

Hadoop 备份数据存放策略的改进

周长俊¹, 宗平²

(1. 南京邮电大学 计算机学院, 江苏 南京 210003;

2. 南京邮电大学 海外教育学院, 江苏 南京 210023)

摘要:对于默认的 Hadoop 备份数据存放策略来说,一旦本地的数据副本发生失效,那么就需通过远端机架上存放的备份数据来实现恢复,而对于默认的备份数据存放策略,备份数据存放节点的选择具有随机性,那么可能带来的问题是不同节点间备份数据存放不均衡,数据恢复时由于距离的因素造成内部带宽的巨大消耗。针对上述问题,提出一种改进的备份数据存放策略。该策略将节点之间的距离,节点的负载以及备份数据恢复次数纳入节点选择的考虑范围,由此计算出每个节点的匹配度,随之选出匹配度最高的节点作为远端机架间的备份数据存放的最优节点。该策略不但实现了节点间备份数据放置的负载均衡,而且兼顾了数据恢复时消耗的带宽,将数据副本失效次数纳入考虑,实现了经常失效数据副本的快速恢复。通过在 Hadoop 平台上实现所提出的改进策略,结果达到了预期的要求。

关键词:Hadoop;备份数据存放策略;内部带宽;负载均衡;热点数据

中图分类号:TP31

文献标识码:A

文章编号:1673-629X(2019)01-0011-06

doi:10.3969/j.issn.1673-629X.2019.01.003

Improvement of Backup Data Placement Policy of Hadoop

ZHOU Chang-jun¹, ZONG Ping²

(1. School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;

2. School of Overseas Education, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: On the topic of the default Hadoop backup data storage strategy, once the local data copy fails, backup data stored in the remote rack should be used to restore. However, for the default backup data storage strategy, the choice of storage nodes is random, so the problem that may arise is that backup data is stored unevenly among different nodes, and the internal bandwidth is greatly consumed due to the distance when data is recovered. In order to solve these problems, we propose an improved backup data storage strategy. The strategy considers the distance between nodes, the load of nodes and the number of backup data recovery into consideration, and calculates the matching degree of each node. Thus node with the highest matching degree is selected as the optimal node for storing the backup data between the remote racks. This strategy not only realizes the load balancing of backup data placement between nodes, but also takes the internal bandwidth consumed during data recovery into account, besides that it covers the number of data copy failures and achieve rapid recovery of frequently failed data copies. By implementing the proposed improvement strategy on the Hadoop platform, the results meet the expected requirements.

Key words: Hadoop; backup data placement policy; internal bandwidth; load balance; hot data

0 引言

随着信息技术的迅速发展,互联网以及电子商务的用户数量急剧增加,在互联网中有着各种各样的应用,这些应用无时无刻不在产生数据,最终带来了海量数据^[1]。当前主流的分布式文件系统包含 GFS、TFS、HDFS 等。GFS 在保留传统分布式文件系统特点(伸缩性、可用性等)的基础上添加新的特性,将系统中组

件的失效现象视为常态,处理文件大小为 GB、TB 甚至 PB 等^[2]。TFS 主要用于对海量的非结构化数据的存储与管理,文件大小通常不会超过 1 MB^[3]。王鲁俊等针对 TFS 存在的读写延迟,提出一种新的低延迟、高可用的分布式存储方案,而且也实现了分布式文件系统^[4]。HDFS(Hadoop distributed filesystem),是在通用硬件(commodity hardware)上运行的分布式文

收稿日期:2018-02-02

修回日期:2018-06-05

网络出版时间:2018-11-15

基金项目:国家“863”高技术发展计划项目(2006AA01Z208);江苏省高校自然科学基金基础研究项目(06KJB520079)

作者简介:周长俊(1991-),男,硕士研究生,研究方向为大数据、备份数据存储策略相关;宗平,教授,研究方向为云计算与数据处理。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20181114.1554.006.html>

件系统之一^[5],也是具有高度容错性能的分布式文件系统。HDFS 针对数据访问具有高吞吐量,因此十分适合那些需要操作大规模数据集的应用。HDFS 通过流式来实现文件系统数据的读取。

