

HBase 负载均衡分析及优化策略

黄伟建, 宋园园

(河北工程大学 信息与电气工程学院, 河北 邯郸 056038)

摘 要: HBase 作为 Hadoop 分布式开源云数据库因其高可用性等优势越来越受到青睐, 但是随着大量数据的注入, HBase 对负载的分配状况将直接影响到整个集群的性能优劣. 针对原有负载均衡算法在负载分配过程中可能产生的负载严重不均衡问题, 通过分析原有算法和问题出现的因素, 提出一种基于子表限制的负载均衡改进方法, 并通过与不均衡状况下的对比实验, 验证改进后的分配方式可以有效利用集群中各个节点的资源, 从而提高分布式集群性能.

关键词: 云计算; HBase; 负载均衡; 节点资源; 集群性能

中图分类号: TP302

文献标识码: A

文章编号: 1000—7180(2016)04—0125—04

HBase Load Balancing Algorithm Analysis and Improvement

HUANG Wei-jian, SONG Yuan-yuan

(School of Information and Electronic Engineering, Hebei University of Engineering, Handan 056038, China)

Abstract: As the Hadoop distributed open source database, HBase gradually gained a lot of attention due to its high-performance. However, with a large amount of data pouring into HBase, the situation of load distribution by HBase will directly affect the performance of the whole cluster. With the problem of load seriously imbalances in the process of original algorithm, proposed a novel load-balancing approach based on the limited child table, with the analysis of original algorithm and the causes of the problem. The contrast experiment showed that the improved approach of scheduling can make effective use of each node resources in the cluster, thereby improved the performance of the distributed cluster.

Key words: cloud computing; HBase; load balancing; node resources; cluster performance

1 引言

云计算技术的产生和发展为海量数据处理提供新的服务方式, 其中, 平台即服务层的解决方案 Hadoop 可以为大量数据提供可靠、高效的分布式计算和存储, 现已发展成包括并行编程模式 MapReduce、分布式文件系统 HDFS 和分布式开源数据库 HBase^[1-4]等在内的多个子项目的集合. 同时, 负载均衡作为系统性能的影响指标逐渐在各相关领域受到重视, 目前 HBase 负载均衡算法相关的研究和成果主要集中在某些大公司, 且很少具有开源信息. 本文在原有负载均衡算法的基础上, 通过添加表限制

条件, 解决负载分布的极端状况.

2 HBase 负载均衡算法

在 HBase 中, 当集群重新启动时, HMaster 首先会使用轮询的方式进行负载分配, 之后把负载均衡问题交给负载均衡器去处理. 文献[5]记录了 HBase 负载均衡算法的基本过程, 具体流程如图 1 所示.

算法的基本思想是利用 HRegionServer 的临界值限制条件转移相应负载, 使所有的 HRegionServer 的负载量均在限制范围内.

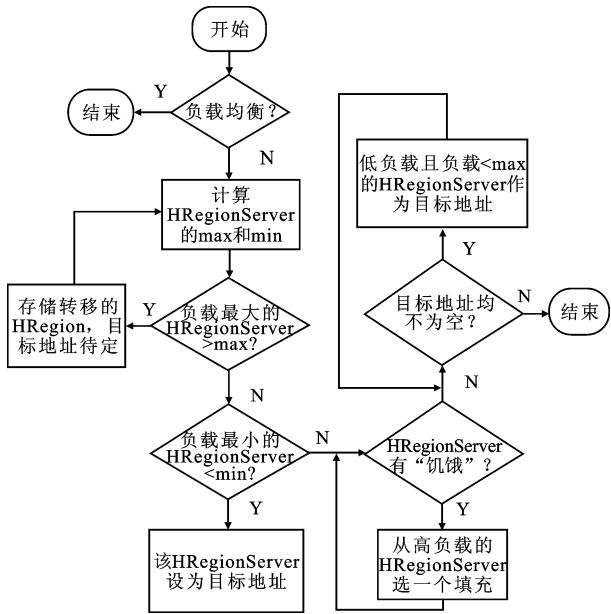


图 1 HBase 负载均衡算法流程

3 算法优化

3.1 问题描述

若 HRegionServer 上负载分布不均衡,在运行中容易引起部分资源节点的堵塞,因此负载均衡策略对 HBase 集群具有很重要的意义. HBase 可以通过多种措施进行性能调优^[6],现只考虑负载调度对其产生的影响.

