计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100012

面向云服务的分布式消息系统负载均衡策略

高子妍　王　勇
北京工业大学信息学部计算机学院　北京 100124
(641909633@qq.com)

摘　要　针对云服务下分布式消息系统存在的节点间负载倾斜问题,提出基于副本角色的动态负载均衡策略,并将算法应用于Apache Kafka分布式流平台中。基于消息系统的主要功能为读写及存储消息,算法以CPU、磁盘、网络读写流量为节点的主要负载因素,并根据不同的负载类型提出相应的首领角色迁移策略和副本迁移策略。从时间代价、空间代价、服务可用性等多个角度论证该算法的可行性,并讨论算法中涉及参数对算法执行效果的影响。经实验验证,所提算法能够实现集群中各节点的资源使用量均不大于规定阈值,并且与缺省系统相比,集群CPU占用率均方差下降72.1%,磁盘占用率均方差下降86.1%,网络流入速度均方差下降79.2%,网络流出速度均方差下降63.9%,优化效果显著。

关键词:分布式消息系统;负载均衡;云服务;Apache Kafka;多副本机制
中图法分类号　TP393.4

Load Balancing Strategy of Distributed Messaging System for Cloud Services

GAO Zi-yan and WANG Yong
College of Computer Science and Technology,Faculty of Information Technology,Beijing University of Technology,Beijing 100124,China

Abstract　Aiming at the problem of load skew between nodes in distributed messaging systems under cloud services,a dynamic load balancing strategy based on the role of replica is proposed and the algorithm is applied to Apache Kafka,the distributed streaming platform. Because the function of the messaging system is to read,write and store messages,the algorithm used CPU,disk and bytes in/out as the main load factors of nodes,and proposed the corresponding Leadership Movement strategy and Replica Movement strategy according to different load types. The feasibility of the algorithm is demonstrated from the perspectives of time cost,space cost and service availability,and the influence of parameters involved in the algorithm on the execution of the algorithm was discussed. Experiment results show that,the algorithm can achieve that the resource usage of each node in the cluster is not greater than the specified threshold. Compared with the default system,the standard deviation of cluster CPU occupancy rate decreases by 72.1%,the standard deviation of disk occupancy rate decreases by 86.1%,the standard deviation of bytes in rate decreases by 79.2%,and the standard deviation of bytes out rate decreases by 63.9%. The optimization effect is remarkable.
Keywords　Distributed messaging system,Load balancing,Cloud service,Apache Kafka,Multi-replica mechanism

随着互联网与大数据技术的蓬勃发展,云计算也更加普及[1]。个人或企业级用户无需自己搭建流批数据处理系统或消息中间件等大规模的集群与应用,直接使用更加便捷的公有云、私有云或者混合云服务,即可实现随时从共享应用资源池中获取所需的应用资源[2]。在云服务提供过程中,服务提供者需根据服务等级协议(Service-Level Agreement,SLA)向客户提供云服务可用性的等级指标[3],当服务提供方没有达到SLA中承诺的标准时就要向客户提供赔偿。然而,面对大规模源源不断的数据,云服务下共享集群节点之间的负载倾斜问题严重影响了云服务的吞吐性能、可用性及稳定性[4-5],进而成为了SLA不达标的原因。目前已有的研究成果大多面向流批数据处理环境[6],无法较完善地解决分布式消息系统[7]负载不均衡的问题。

Apache Kafka[8](下文简称Kafka)作为当前最主流的分布式消息系统之一[9-11],能够成功解决海量实时数据的接入和处理问题[12-14]。其凭借高吞吐和低延时的性能优势,在工业界和学术界得到了广泛的认可。与此同时,Kafka也面临着一些挑战,传统的负载均衡算法在实际Kafka的应用场景中存在各种各样的问题[15]。本文提出了一种面向云服务的大规模Kafka集群动态负载均衡策略。

1　Kafka架构与问题描述

如图1所示,Kafka的架构主要由4部分组成:生产者、代理(Broker)、消费者,以及通过Zookeeper[16]集群来维护部分元数据信息。生产者主动(push)发布消息到代理节点,消费者主动从代理节点中拉取(pull)消息进行消费。每一条消息都会属于一个主题,一个主题实际就是一个消息队列。Kafka为了实现更高的吞吐率,提供将一个主题切分为多个分区(Partition)的功能,同一主题下多个分区的消息能够实现并发地生产和消费。分区是存储的逻辑单元,同一分区不能跨代理节点或跨同一节点的多个磁盘存储。如图2所示,分区通过使用多副本机制来保证可靠性和可用性。每个分区包含一个首领副本(LeaderReplica)和多个追随者副本(FollowerReplica)。由于Kafka不提供读写分离,只有首领副本

通信作者:王勇(wangy@bjut.edu.cn)

能够对外提供读写服务，生产者与消费者只允许和首领副本进行读写交互。追随者副本负责在内部进行消息同步，当首领副本所在的代理节点宕机后，系统能够迅速将追随者副本升级为首领副本（原首领副本自动降级为追随者副本），继续对外提供服务，从而不牺牲该分区的可用性。

