- Special files, such as devices and linked files, can be migrated.
- Resumable data transfer is supported.
- Properties, such as permissions, time, soft and hard links, owner, and group, of the original files and directories can be retained after data migration.
- The rcp, rsh, and ssh tools are supported during file transfer.
- Incremental migration is supported, so that only changed data is migrated.
- If there are multi-level directories or massive small files, you are recommended to use a multi-process script or the rclone tool. This is because the rsync tool has low efficiency in these scenarios.

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.
- A trust relationship has been established between the local NAS and the ECS.

Resource Planning

Table 1-3 describes the resource planning in this solution.

Table 1-3 Resource planning

Resource	Example Configuration	Description
ECS	Specifications: 8 vCPUs 16 GB c7.2xlarge.2 OS: Linux Region: CN-Hong Kong VPC: VPC1	Ensure that the <code>/mnt/src</code> and <code>/mnt/dst</code> directories have been created.

Procedure

Step 1 Log in to the created Linux ECS to access the local NAS and SFS Turbo file system.

Step 2 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 3 Run the following mount command to access the SFS Turbo file system:

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

Step 4 Run the following command on the Linux ECS to install the rsync tool:

```
yum install rsync
```

Figure 1-1 Installing rsync

```
[root@ ~]# yum install rsync
Loaded plugins: fastestmirror
Determining fastest mirrors
epel/x86_64/metalink
* base: mirrors.huaweicloud.com
* epel: mirrors.bfsu.edu.cn
* extras: mirrors.ustc.edu.cn
* updates: mirrors.huaweicloud.com
```

NOTE

Ensure that the rsync tool is installed on both the source and destination servers. Or, an error will be reported.

Step 5 After the installation is complete, run the following command to query the installation result and version of rsync:

```
rsync -version
```

Figure 1-2 Viewing the installation result

```
[root@          ]# rsync -version
rsync version 3.1.2 protocol version 31
Copyright (C) 1996-2015 by Andrew Tridgell, Wayne Davison, and others.
Web site: http://rsync.samba.org/
Capabilities:
 64-bit files, 64-bit inums, 64-bit timestamps, 64-bit long ints,
 socketpairs, hardlinks, symlinks, IPv6, batchfiles, inplace,
 append, ACLs, xattrs, iconv, symtimes, prealloc

rsync comes with ABSOLUTELY NO WARRANTY. This is free software, and you
are welcome to redistribute it under certain conditions. See the GNU
General Public Licence for details.

rsync is a file transfer program capable of efficient remote update
via a fast differencing algorithm.

Usage: rsync [OPTION]... SRC [SRC]... DEST
      or rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
      or rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
      or rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
      or rsync [OPTION]... [USER@]HOST:SRC [DEST]
      or rsync [OPTION]... [USER@]HOST::SRC [DEST]
      or rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
The ':' usages connect via remote shell, while '::' & 'rsync://' usages connect
to an rsync daemon, and require SRC or DEST to start with a module name.
```

Step 6 Run the following command to migrate the full data in the `/mnt/src` directory on the source server to the `/mnt/dst` directory (file system) on the destination server:

```
rsync -avP /mnt/src /mnt/dst
```

Figure 1-3 Full data synchronization

```
103669.dat
 1,024 100% 38.46kB/s 0:00:00 (xfr#4080, to-chk=295920/300001)
10367.dat
 1,024 100% 37.04kB/s 0:00:00 (xfr#4081, to-chk=295919/300001)
103670.dat
 1,024 100% 37.04kB/s 0:00:00 (xfr#4082, to-chk=295918/300001)
103671.dat
 1,024 100% 35.71kB/s 0:00:00 (xfr#4083, to-chk=295917/300001)
103672.dat
 1,024 100% 35.71kB/s 0:00:00 (xfr#4084, to-chk=295916/300001)
103673.dat
 1,024 100% 34.48kB/s 0:00:00 (xfr#4085, to-chk=295915/300001)
103674.dat
 1,024 100% 34.48kB/s 0:00:00 (xfr#4086, to-chk=295914/300001)
103675.dat
 1,024 100% 33.33kB/s 0:00:00 (xfr#4087, to-chk=295913/300001)
```

----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

1.5 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.

Resource Planning

[Table 1-4](#) describes the resource planning in this solution.

Table 1-4 Resource planning

Resource	Example Configuration	Description
ECS	Specifications: 8 vCPUs 16 GB c7.2xlarge.2 OS: Linux Region: CN-Hong Kong VPC: VPC1	Ensure that the /mnt/src and /mnt/dst directories have been created.

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 Download and install rclone. For the download address, see <https://rclone.org/downloads/>.

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:

- **/mnt/src**: source path
- **/mnt/dst**: destination path
- **--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

1.6 Migrating Data from SFS Capacity-Oriented to Another Type of File Systems

Solution Overview

You can migrate data from an SFS Capacity-Oriented file system to a General Purpose File System or an SFS Turbo file system.

In this solution, a Linux ECS is used to connect the SFS Capacity-Oriented file system and the destination file system.

