doi:10.3772/j.issn.1002-0470.2023.01.004

面向 Kafka 共识消息传输的负载均衡算法①

苏玉钊② 孙恩昌③ 杨睿哲 李 萌 张延华 司鹏搏 张 卉

(北京工业大学信息学部 北京 100124)

摘 要 Kafka 是一种高吞吐量消息中间件,但该算法存在负载均衡导致消费者消息处理效率下降的问题。本文提出一种改进的 Kafka 负载均衡算法,其中协调者基于消费者和分区的对应关系,根据不同的负载均衡场景优化调整消费者数目,按照业务负载倒序的方式更新消费者和分区的对应关系,优先处理负载较大的业务分区,提高消息传输效率,并且搭建 Fabric 联盟链验证分析算法的性能。实验结果表明,本算法在中央处理器(CPU)资源消耗比其他算法低 5% 的情况下,共识速度提升了 2% ~7%,并且在 6 个 Kafka 节点中 3 个宕机的情况下仍然能共识上链,提升了 Kafka 负载均衡算法的效率和稳定性。

关键词 Kafka;负载均衡;区块链;消费者;优化

0 引 言

近年来,随着业务数据传输量的快速增加,大数据、云计算需求爆发式增长,传统的集中搭建服务器面向用户提供服务的方式已经跟不上大数据的时代。为了满足快速增长的数据传输速度需求,分布式系统的应用成为必然。作为一个分布式发布-订阅消息系统,Kafka 是当前较为主流的高吞吐量消息中间件[1],它支持上层的应用端多语言开发,并且支持实时性的大规模消息处理。相比于其他的消息传输系统,如 Scribe、Chukwa、Flume,Kafka 凭借着高吞吐量、可扩展性和高并发等优势[2],已经获得广泛的行业应用。区块链是一种分布式账本技术,其早期交易速度较慢,但拥有的防篡改、可追溯等特点,与 Kafka 拥有高吞吐量但是缺乏防篡改等特点高度契合。Hyperledger Fabric 是使用 Kafka 的一种区块链解决方案,Fabric 拥有高度模块化的体系结构,提高了区块链的机密性和灵活性。

近年来国内外针对 Kafka 开展了广泛的研究。文献[3]基于 Kafka 和区块链的分布式信任机制,提出了数据存储和共享系统,具有应用的普适性。文献[4]针对目前存在的信息共享困难、信息更新传递速度不及时等问题,通过区块链技术将其应用于煤矿安全管理领域。文献[5]针对分区过载问题中文件配置管理的被动、僵化及鼓励不足的问题,提出一种改进型分区过载优化算法,有效缓解了分区过载问题。文献[6]在政府信息系统领域,根据政务系统多部门合作开放、半开放的特点,采用联盟链架构,将部分关键数据上传上链,实现了部分审核的自动化,进一步提高了政务工作的可信度。面对云服务下分布式消息系统存在的节点间负载不均衡问题,文献[7]针对云服务下分布式消息系统的节点间负载不均衡问题,提出基于消息副本的动态负载均衡策略,优化了整体的资源消耗。文献[8]针对原生负载均衡存在的吞吐量不高、可靠性不足问题提出了一种动态负载均衡算法,利用节点的负载值给出副本迁移策略,达到节点间负载均衡。在 Kafka 应用方面,文献[9]提出了一种基于 Kafka 的具有高度扩展性的框架,超越了传统的数据监控系统,保留了受监控数据的有限历史记录,在功能评估、可

① 国家自然科学基金(61901011)和北京市教育委员会科技计划一般项目(KM202110005021,KM202010005017)资助。
② 男,1998 年生,硕士生;研究方向:工业互联网,区块链;E-mail:iamsuyuzhao@emails.bjut.edu.cn。
③ 通信作者,E-mail:ecsun@bjut.edu.cn。
(收稿日期:2021-12-09)

扩展性、弹性和端到端消息延迟方面表现良好。文献[10]扩展了 Kafka-ML 框架,将 Kafka 和深度神经网络(deep neural networks, DNN)相结合,实验表明,通过在整个云到物连续体中分布 DNN 模型来显著提高相应时间和吞吐量。文献[11]提出了一个概率的、优先级驱动的 Kafka 消费者,在边缘容器中执行调度卸载的任务,实验表明其准确率提高了 26%,减少了 7% 低优先级任务的长堆积,降低了低容量设备的负载。针对订阅量较大的情况,文献[12]提出了一种新的主题类型,改善了基于内容的数据分发延迟,实验证明,其降低延迟能力较原本 Kafka 主题提高约 3.7 倍。

