ComputerSystemEngineering

19302010020 袁逸聪

命令行使用说明

运行Main.java中的main方法后启用命令行

命令行设置了当前访问的FM和当前打开的file

1. 全局功能
    1. 指令集
    2. 切换当前访问的FM
    3. 新建FM与BM
    4. 列出FM,BM
    5. 使用smart工具函数
    6. 持久化
    7. 安全退出(提醒是否要持久化)
2. FM内功能
    1. 列出文件
    2. 新建文件
    3. 打开文件
3. 文件打开后功能
    1. 查看文件全局id
    2. 查看指针位置
    3. 更改指针位置
    4. 查看文件大小
    5. 写入
    6. 更改大小
    7. 关闭

指令	功能
help	指令集
cf	改变当前所在FM
new fm	新建FM
new bm	新建BM
smart-ls	数据系统完整信息
smart-cat	根据文件全局ID读取其全部内容
smart-copy	根据文件全局ID,在FM下复制一个一样的文件
smart-hex	根据LogicBlock ID用16进制输出数据内容
smart-write	给出指针位置和文件全局ID,写入数据

指令	功能
new file	当前FM下新建文件
ls	列出当前FM下文件
ls fm	列出所有FM
ls bm	列出所有BM
open	打开当前FM下的文件
fid	查看打开文件的全局ID
pos	查看打开文件的指针位置
move	更改打开文件的指针位置
size	查看打开文件的大小
write	向打开文件的指针位置写入
read	查看打开文件从光标起一定长度的内容，以UTF-8字符串输出
read all	查看打开文件的所有内容
set size	更改打开文件的大小
close	关闭文件(buffer方式打开必须关闭,否则修改将无效)
save	系统数据持久化为文件
quit	安全退出

架构设计

文件系统分为File,Block两大层

File层

File层由File Manager(简称FM)与File组成,分别代表着"对某个用户课件的文件管理器"和"文件"的概念

在同一个FM内部,文件名禁止重复,不同FM之间则可以

所以对于一个文件而言,可以通过FM编号-文件名唯一确定它,它也具有一个系统维护的id,即便在不同FM之间也不能重复

Block层

Block层的结构与File层类似,同样由File Manager(简称BM)与Block组成,分别代表着"存放真实数据的服务器"和"服务器上的数据包"

Block不需要name,所以对于Block而言,id就是它在BM中的编号,这个编号空间对每个BM是独立的,因为我不能假设服务器内部的资源能够被统一索引,索引Block的id更像是File的name而非id

Logic Block层

在File层和Block层之间,还设置了Logic Block层(简称LB)

对File层而言,所有的数据都存放在Logic Block中,LB层维护从一个logic block到不同BM中blocks的一对多映射关系,以实现block的数据备份

File层总是向LB层申请资源,由LB层自动根据配置的备份数量和当前可用的BM将数据写入多个BM(读取的过程也是一样)

持久化

File

File的重要信息将被持久化写入file meta文件

FM与File之间的从属关系通过目录结构表示

```
/FMs
    /0
        /1.meta
        /2.meta
    /1
    ...
```

meta文件中则记录了文件名,文件的全局id与其引用的logic block编号们

Logic Block

从设计上说,LB层其实是File层的一部分

因为完全可以将File meta中记录的logic block编号全部替换成LB层映射后的结果

添加LB层完全是出于精简(个人认为这也更加便于理解)

LB层将被持久化到如下文件中

```
/Maps
    /logicBlockMap.txt
```

其中记录了Lb的个数,每个Lb自身的编号,映射了多少个Block,以及每个Block的定位信息(即BM编号和BM内部的Block编号)

Block

Block层的结构与File层相似,不同的是除了meta之外,每个block还需要存储data文件

meta中记录block的基本信息,data中记录数据内容

进行持久化时,将对data做一次md5存入meta中,这样就能在data被改动时发现

```
/FMs
    /0
        /data
            1.data
            2.data
        /meta
            1.meta
            2.meta
    /1
    ...
```

重新启动系统时,可以根据持久化信息复原系统的内存数据

如果发生了BM文件夹的丢失,少于备份数量的meta与data的丢失或遭到篡改,系统将在重启恢复时自动根据备份还原丢失或被修改的文件

备份与修复

备份由LB层执行

每当File层向Lb层申请logic block时,Lb层就要募集BM存储数据了

系统默认备份3份,如果可用BM超过3个,则随机选取3个存储,如果少于3个则对每个可用BM都做存储

持久化的文件在做恢复时,会根据LB层持久化数据逐个检查被映射的block,找不到BM文件夹,meta或data文件,meta的md5记录与data中数据无法对应时,记录损坏

如果同一个logic block映射的其他block完好,则用完好的block将坏块修复,否则要抛出异常中断恢复程序

一致性考虑

File层申请写入数据时可能遇到各种问题,导致写入失败

最容易人为造成的情况是,未在系统中创建任何BM,这样空File能够正常创建,但是一旦进行写入,LB层将募集不到任何BM导致写入失败

类似的写入失败一旦发生,FM将根据备份,还原写入前的meta信息(如果前几个block是写入成功的,那么对它们的索引将被抛弃)

对于Block层,除了系统恢复时执行的自动修复之外,block在任何情况下都是不被修改的,File层总是申请新的block来应对变化

而LB层中logic block对block的引用永远不会修改

(由此,文件复制只要复制对logic block的引用即可)

buffer

打开文件可以选择buffer方式或者非buffer方式

用buffer方式打开时,对文件内容的所有查询和修改都将与LB层和Block层断开联系(模拟真实文件系统中的磁盘操作),在最终close时统一向Lb层发出资源请求

而非buffer的每一次读取和修改都将与LB层交互,close时则不做任何动作