基于 HDFS 的小文件存储与读取优化策略①

张 海, 马建红

(河北工业大学 计算机科学与软件学院, 天津 300401)

摘 要: 本文对 HDFS 分布式文件系统进行了深入的研究, 在 HDFS 中以流式的方式访问大文件时效率很高但是对海量小文件的存取效率比较低. 本文针对这个问题提出了一个基于关系数据库的小文件合并策略, 首先为每个用户建立一个用户文件, 其次当用户上传小文件时把文件的元数据信息存入到关系数据库中并将文件追加写入到用户文件中, 最后用户读取小文件时通过元数据信息直接以流式方式进行读取. 此外当用户读取小于一个文件块大小的文件时还采取了数据节点负载均衡策略, 直接由存储数据的 DataNode 传送给客户端从而减轻主服务器压力提高文件传送效率. 实验结果表明通过此方案很好地解决了 HDFS 对大量小文件存取支持不足的缺点, 提高了 HDFS 文件系统对海量小文件的读写性能, 此方案适用于具有海量小文件的云存储系统, 可以降低 NameNode 内存消耗提高文件读写效率.

关键词: HDFS; 小文件优化; 文件合并; 负载均衡; 云存储

Optimizational Strategy of Small Files Stored and Readed on HDFS

ZHANG Hai, MA Jian-Hong

(Computer Science and Software Engineering, Hebei University of Technology, Tianjin 300401, China)

Abstract: In this paper, the HDFS distributed file system is conducted in-depth research. In HDFS the way of streaming to read and write large files is very efficient, but the efficiency on reading and writing of the mass of small files is relatively low. According to this problem this paper presents a small files based on relational database consolidation strategy. Firstly creating a user's file for each user, then uploading file's metadata information to relational database and the file is written to the user's file when user uploads small files. Finally user via streaming mode to read small files according to the metadata information. When user reads file which size is smaller than the file block, datanode takes load balancing strategy, the datanode of storing data transfers data directly so as to reduce the pressure of the main server and improve the efficiency of file's transfer. The experimental results show that this scheme solves the shortcoming of HDFS reading and writing small files, improves the HDFS file system of reading and writing performance on massive small files. This scheme can apply to massive small files on cloud storage system, and reduce memory consumption of NameNode to improve the efficiency of file's reading and writing.

Key words: HDFS; optimization of small files; merge files; load balance; cloud storage

随着互联网的高速发展, 互联网中的数据也急剧的膨胀, 为了给用户提供更好的服务互联网企业就要保存和挖掘这些数据. 由此产生了云计算[1]的概念, 云计算是当今研究的热门课题, 它很好的解决了大数据的运算与存储的难题, 其中云存储[2]作为云计算的衍生也成为了国内外研究的热点, 在众多云存储的研究中, Hadoop 的分布式文件系统 HDFS[3]作为 Google File System 的开源实现已成为业界研究云计算与云存储, 实现云应用提供云服务参考的标准模型. HDFS 可以用于大规模的分布式存储, 能够构建一个易扩展、

① 收稿时间:2013-10-04;收到修改稿时间:2013-10-29

容错性高、高性能的云存储平台，并且它还为用户提供了一组可靠的稳定的接口可以使开发人员根据自己实际的需求进行开发和扩展，Hadoop 目前已得到许多大公司的青睐，它在海量数据的存储与处理上表现优异得到了广泛的应用.

采用 HDFS 文件系统能够有效地保持数据的一致性，适合一次写入多次读的场合，该架构可以搭建在任意的计算机上来运行保障了可扩展性，HDFS 的备份恢复机制与对分配任务的监控机制都保障了分布式存储的可靠性，HDFS 采用的是流式文件读取非常适用于读取海量级数据. 然而 HDFS 文件系统并非是完美的，它在海量小文件存取上有着一些限制. 本文提出了一种小文件合并方案，通过记录文件的元数据信息将小文件合并到一个大文件中进行存储，并对小于一个块的小文件读取设计了一种负载均衡策略，很好的解决了海量小文件在 HDFS 中的存取问题.