本文针对 Kafka 负载均衡导致消费者消息处理效率下降的问题,提出一种基于区块链的 Kafka 共识算法优化机制。当 Kafka 消费者端存在负载消耗过大、不均衡问题时,以协调者统一分配部署的方式,优化 Kafka 负载均衡过程,并对所有消费者的运行状态进行监控,根据消费者和分区的数量动态调整其对应关系,适时发起负载均衡,调整系统工作状态。算法的有效性通过构建一个 Fabric 联盟链系统进行验证,实验结果表明,本文提出的 Kafka 改进算法在中央处理器(central processing unit, CPU)资源消耗比其他算法低 5% 的情况下,共识速度提升了 2% ~7%,并且在 6 个 Kafka 节点中 3 个宕机的情况下仍然能共识上链,提升了 Kafka 负载均衡算法的效率和稳定性。

1 Kafka 负载均衡

Kafka 是一种高吞吐量、分布式、基于发布订阅的消息系统[13],包括生产者(Producer)、消费者(Consumer)和代理(Broker)等角色[14],其每秒可以处理几十万条消息[15],有良好的容错性。其中,生产者集群由若干向 Kafka 主题发布消息的生产者组成,消费者集群由若干订阅 Kafka 主题消息的消费者组成,Kafka 集群则由若干独立的 Kafka 服务器即 Broker 组成。Broker[16]用于接收来自生产者的消息并存储至服务器,同一个类型的消息属于一个主题[17],而一个主题往往分为若干个分区存储在不同的 Broker 上,同时,该主题还会有一个副本存储在 Broker 上。副本的存在保障了信息的高可用,提高了系统的容错率。信息传输的基本流程为:生产者向某一主题发布消息,对该主题感兴趣的 Kafka 消费者将订阅该主题并且接收处理来自该主题的消息,每个主题由多个分区[18]组成,消息以附加的方式写入分区,并且在分区内 Kafka 消费者将按照消息到达分区的先后顺序读取消息。图 1 为 Kafka 基本框架图,以图中所示分区与副本分配情况为例,生产者 *A* 发送消息至主题 *A* 的两个分区之一,消费者 A 通过订阅主题 *A* 从该分区中读取消息。一般同一个主题的消息在所有分区之间平均分布以保证分布均衡,主题 *A* 在 Broker 1 中有分区 1,在 Broker 2 中有分区 2,同时主题 *A* 会在 Broker 3 中留存一份副本作为冗余备份。对于主题 *B*、生产者 *B* 和消费者 *B* 将以类似的方式发送和接收消息。

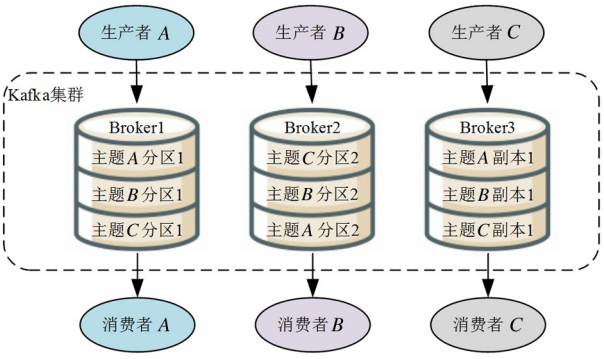


图 1 Kafka 框架

如果部分消费者宕机,则将导致整个消息系统信息传输能力下降,此时 Kafka 会自动平衡消费者与信息分区之间的对应关系,减弱由于消费者的减少或者增多带来的消息处理能力下降问题[19],提高生产环境下 Kafka 的适用性。

Kafka 负载均衡算法的简要流程[20]是将目标主题的分区和该分区对应所有消费者排序,得到消费者所需要分配的分区数量(向上取整),分配新的分区给消费者。原算法中每个消费者仅使用 Zookeeper 来得知分区的情况,消费者之间并无通信,不同的消费者可能会得到不一致的结果,导致在负载均衡的过程中,得到错误的负载均衡结果,造成资源浪费。

2　Kafka 改进算法

针对 Kafka 执行负载均衡操作导致的消费者消息处理效率低下的问题，通过在算法框架中加入协调者角色，集群中每个消费者都向服务器订阅消息，每个消费者遵从协调者指令，由协调者统一分配部署完成信息的传递。基于 Kafka 原有的信息传输流程，由生产者集群发布消息，随后传输至代理服务器处，并且根据一定的存储规则，以不同的主题存储在 Broker 上，算法流程如图 2 所示。