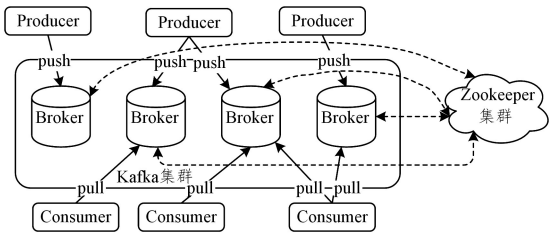


图 1 Kafka 的架构

Fig. 1 Architecture of Kafka

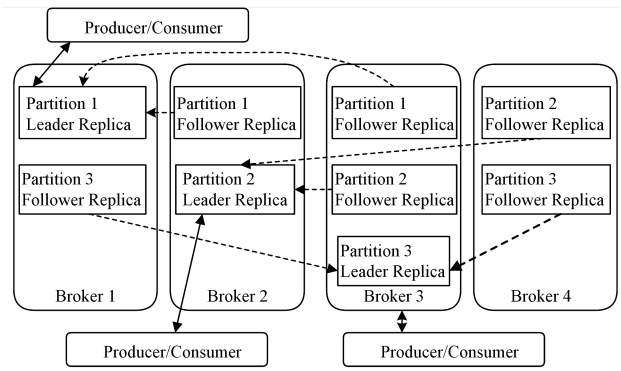


图 2 Kafka 的多副本机制

Fig. 2 Multi-replica mechanism of Kafka

Kafka 原生的负载均衡策略使用静态的轮询（Round-Robin）算法，即轮询所有分区，将每个分区的所有副本分配在不同的代理节点上。设代理节点总数为 N，具体算法如下：

（1）从[0，N）中随机选出一个节点作为起始位置；

（2）从[0，N）中随机生成一个轮询位移步长；

（3）依据副本 id，将分区下所有副本从 X 节点开始分配。用 α 表示当前副本的 id，则副本 α 应放置到 $(\alpha X\omega+X) \bmod N$ 位置的代理节点上；

（4）重复步骤（3），分配该主题下其他分区的所有副本到集群的代理节点。

基于上述架构及缺省的负载均衡策略，Kafka 集群存在如下 4 个负载倾斜相关问题：

（1）Kafka 不提供读写分离机制，对于 CPU 和网络读流量的使用率，首领副本远大于追随者副本。当集群各代理节点之间两种角色副本分配比例不一致时，即便保证各节点下总副本数量相同，各节点间及节点中各磁盘间同样会出现 CPU 和网络读流量使用严重不均衡的现象。

（2）当集群中新增加一个代理节点时，原先已经存在的主题和分区不会被主动迁移到该新增节点上。只有当新的主题被创建时，才有可能被分配到该新增代理节点。而云服务的使用场景常常为用户及主题稳定，流数据源源不断被传递到消息系统中。由此集群中新增节点和原有节点之间的负载会产生严重的倾斜问题。

（3）集群节点不可避免地会遇到崩溃或宕机等问题。当分区首领副本所在节点存在故障时，其他节点上的分区追随者副本被选举为首领副本，导致集群的负载状态混乱无规律，从而影响集群整体的健壮性和稳定性。

（4）系统只考虑到将分区分配给某个代理时使用轮询的负载均衡算法，但实际每个分区中的消息数量也是不一致的。当集中在某些代理节点上的某些分区较为活跃时，即出现热点分区和热点代理的场景，由于分区副本是存储在磁盘中的，导致集群中各节点以及节点中各磁盘空间使用率不均衡的问题。

2 负载均衡策略分析

负载均衡策略依据集群中各节点的资源利用率，动态调整分区与代理节点之间的分配关系，将存在于具有较高资源利用率代理节点中的负载迁移至具有较低资源利用率的代理节点中，在充分利用集群资源的同时减小部分节点的负载压力，以实现集群各节点间实际资源使用均衡，使共享集群能够更好地应对热点代理节点、热点分区等场景。节点负载情况应考虑多种资源，根据分布式消息系统所需的读写消息及存储消息的功能特性，将网络 I/O 流量（BytesIn/Bytes Out）、磁盘（Disk）存储容量及 CPU 占用率 4 个指标作为节点负载的主要评估依据。

分区是生产消费的基本逻辑单元。依据分区副本的角色不同，即首领角色和追随者角色，对各资源的影响也是不尽相同的。由于首领副本需要负责客户端（生产者和消费者）的读写服务，而追随者副本只负责和首领副本同步的消息，不对外提供服务，因此首领副本具有更高的 CPU 占用率及网络读流量；所有副本的数据内容是相同的，无论是首领副本还是追随者副本都具有相同的磁盘容量。不同角色副本的资源使用情况如表 1 所列。

表 1 不同角色的副本对不同资源的影响

Table 1 Impact of replica of different roles on different resources

资源	副本对资源负载的影响
CPU	Leader＞Follower
磁盘容量	Leader＝Follower
网络写流量	Leader＝Follower
网络读流量	Follower＝0

针对以上 4 种资源负载过高的场景，可定义以下两种应对策略。

定义 1（首领角色迁移，Leadership Movement） 将源节点上某分区首领副本的角色转移至目的节点上该分区的追随者副本。该策略需满足的必要条件为源副本是首领角色且目的代理节点上必须具有与源副本相同分区的追随者副本。

定义 2（副本迁移，Replica Movement） 将副本物理转移到其他代理节点上。该策略需满足的必要条件为目的代理节点在转移之前不具有与该副本相同分区的其他副本。满足该条件的目的是在执行副本迁移策略后，该分区仍然满足集群内的多副本机制，以保证分布式下该分区的可用性。