1 小文件存储方案

1.1 HDFS 海量小文件存取问题

HDFS 的框架设计基本上是仿照 Google 的 GFS 理念设计出来的，因此它十分适合处理数据规模大的文件，这样就使得它在处理小文件时效率比较低. 造成 HDFS 在处理大量小文件效率不高主要是由两方面原因引起的: 首先就是 HDFS 将整个系统的命名空间存储在 FSImage 文件中，当集群启动时 NameNode 会将这个文件加载到内存中去，所以当大量小文件存在时这个文件会变得十分庞大因此将消耗 NameNode 大量内存，这样就降低了 NameNode 元数据的检索效率使整个系统的性能降低. 其次就是 HDFS 文件中是分块存储的，默认的数据块大小是 64M，这样每个小文件都会至少占据一个文件块，当客户端对小文件进行操作时都要经过 NameNode 的查找获得文件块所在的地址进行存储与读取，这是一个比较消耗时间的工作，并且在文件读取时要进行大量的 I/O 的操作，与此相对应的是小文件数据流的传输时间远小于控制时间，这也将导致系统存在严重的性能瓶颈[4].

1.2 现有的小文件存储方案

随着微博、空间等社交网站的兴起产生了大量的如日志、用户文件、图片等小文件，所以针对 HDFS 小文件存储问题现在已经有了一些解决方案，例如将小文件存储到 Hbase 中通过文件合并与分解提高文件的存储效率，这种方案的一个缺点就是随着文件的增多会造成 Hbase 大量的合并与分解操作占用大量系统资源严重影响系统的性能，并且 Hbase 只支持简单的字符类型，对其它的图片等类型支持不好还需用户单独的处理[5]. 还有就是利用 Hadoop 提供的归档工具 Hadoop Archives (简称 HAR Files)对小文件进行归档打包[6]，这种方式虽然能够有效的降低大量小文件对 NameNode 的内存消耗但是当用户需要访问时需要超找两次索引才能找到文件效率不高，同时还需要管理员维护运行命令进行归档操作不适合构建基于互联网的云存储平台. 还有一种文件合并的方案是利用 Sequence File 来进行文件的合并[7], Sequence File 是用来存储二进制 key-value 形式的文件，通常利用 Sequence File 来存储小文件时将文件名存入到 key 中文件内容存储到 value 中，这种方式最大的缺点就是由于其中的键值是未排序的文件随机读取效率较低，需要遍历整个文件才能进行读取，并且这种方式不支持文件追加操作，因此合并之前的小文件要在服务器进行缓存，这样文件的安全性就不能得到保障.

2 小文件读写算法的设计与实现

鉴于以上解决小文件方案的不足本文提出了一种基于关系数据库的小文件合并算法，在集群中为每一个用户都建立一个专有文件，利用 HDFS 提供的 Append 文件追加操作以流式的形式将小文件合并到这个文件中，并且还利用关系数据库为小文件建立索引记录小文件的偏移信息，提高了小文件的读取效率. 同时当访问小于一个块(默认 64M)的文件时将其请求转发到数据节点 DataNode 上处理减轻主服务器的压力提高文件传送效率进而提升整个系统的性能.

2.1 架构的设计

改进后的小文件读写算法采取了如图 1 架构图的设计.

图中的 DFS 应用服务器封装了对 HDFS 中文件的基本操作，客户端只需要通过浏览器访问 DFS 服务器就可以对 HDFS 集群中的文件进行操作，这样极大的提高了 HDFS 集群的易用性. HDFS 集群位于架构中的最底层是文件实际存储的地方它由一个名称节点和三个数据节点构成，每个数据节点都部署了封装好的 DFS 应用程序用于减轻 DFS 服务器的压力提高文件传送效率. Metedata 数据服务器(即元数据信息服务器)是

针对优化小文件存取用的，它位于架构中的最上层由 DFS 服务器维护其中的元数据信息，在 Metedata 服务器中利用关系数据库保存了小文件合并时两个十分重要的信息即小文件在合并文件中的偏移值与其文件的大小，在数据库中通过元数据信息建立索引从而提高小文件检索效率.

