文章编号:1674－2869(2014)01－0069－05

# 中小规模分布式文件系统集群构架的优化方案

白 俊[1],王 新[2],耿 昕[2]

(1. 北京京北职业技术学院机电工程系,北京 101400;
2. 中国石油大学计算机系,北京 102249)

**摘 要**:针对分布式文件系统的应用存在处理小文件效率不高的问题,提出一种适用于中小规模分布式文件系统集群的应用架构,将传统分布式文件系统集群中的内网划分为两个子网:对外子网和对内子网,对外子网中传输与外网之间的交互数据,对内子网中传输分布件系统集群的管理数据。每个数据结点同时与对外和对内两个子网连接,并代替名称节点负责与外网直接的数据交流;名称节点本身只与对内子网连接。对外子网与外网之间使用防火墙设备加强安全性,并使用负载均衡设备将来自外网的数据请求合理的分配到每个数据节点上;增加了缓存机制对系统处理小文件操作进行优化,部署实验环境,设计一个测试程序对缓存效率测试,对1000个100KB的文件,通过模拟多线程连续读取大量文件来测试缓存的性能,实验证明系统设计方案可行,增加磁盘缓存有利于提高系统处理小文件的存取效率,系统优化效果显著.

**关键词**:缓存;中小规模分布式文件系统;管理数据

**中图分类号**:TP311.13 **文献标识码**:A **doi**:10.3969/j.issn.1674-2869.2014.01.014

## 0 引 言

当前,如何更为有效而又廉价地处理海量的用户数据,已经成为互联网公司普遍面临的一个难题.传统的企业架构采用企业级服务器或者小型机等高端硬件,并搭配昂贵的企业数据库软件,不但给互联网公司增加了非常高的运营成本,一定程度上阻碍了互联网公司的发展;也没有足够的扩展性来处理未来互联网公司所面临的惊人数据量.

面对以上问题,很多互联网公司开始自行研发更加廉价并且具有良好扩展性的解决方案.Google推出的分布式文件系统Google File System(简称GFS)是一种已经被证明的高效、高扩展性并且较为廉价的解决方案.GFS配合MapReduce分布式计算框架,可以提供一套有效的数据存储和处理系统,然而GFS并没有向外界开放.Apache和Yahoo! 也推出了一套类似的开源系统Hadoop,并且已经在很多互联网公司得到了广泛的应用.Hadoop主要包括了三部分:

(1) Hadoop Common:一系列用于分布式文件系统和I/O的组件和接口(串行化,Java RPC,稳定的数据结构);

(2) Hadoop Distributed File System(HDFS):运行在大量普通商用机器上的、支持高吞吐量的分布式文件系统;

(3) Hadoop MapReduce:一种在分布式系统上有效处理大数据集的数据处理框架.

Hadoop Distributed File System(简称HDFS)是Hadoop包含的分布式文件系统,具有高效、高扩展性和廉价的特点,很适合存储海量的互联网数据.中国的很多互联网公司尤其是有搜索引擎业务的公司已经开始广泛的使用HDFS,并且越来越深的感受到HDFS带来的好处.但是,HDFS的应用仍然存在一些阻碍:一是没有一种完整、成熟、易于部署的架构;二是HDFS为处理大文件而优化,处理小文件的I/O效率不高.本文主要针对这两个问题进行了研究.

## 1 HDFS应用架构设计

### 1.1 传统HDFS集群架构

HDFS是Hadoop的文件系统组件,它与其他的分布式文件系统有很多不同之处.HDFS具有很强的错误恢复机制,因为它运行的机器往往是价格相对低廉、损坏率较高的机器,所以它要具有快速检测错误和自动恢复数据的能力;HDFS提供以

**收稿日期**:2014-01-03

**基金项目**:十二五国家科技重大专项课题子课题(2011ZX05023-005-006)

**作者简介**:白 俊(1981-),女,河南洛阳人,讲师,硕士.研究方向:数据库,数据分析.

数据流的方式访问数据，适用于批量处理数据，而不是与用户交互进行数据操作，这种方式带来了很高的数据吞吐流量，适合于对大量数据的处理.

传统的HDFS使用了主/从架构来管理集群中的结点，每个集群中都有一个NameNode结点和多个DataNode结点，如图1所示. NameNode作为主结点，管理着文件系统的名字空间和对集群中数据的访问；DataNode是从结点，负责存储和管理数据. 用户将文件存储到HDFS中后，文件会被分割为若干个数据块，并存储到各个DataNode上. NameNode管理着各个文件和数据块的映射表，以及所有对文件的操作，例如文件打开、关闭和重命名. DataNode负责处理客户端的数据读写请求，同时也根据NameNode的指令进行数据块的创建、删除和拷贝操作.