当代理节点的 CPU 占用率过高或网络读流量过高时，应首先采取首领角色迁移策略。将该节点上副本的首领角色转移至其他节点上的该分区的追随者副本，并将该节点上的原首领副本降级为追随者副本。若首领角色迁移策略实施完成后，代理节点的该项资源负载情况仍然过高，则应再采取副本迁移策略。这是由于首领角色转移操作比副本位置转移操作的代价小很多。首领角色的转移在本质上是一种逻辑转移，仅仅对元数据执行修改，数据本身没有转移或变动；而副本的

移动本质上是数据的物理移动，持续时间长并且很难被中断，中断后的状态也不能被精准地确定。因此，针对 CPU 和网络读流量负载倾斜问题，应先采取首领角色迁移策略，若不能满足负载均衡的目标则再进行副本迁移。

当磁盘占用容量过高或网络写流量过高时，无论是首领副本还是追随者副本对其资源的影响是相同的，因此没有必要采取首领角色迁移策略，直接实施副本迁移策略即可。

3 负载均衡算法

负载均衡算法的流程图如图 3 所示。

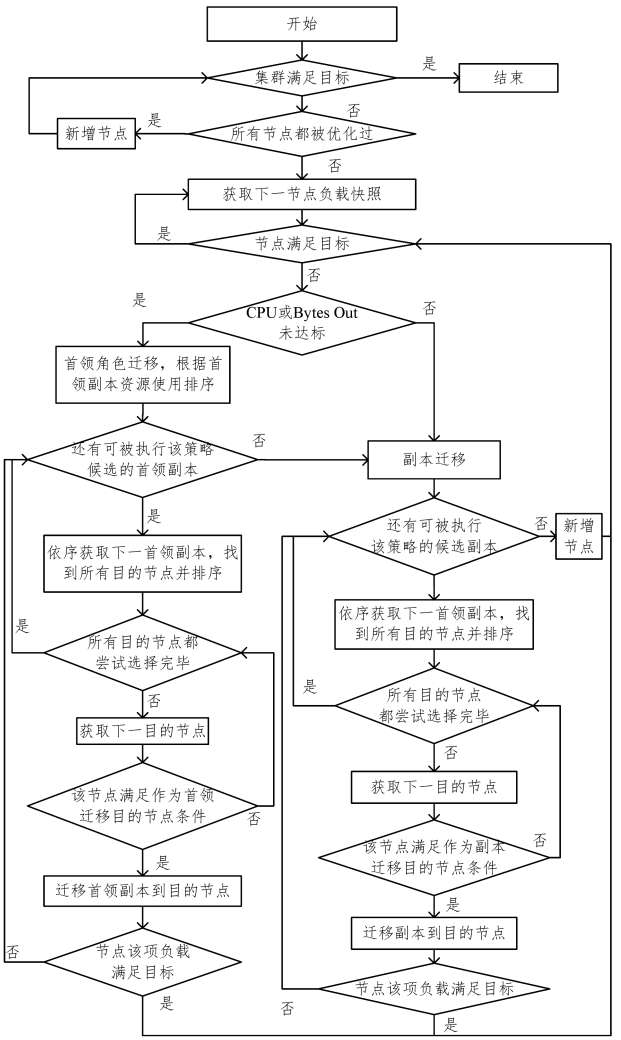


图 3 负载均衡算法流程图

Fig. 3 Flow chart of load balancing algorithm

一个集群能够被判定为负载均衡，应同时满足以下两个目标。

定义 3（容量目标，Capacity Goal） 集群内所有节点的资源使用都低于额定容量。设定 θ 为节点任一资源容量占用率的上限，则节点资源实际最大占用率 $B_{resource}(j)$ 应满足 $B_{resource}(j)\leqslant\theta$。

定义 4（分布均衡目标，Distribution Goal） 集群内所有节点的资源使用情况是均衡的，即集群资源使用均方差不超过额定阈值且集群内每一节点的负载偏离率不超过负载偏离的上限。设定集群线上代理节点数量为 N，$B_{resource}(j)$ 表示集群内代理节点 j 对某一资源的负载情况，则所有代理节点对该资源的负载均值 $\overline{B}_{resource}$ 为：

$$\begin{aligned}\overline{B}_{resource}&=\frac{\sum_{j=1}^{N}B_{resource}(j)}{N}\\&=\frac{B_{resource}(1)+B_{resource}(2)+\cdots+B_{resource}(N)}{N}\end{aligned}\tag{1}$$

对 $B_{resource}(j)$ 做归一化处理，归一化值 $D_{resource}(j)$ 为：

$$\begin{aligned}D_{resource}(j)&=\frac{B_{resource}(j)}{\sum_{j=1}^{N}B_{resource}(j)}\\&=\frac{B_{resource}(j)}{B_{resource}(1)+B_{resource}(2)+\cdots+B_{resource}(N)}\end{aligned}\tag{2}$$