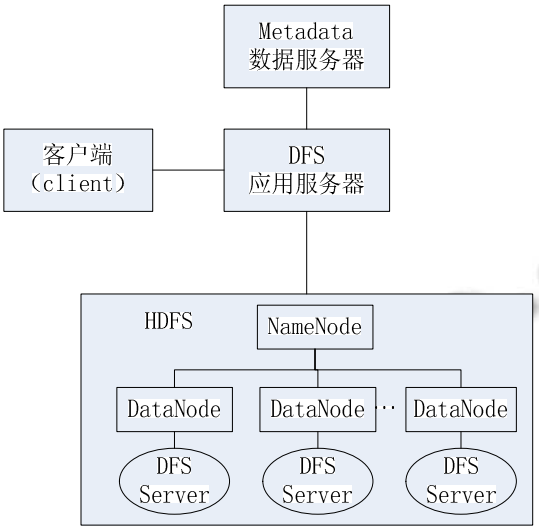

图 1 系统架构图

DFS 应用服务器只关心对外的一个接口，只需提供给它 HDFS 集群中文件所在块的地址它就可以获得数据流并传递给用户，当用户需要读取数据时 DFS 服务器就需要从不同的数据节点将数据拉过来再传递给客户端，这样就会影响文件的读取效率. 如架构图中所示将封装好的 DFS 应用程序部署在数据节点使其直接向客户端发送数据这样会减轻 DFS 服务器压力提高文件的传送效率，在这里需要注意的是由于浏览器不支持分段获取数据这里处理的都是小于一个文件块的文件.

2.2 算法的实现

本文通过设计一个小文件处理模块来处理用户对小文件的存取操作，据统计分析随着 Web2.0 网站的快速发展产生了如日志文件、图片等小于 1M 的海量小文件，这些互联网资源会占据整个集群小文件总数的 90%以上，例如在美国西北太平洋国家实验室的一份研究报告中提到在他们的集群中有 1200 万个文其中 94%的文件小于 64 MB，58%的小于 64KB[8]，因此在这里本文将小文件的阈值设定为 1M. 如图 1 所示当用户上传文件时首先向 DFS 服务器发送上传文件请求，之后进行大小判断如果是大文件则直接上传到 HDFS 集群中，如果文件大小小于设置的阈值我们就将文件以追加写的方式 Append()写到 HDFS 集群中该用户的用户文件中，写入完成后我们要将将文件的大小与偏移值记录到 Metedata 服务器的数据库中以便用户能够快速定位到该文件进行读取. HDFS 文件系统中默认是不支持追加写操作的因此需要进行一下配制，这里需要修改 NameNode 服务器下的 hdfs-site.xml 文件，将 dfs.support.append 的值设为 true. 整个流程如图 2 表示.

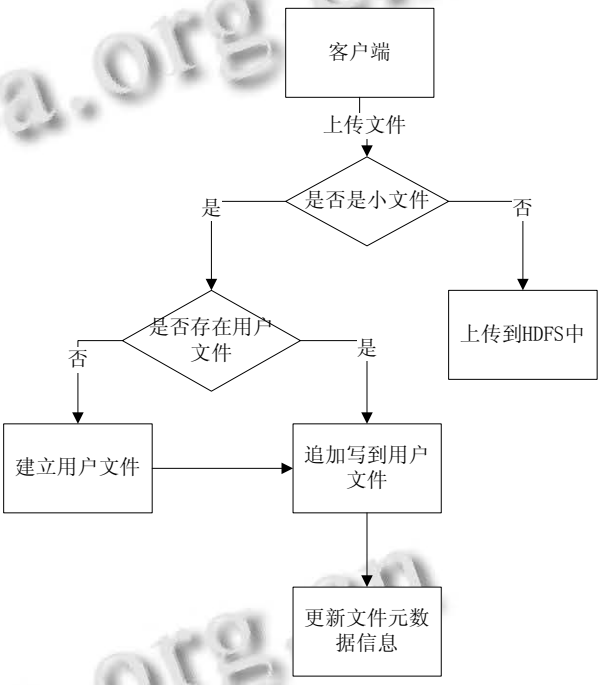

图 2 小文件存储流程

用户读取文件相对写入文件稍复杂一些，如图 1 所示当用户向 DFS 服务器发起读取文件请求时首先也是对文件大小进行判断，当文件小于设定的阈值时就从 Metedata 服务器中的数据库中读取文件的偏移值与大小，然后通过 seek()函数定位到小文件所在用户文件中的位置，最后根据 length 长度以流式的方式从 HDFS 集群中读取小文件. 当文件大于设定的阈值时就进行进一步的判断，如果文件大于一个文件块的大小时我们直接从 HDFS 集群中读取文件，如果文件小于一个块的大小就通过 getFileBlockLocations()函数得到文件块的相关信息，其中包括数据节点的名称与端口号以及该文件在块中的偏移量等信息，这样就可以将请求重新定向到数据节点的 DFS 应用上了，直接由

数据节点来向客户端传送数据提高了文件传送效率，减轻 DFS 服务器的压力. 具体的文件读取流程如图 3 所示.