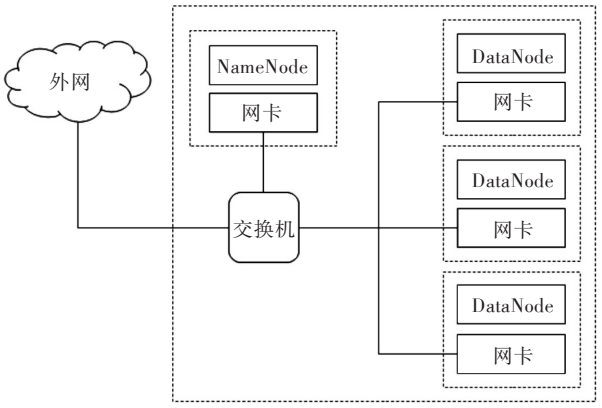


**图1 传统HDFS集群架构**

Fig.1 Traditional HDFS architecture cluster

传统HDFS集群的优点是集群内部结构简洁，只有NameNode、DataNode和交换机三种设备，MapReduce等分布式计算模型可以充分利用这种简洁性进行数据计算的转移，因此集群内部可以达到很高的数据吞吐量，同时结构上的简洁也降低了维护工作的难度，相对提高了数据可靠性. 但是在用于MapReduce之外的环境时，这种架构存在很多不足之处[1-2]：

（1）外网读写数据时需要直接访问NameNode，当访问频繁时会给NameNode造成很大的性能压力.

（2）单交换机的设计使内外网之间的数据传输流量与HDFS集群管理的流量都集中于单个网段中，造成频繁的网络繁忙，限制了数据的传输效率.

（3）外网与内网的直接连接会带来数据安全上的隐患，尤其对于互联网企业，安全远比性能更重要.

### 1.2 应用架构设计

针对传统HDFS集群架构的不足，提出以下优化方案，如图2所示：

（1）将传统HDFS集群中的内网划分为两个子网：对外子网和对内子网，对外子网中传输与外网之间的交互数据，对内子网中传输HDFS集群的管理数据.

（2）每个DataNode同时与对外和对内两个子网连接，并代替NameNode负责与外网直接的数据交流；NameNode本身只与对内子网连接.

（3）对外子网与外网之间使用防火墙设备加强安全性，并使用负载均衡设备将来自外网的数

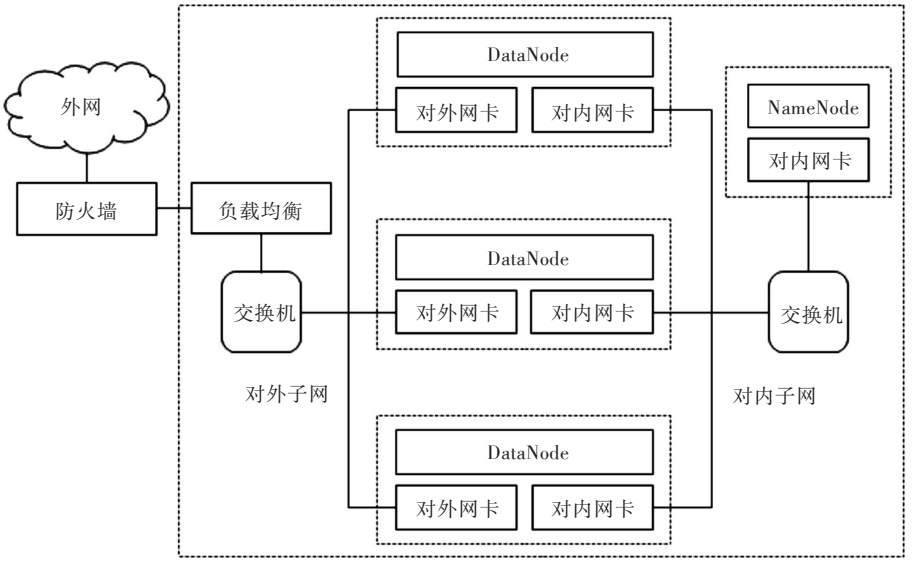


**图2 HDFS应用架构**

Fig.1 HDFS application cluster

据请求合理的分配到每个DataNode上[3].

采用这种架构的优点是:

(1) 采用内外子网的划分方式,减轻了传统HDFS集群中NameNode的性能瓶颈和单个网络中的频繁数据拥堵问题.

(2) 由DataNode负责直接处理外网的数据请求,配合使用负载均衡设备,在减少NameNode压力的同时,提高了数据请求的处理速度和整个集群的数据处理效率.

(3) 加入防火墙设备,增强了数据的安全性.

