文章编号:1671-4598(2014)08-2669-04　　中图分类号:TP391　　文献标识码:A

基于 Cache 的海量图片存取优化方案

陈　渝
(四川民族学院 计算机科学系，四川 康定　626001)

摘要：针对 Hadoop 分布式文件系统（Hadoop distributed file system，HDFS）存储海量图片效率低下的问题，在分析 HDFS 的基本框架以及其固有的文件读写流程基础上，提出了基于 Cache 的海量图片存储优化方案（HDFS based on Cache，CHDFS）；该方案引入了 Cache、预读、文件合并等机制，来提高图片读写的性能，弥补了 HDFS 存储海量图片时的缺陷；采用图片合并的方式减少 Namenode 中元数据的个数，同时提高 Datanode 存储空间的利用率；由于 Cache、预读以及图片合并等操作对用户都是透明的，所以，该方案并没有增加用户使用 HDFS 的复杂性；实验结果表明，CHDFS 方法可以有效地提高图片的存取效率。

关键词：HDFS；海量图片；存储优化；Cache

High Concurrent Access Optimization in HDFS Based on Cache for Mass Images

Chen Yu
(Department of Computer Science, Sichuan University for Nationalites, Kangding　626001, China)

Abstract: To solve the problem of the low efficiency of HDFS (Hadoop Distributed File System) to store mass images, this paper studied the HDFS architecture and the flow of reading and writing files into HDFS, and then proposed an access optimization solution for mass pictures which is based on Cache. It is called CHDFS (HDFS Based on Cache). CHDFS adopts the following ideas to improve the performance, such as establishing appropriate cache , reading ahead pictures, merging more than one images into a big file and so on. File merge can decrease the number of metadata in Namenode and improve the capacity factor of storage space in Datanode. To the client, this solution does not complex the operations to use the HDFS, due to the transparency of cache, read ahead and pictures merge. The experimental data indicates that CHDFS can increase the performance of storing and accessing mass pictures in HDFS without affecting the normal running of HDFS.

Keywords: HDFS; mass pictures; storage optimization; Cache

0　引言

云计算平台利用其超大规模的计算能力、可靠的数据存储、廉价的服务方式等优势，使得其应用变得越来越广泛[1-2]。而分布式文件系统是完成云计算平台必不可少的技术实现[3]，HDFS（Hadoop distributed file system）凭借其开源的优势迅速得到了研究领域和商业领域的认可[4]。

HDFS 采用的主从架构设计模式，是由一个元数据管理节点 Namenode 和多个数据存储节点 Datanode 构成的。图片一般都在 2 M 以下，如果将海量的图片存放在 HDFS 中，就会使得 Namenode 节点的内存成为瓶颈以及浪费 HDFS 集群中大量的存储空间[5]。另外，业务繁忙时，会频繁地与 Namenode 节点进行通信，从而不但增加了网络中数据传输和请求响应时间，也会影响整个云平台的性能[6-7]。

Machey 等利用 HAR 技术实现小文件的合并，提高了 Namenode 中元数据的存储效率，但需要用户自己维护已经被合并的小文件，HAR 并不会主动删除合并后的小文件，而且，这并不能减少对 Namenode 节点的访问次数，无法提高用户读写文件的性能[8]；Dong Bo 等针对 BlueSky 系统中 PPT 课件的存储问题，提出了将 PPT 合并以及采用预取机制对 PPT 的存储进行了优化，但是，文中提到的文件合并是在 Client 实现的，用户必须维护需要合并的文件，无法做到小文件上传至 HDFS 的透明性[9]。

本文针对图片存储的特性设计出了基础 Cache 的海量图片存储优化方案。在 HDFS 集群中引入了 CacheNode 节点，使得数据存入 HDFS 的过程由直接请求 Namenode 写入 Datanode 变为先将数据写入 CacheNode 节点，再由 CacheNode 节点请求 Namenode 从而写入 Datanode。这种设计方式，不但有效地减少了对 Namenode 的请求次数以及 Namenode 节点存储的元数据个数，而且还可以有效利用 Datanode 的存储空间。

