

基于BFT-SMaRt的RBFT共识算法

郭 熙, 何 晋, 梁培利

(成都信息工程大学 区块链产业学院, 四川 成都 610225)

摘 要: 为提高区块链联盟链的共识效率和安全性, 提出一种基于随机化领导者选举与消息优化的BFT算法——RBFT。该算法通过引入一个伪随机函数实现了领导者节点的动态变化, 从而增强了共识的鲁棒性和抵抗拜占庭错误的能力。通过在无拜占庭错误的情况下减少一轮消息的数量, 从而降低网络通信开销和延迟, 提升了共识效率和吞吐量。在BFT-SMaRt开源库的基础上实现RBFT算法, 并在相同的网络环境和参数设置下进行安全性和性能分析实验。实验结果表明, 在相同情况下, RBFT算法比BFT-SMaRt算法具有更高的共识效率、更低的网络通信开销、更好的容错能力和可扩展性。

关键词: 区块链; 联盟链; 状态机复制; 拜占庭错误; 领导者选取

DOI: 10.11907/rjdk.241186

中图分类号: TP311

文献标识码: A

开放科学(资源服务)标识码(OSID):

文章编号: 1672-7800(2025)007-0111-08



RBFT Consensus Algorithm Based on BFT-SMaRt

GUO Xi, HE Jin, LIANG Peili

(College of Blockchain Industry, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: To improve the consensus efficiency and security of blockchain consortium chains, a BFT algorithm based on randomized leader election and message optimization, called RBFT, is proposed. This algorithm enhances the robustness of consensus and the ability to resist Byzantine errors by introducing a pseudo-random function to dynamically change the leader nodes. It also reduces the number of messages in one round without Byzantine errors, thereby reducing network communication overhead and latency, and improving consensus efficiency and throughput. Implement the RBFT algorithm based on the BFT-SMaRt open-source library, and conduct security and performance analysis experiments under the same network environment and parameter settings. The experimental results show that under the same conditions, the RBFT algorithm has higher consensus efficiency, lower network communication overhead, better fault tolerance, and scalability than the BFT-SMaRt algorithm.

Key Words: blockchain; consortium chain; state machine replication; Byzantine error; leader election

0 引言

区块链是一种去中心化、安全、不可篡改的分布式账本技术,最早是由中本聪于2008年提出,用作比特币的底层技术^[1]。区块链技术通过利用密码学、共识算法、激励机制等手段,在没有可信第三方的情况下使网络中的多个参与者能够就某些事实达成一致,并将其记录在一个公开、透明、持续增长的链式数据结构中,从而保证了数据的

完整性、一致性和可追溯性。该技术在数字货币、数据存储、数据鉴定、金融交易、供应链管理、智能合约、物联网等领域均有较好的应用前景^[2]。

共识算法是一种允许分布式系统中多个节点在不可靠的网络环境中达成一致决策的机制,是区块链技术的核心组成部分,保证了区块链的安全性和去中心化特性。不同的区块链系统可能采用不同的共识算法。根据参与者是否需要经过授权,区块链可以分为公有链、私有链和联盟链3种类型。公有链是指任何人都可以加入和退出的

收稿日期:2024-02-29

扫描二维码阅读全文:



基金项目:四川省高等教育人才培养质量和教学改革项目(JG2021-995)

作者简介: 郭熙(1998-),男,成都信息工程大学区块链产业学院硕士研究生,研究方向为区块链;何晋(1978-),男,成都信息工程大学区块链产业学院教授、硕士生导师,研究方向为智能治理、共识机制;梁培利(1989-),男,成都信息工程大学区块链产业学院系统架构师,研究方向为区块链金融、人工智能。

区块链网络,其数据对所有人开放,共识算法需要解决激励和作弊的问题,例如比特币使用的工作量证明(Proof of Work, PoW)算法和以太坊使用的权益证明(Proof of Stake, PoS)算法^[3-4]。私有链是指只有特定的组织或个人可以加入和退出的区块链网络,其数据只对内部成员开放,共识算法不需要解决拜占庭错误的问题。联盟链是指由多个经过授权的组织或个人共同维护的区块链网络,数据对联盟成员开放,其共识算法是基于拜占庭容错(Byzantine Fault Tolerance, BFT)的算法,如实用拜占庭容错(Practical Byzantine Fault Tolerance, PBFT)算法^[5-6]。然而, BFT 算法存在一些不足之处,如领导者节点的单点故障、消息通信的高开销等,可能会对区块链的共识效率和安全性产生影响。

1 相关研究

1.1 BFT-SMaRt

BFT-SMaRt 是 BFT 协议的一种高效、模块化的实现,通过优化和改进提高了 BFT 协议的性能和实用性,可用于提高区块链网络的可靠性和安全性。BFT-SMaRt 旨在用一个简洁而健壮的代码库实现 BFT 状态机复制,构筑可靠的服务,还能扩展和增加新的协议。其具有动态性,用户可根据自身需要在不停机的情况下快速进行共识网络拓扑结构的更新。BBFT-SMaRt 库具有良好的模块化设计与扩展性能,实现了共识算法与状态机复制、通信等模块的完全分离,使得用户可以更容易地定制和扩展自己的应用,也方便了与其他区块链平台的集成与测试。然而, BFT-SMaRt 也存在一些不足,如领导者节点的单点故障影响共识效率、在无拜占庭错误的情况下仍然需要 3 轮消息交换过程等。