这种架构也存在不足之处:

(1) 没有改变集群中单NameNode的方式,仍然存在单NameNode带来的性能瓶颈和可靠性问题.

(2) 内外子网划分的方式增加了系统实现和维护的复杂度.

(3) 仍然存在小文件I/O的效率问题.

## 2　缓存设计

### 2.1　缓存设计需要解决的问题

由于上述应用架构仍然存在小文件I/O的效率问题,而互联网中的数据一般具有以下特点:(1)小文件所占的比例很高;(2)大部分文件在创建后很少会被修改;(3)大部分数据读取请求集中于小部分经常被访问的文件,而其余大部分文件被访问到的次数很少.这些特点正符合缓存的使用场景,因此本文使用缓存技术对上述应用架构进行优化.在设计缓存时,主要应解决缓存置换策略问题[4]、所占空间大小、需要缓存文件[5]的大小等问题.

### 2.2　缓存设计方案

(1)置换策略:由于缓存的空间是有限的,在缓存填满时的置换策略,本文使用最早被访问置换策略.

(2)缓存空间:由于DataNode结点需要将更多的空间留给HDFS,如果要求所有经常被访问到的文件都被存储在缓存中,则缓存在本地磁盘中所占的比例$p$由下式可得:

$$p=(f\div n)\div(1+f) \tag{1}$$

其中:$n$为数据块的冗余度,$f$为所有文件中经常被访问的比例.

(3)需要缓存文件的大小限制:过大的文件在置换到缓存中时,会导致很多小文件被置换出缓存;同时文件越大,文件寻址时间造成的效率问题就越小,因此有必要对需要缓存的文件大小进行限制.当寻址时间占读取总时间的比例大于或等于某一给定值时,需要被缓存.数据块大小f由下式可得:

$$f\leq t\times s\times(1-r) \tag{2}$$

其中:$f$为数据块大小(MB),$t$为平均寻址时间(ms),$s$为网络传输速度(MB/s),$r$为设定缓存时寻址时间占读取总时间应达到的比例.

## 3　实验设计与结果分析

### 3.1　实验环境与目的

系统中设置1个NameNode,2个DataNode,2台100M以太网交换机.其上部署Ubuntu 9.04,Hadoop 0.20.203.0,JDK 6,Bash等软件环境.通过部署实验平台,验证设计的HDFS应用架构是否可行.在无缓存和有缓存情况下进行大量小文件读取测试,记录和比较每项测试花费的时间,来验证缓存对于提高HDFS中小文件读取效率的效果.

### 3.2　实验内容

实验的内容是进行缓存效率测试.缓存效率实验所需的数据设定为1 000个100 KB的文件,并设计一个测试程序,通过模拟多线程连续读取大量文件来测试缓存的性能.测试分为6次进行,根据缓存大小与HDFS中所有文件体积之和的比例,将缓存条件分别设置为无缓存、20%缓存,40%缓存,60%缓存,80%缓存,100%缓存,每次测试中读取文件系统中的任意文件2 100次.

### 3.3　实验结果分析

从图3中可以看出:随着缓存容量增大,文件读取时间显著减少,说明缓存对小文件效率优化起到了较好的作用,证明了缓存优化的可行性.而随着缓存容量减少,读取时间快速增加.在20%缓存时,读取时间已经超过了没有缓存的情况,这是因为缓存容量减少时,被访问文件存在于缓存中的几率减少,缓存程序需要频繁地处理文件的置换工作,比较明显的影响了缓存工作的效率.尤其当缓存容量非常小时,这种置换工作带来的消耗使效率降低到了比不使用缓存更差的程度,因此在实际应用中,应当考虑给缓存尽可能大的空间,在空间不足时避免使用缓存.