通过 HBase 负载均衡算法的流程分析看出,该算法试图将所有 HRegion 尽量均等的分发到每个 HRegionServer 上,但这种表面上的均等并不能保证运行过程中资源访问的均衡. 在实际中,这个算法可能会出现负载极端情况,如图 2 所示.

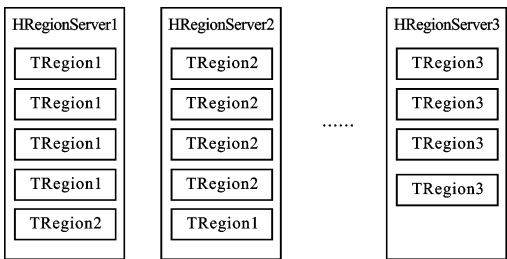


图 2 HBase 负载均衡过程中可能会出现的情况

图 2 中 TRegion1 表示表 Table1 的各分割子表 HRegion. 可以看出, Table1 的子表几乎都被分配到 HRegionServer1 上,如果实际负载请求中, Table1 具有极高的访问频率,就会给 HRegionServer1 极大的访问负荷,成为整个集群的瓶颈^[7-8],从而降低整个集群资源利用率,进一步引起性能的急剧下

降,无法有效发挥分布式系统的优势.

3.2 算法策略及相关定义

在原算法试图将所有的 HRegion 均衡分配到各域服务器的基础上,使各表的分割子表 HRegion 在集群上也均衡分布,算法相关定义如下所示.

定义 1 创建 map 容器 RS 和 T,将 HRegionServer、表的临界值和负载个数分别存储在以服务器名、表名为 key(键)的 Map RS、T 中. MAX_s 和 MIN_s 表示 HRegionServer 承载子表 HRegion 的临界值; MAX_t 和 MIN_t 表示子表 HRegion 分配到 HRegionServer 的临界值.

定义 2 numRegions 统计负载 HRegion 的总个数, tableRegions 统计表分割出的子表 HRegion 的个数, numServers 表示域服务器 HRegionServer 的个数.

定义 3 计算各个临界值的公式. 首先计算 HRegionServer 和表的平均值, $average_s = \text{numRegions} / \text{numServers}$, $average_t = \text{tableRegions} / \text{numServers}$; 各临界值即可以表示为: $MAX_s = \text{ceil}(average_s)$, $MIN_s = \text{floor}(average_s)$; $MAX_t = \text{ceil}(average_t)$, $MIN_t = \text{floor}(average_t)$.

3.3 基于子表限制的负载均衡改进算法

利用域服务器和表临界值所做的改进算法如下所示.

(1)遍历集群,统计每个表和域服务器的负载信息,并计算临界值,包括域服务器的临界值 MAX_s 、 MIN_s 和表的临界值 MAX_t 、 MIN_t ,记录到 RS 和 T 中.

(2)检查集群是否处于负载均衡状态,即负载的最小值大于 MIN_s 或 MIN_t ,并且最大值小于 MAX_s 或 MAX_t ,若是,则无需进行均衡操作.

(3)取出每个 HRegionServer 上超过负载临界值的 HRegion,具体步骤:

- ① 取出超过表最大临界值 MAX_t 的 HRegion;
- ② 检查是否仍然超过 HRegionServer 的最大承受负载,如果是,则取满足表最小临界值 MIN_t 的 HRegion,直到 HRegionServer 不再超载;
- ③ 将取出的 HRegion 存储到以子表名为 key 的临时 Map RP 中,其结构表示为 $\text{map} < \text{key}, [(\text{HRegion}, \text{resource}, \text{destination})] >$ (其中 destination 先置 null);

(4)遍历 RS,尝试为每个 RP 添加目标地址,比较 HRegionServer 中每个表的负载和 MIN_t ,如果负载小于 MIN_t ,则可将属于该表的 RP 目标地址设为该 HRegionServer.

(5)检查集群负载状况,若 RP 都分配了目的地

址且 RS 中不存在“饥饿”即负载均大于 MAX_i , 则程序返回。

(6)若 RS 中有存在“饥饿”的情况,则遍历所有的 HRegionServer;对于 HRegionServer 上的每个表判断是否“饥饿”,若是,则遍历其他域服务器,然后将满足该表最小限制(即大于 MIN_i)的 HRegion 分发到该 HRegionServer 上。

(7)若存在没有分配目标地址的 RP,则再次进行分配:

① 遍历 RS,对于每个 HRegionServer,统计它当前所有 HRegion 的个数 numRegions 和表分割子表的个数 tableRegions;

② $\min(MAX_i - \text{numRegions}, MAX_i - \text{tableRegions})$ 个 RP 的目标地址设为该 HRegionServer;

