

基于分片一致性哈希负载均衡策略与应用

苏跃明^{1,2}, 李 晨², 田丽华²

(1. 西安交通大学 软件学院, 陕西 西安 710000;
2. 百度(中国)有限公司, 北京 100000)

摘 要:采用一致性哈希进行数据分区和负载均衡的分布式键值存储系统具有高可扩展性的特点,但一致性哈希中哈希函数静态负载均衡的特性不能满足日益多样化的应用场景需求。为了适应以上需求,从一致性哈希策略出发,结合动态负载均衡技术,设计了一种基于一致性哈希的动态负载均衡策略。该策略使用与物理节点解耦的分片代替传统的虚拟节点,并利用针对分片的监控信息,从分片级和节点级两个层面对系统负载进行均衡调度,通过更细的调度粒度优化均衡效果。实验结果表明,该策略保留了一致性哈希策略在系统扩展性上的优势,同时优化了一致性哈希策略负载均衡的总体效果。利用基于分片的一致性哈希负载均衡策略,可以有效地均衡系统负载,提高存储系统的效率。

关键词:一致性哈希;分片;动态负载均衡;分布式键值存储

中图分类号:TP39

文献标识码:A

文章编号:1673-629X(2017)11-0062-04

doi:10.3969/j.issn.1673-629X.2017.11.013

A Consistent Hashing Load Balancing Strategy Based on Fragmentation and Its Application

SU Yue-ming^{1,2}, LI Chen², TIAN Li-hua²

(1. School of Software Engineering, Xi'an Jiaotong University, Xi'an 710000, China;
2. Baidu Corporation, Beijing 100000, China)

Abstract: The distributed key-value storage system, which uses consistent hashing for data partitioning and load balancing, has high expansibility. However, the static load balancing strategy in consistent hashing cannot meet the increasingly diverse needs of application. In order to adapt to above needs, a dynamic load balancing strategy is designed based on the consistent hashing and combined with dynamic load balancing. It adopts the fragment decoupled physical nodes instead of traditional virtual nodes and uses the monitoring information of fragments to make decisions for load balancing scheduling from two aspects of fragment level and node level. Experimental results show that it has retained the advantage of consistent hashing strategy in system scalability, while optimizing the overall performance of consistent hashing load balancing. The system load can be effectively balanced and the utilization of the system can be improved.

Key words: consistent hashing; fragment; dynamic load balancing; distributed key-value storage

1 概 述

为了满足社交和移动互联网时代,海量读写请求和数据爆发式的增长需求,分布式存储系统被广泛应用。为了快速定位数据,同时提高集群的可扩展性,分布式存储系统多采取一致性哈希算法对数据进行分区,但一致性哈希依赖哈希函数均衡系统负载,属于静态均衡策略^[1]。

随着键值存储应用领域发展的多样化,采用静态均衡的存储系统给企业造成了极大的资源浪费。如何

结合分布式存储系统的数据分区特点对其进行动态负载均衡成为迫切需要解决的问题。

负载均衡策略主要分为动态负载均衡策略和静态负载均衡策略。静态负载均衡根据对系统负载状态的先验分析^[2],预先制定系统负载策略,实现简单;而动态负载均衡通过分析当前的系统负载情况,动态地分配和调度任务,适应场景多,均衡效果好。一般分布式系统中,常用的负载均衡算法有随机算法、轮询、权重轮询、最小连接数和动态反馈^[3-5],在利用这些策略进

行负载均衡时,多要求分布式节点之间具有一定的可替代性,即一个任务可由分布式系统中任意的节点进行处理^[6]。然而,对于分布式存储系统,单个节点并不能存储全量的数据,数据需要进行分区块存储,导致存储系统的读写任务只可以由特定节点处理。因此,分布式存储系统的负载均衡与系统的数据分区方式紧密相关^[7]。

目前主流的分区方式有范围分区、列表分区和哈希分区三种方式。分布式键值存储系统只有一个列值,无法基于列表分区;范围分区是按照键的范围来进行数据分区,典型的有 MongoDB 的 Sharding 分区方式。文献[8]通过对 MongoDB 的负载均衡策略进行改进,优化了 MongoDB 的应用场景;而在分布式键值存储系统中,应用最广泛的是哈希分区方式。相比于范围分区,哈希分区具有天然的静态负载均衡特性,通过选择合适的哈希函数,就可以实现数据在哈希空间内的均衡分布。