1.2 BFT 类共识算法

BFT 类算法是一种经典的分布式系统容错技术,其可以在存在恶意节点的情况下保证系统的正确性和活性。该类算法的基本思想为通过多轮消息交换,使得系统中的大多数节点达成一致决定。例如, PBFT 算法已被广泛应用于区块链领域的 Hyperledger Fabric、Tendermint 等^[7-8]。然而, BFT 类算法也存在一些缺陷,如通信复杂度高、性能随节点增加而下降、领导者节点的固定性导致单点故障等。为解决这些问题,许多研究者对 BFT 类算法进行了改进,主要分为 3 类。

1.2.1 领导者选举机制改进

该类改进主要是为了解决 BFT 类算法中领导者节点的固定性问题,通过引入一些随机化或动态方法,使得领导者节点的身份不再固定,而是随着视图的变化而变化,从而增加了共识的鲁棒性和安全性。例如,文献[9]提出的 EZBFT (Efficient Zero-Knowledge Byzantine Fault Tolerance) 算法使用一个基于哈希链的随机数生成器,使得每

个节点在每个视图中都有一个随机的优先级值,然后根据优先级值的大小选择领导者节点。文献[10]提出的 Hot-Stuff 算法使用一个基于签名的领导者旋转机制,使得每个节点轮流担任领导者节点,并通过签名证明自己的身份。文献[11]提出的 BRaft 算法利用公钥加密算法(Rivest Shamir Adleman, RSA)签名解决拜占庭主节点篡改日志的问题,同时引入标识位解决拜占庭节点恶意响应主节点的问题。文献[12]提出的 RT-DPOS 共识机制在综合考虑节点的信任关系后采用两阶段策略选出代理节点集合,可以避免节点盲目投票。该类改进措施的优点是能够有效防止领导者节点的故障或恶意行为,提高了共识的安全性;缺点是可能会增加领导者节点的切换频率和开销,降低了共识效率。

1.2.2 消息交换过程改进

该类改进主要是为了解决 BFT 类算法中消息交换过程的复杂性和冗余性问题,通过减少消息数量或内容、改变消息发送方式降低网络通信的开销和延迟,提高共识效率和吞吐量。例如,文献[13]提出的 Zyzzyva 算法在无拜占庭错误的情况下仅使用两轮消息交换过程,减少了一轮消息的开销。文献[14]提出的 SBFT (Simplified Byzantine Fault Tolerance) 算法使用一种基于梅克尔树的摘要机制,使得每个节点只需要发送和接收摘要信息,而不是完整的提议内容,从而减少了消息量。文献[15]提出的 F-PBFT (Federated Practical Byzantine Fault Tolerance) 算法通过优化 PBFT 一致性协议减少节点间的通信时间,提高了共识效率。文献[16]提出基于信誉机制的改进 PBFT 共识算法,将参与节点分为收集器节点和普通节点,无拜占庭节点时通过增加收集的投票数量减少一次节点间的通信,从而快速达成共识。该类改进措施的优点是能够有效降低网络通信的开销和延迟,提高了共识效率和吞吐量;缺点是可能会降低共识的容错能力和安全性,或增加恢复机制的复杂性。

1.2.3 共识模型改进

该类改进主要是为了解决 PBFT 算法中共识模型的局限性问题,通过引入一些新的概念或机制,使得共识模型更加灵活和具有适应性,从而适应不同场景的需求。例如,文献[17]提出的 BFT-Mencius 算法使用一个基于多 Paxos 的共识模型^[18],使每个节点都可以提出提议,并通过多个并行的共识实例达成一致。文献[19]提出的 RS-PBFT (Redundant Sharding Practical Byzantine Fault Tolerance) 算法通过对共识过程网络进行分组的形式减少节点间的通信开销,提升了共识效率。文献[20]提出的 CRPBFT (Certificate-based Reputation Practical Byzantine Fault Tolerance) 共识算法通过一致性哈希对所有节点进行分组,通过计算节点阈值划分层次,在此基础上增加节点动态变化机制,提高了 PBFT 算法的动态性。该类改进措施的优点是能够有效提高共识的灵活性和适应性,适应不