$\overline{B}_{resource}$ 对应的归一化值 $\overline{D}_{resource}$ 为：

$$\begin{aligned}\overline{D}_{resource}&=\frac{\sum_{j=1}^{N}D_{resource}(j)}{N}=\frac{\sum_{j=1}^{N}\frac{B_{resource}(j)}{\sum_{j=1}^{N}B_{resource}(j)}}{N}\\&=\left(\frac{B_{resource}(1)}{B_{resource}(1)+B_{resource}(2)+\cdots+B_{resource}(N)}+\cdots+\frac{B_{resource}(N)}{B_{resource}(1)+B_{resource}(2)+\cdots+B_{resource}(N)}\right)/N\\&=\frac{B_{resource}(1)+B_{resource}(2)+\cdots+B_{resource}(N)}{\overline{B}_{resource}(1)+B_{resource}(2)+\cdots+B_{resource}(N)}\\&=\frac{1}{N}\end{aligned}\tag{3}$$

代理节点 j 对该资源的负载偏离率 $\gamma_{resource}(j)$ 可表示为：

$$\gamma_{resource}(j)=\frac{|B_{resource}(j)-\overline{B}_{resource}|}{\overline{B}_{resource}}=\frac{|D_{resource}(j)-\overline{D}_{resource}|}{\overline{D}_{resource}}\tag{4}$$

集群负载均方差 σ 为：

$$\sigma=\sqrt{\frac{\sum_{j=1}^{N}(D_{resource}(j)-\overline{D}_{resource})^2}{N}}=\sqrt{\frac{[(D_{resource}(1)-\overline{D}_{resource})^2+(D_{resource}(2)-\overline{D}_{resource})^2+\cdots+(D_{resource}(N)-\overline{D}_{resource})^2]}{N}}\tag{5}$$

设定 ε 为节点负载偏离率额定上限，则集群负载均方差阈值 η 可表示为：

$$\begin{aligned}\eta&=\sqrt{\frac{\sum_{j=1}^{N}(D_{resource}(j)-\overline{D}_{resource})^2}{N}}\\&=\sqrt{\frac{N\times(\varepsilon\times\overline{D}_{resource})^2}{N}}\\&=\varepsilon\times\overline{D}_{resource}=\frac{\varepsilon}{N}\end{aligned}\tag{6}$$

σ 和 $\gamma_{resource}(j)$ 应满足 $\sigma\leqslant\eta$ 并且 $\gamma_{resource}(j)\leqslant\varepsilon$。

当 $B_{resource}(j)\leqslant\theta$ 且 $\sigma\leqslant\eta$ 时，集群节点与负载之间的分配是均衡的，并且各节点的负载压力是正常的。此场景不需要开启负载均衡策略。当 $B_{resource}(j)>\theta$ 且 $\sigma>\eta$ 时，集群节点与负载之间的分配是不均衡的，并且这种不均衡导致部分节点的负载压力过大。此场景下应执行负载均衡策略，平衡各节点间的资源占用率，以减小部分节点的负载压力。当 $B_{resource}(j)>\theta$ 且 $\sigma\leqslant\eta$ 时，部分节点的负载压力过大，但集群节点与负载之间的分配仍然是均衡的，说明各节点的负载压力均不会过小。此时应尝试执行迁移策略，减小对应部分节点的负

载压力，若无效则说明集群的整体压力过大，应在集群中加入新的代理节点。

3.1 负载均衡算法的具体流程

(1)遍历集群中的每一个代理节点，逐一执行步骤(2)－步骤(6)。

(2)获取代理节点资源负载情况的快照，当发生 $B_{resource}(j)>\theta$ 或者 $\gamma_{resource}(j)>\varepsilon$ 时：

1)若资源是网络输入带宽或 CPU，则节点负载 $B_{resource}(j)>\theta$ 取值为最近一段时间资源负载的均值，并执行步骤(3)－步骤 (6)。

2)若资源是磁盘容量或网络输出带宽，节点负载 $B_{resource}(j)$ 取值为最近一次时间的磁盘空间利用率值，并执行步骤(4)－步骤(6)。

(3)采取首领角色迁移策略。将该节点上所有首领副本按照该资源使用率降序排序，按照排好的顺序逐一对副本进行首领角色的转移，直到代理节点上该资源的使用情况满足 $B_{resource}(j)\leqslant\theta$ 并且 $\gamma_{resource}(j)\leqslant\varepsilon$ 时，停止迁移该节点上的首领角色。

1)首领角色迁移目的节点选择策略。找到首领副本所属分区所有追随者副本及追随者副本所在的代理节点，将这些代理节点根据对该资源的使用情况按照升序排序，按照排列好的顺序逐一对目的代理节点执行步骤①和②：

①尝试将目标代理节点上该副本的首领角色转移到目的代理节点的原追随者副本上，但该次迁移需满足：1)迁移后目的节点 $B_{resource}(j)$ 及 $\gamma_{resource}(j)$ 仍不超过阈值；2)不会破坏本次迁移流程中涉及到的已经执行完成的其他迁移策略。否则，该节点不能作为目的代理节点，即跳过该节点，不予执行首领角色迁移策略。

②完成该副本首领角色的迁移后，对当前节点执行一次负载感知，若新负载值 $B_{resource}(j)$ 小于上限 θ 并且 $\gamma_{resource}(j)$ 低于负载偏离上限 ε，则停止步骤(3)，否则依顺序迁移该节点上下一首领副本。

2)待该节点上所有首领角色全部转移后，对当前节点执行一次负载感知，若新负载值 $B_{resource}(j)$ 依旧超过阈值 θ 或者 $\gamma_{resource}(j)$ 负载偏离仍超过上限 ε，则执行步骤(4)，否则执行步骤(5)。