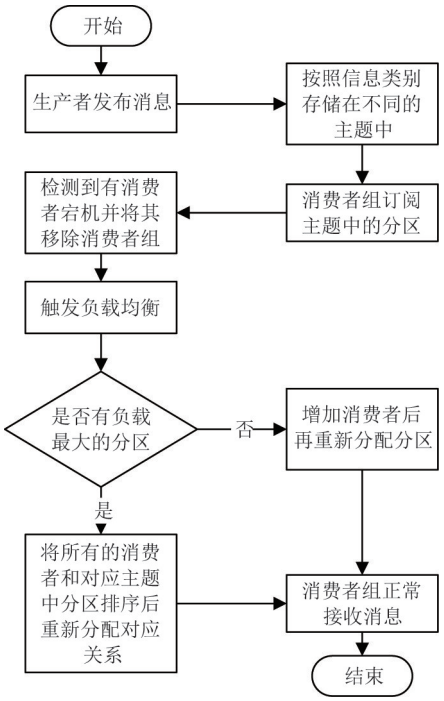


图 2　算法流程

在主题中有很多的分区，消费者组通过订阅主题中的不同分区来接收所需信息。本文中定义 Kafka 集群所有消费者组中正常接收消息时间最长的消费者为协调者，它负责一个或者多个消费者组的负载均衡工作，每一个消费者都向协调者提交订阅情况，由协调者统一分配初始分区。如果协调者接收不到来自消费者的心跳检测信号，则判断存在出现异常情况的消费者，随即触发负载均衡。

首先，由协调者检测出现异常情况消费者所在的消费者组，判断该消费者组中消费者与对应主题的负载情况。如果该主题中存在无消费者接收的分区，不加以处理则此分区中消息会因为无法被接收导致消息堆积，分区负载过大。协调者获取主题中每一个分区的负载状态，并记为 $\delta(t)$，将分区数量记为 N，得出这一分区的平均负载为 $\Delta^N(t)$，值为 $\{\delta_1(t),\ \delta_2(t),\ \delta_3(t),\cdots,\delta_i(t),\cdots,\delta_N(t)\}$。此时，如果有负载较大的分区，则将该分区所在主题中所有的分区按照负载情况由大到小排序，并将排序结果存于数据表中。分区的集合为 P，数据表中第 n 个分区记为 n_n，并将该主题对应的消费者组中所有消费者按照正常接收信息时间长短由大到小排序，消费者集合为 C，消费者组中第 n 个消费者记为 m_n。若此时 $n = m$，则将分区和消费者一一对应，即为

$$n_n = m_n \tag{1}$$

若此时 $n > m$，则一个消费者对应 y 个分区，表示为

$$y = n/m \tag{2}$$

若 y 为正整数，则每个消费者对应 y 个分区，若 y 不是正整数，每个消费者对应 $y+1$ 个分区，即向上取整，且无论 y 是否为正整数，分区分配至最后一次时，剩余的分区分配给最后的消费者。

若没有负载最大的分区，即 $\delta_i(t) = \Delta^N(t)/n$，说明此时所有分区都有消费者负责接收信息。由于消费者组中消费者与分区的数量不匹配，触发了负载均衡，所以仍需要平衡消费者数量，此时应增加或减少 z 个消费者，满足：

$$k(m \pm z) = n \tag{3}$$

其中 k 为正整数，再按照顺序重新将分区平均分配给消费者。此时整个 Kafka 系统将不会有信息分配不均衡的情况，负载均衡工作也已完成。具体算法描述如算法 1 所示。

算法 1　基于消息流通的 Kafka 负载均衡优化算法

协调者获取每个分区的负载状态。分区的数量为 N，分区的平均负载表示为

$\Delta^N(t) = \{\delta_1(t),\ \delta_2(t),\ \delta_3(t),\cdots,\delta_i(t),\cdots,\delta_N(t)\}$。

If　$\delta_i(t) > \dfrac{\Delta^N(t)}{n}$　then

将所有分区记为

$P = \{n_1,\ n_2,\ n_3,\cdots,n_i,\cdots,n_n\}$，

将所有消费者记为

$$C = \{m_1, m_2, m_3, \cdots, m_i, \cdots, m_n\}$$

If $M = N$ then

$$m_1 \leftarrow n_n, m_2 \leftarrow n_{n-1}, m_3 \leftarrow n_{n-2}, \cdots, m_m \leftarrow n_1$$

Else if $M < N$ then

if $n - \left(\frac{kn}{m} - 1\right) > 0$ then

$$m_1 \leftarrow n_n \sim n_{n-(\frac{n}{m}-1)}, m_2 \leftarrow n_{n-\frac{n}{m}} \sim n_{n-(\frac{2n}{m}-1)}, m_3 \leftarrow n_{n-\frac{2n}{m}} \sim n_{n-(\frac{3n}{m}-1)}, \cdots, m_i \leftarrow n_{n-\frac{(k-1)n}{m}} \sim n_{n-(\frac{kn}{m}-1)}, m_m \leftarrow n_{n-\frac{kn}{m}} \sim n_1$$