同场景的需求;缺点是可能会增加共识的复杂性和开销,或降低共识效率和吞吐量。

为解决以上问题,本文提出一种基于随机化领导者选举与消息优化的BFT算法,简称为改进的拜占庭容错算法(Revised Byzantine Fault Tolerance, RBFT)。RBFT算法在BFT-SMaRt开源库的基础上实现,继承了一些特性,如动态配置、多种通信模式、多种应用接口等,同时通过改变领导者节点的选取方式与优化共识阶段的消息交换过程,既保证了共识的安全性和鲁棒性,又提高了共识的效率和吞吐量^[21]。RBFT算法引入一个伪随机函数,实现了领导者节点的动态变化,从而增强了共识的鲁棒性和抵抗拜占庭错误能力。相比之下,BRaft算法是利用RSA公钥签名算法解决领导者节点作恶问题,而RBFT算法则是从源头上改变领导者节点选取方式,从而提高共识算法的安全性。RT-DPOS共识机制通过代理节点集合的方式选取领导者节点,同样降低了领导者节点的作恶可能,但与RBFT相比增加了领导者节点的切换频率,降低了共识效率。RBFT与EZBFT、HotStuff等领导者选举类算法有一定的相似性,但也有一些不同之处。首先,RBFT算法的领导者选举是基于每个区块的哈希值和当前视图编号,而不是基于客户端的签名或Paxos提议,这样可以避免客户端或提议者的影响或攻击;其次,RBFT算法的领导者选举是随机的,而不是轮询的,这样可以增加领导者节点的不可预测性,从而防止恶意节点的针对性攻击或串谋。RBFT算法还通过无拜占庭错误的情况下减少一轮消息的数量,从而降低网络通信的开销和延迟,提升共识效率和吞吐量。这一点与Zyzzyva算法有一定的相似性,但也有一些不同之处。RBFT算法的消息优化是基于预准备和准备两轮消息,而不是基于准备和提交两轮消息,这样可以更早地达成共识,从而缩短共识时间。F-PBFT算法通过优化PBFT一致性协议减少节点间的通信时间,提高了共识效率,虽然同样优化了节点间的通信时间,但RBFT能满足节点动态加入的需求。RBFT算法的消息优化是基于无拜占庭错误的假设,在发生拜占庭错误的情况下通过新一轮消息传递仍能达成一致,且能满足节点动态变化要求,这样可以更好地适应实际的网络环境,从而提高共识的稳定性。CRPBFT共识算法基于节点的信誉度分组机制,相较PBFT算法共识节点的可靠度更高,与RBFT同样都支持节点动态变化,但由于需要计算节点的信誉值进行分组,相较RBFT时延更长。综上所述,RBFT算法在保证共识的安全性和活性的同时也提高了共识的效率和容错能力,是一种适合联盟链场景的共识算法。

2 理论基础

2.1 BFT-SMR方案

BFT-SMaRt基于BFT和状态机复制理论(State Ma-

chine Replication, SMR),通过smart(即SMaRt)的概念改进了BFT-SMR方案^[22]。BFT-SMR方案需要较多通信步骤。在SMR模型中,任意数量的客户端进程向一组副本进程(分布式集群)发出命令,这些副本实现一个有状态的服务,例如一个账户的余额。该服务在处理客户端命令后更改其状态,并向发出这些命令的客户端发送响应。该技术的目标是使每个副本上的状态保持一致,从而使服务在每个副本上完整并准确地被复制。为实现SMR的运行流程,需要满足:①两个正确的副本将操作应用于状态,它们均获得一个相同的状态;②任意两个正确的副本均从相同的初始状态开始;③任意两个正确的副本均执行相同的操作序列;④正确客户端发出的请求能被执行。例如,PBFT算法包含客户端请求、节点三阶段共识(包含3个步骤)、节点回复5个步骤,适用于联盟链环境,可以为交易排序,进而达成顺序共识,但没有状态校验,在不可信的环境中需要有状态校验作为双重检查。因此,BFT-SMR方案需要客户端请求、节点三阶段共识、三阶段状态校验7个步骤。基于可靠的广播信道,SMR过程包含客户端广播请求、节点相互回应、节点同步完成3个步骤;然后执行与PBFT类似的三阶段共识;最后再回复客户端。具体见图1。为了优化通信步骤,BFT-SMaRt还提出模块化SMR(MOD-SMaRt)的概念,其使用一种叫作已验证可证明共识(VP-Consensus)的算法。

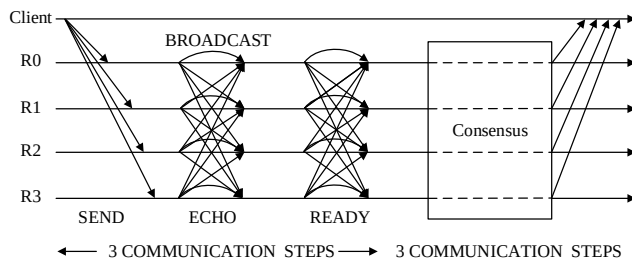


Fig. 1 BFT-SMR consensus steps

图1 BFT-SMR共识步骤

2.2 VP-Consensus共识算法

VP-Consensus是一种兼具验证属性与可证明特性的共识算法,其核心特征为:①验证属性。当协议接受某个操作对特定值的修改时,要求所有节点维护的对应值必须满足该修改操作的约束条件,且能通过分布式验证流程;②可证明特性。基于零知识证明机制,协议将为每个被采纳的值生成加密证明证据,可被任何验证方独立验证其正确性。协议实现需满足3点基础原语:①VP-Propose(i, a, b, n)。在共识实例 i 中,由领导者进程 a 发起对值 n 的提案,该提案必须基于预定义操作 b 执行状态转换;②VP-Decide(i, n, d)。当共识实例 i 中达成对值 n 的采纳决议时自动触发, d 表示最终确定的决策状态;③VP-Timeout(i, a)。共识实例 i 出现超时事件时激活容错机制,指定新领导者进程 a 接管共识流程。协议架构采用改进型BFT共识

范式,在保留传统BFT共识核心特征的同时引入动态领导者切换机制。超时事件被触发后,各节点基于本地决策算法自主选举新领导者,无需额外协调,所有正确运行的进程将在同一共识实例*i*中调用VP-Propose方法,并严格遵循预定义的操作*b*执行。