Notes and Constraints

- Only Linux ECSs can be used to migrate data.
- The Linux ECS, SFS Capacity-Oriented file system, and the destination file system must be in the same VPC. If the destination file system is a General Purpose File System, you need to configure a VPC endpoint. For details, see [Configure a VPC Endpoint](#).
- Incremental migration is supported, so you can only migrate the changed data.

Prerequisites

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

Resource Planning

[Table 1-5](#) describes the resource planning in this solution.

Table 1-5 Resource planning

Resource	Example Configuration	Description
ECS	Specifications: 8 vCPUs 16 GB c7.2xlarge.2 OS: Linux Region: CN-Hong Kong VPC: VPC1	Ensure that the /mnt/src and /mnt/dst directories have been created.

Procedure

Step 1 Log in to the ECS console.

Step 2 Log in to the created Linux ECS, which can access both the SFS Capacity-Oriented and the destination file systems.

Step 3 Mount the SFS Capacity-Oriented file system, which is *file system 1* in this example.

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

Step 4 Mount the General Purpose File System or SFS Turbo file system, which *file system 2* in this example.

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

Step 5 Install the rclone tool on the Linux ECS.

```
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-*
```

NOTE

The rclone tool does not retain the file permissions or owner group information on the source. Use the rsync tool if you have such requirements.

Step 6 Synchronize data to the destination file system.

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

NOTE

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

- **/mnt/src**: source path
- **/mnt/dst**: destination path
- **--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 destination file system to check whether data is migrated.

----End

Verification

Step 1 Log in to the Linux ECS.

Step 2 Check the file synchronization results in the destination file system.

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

If the data volume is the same as that in the source file system, data is migrated successfully.

----End

2 Testing SFS Turbo Performance

Fio is an open-source I/O tester. You can use fio to test the throughput and IOPS of SFS Turbo file systems.

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 2-1](#).

Table 2-1 File system performance data

Parameter	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)] Capacity unit: GB	IOPS = Min. [20,000, (1,500 + 20 x Capacity)] Capacity unit: GB	IOPS = Min. (1,000,000, 6,000 x Capacity) Capacity unit: TB	IOPS = Min. (1,000,000, 12,500 x Capacity) Capacity unit: TB

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

- fio command:

```
fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --direct=1 --filename=/mnt/nfs/test_fio --bs=1M --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=397477, avg=4145.45, stdev=3344.40
  clat (usec): min=245, max=133325, avg=11162.10, stdev=12136.31
  lat (usec): min=252, max=133330, avg=11166.32, stdev=12136.34
  clat percentiles (usec):
  | 1.00th=[ 2245],  5.00th=[ 2540], 10.00th=[ 2671], 20.00th=[ 2901],
  | 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=[98042]
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=1480, max=302730, avg=4613.59, stdev=3359.60
  clat (usec): min=1447, max=140666, avg=14166.05, stdev=13373.72
  lat (usec): 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=3182.70, stdev=590.96, samples=494
lat (usec) : 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
  
```

- fio command:

```

fio --randrepeat=1 --ioengine=libaio --name=test -output=output.log --
direct=1 --filename=/mnt/nfs/test_fio --bs=1M --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=20350: Mon Jun 0 11:57:14 2020
read: IOPS=5065, BW=19.8MiB/s (20.7MB/s)(3073MiB/155200msec)
  slat (msec): min=1271, max=269500, avg=4073.51, stdev=3048.12
  clat (usec): min=226, max=80105, avg=5711.35, stdev=7079.46
  lat (usec): min=232, max=80107, avg=5715.49, stdev=7079.40
  clat percentiles (usec):
  | 1.00th=[ 1221],  5.00th=[ 1950], 10.00th=[ 2100], 20.00th=[ 2442],
  | 30.00th=[ 2606], 40.00th=[ 2802], 50.00th=[ 2999], 60.00th=[ 3220],
  | 70.00th=[ 3607], 80.00th=[ 5604], 90.00th=[14222], 95.00th=[21000],
  | 99.00th=[35914], 99.50th=[40633], 99.90th=[51643], 99.95th=[55037],
  | 99.99th=[66047]
bw ( KIB/s): min=13360, max=28040, per=99.99%, avg=20257.97, stdev=2913.05, samples=310
iops      : min= 3340, max= 7212, avg=5064.48, stdev=720.27, samples=310
write: IOPS=11.0k, BW=46.2MiB/s (48.4MB/s)(7167MiB/155200msec)
  slat (msec): min=1396, max=390604, avg=4405.68, stdev=3091.75
  clat (usec): min=857, max=140259, avg=8377.47, stdev=8400.15
  lat (usec): min=867, max=140264, avg=8382.02, stdev=8400.16
  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=[ 201],
  | 99.00th=[ 421], 99.50th=[ 471], 99.90th=[ 601], 99.95th=[ 601],
  | 99.99th=[ 1201]
bw ( KIB/s): min=32224, max=67456, per=99.90%, avg=47254.23, stdev=6792.41, samples=310
iops      : min= 8056, max=16064, avg=11013.55, stdev=1690.11, samples=310
lat (usec) : 250=0.01%, 500=0.04%, 750=0.07%, 1000=0.03%
lat (msec)  : 2=1.53%, 4=36.85%, 10=41.27%, 20=11.30%, 50=0.61%
lat (msec)  : 100=0.23%, 250=0.01%
cpu         : usr=2.13%, sys=9.90%, ctx=925770, 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%, 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=706597,1034043,0,0 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=19.8MiB/s (20.7MB/s), 19.8MiB/s-19.8MiB/s (20.7MB/s-20.7MB/s), io=3073MiB (3222MB), run=155200-155200msec
  WRITE: bw=46.2MiB/s (48.4MB/s), 46.2MiB/s-46.2MiB/s (48.4MB/s-48.4MB/s), io=7167MiB (7516MB), run=155200-155200msec
  
