Lab1 分布式文件系统

黄悦源 21307110014

设计思路

一共用5个类来实现主体功能：

FileMetadata

维护一个文件的元数据，记录该文件的名称、大小和所属块的集合

NameNode

管理文件元数据，并负责元数据的持久化和恢复。主要维护两张表：

- `FileTable`记录`fileName -> metadata`的映射
- `blockTable`记录`blockId -> dataNode`的映射，查找一个block存在哪个节点上

持久化是借助java自带的序列化机制，只要在类上声明`implements Serializable`就可以支持被序列化为二进制文件。用这样一组函数即可实现持久化：

```
public void persist() {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("fsimage"))) {
            oos.writeObject(this);
        }
    }

    public static NameNode load() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("fsimage"))) {
            return (NameNode) ois.readObject();
        }
    }
```

DataBlock

对数据块的封装，存储一个固定大小的数据块`byte[] data`

DataNode

管理自己路径下的数据块读写

Client

提供用户接口，实例化时创建一个`NameNode`和若干个`DataNode`，以及维护一个已打开文件的集合（这里可以用表记录每个文件当前的offset，简洁起见就没有实现）。主要的两个用户接口如下：

- `read(fileName, length)` 读取指定文件中指定长度的数据，或者读整个文件（`length = -1`）。
- `write(fileName, data)` 往指定文件末尾追加提供的数据

这里有一些实现上的细节。读写中都是通过一个这样的循环访问数据块：

```
for (int i = 0; i < length; ) {
            int inc = Math.min(length - i, b - (offset + i) % b);

            // block[blockIndex*r..blockIndex*r+r-1] 是r个含有相同内容（当前所需内容）的块
            int blockIndex = (offset + i) / b;

            // 这里处理当前块的读写
            // 读只需访问block[blockIndex*r]，写需要修改block[blockIndex*r..blockIndex*r+r-1]
这r个块

            i += inc;
        }
```

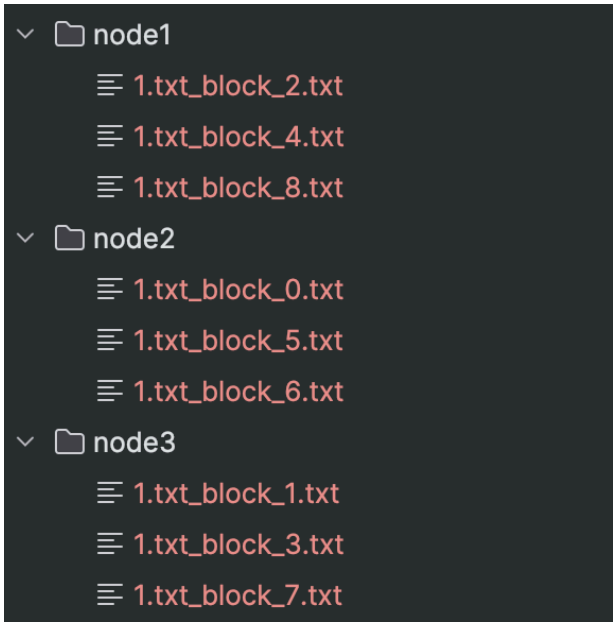
在`fileMetadata`中会记录该文件对应的所有数据块，这些数据块包含了冗余的块，所以比如冗余度r=3时，metadata中会按[0-0, 0-1, 0-2, 1-0, 1-1, 1-2, ...]这样记录数据块。我们通过i可以计算出当前位置对应第几个块，从而定位到记录当前这块数据的r个数据块。

运行结果

首次运行时没有持久化文件可以读，默认初始化：

```
Simple Distributed File System Started.
Commands: open read write close exit
> open 1.txt
File 1.txt opened.
> write 1.txt
Enter data to write:
12345678901234567890
Written data to 1.txt_block_0
Written data to 1.txt_block_1
Written data to 1.txt_block_2
Written data to 1.txt_block_3
Written data to 1.txt_block_4
Written data to 1.txt_block_5
> read 1.txt
Data read from 1.txt_block_0
Data read from 1.txt_block_3
12345678901234567890
> write 1.txt
Enter data to write:
12345678901234567890
Written data to 1.txt_block_3
Written data to 1.txt_block_4
Written data to 1.txt_block_5
Written data to 1.txt_block_6
Written data to 1.txt_block_7
Written data to 1.txt_block_8
> read 1.txt
Data read from 1.txt_block_0
Data read from 1.txt_block_3
Data read from 1.txt_block_6
1234567890123456789012345678901234567890
> close 1.txt
File 1.txt closed.
> exit
Persisting to fs image
System exited.
```

退出后可以看到存储文件的数据块：



再次运行时会从fsimage里加载元数据（从而把文件定位到数据块且得到文件大小），可以直接read之前写的文件

```
Simple Distributed File System Started.
Commands: open read write close exit
> open 1.txt
File 1.txt opened.
> read 1.txt
Data read from 1.txt_block_0
Data read from 1.txt_block_3
Data read from 1.txt_block_6
12345678901234567890123456789012345678901234567890
> write 1.txt
Enter data to write:
111111
Written data to 1.txt_block_6
Written data to 1.txt_block_7
Written data to 1.txt_block_8
> read 1.txt
Data read from 1.txt_block_0
Data read from 1.txt_block_3
Data read from 1.txt_block_6
123456789012345678901234567890123456789011111
> exit
Persisting to fs image
System exited.
```