**Glusterfs 客户端增加修复概要设计**

|  |  |  |  |
| --- | --- | --- | --- |
| 文 档 编 号 | NxSDS | 保 密 等 级 | 内部 |
| 作 者 | 林世跃 | 最后修改日期 | 2020-08-07 |

[文档修订记录 3](#_Toc17343)

[1. 简介 3](#_Toc29949)

[1.1. 功能背景概述 3](#_Toc14982)

[2. 总体设计 3](#_Toc23504)

[2.1. 需求规定 3](#_Toc26930)

[2.2. 运行环境 3](#_Toc22649)

[2.3. 设计思路 3](#_Toc17101)

[2.3.1配置方式 4](#_Toc8198)

[2.3.2 使用方式 7](#_Toc9648)

# 文档修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 修订内容简述 | 修订日期 | 作者 |
| V1.0 | 初稿 | 2020-08-7 | 林世跃 |

# 简介

## 功能背景概述

纠删码需要cpu计算能力，纠删码消耗的是计算能力更多，在一些纠删码引擎是需要一些硬件指令集加速的，如glusterfs使用到的纠删码引擎需要avx，sse等指令集支持。在ram上，cpu很弱，纠删码的编码解码速率极低，如385和A72上编码解码的速率是3m/S，远远不能保证glusterfs在arm上修复速率的保证，所以设计一套程序，在x86等高性能的机器上做glusterfs的纠删码修复

# 总体设计

## 需求规定

Glusterfs fuse挂载客户端记录读写时候错误的文件，采用sqlite数据库（Linux默认带的数据库）或者目录结构形式，这里采用目录结构形式，因为glusterfs本身自带这样的代码，设计实现方便也减少代码bug。

1.客户端修复 2. 可控修复速率 3. 修复进度和损坏文件可查询

## 运行环境

X86环境，跟随fuse客户端

## 设计思路

修改点

1. 在fuse客户端init增加一个线程gf\_thread\_create(ec->record\_pid, NULL, ec\_record\_bad\_file,this)，线程主要功能在/var/lib/gluster/.ec\_reord/目录下创建目录，目录结构采用32位字符创建目录结构（参考POSIX层的.glusterfs方式posix\_handle\_init函数和posix\_create（）函数）。这个目录结构采用32字符构建成一个目录树或者直接创建一个目录，这个目录下放所有的文件（损坏文件一般较少可接受）
2. 在fuse客户端ec代码增加一个线程池，线程池可控（用shd-max-threads参数控制或者简答的.ec\_reord记录一个数值，平台控制修改，定期查询），主线程主要是收集.ec\_reord下的的文件，其他线程则进行修复，修复进程可用ec\_shd\_selfheal（），修复进程不能重写，用ec\_do\_heal这些函数修复。
3. 记录损坏文件方式，本程序只会记录客户端在读写时候有损坏的文件，其他如后端删除掉，磁盘静默损坏等情况的修复就交给glusterfs 原生shd的修复。记录函数应该包含：路径，gfid，layout等这些信息（要不然找不到文件在glusterfs哪里）。修改原生的glusterfs的lookup函数EC\_MINIMUM\_MIN 为EC\_MINIMUM\_ALL（要求全部查找），在ec\_lookup\_cbk（）函数下记录，每一次返回有不完整则记录这个文件

**整体架构**

1. **初始化记录函数**

ec\_t结构体增加char reord\_path\_prex = “/var/lib/gluster/.ec\_reord/”

init 初始化函数在配置参数增加gf\_thread\_create(ec->record\_pid, NULL, ec\_record\_bad\_file,this)线程创建 /var/lib/gluster/.ec\_reord/以及目录结构，随后进入监控状态，主要维护一个文本，文本记录目录里面的文件数变化，为修复进度做呈现

1. **记录读写时候出现的损坏文件**

修改lookup的入参中EC\_MINIMUM\_MIN 为EC\_MINIMUM\_ALL则纠删码查找时候会所有的分片都会查找，出错的文件都能查找到。

ec\_lookup\_cbk函数的ec\_combine里面如果查找的有出错的分片则记录下这个文件的path，layout（xlator里面的部分结构体），损坏分片的brick节点……等等信息

实现write\_fuse\_record函数（），主要整理要记录的信息，写到/var/lib/gluster/.ec\_reord/，注意这里加目录锁，避免修复进程或者其他写进程/线程影响

重点是收集损坏文件包含的信息

1. **修复损坏文件**

ec\_t 结构体增加线程池标识符组和修复双向链表list\_head（可自己重新定义一个结构体）

init函数启动动态线程池thread\_pool（），主线程定期去遍历/var/lib/gluster/.ec\_reord/目录下，借用 读取第二步记录的信息，调用ec\_heal\_do（）

posix\_create

ec\_need\_heal

Ucontext

Syncop\_mt\_dir\_scan(loc) 构建loc

ec\_shd\_index\_heal

ec\_heal\_do

posix\_handle\_path

ec\_gf\_getxattr

fuse-lookup ----

fuse-create -- iostat -quickread -iocache.... - ec\_statfs -ec\_trace- ec\_child\_select

ec\_fxattrop - -ec\_wind\_fxattrop