2.3 MOD-SMaRt

MOD-SMaRt是一种模块化的SMR方案,分为3个子算法。具体见图2。

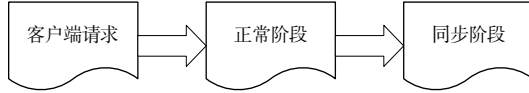


Fig. 2 MOD-SMaRt consensus steps

图2 MOD-SMaRt 共识步骤

MOD-SMaRt模块化方案建立在默认可靠的点对点通信信道以及VP-Consensus的基础之上。VP-Consensus在共识实例*i*中执行一个正确的操作序列,这个操作序列也会在该共识实例中的其他副本中正确执行。如图3所示,在第二阶段的正常阶段中,一个正确的客户端发送请求到所有副本,一个共识实例会建立整个执行顺序并执行操作,然后回复给客户端。

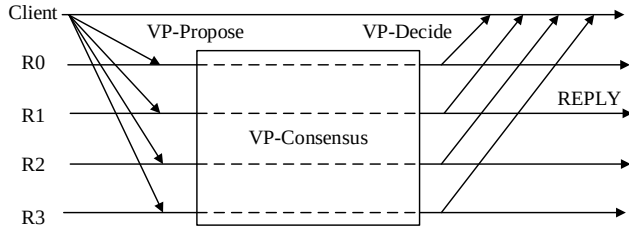


Fig. 3 MOD-SMaRt normal stage

图3 MOD-SMaRt 正常阶段

在第三阶段的同步阶段中,第二次触发消息超时将启动消息交换流程,启动时同步运行VP-Consensus。具体流程见图4,其中虚线部分表示VP-Consensus协议的消息。

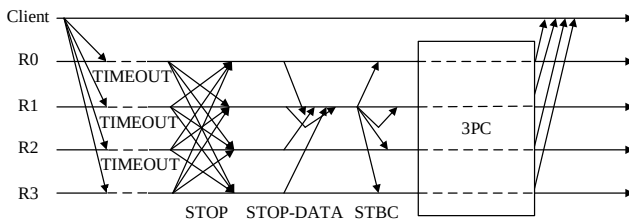


Fig. 4 MOD-SMaRt message flow

图4 MOD-SMaRt 消息流程

MOD-SMaRt的具体步骤为:①初始领导者为节点R0,客户端可以向任意一个节点定向发送消息,收到消息的节点会为消息排序,非领导者一定会处理超时,那么它就会将原消息转发给其他节点,可以确保领导者节点收到原消息并排序;②如果同一个请求有第二个超时,说明领导者排序失败,则更换领导者。该节点会暂停本地所有倒计时器,然后启动同步阶段,领导者默认更换为下一个节点

R1,并广播STOP消息。某节点统计本地收到STOP消息的节点数量达到确认数,即至少 $n/2+1$ (CFT), $2/3n+1$ (BFT),则本地确定领导者更换成功,并向新领导者发送STOP-DATA消息;③新领导者接收该消息达到确认数,则向其他节点发送SYNC消息,包含原消息决策;④节点接收到SYNC消息后,会将原消息执行相同的计算,从而校验领导者的信息是否正确,然后向外广播READ消息;⑤节点本地收集READ消息到确认数,则发送COLLECT消息给领导者;⑥领导者接收COLLECT消息到确认数,则发送PROPOSE消息,表示原消息状态已同步;⑦节点接收领导者发来的PROPOSE消息,完成本地对于该消息的处理。

如果原领导者没有执行异常,则不需要更换,只需要READ-COLLECT-PROPOSE(⑤⑥⑦)3个步骤即可完成对于消息的共识处理。以下为BFT-SMaRt共识流程的伪代码:

```

1.function ConsensusProcess
2.Leader = R0
3.Client -> Node: Message
4.Node: Sort(Message)
5.IF Node != Leader AND Timeout
6.Node -> Other Nodes: Message
7.END IF
8.IF Node == Leader
9.Leader: Receive and Sort(Message)
10.END IF
11.IF Same Request AND Second Timeout
12.Node: Pause All Timers
13.Node: Start Sync Phase
14.Leader = R1
15.Node -> All Nodes: STOP
16.Node: Count STOP Messages
17.IF Count >= n/2 + 1 (CFT) OR Count >= 2/3n + 1 (BFT)
18.Node: Confirm Leader Change
19.Node -> Leader: STOP-DATA
20.END IF
21.END IF
22.IF Node == Leader
23.Leader: Receive STOP-DATA Messages
24.IF Count >= n/2 + 1 (CFT) OR Count >= 2/3n + 1 (BFT)
25.Leader -> Other Nodes: SYNC(Message Decision)
26.END IF
27.END IF
28.Node: Receive SYNC Message
29.Node: Verify Leader's Information
30.Node -> All Nodes: READ
31.Node: Collect READ Messages
32.IF Count >= n/2 + 1 (CFT) OR Count >= 2/3n + 1 (BFT)
33.Node -> Leader: COLLECT
34.END IF

