

doi: 10.12068/j.issn.1005-3026.2019.06.007

# 基于 HDFS 的分布式文件系统

刘 军<sup>1</sup>, 冷芳玲<sup>2</sup>, 李世奇<sup>2</sup>, 鲍玉斌<sup>2</sup>

(1. 东北大学 信息化建设与网络安全办公室, 辽宁 沈阳 110819; 2. 东北大学 计算机科学与工程学院, 辽宁 沈阳 110169)

**摘 要:** 在现有的开源分布式文件存储系统 HDFS 上, 构建一个智能大数据存储系统 IHDFS. 该系统提出了大数据去重模块、大数据放置模块、大数据智能迁移模块和大数据编码模块, 构造了智能分布式文件存储系统, 可以提高用户访问效率, 节省集群的存储空间. 实验结果表明, 数据去重模块很好地节省了存储空间; 数据放置模块合理地分配文件上传的存储层, 使数据上传速度提高一倍; 数据智能迁移模块提高了用户在高等存储层上文件的命中率, 提高了用户获取数据的效率; 数据编码模块节省了集群的存储空间, 节省了大约原来存储空间的三分之一.

**关 键 词:** 多层存储架构; HDFS; 智能; 优化; 分布式

**中图分类号:** TP 311      **文献标志码:** A      **文章编号:** 1005-3026(2019)06-0795-06

## A Distributed File System Based on HDFS

LIU Jun<sup>1</sup>, LENG Fang-ling<sup>2</sup>, LI Shi-qi<sup>2</sup>, BAO Yu-bin<sup>2</sup>

(1. Information Construction and Network Security Office, Northeastern University, Shenyang 110819, China;  
2. School of Computer Science & Engineering, Northeastern University, Shenyang 110169, China. Corresponding author: LENG Fang-ling, E-mail: lengfangling@mail.neu.edu.cn)

**Abstract:** This paper establishes an intelligent big data storage system IHDFS, based on the existing open source distributed file storage system HDFS. The system proposes and implements big data de-duplication module, big data placement module, big data intelligent migration module, and big data encoding module, which improves the efficiency of user visits and saves the storage space of the cluster. Experimental results show that the data de-duplication module can save the storage space. The data placement module provides a reasonable distribution of file upload storage layer, which twice the uploading speed; the data intelligent migration module improves the hit rate of files on the upper storage layer, which improves the efficiency of obtaining data; the data encoding module saves the storage space of the cluster about one third of the original.

**Key words:** multi-layer storage architecture; HDFS; intelligence; optimization; distributed

随着云存储和云计算的快速发展,生活和工作中以及其他活动中产生的数据快速增长,大数据存储和管理成为人们关注的焦点. 大数据的多样化和持续性的增长,使大数据的存储和管理策略更加多样化,带来的问题也更加多样.

本文在现有开源分布式文件存储系统 HDFS 上构建了一个智能大数据存储系统 IHDFS. 该系统提出了四大模块:大数据去重模块、大数据放置模块、大数据智能迁移模块和大数据编码模块. 构建了智能分布式文件存储系统,可以提高用户访问效率,节省集群的存储空间,这是人们使用大数

据存储系统的最迫切需求.

## 1 技术基础

Hadoop Distributed File System(简称 HDFS)是 Hadoop<sup>[1]</sup>核心子项目之一. HDFS<sup>[2-4]</sup>是典型的主/从架构模型系统,是一个可扩展的分布式文件系统,可管理大型分布式数据密集型计算.

HBase 是一个分布式、面向列的开源数据库,源自 Fay Chang 撰写的 Google 论文“Bigtable:一个结构化数据的分布式存储系统”. HBase 在

收稿日期: 2018-04-25

基金项目: 国家自然科学基金青年基金资助项目(61602103); 国家自然科学基金联合基金资助项目(U1435216).

作者简介: 刘 军(1978-),男,辽宁沈阳人,东北大学博士研究生; 鲍玉斌(1968-),男,吉林吉安人,东北大学教授.