```

Sequential read IOPS

- fio command:

```
fio --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --rw=read --bs=4k --size=1G --iodepth=128 --runtime=120 --numjobs=10
```

 **NOTE**

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- 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)(18.0GiB/271519msec)
slat (usec): min=1233, max=662160, avg=4118.17, stdev=4773.23
clat (usec): min=365, max=131116, avg=13253.18, stdev=13958.09
lat (usec): min=371, max=131118, avg=13257.29, stdev=13958.09
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=[ 16909], 80.00th=[ 23462], 90.00th=[ 33162], 95.00th=[ 41681],
| 99.00th=[ 59507], 99.50th=[ 66847], 99.90th=[ 83362], 99.95th=[ 98702],
| 99.99th=[103285]
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=0.01%, 750=0.01%, 1000=0.02%
lat (msec)  : 2=5.25%, 4=36.35%, 10=12.76%, 20=20.56%, 50=22.62%
lat (msec)  : 100=2.42%, 250=0.02%
cpu         : usr=1.04%, 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=2621440,0,0,0 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=18.0GiB (18.7GB), run=2
```

Random read IOPS

- fio command:

```
fio --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --rw=randread --bs=4k --size=1G --iodepth=128 --runtime=120 --numjobs=10
```

 **NOTE**

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- 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, agrbw=18597KB/s, minb=18597KB/s, maxb=18597KB/s, mint=563832msec, maxt=563832msec
```

Sequential write IOPS

- fio command:

```
fio --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --rw=write --bs=4k --size=1G --iodepth=128 --runtime=120 --numjobs=10
```

NOTE

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- 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=360726, avg=4300.87, stdev=3600.87
  clat (usec): min=1953, max=106548, avg=11588.61, stdev=5076.84
    lat (usec): min=1959, max=106552, avg=11593.06, stdev=5076.86
  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=[10014],
    | 70.00th=[12510], 80.00th=[15533], 90.00th=[19268], 95.00th=[22676],
    | 99.00th=[32637], 99.50th=[37487], 99.90th=[49021], 99.95th=[53740],
    | 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=13107, avg=11039.50, stdev=996.03, samples=474
  lat (msec) : 2=0.01%, 4=1.00%, 10=51.94%, 20=30.58%, 50=0.39%
  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=0,2621440,0,0 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=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 --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --rw=randwrite --bs=4k --size=1G --iodepth=128 --runtime=120 --numjobs=10
```

NOTE

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- 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=398, BW=391MiB/s (409MB/s)(18.0GiB/26221msec)
slat (usec): min=78, max=595, avg=99.58, stdev=39.89
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=[ 188],
| 30.00th=[ 368], 40.00th=[ 372], 50.00th=[ 388], 60.00th=[ 384],
| 70.00th=[ 388], 80.00th=[ 393], 90.00th=[ 481], 95.00th=[ 414],
| 99.00th=[ 472], 99.50th=[ 506], 99.90th=[ 535], 99.95th=[ 542],
| 99.99th=[ 542]
bw ( KiB/s): min=381856, max=768888, per=99.52%, avg=397987.65, stdev=81583.56, samples=52
iops       : min= 234, max= 758, 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=0.88%, sys=4.21%, ctx=18395, 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=0.8%, 4=100.8%, 8=0.8%, 16=0.8%, 32=0.8%, 64=0.8%, >=64=0.8%
complete : 0=0.8%, 4=100.8%, 8=0.8%, 16=0.8%, 32=0.8%, 64=0.8%, >=64=0.1%
issued rws: total=18240,0,0 short=0,0,0 dropped=0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
READ: bw=391MiB/s (409MB/s), 391MiB/s-391MiB/s (409MB/s-409MB/s), io=18.0GiB (18.76B), run=26221-26221msec
```

Random read bandwidth

- fio command:

```
fio --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --
group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --
rw=randread --bs=1M --size=10G --iodepth=128 --runtime=120 --
numjobs=1
```

NOTE

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- fio result:

```
test: (g=0): rw=randread, bs=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 --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --
group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --
rw=write --bs=1M --size=10G --iodepth=128 --runtime=120 --numjobs=1
```

