计算机科学
COMPUTER SCIENCE

支持分片内多轮PBFT验证算法的状态同步方案

高冬雪, 李志淮, 段培培, 陈玉华

引用本文

高冬雪, 李志淮, 段培培, 陈玉华.支持分片内多轮PBFT验证算法的状态同步方案[J]. 计算机科学, 2022, 49(11A): 211000125-7.

GAO Dong-xue, LI Zhi-huai, DUAN Pei-pei, CHEN Yu-hua. State Synchronization Scheme Supporting Multiple Rounds of PBFT VerificationAlgorithm in Sharding [J]. Computer Science, 2022, 49(11A): 211000125-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000125

支持分片内多轮 PBFT 验证算法的状态同步方案

高冬雪　李志淮　段培培　陈玉华
大连海事大学信息科学技术学院　辽宁 大连 116026
(1720299464@qq.com)

摘　要　分片是区块链扩容的链上解决方案之一,其中的状态分片可以在不降低安全性的前提下解决公链可扩展性问题。在状态分片中,每个分片只存储部分状态,若采用实用拜占庭容错(Practical Byzantine Fault Tolerance protocol,PBFT)共识算法,即使总体拜占庭节点比例不超过1/3,随机分配到单个分片内的拜占庭节点的比例也存在一定概率超过1/3,导致分片共识失效。因此分片内节点需要定期在不同分片间重新分配,并采用时隙较小的多轮 PBFT 验证算法可以有效解决这个问题。但是无状态节点难以有效工作,新加入的节点需要同步分片的状态。为此,提出了适应于多轮 PBFT 验证共识算法的基于候补节点序列的状态同步方案,完成状态同步的节点先进入候补节点序列,为每一轮 PBFT 共识验证提供不同的共识验证节点。同时,在状态同步过程中,节点根据其历史行为记录获得相应的积分,后续可根据节点积分对算法进行优化。实验结果表明,所提方案在解决状态同步问题的同时,提高了节点共识验证效率,提升了系统的吞吐量。

关键词:区块链;状态分片;状态同步;多轮验证;PBFT
中图法分类号　TP311

State Synchronization Scheme Supporting Multiple Rounds of PBFT Verification Algorithm in Sharding

GAO Dong-xue,LI Zhi-huai,DUAN Pei-pei and CHEN Yu-hua
School of Information Science and Technology,Dalian Maritime University,Dalian,Liaoning 116026,China

Abstract　Sharding is one of the on-chain solutions for blockchain scalability. State sharding can solve the scalability problem of the public chain without reducing security. Each state sharding only maintains a part of the state. There is a certain probability that the proportion of Byzantine nodes will exceed one third in a single sharding,even if the probability of Byzantine nodes is no more than one third in all nodes with PBFT consensus algorithm,resulting in the failure of verifying the consensus. Therefore,the nodes in the sharding need to be reconfigured periodically,and multi-round PBFT consensus verification algorithm with small time slots can effectively solve this problem. However,stateless nodes cannot work effectively,and new nodes need to synchronize the state of the sharding. The state synchronization scheme based on candidate nodes queue for multi-round PBFT consensus verification algorithm is proposed to solve this problem. Nodes that are in synchronized state first enter the queue of candidate nodes,and different candidate nodes are provided for each round of PBFT consensus verification. At the same time,a node gets a corresponding credits based on its historical behavior record during status synchronization to help optimizing the subsequent algorithm. Finally,experiment shows that the proposed scheme not only solves the problem of state synchronization,but also improves the efficiency of consensus verification and the throughput of the system.

Keywords　Blockchain,State sharding,State synchronization,Multiple rounds verification,PBFT

1　引言

2008年,中本聪在《比特币:一个点对点电子现金系统》[1]一文中首次提出了比特币的概念,比特币的核心技术便是区块链技术。2015年,以太坊[2]的出现使得区块链技术被更多人了解和研究。区块链技术被视为继人工智能、物联网之后的又一项颠覆性技术,受到了全世界的高度关注。

然而,区块链技术无法同时满足去中心化、安全性以及可拓展性。比特币和以太坊都是优先考虑去中心化和安全性,牺牲可拓展性[3],可拓展性已经成为区块链应用落地的最大瓶颈。若想满足大规模的应用,需要找到一种方法来提高可拓展性。现有的扩容技术包括分片技术[4-6]、DAG 技术[7]、状态通道[8]、侧链[9]等。其中分片技术是最有希望能提高事务吞吐量而不降低去中心化程度的扩容方案。

分片是一种分区方法,将计算和存储负载分散到对等(P2P)网络中。分片技术可以分为网络分片、交易分片和状态分片[10]。网络分片指将不同的节点在网络层划分到不同的分组中,网络分片是交易分片和状态分片的基础。交易分片将交易按照某种规则分配到不同分片,多个分片同时打包和验证不同的交易。交易分片仅仅实现交易的并行处理,每个节点依然需要保存所有分片的数据,节点存储压力大且同步状态时间长。状态分片指特定的分片只存储部分状态,从而达到减少状态存储冗余的效果。相比之下,状态分片不仅仅将交易的处理并行化,而且每个分片只需要保存部分状态

通信作者:李志淮(1720299464@qq.com)

信息，降低了节点的存储压力。因此，状态分片能从本质上解决区块链的可拓展性问题。

区块链分片后，共识节点数目减少，PBFT 共识算法[11]可以充分发挥优势，通信开销小且能实现交易的最终一致性。PBFT 共识机制允许网络中存在一定数量的恶意节点，只要恶意节点数量不超过网络中全部节点数量的 1/3，就可以达成有效性共识。然而，在采用 PBFT 共识机制的分片系统中，即使网络中总的拜占庭节点数量不超过系统总节点数量的 1/3，将节点随机分配到各分片后，很容易导致某个分片内拜占庭节点数量超过分片内总节点数量的 1/3，无法达成共识，降低了单个分片的安全性。

