设计文档

1、实现

本次 lab1 利用已有的 windows 文件系统，通过程序在对应的”bm-xx”文件夹（对应一个 block manager）下创建”xx.data”，”xx.meta”文件来表示 block。block 有 size、indexId、blockmanager 三个成员。而每个 block manager 对应的文件夹下还有一个 id_count 文件来记录 block 的下一个 indexId。

而 Block manager 主要通过属性 block 的 arraylist 来记录属于本 manager 的 block。newBlock()函数调用 newEmptyBlock()函数，再将信息写入 block 对应的 data 和 meta 文件。

File 在读取信息的时候使用 read()函数，通过 meta 文件找到各个 block 以汇总信息。write()函数利用 StringOperation 中对于 byte[]的分割操作，变成二维数组，从而得到与块等大的单个 byte[]，进而写入对应 block。File 中需要维护的指针使用 pos 属性来保持。①读取的时候，获取指针 pos，接着根据 length 获取对应数量 block 的所有 bytes。最后在根据 pos 和 length 对这个 byte[]进行裁剪以获得最终 read 的结果。②写入数据的时候同样获取 pos，计算出需要变更的 block，由于 block 内容不可变，所以，从 pos 所在 block 开始往后的块从 meta 文件中删去，但不去理会，而再创建新的 block 将 byte 写入。写入完成后将 meta 信息进行修改。

File manager 只需要负责索引和创建 File 能力即可，实现比较简单。

2、接口接入

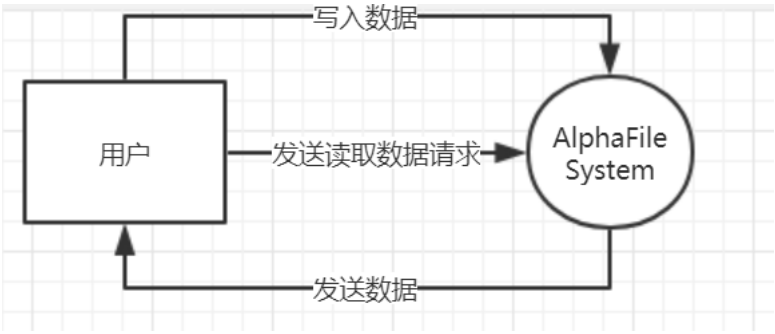
①为实现接口中的方法，我创建了 FileUtil 类，主要在已有文件系统层级提供读写的便利。②另外，我还创建了 `StringOperation` 类来帮助处理一些字符串和 `byte` 数组。③在 `File` 接口的实现中，我另外增加了以下几个方法：`getFileMetaStr()`、`getBlkMngName(String str)`、`getBlkIdStr(String str)`、`getValidBlk(String lineData)`、`randomCommon(int min, int max, int n)`。其中 `getFileMetaStr()`和 `getBlkMngName(String str)`、`getBlkIdStr(String str)`主要用于获取文件保存 `File` 的 `meta` 信息的文件路径和根据其中的信息进行分割获得 `block manager name` 和 `block id` 对应的字符串。`getValidBlk(String lineData)`用于根据 `FileMeta` 内容寻找 `checksum` 一致、对应 `block manager` 存在且对应 `block` 存在其中的 `Block`，否则返回 `null`。`randomCommon(int min, int max, int n)`生成两个随机数，用于 `Dupication` 的时候随机选择两个 `block manager` 存储对应 `data`。

有了以上的一些辅助类，我们在实现 `block` 的 `read` 函数的时候就可以调用 `FileUtil` 的相关静态函数直接读取。实现 `block manager` 的 `newEmptyBlock(int blockSize)`函数的时候首先维护一个 `id_count` 文件和自身的 `blocks` 数组，创建对应 `data` 和 `meta` 文件。`newBlock(byte[] b)`调用上述方法后操作 `FileUtil` 写入 `b` 中内容到对应文件，并修改 `meta` 信息。实现 `File` 则需要维护一个光标，`read(int length)`和 `write(byte[] b)`需要充分利用光标来实现一些细节，不过好在前面提到的三个辅助类以及为此做好了准备。

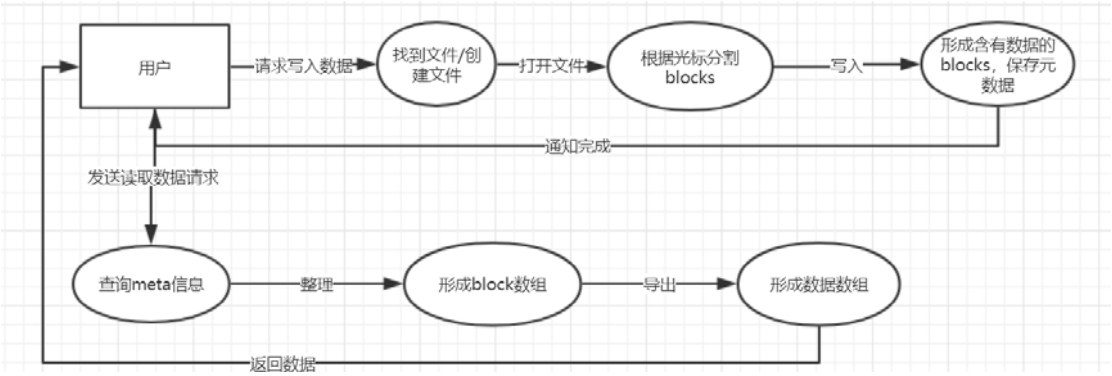
3、Scene

在 File 中读取数据首先调用 java.io.File 来访问某个 File 对象所属的 meta 文件，读取元数据。根据元数据判断所要读取的字节数是否超出，并进行调整（如果超出则改为最大值）。根据 logic block 的信息组成一个 block 数组，读取所需长度的字节并返回。

0 级流程图：



1 级流程图：



2 级流程图