**OSS与文件系统的对比**

| **对比项** | **OSS** | **文件系统** |
| --- | --- | --- |
| 数据模型 | OSS是一个分布式的对象存储服务，提供的是一个Key-Value对形式的对象存储服务。 | 文件系统是一种典型的树状索引结构。 |
| 数据获取 | 根据Object的名称（Key）唯一的获取该Object的内容。  虽然用户可以使用类似test1/test.jpg的名字，但是这并不表示用户的Object是保存在test1目录下面的。对于OSS来说，test1/test.jpg仅仅只是一个字符串，和a.jpg这种并没有本质的区别。因此不同名称的Object之间的访问消耗的资源是类似的。 | 一个名为test1/test.jpg的文件，访问过程需要先访问到test1这个目录，然后再在该目录下查找名为test.jpg的文件。 |
| 优势 | 支持海量的用户并发访问。 | 支持文件的修改，比如修改指定偏移位置的内容、截断文件尾部等。也支持文件夹的操作，比如重命名目录、删除目录、移动目录等非常容易。 |
| 劣势 | OSS保存的Object不支持修改（追加写Object需要调用特定的接口，生成的Object也和正常上传的Object类型上有差别）。用户哪怕是仅仅需要修改一个字节也需要重新上传整个Object。  OSS可以通过一些操作来模拟类似文件夹的功能，但是代价非常昂贵。比如重命名目录，希望将test1目录重命名成test2，那么OSS的实际操作是将所有以test1/开头的Object都重新复制成以test2/开头的Object，这是一个非常消耗资源的操作。因此在使用OSS的时候要尽量避免类似的操作。 | 受限于单个设备的性能。访问越深的目录消耗的资源也越大，操作拥有很多文件的目录也会非常慢。 |

因此，将OSS映射为文件系统是非常低效的，也是不建议的做法。如果一定要挂载成文件系统的话，建议尽量只做写新文件、删除文件、读取文件这几种操作。使用OSS应该充分发挥其优点，即海量数据处理能力，优先用来存储海量的非结构化数据，比如图片、视频、文档等。

以下是OSS与文件系统的概念对比：

| **对象存储 OSS** | **文件系统** |
| --- | --- |
| Object | 文件 |
| Bucket | 主目录 |
| Region | 无 |
| Endpoint | 无 |
| AccessKey | 无 |
| 无 | 多级目录 |
| GetService | 获取主目录列表 |
| GetBucket | 获取文件列表 |
| PutObject | 写文件 |
| AppendObject | 追加写文件 |
| GetObject | 读文件 |
| DeleteObject | 删除文件 |
| 无 | 修改文件内容 |
| CopyObject （目的和源相同） | 修改文件属性 |
| CopyObject | 复制文件 |
| 无 | 重命名文件 |

## OSS术语表

| **英文** | **中文** |
| --- | --- |
| Bucket | 存储空间 |
| Object | 对象或者文件 |
| Endpoint | OSS 访问域名 |
| Region | 地域或者数据中心 |
| AccessKey | AccessKeyId和AccessKeySecret的统称，访问密钥 |
| Put Object | 简单上传 |
| Post Object | 表单上传 |
| Multipart Upload | 分片上传 |
| Append Object | 追加上传 |
| Get Object | 简单下载 |
| Callback | 回调 |
| Object Meta | 文件元信息。用来描述文件信息，例如长度，类型等 |
| Data | 文件数据 |
| Key | 文件名 |
| ACL (Access Control List) | 存储空间或者文件的权限 |