HDFS 中的一种数据放置策略

王永洲[1,2,3],茅 苏[1,2,3]

(1. 南京邮电大学 计算机学院,江苏 南京 210003;
2. 江苏省无线传感网高技术研究重点实验室,江苏 南京 210003;
3. 宽带无线通信与传感网技术教育部重点实验室,江苏 南京 210003)

摘 要:将 HDFS 应用于云存储备份系统已引起学术界和企业界的广泛关注,但是 HDFS 假设集群中的节点是同构的,且在数据块的放置时采用了随机的数据放置策略,而在实际的云存储系统中节点的同构性并不理想并且随系统的运行节点的存储性能动态变化,随机选取节点的数据放置策略也可能会导致集群中数据放置的不均衡性。为此,文中综合考察了影响节点存储性能的 CPU、内存、系统结构、磁盘的读写速度等因素,提出了一种对节点存储性能进行度量的方法。在数据存储时综合考虑节点的存储性能和节点与客户端的网络距离两个因素进行数据节点的选择。实验结果表明,与系统默认策略相比提出的策略不仅可以改进数据放置的负载均衡而且可以提高异构集群的数据传输速率。

关键词:HDFS;云存储;数据放置;吞吐量
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2013)05-0090-03
doi:10.3969/j.issn.1673-629X.2013.05.023

A Blocks Placement Strategy in HDFS

WANG Yong-zhou[1,2,3], MAO Su[1,2,3]

(1. College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2. Jiangsu High Technology Research Key Lab for Wireless Sensor Networks, Nanjing 210003, China;
3. Key Lab of Broadband Wireless Communication and Sensor Network Technology of Ministry of Education, Nanjing 210003, China)

Abstract: The application of HDFS in cloud storage has caused wide attention in academic circles and corporate community, but HDFS assumes that the nodes in a cluster are homogeneous and the data node selection for block placement in cluster is random. The current strategy ignores the heterogeneity of cluster and could lead to unbalanced load between data nodes. In this paper, put forward a method for measuring the storage performance of the data node which takes CPU, internal storage, system structure and disk speed for account. The new strategy places data according to the storage performance and the network distance between client and data node. The experimental results indicate that the strategy can not only behave better in load balancing but also improve the data transmission speed.

Key words: HDFS; cloud storage; data placement; throughput

0 引 言

云存储作为云计算[1~4]的基础架构和关键技术之一,不但对云计算技术的发展起着至关重要的作用,而且作为一种独立的存储业务也已经在很多企业得到了广泛的研究和应用。但是云存储作为一项较前沿的发展技术距离大规模的商用还有一段时间,目前能被大多数用户接受和使用的云存储是较为简单的云备份业务[5,6]。研究表明,云存储中的任务通常具有数据量大、计算量小和数据传输速率敏感的特点,即云存储系统对吞吐量有着较高的要求。HDFS(Hadoop Distributed File System)作为开源的 Hadoop[7,8]项目的核心组件之一,负责系统的数据存储和管理、文件管理及出错处理等基础工作,是谷歌分布式文件系统 GFS[9]的开源实现,也是文中的研究依托。

在目前的 HDFS[10]版本中都假设集群中的节点是同构的,且在数据块的放置时采用了随机的数据放置策略[11],而在实际的云存储系统中节点的同构性并不理想并且随着系统的运行节点的存储性能动态地变化,随机选取节点的数据放置策略也可能导致集群中数据放置的不均衡性。文献[12]提出了一种基于节

收稿日期:2012-07-27;**修回日期**:2012-10-30
基金项目:江苏高校优势学科建设工程项目(yx002001)
作者简介:王永洲(1985-),男,河北晋州人,硕士研究生,研究方向为基于网络的计算机软件应用技术;茅 苏,高级工程师,研究方向为基于网络的计算机软件应用技术。