为了解决这一问题，文献[12]提出了多轮 PBFT 验证(Multiple Rounds of PBFT Verfication，MRPV)方案，其主要思想是：当某分片在处理交易时，如果验证成功，则将交易打包到区块上；如果未得到有效共识，不是直接将这笔交易丢弃，而是分配一组新的节点对这笔交易进行新一轮的验证，以此类推，直到该交易被成功验证或达到验证轮数上限后放弃这笔交易。MRPV 方案只是在网络分片和交易分片的基础上进行考虑，不必考虑节点状态同步的问题，但是可能会出现节点不断切换分片，迫使节点存储全局状态的情况，从而影响系统性能。在考虑状态分片的情况下，一旦有新的节点加入分片，则必须确保新节点有足够的时间进行状态同步，否则未完成同步的节点将无法参与验证。

综上所述，针对 MRPV 方案中的节点状态同步问题，本文提出了一种基于候补节点序列的状态同步方案。该方案确保 MRPV 方案在状态分片的基础上可行的同时，有效降低了单个分片的共识通信量，提高了交易验证率。

2 状态同步问题描述与分析

2.1 状态同步问题描述

状态同步指让区块链节点的状态保持最新，只有拥有最新状态的节点，才能参与到共识中去，进行下一个新区块的共识。状态同步是区块链节点参与共识的重要前提，为区块链共识提供必要的运行条件，所有的共识都是基于最新的区块进行。状态同步保证了区块链网络的安全性和健壮性。

当前的区块链中，每个节点存储所有的状态，处理所有的事务。若任意节点都需要对全网状态进行同步和存储，系统的总体性能将受限于单个节点的性能。这样，即使加入了大量节点，系统的总体性能也无法提升。

为了解决这一问题，状态分片概念被提出。该技术的关键是将整个存储区分开，让不同的分片存储不同的部分。每个节点只负责同步并存储自己所在的分片状态，而不是存储完整的区块链状态。

状态分片最大的优势是解决目前日益庞大的状态存储问题。然而，在单一分片内，当离线节点或恶意节点过多时，会导致分片内节点无法正常进行共识验证，甚至会在该分片账本上记录错误的信息。因此，为了避免这种情况发生，需要定期进行节点的重新分配。由于每个分片只存储部分状态，当节点切换到其他分片后，需要重新同步新分片的状态，这将极大增加节点的通信开销。同时，在节点同步状态完成前，分片的重新配置会导致节点无法参与共识验证，降低了系统吞吐量。

2.2 状态同步问题研究现状

针对因节点重组导致系统吞吐量低的问题，OmniLedger[13]和 RapidChain[14]均采用了每次只轮换一部分节点的思想。新节点同步状态信息的同时，为了防止中断，每个分片内仍然有足够的旧节点进行验证，避免了浪费过多时间进行状态信息同步。但是，如果在轮换期间，旧节点被破坏，剩余节点的数量将不足以达成共识，此时新节点还没有同步完状态，系统将无法继续处理事务。

Harmony[15]提出快速状态同步方案，为了可以快速验证事务，新加入的节点不会下载整个区块链历史状态，而是下载该分片的当前状态，并通过下载历史区块头检查其签名来验证当前状态。为了确保分片内共识安全，新节点需要在验证事务的同时进行分片状态同步，因为分片内应该存在部分节点同步完整的分片状态，否则将影响区块链的安全性。

为了让节点同步更少的状态，更快加入共识验证过程，文献[16]提出了运行轻客户端的思想。轻量级用户只需存储最近的摘要，就足以让用户验证事务的有效性。然而，轻客户端需要依赖全节点，无法独立验证交易，容易遭受攻击。而且只有少量的超级节点能够成为区块链全节点，这将导致中心化的出现。

无状态(stateless)[17-18]是 Eth 1.x 研究的新方向。完整状态节点需额外计算一个见证内容并将其附加到新区块；部分状态节点可以只在很短的一段时间内保持完整的状态，或者只“观察”它们感兴趣的一段状态，然后从见证内容那里获取它们验证区块所需的其余数据；零状态节点完全依赖见证内容来验证新的区块。但是以太坊无状态思想还在探索中，面临着证明信息过大、见证更新代价高的问题。

2.3 多轮 PBFT 验证方案的状态同步需求

MRPV 方案仅仅是针对网络分片和交易分片，当针对状态分片时，需要考虑状态分片中涉及的状态同步问题。在状态分片的情况下，MRPV 方案将带来因节点重组导致系统吞吐量低的问题。由于每个分片内节点只拥有当前分片的状态，一旦新节点加入分片，就必须确保新节点有足够时间与分片进行状态同步，否则新加入的节点将无法参与验证。

在 MRPV 方案中，每轮验证周期短，节点更换快，每一轮次都需要重新分配节点。如果采用无状态客户端思想，则由于见证更新慢，延迟太长，无状态节点将大概率被认为是拜占庭节点。因此，本文针对 MRPV 方案，提出了基于候补节点序列的状态同步方案，解决了节点的状态同步问题，从而有效提升了 MRPV 方案的共识效率。

3 基于候补节点序列的状态同步方案

在分片内增设候补节点序列以及待同步节点序列。同步好状态的节点先进入候补节点序列，当开始新一轮 PBFT 共识验证时，可以直接通过可验证随机函数(Verifiable Random Function，VRF)[19]随机选取候补节点序列中的节点，避免新加入分片的节点因同步状态而带来额外的延迟。