Else if $\delta_i(t) = \dfrac{\Delta^N(t)}{n}$, then

保持消费者数量满足 $k(m \pm z) = n$,k 为正整数。按顺序分配分区,恢复消息传输过程。

End if

如果某一消费者宕机,不能正常接收消息,此时会暂停消息拉取工作,重新分配消费者与分区的对应关系,即触发负载均衡,如图 3 和图 4 所示。图 3 和图 4 分别对应触发负载均衡时是否存在负载最大的分区的 2 种情况。图 3 中,由于消费者 3 宕机下线,此时原本由消费者 3 负责的分区 3 中的消息无法被接收,导致分区 3 中的消息传输滞后。由于协调者接收不到来自消费者 3 的心跳检测反馈,此时将触发负载均衡,暂停消息传输工作,同时检查消费者组中有能力接收消息的消费者个数和当前主题中分区的数量。协调者将此主题中所有分区和消费者排序,然后按照如图 3 所示关系分配对应关系,在宕机的消费者重新启动之前,都会由新的消费者接收此分区。

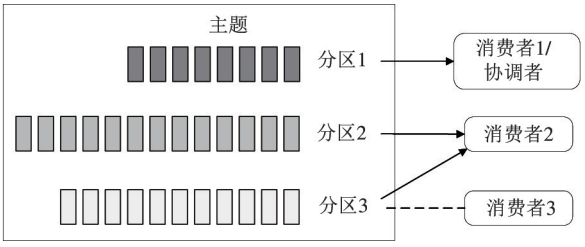


图 3　主题中有最大负载分区时

图 4 中,2 个消费者组共 4 个消费者接收同一主题中的 4 个分区,每一个分区对应不同消费者组中的 2 个消费者。若消费者组 2 中的某一消费者突然宕机下线,此时协调者接收不到来自消费者 4 的心跳信号,随即触发负载均衡。但此时在主题中没有负载最大的分区,则协调者将增加一个新的消费者来替代宕机下线的消费者,由新的消费者重新接收来自分区 3 和分区 4 的消息,从而达到 1 个分区对应 2 个消费者的均衡情况,此时负载均衡过程完成。

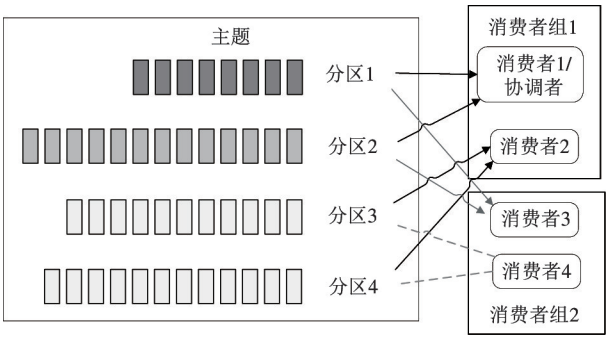


图 4　主题中无最大负载分区时

3　Kafka 共享算法实现

3.1　区块链设计

本文构建了 Fabric 联盟区块链网络验证算法,采用了 MySQL 数据库的方式验证用户身份信息。Fabric 联盟链[21]主要通过链式结构来保证数据的不可篡改,通过智能合约[22]来约束各参与方的行为。区块链架构保证信息可追溯、不可篡改,摆脱了第三方的参与,提高了交易的安全性和可靠性,也节省了能耗。通过采用在 MySQL 数据库中存储用户信息的方式来区分管理员用户和普通用户,让有操作权限的管理员用户查看区块链内部具体信息,如通道交易信息等,而没有操作权限的普通用户只能查询自己的私有信息,一定程度上提高了信息的安全性和隐私性。

Fabric 联盟链在多机模式下使用的共识方法是 Kafka,在 Fabric 联盟链中通过 Zookeeper 分布式协调服务来管理 Kafka 集群,Kafka 增加或者减少服务器都会在 Zookeeper 节点上触发相应的事件,Kafka 系统会捕捉这些事件,进行新一轮的负载均衡。组织 1 和组织 2 中所有节点均通过业务通道传输信息至排序节点上链,提交的共识信息上传至 Kafka 节点,系统的基本网络架构如图 5 所示。