(4)采取副本迁移策略。将该代理节点上的所有副本按照资源使用情况降序排序，依据排列好的顺序逐一迁移副本直到该代理节点资源使用满足 $B_{resource}(j)\leqslant\theta$ 并且 $\gamma_{resource}(j)\leqslant\varepsilon$ 时停止。

1)副本迁移目的节点选择策略。将所有代理节点按照对该资源的使用情况升序排序，依据排列好的顺序逐一对目的代理节点执行如下的①和②：

①尝试将目标节点上的副本转移到目的节点，但该次迁移需满足：1)目的节点上原本不具有该副本相同分区的其他副本；2)迁移后目的节点 $B_{resource}(j)$ 及 $\gamma_{resource}(j)$ 仍不超过阈值；3)不会破坏本次迁移流程中涉及到的已经执行完成的其他迁移策略。否则，该节点不能作为目的代理节点，即跳过该节点执行，不予执行副本转移策略。

②完成该副本的迁移后，对当前节点执行一次负载感知，若新负载值 $B_{resource}(j)$ 小于上限 θ 并且 $\gamma_{resource}(j)$ 低于负载偏离上限 ε，则停止步骤(4)，否则依顺序迁移该节点上下一副本。

2)该节点上所有副本全部转移后，对当前节点执行一次负载感知，若新负载值 $B_{resource}(j)$ 依旧超过阈值 θ 或者 $\gamma_{resource}(j)$ 已经超过负载偏离上限 ε，则集群应加入新的代理节点，否则执行步骤(5)。

(5)结束该节点的迁移策略，回到步骤 1)检测集群中下一代理节点的负载情况。

3.2 参数影响与代价评估

算法中部分参数的取值对算法的执行效果存在影响：θ 和 ε 本质上是负载均衡迁移策略启动的阈值，由集群服务端人为设定。这两个参数值共同决定了对负载感知的敏感程度：阈值过大会导致负载均衡算法反应迟钝，不能及时地反馈集群问题，无法实际起到集群负载均衡的作用来提高服务的可用性和集群的稳定性；而阈值过小会导致负载均衡算法过于敏感，首领角色及副本的迁移过于频繁，出现迁移抖动的现象。且实施算法时需要集群使用额外的 CPU 等资源，给集群增加了不必要的资源损耗。

在时间方面，负载迁移策略中为每一个负载压力较大的节点选择迁移目标需要涉及排序等步骤，时间复杂度 $T(n)=O(n\log n)$，其中 n 为目标节点中可作为迁移目标的数量。被迁移负载目的节点的选择同样需要排序，时间复杂度为 $T(n)=O(nm\log m)$，其中 n 为目标节点中可能作为迁移目标的数量，m 为被迁移负载可选的目的节点数量。首领角色的迁移仅仅修改 Zookeeper 中的元数据，时间消耗极小，通常为毫秒级别。副本迁移的时间开销与被迁移副本的数据量大小及网络传输速率相关，时间消耗大于首领角色迁移，通常为百毫秒至秒级别。

在空间方面，负载迁移策略并不会产生新的数据，仅仅是原数据的修改和物理位置的转移，不会对空间造成额外压力。

在执行负载迁移策略的过程中，不会影响包括分区或主题在内的任何服务的可用性。对于首领角色迁移策略，其速度极快，客户端几乎不会感知；对于副本迁移策略，即从目标节点将副本转移到另一目的节点，其过程与追随者副本向首领副本同步的过程完全一致，当目的节点上的数据完全同步后，才会修改元数据中新副本的存储位置，新副本才会对整个集群的服务可见，因此转移过程中不会牺牲该分区的可用性。

4 实验与分析

本实验以 Kafka 为平台，通过对比实验来测试本文提出的负载均衡策略的性能。实验搭建了两个 Kafka 集群，其中一个集群采用 Kafka 系统缺省的静态负载均衡策略，另一个集群使用本文提出的动态负载迁移策略。两集群所处的环境完全一致，包括不限于相同的集群规模、节点规格参数、主题与分区数量、分区副本数量、分区总数据量大小、分区每秒读写数据大小(Queries-per-second，QPS)，并采用相同的生产者和消费者同时与两个集群交互。

4.1 实验环境

为了更真实地模拟云计算场景，每个 Kafka 集群使用 3 台性能较高的服务器。在每台服务器中创建 2 台性能完全一

致的虚拟机作为代理节点，即每个 Kafka 集群中共有 6 个代理节点。Kafka 需要依赖 Zookeeper 维护部分元数据，在各台服务器中再创建 3 个虚拟机作为 Zookeeper 节点，即每个 Zookeeper 集群中共有 3 个节点。服务器具体规格参数和 Kafka 集群中节点的测试环境如表 2 和表 3 所列。

表 2 服务器规格参数

Table 2 Specification parameters of server

配置项	参数
操作系统	CentOS 6.3
处理器型号	Intel® Xeon® CPU E5-2683 v3 @2.00 GHz
处理器个数	4
内存/GHz	128
硬盘/TB	3