HDFS 拥有两类节点,分别为 namenode(名称节点)和 datanode(数据节点),以 master-worker 方式运行^[6]。名称节点的主要职责是存储每个文件的各个块到其所在的数据节点信息的映射^[7]。数据节点的主要

职责是存储和检索数据块,而且需要通过心跳机制(heart beat)来实现定期向名称节点发送它们当前所存放的数据块的列表^[8]。客户端(client)表示用户,它通过名称节点获取读取文件的数据块所在数据节点的信息或者是通过名称节点获取写入的数据节点的信息^[9]。

HDFS 整体架构可参见图 1。

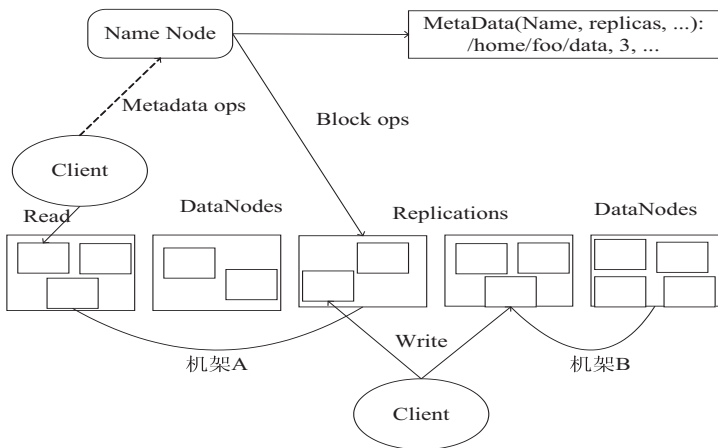


图 1 HDFS 整体架构

1 数据流

1.1 文件读取

客户端通过对名称节点发起文件访问请求,名称节点接收到来自客户端的请求之后查询该文件的起始块的地址。需要注意的是,由于默认复制因子为 3,故每个数据块在文件系统中共有 3 块,其中 namenode 会

考虑客户端和所要读取的数据块所在的数据节点的距离,返回最优的数据节点的位置供客户端读取。当然,若客户端自身为文件系统中的一个数据节点,同时还拥有所请求的数据块的一个副本,那么则直接通过本地读取数据。

文件读取的具体流程可参见图 2。

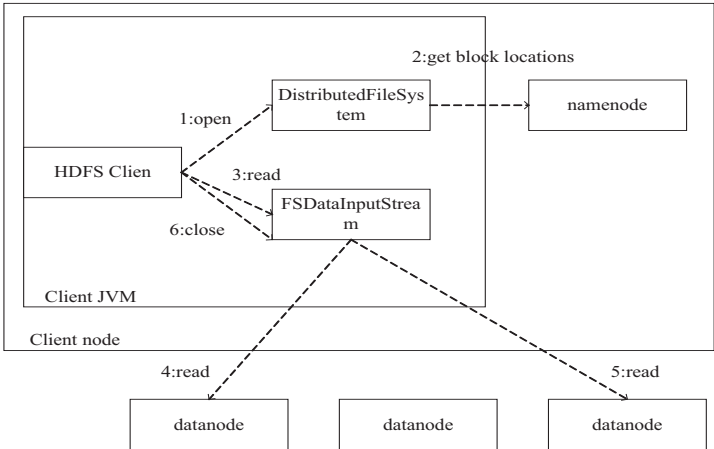


图 2 HDFS 中的文件读取

1.2 文件写入

客户端创建文件时,首先需要访问名称节点来验证用户是否具有新建文件的权限以及新建的文件是否存在于文件系统中,若通过上述验证,则开始写操作。客户端在写入操作之前,还需要考虑的是文件系统的复制因子,需要根据复制因子来选取对应数目的

数据节点数,默认复制因子数为 3。数据写入的过程可以类比为流水线形式,客户端写入数据至第一个数据节点,第一个节点写数据至第二个数据节点,以此类推。这种流式写入提高了写的效率,提升了文件系统的吞吐量。直至每个节点数据写入完毕才能确认客户端的数据写入完毕。