(8)返回 RP 即为优化负载均衡策略。

4 实验测试

在 HBase 集群上对改进算法进行实验验证,使用 4 台 Linux 主机搭建环境^[9],各主机角色分配如表 1 所示。由于实验环境的版本存在兼容问题,故本实验采用版本号为:Ubuntu12.04.3,jdk1.7.0,Hadoop2.2.0,Hbase0.98。

表 1 实验环境设置情况		
主机名	IP 地址	作用
Hbase	192.168.3.64	HMaster
Hbase ₁	192.168.3.65	HRegionServer
Hbase ₂	192.168.3.66	HRegionServer
Hbase ₃	192.168.3.67	HRegionServer

4.1 基于子表限制的 HBase 负载均衡算法的有效性测试

为了更好地表现出负载的分配情况,实验以集群节点的网络流量作为对比指标,使用伯克利大学开发的分布式系统监控工具 Ganglia,直观的观察每个节点的网络流量走势。

同时,为了更明显地表现新算法的效果,实验开发一个不平衡的负载分配器,开发过程参考文献[10]所示,它的目的是模拟子表分配极端不平衡情况,具体的操作是使集群在重启后对 HRegion 重新分配,分配的结果是将所有属于同一个表的 HRegion 分配到同一个 HRegionServer 上。

实验中分别对这两种情况进行测试。在 HBase 上生成索引表,重启集群,运行极端负载策略,在 Ganglia 客户端上对其中一个表进行表扫描操作,监测扫描过程中集群及各节点的网络流量,收集的结果如图 3 所示。

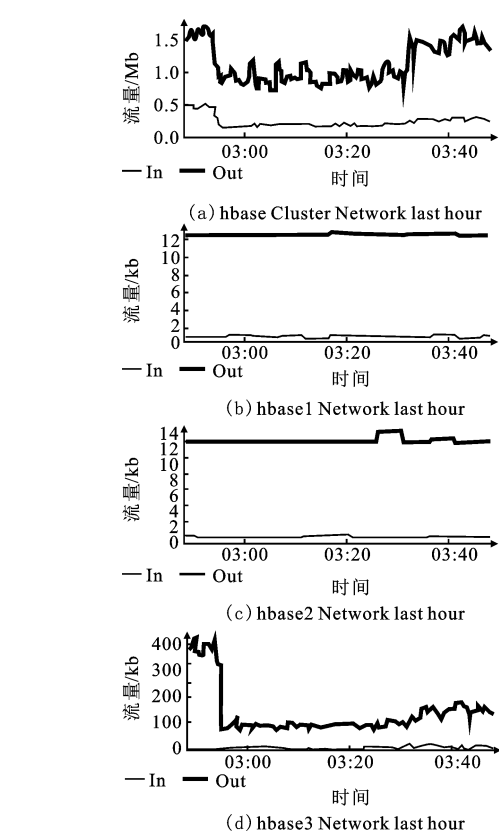


图 3 不平衡负载策略下各个节点的网络情况

重启集群,加载基于子表限制的负载均衡策略,Ganglia 客户端上对集群和各个节点的网络流量监控情况如图 4 所示。

对比两个实验可以看出,此时的网络上都具有较大访问流量,但两种策略下集群中各节点的网络状态有很大差异,图 3 所示情况下,对于表的访问情况,网络负载全部集中在 hbase₃ 节点上,而其他两个节点的网络则只有很小的流量。实验中只是一个客户端进行表的访问操作,当在实际中对该表的请求急剧增多时,hbase₃ 节点就会成为整个集群的瓶颈,必然导致集群性能的下滑。而在图 4 所示情况下,子表 HRegion 是均匀分配在 HRegionServer 中的,这就使各客户端对表的遍历操作可以分发到不同的网络节点上,当实际中对表的访问量增加时,也会相对均衡的增加每个节点的访问流量,避免出现资源拥堵现象。因此可以看出,本文的优化算法可以充分利用集群中各节点资源,有效避免访问“热点”,从而提高分布式集群的性能。

4.2 优化后算法的高效性分析

在 4.1 节的实验中,通过对比实验验证了优化算法在负载均衡上的有效性,而该算法是在系统原算法根据 HRegionServer 临界值均衡负载的基础上,添加了表的临界值限制条件,并不涉及复杂改