为了充分利用候补节点序列中的节点，候补节点需要对共识验证节点集合达成的共识进行验证，并提供欺诈证明[20-21]，同时能够获得相应的奖励；候补节点需要为待同步节点提供本分片状态；需要对完成状态同步的待同步节点进行验证，只有状态同步成功的待同步节点才允许进入候补节点序列。

3.1 状态同步方案总体设计

区块链中新加入的节点以及从各分片退出的验证节点包含在全局待选择节点序列 Ts 中，通过 VRF 随机算法从 Ts 中随机选取节点进入各个分片。分片内的所有节点被划分为 3 部分：1）分片内共识验证节点集合 Vsv，其功能为参加分片内的共识验证过程；2）分片内候补节点序列 Hs，当一轮共识验证结束，新一轮的共识验证节点可以直接在 Hs 中随机选取，降低了新一轮共识验证节点同步状态时所需的时延；3）分片内待同步节点序列 Ws，只有优先同步完成分片状态的节点才允许进入 Hs 中。节点状态转换如图 1 所示。

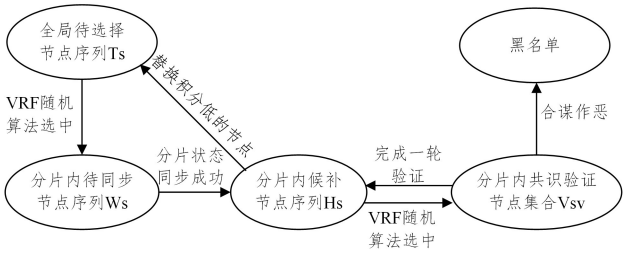

图 1　节点状态转换示意图

Fig. 1　Diagram of node state transition

当 Ts 中的节点被随机选择到分片内，首先进入 Ws 中。Ws 中的节点需要向 Hs 中的节点请求状态，并进行同步，优先同步状态成功且通过验证的节点才允许进入 Hs 中。每一轮共识结束后，在同步好状态的 Hs 中随机选择节点进行新一轮验证。共识验证流程如图 2 所示。

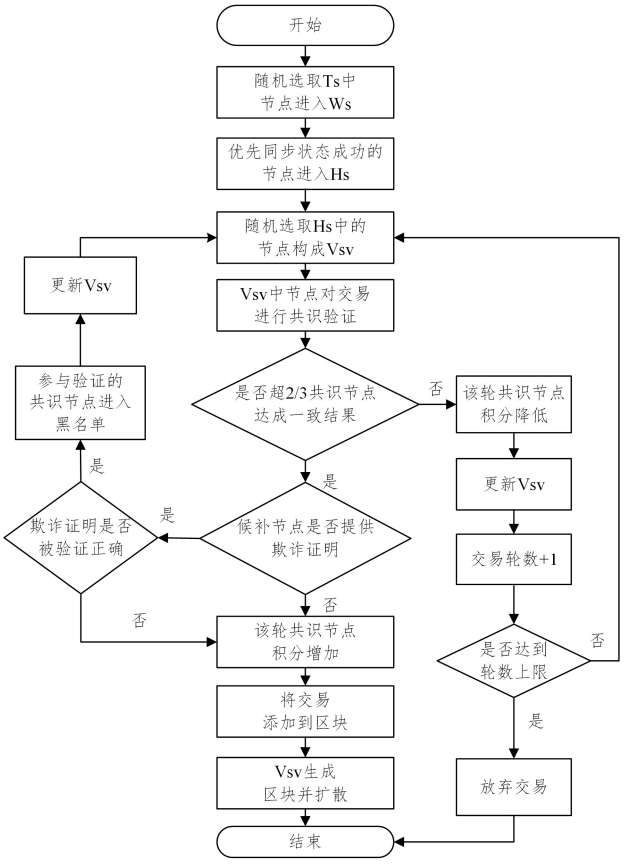

图 2　共识验证流程图

Fig. 2　Consensus verification flowchart

在 Hs 中随机选取节点组成共识验证节点集合 Vsv 对交易进行共识验证，在一轮共识验证结束后，Vsv 中的共识验证节点的表现成为了历史行为数据，构成后续的积分核算的依据。积分核算的流程如下：

（1）若未超过 2/3 的共识验证节点共识达成一致，则认为本轮共识失败，该轮共识验证节点的积分降低，并重新选取 Vsv 进行新一轮共识验证，直到交易验证成功或达到轮数上限，放弃该批交易；

（2）若超过 2/3 的共识验证节点共识达成一致，则等待一个时间范围，即争议时间段（Dispute Time Frame，DTF），排除可能存在的错误共识。在这一时间范围内，候补节点对 Vsv 达成的共识进行验证，筛选出错误共识。

（3）若存在候补节点验证 Vsv 达成的共识是错误共识，则提交欺诈证明，其余候补节点重新验证该共识。如果超过 50％的候补节点认为欺诈证明正确，则该轮 Vsv 中参与共识的共识验证节点进入黑名单；反之，提交欺诈证明的候补节点进入黑名单。

（4）若在 DTF 内无候补节点提交欺诈证明，则认为该轮 Vsv 达成的共识是正确的，该轮 Vsv 中节点的积分增加，将验证通过的交易进行打包，并生成新的状态。

每隔一段时间，Hs 中积分较低的节点被更换进入 Ts 中。被更换的节点积分清零，静默一段时间，再通过 VRF 随机算法被分配到某个分片内的 Ws 中，重新同步新分片的状态，获得新的积分。