```

3 RBFT共识算法设计与实现

RBFT算法是针对BFT-SMaRt存在的两点不足提出,即领导者节点的单点故障导致共识效率降低且影响系统的安全性以及在无拜占庭错误情况下仍需要繁杂的消息交换过程而造成不必要的通信开销。RBFT算法通过随机化领导者选举机制增加系统的安全性和鲁棒性,通过只使用READ消息的优化协议在无拜占庭错误的情况下减少了消息交换过程,降低了通信开销,大大提高了共识效率。此外,该算法在发生拜占庭错误的情况下需要回退到传统的三轮消息交换过程达成共识,相较BFT-SMaRt吞吐量下降,共识效率降低,即在增强安全性的同时牺牲了系统活性。

3.1 随机化领导者选举机制

BFT-SMaRt的领导者节点选取方式是固定的,即网络中的第一个节点(编号为0)作为领导者节点,负责对客户端的请求进行排序与提议,当需要进行视图更换时选择(轮换法)下一个replica节点作为主节点,视图编号为 $v+1$ 。RBFT引入一个随机化的领导者选举机制,使每个节点根据自身的视图编号和节点编号生成一个优先级值,并将其广播给其他节点。每个节点收到所有节点的优先级值后选择最高优先级值对应的节点作为领导者节点。如此以来,领导者节点的身份不是固定的,而是随着视图的变化而变化。该机制不仅可以防止领导者节点固定化,而且确保了在视图变更时能够平衡网络负载,减少了单点故障风险。此外,随机化机制增强了系统对网络动态变化的适应性,使得每个节点都有成为领导者的机会,从而提高了整个网络的灵活性和公平性。具体流程为:①节点在初始化时使用一个伪随机函数,例如SHA-256,根据自身节点编号生成一个随机种子,并将其存储在本地;②节点使用一个伪随机函数根据自身随机种子和视图编号生成一个优先级值,并将其存储在本地;③节点将自身节点编号和优先级值封装成一个优先级消息(PriorityMessage),并广播给其他节点;④节点收集其他节点的优先级消息,直到收到至少 $3f+1$ 个一致的优先级消息,其中 f 为网络中允许的最大拜占庭错误节点数;⑤节点根据收到的优先级消息选择最高优先级值对应的节点编号作为领导者节点,并将其存储在本地。

3.2 只使用READ消息的优化协议

在BFT-SMaRt中,消息交换过程分为READ、COLLECT和PROPOSE 3个阶段。在READ阶段,节点本地收集 $2f+1$ 条READ消息,发送COLLECT消息给leader。在COLLECT阶段,leader接收 $2f+1$ 条COLLECT消息,发送PROPOSE消息,代表该原消息状态已同步。在PROPOSE阶段,节点接收leader发来的PROPOSE消息,完成本地对于该消息的处理。若未接收到要求数量的消息则说明

leader执行异常,需要更换,启动视图变更,更换主节点。BFT-SMaRt 3轮消息交换过程中消息数量和内容过多,导致网络通信开销和延迟增加,降低了共识效率。事实上,COLLECT消息的作用不明显,仅用于确认READ消息的有效性,而不是用于传递新的信息,增加了消息的冗余性。在无拜占庭错误的情况下,COLLECT消息是不必要的,因为READ消息已经足够保证提议的一致性,不需要额外的确认,没必要浪费消息资源。

因此,在无拜占庭错误的理想状态下,RBFT仅使用READ消息,省略了COLLECT消息,简化了共识过程。这种优化不仅提高了共识的效率和吞吐量,而且保持了系统的容错能力。在出现拜占庭错误时,系统可以回退到使用COLLECT消息的步骤,从而确保共识的可靠性。具体流程为:①客户端向所有节点发送请求消息,包含请求的内容和序列号;②每个节点收到请求后,将其转发给领导者节点,领导者节点负责对请求进行排序与提议;③领导者节点根据收到的请求生成一个提议消息,包含提议内容、视图编号、序列号和摘要,然后将提议消息发送给所有节点;④每个节点收到提议后验证其合法性,如果通过,则向所有节点发送一个READ消息,包含提议的摘要、视图编号、序列号和自己的签名,同时执行提议的内容,并将执行结果缓存起来;⑤每个节点收到至少 $3f+1$ 个一致的READ消息后,认为提议已经被写入,然后将执行结果从缓存中取出,并发送给客户端;⑥客户端收到相同的执行结果后,认为共识已经完成,然后返回结果;⑦如果有拜占庭错误发生,或者客户端没有收到足够的执行结果,或者节点没有收到足够的READ消息,则按照PBFT算法的视图切换和恢复机制来处理,即使用接受消息来达成共识,并更换领导者节点。

只使用read消息的优化协议能降低网络通信的开销和延迟,简化共识的逻辑和实现,亦能保证共识的容错能力和安全性。以下为RBFT共识流程的伪代码:

```

1. function ConsensusProcess
2. Leader = R0
3. Client -> All Nodes: Request(Content, Sequence)
4. Node: Forward(Request) to Leader
5. IF Node == Leader
6. Leader: Receive and Sort(Request)
7. Leader: Generate Proposal(Content, View, Sequence, Digest)
8. Leader -> All Nodes: Proposal
9. END IF
10. Node: Receive and Verify(Proposal)
11. IF Proposal is valid
12. Node: Execute(Content) and Cache(Result)
13. Node -> All Nodes: Read(Digest, View, Sequence, Signature)
14. END IF
15. Node: Receive and Count(Read)

