Object Storage Service

Parallel File System Feature Guide

Issue 03

Date 2024-03-19

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Contents

1 Introduction.. 1
1.1 About PFS..1
1.2 Application Scenarios..1
1.3 Constraints..1
1.4 Using PFS...2
1.5 Billing...2
1.6 Features..2
1.6.1 Lifecycle Management...3
1.6.2 Permissions Configuration..4
2 Using PFS on a Console...6
2.1 Creating a Parallel File System...6
3 Using PFS with OBS APIs.. 8
3.1 Compatibility Between OBS APIs and PFS..8
A Change History.. 15

1 Introduction

1.1 About PFS

Parallel File System (PFS) is a high-performance semantic file system provided by OBS. It features access latency in milliseconds, TB/s-level bandwidth, and millions of IOPS.

1.2 Application Scenarios

PFS is highly compatible, scalable, and reliable, and delivers amazing performance.

It is mainly used in the following scenarios:

Big data: log analysis, content recommendation, operation reports, user profiling, and interactive analysis

1.3 Constraints

Operations

- An existing OBS bucket cannot be changed to a parallel file system. For details about how to create a parallel file system, see **Creating a Parallel File System**.
- Custom domain names of parallel file systems cannot be configured on OBS Console.

Functions

- Image processing currently cannot be used to process (such as downsize, resize, or watermark) images stored in parallel file systems.
- Server-side encryption is not supported.
- Versioning is not supported.
- Bucket inventory is not supported.
- Static website hosting is not supported.

- Configuration of default storage class for a parallel file system is not supported.
- A parallel file system can be mounted to multiple Linux servers for concurrent reads, but this is not recommended for concurrent writes.

Performance

- A parallel file system provides a maximum bandwidth of 10 MB/s per TB by default.

Naming

- In a parallel file system, a file name cannot contain two consecutive slashes (//). For example, if you name a file as **test//123.txt**, an error will be reported.

1.4 Using PFS

PFS provides both a console and REST APIs for managing and accessing data, so that your applications can seamlessly interconnect with PFS without requiring any modifications. This allows you to process files stored in PFS anytime, anywhere and retrieve the processed files quickly. PFS supports both Portable Operating System Interface (POSIX) and OBS APIs, so you can process files the same way you process objects. This achieves interoperability between objects and files.

The table below describes the ways to use PFS in detail.

NOTE

Access permissions for OBS also apply to PFS. Before using PFS, make sure that you have the permissions required to access OBS resources.

Table 1-1 Ways to use PFS

Way	Function	Reference
PFS Console	On the console, you can create parallel file systems and manage them.	**Creating a Parallel File System**
OBS API	You can make API calls to use parallel file systems.	**Compatibility Between OBS APIs and PFS**

1.5 Billing

Parallel file systems support both pay-per-use and yearly/monthly (resource packages) billing modes. For details about resource packages, see **Resource Package Overview**.

1.6 Features

1.6.1 Lifecycle Management

The use cases and main functions of object lifecycle management also work on files in parallel file systems. For more information, see **Lifecycle Management**.

Differences Between File and Object Lifecycle Management

- A lifecycle rule can be used to manage files, but it cannot transition a folder to the Archive storage class. However, a lifecycle rule can delete an empty folder upon expiration.
- File deletion upon expiration and transition to the Archive storage class can be configured using either the API or console, while transition to Infrequent Access can only be configured using the API. PFS does not support versioning, so lifecycle actions (including deletion upon expiration and transition to the Archive or Infrequent Access storage class) that are related to versioning cannot be applied.
- If direct reading is enabled for a parallel file system, you can read files stored in the Archive storage class without restoring them first.
- A maximum of 20 lifecycle rules can be configured for a parallel file system.
- The time applied for a lifecycle rule to work on a file is when the data of the file was last updated.
- After a lifecycle rule is configured for a parallel file system, there are limits on how many directories that the rule can be applied to. If the configuration exceeds the limit, the lifecycle rule execution will be prolonged.
  a. There can be no more than 100,000 level-1 subdirectories in each directory.
  b. There can be no more than 10 million subdirectories (folders) matching the prefix defined in the rule in total.
  c. There can be no more than 30 million files matching the prefix defined in the rule in total.