点评价值的选取策略,考虑到了随机策略可能引起的节点负载不均衡问题,但是忽略了集群的异构性问题。异构集群中不同的节点有着不同的存储性能。对于存储性能不同的两个节点放置同样多的数据不利于高性能节点上存储资源的利用,有理由在存储性能高的节点放置更多的数据。因此,文中提出一种根据节点的存储性能按比例存放数据的策略以提高整个系统的存储性能和平衡节点存储资源的利用率。

1　HDFS 的数据放置策略

1.1　默认的数据放置策略

HDFS 采用主从式的结构管理体系。在一个 HDFS 集群中有一个名称节点和多个数据节点。名称节点负责数据存放节点的选择和命名空间的管理,数据节点负责数据块的存储并受名称节点的管理。当有数据块上传请求时名称节点根据一定的选取策略筛选出最优的存储节点,然后由客户端将数据上传到指定的节点中。HDFS 默认为每一个数据块存放 3 个副本,其中两个副本按照随机选择的策略存放在本地机架中的两个节点上,第三个副本放在一个随机选择的其他机架中的一个随机节点上。

1.2　默认策略的局限性

HDFS 采用了简化模型的设计理念,数据放置时依靠随机选择的数据放置策略,在数据量较大时以概率来保证数据较均衡地分布到各个节点上,但是云计算复杂并且可能动态变化,依靠概率来保证数据放置的均衡性本身就不可靠。再者,HDFS 假设集群中的节点是同构的,而在实际的集群中节点存储性能的同构性并不理想,部分节点之间还可能存在较大的性能差异,因此,给各个数据节点平均分配数据量的做法一方面并不能充分发挥部分高性能节点的作用,另一方面还可能因为部分低性能节点负载过重给用户带来较差的用户体验。

2　改进的数据放置策略

2.1　相关概念的定义

(1)节点的存储性能:节点的存储性能包含的范围较广,主要包括节点 CPU 性能、内存大小、存储设备的存取速度、系统结构;另外还包括节点的可靠性评价,节点对存储数据的安全性、完整性的保障程度等。节点的可靠性和数据的安全性不属于文中的讨论范围。假设集群中所有的节点都是可靠的,数据节点上所有的数据都是安全完整的。

节点的 CPU 性能:其影响因素有处理器主频、每条指令所花的时钟周期数(即 CPI)、操作的指令条数,用参数 C 表示。

磁盘读写速度:用参数 V 表示。

$$V = \lambda V_{\text{Write}} + (1 - \lambda) V_{\text{Read}} \tag{1}$$

V_{Write} 为磁盘的写速度, V_{Read} 为磁盘的读速度, λ 称为更新频率,为写操作相对于其它所有操作的百分比。

内存大小:用参数 M 表示。

系统结构:可在很大程度上影响节点吞吐量,如并行处理结构可增大吞吐量,用参数 S 表示, S 的取值范围为[0,1]。

综上所述,集群中某一节点 i 的存储性能:

$$P(i) = A_1 * C(i) + A_2 * V(i) + A_3 * M(i) + A_4 * S(i) \tag{2}$$

其中 A_1、A_2、A_3、A_4 为各参量的权重因子且 $A_1 + A_2 + A_3 + A_4 = 1$。

(2)相对节点存储性能。

$$P_{\text{Re}}(i) = \frac{P(i)}{P(i)_{\min}} \tag{3}$$

$P(i)$ 为节点 i 的存储性能, $P(i)$ min 为集群中所有节点存储性能的最小值, $P_{\text{Re}}(i)$ 即为定义的相对节点性能。

(3)相对负载。

在异构的集群中用数据节点上放置的数据量(绝对负载)来衡量节点的负载状况并不准确,在此定义异构集群中节点相对负载的概念。

$$\text{LOAD}(i) = N(i)/P_{\text{Re}}(i) \tag{4}$$

$N(i)$ 为节点 i 上已存放的数据块数。

前面已经提到,对于每一个数据节点,改进的策略根据节点性能按比例存放数据,在此将节点负载作为节点的选择代价,在选择数据块的放置节点时优先选择“节点选择代价”值最低的节点进行数据的放置。

$$\text{COST}(i) = \text{LOAD}(i) \tag{5}$$

(4)节点平均负载。

为了统计整个集群中的负载状态,定义平均负载的概念:

$$\text{LOAD\_AVERAGE}(i) = \sum \text{LOAD}(i)/\text{SUM} \tag{6}$$

SUM 为集群中的节点数。

(5)节点的平衡条件判定。

HDFS 中实现了一个 Balancer 程序来调节系统中各节点的负载均衡,在异构的集群中判定一个节点是否均衡的条件需要重新定义:

当一个节点中

$$\text{LOAD}(i) - \text{LOAD\_AVERAGE}(i) > \text{L\_CONSTANT} \tag{7}$$

时,称此节点负载过重;

当一个节点中

$$\text{LOAD\_AVERAGE}(i) - \text{LOAD}(i) > \text{L\_CONSTANT} \tag{8}$$

时,称此节点负载过轻。

L_CONSTANT 为常量,称为负载阈值,可根据系统实际情况进行调整。

2.2 改进的数据放置策略的算法描述

按照系统的默认策略,对于同一个数据块应该至少放置在两个不同的机架上以保证数据的安全。其中在第一个机架上选择两个不同的节点,第二个机架上选择一个节点,来进行数据块的放置。

基于上面的基本思想,下面给出改进策略的算法描述当有数据块需要上传时:

(1)计算网络拓扑中与客户端节点网络距离最近的机架,并返回机架 ID。已标记为选择的机架不在选择之列。如果没有可用机架则返回 NULL。

(2)如果返回值为空,即已没有可用机架,程序结束,否则继续下一步。

(3)如果机架 1 已选,跳转至第 5 步,进行机架 2 的选取,否则继续下一步,进入机架 1 的选取。

(4)根据公式(5)计算返回机架中节点选择代价最低的两个节点。根据公式(7)判断将数据块放到这两个节点上之后系统的负载平衡是否被打破? 如果节点负载仍然平衡,那么将这两个节点确定为数据块的放置节点,将此机架确定为机架 1,并将此机架标记为已选,否则跳转至步骤(1)重新进行机架的选取。

(5)根据公式(5)计算返回机架中节点选择代价最低的节点。根据公式(7)判断将数据块放到这个节点上之后系统的负载平衡是否被打破? 如果节点负载仍然平衡,那么将这个节点确定为数据块的放置节点,将此机架确定为机架 2,并清楚机架选择标记,否则跳转至步骤(1)重新进行机架的选取。

(6)如果所有数据块均已放置完毕,程序调用结束,否则,进行下一个数据块的放置。

改进策略的数据块放置流程图如图 1 所示:

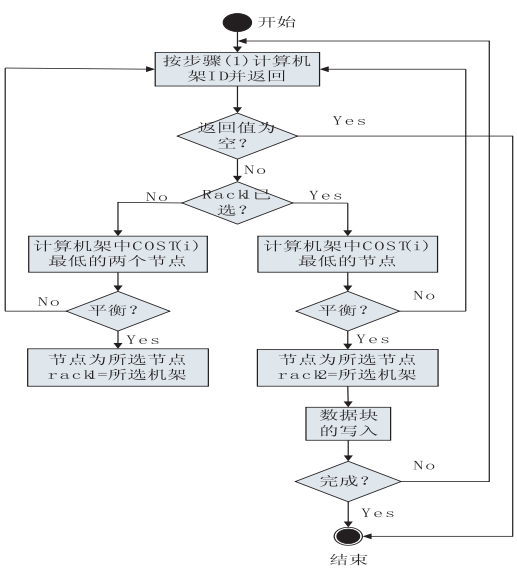


图 1 数据块放置流程图

3 实验结果与分析

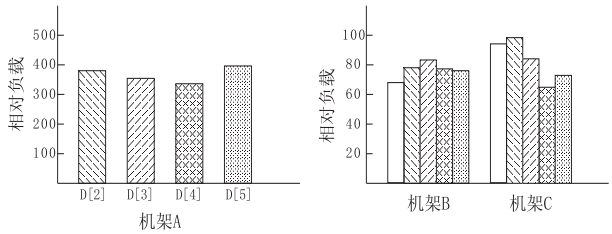
文中使用 Hadoop-1.0.0 进行了实验环境的搭建并对所提出的策略进行了实现。实验集群中有 3 个机架(机架 A、机架 B 与机架 C),每个机架有 5 个节点,由于集群中各节点存储性能不同,根据公式(3)计算出各节点的相对存储性能如表 1 所示。