3.2 共识验证节点状态生成

共识验证节点对交易进行多轮验证，将共识成功的交易打包至区块，生成新的状态。若交易超时未得到有效验证，则重新选取新的 Vsv 进行新一轮共识验证。由于在 MRPV 方案中，分片内的节点全部参与共识验证过程，而本文提出的状态同步方案中选取部分节点进行共识验证，分片内存在足够多的候补节点组成新的 Vsv 并行验证，以此提高交易共识验证效率。改进后的多轮验证原理如图 3 所示。

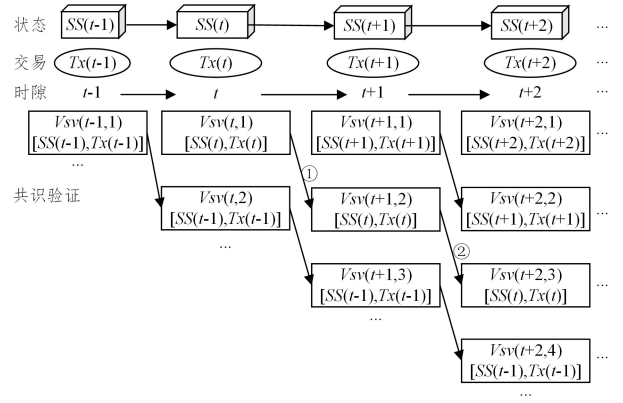

图 3　多轮验证原理

Fig. 3　Multi-round verification principle

图 3 中，在第 $t(t>1)$ 时隙，分片状态为 $SS(t)$，待验证的交易集合为 $Tx(t)$。在 Hs 中随机选取候补节点组成共识验证节点集合 Vsv 对交易集合 $Tx(t)$ 进行共识验证，记为 $Vsv(t,1)[SS(t),Tx(t)]$，即：在第 t 时隙，共识验证节点集合 Vsv 利用当前状态 $SS(t)$ 对当前交易集合 $Tx(t)$ 进行第 1 轮共识验证。

若交易在第 t 时隙被验证成功，则将交易打包进区块，并进行状态更新；第 t 时隙结束后，若 $Tx(t)$ 中存在验证失败的交易，则重新分配共识验证节点对该批交易进行新一轮验证，

即在$(t+1)$时隙进行第 2 轮验证，如图 3 中的①所示，记为$Vsv(t+1,2)[SS(t),Tx(t)]$。若仍未验证成功，则继续重新分配节点进行第 3 轮验证，如图 3 中的②所示，记为$Vsv(t+2,3)[SS(t),Tx(t)]$，直至交易验证成功或达到轮数上限时放弃交易。

在第$(t+1)$时隙分片状态更新为$SS(t+1)$。此时，将有新的交易集合$Tx(t+1)$等待被处理。需要在Hs中随机选取另外一组候补节点组成Vsv进行共识验证并生成区块，记为$Vsv(t+1,1)[SS(t+1),Tx(t+1)]$，即在第$(t+1)$时隙，共识验证节点集合$Vsv$利用当前状态$SS(t+1)$对当前交易集合$Tx(t+1)$进行第 1 轮共识验证，依此类推。

3.3 候补节点状态更新

由于每一时隙都可能产生新的区块，因此候补节点需要随机从拥有最新区块的共识验证节点同步区块，进行状态更新。由于分片内候补节点数量较多，共识验证节点不可能向每个候补节点发送区块。假设每个区块大小为 block_size，网络带宽为 bandwidth，则每个共识验证节点最多可向(bandwidth/block_size)个候补节点发送区块。在带宽较小、区块较大的情况下，能容纳的候补节点数量有限，因此该区块同步策略不具有可扩展性。

为了防止多个候补节点向同一共识验证节点同步区块，本文将候补节点随机分配给每个共识验证节点，以方便候补节点同步最新区块。分片内节点分布示意图如图 4 所示。

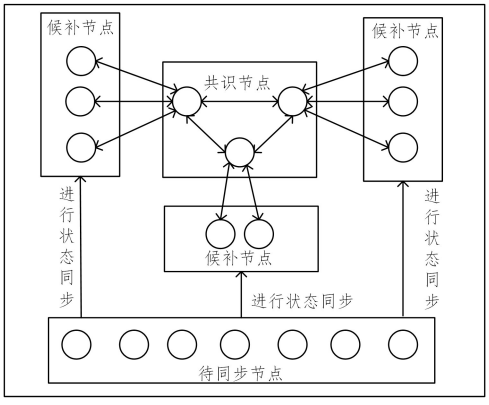

图 4 分片内节点分布示意图

Fig. 4 Diagram of node distribution in sharding

共识验证节点共识成功并提交新区块 block j，若共识验证节点与候补节点连通，则向候补节点发送最新区块状态，包括最新区块和区块哈希，记为{block j, block_hash(j)}。假设分片内共有Ns个节点，有Nsv个参与共识验证的节点，则最多有$(NsNsv)$个候补节点，将候补节点随机分配后，每个共一识验证节点最多向$((Ns-Nsv)/Nsv)$个候补节点发送区块信息。设区块大小为 block_size，则用于区块同步的网络带宽最大为$((Ns-Nsv)/Nsv*block_size)$，与分片内总的候补节点数无关，具有可扩展性。

若共识验证节点出错，或者同步状态时，由于部分候补节点断连，可能导致部分候补节点无法同步最新区块状态。本文采用 gossip 协议的思想，即分片内候补节点随机挑选若干个相通的候补节点，定期发送区块高度和区块哈希，记为{j, $block\_hash(j)$}。