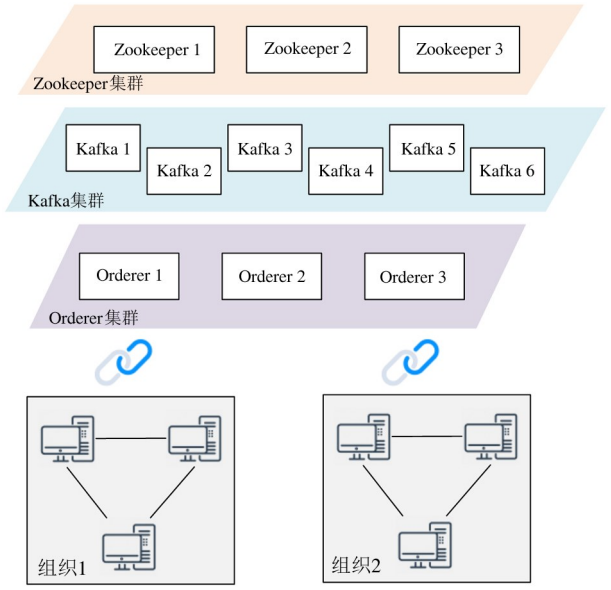


图 5　系统功能架构

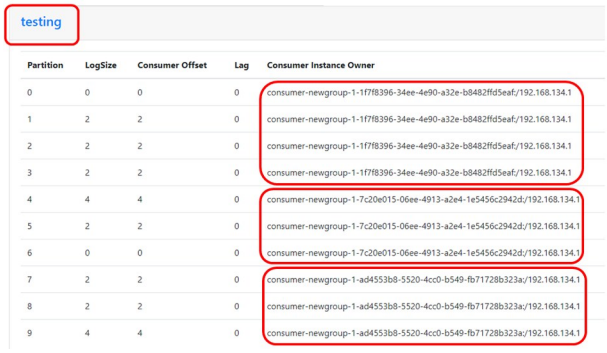
如图 5 所示，本文用 6 台电脑模拟一个 Fabric 联盟链环境。由 3 个 Zookeeper 节点管理 6 个 Kafka 节点，6 台电脑分别作为 6 个消费者，每一个消费者相当于一个 Kafka 节点。其中 3 个消费者加入组织 1，另外 3 个消费者加入组织 2，6 个消费者均加入一个业务通道，2 个组织在业务通道中对业务信息达成共识并上链。

本文中的区块链系统采用开源项目超级账本 Fabric 和 Kafka 共识协议。为了测试本文提出的改进 Kafka 共识算法的稳定性与可靠性，通过使用 Kafka 传输上链数据的 Fabric 联盟链，分别测试共识上链速度和 CPU 资源消耗情况来验证改进算法的稳定性与可靠性。

3.2　算法设计与实现

基于上述区块链系统设计，本文首先在 Newgroup 消费者组中创建 3 个消费者，如图 6 所示。在消费者组中每个消费者都会发送心跳检测，即每间隔一段时间发送心跳检测信号至协调者证明该消费者仍处于工作状态。如果超过设定的时间阈值协调者没有收到心跳检测信号，则认为该消费者宕机，将其从所在的消费者组中移除。

如图 6 所示，在名为 Testing 的主题中有 10 个分区，并且消费者组 Newgroup 对应此主题，每个消费者都拥有自己独一无二的数字身份，且负责主题中

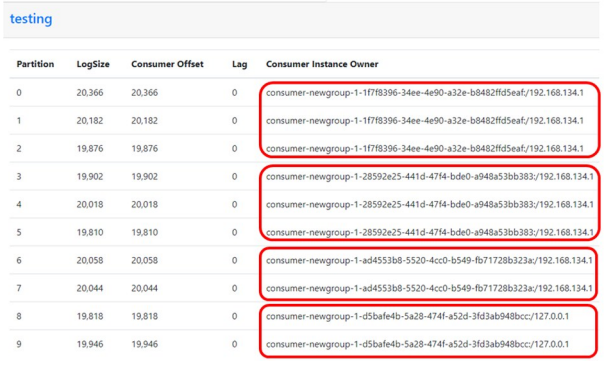

testing

Partition	LogSize	Consumer Offset	Lag	Consumer Instance Owner
0	0	0	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
1	2	2	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
2	2	2	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
3	2	2	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
4	4	4	0	consumer-newgroup-1-7c20e015-06ee-4913-a2e4-1e5456c2942d;/192.168.134.1
5	2	2	0	consumer-newgroup-1-7c20e015-06ee-4913-a2e4-1e5456c2942d;/192.168.134.1
6	0	0	0	consumer-newgroup-1-7c20e015-06ee-4913-a2e4-1e5456c2942d;/192.168.134.1
7	2	2	0	consumer-newgroup-1-ad4553b8-5520-4cc0-b549-fb71728b323a;/192.168.134.1
8	2	2	0	consumer-newgroup-1-ad4553b8-5520-4cc0-b549-fb71728b323a;/192.168.134.1
9	4	4	0	consumer-newgroup-1-ad4553b8-5520-4cc0-b549-fb71728b323a;/192.168.134.1

图 6　创建消费者

的 3 个或 4 个分区。由于分区 0 中 LogSize 为 0，可以得知当前第 1 个消费者对应的第 1 个分区并没有消息，但是第 1、2 和 3 个分区都已经有了 2 条消息，并且已经被接收。