表 1 集群中各节点的相对存储性能

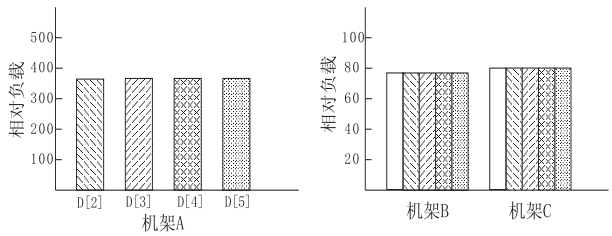
	D[0]	D[1]	D[2]	D[3]	D[4]
机架 A	Client	1.26	1.46	1.45	1.31
机架 B	1.41	1.36	1.17	1.27	1.29
机架 C	1.13	1	1.23	1.48	1.33

实验中,提交数据的客户端程序是机架 A 中的节点 D[0](此节点不作为集群中的数据节点),共有 1000 个数据块从这个节点上传到集群。假设系统启动时所有的 DataNode 都处于空载状态。

由于异构集群中节点的性能各不相同,采用节点上的数据块数来衡量各节点的负载状况并不准确,在此利用公式(4)采用相对负载的概念对各数据节点的负载情况进行重新统计,如图 2 所示。



(a)采用默认策略时集群中各节点的相对负载情况



(b)采用改进策略时集群中各节点的相对负载情况

图 2 不同策略下集群中各节点的相对负载情况

对比图 2 可知,使用相对负载的概念后,改进的数据放置策略能保证机架中各个数据节点间的负载均衡。为了验证改进策略对集群整体性能的提升,对 1000 个数据块的上传时间进行了统计如图 3 所示。

由图 3 可知,使用改进的数据放置策略可缩短数据块的上传时间,使客户端的数据传输速率提升了 7.7%。

4 结束语

文中讨论了 HDFS 应用于云存储系统的相关问题,分析了 HDFS 系统默认数据放置策略的局限性,提

(下转第 96 页)

均吞吐量最大。因此从平均队列长度、平均链路时延或平均吞吐量这三方面来讲,ISNAPID 算法的性能都优于 PI、SNAPID 算法。

3 结束语

文中针对单神经元自适应 PID 主动队列管理算法中权值学习修正不够精确等问题,提出了改进的单神经元自适应 PID 算法—ISNAPID 算法,该算法对学习修正部分进行了相应的调整,并引入了一个附加动量避免权值在学习过程中出现振荡,加快了系统的收敛速率。改进的算法的性能分析表明其性能优于 PI、SNAPID 算法。

参考文献:

[1] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance[J]. IEEE/ACM Transactions on Networking,1993,1(4):397-413.

[2] 汪华斌,刘卫国. 一种快速收敛的 RED 改进算法[J]. 计算机系统应用,2008,17(7):62-71.

[3] 吴清亮,陶 军,姚 婕. 一种基于预测 PI 控制器的自相似网络主动队列管理算法[J]. 电子学报,2006,34(5):938-942.

[4] Ren Fengyuan, Lin Chuang. Speed up the responsiveness of active queue management system[J]. IEICE Transactions on Communication,2003,86(2):630-636.

[5] 章 淼,吴建平,林 闯. 一种新的主动队列管理算法[J]. 计算机学报,2003,26(10):1288-1294.

[6] Rouhani M, Tanhatalab M R, Shokahi-Rostami A. Nonlinear Neural Network Congestion Control Based on Genetic Algorithm for TCP/IP Networks[C]//2010 Second International Conference on Computer Intelligence, Communication Systems and Networks. Liverpool:[s. n.],2010:1-6.