表 3 集群节点参数规格

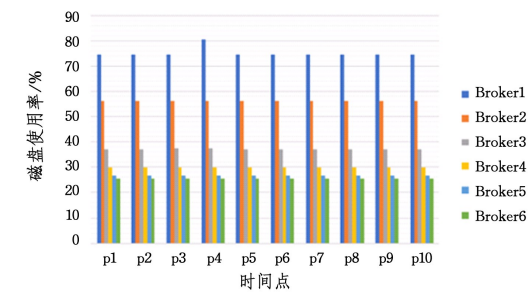
Table 3 Specification parameters of nodes in cluster

配置项类型	配置项	参数
硬件	处理器核数	20
	内存/GB	30
	硬盘/GB	500
软件	操作系统	CentOS 6.3
	Zookeeper 版本	3.4.13
	Kafka 版本	2.0.1

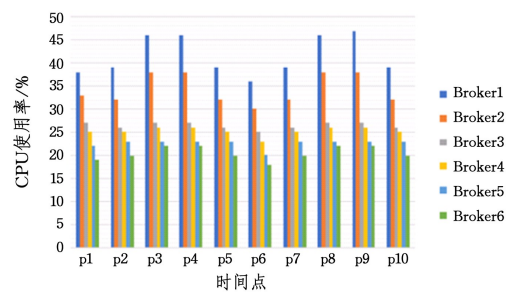
4.2 实验方法与结果分析

通过多次预实验及反馈调节，最终确定实验相关参数：$\theta=80\%$，$\varepsilon=10\%$。为避免个别任务存在偶然性误差，本实验分别采集 10 次两集群同一时刻的所有代理节点的负载情况，实验结果如图 4—图 5 所示，其中图 4 为原 Kafka 系统，图 5 为使用本文负载均衡策略的 Kafka 系统。

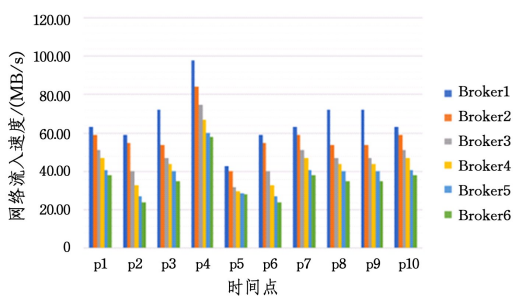
如图 3(a)所示，原系统使用的分区轮询策略对磁盘占用率不均衡的问题的影响较为明显。其中，Broker1 的磁盘负载最高，P1—P10 时刻 Broker1 的磁盘占用率均大于 70%，且在 P4 时刻 Broker1 的磁盘占用量大于阈值 80%，而负载较低的 Broker5 和 Broker6 在 P1—P10 时刻磁盘使用量均低于 30%，此时集群负载已出现严重的不均衡，该结果说明原系统不能够对不同数据量的分区做出对应的负载均衡策略。如图 3(b)—图 3(d)所示，对于 CPU 使用率，网络流入及流出速度的资源使用情况，虽然没有逼近额定上限值，但各个代理节点之间的资源使用量偏差较大。如图 5 及图 6 所示，集群在 P5 时刻网络读写流量骤降，但生产端与消费端并未暂停或减少读写请求，说明此时集群发生不可用状况。此现象在云计算环境下会使得该消息服务对外的可用性降低，造成 SLA 不达标。