1　HDFS 体系结构分析

1.1　主要组成部分

HDFS 主要是由 Namenode，Datanode 及 Client 共 3 部分组成。其中，Namenode 用来操作元数据，主要用来存储和响应元数据的读写请求；Datanode 用来存放数据，内部将文件分割成多个数据块进行存储；Client 从 HDFS 读写数据时，与 Namenode 通信，对 metadata 进行操作，然后，与 Datanode 通信，对数据进行操作。Client 可以看作是提供给用户使用的接口，用户通过调用相应的 API 即可完成上传与下载的功能。

HDFS 这种单个 Namenode、多个 Datanode 的设计方式大大地简化了系统的设计，但是，同时也是造成其存储海量图片时性能瓶颈的罪魁祸首。首先，频繁的文件请求会造成 Name-

收稿日期:2014-04-15；　**修回日期**:2014-05-04。
基金项目:四川省教育厅科研项目(13ZA0135)。
作者简介:陈　渝（1974-），男，四川仪陇人，硕士，副教授，主要从事计算机应用、人机交互技术方向的研究。

node 响应延时；其次，大量的图片元数据会挤爆 Namenode 节点内存。为了更好地优化海量图片的存取性能，我们有必要认真地分析 HDFS 对文件读写流程的支持。

1.2 读写文件流程分析

1.2.1 读流程

通过分析 HDFS 源码，获知其读文件的详细流程，可以简要描述为以下几个步骤：

1）Client 向远端的 Namenode 发起 RPC 请求；

2）Namenode 返回相关数据块列表以及相关的 Datanode 地址；

3）Client 选择最优 Datanode 读取数据块，完成后，关闭相关 Datanode 连接；如果 Client 读取文件没有完成，会继续向 Namenode 发送 RPC 请求；

4）完成读取文件的任务后，Client 就关闭连接。

1.2.2 写流程

通过分析 HDFS 源码，获知其写文件的详细流程，可以简要描述为以下几个步骤：

1）Client 向远端的 Namenode 发起 RPC 请求；

2）Namenode 成功创建一条记录，并返回相应的数据块 ID 和对应的 Datanode 的位置；

3）Client 将文件分成一个个数据包，放入数据队列，并向 Namenode 申请存储副本的 Datanode 列表；

4）将数据包以流的方式依次写入第一个 Datanode，并以管道的形式写入所有的副本中；

5）Datanode 成功存储数据后会返回一个确认数据包，Client 收到确认数据包以后，就会将相应的数据包从确认队列中移除。

6）Client 完成全部数据的写入后，会关闭数据流，等待 Namenode 发出的文件写入完成确认。

从 HDFS 架构设计以及文件读写流程可以看出，每一次文件读写都需要至少两次与 Namenode 进行通信，并且 Client 只有接收到 Namenode 的回应后，才有可能从 Datanode 中读取数据或者将数据写入 Datanode。数据写入 Datanode 时，会将数据写入相关的几个副本中，只有每个副本都写入成功了，数据才算真正写入 HDFS，所以，写入 HDFS 的效率随着副本数的增加而降低。

这种设计模式，随着文件读写频度和副本数的增加，系统的性能就会降低。当然，对于图片这种小文件而言，还会造成 Namenode 内存的紧张以及浪费 Datanode 大量的存储空间。故而，本文提出了基于 Cache 的海量图片存储优化方案，用来解决 HDFS 存储海量图片时的性能问题。

2 基于 Cache 的优化方案设计

本优化方案在设计过程中遵循如下原则：保留原有 HDFS 所有的设计，在此基础上，为用户提供一套新的 API 接口。该接口工作在 Client 与 CacheNode 节点之间。用户只需要调用此套接口即可与 HDFS 交互，完成图片的读写与删除操作。我们称该方案为 CHDFS。

2.1 总体设计方案

如图 1 所示，所设计的 CHDFS 主要由 Client，CacheNode，Namenode 及 Datanode 共 4 部分构成。其中，CacheNode 节点由 Cache 和 U－Metadata 共同构成，Cache 存储实际的图片信息，U－Metadata 存储文件映射关系。Namenode 和 Datanode 对用户是透明的，它们在 CHDFS 架构中，为 CacheNode 提供服务。