此时消费者端触发负载均衡，在 Newgroup 消费者组中加入新的消费者，在重新衡量了消费者组中消费者的个数与主题中分区的个数后，按照倒序分配，靠前的每个消费者分配 2 个，最后分配 3 个分区。如图 7 所示。


testing

Partition	LogSize	Consumer Offset	Lag	Consumer Instance Owner
0	20,366	20,366	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
1	20,182	20,182	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
2	19,876	19,876	0	consumer-newgroup-1-1f7f8396-34ee-4e90-a32e-b8482ffd5eaf;/192.168.134.1
3	19,902	19,902	0	consumer-newgroup-1-28592e25-441d-47f4-bde0-a948a53bb383;/192.168.134.1
4	20,018	20,018	0	consumer-newgroup-1-28592e25-441d-47f4-bde0-a948a53bb383;/192.168.134.1
5	19,810	19,810	0	consumer-newgroup-1-28592e25-441d-47f4-bde0-a948a53bb383;/192.168.134.1
6	20,058	20,058	0	consumer-newgroup-1-ad4553b8-5520-4cc0-b549-fb71728b323a;/192.168.134.1
7	20,044	20,044	0	consumer-newgroup-1-ad4553b8-5520-4cc0-b549-fb71728b323a;/192.168.134.1
8	19,818	19,818	0	consumer-newgroup-1-d5bafe4b-5a28-474f-a52d-3fd3ab948bcc;/127.0.0.1
9	19,946	19,946	0	consumer-newgroup-1-d5bafe4b-5a28-474f-a52d-3fd3ab948bcc;/127.0.0.1

图 7　加入消费者后

协调者与消费者之间的心跳检测提示系统正在进行负载均衡，如图 8 所示，此时消息系统无法进行正常的消息传输功能直至负载均衡完成。

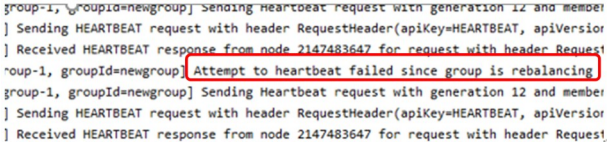


图 8　负载均衡过程中

3.3 实验结果分析

基于上述区块链网络系统设计与负载均衡算法和 Fabric 联盟链,每台机器上部署有 Kafka 节点,同时每台电脑上也部署了业务上链的智能合约。系统中的每一台电脑都作为一个 Kafka 节点加入区块链网络,本文采取测试信息在提交交易到区块间所有节点共识时间的方式来验证算法。

当发生宕机等异常情况时,在采用原算法的区块链环境中选择消费者进行上链操作,向区块链网络中传输信息,每当有一个节点宕机都会触发负载均衡,此时系统由于负载均衡而导致整体消息传输效率不高且容易崩溃,整个信息传输验证的时延增大。

为验证算法性能,本文分别采取 Kafka 原算法、本文算法和基于 Consumer 负载均衡的 CR 算法[20]相比较,在采用 Kafka 原算法时,运行数据上链业务,当 6 台机器中关闭 3 台机器后发现已不能上链,说明原算法并不能很好地应对恶劣的工作环境。在相同的区块链环境下分别测试 3 种算法的共识速度与 CPU 资源消耗情况,测试结果如图 9、10 所示。当 6 个消费者均工作时,3 种算法的信息共识速度相同;一位消费者宕机后,本文算法共识速度略快;当 3 位及 3 位以上的消费者宕机时,原算法和 CR 算法已经无法完成共识,但本文算法在 3 个消费者工作的情况下仍可完成共识。