NOTE

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- fio result:

```
test: (groupid=0, jobs=1): err= 0: pid=21889: Mon Jun 8 14:53:44 2020
write: IOPS=243, BW=244MiB/s (255MB/s)(18.06GiB/42848msec)
slat (usec): min=183, max=584, avg=198.38, 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=[ 188], 10.00th=[ 167], 20.00th=[ 292],
| 30.00th=[ 422], 40.00th=[ 468], 50.00th=[ 586], 60.00th=[ 558],
| 70.00th=[ 625], 80.00th=[ 768], 90.00th=[ 982], 95.00th=[ 978],
| 99.00th=[ 1836], 99.50th=[ 1845], 99.90th=[ 1878], 99.95th=[ 1899],
| 99.99th=[ 1899]
bw ( KiB/s): min= 4896, max=468992, per=100.00%, avg=249588.99, stdev=147656.62, samples=83
iops      : min=    4, max= 458, avg=243.63, stdev=144.22, samples=83
lat (msec): 20=0.83%, 50=0.96%, 100=3.36%, 250=12.55%, 500=31.63%
lat (msec): 750=38.87%, 1000=18.96%
cpu      : usr=2.28%, sys=2.58%, 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=0.8%, 4=100.8%, 8=0.8%, 16=0.8%, 32=0.8%, 64=0.8%, >=64=0.8%
complete : 0=0.8%, 4=100.8%, 8=0.8%, 16=0.8%, 32=0.8%, 64=0.8%, >=64=0.1%
issued   : total=r=18240,0,0 short=r=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=244MiB/s (255MB/s), 244MiB/s-244MiB/s (255MB/s-255MB/s), io=18.06GiB (18.76GiB), run=42848-42848msec
```

Random write bandwidth

- fio command:

```
fio --ioengine=libaio --direct=1 --fallocate=none --time_based=1 --
group_reporting=1 --name=iops_fio --directory=/mnt/sfs-turbo/ --
```



```
rw=randwrite --bs=1M --size=10G --iodepth=128 --runtime=120 --  
numjobs=1
```

 NOTE

Variable `/mnt/sfs-turbo/` is the local path where the file to be tested is stored. Set it to the actual file name.

- 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=156001KB/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=[ 195]  
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 : 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, aggrb=156000KB/s, minb=156000KB/s, maxb=156000KB/s, mint=67216msec, maxt=67216msec
```

3 Testing SFS Turbo Latency

This section uses SFS Turbo Performance - Enhanced to test the file system latency. The specifications of the ECSs to be used are as follows:

Specifications: General computing-plus | c6.4xlarge.4 | 16 vCPUs | 64 GB

Image: EulerOS 2.5

Single-queue random read

- fio command:

```
fio -direct=1 -iodepth=1 -rw=randread -ioengine=libaio -bs=4k -size=10G -
numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/
fio_test_01 -name=randread_test
```

NOTE

Variable `/mnt/sfsturbo/fio_test_01 -name` indicates the location of the file to be tested. The location must be specific to the file name, which is the `fio_test_01 -name` file in the `/mnt/sfsturbo` directory in this example. Set the location based on site requirements.

- fio result:

```
[root@100 ~]# fio -direct=1 -iodepth=1 -rw=randread -ioengine=libaio -bs=4k -size=10G -numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/fio_test_01 -name=randread_test
randread_test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.5
Starting 1 process
randread_test: Laying out IO file (1 file / 10240MiB)
Jobs: 1 (f=1): [r(1)][100.0%][r=5606KiB/s,w=0KiB/s][r=1400,w=0 IOPS][eta 00m:00s]
randread_test: (groupid=0, jobs=1): err=0 pid=29815: Mon Nov 7 11:44:12 2022
read: IOPS=1401, BW=5606KiB/s (5740kB/s)(328MiB/60001msec)
slat (nsec): min=2370, max=51192, avg=4228.52, stdev=1319.26
clat (usec): min=559, max=8403, avg=707.69, stdev=197.66
lat (usec): min=562, max=8407, avg=712.00, stdev=197.71
clat percentiles (usec):
| 1.00th=[ 603], 5.00th=[ 627], 10.00th=[ 635], 20.00th=[ 652],
| 30.00th=[ 660], 40.00th=[ 668], 50.00th=[ 676], 60.00th=[ 693],
| 70.00th=[ 701], 80.00th=[ 725], 90.00th=[ 758], 95.00th=[ 807],
| 99.00th=[ 1369], 99.50th=[ 2040], 99.90th=[ 3523], 99.95th=[ 3982],
| 99.99th=[ 5735]
bw ( KIB/s): min= 5120, max= 5856, per=100.00%, avg=5605.12, stdev=134.52, samples=119
iops : min= 1280, max= 1464, avg=1401.28, stdev=33.63, samples=119
lat (usec) : 750=88.78%, 1000=9.31%
lat (msec) : 2=1.39%, 4=0.46%, 10=0.05%
cpu : usr=0.86%, sys=1.77%, ctx=84892, majf=0, minf=33
IO depths : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
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=0.0%
issued rwts: total=84887,0,0,0 short=0,0,0,0 dropped=0,0,0
latency : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
READ: bw=5606KiB/s (5740kB/s), 5606KiB/s-5606KiB/s (5740kB/s-5740kB/s), io=328MiB (344MB), run=60001-60001msec
[root@100 ~]#
```

Single-queue random write

- fio command:

```
fio -direct=1 -iodepth=1 -rw=randwrite -ioengine=libaio -bs=4k -size=10G -
numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/
fio_test_02 -name=randwrite_test
```

NOTE

Variable `/mnt/sfsturbo/fio_test_02 -name` indicates the location of the file to be tested. The location must be specific to the file name, which is the **fio_test_02 -name** file in the `/mnt/sfsturbo` directory in this example. Set the location based on site requirements.

- fio result:

```
[root@100 ~]# fio -direct=1 -iodepth=1 -rw=randwrite -ioengine=libaio -bs=4k -size=10G -numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/fio_test_02 -name=randwrite_test
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][r=0KIB/s,w=6708KIB/s][r=0,w=1677 IOPS][eta 00m:00s]
randwrite test: (group=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.5
write: IOPS=1661, BW=6645KIB/s (6884kB/s)(389MiB/6000msec)
slat (nsec): min=2566, max=34191, avg=4528.29, stdev=1661.31
clat (usec): min=472, max=8239, avg=595.67, stdev=160.48
lat (usec): min=477, max=8234, avg=680.38, stdev=160.55
clat percentiles (usec):
| 1.00th=[ 515], 5.00th=[ 529], 10.00th=[ 537], 20.00th=[ 553],
| 30.00th=[ 562], 40.00th=[ 570], 50.00th=[ 578], 60.00th=[ 586],
| 70.00th=[ 594], 80.00th=[ 611], 90.00th=[ 635], 95.00th=[ 660],
| 99.00th=[ 938], 99.50th=[ 1303], 99.90th=[ 3130], 99.95th=[ 3949],
| 99.99th=[ 5669]
bw ( KIB/s): min= 6208, max= 6968, per=100.00%, avg=6644.03, stdev=152.50, samples=119
iops : min= 1572, max= 1742, avg=1660.99, stdev=38.16, samples=119
lat (usec) : 500=0.12%, 750=97.53%, 1000=1.51%
lat (msec) : 2=0.59%, 4=0.29%, 10=0.05%
cpu : usr=1.13%, sys=2.03%, ctx=99679, majf=0, minf=34
IO depths : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
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=0.0%
issued rwts: total=0,99676,0,0 short=0,0,0 dropped=0,0,0
latency : target=0, window=0, percentile=100.00%, depth=1
Run status group 0 (all jobs):
WRITE: bw=6645KIB/s (6884kB/s), 6645KIB/s-6645KIB/s (6884kB/s-6884kB/s), io=389MiB (408MB), run=60001-60001msec
[root@100 ~]#
```

Single-queue sequential read

- fio command:

fio -direct=1 -iodepth=1 -rw=read -ioengine=libaio -bs=4k -size=10G -numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/fio_test_03 -name=read_test

NOTE

Variable `/mnt/sfsturbo/fio_test_03 -name` indicates the location of the file to be tested. The location must be specific to the file name, which is the **fio_test_03 -name** file in the `/mnt/sfsturbo` directory in this example. Set the location based on site requirements.

- fio result:

```
[root@100 ~]# fio -direct=1 -iodepth=1 -rw=read -ioengine=libaio -bs=4k -size=10G -numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/fio_test_03 -name=read_test
Starting 1 process
Jobs: 1 (f=1): [R(1)][100.0%][r=6412KIB/s,w=0KIB/s][r=1603,w=0 IOPS][eta 00m:00s]
read_test: (group=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.5
read: IOPS=1576, BW=6306KIB/s (6457kB/s)(370MiB/6000msec)
slat (nsec): min=2349, max=38953, avg=4152.29, stdev=1412.71
clat (usec): min=473, max=8642, avg=628.77, stdev=155.53
lat (usec): min=477, max=8646, avg=633.10, stdev=155.62
clat percentiles (usec):
| 1.00th=[ 545], 5.00th=[ 562], 10.00th=[ 570], 20.00th=[ 578],
| 30.00th=[ 594], 40.00th=[ 603], 50.00th=[ 611], 60.00th=[ 619],
| 70.00th=[ 635], 80.00th=[ 644], 90.00th=[ 676], 95.00th=[ 717],
| 99.00th=[ 1020], 99.50th=[ 1369], 99.90th=[ 2900], 99.95th=[ 3818],
| 99.99th=[ 5473]
bw ( KIB/s): min= 5080, max= 6552, per=99.96%, avg=6303.65, stdev=141.48, samples=119
iops : min= 1452, max= 1638, avg=1575.87, stdev=35.37, samples=119
lat (usec) : 500=0.01%, 750=96.40%, 1000=2.54%
lat (msec) : 2=0.81%, 4=0.20%, 10=0.04%
cpu : usr=0.99%, sys=1.04%, ctx=94596, majf=0, minf=34
IO depths : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
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=0.0%
issued rwts: total=94593,0,0 short=0,0,0 dropped=0,0,0
latency : target=0, window=0, percentile=100.00%, depth=1
Run status group 0 (all jobs):
READ: bw=6306KIB/s (6457kB/s), 6306KIB/s-6306KIB/s (6457kB/s-6457kB/s), io=370MiB (387MB), run=60001-60001msec
[root@100 ~]#
```

Single-queue sequential write

- fio command:

fio -direct=1 -iodepth=1 -rw=write -ioengine=libaio -bs=4k -size=10G -numjobs=1 -runtime=60 -group_reporting -filename=/mnt/sfsturbo/fio_test_04 -name=write_test

 NOTE

Variable `/mnt/sfsturbo/fio_test_04` -name indicates the location of the file to be tested. The location must be specific to the file name, which is the `fio_test_04` -name file in the `/mnt/sfsturbo` directory in this example. Set the location based on site requirements.

- fio result:

```
[root@l80 ~]# fio --direct=1 --ioengine=libaio --bs=4k --size=100 --numjobs=1 --runtime=60 --group_reporting --filename=/mnt/sfsturbo/fio_test_04 --name=write_test
write_test: (g=0): rw=write, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.5
Starting 1 process
Jobs: 1 (f=1): [W(1)][100.0%][r=6KiB/s,w=6886KiB/s][r=0,w=1721 IOPS][eta 00m:00s]
write_test: (groupid=0, jobs=1): err=0: pid=989: Tue Nov 8 21:40:23 2022
write: IOPS=1790, BW=7194KiB/s (7367kB/s)(422MiB/60001msec)
slat (msec): min=2266, max=32022, avg=4296.55, stdev=1452.74
clat (usec): min=443, max=10894, avg=559.41, stdev=168.67
lat (usec): min=447, max=10990, avg=554.82, stdev=168.73
clat percentiles (usec):
| 1.00th=[ 469], 5.00th=[ 486], 10.00th=[ 494], 20.00th=[ 502],
| 30.00th=[ 515], 40.00th=[ 523], 50.00th=[ 529], 60.00th=[ 537],
| 70.00th=[ 553], 80.00th=[ 562], 90.00th=[ 594], 95.00th=[ 627],
| 99.00th=[ 963], 99.50th=[ 1483], 99.90th=[ 3064], 99.95th=[ 3654],
| 99.99th=[ 5211]
bw ( KIB/s): min= 6520, max= 7680, pcr=99.99%, avg=7193.45, stdev=213.10, samples=120
iops      : min= 1632, max= 1922, avg=1798.35, stdev=53.28, samples=120
lat (msec): 500=16.99%, 750=80.94%, 1000=1.14%
lat (msec) : 2=0.63%, 4=0.26%, 10=0.03%, 20=0.01%
cpu       : usr=0.97%, sys=2.16%, ctx=107916, majf=0, minf=33
IO depths : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
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=0.0%
issued rwts: total=0,107912,0,0 short=0,0,0,0 dropped=0,0,0,0
latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
WRITE: bw=7194KiB/s (7367kB/s), 7194KiB/s-7194KiB/s (7367kB/s-7367kB/s), io=422MiB (442MB), run=60001-60001msec
[root@l80 ~]#
```

4 Creating a Readable and Writable Subdirectory on the File System for a Common User

4.1 Solution Overview

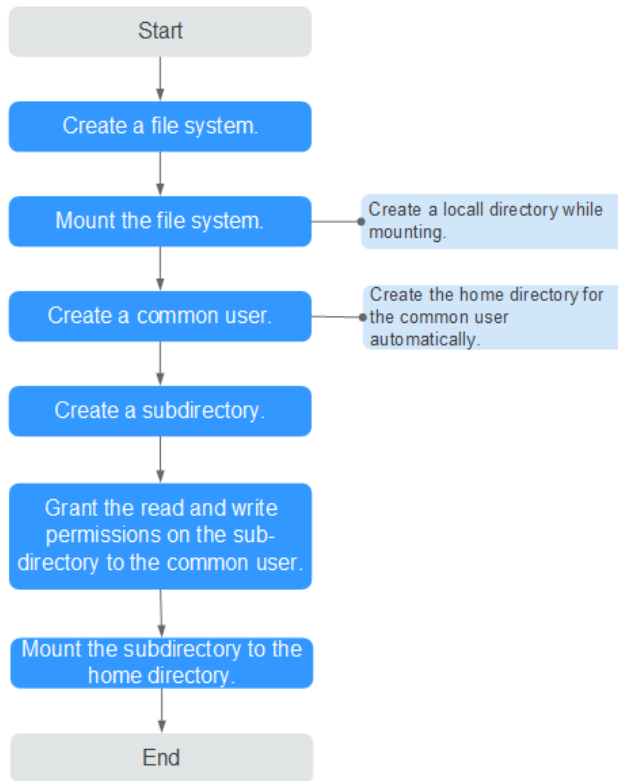
Scenarios

After a file system is created, only the **root** user has the read and write permissions on the file system by default. The **root** user can grant the access permission to multiple common users if needed. The **root** user can also create multiple subdirectories for each common user and mount them to the home directory of the file system, granting the read and write permissions to common users at the same time.

Process

Figure 4-1 illustrates the procedure of creating a readable and writable subdirectory on the file system for a common user.

Figure 4-1 Flowchart



4.2 Resource and Cost Planning

The following table describes the resource planning in this best practice.

Table 4-1 Resource and cost planning

Resource	Description
Elastic Cloud Server (ECS)	A file system and the ECSs must belong to the same project so that data can be shared between the ECSs through the file system.
Virtual Private Cloud (VPC)	VPC provisions an isolated virtual network environment defined and managed by yourself, improving the security of cloud resources and simplifying network deployment. A server cannot access file systems in a different VPC. Before using SFS, assign the file system and the servers to the same VPC.

Resource	Description
File system	A file system provides users with shared file storage through NFS and CIFS. It is used for accessing network files remotely. After a user creates a file system on the console, the file system can be mounted to multiple servers and is accessible through the standard POSIX.

4.3 Implementation Procedure

4.3.1 Creating a Local Directory for a File System

After creating a file system, you need to mount the file system to an ECS and create a local directory for the **root** user.

If the file system has been mounted, skip this section. Record the local directory in [Step 4](#) and perform steps in [Creating a Readable and Writable Subdirectory on the File System for Each User](#).

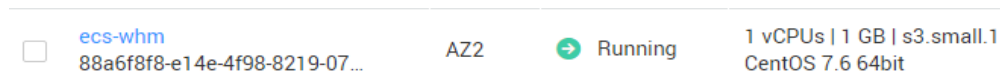
Prerequisites

- You have checked the type of the ECS operating system. Different operating systems require different commands for NFS client installation.
- You have created a file system and have obtained the mount point of the file system.
- The ECS to which a file system is mounted belongs to the same VPC as the file system.
- The IP addresses of the DNS server used to resolve the file system domain name have been configured on the ECS.

Procedure

Step 1 Create an ECS running CentOS in AZ2 of the CN North-Beijing1 region, for example, **ecs-whm**. See [Figure 4-2](#).

Figure 4-2 Creating an ECS



<input type="checkbox"/>	ecs-whm 88a6f8f8-e14e-4f98-8219-07...	AZ2	→ Running	1 vCPUs 1 GB s3.small.1 CentOS 7.6 64bit
--------------------------	--	-----	-----------	---

Step 2 Log in to the ECS as user **root**. Install the NFS client.

1. Run the following command to check whether the NFS software package is installed.
 - On CentOS, Red Hat, Oracle Enterprise Linux, SUSE, Euler OS, Fedora, or OpenSUSE:

rpm -qalgrep nfs

- On Debian or Ubuntu:

dpkg -l nfs-common

If a command output similar to the following is displayed, the NFS software package has been installed and you can go to [Step 3](#). If nothing is displayed, go to [Step 2.2](#).

- On CentOS, Red Hat, Euler OS, Fedora, or Oracle Enterprise Linux:

```
libnfsidmap  
nfs-utils
```

- On SUSE or OpenSUSE:

```
nfsidmap  
nfs-client
```

- On Debian or Ubuntu:

```
nfs-common
```

2. Run the following command to install the NFS software package.

NOTE

The following commands require that the ECS be connected to the Internet. Otherwise, the installation will fail.

- On CentOS, Red Hat, Euler OS, Fedora, or Oracle Enterprise Linux:

sudo yum -y install nfs-utils

- On Debian or Ubuntu:

sudo apt-get install nfs-common

- On SUSE or OpenSUSE:

zypper install nfs-client

- Step 3** Run the following command to check whether the domain name in the file system mount point can be resolved. See [Figure 4-3](#).

nslookup *File system domain name*

```
nslookup sfs-nas1.xx-xxxx-xx.xxxxxxxxxx.com
```

NOTE

- A file system domain name is just a part of the mount point, for example, **sfs-nas1.xx.com**. You can obtain a file system domain name from the mount point of a file system. In this step, you are not supposed to enter the entire mount point but only the domain name.
- If the **nslookup** command cannot be used, install the **bind-utils** software package by running the **yum install bind-utils** command.
- If the resolution succeeds, go to [Step 4](#).
- If the domain name cannot be resolved, configure the DNS server IP address and then mount the file system. For details, see [Configuring DNS](#).

Figure 4-3 Domain name resolution

```
[root@ecs-whm ~]# rpm -qalgrep nfs  
libnfsidmap-0.25-19.el7.x86_64  
nfs-utils-1.3.0-0.61.el7.x86_64  
[root@ecs-whm ~]# nslookup sfs-nas1.  
COM  
-bash: nslookup: command not found  
[root@ecs-whm ~]# yum install bind-utils  
Loaded plugins: fastestmirror  
Loading mirror speeds from cached hostfile
```


Step 4 Run the following command to create a local directory for mounting the file system. Record the local directory name, for example, **root001**.

mkdir *Local directory*