常用的哈希函数有 MD5 和 SHA。文献[9]给出的一致性哈希解决了哈希分区在集群扩缩容时的数据重分布问题,在此基础上,Amazon 的 Dynamo 在实现一致性哈希策略时提出了虚拟节点的理论来解决集群内节点之间异构的问题^[10]。

然而,这些方法都属于静态一致性哈希负载均衡策略,它们仅依赖于哈希函数进行均衡负载,面对集群中部分数据的热点访问带来的负载不均,静态负载均衡算法将无法调节^[11]。

通过分析基于虚拟节点的一致性哈希策略,结合动态反馈的负载均衡技术,提出了一种基于分片的一致性哈希动态负载均衡策略。使用与物理节点解耦的分片代替传统的虚拟节点,并利用针对分片的监控信息对负载进行均衡调度。同时,调度策略分为分片级和节点级两个层面,从而进一步优化负载均衡的效果。相比于传统基于节点负载的动态反馈均衡策略,该策略具有更细的监控和调度粒度,从而实现了更高效的均衡调度。

2 一致性哈希策略

一致性哈希由 MIT 的 Karger 及其合作者提出,利用哈希函数将存储的 key 和服务器的哈希到同一个环形地址空间中,从而确定数据和服务器的映射关系。

相比于固定哈希系统扩容时需要对所有节点上的数据进行重新分布,当采用一致性哈希的系统扩容时,只需要迁移相邻节点上的数据。为了合理分配集群内异构节点的负载,Amazon 在 Dynamo 中引入虚拟节点(Virtual Node)概念,一个物理节点可配置成多个虚拟节点。 万方数据

3 基于分片的一致性哈希动态负载均衡策略

基于虚拟节点的一致性哈希仍然是通过哈希函数的平衡性,在概率上保证不同虚拟节点上分配到的数据量一致^[12]。然而,存储系统的负载不均还表现在系统访问压力的分布不均,面对动态变化的访问负载,数据的平均分布并不足以保证访问压力的均衡分布。面对复杂的负载场景,就需要对系统进行动态的负载均衡。改进策略利用分片机制对一致性哈希策略进行了相应的改进,主要包括两点:

(1)将虚拟节点和物理节点彻底解耦,采用分片(Fragment)作为数据分区存储的逻辑单元。

(2)基于分片负载监控,对系统负载进行动态的均衡调度。

基于虚拟节点的一致性哈希策略多采用静态配置文件维护虚拟节点与物理节点之间的映射关系^[13]。改进策略利用基于分片的一致性哈希,通过动态的分片管理物理节点与数据之间的映射关系,分片可以进行分裂、合并、迁移操作。

3.1 数据模型

在改进的一致性哈希策略中,数据与分片之间的位置关系使用哈希函数确定。一个分片处理哈希值在当前分片 ID 与前驱分片 ID 之间的数据,数据与分片之间的映射关系为:

$$F_i = (F_{i-1} \leq \text{Hash}(\text{Key}) < i) \quad (1)$$

其中, F_i 为目标分片 ID; F_{i-1} 为目标分片的前驱分片 ID; Hash 为选用的哈希函数,如 MD5、SHA-1 等。

利用哈希分区的静态负载均衡特性,使得数据在一致性哈希环地址空间内均匀分布^[14]。在获得目标分片 ID 后,需要查询目标分片所在的物理节点,分片与物理节点的映射关系在负载均衡过程中会变化,通过有序的 Map 表管理。在取得物理节点的地址后,客户端与物理节点直接进行数据操作。

物理节点上数据的读写操作都会被记录,存储系统的负载与单位时间内数据的读写请求次数以及数据传输量和存储容量消耗成正比。因此,存储系统中服务器的负载状况的主要影响因素有:单位时间读请求次数(记为 QPS)、单位时间写请求次数(记为 UPS)、单位时间内平均传输的数据大小(记为 DPS)和服务器的存储空间(记为 S)。而每台服务器所能达到的最大负载阈值能够通过硬件指标进行计算,或者通过性能压力测试来计算,因此,服务器的负载可以通过式(2)计算:

$$\text{load} = \alpha \frac{\text{QPS} \times \text{DPS}}{\text{QPS}_{\max}} + \beta \frac{\text{UPS} \times \text{DPS}}{\text{UPS}_{\max}} + \gamma \frac{S_{\text{used}}}{S_{\max}} \quad (2)$$