Client 通过 TCP/IP 协议与 CacheNode 通信，完成图片读写操作。对 Namenode 和 Datanode 来说，CacheNode 基本上可以看作是 HDFS 架构中 Client 的扩展，因为 CacheNode 需要完成图片合并等相关操作，并将最终的文件上传至 HDFS 集群。

CHDFS 系统是为存储海量图片而专门设计的，对一个图片而言，在实际的应用中，存储一个图片后，几乎不可能会有对图片的修改操作，即使有图片的修改操作，我们也可以按照删除旧图片、存储新图片的方式来实现。故，在 CHDFS 的设计中，并没有考虑修改操作的实现。

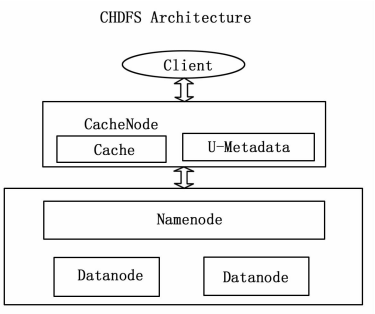

图 1 CHDFS 架构设计

所设计的 CHDFS 是针对海量图片而设计的，它并没有改变 HDFS 的架构设计，而是针对海量图片的特点，新增了 CacheNode 节点，该节点对系统的性能起着关键的作用。接下来将对 CacheNode 节点做详细说明。

2.2 CacheNode 节点设计

在 CHDFS 中，CacheNode 节点起着承前启后的作用。对用户，它提高了用户请求读写图片的响应速度；对后端的 HDFS，它能够缓冲用户请求、合并用户请求，减少用户与 Namenode 的交互。

2.2.1 Cache 块组织方式

在 CacheNode 节点中，最重要的一个设计就是 Cache 的设计，它对性能的提升起着至关重要的作用。为了更好地利用 CacheNode 的内存来提高 Cache 读写效率，这里将 Cache 设计为多内存池结构，在系统初始化阶段，将内存初始化为多个内存池，各个内存池中的 Cache 块大小都不相同，从 16 KB～2 MB 不等，但都是 2 KB 的倍数。

初始化后的 Cache 块处于空闲状态，称为空闲 Cache 块（free cache），是通过各链表组织起来的。每个 free cache 块链表都对应着两个红黑树，分别用来组织有效 Cache 块（valid Cache）、脏 Cache 块（dirty cache），用文件名的 md5 值作为 Cache 块在红黑树中的 key 值。随着系统的运行，Free Cache 块会被读请求和写请求使用，变为 Valid Cache 块或者 Dirty Cache 块组织在红黑树中，并从 Free 链表中摘除。如图 2 所示，给出了某一大小的 Cache 块，其中的一个可能状态。

2.2.2 Cache 块状态变化

Cache 块的状态有 free，valid，dirty 共 3 种，本设计中，Cache 块的状态变化过程如下：

1）系统初始化时，cache 块的状态全部为 free；

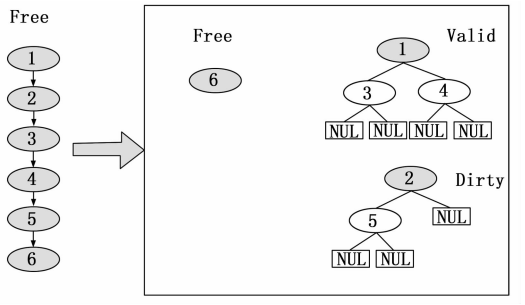

图 2　某大小 Cache 块的某一个中间状态

2）当有一个 Client 读请求，CacheNode 会获取 Cache 块，用来存储从 HDFS 读取的图片信息，此时的 Cache 块，状态变为 valid；

3）当有一个 Client 写图片请求，CacheNode 会获取 Cache 块，用来存储 Client 写入的图片信息，此时的 Cache 块状态变为 dirty；

4）当 dirty Cache 块中的数据持久化到 HDFS 中后，其状态变为 valid；

2.2.3　Cache 块的申请

当 CacheNode 需要一个 Cache 块来存储数据时，需要从 Cache 块内存池中获取一个 Cache 块，获取 Cache 块的优先顺序如下：

1）free Cache 块；

2）valid Cache 块；

3）dirty Cache 块；此时，需要将与此 dirty 相关的 Cache 块内的数据持久化到 HDFS。

其具体实施过程如下：先从 free 链表中获取 cache 块，如果 free 链表为空，则从 valid cache 红黑树中获取 cache 块，如果 valid cache 红黑树为空或所有的 cache 块都正在被使用，再从 dirty cache 红黑树中视图获取。