Hadoop 之上提供了类似于 Bigtable 的能力.

## 2 相关工作

目前大多数文件去重技术研究都基于传统的文件系统,如基于文件相似性的去重方法,微软等公司也提出了基于粒度的去重方案.

关于数据迁移技术,现有的研究已经涉及在不同的工具之间迁移,例如 HDFS 迁移到 HBase,以及关系数据库到非关系数据库迁移的研究,而关于 HDFS 内部文件如何迁移的研究几乎没有.

关于 HDFS 的编码优化,文献[5]提出了基于 GE 码的编码方式,通过较少的校验块来保证数据可靠传输,然而,解码过程更复杂且效率更低.文献[6]提出了基于范德蒙码的优化技术,相比本文提出的基于柯西矩阵的编码,复杂度更高.

## 3 系统架构设计与实现

该系统设计的初衷是在原有的 HDFS 上进行优化改进,使它成为一个能够快速满足用户存储需求的智能分布式文件系统.系统总体架构如图 1 所示. IHDFS 核心层主要包括:数据去重模块、数据放置模块、数据迁移模块和数据编码模块.

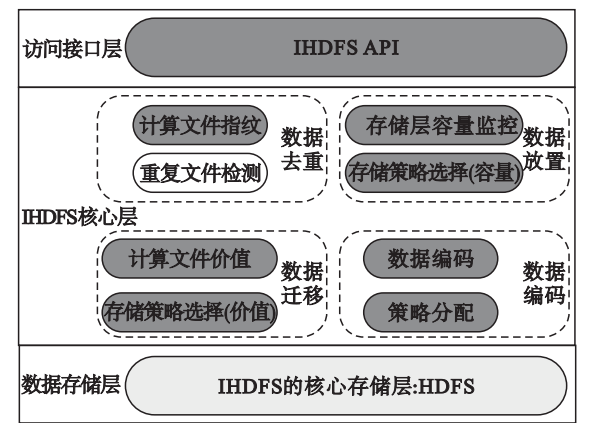


图 1 系统总体架构图  
Fig. 1 Overall system architecture

### 3.1 系统功能需求

1) 大数据存储的去重和放置需求. 对于大数据存储,去除重复的数据能够节省很大的存储空间,降低系统的存储成本,给运营商带来更多的经济价值.

同时,用户想更加快速地将文件放置到大数据系统中,快速地得到反馈. I/O 速度更快的存储设备意味着更高的价格. 一个折中的方法就是将高 I/O 的设备和低 I/O 的设备一起在存储系统

中使用,这就引出了多层存储结构. 对于多层存储架构,大数据的放置是人们重点考虑的问题.

2) 大数据智能化迁移的需求. 如今人工智能离生活越来越近. 在多层存储架构下,大数据系统存储数据的价值也在不断变化,数据管理人员会将价值低的文件存储到低 I/O 设备上,高 I/O 设备用于存储更多高价值文件,这将带来很大的工作量,智能化的迁移数据是解决人工处理数据更加优良的方法.

3) 大数据编码的需求. 每个大数据存储系统都需要确保自身数据的可靠性. 有许多方法可以确保可靠性. 存储多副本来达到可靠性,对于集群的存储资源消耗是巨大的. 还有一种选择就是使用纠删码来为系统提供可靠性.

纠删码的使用可以很好地减少副本数量,为系统节省大量的存储资源,增加系统的存储容量,更有效地利用存储资源.

### 3.2 系统设计与实现

1) 数据存储层. 该系统使用 HDFS 作为底层核心存储层,将数据存储到 HDFS 的各个 DataNode 节点上, IHDFS 主要通过调取 HDFS 的 API 来完成数据的存储, IHDFS 还在 API 上层进行相应的数据处理与调度.