```

```

16. IF Count >= 3f + 1
17. Node: Fetch(Result) from Cache
18. Node -> Client: Result
19. END IF
20. Client: Receive and Count(Result)
21. IF Count >= 1
22. Client: Return(Result)
23. END IF
24. IF Byzantine fault occurs OR Client does not receive enough
Result OR Node does not receive enough Read
25. Node: Follow PBFT view change and recovery mechanism
26. Node: Use Accept messages to reach consensus
27. Node: Change Leader
28. END IF

```

4 实验方法与结果分析

4.1 实验环境与参数设置

服务器配置为4核CPU,8 GB内存,1 Gbps网络带宽,运行Ubuntu 18.04操作系统。使用BFT-SMaRt开源库,改进方案在其基础上修改代码,使用Java语言编写。使用BFT-SMaRt提供的测试客户端和服务端,分别运行原始的BFT-SMaRt与RBFT算法。节点之间的网络时延在5~10 ms范围内;网络带宽设置为100 MB(假设每条请求消息平均100 B大小)。消息发送间隔为1 ms网络中的共识次数为50,即每个算法运行50次共识,测量每次共识的延迟,然后计算平均值和标准差。实验中假设无拜占庭错误的场景节点总数分别为4、7、22、100个,有拜占庭错误的情况下节点总数设置为22个,拜占庭节点数设置为0~7个。

4.2 评价指标

改进方案的性能评价指标包括:①共识延迟。指从客户端发送请求到客户端收到执行结果的时间;②共识吞吐量。指单位时间内完成的共识数量;③共识通信开销。指完成一个共识所需要的消息数量和大小。

4.3 实验结果与分析

4.3.1 共识延迟

共识延迟主要取决于提议消息和READ消息的传输时间。假设提议消息的大小为 M ,READ消息的大小为 W ,每个节点的网络带宽为 B ,每个节点的处理时间为 T ,每个消息的传输时间为 D ,那么共识延迟可近似表示为 $L=T+M/B+D+W/B+D$ 。如图5所示,RBFT算法在无拜占庭错误的情况下只使用了两轮消息交换过程,共识延迟比原始的BFT-SMaRt更少,而在有拜占庭错误的情况下共识延迟与BFT-SMaRt相当。

4.3.2 共识吞吐量

共识吞吐量主要取决于提议消息和READ消息的处理能力。假设每个节点的计算能力为 C ,每个节点的网络

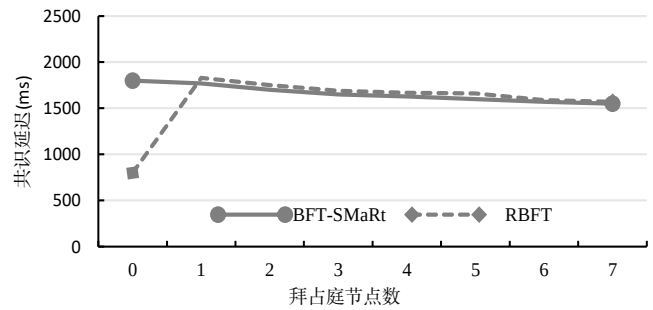


Fig. 5 Consensus delay

图5 共识延迟

带宽为 B ,提议消息的大小为 M ,READ消息的大小为 W ,共识吞吐量可近似表示为 $Q=C/(M+W)$ 。如图6所示,在无拜占庭错误的情况下,RBFT共识算法的吞吐量优于BFT-SMaRt,随着节点数的增多,BFT类共识算法的吞吐量均会变低,且节点数在超过100时,BFT类算法性能会急剧下降,因此不作统计。如图7所示,在有拜占庭错误的情况下,由于RBFT使用的领导者选取方法略复杂于BFT-SMaRt,BFT-SMaRt的吞吐量略优于RBFT。

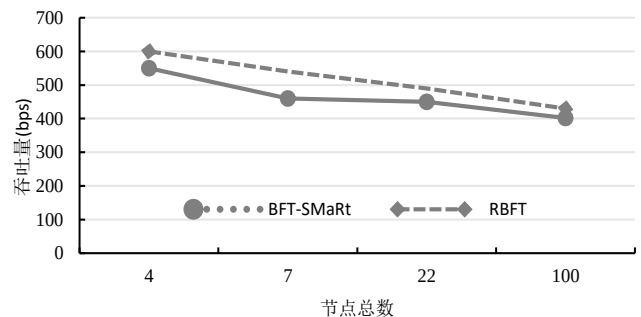


Fig. 6 Consensus throughput without Byzantine errors

图6 无拜占庭错误情况下的共识吞吐量

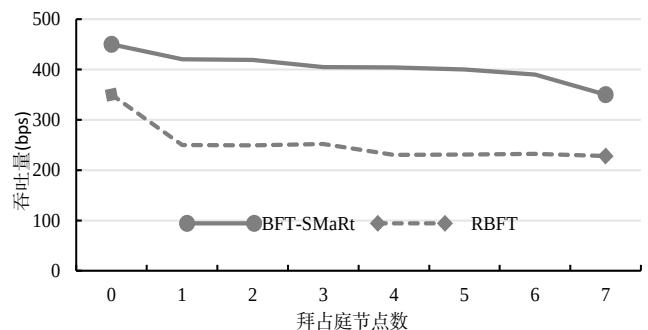


Fig. 7 Consensus throughput under Byzantine error conditions

图7 有拜占庭错误情况下的共识吞吐量

4.3.3 共识通信开销

共识的通信开销主要取决于提议消息和READ消息的数量和大小。假设网络中的节点总数为 N ,提议消息的大小为 M ,READ消息的大小为 W ,那么共识的通信开销可近似表示为 $K=M+N*W$ 。如图8所示,在无拜占庭错误的