由于部分候补节点可能已经同步了最新区块，因此为了防止网络风暴，候补节点不必再广播最新区块，仅需要向其他候补节点广播区块哈希，若部分候补节点收到 Hash 后发现本地没有，则向该节点主动获取，大大降低了网络的负载。同步区块状态示意图如图 5 所示。

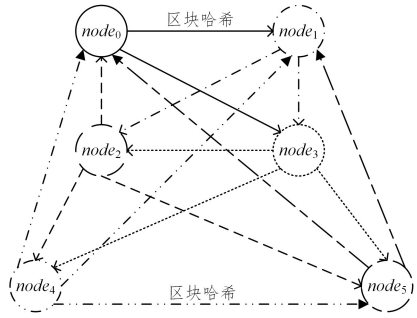

图 5 节点定期同步区块状态

Fig. 5 Nodes synchronize block status periodically

如图 5 所示，各个候补节点每隔一段时间随机选择多个相邻的候补节点广播{j, $block\_hash(j)$}；当候补节点收到其他候补节点发送的消息后，进行对比；若发现某节点区块高度高于本节点，则对比区块哈希，向该节点主动获取，更新本地状态到最新。通过这种方式，即使某共识验证节点出现问题或候补节点之间断连，也可以保证每个候补节点尽量同步完整的区块状态，增强了整个系统的健壮性。

3.4 待同步节点状态同步

未同步成功的节点验证交易时将直接拒绝交易，无法处理事务。因此，本文在分片内增设候补节点序列和待同步节点序列，同步分片状态成功的节点进入候补节点序列，未同步成功的节点进入待同步节点序列。候补节点序列中的节点需要为待同步节点提供分片状态，并对同步状态成功的待同步节点进行验证。待同步节点状态的同步流程如下：

(1)候补序列中的节点向待同步序列中的节点广播自身状态，包括区块高度和区块哈希，记为{j, $block\_hash(j)$}。

(2)待同步序列中的节点对比不同候补节点广播的区块哈希，随机挑选满足要求的候补节点，发送需要的区块区间。收到请求的候补节点，回复相应的区块，获得相应的奖励。

(3)收到回复的待同步节点，在本地形成一个下载序列，用于对下载完成的区块进行缓冲和排序。当序列中的区块能连接上节点当前本地的区块链时，则将区块从下载序列中取出，真正地连接到当前本地的区块链上。

(4)若待同步节点在同步状态的过程中，发现候补节点向其提供的状态有误，可以对其进行举报，在扣除相应奖励的同时该候补节点的积分将会降低，以此降低候补节点提供错误状态的概率。

(5)待同步节点同步状态完成后，向候补序列中的候补节点广播同步完成消息以及同步的各区块哈希值，消息格式为$\langle\langle SYN\text{-}DATA, M_i, i\rangle, Hash(blocks)\rangle$。其中，$M_i$为消息签名，$i$为节点自身编号，$Hash(blocks)$为同步成功的各区块的哈希值。

(6)候补节点对待同步节点发送的同步完成消息进行验证，若验证通过，则广播验证同步消息，消息格式为$\langle SYN\text{-}ACK, M_i, i, d\rangle$。其中，$d$为收到最后一个区块的哈希值。当半数以上的候补节点通过消息且消息中的d相同时，证明该节点同步状态成功。

待同步节点根据同步状态的过程表现获得积分。积分高

的待同步节点优先进入候补节点序列，以此防止某些待同步节点由于网络及自身能力的原因，在共识过程中不可避免地成为拜占庭节点。

每个新加入分片的待同步节点的初始积分为C_{init}，并且初始化 $T=0$ 进行同步时，每经过一个时隙，T 将加 1。直至节点同步完成状态并通过验证后，当前的积分C_j如式(1)所示：

$$C_j = C_{\text{init}} - T \tag{1}$$

每当有候补节点被更换出分片，从同步好状态的待同步节点中选取积分较高的节点进入候补节点序列。

3.5 共识验证节点数量选择

在多轮 PBFT 验证方案中，每一轮共识时间较短，需要有充分数量的候补节点进行候补。候补节点还需要同步每一轮共识后产生的区块，增加了分片内的通信量，因此共识节点数量应小于 MRPV 方案中设置的 60。共识节点数量减少，PBFT 共识算法通信开销显著降低，但是单个分片失效概率增加，可以通过增加一定的验证轮数来降低失效概率。本节将分析选择多少节点来进行共识，在降低分片内通信开销的同时，保证每个分片内的验证有效性。

假设系统中节点总数为 $N=2\,000$，系统中能容忍的总的拜占庭节点比例为$P_n=0.3$，将所有节点平均分成 10 个分片，则分片内总节点数量为$N_s=200$，X_1表示选取的N_s个节点中拜占庭节点的数量，则X_1服从超几何分布，记作$X_1 \sim H(N_s, N \cdot P_n, N)$。当系统中总的拜占庭节点比例$P_n$固定为 0.3，系统进行节点分配时，可看作从系统 N 个节点中为每个分片选取N_s个节点，则在分片内拜占庭节点数量为k_1时，分片内拜占庭的节点比例P_s的概率P_1如式(2)所示：

$$P_1(X_1=k_1)=\frac{\binom{N \cdot P_n}{k_1}\binom{N \cdot (1-P_n)}{(N_s-k_1)}}{\binom{N}{N_s}} \tag{2}$$