```
mkdir root001
```

Step 5 Run the following command to mount the file system to the ECS. SFS supports mounting only file systems complying with NFSv3 to ECSs running Linux. [Table 4-2](#) describes the variables.

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

NOTICE

After an ECS that has mounted file systems restarts, it loses the file system mount information. You can configure automatic mount in the **fstab** file to ensure that an ECS automatically mounts file systems when it restarts. For details, see [Mounting a File System Automatically](#).

Table 4-2 Parameters

Parameter	Description
vers	File system version. Currently, only NFSv3 is supported, so the value is fixed to 3 .
timeo	Waiting time before the NFS client retransmits a request. The unit is 0.1 second. Recommended value: 600
lock/nolock	Whether to lock files on the server using the NLM protocol. If nolock is selected, the lock is valid for applications on one host. For applications on another host, the lock is invalid. Recommended value: nolock
Mount point	The format for an SFS file system is <i>File system domain name:/Path</i> , for example, example.com:/share-xxx . The format for an SFS Turbo file system is <i>File system IP address./</i> , for example, 192.168.0.0/ . NOTE <i>x</i> can be a digit or letter. If the mount point is too long to display completely, expand the column to view the full mount point.
Local directory	Local directory on the ECS, used to mount the file system, for example, /local_path .

Figure 4-4 Mount point



Step 6 Run the following command to view the mounted file system:

mount -l

If the command output contains the following information, the file system is mounted successfully.

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

Step 7 After the mounting is successful, create a common user and subdirectory by referring to the next section.

If the mounting fails or times out, rectify the fault. For details, see [Troubleshooting](#).

 **NOTE**

The supported maximum size of a file to be written is 240 TB.

----End

4.3.2 Creating a Readable and Writable Subdirectory on the File System for Each User

Prerequisites

- A file system has been created and can be mounted to the ECS by the **root** user. For details, see [Creating a Local Directory for a File System](#).
- You have obtained the mount point of the file system.

Procedure

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

Step 2 Run the following commands to add a common user under the **root** account. User **Tom** is added as an example.

```
adduser Tom  
passwd Tom
```

Change the password of common user **Tom** as prompted. After the user is created, the home directory **/home/Tom** is automatically created.

Step 3 Run the following command in the local directory of the **root** user to create a subdirectory.

According to [Step 4](#) in [Creating a Local Directory for a File System](#), the local directory of the **root** user is **root001**. Run the following command to create subdirectory **Tom** for common user **Tom**. Replace **root001** with the actual local directory.

```
mkdir /root/root001/Tom
```

Step 4 Run the following command to assign the read and write permissions of subdirectory **Tom** to common user **Tom**. Replace **root001** with the actual local directory.

```
chown Tom:Tom /root/root001/Tom
```

After the subdirectory is created, run the following commands to check whether common user **Tom** has the read and write permissions on subdirectory **Tom**. See [Figure 4-5](#).

```
cd /home  
cd /root/root001  
ll
```

Figure 4-5 Checking the permissions

```
[root@ecs-whm home]# cd /root/root001
[root@ecs-whm root001]# ll
total 4
drwxr-xr-x 2 Tom Tom 4096 May 13 14:31 Tom
```

The preceding command output indicates that user **Tom** has obtained the read and write permissions on subdirectory **Tom**.

- Step 5** Run the **mount** command to mount subdirectory **root001/Tom** to the home directory of Tom, **/home/Tom**. In the following command, **xx-xxxx-xx** needs to be replaced with the region where the file system is located, and **share-xxxx** needs to be replaced with the actual file system. The first **Tom** needs to be replaced with the actual subdirectory name. See [Figure 4-6](#).

```
mount -t nfs sfs-nas1.xx-xxxx-xx.xxxxxxxxxx.com:/share-xxxx/Tom /home/Tom
```

Figure 4-6 Mounting the subdirectory

```
[root@ecs-whm root001]# mount -t nfs sfs-nas1.xxxxxxxxxx.com:/share-xxxx/Tom /home/Tom
[1282638.340788] Key type dns_resolver registered
[1282638.381991] NFS: Registering the id_resolver key type
[1282638.382851] Key type id_resolver registered
[1282638.383689] Key type id_legacy registered
```

Run the following command to check whether the subdirectory has been mounted to the home directory of **Tom**. See [Figure 4-7](#).

```
df -h
```

Figure 4-7 Checking the mounting result

```
[root@ecs-whm root001]# df -h
Filesystem                                Size  Used Avail Use% Mounted on
/dev/vda1                                  48G   1.9G   36G   5% /
devtmpfs                                   486M    0   486M   0% /dev
tmpfs                                       496M    0   496M   0% /dev/shm
tmpfs                                       496M   26M   471M   6% /run
tmpfs                                       496M    0   496M   0% /sys/fs/cgroup
tmpfs                                       188M    0   188M   0% /run/user/0
sfs-nas1.xxxxxxxxxx.com:/share-xxxx/Tom  188G    0   188G   0% /root/root001
sfs-nas1.xxxxxxxxxx.com:/share-xxxx/Tom  188G    0   188G   0% /home/Tom
```

----End