2) IHDFS 核心层. 该层主要是针对现存的 4 大问题而设计的,核心层主要包括 4 大模块:数据去重、数据放置、数据迁移和数据编码. 对于每一个模块又针对相应问题而设计:数据去重模块计算每一个文件的文件指纹,将文件在上传时进行去重工作;数据放置模块选择合理的存储层来存储数据;数据迁移模块是计算文件的价值,根据价值对文件进行迁移,满足用户需求;数据编码模块是通过数据编码产生相应的校验码,通过校验码保证系统的可靠性,减少文件副本数量.

3) 访问接口层. 该层主要是为用户提供 IHDFS 系统的 API 来高效存取数据. 用户通过 API 向 IHDFS 发起上传文件请求;根据上传的文件,计算文件指纹,进行数据重复检测. 间隔一段时间后且系统空闲时,系统会进行文件价值计算,对文件进行迁移.

## 4 系统实现

### 4.1 去重和放置模块

考虑到减小系统的存储空间开销,数据创建过程主要基于去重技术<sup>[7]</sup>进行设计. 对于不同的文件,为它们制作一个唯一标识,来区分文件是否

相同. 对于大数据来说, 使用 MD5 算法做标识可能会有冲突, 所以该系统结合文件的两种 Hash 值来制作唯一标识. 考虑到使用 key - value 的数据格式进行存储, 因此使用 HBase 作为数据库, 将唯一标识作为 Key 值, 指向文件的元数据信息, 用户直接收到上传完成, 如果未找到相同 Key 值, 系统将正常存储文件.

该系统是部署在多层存储设备上的分布式文件系统, 包括异构存储设备, 如虚拟内存、固态硬盘和机械硬盘. 使用集群的存储运行状况来合理地将文件分配给系统, 以实现文件的高效存储. 对于集群的运行状态, 该系统的设计是求出对于集群中总的各个层次的硬盘空闲占比, 分别求出集群中总的空闲 RAMDISK, SSD 和 HDD 的各自空闲空间占各层总空间的占比情况, 通过这个占比情况来为即将创建的文件分配存储策略, 以便能够提高存储效率. 具体的计算如式(1)所示.

$$FR_{ram\_disk} = \frac{\sum_i^n FC_{ri}}{\sum_i^n N_{ri}} \quad (i = 0, \dots, n). \quad (1)$$

式中:  $FR_{ram\_disk}$  代表集群中虚拟内存盘空闲容量的百分比;  $FC_{ri} (i = 1, \dots, n)$  表示每个节点上虚拟内存磁盘的空闲存储容量;  $n$  是集群节点的数量;  $N_{ri} (i = 1, \dots, n)$  代表每个节点上虚拟内存盘的总存储容量. 固态硬盘和虚拟内存公式类似.

由式(1)可以清楚知道集群中各个层次的空闲存储使用率, 可以人为设置阈值, 来判断集群中哪个存储层可用. 系统默认情况下, 设置在 RAMDISK 存储层上, 当  $FR_{ram\_disk}$  大于 20% 时, 集群认为 RAMDISK 存储层可用, 即将创建的文件设置为 Lazy\_Persist 策略; 若  $FR_{ram\_disk}$  小于 20%, 则认为当前 RAMDISK 健康状况不好, 该层无法使用, 会去判断下一存储层是否健康, 后续部分与之前类似.

#### 4.2 智能迁移模块

针对目前多层存储架构中数据放置不合理的问题, 该系统设计了一个智能迁移模块来加快文件的访问速度, 将高价值文件放在更高效的存储层上进行存储. 而影响文件价值的因素有很多, 本文只考虑用户的相似度、用户的活跃度、文件的访问间隔和文件的访问量这四个因素.

系统使用基于项目的协同过滤算法(称为 ItemCF)<sup>[8]</sup>来推荐每个集群中每个用户的文件列表. 它通过分析用户的行为来计算项目之间的相似性. 原始公式如式(2)所示.

$$w_{ij} = \frac{|N(i) \cap N(j)|}{|N(i)|}. \quad (2)$$

其中:  $|N(i)|$  是喜欢项目  $i$  的用户数量; 而分子  $|N(i) \cap N(j)|$  表示同时喜欢项目  $i$  和  $j$  的用户数量. 对于分布式文件系统, 如果文件  $j$  访问很频繁, 那么其中  $w_{ij}$  的值就会很大, 可能接近 1. 该算法会使任何文件都会与比较高频访问的文件有很大的相似度, 这对于分布式文件系统不是一个好的特性. 因此优化了算法惩罚文件  $j$  的权重, 减少高频访问文件和许多文件相似的可能性.