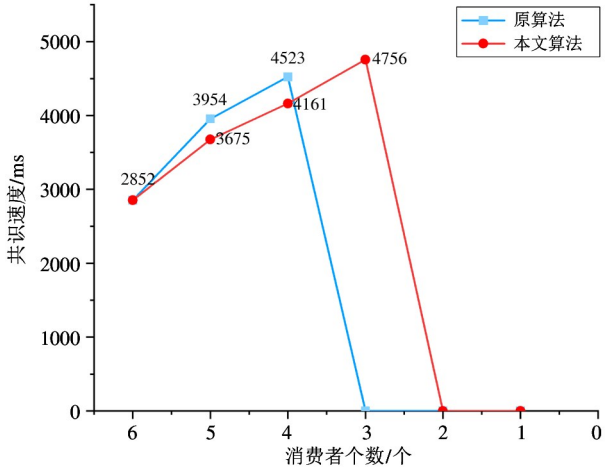


图 9 共识算法测试结果

图 9 记录了 3 种算法在相同环境下的测试结果。当采用系统自带的 Kafka 共识算法,也就是原算法时,在没有消费者宕机的情况下表现与本算法下一致;但是在宕机 1 位和 2 位消费者后,原算法及 CR 算法所在系统的共识速度不如本算法。5 个消费者环境下本算法的共识速度比其他 2 种算法快了 2% ~7% ,4 个消费者环境下本算法的共识速度比原算法快了 8% 。这是因为改进后的算法在消费者宕机时能迅速由协调者统一部署调整,调整后消费者和分区的对应情况更加合理且不需要其他消费者进行过多的负载均衡尝试,此时共识速度约 4.5 s。在 3 个消费者环境下原算法不能达成共识,但是本算法仍可以达成共识,其共识速度约 4.7 s,2 个和 1 个消费者环境下三者均不能达成共识,此时共识速度为 0。由此可见原算法和 CR 算法的稳定性不如本文提出的共识算法,当采用本文提出的共识算法时,系统整体共识速度得到了提升,安全性得到了稳固,能有效应对多消费者宕机的恶劣工作情况出现。算法精度方面,经仔细比对检查,信息完成共识前后一致,均未出现差错,所以 3 种算法的算法精度一致;由于及时避免了消息堆积情况的出现,本文算法在 3 位消费者情况下仍可以达成共识,所以本文算法在实时性方面优于其他 2 种算法。

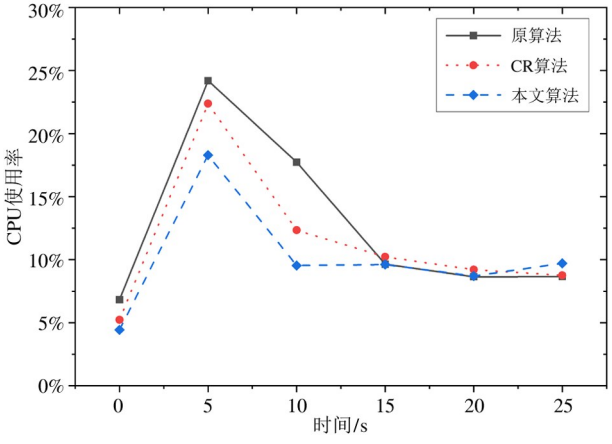


图 10 CPU 使用率测试结果

测试宕机情况下 3 种算法的系统 CPU 使用率,结果如图 10 所示。在 5 s 时宕机 1 位消费者,此时 3 种算法均消耗了更多的系统资源,造成了 5 s 时 CPU 使用率急剧上升的现象。但是本算法可以迅速调整分区与消费者的分配关系,在 5 s 内降低 CPU 使用率至 10% 左右,而原算法需要 5 ~10 s 的

时间来调整,且在 10 s 时原算法和 CR 算法消耗的系统资源更多,比本算法高出 5% 左右。可知在负载均衡时,本算法在快速降低系统资源消耗方面的性能高于其他 2 种算法,有效节省了能耗。

4 结 论

本文提出一种改进的 Kafka 负载均衡算法,以解决原算法中 Kafka 负载均衡过程不稳定、资源消耗过大的问题。在算法中设计了一种新的消费者检测策略,各个消费者不用时刻检测其消费者组中是否有宕机或者其他异常情况出现,每次负载均衡都由协调者统一管控,提升了消费者组的协调性;且每次负载均衡能根据实际情况动态调整消费者和分区的对应情况,所以系统中分区和消费者对应关系更加简洁高效。实验结果表明,改进后算法的共识速度比其他算法共识速度快 2% ~8%,能耗比其他算法降低 5%,且本算法能适应更恶劣的消费者宕机环境,在 6 个 Kafka 节点中 3 个宕机的情况下仍然能共识上链,在算法精度一致的情况下,提升了系统稳定性和实时性。值得一提的是,本算法已在机场群区块链网络中运行,通过上链行李和航班信息来验证算法可行性,结果表明本算法提高了机场群信息共享的安全性,保证了机场群间区块链网络的高效协同稳定运行。

参考文献

[1] WU H. Research proposal: reliability evaluation of the Apache Kafka streaming system[C]//2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). Berlin: IEEE, 2019:112-113.

[2] 蒋亮,唐紫珺. 基于 Flume、Kafka、HDFS 的多源数据采集系统[J]. 信息技术与信息化, 2021(6):115-117.

[3] 顾鑫,程淑玲. 一种物联网设施管理系统中的 Kafka 优化方法[J]. 计算机与数字工程, 2021,49(3):546-549.

[4] 魏引尚,崔恩伟. 基于区块链技术的煤矿安全信息管理模型优化构建[J]. 煤矿安全, 2021,52(10):252-255.

[5] 颜晓莲,章刚,邱晓红. Kafka 中改进型 Partition 过载优化算法[J]. 计算机技术与发展, 2020,30(12):88-91.