Notes

- If renamed files or files in a renamed folder meet the conditions specified in a lifecycle rule, the time when the file data was most recently updated, not when the files were renamed, will be applied for the lifecycle rule to take effect. In addition, the effective time of the lifecycle rule may be delayed for up to seven days.
- For a file copy on a client, the lifecycle rule determines when to expire the file copy or transition it to the Archive storage class based on the file copy creation time.
  – For example, if a file, **src.txt**, was created on January 1, 2019, and was then copied to the **des.txt** file by running the **cp -a src.txt des.txt** command on September 1, 2019. Then the lifecycle rule calculated when to perform the specified actions on the file copy based on September 1, 2019.
- In the lifecycle rule of a parallel file system, directories in the file system are periodically scanned, and then deleted if they meet the expiration conditions. Scan intervals (usually seven days) vary depending on cluster configurations. A scan starts from the deepest directory, and the scanned empty directories

that meet the expiration conditions will be deleted, but those non-empty directories will not be processed. For this mechanism, a single-level directory that meets the expiration conditions will be deleted 0 to 7 days after it has been emptied. Accordingly, a two-level directory will be deleted 0 to 14 days after it has been emptied. Each time a directory level is added, the waiting time for deletion increases by seven days.

1.6.2 Permissions Configuration

The use cases and main functions of object access control also work on files in parallel file systems. For more information, see **Permissions Configuration Guide**.

Differences Between File and Object Permission Configurations

To exactly match a specific directory, the resource path in the policy must end with a slash (/). When checking permissions, parallel file systems consider objects as directories. If the object identifier does not end with a slash (/), the system will add a slash (/) to the end of the object identifier and then performs a policy matching.

IAM Permission Configuration Examples

Example 1: Grant a user the permissions required to download **dir_1**, excluding its subdirectories.

In the following configuration, the resource path ends with a slash (/). In such case, a success response can be returned when **dir_1** or **dir_1/** is contained in the URL of a head request.

Note that this configuration is not applied to subdirectories or files in **dir_1**. Therefore, a failure response will be returned if a head request is sent to **dir_1/file1**.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "obs:object:GetObject",
            ],
            "Resource": [
                "obs:*:*:object:examplebucket/dir_1/",
            ]
        }
    ]
}
```

Example 2: Grant a user the permissions required to download **dir_1** and its subdirectories.

In the following configuration, the resource path uses prefix-based matching and ends with a wildcard (*). In such case, a success response can be returned when a head request is sent to **dir_1/file1**.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "obs:object:GetObject",
            ],
            "Resource": [
                "obs:*:*:object:examplebucket/dir_1/*",
            ]
        }
    ]
}
```

Bucket Policy Configuration Examples

Example: Grant a user the permissions required to download **dir_1**, excluding its subdirectories.

In the following configuration, the resource path ends with a slash (/). In such case, a success response can be returned when **dir_1** or **dir_1/** is contained in the URL of a head request.

```
{
    "Statement":[
    {
      "Sid":"test",
      "Effect":"Allow",
      "Principal": {"ID": ["domain/b4bf1b36d9ca43d984fbcb9491b6fce9:user/
71f3901173514e6988115ea2c26d1999"]},
      "Action":["*"],
      "Resource":[
        "examplebucket/dir_1/",
      ]
    }
  ]
}
```

2 Using PFS on a Console

2.1 Creating a Parallel File System

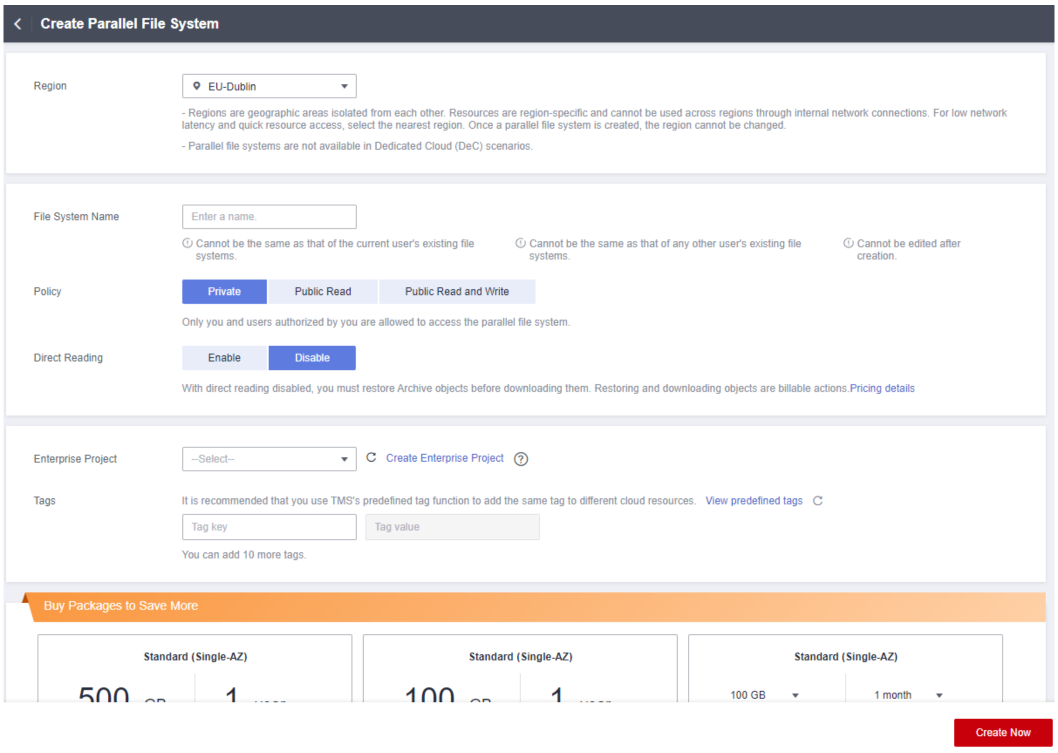
You can create a parallel file system on OBS Console.

