深入浅出Lustre Distributed Namespace特性

作者	时间	QQ技术交流群
perrynzhou@gmail.com	2022/09/17	672152841

存储内核技术交流

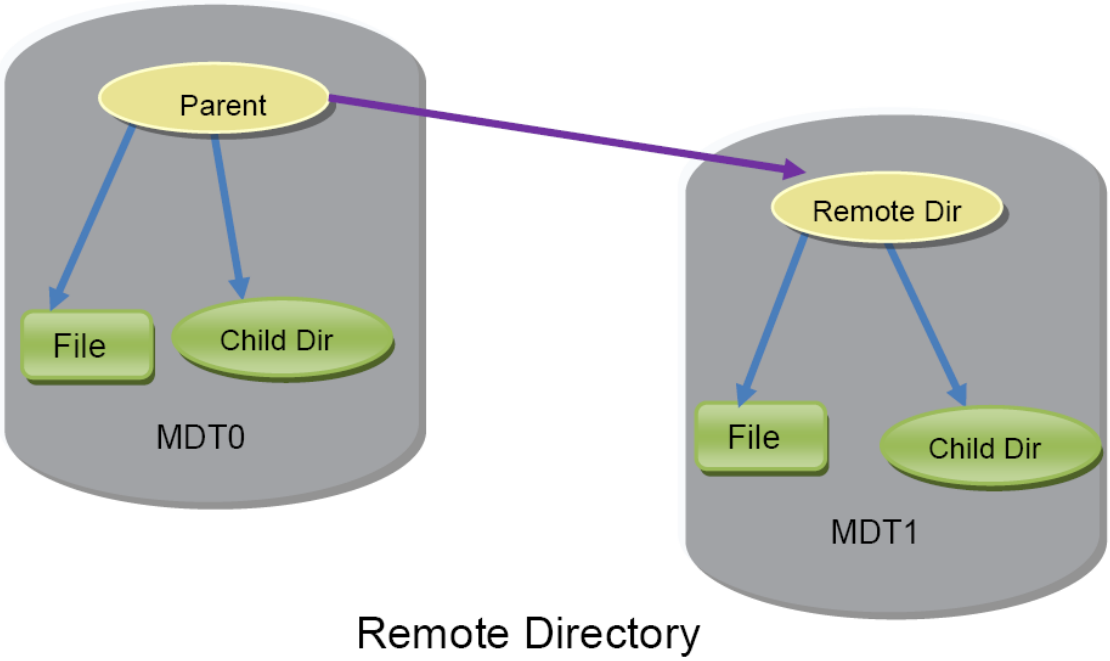
微信扫描二维码，关注我的公众号

`Distributed Namespace(DNE)`设计意图

- 在没有`DNE`功能之前`Lustre`文件系统元数据查询和操作仅支持在一个`MDT`上，如果应用的`Workload`对于元数据操作要求高，单个`MDT`不管从`容量`或者`性能`上考虑，这些都将会是一个瓶颈点。基于这些因素考虑，`Lustre`设计`DNE`来解决`MDT`的扩展性和性能问题。`DNE`核心解决了`文件`或者`目录`的元数据可以分散存储在多个`MDT`上，应用请求到不同的文件或者目录元数据请求会分摊到不同的`MDT`上，`MDT0`存储了`Lustre`文件系统的根。

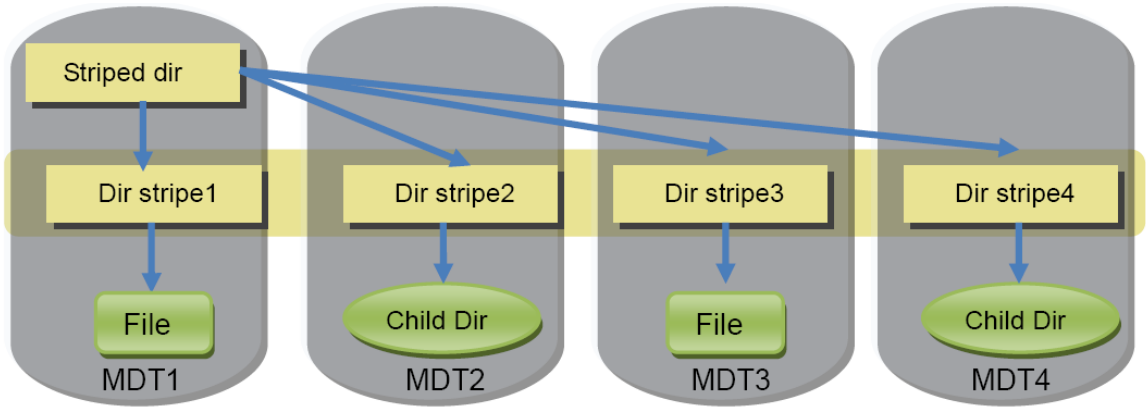
- `DNE`设计分为2个阶段.`阶段一`解决`根目录`下的子目录可以分布到多个`MDT`，这个就涉及到`Remote Directory`.`阶段二`解决了给定的目录，把目录的实际存储数据分布到多个`MDT`上，这个就涉及到了`Striped Directories`。