HDFS 中的文件写入具体参见图 3。

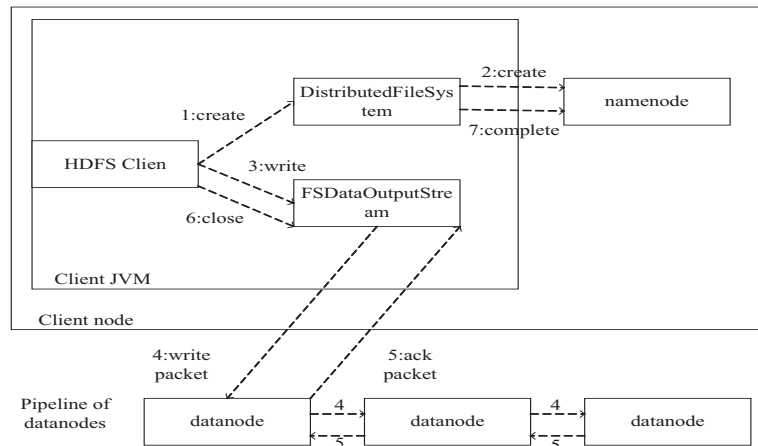


图 3 HDFS 中的文件写入

2 Hadoop 备份数据存放策略

2.1 HDFS 默认备份数据存放策略

HDFS 的副本存放策略是机架敏感 (rack awareness), 默认复制因子的数目 (文件的副本数) 为 3^[10]。假如客户端自身是集群中的某个数据节点, 那么就把第一个副本存放在运行客户端的节点中, 否则 (即客户端运行在集群之外) 随机选择集群中的一个节点作为第一个副本存放的节点。第二个副本的存放需要通过名称节点随机选择与第一个副本不是同一个机架的其他机架上的节点当中。对于第三个副本, 则需要存放在和第二个副本同一个机架但不同的节点当中。上述方法^[11]一方面降低了机架间的写的流量, 实现了写的性能的提升; 另一方面, 显然机架发生故障的概率要远远低于节点发生故障的概率, 在没有改变数据的可靠性和可用性的同时, 也降低了读操作消耗的网络带宽。

HDFS 默认备份数据存放策略参见图 4。

因为 HDFS 的默认副本存放策略为一个任意选择远端机架策略, 所以副本最终存放的节点是无法被控制的。其具体表现如下:

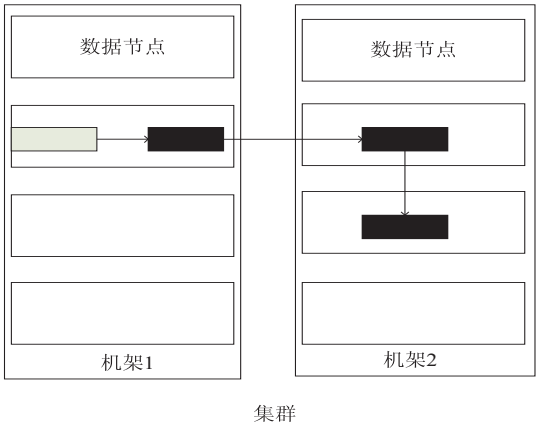


图 4 HDFS 默认备份数据存放策略

(1)使用不同机架来存放同一数据块的不同副本,如果集群包含多个机架,那么可以实现数据块的高可用性以及数据的均衡分布,但是要是集群只拥有一个机架,那么随之而来的问题是无法实现数据块的均匀分布以及无法保证数据块的高可用性。

(2)节点当前的负载数目没有纳入节点选择的考虑范围之内,可能发生的情况是负载较高的节点会被持续写入,使得集群中不同节点负载出现两种极端现象 (某些节点负载很高,其他节点负载很低)。如果产生上述问题,那么带来的问题是当数据块恢复时会造成集群内部网络带宽的巨大消耗。

(3)如果存在对数据块有较大的读请求,那么通过增加副本数目来提升读性能,但是对于超过三个的备份数据块,后续的每一个备份数据存放节点的选择都是随机的,这就会造成上述 (2) 的问题。

当然,HDFS 内部的均衡器 (balancer) 守护进程对上述问题做出了回应。它通过移动数据块,实现降低负载较高的节点的当前数据块数目,从而提升负载较低的节点的当前数据块数目,最终目的是实现集群中数据块数目的均匀分布。诚然均衡器守护进程在一定程度上解决了上述问题,但缺陷如下:对数据块的移动调整是滞后的;存在集群内部带宽资源的不合理消耗,与其事后纠正前期存在的问题,不如在进行远端机架节点选择时就进行优化,从而避免后期的纠正,减少损耗。