[6] TANG J. The application of blockchain in sunshine government affairs: student allocation system as an example [C]//2021 International Conference on Computer Technology and Media Convergence Design (CTMCD). Sanya: IEEE, 2021:183-186.

[7] 高子妍,王勇. 面向云服务的分布式消息系统负载均衡策略[J]. 计算机科学, 2020,47(S1):318-324.

[8] 车思阳. 基于 Kafka 的大容量实时预警数据汇集分发技术研究[D]. 成都:电子科技大学信息与通信工程学院, 2021:6-9.

[9] SGAMBELLURI A, PACINI A, PAOLUCCI F, et al. Reliable and scalable Kafka-based framework for optical network telemetry[J]. Journal of Optical Communications and Networking, 2021,13(10):42-52.

[10] CARNERO A, MARTÍN C, TORRES D R, et al. Managing and deploying distributed and deep neural models through Kafka-ML in the cloud-to-things continuum[J]. IEEE Access, 2021,9:125478-125495.

[11] SAIBHARATH S, MISHRA S, HOTA C. QoS driven task offloading and resource allocation at edge servers in RAN slicing[C]//2021 IEEE 18th Annual Consumer Communications and Networking Conference (CCNC). Las Vegas: IEEE, 2021:1-4.

[12] QIAN S Y, XU J W, Cao, et al. Fat topic: improving latency in content-based publish/subscribe systems on Apache Kafka[J]. Lecture Notes in Computer Science, 2021(1):547-558.

[13] TSENOS M, KALOGERAKI V. Dynamic rate control for topic-based pub/sub systems[C]//2021 22nd IEEE International Conference on Mobile Data Management (MDM). Toronto: IEEE, 2021:272-273.

[14] 王勇, 张跃. Kafka 与 HBase 在健康监测大数据平台中的应用研究[J]. 软件导刊, 2021, 20(4):188-193.

[15] LENOACH P, COSTAN A, BOUGÉ L. A performance evaluation of Apache Kafka in support of big data streaming applications[C]//2017 IEEE International Conference on Big Data (Big Data). Boston: IEEE, 2017: 4803-4806.

[16] 汪涛. Kafka 中 Broker 节点磁盘问题的故障处理方法[J]. 现代信息科技, 2020,4(13):148-150.

[17] HIRAMAN B R, CHAPTE V M, ABHIJEET C K. A study of Apache Kafka in big data stream processing[C]

//2018 International Conference on Information , Communication, Engineering and Technology. Pune: IEEE, 2018: 1-3.

[18] 孟月昊,林荣霞,冯文,等. 关于 Kafka 分区策略对系统性能影响的研究[J]. 计算机时代, 2020(11):11-15.

[19] DEDOUSIS D, ZACHEILAS N, KALOGERAKI V. On the fly load balancing to address hot topics in topic-based pub/sub systems[C]// 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). Vienna: IEEE, 2018: 76-86.

[20] 王郑合,王锋,邓辉,等. 一种优化的 Kafka 消费者/客户端负载均衡算法[J]. 计算机应用研究, 2017,34(8):2306-2309.

[21] HUA S, ZHANG S, PI B, et al. Reasonableness discussion and analysis for Hyperledger Fabric configuration[C]// 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). Toronto: IEEE, 2020:1-3.

[22] ALEKSIEVA V, VALCHANOV H, HULIYAN A. Implementation of smart-contract, based on Hyperledger Fabric blockchain[C]// 2020 21st International Symposium on Electrical Apparatus and Technologies (SIELA). Varna: IEEE, 2020:1-4.

Load balancing algorithm for Kafka consensus message transmission

SU Yuzhao, SUN Enchang, YANG Ruizhe, LI Meng, ZHANG Yanhua, SI Pengbo, ZHANG Hui

(Faculty of Information Technology, Beijing University of Technology, Beijing 100124)

Abstract

Kafka is a kind of message middleware with high throughput. However, this algorithm has the problem that load balancing can reduce the efficiency of consumer message processing. The coordinator in this algorithm optimizes and adjusts the number of consumers based on the correspondence between consumers and partitions according to different load balancing scenarios, and then updates the correspondence between consumers and partitions in a reverse order of business load. This algorithm prioritizes the business partitions with heavy load and improves the efficiency of message transmission. And build the Fabric consortium chain to verify the performance of the algorithm. Experimental results show that the proposed algorithm improves the consensus speed by 2% -7% when the central processing unit (CPU) resource consumption is 5% lower than that of other algorithm, and the consensus link can still be available even when three out of six Kafka nodes are out of service, which improves the efficiency and stability of the Kafka load balancing algorithm.

Key words: Kafka, load balancing, blockchain, consumer, optimization