    - `Remote Directory`



Remote Directory

- `Striped Directories`



Striped Directory

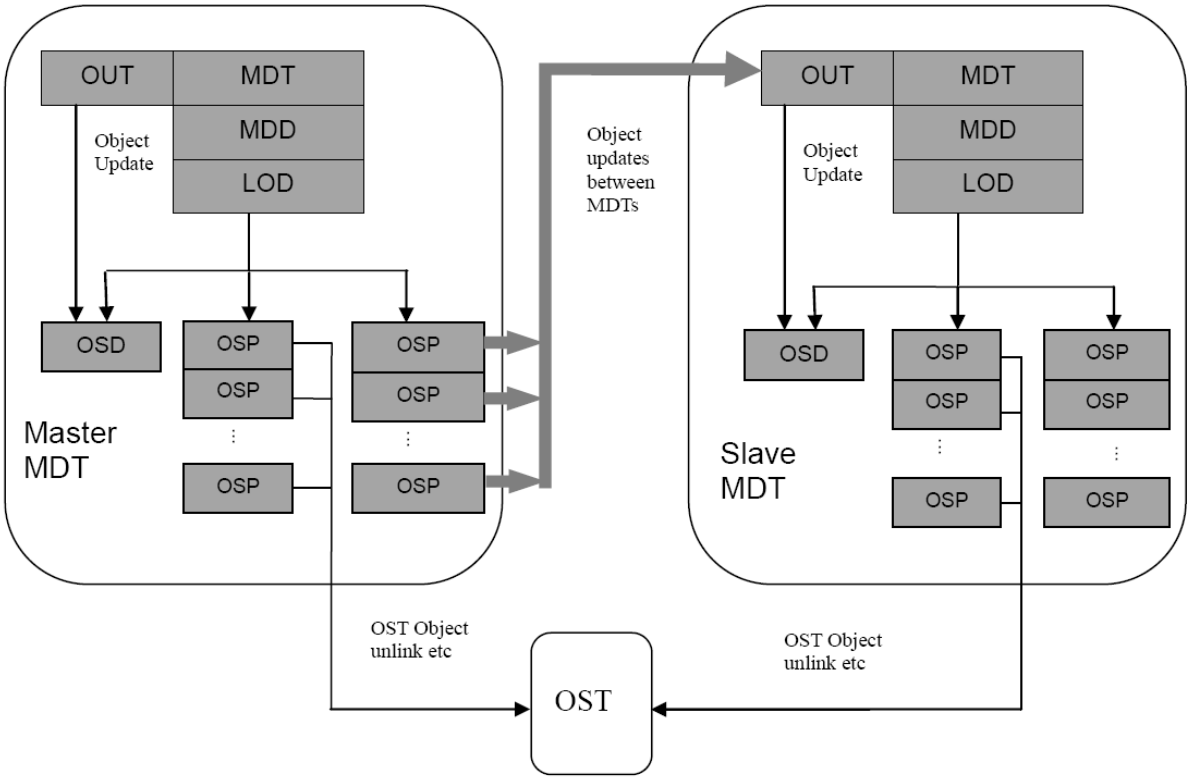
DNE使用命令

- 管理员可以通过`lfs mkdir -i {mdt_index} {remote_dir}`命令在第`{mdt_index}`mdt上创建`Remote Directory`;通过`rmdir {remote_dir}`删除远程目录
- 设置普通用户在其他的MDT(非MDT0)上创建远程目录需要调整如下参数

```
$ lctl set_param mdt.fsname-MDT0000.enable_remote_dir=1
$ lctl set_param.mdt.fsname-MDT0000.enable_remote_dir_gid=allowed_gid
```

