TFS 小组讨论 20070404

10:00,ScienceBuliding 1807S

上周工作进展：

樊楷：	分析了 read/write，fread/fwrite，fstrem 几种不同的文件访问接口设计
涂启琛：	编码实现 datatransfer 服务
杨志丰：	学习 ICE 的 IPC 机制，并行 RPC 机制; 学习 C++编程，Effective C++ master 节点的功能分析和进程功能划分
彭波：	分析 hadoop 的 lock/lease 机制的实现， 分析了 hadoop 的 open/create 接口对 write-once-read-many 模式支持的实现

讨论问题：

1. 我们使用什么风格的文件访问接口？

hadoop 分离了 open/create，分别打开一个 inputstream 和 outputstream, 使用也算简单清晰，但需要程序员适应新的编程模式。

为尽量保持编程习惯，决定：尽量遵循 POSIX 接口方式，open 函数里传递打开文件方式 O_RDONLY,O_APPEND，etc；这样 append 操作，用户应该通过 O_APPEND 方式打开文件的 write 操作来使用。

2. datatransfer 需要支持断点续传否？

考虑主要在局域网环境下使用，不必设计得太复杂。

出现错误，返回合适的错误代码即可。

3. block len 在 master 上是否保存？是否在 chunkreport 中需要报告？文件大小怎么得到的？

hadoop 的 master 实现里，namespace 数据结构中没有 filelen 一项，也没有其它的 file attribute 数据，比如时间，permission 等。

block len 在 hadoop 中是 Block 类的成员，应该是在 chunkreport 中报告，并保存在 master 上，这样计算文件长度 is ok。

4. client 端按 block 缓存 append，是否可以使得多个 appender 的实现更简单？

因为每个 block 还维护了 blocklen，那么每个 block 长度不一样在系统应该是允许的，从 chunkserver,master 维护的信息来说 hadoop 里已经包含了 len 数据。多个 appender 操作，如果给每个 client 分配完整的 block，让 client 在本地 buffer 里写，写满后提交，那么系统实现将会很简单就能保证多个 replica 节点上的数据一致性。唯一的开销是：client 端计算 offset/len 到 blockid 时，必须知道每个 block 的 len 才可以；增加了计算开销。这个计算应该在 client 端完成，那么 open 操作时，master 应该把 block len 的 list 传递给 client，再一次优化，只传递非标准长度的<blockid,len>也可以保证这个数据不大，那么这个想法应该是可行的。

google 的实现，每写一个 record 都需要 master 来协调，保证在一个 block 里有足够的空间（**right?**），这样文件里也会有许多的 padding 数据，**right?**

本版本不考虑多个 appender 支持。那么这个方案预留。（datatransfer 是否会因为它而受影响？）

5. write/append 操作中 chunkserver 是否需要报告 blockrecieved 给 master?

发现 write 操作中，如果出现错误，很难在多个 replica 的 chunkserver 上保证数据一致性。每个 chunkserver 独立对 master 报告，然后在 client 端报告错误的时候，master 可以选择一个 chunkserver 作为最终标准，通过 heartbeat 告诉错误的 chunkserver 删除数据/降低 versionnumber 等方法，这是一种可能的方案；而当所有 replica 数据都写错误时，master 可以任意选择一个，来保持一致。

那么我们集中考虑 append 操作，由 client 最后的报告为准，master 来决定这次操作是 complete 还是 abort，那么不管每个 chunkserver 上新的 block 数据状态，而以 master 的为准。这样可能丢失那个 abort 的 block，并且需要 master 分配 blockid 时具有 sequence 的特点，任意一个 blockid 只被分配一次，hadoop 使用 random 函数来实现，**is it OK?**

最后确定，只支持 append，并且由 client 报告为准，chunkserver 不需要报告。

下周工作安排：

樊楷：	1. 我们确定采用首先实现 POSIX 风格的访问接口，即 open/read/write/close 系列函数。那么详细的接口函数有哪些？ 2. 考虑 C++ stream 的访问接口如何在这样的实现基础上实

	现？ 3． 在 yzf 写的 client 端伪码基础上考虑接口函数实现，同时需要考虑文件句柄表等数据结构的设计
涂启琛：	1．数据传输协议的详细设计。包括控制命令定义(blockid,offset,len,forward machine list etc)，各种可能出现的错误的详细定义。 2．chunkserver 上的数据管理。使用本地文件系统，怎样 hash 文件名来管理 block 文件？checksum,versionnum 怎么存储？
杨志丰：	master 管理 chunkserver 的相关功能分析：围绕 chunk migration between chunkservers 这个功能分析 location of chunks 相关数据结构和 heartbeat 通信内容
彭波：	master 管理 namespace 元数据的相关数据结构和功能分析； 围绕 garbage collection 这个功能分析 location of chunks 相关数据结构和 heartbeat 通信内容