Procedure

Step 1 In the navigation pane, choose **Parallel File Systems**.

Step 2 In the upper right corner of the page, click **Create Parallel File System**.

Figure 2-1 Creating a parallel file system



Create Parallel File System

Region: EU-Dublin

- Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region. Once a parallel file system is created, the region cannot be changed.

- Parallel file systems are not available in Dedicated Cloud (DeC) scenarios.

File System Name: Enter a name.

Cannot be the same as that of the current user's existing file systems.
Cannot be the same as that of any other user's existing file systems.
Cannot be edited after creation.

Policy: [✓ Private] Public Read | Public Read and Write

Only you and users authorized by you are allowed to access the parallel file system.

Direct Reading: Enable [✓ Disable]

With direct reading disabled, you must restore Archive objects before downloading them. Restoring and downloading objects are billable actions.Pricing details

Enterprise Project: --Select-- Create Enterprise Project

Tags: It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags

Tag key | Tag value

You can add 10 more tags.

Step 3 Select a region and enter a name for the parallel file system.

NOTE

- Once a parallel file system is created, its name cannot be changed.
- URLs do not support uppercase letters and cannot distinguish between names containing uppercase or lowercase letters. For example, if you attempt to access the parallel file system **MyFileSystem** using a URL, the file system name will be resolved to **myfilesystem**, causing an access error. For this reason, a parallel file system name can contain only lowercase letters, digits, periods (.), and hyphens (-).

Step 4 Configure a policy. You can select **Private**, **Public Read**, or **Public Read/Write** for the parallel file system.

Step 5 Configure direct reading. With direct reading enabled, you can directly download objects in the Archive storage class without restoring them first.

Step 6 Configure **Enterprise Project**. Add the parallel file system to an enterprise project for unified management.

Create an enterprise project by referring to **Creating an Enterprise Project**. The default enterprise project is named **default**.

On the **Enterprise Project Management** page, **create an enterprise project**, and **add a user group to the enterprise project**. By doing so, users in this user group obtain the operation permissions for the parallel file systems in the enterprise project.

NOTE

- Only an enterprise account can configure enterprise projects.
- OBS ReadOnlyAccess and OBS OperateAccess are the fine-grained authorizations of the enterprise project user group in OBS.

Step 7 (Optional) Add tags. Tags are used to identify parallel file systems in OBS, for the purpose of classification. Each tag is represented by one key-value pair. For details about how to add a tag, see **Tags**.

Step 8 (Optional) Buy storage packages. By default, you are billed on a pay-per-use basis for using parallel file systems. You can also purchase **storage packages** to save more. After selecting a required package, go to the **Confirm** page to complete the purchase.

Storage packages can also be purchased after the parallel file system is created.

Step 9 Confirm the settings at the bottom of the page and click **Create Now**.

Step 10 View the file system you created just now in the parallel file system list.

Then, you can use the parallel file system the same way you use a bucket. For details, see **Using PFS**.

----End

3 Using PFS with OBS APIs

3.1 Compatibility Between OBS APIs and PFS

You can call some OBS APIs to use PFS. There may be additional requirements when you call these APIs.

For details about the OBS APIs, see **Object Storage Service API Reference**.