根据式(2)运用 python 模拟计算出单个分片内不同拜占庭节点比例的概率如图 6 所示。

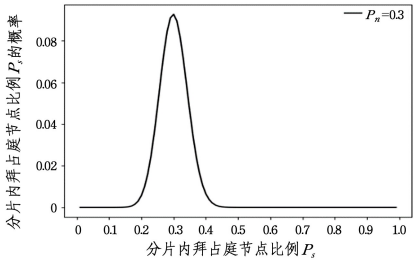

图 6 单个分片内不同拜占庭节点比例的概率

Fig. 6 Probability of different ratios of Byzantine nodes in a single sharding

由图 6 可知，当系统内总拜占庭节点比例 P_n 为 0.3 时，单个分片内拜占庭节点比例 P_s 在 0.15 到 0.4 范围内的概率较大。由于在分片内采用部分节点作为共识验证节点，设分片内共识验证节点的数量为 N_{sv}。当分片内共识验证节点数量不同时，共识验证节点数中拜占庭节点所占的比例也不同，从而导致节点共识失效概率不同，如式(3)所示：

$$P_{\text{failure}}(X \geqslant \frac{N_{sv}}{3})=\sum_{X=\frac{N_{sv}}{3}}^{X=N_{sv}}\frac{\binom{N_s \cdot P_s}{x}\binom{N_s \cdot (1-P_s)}{(N_{sv}-x)}}{\binom{Ns}{N_{sv}}} \tag{3}$$

根据式(3)运用 python 进行模拟运算，当单个分片内拜占庭节点比例 P_s 在 0.15 到 0.4 范围内，不同共识验证节点数导致共识失效的概率如图 7 所示。

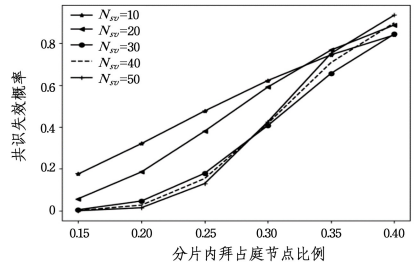

图 7 分片内拜占庭节点比例

Fig. 7 Proportion of Byzantine nodes in a sharding

由图 7 可知，当分片内共识验证节点数量 N_{sv} 为 10，20 时，即使分片内拜占庭节点比例较低，共识失效概率依然较高。当 $N_{sv}=40$，50 时，共识失效概率虽然小于 $N_{sv}=30$ 时的共识失效概率，但三者之间共识失效概率相差不大。但是，通过对 PBFT 共识机制通信开销进行分析，当共识验证节点增多，分片内通信开销显著增加，应选取尽量少的共识验证节点进行共识。因此，本文选定分片内共识验证节点数 $N_{sv}=30$。

由图 7 可知，当分片内拜占庭节点比例较高时，共识失效概率不可忽视，威胁系统的安全性。因此，本文将通过多轮验证，牺牲一定的延迟，来提高交易验证成功的概率。假设交易验证成功的概率是 P_w，交易在每轮验证失效的概率如式(3)所示。多轮验证中用 X 表示验证次数，多轮验证可看作交易进行 r 轮验证才得到第一次验证成功的机率。即交易在前$(r-1)$轮验证失败，在第 r 轮交易验证成功，则 X 服从几何分布，记为 $X \sim GE(P)$。交易验证成功的概率 P_w 如式(4)所示：

$$\begin{aligned} P_w(X \leqslant r) &= \sum_{X=1}^{r} P_w(X=r) \\ &= \sum_{X=1}^{r}(1-P_{\text{failure}})(P_{\text{failure}})^{r-1} \end{aligned} \tag{4}$$

根据式(4)模拟计算在分片内拜占庭节点比例不同时，轮数 r 对交易验证成功的概率 P_w 的影响，如图 8 所示。

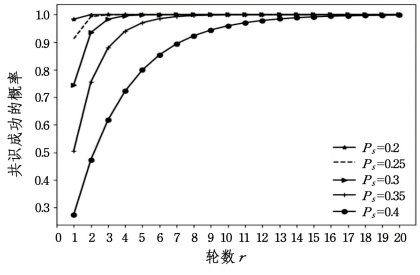

图 8 多轮轮数对共识成功概率的影响

Fig. 8 Effect of multi-round on consensus success probability

由图 8 可知，当分片内拜占庭节点比例 P_s 不变时，随着验证轮数 r 的增加，交易验证成功率 P_w 逐渐增大。当 P_s 低于 0.3 时，大多数交易 2 轮内共识成功。当 P_s 为 0.35 时，在 r 为 7 时，交易验证成功率接近 100%。当 r 达到 15 时，即使在 P_s 为 0.4 的情况下，交易验证成功率也达到了 99%。综上所述，多轮共识验证方法相比传统的单轮验证方法，交易验证成功率大幅提高。

4 实验设计

本节以采用分片技术和 PBFT 共识算法中最具有代表性的 OmniLedger 为基础，将通过多轮 PBFT 共识验证，验算更换积分低的节点后，多轮轮数对分片内拜占庭节点比例的影响；并从通信量和系统吞吐量两个方面对本文提出的状态同步方案进行测试，同时与原 MRPV 方案和 OmniLedger 方案进行对比，以此验证本文方案的有效性。

4.1 实验的基本设置

本实验以实验室局域网搭建的 P2P 测试网络为实验所需的网络环境，通过实验室 10 台服务器构建 10 个分片，用不同服务器的不同端口号模拟创建不同的共识验证节点，以普通 PC 机模拟网络中发起交易请求的客户端。实验数据用 MATLAB 进行绘制。在本文的设计方案中，实验前需要对一些参数条件进行设定，具体如下：