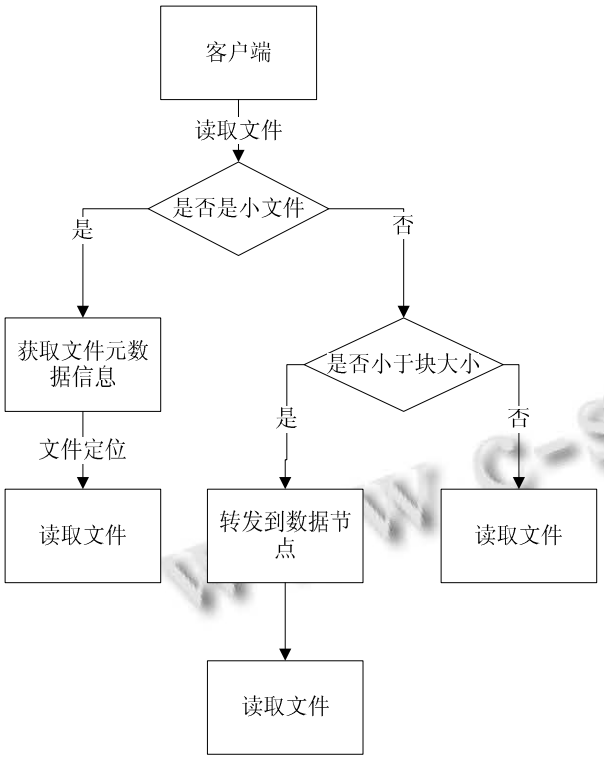

图 3　小文件读取流程

3　实验结果及分析

3.1 实验的软硬件环境

为了测试改进后的 HDFS 架构小文件的存取能力搭建了两组测试环境进行比对. 一个是未经修改的 HDFS 架构, DFS 服务器中部署的 Web 应用只是封装了 HDFS 的常用操作, 另一个是经我们上述优化方案优化后的架构图(如图 1 所示)引入了 Metedata 服务器存储小文件元数据, 并在 DataNode 节点部署了 Web 应用减轻 DFS 服务器压力, 其中两组环境中的 HDFS 集群都是由 1 个 NameNode 节点 4 个 DataNode 节点组成. 本平台采用的软硬件环境如表 1、表 2 所示.

3.2 实验的结果及分析

这里主要分为两组测试, 分别测试改进前与改进后 HDFS 架构在写入文件时间、读取文件时间. 本文的测试流程主要是分为以下几个步骤: 首先我们要由程序生成测试需要的文件, 分别生成 5000、10000、20000、50000、100000 个小文件, 其中这些文件的大小都是小于 1M 的, 平均大小约为 20KB, 其次我们分别在两种架构上读写这几组文件并记录下写入与读取的时间来进行比对, 最后通过对比结果来得出结论. 写入与读取文件的时间如表 3、表 4 所示.

表 1　实验的硬件环境

机器名称	详细配置
DFS 服务器	戴尔台式机 酷睿双核 4G 内存 500G 硬盘
Metedata 服务器	戴尔台式机 酷睿双核 4G 内存 500G 硬盘
NameNode 节点	戴尔台式机 酷睿双核 4G 内存 500G 硬盘
DataNode 节点	联想台式机 AMD 双核 2G 内存 500G 硬盘

表 2　实验的软件环境

机器名称	详细配置
Linux	CetOS 6.0
Hadoop	hadoop-1.0.0
JDK	jdk-6u31-linux-i586.bin
Tomcat	Apache-tomcat-7.0.27.tar.gz
MySQL	mysql-5.5.21-linux2.6-i686.tar.gz

表 3　小文件写入时间

环境 \ 文件数	5000	10000	20000	50000	100000
改进前	573s	1204	2419s	6072s	12214s
改进后	672s	1112s	2143s	5246s	10084s

表 4　小文件读取时间

环境 \ 文件数	5000	10000	20000	50000	100000
改进前	249s	493s	966s	2416s	4782s
改进后	122s	243s	486s	1228s	2466s

通过实验数据我们可以得出经过改进后的 HDFS 架构在上传小文件时随着文件的增多效率要高于原有的 HDFS 架构, 但是优化的效果并不十分明显这是由于 HDFS 架构对文件的 append 操作支持不足造成的, 当进行下载数据时下载的时间远小于文件的上传时间这是由于 HDFS 架构本身就是应用在一次写入多次读取的情景下, 从实验结果我们可以看出优化后的架构在小文件读取上效率提高了很多, 并且随着小文件数

目的不断增多改进前的架构 NameNode 内存使用率会不断的变大，相比之下改进后架构由于将小文件合并为一个文件存储 NameNode 内存并不会随着文件的增多而改变从而降低了 NameNode 内存的消耗提升了整个系统的性能. 本实验采取的小文件的合并策略有效的解决文件格式的限制，还能将用户小文件随时追加到用户文件中避免先将文件缓存到服务器达到一定大小后再上传产生的文件安全性问题，最后通过为小文件建立索引信息能够有效的提高文件的检索效率.