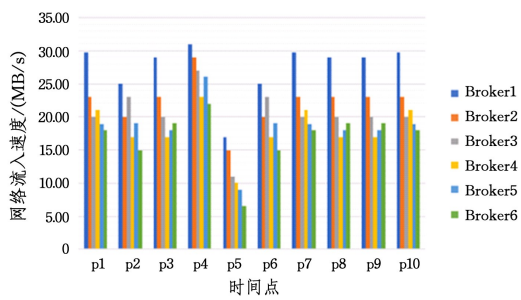
(a)磁盘使用率对比



(b)CPU 使用率对比



(c)网络流入速度对比



(d)网络流出速度对比

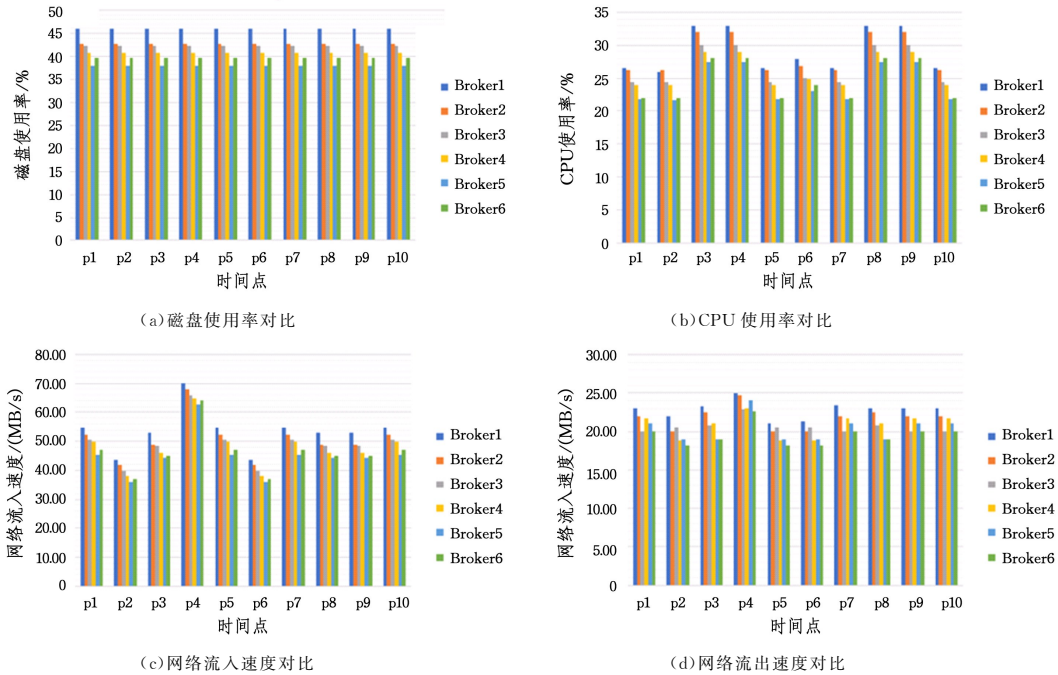
图 4 原策略下各代理节点的负载情况

Fig. 4 Load comparison of broker nodes under default strategy

如图 4 所示，对于运用本文负载均衡策略的系统，各代理节点之间的资源使用情况相差较小。如图 4(c)、图 4(d)所示，在 P5 时刻，新系统流网络量稳定，未发生读写流量骤减事故。

为了更精确地获得集群中各节点的负载差异，根据式(2)、式(3)及式(5)计算各集群各时刻的负载均方差，并取多次结果的均值。在原系统中，CPU 负载均方差为 0.043，磁盘负载均方差为 0.079，网络流入速度均方差为 0.048，网络流出速度均方差为 0.036；在运用本文策略的系统中，CPU 负载均方差为 0.012，磁盘容量负载均方差为 0.011，网络流入速度均方差为 0.010，网络流出速度均方差为 0.013。根据式(6)计算可得，集群的额定负载均方差为 0.017。上述计算结果如图 5 所示。在原系统中，CPU 占用率及网络流入/流出流量均大于额定负载均方差的 2 倍，而集群磁盘使用均方差大于额定均方差的 4 倍。可见，原系统不能保证各节点之间资源使用的均衡状态，且分区间数据量不一致的情况对负载倾斜的影响尤为严重。在实施本文负载均衡策略的系统中，集群四项资源使用均低于额定均方差，集群 CPU 占用率均方差降低 72.1%，磁盘占用率均方差降低 86.1%，网络流入均方差降低 79.2%，网络流出速度均方差降低 63.9%。本

文的负载均衡策略从副本角色、副本数量、各副本数据量大小等多个角度,综合考虑负载对节点的影响从而采取均衡策略,即便同样很难保证各节点之间的负载完全一致,但能够实现各节点资源使用尽量均衡,优化效果显著。



(a)磁盘使用率对比 (b)CPU 使用率对比

(c)网络流入速度对比 (d)网络流出速度对比

图 5 本文策略下各代理节点的负载情况

Fig. 5 Load comparison of broker nodes under strategy in this paper

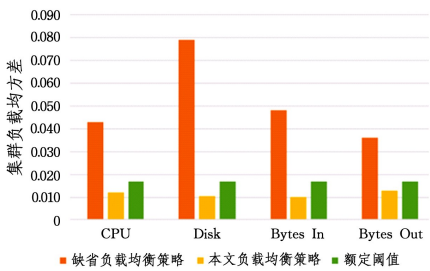


图 6 集群各负载均方差对比

Fig. 6 Standard deviation comparison of each load in two clusters

结束语 在分布式消息系统中,节点间负载倾斜问题是造成服务稳定性及可用性下降的重要原因。本文针对包括CPU、磁盘、网络读写流量在内的不同负载因素及不同副本角色,做出相应的首领角色迁移或副本迁移策略,以实现分布式消息系统的负载均衡。通过优化集群的负载状态,使集群中所有节点的资源使用及集群各负载均方差均不大于阈值。但本文算法也存在一定不足:本文只考虑以代理节点作为负载均衡的单位,但随着虚拟化及容器技术的成熟,一台主机上能够存在任意多个代理节点。下一步研究的重点应将主机负载和节点负载同时纳入算法比较的单位因素内,从而整体优化集群负载均衡状态。

参 考 文 献

[1] MENG X F,CI X. Big data management:concepts,techniques and challenges [J]. Journal of Computer Research and Development,2013,50(1):146-169.

[2] LUO Z J,JIN J H,SONG A B,et al. Cloud computing: architecture and key technology[J]. Journal on Communications,2011.

[3] DUAN Y C,WANG D P. A comparative study on SLA content of cloud service contract[J]. Computer and Networks, 2018 (21).

[4] VILLARS R L,OLOFSON C W,EASTWOOD M. Bigdata: What it is and why you should care[J]. IDC Analyze the Futhre White Paper,2011.

[5] RABL T,GOMEZ-VILLAMOR S,SADOGHI M,et al. Solving big data challenges for enterprise application performance management[C]// Proceedings of the VLDB Endowment. 2012: 1724-1735.

[6] COLLINS R L,CARLONI L P. Flexible filters:load balancing through backpressure for stream programs[C]// Proceedings of the Seventh ACM International Conference on Embedded Software. New York:ACM,2009:205-214.

[7] BELLAVISTA P,et al. Quality of Service in Wide Scale Publish-Subscribe Systems[J]. IEEE Communications Surveys & Tutorials,2014,16(3):1591-1616.

[8] Apache Kafka[EB/OL]. http://kafka.apache.org/.