其中, QPS_{\max} 、 UPS_{\max} 为指定配置的服务器在系统

要求的读写延迟指标下,单位时间内的最大读写次数; α, β, γ 为权重因子。根据服务器压测数据设定,以分片为单位统计 UPS、QPS、DPS 三项负载元数据。

基于上述数据模型可以实现对分布式存储系统的动态负载均衡,存储系统的负载均衡涉及数据的跨节点迁移,均衡效果具有一定的滞后性。基于分片的一致性哈希动态负载均衡策略采用时间窗口的方式,对一个时间窗内的平均负载进行分析,从而决定当前节点或者分片是否过载。负载均衡的主要代价是数据跨节点迁移带来的网络资源占用,通过分片级和节点级两个级别的负载均衡操作,降低负载均衡数据迁移的粒度,从而优化负载均衡资源的消耗问题。

3.2 分片级负载均衡

除了节点间的负载不均问题之外,分片间的负载不均问题也会导致系统资源的浪费。同时,由于分片是跨节点数据迁移的最小单位,分片之间负载不均,提高了制定节点之间负载均衡策略的复杂度。而基于分片的一致性哈希动态负载均衡,在负载统计的粒度上可以由传统的节点级细化到分片级,利用分片的分裂、合并操作实现分片级负载均衡。

分片级负载均衡以同一节点上分片的平均负载为目标,首先对时间窗内整个节点上所有的分片计算出一个平均负载,记为 load_{avg} 。再以平均负载为基准设定阈值,负载高出阈值的分片加入超载队列,低于阈值的加入低载队列。对超载队列中的分片进行分裂操作,分裂为 M 个均载分片,若记待分裂分片负载为 load_{cur} ,则分裂的分片个数与当前负载的关系为:

$$M = \text{int}\left(\frac{\text{load}_{\text{cur}}}{\text{load}_{\text{avg}}}\right) \quad (3)$$

记 F_i 为分裂的第 i 个分片的分片号,则该分片号可由式(4)给出:

$$F_i = \text{int}\left(F_{i-1} + \frac{\text{load}_{\text{avg}} \times i}{\text{load}_{\text{cur}}} \times (F_i - F_{i-1})\right) \quad (4)$$

其中, F_0 为待分裂分片的前驱分片号。

分片的分裂和合并操作可以实现负载在分片之间的重分配。由于存储服务器同一个节点上磁盘内和磁盘间的数据带宽要远高于机器之间的网络带宽,因此,节点内分片之间的数据迁移效率要远高于节点之间的数据迁移效率。针对分片的负载均衡策略的主要流程如图 1 所示。对低于轻载阈值的分片,加入轻载分片队列,对分片进行合并操作,从而减少管理元信息的数量,降低系统管理开销。