APIs for Basic Bucket Operations

Table 3-1 APIs for basic bucket operations

API	PFS Compatible	Differences
Listing buckets	Yes	The **x-obs-bucket-type:POSIX** header is required for obtaining the list of parallel file systems.
Creating a bucket	Yes	The **x-obs-fs-file-interface:Enabled** header is required for creating a parallel file system.
Listing objects in a bucket	Yes	-
Obtaining bucket metadata	Yes	-
Obtaining bucket region locations	Yes	-
Deleting a bucket	Yes	-

APIs for Advanced Bucket Settings

Table 3-2 APIs for advanced bucket settings

API	PFS Compatible	Differences
Configuring a bucket policy	Yes	-
Obtaining bucket policy information	Yes	-
Deleting a bucket policy	Yes	-
Configuring a bucket ACL	Yes	-
Obtaining bucket ACL information	Yes	-
Configuring logging for a bucket	Yes	-
Obtaining a bucket logging configuration	Yes	-
Configuring bucket lifecycle rules	Yes	-
Obtaining bucket lifecycle configuration	Yes	-
Deleting bucket lifecycle rules	Yes	-
Configuring versioning for a bucket	No	-
Obtaining bucket versioning status	No	-
Configuring event notification for a bucket	Yes	-

API	PFS Compatible	Differences
Obtaining the event notification configuration of a bucket	Yes	-
Configuring storage class for a bucket	No	-
Obtaining bucket storage class information	No	-
Configuring tags for a bucket	Yes	-
Obtaining bucket tags	Yes	-
Deleting bucket tags	Yes	-
Configuring bucket storage quota	Yes	-
Querying bucket storage quota	Yes	-
Querying information about used space in a bucket	Yes	-
Configuring bucket inventories	No	-
Obtaining bucket inventories	No	-
Listing bucket inventories	No	-
Deleting bucket inventories	No	-
Configuring bucket encryption	No	-

API	PFS Compatible	Differences
Obtaining bucket encryption configuration	No	-
Deleting the encryption configuration of a bucket	No	-
Configuring the direct reading policy for Archive objects in a bucket	Yes	-
Obtaining the direct reading policy for Archive objects in a bucket	Yes	-
Deleting the direct reading policy for Archive objects in a bucket	Yes	-

APIs for Static Website Hosting

Table 3-3 APIs for static website hosting

API	PFS Compatible	Differences
Configuring static website hosting for a bucket	No	-
Obtaining the static website hosting configuration of a bucket	No	-
Deleting the static website hosting configuration of a bucket	No	-

API	PFS Compatible	Differences
Configuring bucket CORS	No	-
Obtaining the CORS configuration of a bucket	No	-
Deleting the CORS configuration of a bucket	No	-
OPTIONS buckets	No	-
OPTIONS objects	No	-

APIs for Object Operations

Table 3-4 APIs for object operations

API	PFS Compatible	Differences
PUT objects	Yes	● Headers not supported: **x-obs-storage-class**, **x-obs-website-redirect-location**, **success-action-redirect**, and **x-obs-expires** ● Objects uploaded using this API cannot be directly stored in the Infrequent Access or Archive storage class and are stored in the Standard storage class by default. You can later change the storage class by using a lifecycle rule or modifying the metadata.
POST objects	Yes	Headers not supported: **x-obs-storage-class**, **x-obs-website-redirect-location**, **success-action-redirect**, and **x-obs-expires**
Copying objects	Yes	Data can be replicated only between parallel file systems or buckets that are in the same cluster.
Obtaining object content	Yes	-
Obtaining object metadata	Yes	-

API	PFS Compatible	Differences
Deleting an object	Yes	-
Batch deleting objects	Yes	-
Restoring Archive objects	Yes	-
Appending objects	No	-
Configuring object ACL	Yes	-
Obtaining object ACL information	Yes	-
Modifying object metadata	Yes	In a parallel file system, the storage class of a directory cannot be changed. To change the storage class of a file in the directory, modify the metadata of the file or use a lifecycle rule to change the storage class of files in batches.
Modifying an object	Yes	This is a PFS only API and is not supported for OBS buckets.
Truncating an object	Yes	This is a PFS only API and is not supported for OBS buckets.
Renaming an object	Yes	This is a PFS only API and is not supported for OBS buckets.

APIs for Multipart Uploads

Table 3-5 APIs for multipart uploads

API	PFS Compatible	Differences
Listing initialized multipart tasks in a bucket	Yes	-
Initiating multipart upload tasks	Yes	-
Uploading parts	Yes	-

API	PFS Compatible	Differences
Copying parts	Yes	Copying parts is not supported for an appended file.
Listing uploaded parts	Yes	-
Merging parts	Yes	-
Canceling multipart tasks	Yes	-

A Change History

Released On	Description
2024-03-19	This is the third official release. This issue incorporates the following change: ● Added naming restrictions in **Constraints**.
2023-07-10	This is the second official release. This issue incorporates the following change: ● Updated the content under **Notes** in section "Lifecycle Management."
2023-03-16	This is the first official release.