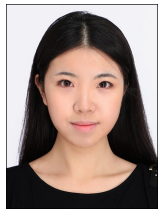
[9] BIRAJDAR P M,UJEDE K,YALAWAR R,et al. Bidirectional Hadoop kafka Managing Messaging Bus[J]. International Research Journal of Engineering and Technology(IRJET),2016, 3(3).

[10] AHUJA S P,MUPPARAJU N. Performance evaluation and comparison of distributed messaging using message oriented middleware[J]. Computer and Information Science,2014,7(4): 9-16.

[11] VIDELA A,WILLIAMS J J W. RabbitMQ in action: distributed messaging for everyone[J]. Manning About this Chapter Title Evaluation of Fairness in Message Broker System,2012.

[12] NARKHEDE N,SHAPIRA G,PALINO T. Kafka:The Defini-

tive Guide: Real-time Data and Stream Processing at Scale[M]. O'Reilly Media, Inc. 2017.
[13] KLEPPMANN M, KREPS J. Kafka, Samza and the Unix philosophy of distributed data[J]. Bulletin of the IEEE CS Technical Committee on Data Engineering, 2015.
[14] WANG G, et al. Building a Replicated Logging System with Apache Kafka[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1654-1655.
[15] BYZEK Y. Optimizing Your Apache Kafka Deployment: Levers for The throughput, Latency, Durability, and Availability[J]. Technical report, Confluent Inc, 2017.
[16] JUNQUEIRA F, REED B. ZooKeeper: Distributed Process Coordination[M]. Sebastopol: O'Reilly Media, Inc. 2013.



GAO Zi-yan, bachelor. Her main research interests include distributed computing and big data.

WANG Yong, born in 1974, Ph.D, associate professor. His main research interests include parallel and distributed computing.

(上接第 298 页)

参 考 文 献

[1] LPEZ-BENITEZ M, CASADEVALL F. Discrete-Time Spectrum Occupancy Model based on Markov Chain and Duty Cycle Models[C]// 2011 IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN). IEEE, 2011.
[2] DALTA D, WYGLINSKI A M, M INDEN G J. A spectrum surveying framework for dynamic spectrum access network[J]. IEEE Transactions on Vehicular Technology, 2009, 58(8): 4158-4168.
[3] HAM ID E, SITHAMPARANATHAN K, BILL M, et al. Spectrum occupancy prediction using a hidden Markov model[C]// Signal Processing and Communication Systems. Cairns, QLD, IEEE, 2015: 1-8.
[4] WANG L, XIE S G, SU D L, et al. An Autonomous Detection and Robust Estimation Method of Spectrum Anomaly Based on Time Series Analysis[J]. Acta Electronica Sinica, 2014, 42(6): 1055-1060.
[5] WEI H H, JIA Y F. A Method for Analysis of Non-linear and Non-stationary Spectrum Occupancy Time Series[J]. Acta Electronica Sinica, 2017, 45(8): 2026-2030.
[6] GERS F A, SCHMIDHUBER, JÜRGEN, et al. Learning to Forget: Continual Prediction with LSTM[J]. NeuralComputation, 2000, 12(10): 2451-2471.
[7] HOCHREITER S, SCMIDHUBER J. Long Short-Term Memroy[J]. Neural Computation, 1997, 9(8): 1735-1780.
[8] NG Y H, HAUSKNECHT M, VIJAYANARASIMHAN S, et al. Beyond short snippets: Deep networks for video classification [C]// 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015.
[9] KOK I, SIMSEK M U, OZDEMIR S. A deep learning model for air quality prediction in smart cities[C]// 2017 IEEE International Conference on Big Data (Big Data). IEEE, 2017.
[10] ZENG Z, LI L, CHEN J. Deeply Hierarchical Bi-directional LSTM for Sentiment Classification[J]. Computer Science, 2018, 45(8): 213-217, 252.
[11] HUANG, NORDEN E. The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-Stationary Time Series Analysis. Proceedings: Mathematical[J]. Physical and Engineering Sciences, 1998(454): 903-995.
[12] QI Y Y, YU M. Anti-jamming Method for Frequency Hopping Communication Based on Single Channel BSS and EMD[J]. Computer Science, 2016, 43(1): 149-153.
[13] MOTIN M A, KARMAKAR C, PALANISWAMI M. Selection of Empirical Mode Decomposition Techniques for Extracting Breathing Rate from PPG[J]. IEEE Signal Processing Letters, 2019: 1-1.
[14] ZHENG D, CUI G, CAO J, et al. Analysis of Brain-Death EEG Data Using 2T-EMD Algorithm[C]// International Conference on Signal-image Technology & Internet-based Systems. IEEE, 2016.
[15] GONG B M, WANG W B, ZHAO P. EMD-FSVM Prediction for Nonstationary Time Series [J]. Computer Science, 2014, 41(S2): 57-60.
[16] ELMAN J L. Finding Structure in Time[J]. Cognitive Science, 1990, 14(2): 179-211.
[17] International Telecommunication Union. Spectrum occupancy measurement and evaluaton[OL]. https://www.itu.int/rec/R-REC-SM.1880/en.

ZHAO Xiao-dong, born in 1991, master. His main research interest include big data, artificial intelligence, deep learning.

XU Jiang-feng, born in 1965, Ph.D. professor. His main research interest include big data, cryptography, database.