计算文件相似度时, 活跃度高的用户对于他访问过的文件的贡献应远小于活跃度低的一般用户. 考虑到这种情况, Breese 等提出了 IUF<sup>[9]</sup>, 它是用户活跃度对数的倒数的参数, 活跃用户对项目相似性的“贡献”往往少于非活跃用户. 添加 IUF 参数修正项目相似度公式:

$$w_{ij} = \frac{\sum_{u \in N(i) \cap N(j)} \frac{1}{\log 1 + |N(u)|}}{\sqrt{|N(i)| |N(j)|}}. \quad (3)$$

因此, 对于式(3), 可以认为在喜欢项目  $i$  的用户中, 有多少用户同时也喜欢项目  $j$ .

在获得文件之间的相似性后, ItemCF 通过式(4)计算用户对文件的兴趣.

$$p_{ij} = \sum_{i \in N(u) \cap S(j, K)} w_{ji} r_{ui}. \quad (4)$$

这里:  $N(u)$  是用户访问过的文件的集合;  $S(j, K)$  是与文件  $j$  最相似的  $K$  个文件的集合;  $w_{ji}$  是文件  $j$  和文件  $i$  之间的相似度;  $r_{ui}$  是用户  $u$  对项目  $i$  的兴趣程度(对于隐式反馈数据集, 如果用户  $u$  已对文件  $i$  执行了操作, 则  $r_{ui} = 1$ ). 式(4)表明, 与用户过去感兴趣的文件更相似的文件更有可能在用户的推荐文件列表中获得更高的排名.

为每个用户得出一个文件推荐列表, 系统通过将推荐的文件放置到高等级的存储层中, 来提高用户的访问速度. 对于文件的迁移, 系统需要考虑多个因素, 包括文件最近的访问时间、文件在一定时间内的访问量, 还有文件的推荐度  $W_F$ . 综合所有用户对于该文件的活跃度, 使该文件所属推荐列表的用户活跃度之和为文件在系统上的一个推荐度. 系统集成上述因素, 设计了合理的文件系统价值 FV, 表明系统中文件的存在价值:

$$FV = \lambda W_F + \phi V_i + \varphi (T_i - ST). \quad (5)$$

式中:  $W_F$  是文件推荐度;  $V_i$  是文件在一定时间范围内的访问变量;  $T_i$  为系统默认的文件生命周期;  $ST$  是文件最近被访问时间到当前时间的生存期, 如果  $T_i > ST$ , 那么认为该值为 0;  $\lambda$ ,  $\phi$  和  $\varphi$  这三个变量是影响文件价值因素的权重值, 这些值

由超级用户配置,针对该集群侧重的因素来分配.

当系统知道每一个文件在系统中的价值时,就可以根据价值来为文件分配相应的存储策略.本文策略将基于 HDFS 本身自带的四种存储策略:Hot,All\_SSD,ONE\_SSD 和 Lazy\_Persist.系统根据文件的价值将集群中的文件从大到小排序,并获得有序的文件列表.对于高等级的存储层来说,资源是有限的,系统设置了 2 个策略:一个是按容量分配,另一个是按层占比分配.这两种分配策略可以根据用户喜好设置.

4.3 编码模块

在 HDFS 中多副本带来的较低的存储利用率成为时下基于 HDFS 应用的主要问题之一.为了解决数据的多拷贝存储问题,有效地使用磁盘并同时保证集群的可靠性,有必要使用纠删码技术.对源数据文件进行编码以生成校验块,以便在数据丢失或损坏时更好地恢复数据.