正如我们前面介绍的那样除了对小于 1M 的文件进行合并存储外还对读取小于一个文件块(64M)的文件进行了优化，当用户访问小于一个块大小的文件时将该请求转发给文件实际物理存储节点进行处理，这样做减轻了 DFS 服务器的压力提高了文件传输效率并且当大量用户并发访问时能够通过分流加快用户读取文件的响应时间，在测试时生成 2000 个小于一个块大小的文件进行读取测试，这些文件中文件的大小由 8M、16M、32M、64M 的文件等比例组成，这里分别模拟 500,、1000、1500、2000、2500、3000、3500、4000 个用户随机读取 2000 个文件中的一个文件来对比优化前与优化后系统的响应时间，测试结果如图 4 所示:

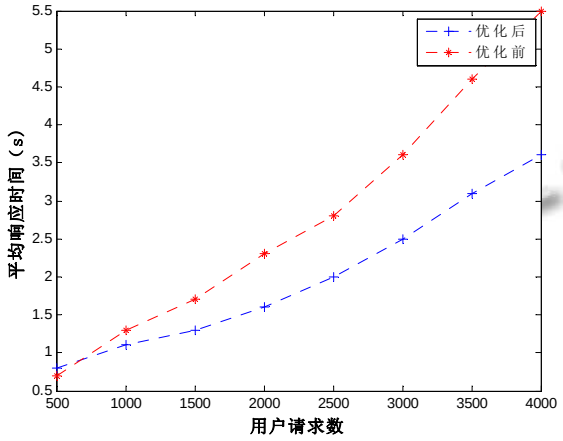

图 4 用户请求响应时间图

通过实验数据我们可以得出优化后的架构当用户请求数较小时优化后的系统响应时间会略高于优化前的架构，这是由于优化后当 DFS 服务器接受到用户请求后会分析用户所需的文件存储的实际物理位置并将请求转发给该数据节点进行处理，但随着用户请求数的不断增多 DFS 服务器的响应时间会不断变大系统会出现性能瓶颈，这时优化后的架构由于会将用户的请求分发到不同的数据节点进行处理从而减轻了 DFS 服务器的压力从而加快了服务器的平均响应时间，与此同时优化后的架构由于直接通过数据存储节点向客户端传送数据从而大大提高了文件的传送效率.

3 结语

本文针对 HDFS 分布式文件系统对海量小文件存储支持不足的问题提出了一种文件合并策略，将每个用户小于 1M 的文件合并到一个用户文件中进行存储，同时还对小于一个块大小文件的读取采取了数据节点负载均衡策略. 通过实验得出我们改进后的架构有效地降低了海量小文件对 NameNode 的内存消耗，提高了小文件读写效率. 同时通过对小于一个块大小文件的请求转发提升了大量用户并发读取文件时的响应时间优化了文件的传送效率，此方案很好地解决了 HDFS 存取小文件问题适用于具有海量小文件的云存储系统.

参考文献

1 刘鹏.云计算.北京:电子工业出版社,2011.
2 Wikipedia.CloudStorage.http://en.wikipedia.org/wiki/Cloud_storage. 2012-5-9.
3 White T. 周敏齐,王晓玲,金澈清,钱卫宁,译.Hadoop 权威指南.北京:清华大学出版社,2010.
4 陈光景.Hadoop 小文件处理技术的研究与实现[学位论文].南京:南京邮电大学,2013.
5 江柳.HDFS 下小文件存储优化相关技术研究[学位论文].北京:北京邮电大学,2010.
6 蔡睿诚.基于 HDFS 的小文件处理与相关 MapReduce 计算模型性能的优化与改进[学位论文].长春:吉林大学,2012.
7 陈剑,龚发根.一种优化分布式文件系统的文件合并策略.计算机应用,2011,31(2):161−163.
8 余思,桂小林,黄汝维,庄威.一种提高云存储中小文件存储效率的方案.西安交通大学学报,2011,45(6):59−63.