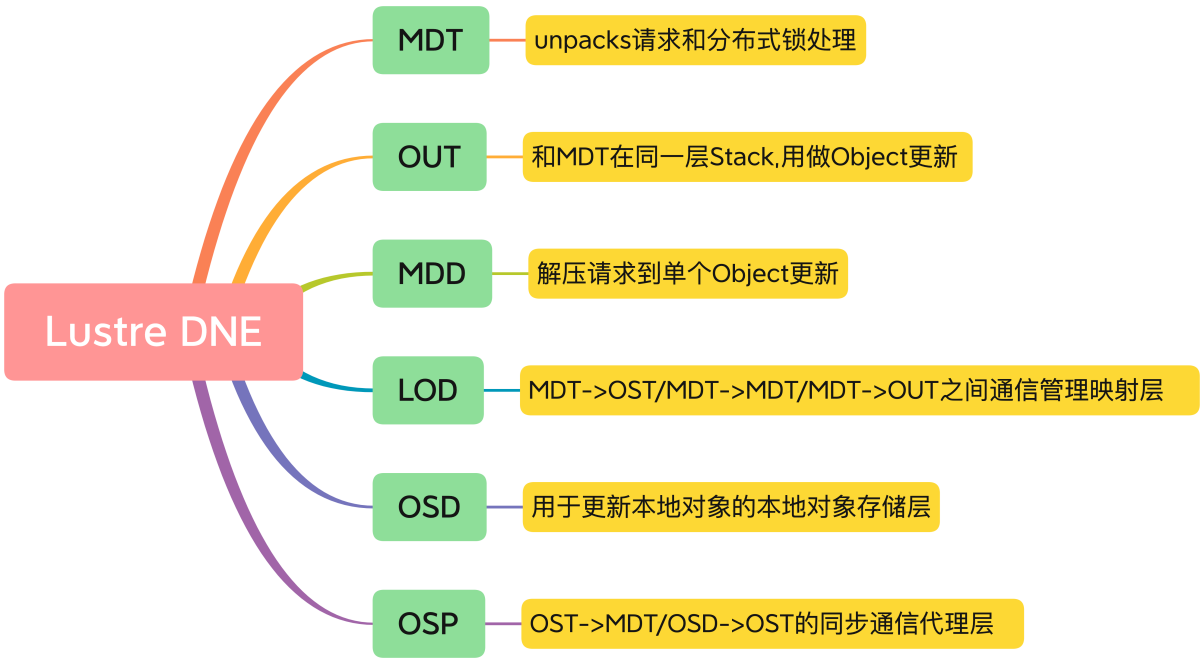
DNE架构

- `DNE`架构是基于原来`MDS Stack`基础设计，其架构中`OSD`是本地`MDS`的后端存储，一部分的`OSP`直接连接到OST；一部分`OSP`直接连接到远端的MDT上。



New MDS layer

- DNE架构Stack介绍

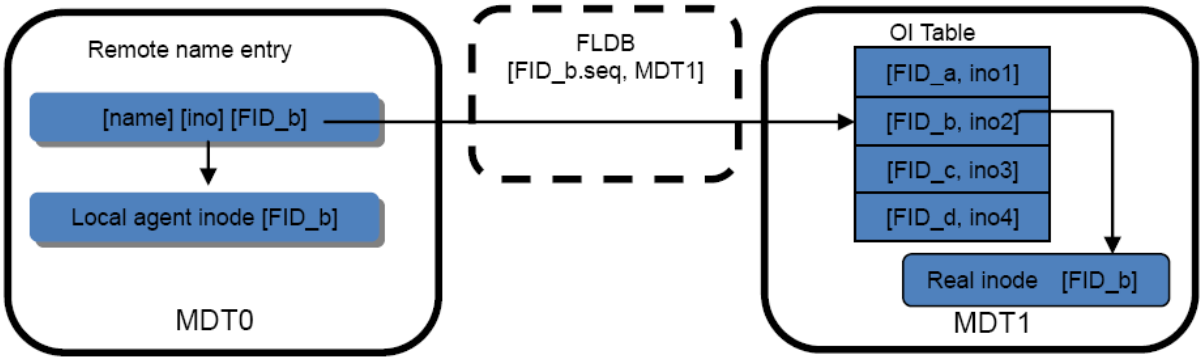


DNE处理流程

- MDT unpacks请求获取分布式锁
- MDD分割操作为多个对象的更新.MDD和LOD并不知道这些对象是本地还是远程更新,这些更新操作都会在一个事务中记性，要么全部成功，要么全部失败
- LOD检查此次更新时本地还是远程的更新操作，如果查询FLD发现是远程的更新就发送给本地的OSD进行更新；如果发现是远程更新，则通过OSP发送给其他的远程MDT进行

更新。

- 这里检查更新的对象是本地FID还是远程FID,还是需要查询`FLDB`。`FID`一般会在`Directory Entry`和`Inode Extended Attribute`中存储。`FID`在`FLDB`中查询到后能够识别到这个FID是在哪个`MDT`上(`MDT INDEX`).下面是MDT0上的一个`Remote Name Entry`和远程MDT1之间的映射关系。



Remote Inode Entry