该系统主要使用 Reed-Solomon 类纠删码<sup>[10]</sup>进行编码. Reed-Solomon Code 采用伽罗华群 GF(2<sup>w</sup>)中定义的四则运算法则. 伽罗华域中有 2<sup>w</sup> 个值,每个值对应于一个低于 w 次的多项式.因此域上的四则运算被转换为多项式空间的运算. GF(2<sup>w</sup>)域中的加法就是异或,并且通过查表来实现乘法.乘法公式为

$$a \times b = g\text{flog}(g\text{flog}a + g\text{flog}b) \% (2^{w-1}). \quad (6)$$

Reed-Solomon Code 以字为编码和解码的单位,对于大数据块来说,将它拆分到字长为 w 的字,然后对字进行编解码. 变量 D<sub>i</sub>, C<sub>i</sub> 代表一个字.

在数据编码模块中,首先以字为单位分割数据块,并将输入数据视为向量  $D = (D_1, D_2, \cdots, D_n)$ ,然后把编码后数据看成向量  $(D_1, D_2, \cdots, D_n, C_1, C_2, \cdots, C_m)$ ,Reed-Solomon 编码可以看成矩阵运算. 最左边的矩阵是编码矩阵,并且对编码矩阵的要求是任何  $n \times n$  子矩阵必须是可逆的. 为了便于数据的存储,编码矩阵的上部是单位矩阵,下部是 m 行 n 列的矩阵. 将矩阵 B 与向量 D 相乘,求得的值中包含原始数据 D 和生成的校验码 C,当客户端有获取文件请求时,如果编码数据没有丢失,原始数据将直接发送到客户端. 使用 Reed-Solomon 编码的好处是不用进行文件的解码计算,就可以直接发送原始数据,但这样做的前提是没有数据缺失.

如图 2 所示,如果数据 D<sub>1</sub>, D<sub>4</sub>, C<sub>2</sub> 丢失,从编码矩阵中删除丢失的数据所对应的行. 由于 B' 是可逆的,记 B' 的逆矩阵为 B'<sup>-1</sup>,则 B'B'<sup>-1</sup> = I (单

位矩阵). 那么在等式两边同时左乘 B' 的逆矩阵 B'<sup>-1</sup>. 经过运算后,可以得到原始数据,如图 3 所示.

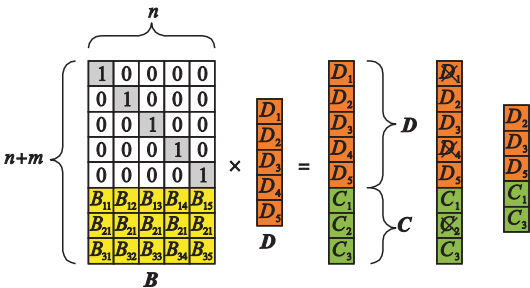


图 2 数据丢失过程  
Fig. 2 Data loss process

在系统的实现过程中,分别使用了范德蒙矩阵和柯西矩阵进行编码,发现柯西矩阵比范德蒙矩阵的优化主要有两点:首先,降低了矩阵求逆的时间复杂度. 范德蒙矩阵求逆的时间复杂度为 O(n<sup>3</sup>),柯西矩阵求逆的时间复杂度仅为 O(n<sup>2</sup>),很明显使用柯西矩阵编码,在效率上要比范德蒙编码快很多;其次,通过有限域变换将 GF(2<sup>w</sup>)域中的每个元素都转换为二进制矩阵,将其中运算的乘法转换成为逻辑相与,降低了乘法运算过程中的时间复杂度.

该系统使用 Reed-Solomon 类纠删码作为系统的容错策略. 恢复数据需要其他原始数据和校验块来恢复该校验块,并且在此过程中,硬盘若经常发生故障,恢复数据对于网络 I/O 和 CPU 的使用都会有比较大的消耗,所以该系统不只采用纠删码,还会借鉴原来 HDFS 的多副本策略,但是副本相对于原来的机制减半处理.