3.3 节点级负载均衡

分片级的负载均衡提高了对分片的管理和操作效率,但并不能改善不同节点之间的负载不均问题,因此,还需要进行物理节点级别的负载均衡。物理节点

级别的负载均衡主要通过分片进行跨节点的迁移来完成。事实上,就是在改变分片与物理节点之间的映射关系,具体流程与分片级负载均衡相似。

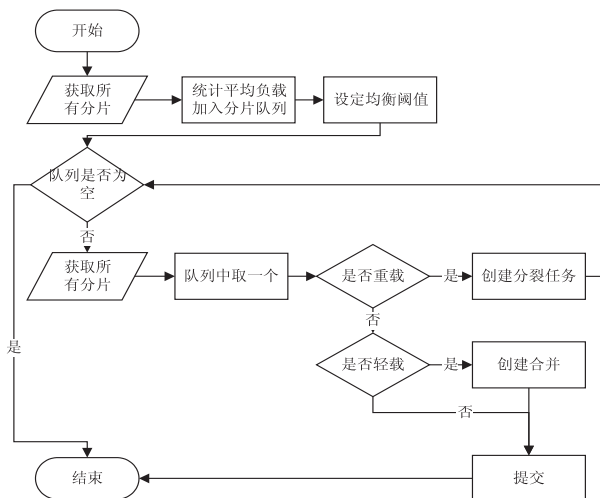


图 1 分片级负载均衡流程

节点间的负载均衡由于涉及到数据在网络中的传输,触发均衡的阈值一般设置的较高。负载均衡的流程与分片级均衡流程相似,同样是确定重载节点后,选取合适的分片向低载节点迁移。考虑到某些节点上分片平均负载较高,即便最小的分片迁移到其他节点也会导致其他节点成为重载节点,当出现这种情况时,均衡策略会先对当前节点中的部分分片进行分裂操作,然后再选取合适的分片迁移。

4 实验及结果分析

为了验证改进的一致性哈希动态负载均衡策略,通过一个负载均衡系统对其进行测试实验。

测试环境中,核心路由连接了动态负载均衡系统的管理集群、数据集群、监控集群以及客户端。管理集群是系统的中心节点;数据集群包含所有数据节点,部署了调度和监控模块;监控集群负责收集系统监控信息;客户端是访问负载均衡系统的入口。

系统采用接口化的设计,通过实现不同的元数据管理接口和均衡决策接口,就可以实现不同的数据分区策略和负载均衡策略。利用这个特性,分别实现了基于分片和基于虚拟节点的负载均衡系统,然后进行实验测试。

实验中,采用相同的元数据规模(3 个物理节点,12 个虚拟节点或分片)和测试数据,均衡的目标值设置为系统平均负载的 20% 以内。初始状态分片和虚拟节点的分布情况如表 1 所示。

通过施加一定规律的负载,对分片级别的负载均衡进行实验,当负载不均超过设定阈值时,负载均衡系统进行负载均衡操作。从图 2 可以看出,通过对重载

分片(85、425、765)的分裂操作,降低了重载分片的负载;同时,通过对轻载分片(255、595、935)的合并操作,平衡了分片之间差异过大的负载。

表 1 分片与虚拟节点分布表

节点	分片 ID
Node-1	85、170、255、340
Node-2	425、510、595、680
Node-3	765、850、935、1024
Vnode-1	1455903884345016...
	6961543807435993...
	4033834360016351...
	1253475815787404...
Vnode-2	1177497052027465...
	7991245042388408...
	1231656650235307...
	4596436882071935...
Vnode-3	26653217281101...
	17252768614305...
	13653161819971...
	15369936492691...

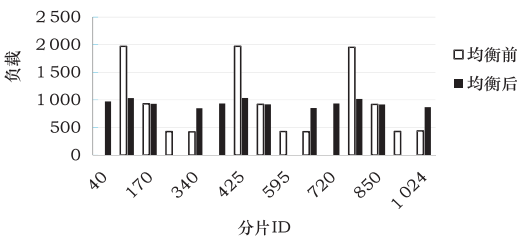


图 2 分片级负载均衡效果

在分片级均衡的基础上,对节点级负载均衡进行实验,分别对基于虚拟节点和基于分片的负载均衡系统施加负载压力。图 3 和图 4 分别是两个系统在负载均衡前的负载分布图。

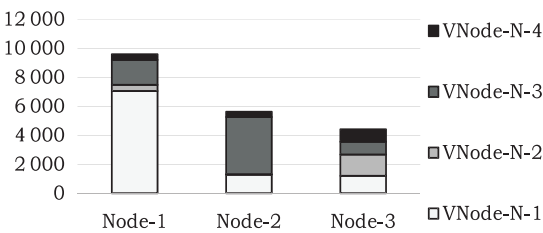


图 3 均衡前基于虚拟节点的系统负载分布图

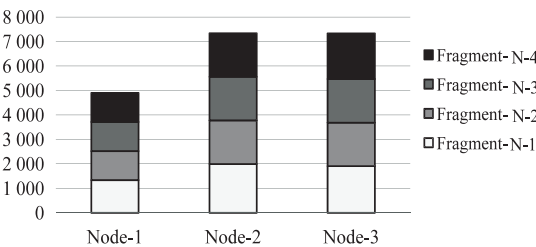


图 4 均衡前基于分片的系统负载分布图

从图 3 可看出,基于虚拟节点的系统不仅出现了物理节点之间的负载不均衡,同时在虚拟节点之间也出现了负载不均。从图 4 可看出,基于分片的系统,虽然出现了节点之间的负载不均,但得益于节点内部分片间的负载均衡策略,分片之间负载分布较为均衡。