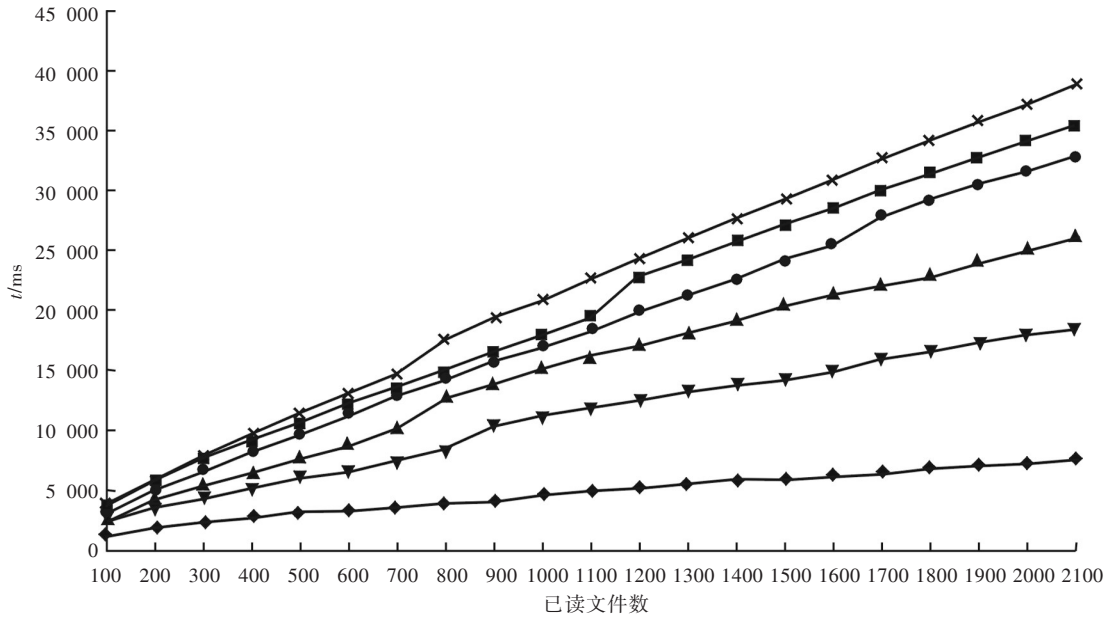


**图3 实验结果折线图**

Fig.3 The line graph of experiment result

注：花费时间(元buffer)； 花费时间(buffer100%)； 花费时间(buffer80%)；
花费时间(buffer60%)； 花费时间(buffer40%)； 花费时间(buffer20%)

## 4 结 语

实验证明：在实际应用中，可以部署本文提出的HDFS应用架构，在集群规模不大时，该架构是一个比较合适的解决方案. 当集群规模增大到一定级别之后，对外子网和对内子网分别被划分为多个交换机连接的网段，该架构需要进行适当的调整. 而磁盘缓存对小文件读取效率的优化有较好的效果.

## 致 谢

衷心感谢北京京北职业技术学院对教师科研工作的大力支持.

**参考文献：**

[1] Borthakur D. The hadoop distributed file system：Architecture and design[R]. Hadoop Docs，2007.

[2] Venner J. Pro Hadoop[M]. New York：Apress，2009：21-53.

[3] 柴黄琪，苏成. 基于HDFS的安全机制设计[J]. 计算机安全，2010，5：22-25.
CHAI Huang-qi，SU Chen. The design of security mechanism based on HDFS[J]. Computer Security，2010，5：22-25.（in Chinese）

[4] 孙玉昕，章瑾. 利用堆排序优化路径搜索效率的分析[J]. 武汉工程大学学报，2013，35(6)：51-55.
SUN Yu-xin，ZHANG Jin. The analysis of heap sort optimization path search efficiency of[J]. Journal of Wuhan Institute of Technology，2013，35（6）：51-55.（in Chinese）

[5] 熊俊俏，周建军，熊诗琪. 快速图形数据采集与现实控制器的设计[J]. 武汉工程大学学报，2012，34（1）：61-63.
XIONG Jun-qiao，ZHOU Jian-jun，XIONG Shi-qi. Design of Journal[J]. fast graphics data acquisition and real controller of Wuhan Institute of Technology，2012，34（1）：61-63.（in Chinese）

# Optimization scheme of cluster architecture of small and medium scale Hadoop distributed file system

***BAI Jun***[1]**,** ***WANG Xin***[2]**,** ***GENG Xin***[2]

（1. Department of Electrical Engineering,Northern BeiJing Vocational Education Institute,BeiJing 101400,China）

2. Department of Computer Science , China University of Petroleum, Beijing 102249,China）

**Abstract:** Aimed at the low efficiency of distributed file system dealing with small files, we proposed an application structure of small and medium-sized distributed file system cluster, the intranet of which was divided into external subnet and internal subnet. The external subnet was used to transport the exchange data to external network. The internal subnet was used to transport the management data in distributed file system. Every data node was connected to both two subnets to exchange data with external network replacing the name node, while the name node was connected only with internal subnet. The safety was enforced by using firewalls between external subnet and internal subnet. The data requests from the external network were assigned to each data node reasonably through load balancing device. Because of the existence of efficiency problem in small files, we optimized small files operation through adding caching behavior, deploying experimental environment and designing a test program for caching efficiency test. We tested the cache performance by simulating multithreading continuous reading large files using 1000 files of 100KB. Experiments prove that the efficiency of processing small file in system is improved by adding disk buffer and the system optimization effect is remarkable.

**Key words:** cache; small and medium-sized distributed file system; management data

本文编辑:陈小平