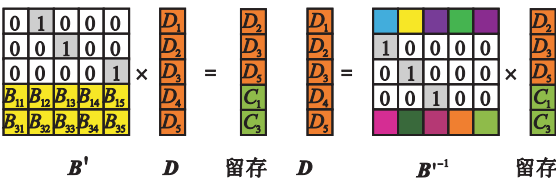


图 3 数据恢复过程  
Fig. 3 Data recovery process

结合多副本机制,该系统实现了两种存储策略,第一种是原始副本存储,该策略是只对原始数据进行备份,而不备份校验数据,当数据缺失时,如果数据的副本没有缺失,那么可以直接从其他节点获得该数据,如果副本也缺失了,那么就只能采用恢复数据的方法来获得原始数据. 原始副本存储策略实质上就是 Reed-Solomon 编码和多副本策略的结合;另一种策略是全备份策略,该策略将编码后生成的原始数据和校验数据一起进行备

份,副本数量仍然是原始副本数量的一半,因此当数据丢失或校验丢失时,仍然可以进行数据恢复.此外,还消除了缺少校验码时重新计算校验码的过程.

## 5 实验与分析

实验环境为 4 个节点的集群,硬件环境为 Intel I3 CPU, 8 GB 内存, 500 GB 机械硬盘, 256 GB 固态硬盘. 软件环境为 64 位 Red Hat Enterprise Linux 6.1 系统, Hadoop 版本为 2.6.0, HBase 版本为 1.1.2.

数据去重模块实验的数据采用 Linux 上的 dd 命令,随机产生不同大小的数据,将这些不同大小的数据分别在 IHDFS 和 HDFS 上进行存储.实验前提是集群中已经存在该数据, IHDFS 需要做的就是返回用户存储完成,而 HDFS 需要进行再一次存储.

如图 4 所示,集群中已经有了相应的数据,对于新上传的数据是重复数据,看到随着文件大小的增加 HDFS 和 IHDFS 的执行时间相差越来越大,当文件达到 10 GB 时, HDFS 的执行时间是 IHDFS 的 6 倍左右,很明显对于上传重复数据 IHDFS 更优秀,可以更快响应用户.

数据放置模块中主要是为了让用户在第一次上传数据时可以快速将文件放到高等级的介质上去,使用户存储的效率更快. 相对于 HDFS, IHDFS 上的存储都是存储在固态硬盘上,而对于固态硬盘已经存在的数据会根据系统需求自动进行数据的迁移. 实验结果如图 5 所示.

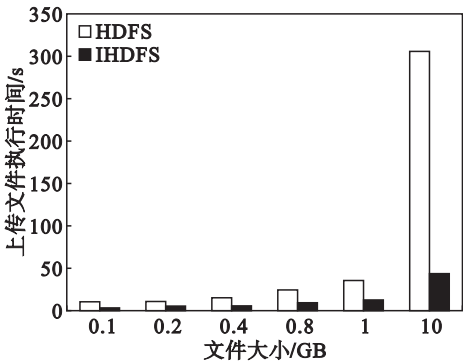


图 4 HDFS 和 IHDFS 上传文件执行时间  
Fig. 4 Execution time of HDFS and IHDFS uploading file

智能迁移模块着重于测试迁移算法的有效性,与原来的 HDFS 文件对比,将文件合理分配到高等存储. 这里对比的是将文件存储在 SSD 上时文件的命中率,实验数据是 MovieLens 百万数据

集和 HP 实验室的 Cello99 Trace 数据集.

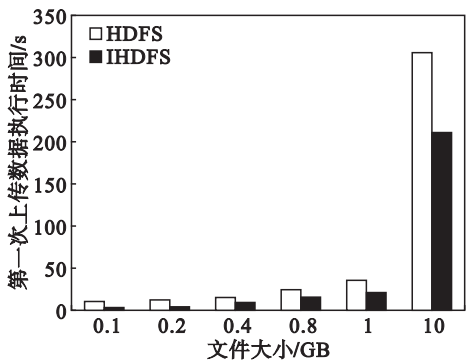


图 5 HDFS 和 IHDFS 上第一次上传数据时间  
Fig. 5 Time to upload data on HDFS and IHDFS for the first time