情况下,BFT-SMaRt的消息量与节点数成平方级关系,而RBFT的消息量与节点数成线性关系,RBFT大幅降低了通信开销。如图9所示,在有拜占庭错误的情况下,BFT-SMaRt与RBFT消息量相当,且消息量随着拜占庭节点数的增多而下降,这是由于出现拜占庭错误时,拜占庭节点的消息会被忽略。

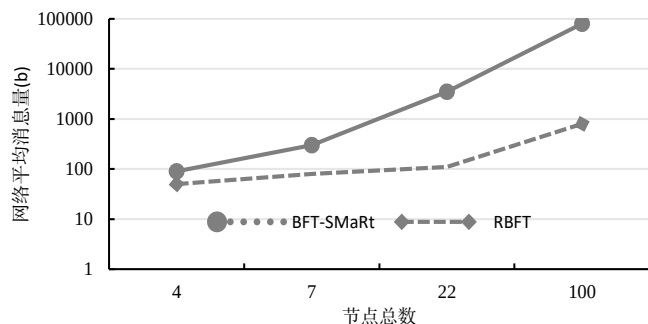


Fig. 8 Consensus communication overhead without Byzantine errors

图8 无拜占庭错误情况下的共识通信开销

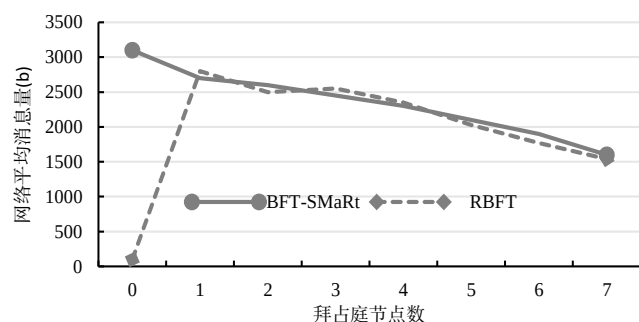


Fig. 9 Consensus communication overhead under Byzantine error conditions

图9 有拜占庭错误情况下的共识通信开销

4.3.4 安全性分析

RBFT算法的安全性体现在其能够抵抗拜占庭错误。其通过使用伪随机函数来动态选择领导者节点,不仅增加了攻击者预测下一个领导者的难度,而且提高了系统安全性。RBFT算法在无拜占庭错误的情况下减少了消息数量,在攻击者尝试进行拜占庭错误时,系统的复杂性降低,从而减少了潜在的攻击面。在有拜占庭错误的情况下,RBFT算法能够通过视图转换机制有效地更换受到攻击的领导者节点,确保系统正常运行。即使在最坏的情况下,RBFT算法也能保证系统的安全性,不会因为单个节点的恶意行为而崩溃。

4.3.5 活性分析

RBFT算法的活性是指能够持续处理请求并达成共识的能力。该算法通过优化消息传递过程,在无拜占庭错误的情况下显著提高了共识效率,在有拜占庭错误的情况下亦能通过视图转换机制迅速恢复,保证系统稳定运行。然而,在有拜占庭错误的情况下RBFT算法活性会有所降低,

这是由于领导者节点选取方式发生变化,以及触发视图转换机制退化为三阶段共识,系统吞吐量随之降低。

5 结语

针对BFT-SMaRt在异步网络环境下通信开销大的问题,本文提出一种基于随机化领导者选举与消息优化的BFT算法改进方案——RBFT。改进方案设计了一个随机化领导者选举机制,增强了算法的安全性和鲁棒性;同时采用一个只使用READ消息的优化协议,减少了一轮消息的数量和大小,从而降低了算法的通信开销和延迟。在云服务器集群上进行的实验结果表明,在相同的网络环境和参数下,RBFT算法的共识时长、吞吐量、通信开销等指标均优于原始的BFT-SMaRt,在联盟链特定场景中具有一定的应用价值,为异步网络环境下的分布式共识提供了一种新思路,也为利用随机化和优化技术手段提升分布式系统的性能和安全性提供了一种新的尝试。

未来还需要完善一些问题,例如RBFT算法只考虑了无拜占庭错误的情况,未对有拜占庭错误的情况进行处理,在有拜占庭错误的情况下如何优化共识算法是一个值得思考的问题。此外,RBFT使用伪随机函数来生成节点的优先级值,未对不同伪随机函数的性能和安全性进行比较与评估,在实际应用中如何选择合适的伪随机函数以达到最佳效果值得深入研究。

参考文献:

- [1] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system [EB/OL]. <https://bitcoin.org/en/bitcoin-paper>.
- [2] TAN M S, YANG J, DING L, et al. Review of consensus mechanism of blockchain[J]. Computer Engineering, 2020, 46(12): 1-11.
谭敏生,杨杰,丁琳,等. 区块链共识机制综述[J]. 计算机工程, 2020, 46(12): 1-11.
- [3] LI S N, YANG Z, TESSONE C J. Proof-of-work cryptocurrency mining: a statistical approach to fairness[C]//Chongqing: 2020 IEEE/CIC International Conference on Communications in China, 2020.
- [4] KING S, NADAL S. PPCoin: peer-to-peer crypto-currency with proof-of-stake [EB/OL]. <https://people.cs.georgetown.edu/~clay/classes/fall2017/835/papers/peercoin-paper.pdf>.
- [5] COWLING J, MYERS D, LISKOV B, et al. HQ replication: a hybrid quorum protocol for Byzantine fault tolerance[C]//Seattle: 7th Symposium on Operating Systems Design and Implementation, 2006.
- [6] HUANG Y X. Optimization of blockchain PBFT consensus algorithm applied to supply chain [J/OL]. Computer System Applications, 2024-02-28. <https://doi.org/10.15888/j.cnki.csa.009472>.
黄宇翔. 应用于供应链的区块链PBFT共识算法优化[J/OL]. 计算机系统应用, 2024-02-28. <https://doi.org/10.15888/j.cnki.csa.009472>.
- [7] ANDROULAKI E, MANEVICH Y, MURALIDHARAN S, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains[DB/OL]. <https://arxiv.org/abs/1801.10228>.

- [8] BUCHMAN E, KWON J, MILOSEVIC Z. The latest gossip on BFT consensus[DB/OL]. <https://arxiv.org/abs/1807.04938>.
- [9] ARUN B, PELUSO S, RAVINDRAN B. ezBFT: decentralizing Byzantine fault-tolerant state machine replication[C]//Dallas: IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019.
- [10] YIN M, MALKHI D, REITER M K, et al. HotStuff: BFT consensus in the lens of blockchain[DB/OL]. <https://arxiv.org/abs/1803.05069>.
- [11] BAI S W, DA H Y, GAO G M, et al. BRaft consensus algorithm based on consortium chain PBFT[J]. Software Guide, 2023, 22(9): 132-137.
白尚旺, 达泓宇, 高改梅, 等. 基于联盟链PBFT的BRaft共识算法[J]. 软件导刊, 2023, 22(9): 132-137.
- [12] YE F, JIANG L Y. Improved DPoS consensus mechanism based on hybrid trust model[J]. Software Guide, 2024, 23(4): 141-149.
叶飞, 蒋凌云. 基于混合信任模型的DPoS共识机制改进研究[J]. 软件导刊, 2024, 23(4): 141-149.
- [13] KOTLA R, ALVISI L, DAHLIN M, et al. Zyzzyva[J]. ACM Transactions on Computer Systems, 2009, 27(4): 1-39.
- [14] GUETA G, ABRAHAM I, GROSSMAN S, et al. SBFT: a scalable and decentralized trust infrastructure [C]//Proceedings of the 2019 USENIX Annual Technical Conference, 2019: 995-1008.
- [15] SHI W Z, CHENG Y Q, LIU X F, et al. An improved PBFT based online tourism review service model[J]. Electronic Design Engineering, 2024, 32(4): 89-93.
史文帜, 程永强, 刘晓峰, 等. 基于改进PBFT的在线旅游点评服务模式[J]. 电子设计工程, 2024, 32(4): 89-93.
- [16] LI J J, ZHANG J Q. Improved PBFT consensus algorithm based on reputation mechanism [J/OL]. Computer Application Research, 2024-02-28. <https://doi.org/10.19734/j.issn.1001-3695.2023.11.0517>.
- 李俊吉, 张佳琦. 基于信誉机制的改进PBFT共识算法[J/OL]. 计算机应用研究, 2024-02-28. <https://doi.org/10.19734/j.issn.1001-3695.2023.11.0517>.
- [17] ATTIYA H, BORRAN F, HUTLE M, et al. Structured derivation of semi-synchronous algorithms[C]//International Symposium on Distributed Computing, 2011: 374-388.
- [18] LAMPORT L. Paxos made simple [EB/OL]. <https://lamport.azurewebsites.net/pubs/paxos-simple.pdf>.
- [19] ZHOU C H, LIU W L, LIANG F, et al. An improved PBFT algorithm based on anti prediction grouping[J]. Computer and Digital Engineering, 2023, 51(9): 1945-1948, 1978.
周昌慧, 刘万里, 梁峰, 等. 基于防预测分组的PBFT算法改进方案[J]. 计算机与数字工程, 2023, 51(9): 1945-1948, 1978.
- [20] ZHAI S P, HUO Y Y, YANG R, et al. Improvement of PBFT algorithm based on consistent hashing and random selection [J]. Computer Engineering and Applications, 2024, 60(12): 294-302.
翟社平, 霍媛媛, 杨锐, 等. 基于一致性哈希和随机选取的PBFT算法改进[J]. 计算机工程与应用, 2024, 60(12): 294-302.
- [21] BESSANI A, SOUSA J, ALCHIERI E E P. State machine replication for the masses with BFT-SMART[C]//Atlanta: 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014.
- [22] WANG J, LI B H, WU J J, et al. A distributed consistency protocol supporting out of order submission of logs[J]. Journal of Zhejiang University (Engineering Edition), 2023, 57(2): 320-329.
王进, 李博涵, 吴佳骏, 等. 支持日志乱序提交的分布式一致性协议[J]. 浙江大学学报(工学版), 2023, 57(2): 320-329.

(责任编辑: 尹晨茹)