(1)节点的属性是可变的，即在本轮诚实的节点，在下一轮可变成拜占庭节点。

(2)认为网络中节点间通信良好，延迟在可控范围内。若由于节点延迟导致分片共识失败，同样按分片拜占庭节点比例大于 1/3 进行处理。

4.2 实验验证和结果分析

4.2.1 多轮轮数对拜占庭节点数量的影响

每轮共识验证结束后，共识验证节点集合中的节点根据其历史表现获得相应的积分，并进入 Hs 中。假设每个节点积分初始值为 100，若一轮 PBFT 验证后达成共识，当前所有共识验证积分加 5；若共识失败，则当前所有共识验证节点积分减 10。规定 Hs 中的节点积分小于等于 70 的节点为拜占庭节点。通过实验可知，随着轮数的增加，节点间积分差距较为明显，且积分小于等于 70 的节点数量不断增加。实验结果如图 9 所示。

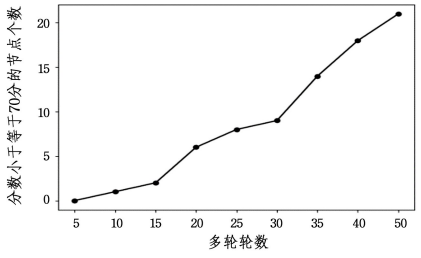

图 9 多轮轮数对拜占庭节点数量的影响

Fig. 9 Effect of number of rounds on number of Byzantine nodes

由图 9 可知，随着轮数的增加，积分小于等于 70 分的节点数量也增加。定期更换积分小于等于 70 分的节点，可以有效降低分片内拜占庭节点的数量，提高分片内共识成功的概率。

4.2.2 共识通信量对比实验

在本文方案中，每个分片内节点数量 $N_s=200$，参与共识验证的节点数量 $N_{su}=30$，虽然每一轮参与共识的节点数量较原 MRPV 方案($N_s=N_{su}=60$)少，每一轮共识验证时间较短，但是分片内的候补节点还需同步每一轮共识后产生的区块，增加了分片内的通信量。

由于 PBFT 共识算法需要经过三阶段的共识过程，算法复杂度为 $O(N^2)$，本文将每轮共识验证节点共识验证的通信量 T 抽象为式(5)。

$$T=(HashSize+BlockSize+P+C)\times N_{su}^2 \tag{5}$$

其中，$HashSize$ 为转发哈希的大小(固定为 32 字节)，$BlockSize$ 为打包的一个区块的大小，P 为 PBFT 共识机制中 prepare 阶段共识验证节点广播的准备消息的大小，C 为 PBFT 共识机制中 commit 阶段共识验证节点广播的准备确认消息的大小。

一个区块包括区块头和交易列表。以以太坊产生的区块为例，以太坊区块大小是不固定的，区块所占内存主要由区块内包含的交易数量决定。则一个区块的大小的计算式如式(6)所示：

$$BlockSize=Header\_size+\sum_{n=1}^{m=TxCount} txSize \tag{6}$$

其中，$Header\_size$ 为区块头的大小，$TxCount$ 为一个区块打包的交易数量，$txSize$ 为一条交易信息的大小。

在本文方案中，由于分片内节点不全都参与共识验证，每轮参与共识验证的节点数为 30，其余同步好状态的节点均为候补节点。每一轮共识验证结束后，候补节点也需要对新产生的区块进行同步，通过 3.3 节的分析可知，本文方案每一轮共识验证后，分片内共识通信量为 T^*，抽象为式(7)。

$$T^*=(HashSize\times 3(N_s-N_{su})^2+P+C) \tag{7}$$

通过具体实验，将两种方案的单个分片内共识通信量进行对比。假设单个分片内每轮共识成功的交易数量相同，但由于共识验证节点数量不同，导致单个分片内共识通信量不同，实验结果如图 10 所示。

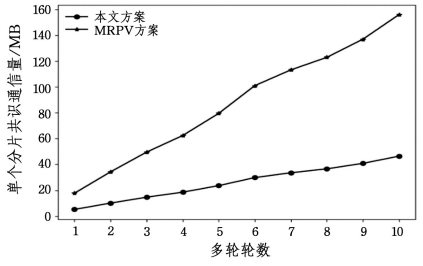

图 10 单个分片共识通信量

Fig. 10 Single sharding consensus communication traffic

由图 10 可以看出，随着轮数的增加，MRPV 方案的共识通信量比本文方案共识通信量多的同时增长幅度较大。因此，本文方案可以有效降低单个分片的共识通信量，提高交易验证速度。

4.2.3 系统吞吐量对比实验

系统吞吐量指单位时间内完成的交易数量，一般用 TPS(Transaction Per Second)来表示。本节通过观察在不同系统拜占庭节点的比例的条件下，MRPV 方案与本文方案和 OmniLedger 方案的平均 TPS 的变化情况。实验结果如图 11 所示。

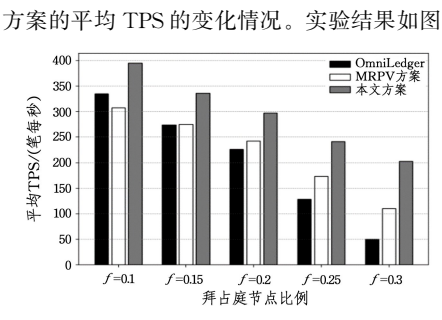

图 11 交易吞吐量

Fig. 11 Transaction throughput

由图 11 可以看出，本文方案的平均 TPS 优于其他两种

方案，拜占庭节点比例越高，本文方案的优势越明显。造成这种差异的主要原因有：1）本文方案在分片内采用多个共识验证节点集合对交易同时进行处理，且共识验证节点集合中节点的数目相对较少，继而通信量低，共识能够在短时间内达成；2）当对共识验证节点集合中的节点更新时，本文方案是从当前所在分片中已完成状态同步的节点进行轮换，更新后的共识验证节点集合无需进行状态同步与初始化，在一定程度上减少了时间的消耗，提高了共识达成的速度；3）本文方案能够将拜占庭节点进行更换，降低了拜占庭节点比例，提高了交易处理的效率。而 OmniLedger 方案与 MRPV 方案由分片内所有节点对交易进行验证，这将带来巨大的通信量；若对节点重新分配时，MRPV 方案对分片内的所有节点进行重分配，新分配的节点需进行分片的初始化与状态同步，这会消耗一定的时间，影响交易验证的有效性，因此平均 TPS 较低。综上，本文方案能够有效提高系统吞吐量。