MovieLens 包含了 6 000 位用户对 4 000 部电影的 100 万条行为数据,将每个行为记录为一次文件访问,按照访问时间排序,抽取 80 万条数据作为训练集,将其中 20 万条作为预测数据. Cello99 Trace 是 HP 实验室的数据集,收集了从 1999 年年初到年底 HP 实验室的存储系统上的数据 I/O 访问记录,抽取 1999 年 1 月的访问记录数据,抽取 1 月 1 ~ 30 日的数据,将该数据作为实验的数据集,将 31 日的数据作为预测数据. 看其是否能在 SSD 上进行命中. 每月根据上述过程计算命中率,最后求平均.

图 6 中,可以看到 IHDFS 在 SSD 层上的命中率高于 HDFS,因为原来的 HDFS 没有针对不同层来存储,而只当作普通存储设备. 在 IHDFS 中,容量分配策略高于层次分配策略,这是因为数据按容量分配可以使用大部分 SSD,可以使该层次达到饱和;而按层分配是按每个层次的比例去分配,每个层次的存储数据相对均匀,而 SSD 层并没有达到饱和状态,在其上存储的文件相对少一些,所以导致文件命中率不如容量分配的高.

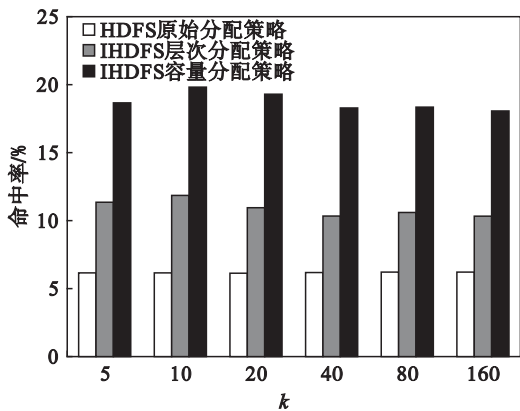


图 6 在 MovieLens 数据集上的命中率  
Fig. 6 Hit rates on the MovieLens dataset

对比图 6 和图 7,发现在 MovieLens 数据集上, $k$  等于 10 时命中率最高,而在 Cello99 Trace 数据集上, $k$  等于 20 时命中率最高, $k$  的不同值会影响推送的准确性,在每个数据集上都应该找到合适的推送数量。

数据编码模块通过纠删码的形式将原始数据编码,生成校验码和原始数据,代替原来的多副本容错机制,本实验就是测试到底纠删码技术在 HDFS 集群上减少了多少的存储量。

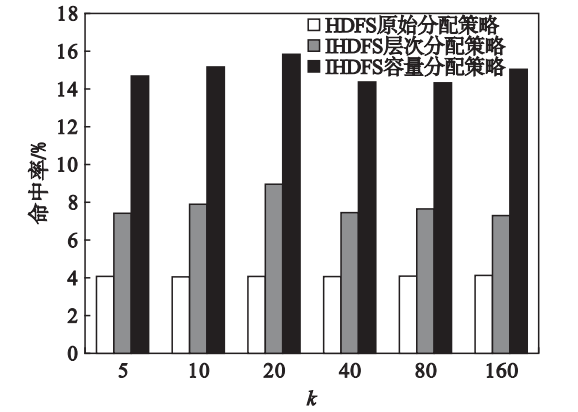


图 7 在 Cello99 Trace 数据集上的命中率  
Fig. 7 Hit rates on the Cello99 Trace dataset

图 8 中,原 HDFS 中的副本数据为 6,那么 IHDFS 上副本数应该为 3,所以对于原备份就是 3 个原始数据加上 1 个校验码,而全备份则是 3 个原始数据和 3 个校验码. 这样的分配策略可以有效防止数据丢失后重新计算校验码。