经过一段时间的负载均衡调度,两个系统中各个物理节点的负载状况如图 5 所示。

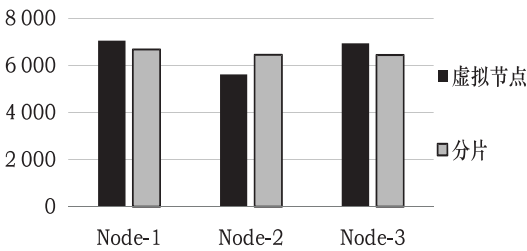


图 5 均衡后节点负载分布图

从负载均衡的效果来看,基于虚拟节点的系统 and 基于分片的系统都能够通过一系列的调度操作对负载不均衡的节点进行有效的负载均衡,达到将节点之间的负载控制在平均负载的 20% 以内的目标。但基于分片的系统在数据迁移量和均衡效果上要略优于基于虚拟节点的系统,迁移量上降低了约 30% 左右,且节点之间的负载也更加均衡。

对以上实验结果进行分析,基于分片的系统,在负载均衡过程中采用了两级均衡策略,其中分片级的负载均衡利用机器空闲资源在物理节点内部进行负载调度,使负载分布均衡。相比之下,基于虚拟节点的系统,在采用相同的元数据复杂度(虚拟节点数与分片数相同)的情况下,虚拟节点之间负载不均,导致在均衡决策时,迁移负载的粒度较大,在多数场景下迁移的数据量要高于基于分片的系统。

5 结束语

从一致性哈希出发,结合动态负载均衡技术,设计了一个基于分片的一致性哈希动态负载均衡策略。利用分片进一步解耦数据和节点的映射关系,并在对分片的监控数据进行动态收集、分析的基础上,对分片间和节点间的负载进行均衡调度。实验结果表明,该策略保留了一致性哈希在系统扩展性上的优势,同时优化了一致性哈希策略负载均衡的总体效果。借助基于分片的一致性哈希负载均衡策略,可以有效均衡系统负载,提高存储系统的效率。

参考文献:

[1] 黄秋兰,程耀东,陈刚. 分布式存储系统的哈希算法研究[J]. 计算机工程与应用,2014,50(1):1-4.

一种基于语义树与 VSM 的主题爬取策略,以优化爬虫的爬行路线,尽可能多地避开无关页面。通过将语义树应用于内容相关度计算,解决了使用传统向量余弦距离计算页面相似度没有考虑语义的问题。另一方面,分析子链接的相关度对当前链接相关度得分的影响,通过对链接进一步的分析,使得链接排序更加合理,有利于爬虫穿越隧道。实验结果及其分析均表明,该策略进一步提高了抓取的准确性,减少了无关的爬取存储操作。但语义相似度的计算需要依赖于语义树的构建,且该策略本身也没有涉及对爬取效率的提升。由于爬取效率与准确率一样,对主题爬取至关重要,因此提升主题爬取准确率将成为下一步工作中的研究重点。

参考文献:

- [1] Chakrabarti S, van den Berg M, Dom B. Focused crawling: a new approach to topic-specific web resource discovery [J]. *Computer Networks*, 1999, 31(11-16): 1623-1640.
- [2] de Bra P M E, Post R D J. Information retrieval in the world-wide web: making client-based searching feasible [J]. *Computer Networks and ISDN Systems*, 1994, 27(2): 183-192.
- [3] Hersovici M, Jacovi M, Maarek Y S, et al. The shark-search algorithm. An application: tailored Web site mapping [J]. *Computer Networks and ISDN Systems*, 1998, 30(1-7): 317-326.
- [4] Page L. The PageRank citation ranking: bringing order to the web [D]. California: Stanford University, 1998.
- [5] Kleinberg J M. Authoritative sources in a hyperlinked environment [J]. *Journal of the ACM*, 1999, 46(5): 604-632.
- [6] 张亮,尹存燕,陈家骏. 基于语义树的中文词语相似度计算与分析 [J]. *中文信息学报*, 2010, 24(6): 23-30.
- [7] 刘冬明,杨尔弘. 话题内相关文本的内容计算 [J]. *中文信息学报*, 2015, 29(5): 98-103.
- [8] Pal A, Tomar D S, Shrivastava S C. Effective focused crawling based on content and link structure analysis [J]. *International Journal of Computer Science and Information Security*, 2009, 2(1): 1-5.
- [9] Aggarwal C C, Al-Garawi F, Yu P S. On the design of a learning crawler for topical resource discovery [J]. *ACM Transactions on Information Systems*, 2001, 19(3): 286-309.
- [10] Hati D, Kumar A. An approach for identifying URLs based on division score and link score in focused crawler [J]. *International Journal of Computer Applications*, 2010, 2(3): 48-53.
- [11] Hati D, Kumar A, Mishra L. Unvisited URL relevancy calculation in focused crawling based on Native Bayesian classification [J]. *International Journal of Computer Applications*, 2010, 3(9): 23-30.
- [12] Ehrig M, Maedche A. Ontology-focused crawling of web documents [C]//*Proceedings of the 2003 ACM symposium on applied computing*. [s.l.]: ACM, 2003: 1174-1178.
- [13] Ganesh S, Jayaraj M, Kalyan V, et al. Ontology-based web crawler [C]//*International conference on information technology: coding and computing*. [s.l.]: IEEE, 2004: 337-341.
- [14] 于甜甜. 基于语义树的语句相似度和相关度在问答系统中的研究 [D]. 济南: 山东财经大学, 2014.
- [15] Mencaer F, Pant G, Srinivasan P. Topical web crawlers: evaluating adaptive algorithms [J]. *ACM Transactions on Internet Technology*, 2004, 4(4): 378-419.

(上接第 65 页)

- [2] 凌云,周华锋. 面向异构集群系统的动态负载均衡技术研究 [J]. *计算机工程与设计*, 2008, 29(12): 3068-3070.
- [3] Rai I, Alanyali M. Uniform weighted round robin scheduling algorithms for input queued switches [C]//*IEEE international conference on communications*. Helsinki, Finland: IEEE, 2001: 2028-2032.
- [4] Kim J S, Lee D C. Weighted round robin packet scheduler using relative service share [C]//*IEEE military communications conference*. [s.l.]: IEEE, 2001: 988-992.
- [5] 陈伟,张玉芳,熊忠阳. 动态反馈的异构集群负载均衡算法的实现 [J]. *重庆大学学报: 自然科学版*, 2010, 33(2): 73-78.
- [6] 朱虹宇,李挺,闫健恩,等. 基于动态负载均衡的分布式任务调度算法研究 [J]. *高技术通讯*, 2014, 24(12): 1261-1269.
- [7] 尹向东,杨杰,屈长青. 云计算环境下分布式文件系统的负载均衡研究 [J]. *计算机科学*, 2014, 41(3): 141-144.
- [8] 邓志飞,应良佳,王军威. 基于 IODA 算法 MongoDB 负载均衡的改进 [J]. *现代电信科技*, 2013(7): 9-13.
- [9] Karger D R, Lehman E, Leighton F T, et al. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web [C]//*ACM symposium on theory of computing*. [s.l.]: ACM, 1997: 654-663.
- [10] Sivasubramanian S. Amazon dynamoDB: a seamlessly scalable non-relational database service [C]//*ACM SIGMOD international conference on management of data*. [s.l.]: [s.n.], 2012: 729-730.
- [11] 张聪萍,尹建伟. 分布式文件系统的动态负载均衡算法 [J]. *小型微型计算机系统*, 2011, 32(7): 1424-1426.
- [12] 赵见. 高性能高可用键值存储系统的设计与实现 [D]. 成都: 电子科技大学, 2010.
- [13] Schintke F, Reinefeld A, Haridi S, et al. Enhanced paxos commit for transactions on DHTs [C]//*IEEE/ACM international conference on cluster, cloud and grid computing*. [s.l.]: IEEE, 2010: 448-454.
- [14] 田浪军,陈卫卫,陈卫东,等. 云存储系统中动态负载均衡算法研究 [J]. *计算机工程*, 2013, 39(10): 19-23.