2.2.4　U－Metadata 的设计

Client 把图片上传至 h _ path，但是，CHDFS 的 CacheNode 节点并不关心 h _ path，而是将图片存储在了一个随机的 Cache 块中，并在随后的某个时刻将该图片连同一些相关的图片合并入一个大文件，上传至了 HDFS 的 h _ path _ alias 路径。可是，Client 并不知道 h _ path _ alias 的存在，它仍然认为自己把图片上传至了 h _ path 路径下，所以，必须使用额外的手段来记录 h _ path 到 h _ path _ alias 的一个映射。

为了完成上述任务，设计了 U－Metadata。U－Metadata 记录 Client 上传的文件的元数据与在 HDFS 中元数据的对应关系，从而使得 Client 做到不关心数据的真正存放形式，只需要知道自己将图片上传的路径即可。

另外，U－Metadata 在设计上还承担着记录各个图片在大文件中的偏移与数据长度的功能。

U－Metadata 记录 h _ path 到 h _ path _ alias 的映射以及图片其他信息都是自动完成的，不需要 Client 做任何额外的工作，对 Client 是完全透明的。

2.3　图片读写流程设计

图片的读写流程主要体现为 Client 与 CacheNode、CacheNode 与 HDFS 之间的通信与数据交互。

2.3.1　读流程设计

在 CHDFS 系统中，读流程设计如图 3 所示，概括来说，有如下几个步骤：

1）Client 与 CacheNode 通过 3 次握手建立 socket 连接；

2）Client 向 CacheNode 发送图片基本信息：大小、文件路径等；CacheNode 查询 dirty 红黑树与 valid 红黑树，以确认该图片是否在 Cache 块中，若在，就将图片数据返回给 Client；若不在，就根据图片大小申请一个大小相当的 Cache 块，查询 U－Metadata，从 HDFS 获取该图片，并返回给 Client。同时，开始预读处理，将与此图片相关的图片，也从 HDFS 读入相应的 Cache 块。

3）读取图片完成后，断开连接。

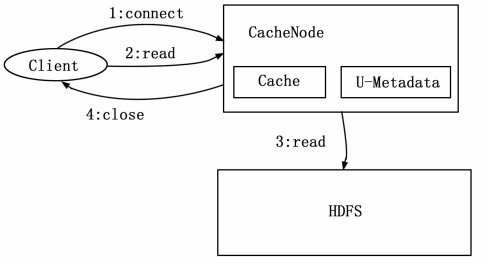

图 3　CHDFS 读文件流程

2.3.2　写流程设计

在 CHDFS 系统中，写流程设计如图 4 所示，概括来说，有以下几个步骤：

1）Client 与 CacheNode 通过 3 次握手建立 socket 连接；

2）Client 向 CacheNode 发送图片基本信息：大小、文件路径等；CacheNode 根据图片大小申请一个大小相当的 Cache 块，将 Cache 块状态设置为 dirty，并将相关元数据写入 U－Metadata。

3）Client 向 CacheNode 写入图片数据

4）写入数据完成后，断开连接。

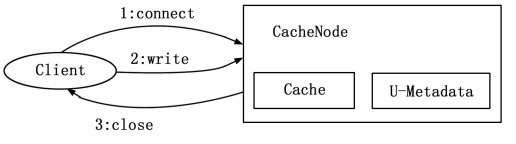

图 4　CHDFS 写文件流程

2.4　图片合并与分解设计

对 CHDFS 系统而言，图片的合并与分解必须是可逆的，因为 CHDFS 是针对图片存储而设计的，图片存储起来是为了将来的使用，如果设计的合并过程是不可逆的，那么，合并之后的数据就可以说是一堆垃圾，毫无利用价值。

1）合并过程流程设计：各个图片通过文件流方式按照顺序依次写入到大文件 BigData 中，图片与图片之间并没有任何分割标记符，但在写入大文件后，在 U－Metadata 中记录每个图片的起始位置和数据长度。