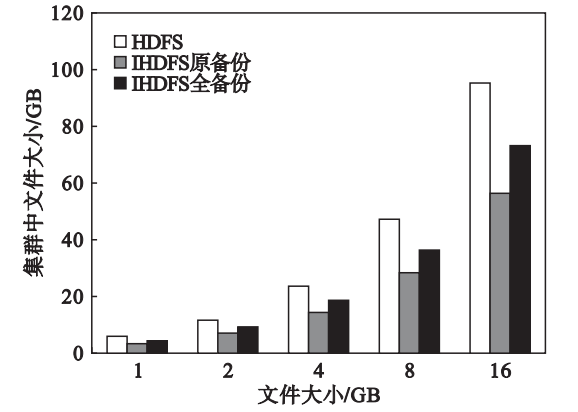


图 8 副本策略的 HDFS 和 IHDFS 文件存储大小对比  
Fig. 8 Comparison of HDFS and IHDFS file storage sizes for copy strategy

6 结 语

本文通过大量的实验证明了 IHDFS 分布式

文件存储系统的高效性和智能化,但是仍然存在不足,系统的进一步提高和完善包括:

- 1) 小文件的存储:对于 HDFS 本身,该设计最初用于存储大文件,存储小文件比较消耗 NameNode 的内存,存储小文件会单独占用一个块。
- 2) 去重过程的原子性:在去重过程中,如果发生中断,系统会返回异常,但系统并不能回退,进一步实现是将去重过程原子化,如果中断或出现异常就撤销,从而利于系统的执行。

参考文献:

[ 1 ] 怀特 T. Hadoop 权威指南 [ M ]. 北京:清华大学出版社,2015.  
( White T. Hadoop: the definitive guide [ M ]. Beijing: Tsinghua University Press,2015. )

[ 2 ] Harter T, Borthakur D, Dong S, et al. Analysis of HDFS under HBase: a facebook messages case study [ C ]// Usenix Conference on File and Storage Technologies. Santa Clara, 2014: 199 – 212.

[ 3 ] Islam N S, Lu X, Wasi-Ur-Rahman M, et al. In-memory I/O and replication for HDFS with Memcached: early experiences [ C ]// IEEE International Conference on Big Data. Washington, DC, 2014: 213 – 218.

[ 4 ] Bok K S, Oh H K, Lim J T, et al. An efficient distributed caching for accessing small files in HDFS [ J ]. Cluster Computing, 2017, 20( 4 ): 3579 – 3592.

[ 5 ] 朱媛媛, 王晓京. 基于 GE 码的 HDFS 优化方案 [ J ]. 计算机应用, 2013, 33( 3 ): 730 – 733.  
( Zhu Yuan-yuan, Wang Xiao-jing. HDFS optimization program based on GE coding [ J ]. Journal of Computer Applications, 2013, 33( 3 ): 730 – 733. )

[ 6 ] 宋宝燕, 王俊陆, 王妍. 基于范德蒙码的 HDFS 优化存储策略研究 [ J ]. 计算机学报, 2015, 38( 9 ): 1825 – 1837.  
( Song Bao-yan, Wang Jun-lu, Wang Yan. Optimized storage strategy research of HDFS based on Vandermonde code [ J ]. Chinese Journal of Computers, 2015, 38( 9 ): 1825 – 1837. )

[ 7 ] Sun Z, Shen J, Yong J. A novel approach to data deduplication over the engineering-oriented cloud systems [ J ]. Integrated Computer-Aided Engineering, 2013, 20( 1 ): 45 – 57.

[ 8 ] 项亮. 推荐系统实践 [ M ]. 北京:人民邮电出版社,2012.  
( Xiang Liang. Practical recommender systems [ M ]. Beijing: The People's Posts and Telecommunications Press, 2012. )

[ 9 ] Breese J S, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering [ C ]//The Fourteenth Conference on Uncertainty in Artificial Intelligence. Madison, 1998: 43 – 52.

[ 10 ] Duursma I, Dau H. Low bandwidth repair of the RS( 10, 4 ) Reed-Solomon code [ C ]//Information Theory and Applications Workshop. San Diego, CA, 2017: 1 – 10.