2.2 HDFS 备份数据存放策略研究现状

段效琛等^[12]提出了基于初始信息素筛选的蚁群优化算法的副本选择机制,结合遗传算法与蚁群优化算法,使用蚁群优化算法筛选初始信息,利用遗传算法进行全局搜索,最终决定副本的存放位置。

王来等^[13]提出由数据驱动任务分配,使得每一个数据节点都在做合适的任务,从而实现效率的提升。

李晓恺等^[14]提出基于纠删码和动态副本策略的 HDFS 改进系统,对副本使用纠删码来计算冗余程度,对数据块进行分解编码,产生很多的数据分片,将分片存储于系统之中来替代多副本策略。

3 改进的备份数据存放策略

3.1 改进的备份数据存放策略思想

依据上述分析发现的问题,解决的关键应该放在备份副本存放的远端机架的选择上,而不是存放完毕之后再去想着补救之法,因此文中对 HDFS 默认备份数据存放策略进行合理的优化。

之所以将备份数据存放在远端机架节点之上,是为了确保当第一个副本所在节点发生故障时,能够通过远端机架上的备份数据进行数据块的恢复工作。与此同时为了降低数据恢复时消耗的带宽,需要尽可能地选择相邻机架上的节点,实现数据恢复时降低带宽的消耗;由于节点选择时的随机性,可能出现某个机架的某个节点备份数据过于冗余,为了避免出现节点的非负载均衡,故选择节点时需要考虑节点目前已存放的数据备份的数目;最后针对需要经常恢复的热点备份数据,通过调整比例系数来实现失效热点数据的快速恢复,通过阶段性的分析,获取不同阶段的备份数据失效情况,从而实现对不同阶段备份数据的存放进行动态选择。

改进的备份数据存放策略思想如下:当用户上传文件到 HDFS 集群时,名字节点首先任意选择特定数值的不同机架上的节点作为目标节点;其次获取所有目标节点到当前节点的距离值,目标节点当前存放的备份数据的数目,依据这两个数值得出每个节点的匹配度,选取匹配度最高的节点作为备份数据的存放目标节点。最后通过定时任务获取阶段性的备份数据的失效次数,一旦发现失效次数超过指定阈值,通过上述方法重新选择备份数据存放节点(此时需要调整比例系数),用于解决热点失效数据恢复时带来的带宽的损耗。

3.2 节点匹配度评价方法

如上所述,对于每一个目标节点,改进策略会依据节点的当前负载信息、节点间的距离来给出该目标节点的匹配度,将该值作为目标节点选择的依据,具体计算方式如下:

$$P_{(i)} = \alpha_1 \frac{1}{C_{(i)}} + \alpha_2 \frac{1}{L_{(i)}}$$

其中, $C_{(i)}$ 表示编号为 i 的目标节点当前的负载数目,该值与节点的匹配度成反比; $L_{(i)}$ 表示编号为 i 的目标节点到当前的网络距离,该值与节点的匹配度成反比; α_1 、 α_2 为比例系数,需要根据不同的匹配度评估

时刻进行合适的指定,分别针对用户上传文件至集群时和针对备份数据失效次数超过阈值时即热点备份数据进行恢复时进行的节点选择。

3.3 算法描述

基于上面的思想,对改进备份数据存放策略给出具体的算法描述:

(1) 当有新的数据块提交或者备份数据失效次数超过阈值,名字节点获取已匹配节点列表中节点的数目,若该数目小于指定值 K ,则转向步骤 2,否则转向步骤 3;

(2) 随机选取节点,若选取节点不在已匹配节点列表中并且该节点与已匹配节点列表中任一节点不在同一机架,则将该节点添加到待匹配节点列表;

(3) 计算待匹配节点列表中每一个待匹配节点的匹配度,将计算结果记录至已匹配节点列表,清空待匹配节点列表;

(4) 对已匹配节点列表按照匹配度进行排序;

(5) 返回已匹配节点列表中匹配度最高的节点,作为备份数据存放的目标节点。

3.4 实现

利用 Hadoop 默认备份策略进行节点选择的步骤如下:

(1) 通过默认数据块存放策略类 BlockPlacementPolicyDefault 的 chooseTarget() 方法进行节点选择;

(2) 通过调用 NetworkTopology 类的 contains() 方法来判断客户端是否是集群中的一个数据节点,若客户端是集群中的一个数据节点,则通过 chooseLocalStorage() 方法选择客户端作为第一个节点;否则通过调用 chooseRandom() 方法随机选择集群中的任一机架上的一个节点,通过 chooseLocalRack() 方法在本地机架上随机选择一个节点作为第一个节点;

(3) 通过调用 chooseRemoteRack() 方法在远程机架上随机选择一个节点作为第二个节点,使用 NetworkTopology 类的 isOnSameRack() 方法来判断该节点是否与前一个节点在同一个机架上,若不在同一机架上,则将该节点作为第二个节点;

(4) 通过调用 chooseLocalRack() 方法在第二个节点的本地机架上随机选择一个节点作为第三个节点。

上述为 Hadoop 默认备份数据存放策略的具体实现,在此基础上,按照文中提出的算法做出如下修改,具体实现如下:

```
public class ImprovedBlockPlacementPolicy extends BlockPlacementPolicy {
    /* NetworkTopology 用来表示 Hadoop 集群的网络拓扑结构 */
    private NetworkTopology clusMap;
```



```
/* 其余私有成员变量未标注 */
DataNodeDescriptor chooseTarget(
// 文件系统相关信息
FSInodeInfo fsInodeInfo,
// 备份文件的份数
int countReplicas
DataNodeDescriptor descriptor,
List<DataNodeDescriptor> targetNodes,
long blockNum) {
// 省略部分重复代码
double[] calc = new double[randomNum];
for (int k = 0; k < randomNum; k++) {
// 进行节点选择
DataNodeDescriptor dataNodeDes = choseRandom(...);
// 计算节点匹配度
calc[k] = match(dataNodeDes);
}
// 获取匹配度最高的节点
double max_calc = sort(calc);
return choose_max(max_calc);
}
```

4 实验与结果分析

为了验证改进的 Hadoop 备份数据存放策略相对于默认备份数据存放策略所能带来的性能上的提升,进行了以下实验。其中 Hadoop 版本为 2.7.0,总共包含 5 个机架(机架分别命名为 A,B,C,D,E),每个机架包含 6 个数据节点(每个数据节点对应一台机器),默认每个数据节点拥有相同的性能。集群中任意两个节点之间的距离为这两个节点到最近公共祖先的距离的和。通过上述定义,可得如下分类:

- (1) 同一节点上的进程;
- (2) 同一机架上的不同节点;
- (3) 同一数据中心不同机架上的节点。

给出如下三种描述(数据中心: d_i ,机架: r_i ,节点: n_i):

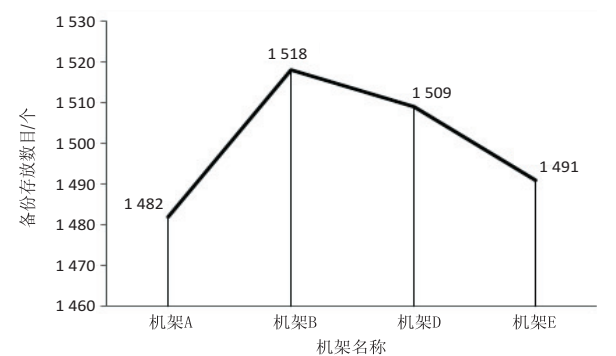
- (1) $distance(d_i/r_i/n_i, d_i/r_i/n_i) = 0$: 同一节点上的进程;
- (2) $distance(d_i/r_i/n_i, d_i/r_i/n_2) = 2$: 同一机架上的不同节点上的进程;
- (3) $distance(d_i/r_i/n_i, d_i/r_2/n_i) = 4$: 同一数据中心不同机架上的节点。

为了规避特殊性,保证选取提交数据节点的随机性,实验中,通过随机算法选中机架 C 中的某个节点作为提交数据的客户端。本次实验累计选取 3 000 个大小相同的数据块,由 Hadoop 默认的备份数据存放策略可知,对于数据一个数据块,只有一个数据块备份存