2）分解流程设计：从合并流程的设计上可以看到，图片与图片之间没有任何分割信息，只是在 U－Metadata 中记录了有关信息。那么识别各个图片就要完全依靠 U－Metadata 中记录的元数据信息，通过读取图片格式、大文件路径、起始位置与数据长度等这些元数据，我们完全可以从大文件中将文件流还原成我们需要的图片。

2.5　图片持久化

Client 将图片信息写入 CacheNode 后，就被告知图片已经被写入 HDFS，而此时，数据并不在 HDFS 中，因此对数据持

久化过程展开设计，将 CacheNode 中的数据写入 HDFS 中去。根据 Cache 块的组织特点，只有 dirty cache 红黑树中的 Cache 块数据需要持久化到 HDFS 中。

持久化过程设计如图 5 所示。遍历 Dirty 红黑树，将其中的每个 Cache 块数据合并入 BigFile，并将 Cache 块从 Dirty 红黑树中移至 Valid 红黑树中。最后，将合成后的 BigFile 写入 HDFS。其中，Dirty Cache 到 Valid Cache 与合并为 BigFile，以及将 BigFile 写入 HDFS 这两个过程，都需要更新 U－Metadata 中相关记录。

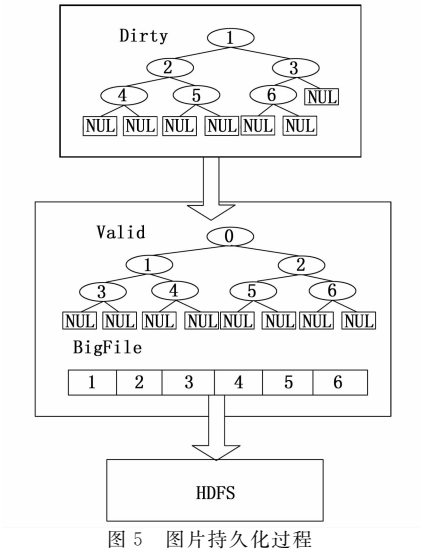

图 5　图片持久化过程

2.6　图片删除

图片经过合并被写入了大文件，作为大文件的一部分而存在，如果在 Client 删除图片时，采用实时删除的方式删除图片在 HDFS 中的实体，那么，既不能减少对 Namenode 的请求次数，也不能减轻 Datanode 读写数据的负担。因此，实时删除图片不能满足本方案要求。

本方案将图片删除设计为异步删除方式，在 Client 请求删除图片时，仅仅在 U－Metadata 中将对应的记录标记为已删除，但并不真正删除在 HDFS 中的数据。只有在一个大文件中包含的所有图片都被删除时，才真正删除 HDFS 中存储的数据。Client 删除图片，只需要修改 U－Metadata 即可，真正删除操作由后台程序异步执行。后台程序周期性地扫描 U－Metadata，获取所有满足条件的文件，执行 HDFS 中文件的删除操作，如图 6 所示。其中，File 是用户上传路径，big 是 HDFS 中的路径，最后的标记 1 表示已删除。

从图 6 可以清晰地看到，从状态 S3 变为 S4 时，由于包含在 big1 中的所有图片都标记为已删除，所以，后台删除程序就将 big1 从 HDFS 中删除，并且删除了 U－Metadata 中的有关记录。big2 中由于还有 File4 没有被删除，故对 big2 不做处理。