[7] 候 萍,王执铨. 基于 PID 神经元网络和内模控制的拥塞控制算法[J]. 计算机应用研究,2009,26(4):1443-1445.

[8] Sun J, Chen J, Wang Z. Adaptive neuron PID: a new active queue management algorithm[C]//Proceedings of the 6th World Congress on Intelligent Control and Automation. Dalian:[s. n.],2006:6308-6312.

[9] 李春来,童耀南. 单神经元自适应 PID 控制器在主动队列管理中的应用[J]. 湖南理工学院学报(自然科学版),2009,22(3):41-43.

[10] 任丰源,王福豹,任 勇,等. 主动队列管理中的 PID 控制器[J]. 电子与信息学报,2003,25(1):94-99.

[11] 张少博,吴介一,张飒兵. 基于独立神经元的自适应主动队列管理算法[J]. 数据采集与处理,2008,23(6):707-712.

[12] 张世韬,杨 风,郝 骞. 单神经元 PID 控制器研究及仿真[J]. 机械工程与自动化,2009,39(3):69-70.

(上接第 92 页)

出了一种按照节点存储性能按比例存放数据的策略,从仿真实验可以看到,该策略可以平衡异构集群中的负载均衡,提高系统平均数据传输速率。

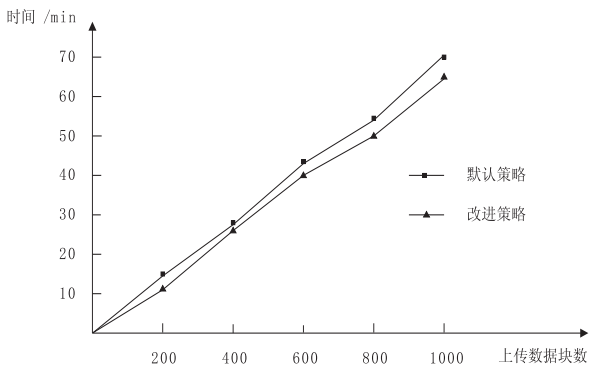


图 3 两种策略的数据块上传时间对比

参考文献:

[1] 王庆波,金 萍,何 乐,等. 虚拟化与云计算[M]. 北京:电子工业出版社,2009:110-180.

[2] 刘 鹏. 云计算[M]. 第 2 版. 北京:电子工业出版社,2011:1-15.

[3] 陈 全,邓倩妮. 云计算及其关键技术[J]. 计算机应用,2009(9):2562-2566.

[4] Armbrust M, x A, Griffith R, et al. Above the Clouds: A Berkeley View of Cloud Computing[J]. Communications of the ACM,2010,53(4):50-58.

[5] 陈 康,郑纬民. 云计算:系统实例与研究现状[J]. 软件学报,2009,20(5):1337-1348.

[6] TT 中国. 云存储:“云”数据备份先行[EB/OL]. [2011-06-15]. http://www.cloudcomputing-chain.cn/Article/luilan/.

[7] 林清滢. 基于 Hadoop 的云计算模型[J]. 现代计算机,2010,7(7):114-116.

[8] 栾亚建,黄骅民,龚高晟,等. Hadoop 平台的性能优化研究[J]. 计算机工程,2010,36(14):262-264.

[9] Ghemawat S, Gogioff H, Leung P T. The google file system[C]//Proc of the 19th ACM Symp on Operating Systems Priciples. New York:ACM,2003:29-43.

[10] Borthakur D. The hadoop distributed file system: architecture and design[EB/OL]. [2011-06-15]. http://hadoop.apache.org/hdfs/docs/current/hdfs_design.html.

[11] Borthakur D. Hadoop[EB/OL]. [2011-06-15]. http://lucene.apache.org/hadoop.

[12] 林伟伟. 一种改进的 Hadoop 数据放置策略[J]. 华南理工大学学报:自然科学版,2012,36(1):152-158.