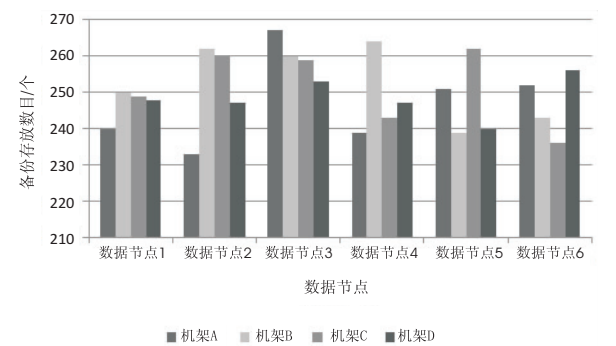
放在本地一机架的节点之上,故另外 4 个机架(A,B,D,E)总共需要存放 6 000 个数据块,本地机架(机架 C)则需要存放 3 000 个数据块。默认实验初始时每个机架的每个数据节点上存放的数据块数目为 0。

对于默认的 Hadoop 备份数据存放策略,备份数据的分布情况如图 5(a)和(b)所示。对于改进的备份数据存放策略,指定系数分别为 $\alpha_1 = 0.5$, $\alpha_2 = 0.5$,目标节点的数目固定为 12,通过改进算法计算这 12 个节点的匹配度,选取其中匹配度最高的作为备份数据存放的节点,备份数据的分布情况如图 5(c)和(d)所示。

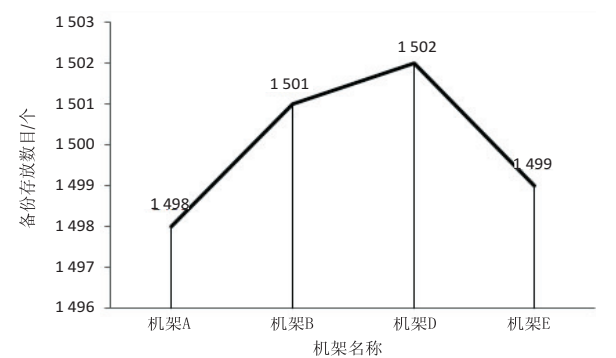
改进算法中考虑到节点间的距离,节点当前的负载情况,热点失效备份数据三个因素,初始时节点当前的负载皆为 0,热点数据也还未有相关统计,故初始时距离机架 C 近的节点能够获得较大的评价价值,因此更多的副本将会被存放在距离较近的节点上。通过采用改进备份策略算法可知,每一个存放在非机架 C 上的



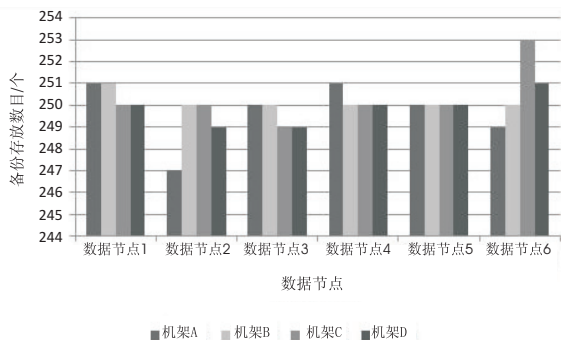
(a) 默认情况下各机架存放备份数据数目



(b) 默认情况下各机架中各节点存放备份数据数目



(c) 改进算法下各机架存放备份数据数目



(d) 改进算法下各机架中各节点存放备份数据数目

图 5 备份数据分布

备份数据到节点 C 之间的距离大致为 2.571 4, 相对于默认的备份策略距离大致缩短了接近 21.4%, 同时由于后续对节点的评价是考虑到节点目前的负载情况, 故节点获得了更好的负载情况, 节点间的备份数据数目之差不超过 4, 而采用默认备份策略时最高可达 32。

当然, 随着改进算法的引入, 随之而来的负面影响则是选取目标节点时增加了时间损耗。通过分析改进算法可知, 时间复杂度为 $O(n)$, 其中 n 为实验时设定的匹配节点数目。本次实验中指定 $n = 12$ 。默认备份数据存放策略与改进备份数据存放策略对从接收到文件存储请求直至文件存储完毕所耗的时间可参见图 6。