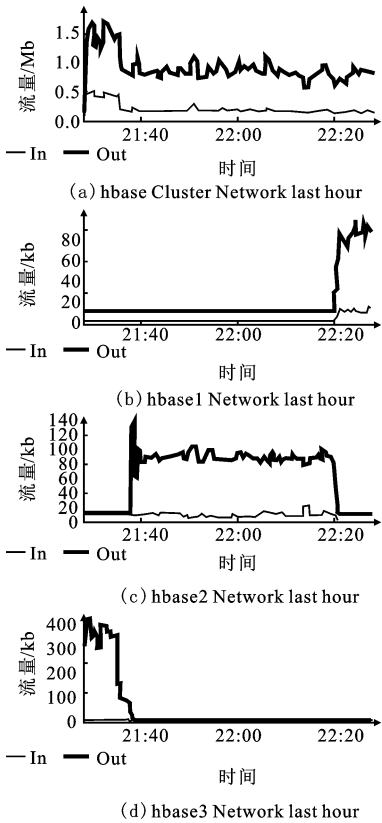


图 4 优化负载策略下各个节点的网络情况

动,保证了算法的稳定性和可靠性.

在 HBase 中,负载优化算法是在加载系统原有算法之后才执行的,在节点规模适度的情况下,对系统的时间影响并不大,而一旦出现访问“热点”,它会极大地降低系统资源利用率和集群性能,所以优化后的算法在综合考虑下是可行的.但是随着集群节点数或者表数目的增多,算法的开销和损耗就会越大,如图 5 所示,导致时间消耗增加,这样降低了改进算法所取得的性能优势,因此本算法适合规模适中的私有云集群环境.

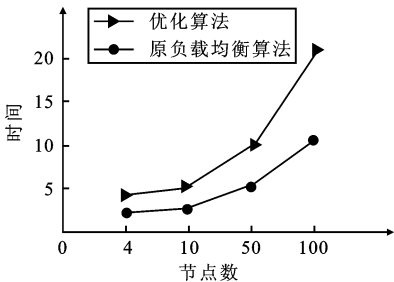


图 5 加载原算法和优化算法的时间对比图

5 结束语

HBase 的负载分配涉及到集群资源的利用率和

性能问题,还会影响分布式系统的并行性,如何有效地分配负载也就具有重要意义.本文中基于子表限制的负载均衡算法有利于解决负载分配的极端问题,但只是其中的一种改进方法,当 HBase 的表增多时就易导致大规模的 HRegion 迁移,同样会造成系统性能的下降.因此,本算法对集群规模有一定要求,下一步的研究可以考虑增加阈值控制或者根据集群的吞吐率、内存等性能指标做参数来进行负载的均衡策略.

参考文献:

[1] The apache HBase reference guide. The Apache HBase Book[EB/OL]. [2015-04-20]. <http://hbase.apache.org/book/book.html>.
[2] 王荣生,杨际祥,王凡. 负载均衡策略研究综述[J]. 小型微型计算机系统,2010,31(8):1681-1686.
[3] 丛珉,陆民燕,白云峰. 软件可靠性预计方法研究及实现[J]. 北京航空航天大学学报,2002,28(1): 34-38.
[4] 刘琨,肖琳,赵海燕. Hadoop 中云数据负载均衡算法的研究及优化[J]. 微电子学与计算机, 2012,29(9):18-21.
[5] Mk78x. HBase Load Balancing Algorithm[EB/OL]. [2015-04-20]. <http://www.linuxidc.com/Linux/2013-07/87233.htm>.
[6] DataScientist. HBase Optimization Case Study[EB/OL]. [2015-04-20]. <http://yanbohappysinaapp.com/?tag=hbase>.
[7] Oredope A, Moessner K, Peoples C, et al. Deploying cloud services in mobile networks[C]//Science and Information Conference (SAI), London: IEEE, 2013: 928-933.
[8] Zhang Changlun, Wang Kaixuan, Mu Haibing. A Priority Queue Algorithm for the Replication Task in HBase [J]. Journal of Software, 2013, 8 (7): 1765-1769.
[9] lhcx1989. Hadoop and Hbase Cluster Setup[EB/OL]. [2015-04-20]. <http://blog.csdn.net/lhcx1989/article/details/8140265>.
[10] Search Technology Blog. How to Develop LoadBalancing Plugins with HBase[EB/OL]. [2015-04-20]. http://www.searchtb.com/2012/05/hbase_loadbalance_plugins.html.

作者简介:

黄伟建 男,(1964-),博士.研究方向为云计算、并行计算、信息系统.

宋园园(通讯作者) 女,(1989-),硕士研究生.研究方向为云计算. E-mail:ysong90@163.com.