3　实验与分析

为了验证本文所设计的基于 Cache 的海量图片存储优化方案的正确性与可行性。分别对优化前后展开实验，对比了 HDFS 与 CHDFS 两个系统的主要性能指标。

3.1　实验设计

3.1.1　实验环境

为了简化实验环境，但尽可能模拟真实的应用环境。搭建

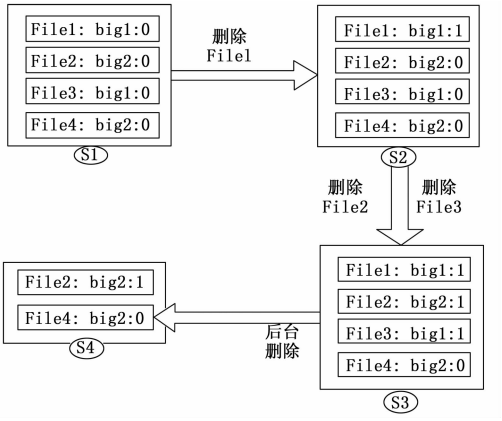

图 6　图片删除过程 U－Metadata 变化

了简易的 HDFS 集群，即 Namenode 与 Datanode 各自被部署在服务器 A、B 上，系统运行时，只有一个 Namenode 节点和一个 Datanode 节点。CacheNode 被部署在服务器 C 上。Client 运行在服务器 D 上，U－Metadata 部署在服务器 E 上。A、B、C、D、E 5 台服务器通过内部 IP 可以相互访问。

注：实验环境中的 U－Metadata 是借助于 Redis 实现的。

3.1.2　实验步骤

1）准备 200 张随机大小的 16 KB～2 M 的图片；

2）上传至优化前（后）的系统中，获取总的上传时间，记为 uptime；

3）待持久化后，分析 Namenode 中新增的元数据个数 metadata _ inc _ num、Datanode 中消耗掉的数据块个数 datablock _ used _ num；

4）读取其中的 20 张图片，记录时间 downtime；

5）对于优化后的系统，手动清空 Cache，再次读取相同的 20 张图片，记录时间 downtime2；

3.2　实验结果与分析

通过实施上述实验，得到了 HDFS 与 CHDFS 两个系统的主要性能指标，如表 1 和表 2 所示。

表 1　上传与下载时间　　(ms)

	uptime	downtime	downtime2
HDFS	27 097	3 378	————
CHDFS	1 684	724	2 490

表 2　元数据与数据块消耗个数　　(个)

	metadata_inc_num	datablock_used_num
HDFS	200	200
CHDFS	5	5

从表格 1 可以看出，CHDFS 与 HDFS 相比，由 uptime 可知，Client 上传图片消耗的时间明显减少了；由 downtime 可知，在图片 Cache 全命中的情况下，Client 请求图片消耗的时间明显减少；由 downtime 与 downtime2 相比较可知，在 Cache 命中率较低时，图片请求消耗的时间有所增加。由此可知，Cache 命中率对请求图片的效率有着很重要的影响。

从表格 2 可以看出，CHDFS 与 HDFS 相比，元数据个数明显减少，消耗的数据块个数也明显减少，间接地说明了

（下转第 2676 页）

5 结束语

本文提出了一种基于 RSSI 的传感器网络定位加密调制算法 AMRSSI，该算法首先通过 RSSI 基本测距原理以及空间传播损耗模型和对数常态分布模型，推导出未知节点和锚节点的关系，为了适应无线传感器网络生存的复杂环境和敌对攻击，算法对数据进行加密，然后利用 RSSI 的差值对载波幅度进行调制，在接收端通过包络检波和解密得到 RSSI，再通过极大似然估计法得到节点的位置信息。该算法对硬件的要求不高，适用于大多数无线传感器网络定位的测距要求。

参考文献：

[1] Chan H，Perrig A. Security and Privacy in Sensor Networks [J]. IEEE Computer，2003，36 (10)：103－105.

[2] Capkun S，Cagalj M，Srivastava M. Secure localization with hidden and mobile base stations [A]. Proc. of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM' 06) [C]. Barcelona，Spain，2006：23－29.

[3] Anjum F，Pandey S，Agrawal P. Secure localization in SN using transmission range variation [A]. Proc. of the 2nd IEEE Intenational Conference on Mobile Ad－hoc and Sensor System (MASS' 05) [C]. Washington DC，2005.

[4] Lazos L，Poovendran R. SeRLoc：Secure range－independent localization for wireless sensor networks [A]. Proc. of the 2004 ACM Worksho Pon Wireless Security (WISE' 04) [C]. Brisbane，Australia，2004：21－30.

[5] Lazos L，Poovendran R，Capkun S. ROPE：Robust position estimation in wireless sensor networks [A]. Proc. of the International Symposium on Information Processing in Sensor Networks (IPSN' 05) [C]. Los Angeles，CA，2005：324－331.