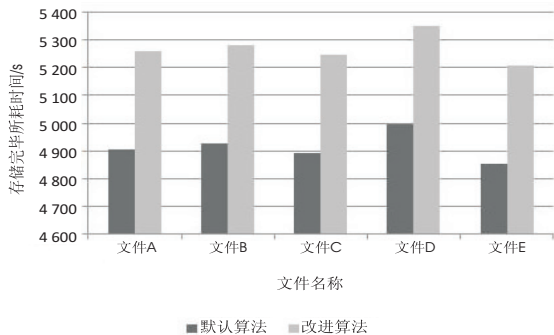


图 6 默认算法与改进算法所耗时间对比

通过图 6 可知, 改进算法相比于默认算法平均需要多耗费 357 ms, 但相对于改进算法在数据恢复时所带来的内部带宽的消耗减少, 各节点分布备份数据更加均匀, 基本可以忽略不计, 故而改进算法确实提升了默认备份数据存放策略的性能, 解决了默认备份数据存放策略所存在的问题。

5 结束语

基于节点目前的负载情况, 节点间的距离以及失效的热点备份数据, 提出一种通过适当的系数来选择最优的备份数据存放节点的算法。该算法首先查询目标节点集合是否达到预期数目, 若未达到预期目标节

点数则选取若干节点填充目标节点集, 之后依次计算出每一个节点的匹配度, 选取匹配度最大的节点作为备份数据的存放节点。实验结果验证了该算法的有效性。

默认备份数据存放策略的备份数据为 3 份, 对于经常需要访问的数据块(热点数据)降低了读的性能, 对于不需要经常访问的数据(冷却数据)则存在存储空间的浪费, 故而后续的研究将着重关注通过实现备份数据块的数目的动态变化来提升 HDFS 的读性能, 降低默认备份数据存放策略存在的存储空间的浪费。

参考文献:

- [1] 孟小峰, 慈 祥. 大数据管理: 概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-149.
- [2] 陈 晨, 陈达丽. 谷歌大数据技术的研究及开源实现[J]. 软件产业与工程, 2015(5): 31-36.
- [3] 王铃惠, 李小勇, 张铁彬. 海量小文件存储文件系统研究综述[J]. 计算机应用与软件, 2012, 29(8): 106-109.
- [4] 王鲁俊, 龙 翔, 吴兴博, 等. SFPS: 低延迟的面向小文件的分布式文件系统[J]. 计算机科学与探索, 2014, 8(4): 438-445.
- [5] 童 明. 基于 HDFS 的分布式存储研究与应用[D]. 武汉: 华中科技大学, 2012.
- [6] 王永洲. 基于 HDFS 存储技术的研究[D]. 南京: 南京邮电大学, 2013.
- [7] 曹 卉. Hadoop 分布式文件系统原理[J]. 软件导刊, 2016, 15(3): 15-17.
- [8] LIAO Wenzhe. Application of Hadoop in the document storage management system for telecommunication enterprise [J]. International Journal of Interdisciplinary Telecommunications and Networking, 2016, 8(2): 58-68.
- [9] BENDE S, SHEDGE R. Dealing with small files problem in Hadoop distributed file system [J]. Procedia Computer Science, 2016, 79: 1011-1012.
- [10] 邵秀丽, 王亚光, 李云龙, 等. Hadoop 副本放置策略[J]. 智能系统学报, 2013, 8(6): 489-496.
- [11] BUYYA R, YEO C S, VENUGOPAL S, et al. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility [J]. Future Generation Computer Systems, 2009, 25(6): 599-616.
- [12] 段效琛, 李英娜, 贾会玲, 等. 初始信息素筛选的蚁群优化算法在 HDFS 副本选择中的研究[J]. 传感器与微系统, 2017, 45(4): 31-33.
- [13] 王 来, 翟健宏. 基于 HDFS 的分布式存储策略分析[J]. 智能计算机与应用, 2016, 6(1): 5-8.
- [14] 李晓恺, 代 翔, 李文杰, 等. 基于纠错码和动态副本策略的 HDFS 改进系统[J]. 计算机应用, 2012, 32(8): 2150-2153.