结束语 因状态分片的约束，MRPV 方案中共识验证节点的随机分配受到了限制。新节点加入分片，必须确保有足够时间与分片进行状态同步，否则未同步成功的节点在验证交易时将无法处理事务。

因此，本文提出了一种支持 MRPV 方案的基于候补节点序列的状态同步方案，在分片内增设候补节点序列以及待同步节点序列。采用部分节点作为共识验证节点，其他同步好状态的节点均进入候补节点序列，未同步好状态的节点进入待同步节点序列。最后通过实验进行验证，本文方案在解决状态同步问题的同时，提高了节点共识验证效率，提升了系统的吞吐量。

但是，本文方案尚需进一步改进：有必要设计有效的激励和惩罚机制，以激励节点正确参与共识验证，惩罚恶意节点的恶意行为；其次节点提供欺诈证明，虽然增加了安全性，但是牺牲了一定时延，之后准备在优化时延方面做进一步研究。

参 考 文 献

[1] NAKAMOTO S. Bitcoin：A Peer-to-Peer Electronic Cash System [J]. Journal for General Philosophy of Science，2008，39(1)：53-67.

[2] BUTERIN V. Ethereum 2.0 Sharding-Faq [EB/OL]. (2019-04-18). https://github.com/ethereum/wiki/wiki/Sharding-FAQ.

[3] SUN Z X，ZHANG X，XIANG F，et al. Survey of Storage Scalability on Blockchain [J]. Journal of Software，2021，32(1)：1-20.

[4] YU G，WANG X，YU K，et al. Survey：Sharding in Blockchains [J]. IEEE Access，2020，8(99)：14155-14181.

[5] DANG H，DINH A，LOGHIN D，et al. Towards Scaling Blockchain Systems via Sharding [C]// 2019 International Conference on Management of Data. 2018.

[6] LUU L，NARAYANAN V，ZHENG C，et al. A Secure Sharding Protocol for Open Blockchains [C]// Proceedings of the SIGSAC Conference on Computer and Communications Security. ACM，2016：17-30.

[7] WANG Q. Improving the Scalability of Blockchain through DAG [C]// the 20th International Middleware Conference Doctoral Symposium. 2019.

[8] Raiden Foundation. Raiden Network Whitepaper [EB/OL]. [2018-05-11]. http://raiden.network.

[9] POON J，BUTERIN V. Plasma：Scalable Autonomous Smart Contracts [EB/OL]. [2017-08-11]. http://plasma.io/plasma.pdf.

[10] WANG G，SHI Z J，NIXON M，et al. SoK：Sharding on Blockchain [C]// the 1st ACM Conference. ACM，2019.

[11] CASTRO M，LISKOV B. Practical Byzantine Fault Tolerance [C]// Proceedings of the Third Symposium on Operating Systems Design and Implementation. USENIX Association. 1999：173-186.

[12] WANG F S，LI Z H，TIAN N. Multile Round PBFT Verification Scheme to Improve the Scale and Effectiveness of Sharding [J/OL]. Computer Engineering and Applications. http://kns.cnki.net/kcms/detail/11.2127.TP.20191207.0904.002.html.

[13] KOKORIS-KOGIAS E，JOVANOVIC P，GASSER L，et al. 2018 IEEE Symposium on Security and Privacy(SP)-OmniLedger：A Secure，Scale-out，Decentralized Ledger via Sharding [C]// 2018 IEEE Symposium on Security and Privacy(SP). IEEE Computer Society，2018：583-598.

[14] MAHDI Z，MAHNUSH M，MARIANA R. RapidChain：Scaling Blockchain via Full Sharding [C]// the 2018 ACM SIGSAC Conference. ACM，2018.

[15] The Harmony Team，Harmony Technical Whitepaper [R/OL]. [2019-06]. https://harmony.one/whitepaper.pdf.

[16] LEI X，LIN C，GAO Z，et al. Efficient Public Blockchain Client for Lightweight Users[J]. Security and Safety，2017，4(13)：153528.

[17] BUTERIN V. The Stateless Client Concept [EB/OL]. [2017-10-01]. https://ethresear.ch/t/the-stateless-client-concept/172.

[18] BONEH D，BUNZ B，FISCH B. Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains [C]// Proc of the 39th Annual Int Cryptology Conf. Berlin：Springer，2019：561-586.

[19] MICALI S，RABIN M，VADHAN S. Verifiable Random Functions [C]// Symposium on Foundations of Computer Science. IEEE Computer Society，1999.

[20] AL-BASSAM M，SONNINO M，BUTERIN V. Fraud Proofs：Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities[OL]. http://arxiv.org/abs/1809.09044.

[21] CAO S，KADHE S，RAMCHANDRAN K. CoVer：Collaborative Light-Node-Only Verification and Data Availability for Blockchains[C]// 2020 IEEE International Conference on Blockchain. 2020：45-52.

GAO Dong-xue，born in 1997，postgraduate. Her main research interests include blockchain technology and so on.

LI Zhi-huai，born in 1964，professor. His main research interests include blockchain technology and network information security.