[6] Lazos L，Poovendran R. HiRLoc：High－resolution robust localization for wireless sensor networks [J]. IEEE Journal on Select－ed Areas in Communications，2006，24 (2)：233－246.

[7] Ekici E，Vural S，McNair J，et al. Secure probabilistic location verification in randomly deployed wireless sensor networks [J]. Ad Hoc Networks，2007.

[8] Du W L，Fang L，Ning P. LAD：Localization anomaly detection for wireless sensor networks [J]. Journal of Parallel andDistributed Computing，2006，66 (7)：874－886.

[9] Liu D G，Ning P，Du W L. Detecting malicious beacon nodes for se－cure location discovery in wireless sensor networks [A]. Proc. of the 25th International Conference on Distributed Computing System (ICDCS' 05) [C]. Columbus，Ohio，2005：609－691.

[10] Li Z，Trappe W，Zhang Y，et al. Robust statistical methods fosecuring wireless localization in sensor networks [A]. Proc. othe International Symposium on Information Processing in Sensor Networks (IPSN' 05) [C]. Washington，2005：91－98.

[11] Liu D G，Ning P，Du W L. Attack－resistant location estimation in sensor networks [A]. Proc. of the International Conference on Information Processing in Sensor Networks (IPSN' 05) [C]. Lo Angeles，CA，2005：99－106.

(上接第 2672 页)

CHDFS 系统中，在图片数量相同的情况下，Namenode 内存中元数据个数得到有效控制，Datanode 中数据块空间得到了有效利用。

从对比结果看，基于 Cache 的海量图片存储优化方案 CHDFS 确实达到了预期的目的。

4 结语

HDFS 是针对大文件存储而设计的分布式文件系统，用它来存储海量的图片时，会造成严重的性能瓶颈，浪费大量的存储空间。本文针对此问题，提出了基于 Cache 的海量图片存储优化方案。该方案改变了 HDFS 原来的 Namenode 和 Datanode 两层设计模式，在 HDFS 中引入 Cachenode，提出了现在的 CacheNode、Namenode 与 Datanode 3 层设计模式。该设计方案集成了 Cache、预读、图片合并等有效机制，共同用来提高海量图片在 HDFS 中的读写性能，弥补了 HDFS 存储海量图片时的缺陷。由于 Cache、预读以及图片合并等操作对用户都是透明的，所以，该方案并没有增加用户使用 HDFS 的复杂性。实验表明，本方案用于海量图片的存储是切实可行的。下一步工作就是通过数据挖掘，使得图片合并的算法更加高效合理，提高 Cache 命中率，使得存储效率的提高更加显著。

参考文献：

[1] 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 [D]. 吉林：吉林大学，2012：

[2] 王玲惠，李小勇，张轶彬. 海量小文件存储文件系统研究综述 [J]. Computer Applications and Software，2012，29 (8)：106－109.

[3] 马 灿，孟 丹. 曙光星云分布式文件系统：海量小文件存取 [J]. Journal of Chinese Computer Systems，2012，33 (7)：1481－1488.

[4] Zhang Y，Liu D. Improving the efficiency of storing for small files in hdfs [A]. 2012 International Conference on Computer Science and Service System，CSSS 2012 [C]. 2012：2239－2242.

[5] Linux 公社：MapReduce 及 Hadoop 国内外研究概况 [EB/OL]. http：//www. linuxidc. com/ Linux /2012 － 03/56687. htm，2013－05－13：

[6] chinaz：Facebook 大数据：每天处理逾 25 亿条内容和 500TB 数据 [EB/OL]. http：//www. chinaz. com/news/2012/0823/270885. shtml，2013－04－12：

[7] Apache：The Apache Software Foundation. HDFS Architecture [EB/OL]. http：//hadoop. apache. org/docs/current/hadoop－project－dist/hadoop－hdfs/HdfsDesign. html，2013－04－12：

[8] Zhou L，Fang Z Y，Xiang L Y，et al. Performance optimization of processing small files based on HDFS [J]. International Review on Computers and Software，2012，7 (6)：3386－3391.

[9] Dong B，Zheng Q H，Tian F，et al. An optimized approach for storing and accessing small files on cloud storage [J]. Journal of Network and Computer Applications，2012，35 (6)：1847－1862.