Computer Science and Application 计算机科学与应用, 2022, 12(7), 1728-1736
Published Online July 2022 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2022.127173

Hans汉斯

结合BLS签名的Raft集群实用拜占庭容错算法

黄　刚

广东工业大学计算机学院，广东 广州

收稿日期：2022年6月5日；录用日期：2022年7月1日；发布日期：2022年7月8日

摘　要

针对联盟链中运用的实用拜占庭容错(Practical Byzantine Fault Algorithm, PBFT)共识算法通信复杂度高，无法支持大规模网络问题，提出一种结合BLS (Boneh-Lynn-Shacham)聚合签名的Raft集群实用拜占庭容错共识(Aggregate-Signature Raft Byzantine Fault Tolerance, ARBFT)算法。首先，对网络节点进行分组，组内采用Raft共识机制选出领导者，每个组内的领导者组成网络委员会；其次网络委员会内部采用改进的PBFT机制进行共识，改进了节点之间的交互方式，在prepare阶段各个副本节点单点发送信息及签名给主节点验证，在Commit阶段由主节点收集签名并验证，结合BLS签名将验证通过的多个签名聚合成一个聚合签名，将该聚合签名以及其它必要信息广播给其他所有副本节点验证，在验证通过后主节点和副本节点再进行组内共识。ARBFT共识算法将网络的通信复杂度降低为$O(N/k)+O(k)$，在多节点的情况下，通过实验对比经典PBFT和RBFT (Raft cluster Byzantine fault tolerance)共识算法，ARBFT共识算法在共识时延、通信开销、吞吐量等方面具有更好的性能。

关键词

区块链，实用拜占庭容错共识算法，联盟链，Raft算法，BLS签名

Raft Cluster Practical Byzantine Fault Tolerance Algorithm Combined with BLS Signature

Gang Huang

School of Computer, Guangdong University of Technology, Guangzhou Guangdong

Received: Jun. 5th, 2022; accepted: Jul. 1st, 2022; published: Jul. 8th, 2022

Abstract

Aiming at the high communication complexity of the Practical Byzantine Fault Tolerance (PBFT)

文章引用: 黄刚. 结合 BLS 签名的 Raft 集群实用拜占庭容错算法[J]. 计算机科学与应用, 2022, 12(7): 1728-1736.
DOI: 10.12677/csa.2022.127173

consensus algorithm used in the alliance chain and unable to support large-scale network problems, a Raft cluster practical Byzantine fault-tolerant consensus (Aggregate-Signature Raft Byzantine Fault Tolerance, ARBFT) algorithm. First, the network nodes are grouped, the group adopts the Raft consensus mechanism to select leaders, and the leaders in each group form a network committee; secondly, the network committee adopts an improved PBFT mechanism for consensus, which improves the interaction between nodes. In the prepare phase, each replica node sends information and signatures to the master node for verification at a single point. In the Commit phase, the master node collects and verifies the signatures. Combined with the BLS signature, the multiple signatures that have passed the verification are aggregated into an aggregate signature. Other necessary information is broadcast to all other replica nodes for verification, and the master node and replica nodes will conduct consensus within the group after the verification is passed. The ARBFT consensus algorithm reduces the communication complexity of the network to $O(N/k)+O(k)$. In the case of multiple nodes, through experiments and comparisons between the classic PBFT and RBFT (Raft cluster Byzantine fault tolerance) consensus algorithms, the ARBFT consensus algorithm has better performance in terms of consensus delay, communication overhead, and throughput.

Keywords

Blockchain, Practical Byzantine Fault Tolerance Algorithm, Consortium Blockchain, Raft Algorithm, BLS Signature

1. 引言

区块链[1]技术起源于 2008 年底中本聪发表的白皮书《比特币：一个点对点的电子货币体系》[2]，其核心是由共识机制[3]、密码学、分布式数据存储、智能合约等多种技术组成，区块链具有去中心化、不可篡改和公开透明的特点。其中共识算法是解决分布式系统一致性问题的核心技术。

联盟链[4]中实用拜占庭容错算法(PBFT) [5]是应用最广泛的共识算法之一，解决拜占庭容错算法效率低问题。IBM 的超级账本(HyperLeger [6])和 FISCOBSCOS 联盟链中均有使用到 PBFT 共识算法。当前 PBFT 共识算法还存在不足，PBFT 的共识效率依赖于参与共识的节点数量，因此随着节点数量的增多，完成共识的时间大大增加，整个网络的通信复杂度也会随节点数量的增加而增大，所以实用拜占庭共识算法不适合大规模的节点共识。

目前，针对 PBFT 共识算法存在的缺点已经有很多研究员提出了改进方法；文献[7]提出一种基于 Raft [8]集群的拜占庭容错共识机制，在大规模网络环境下，提高拜占庭容错能力的同时可以保证高共识效率，因而具有更高的扩展性，但是其通信开销还是较大。文献[9]基于 Kademlia 协议优化了 Raft 的领导者选举和共识机制，进一步提高了 PBFT 算法的领导者选举速度和交易吞吐量，但没有从根本上解决 PBFT 算法的瓶颈，节点间的通信复杂度较高，不适合大规模网络，Raft 共识算法不具备拜占庭容错能力。RBFT (Redundant Byzantine Fault Tolerance)算法[10]和 CBFT (Concurrent Byzantine Fault Tolerance)算法[11]，它们虽然在通信开销、选举安全性方面取得了一定提升，但性能依然不足；文献[12]提出一种基于特征信任模型的优化 PBFT，提高了拜占庭容错能力，但通信开销和共识时延并没有得到提升。

通过对已有改进 PBFT 算法研究成果进行分析，本文提出一种结合 BLS 聚合签名的 Raft 集群实用拜占庭容错共识算法。ARBFT 共识算法其主要贡献在于：

1) 将网络节点进行分组，每个分组内采用 Raft 共识算法，每个分组通过 Raft 协议选举领导者，每个分组的领导者组成委员会，委员会之间采用结合 BLS 聚合签名的 PBFT 共识算法。BLS 聚合签名具有聚合多个节点对同一消息的签名为一个签名的优点，使用 BLS 聚合签名可以改进节点之间的通信方式，降低整个网络的通信复杂度，提高通信效率。

2) 在 Raft 共识算法中引入监督节点，使得该共识算法具有拜占庭容错能力，防止主节点作恶的情况发生。

通过实验分析表明，ARBFT 共识算法在共识时延、通信开销、吞吐量等方面均有一定的提升。

2. 背景知识

2.1. PBFT 共识算法

PBFT 共识算法是由 Castro 等于 1999 年提出，该算法也称为实用拜占庭容错算法，用于解决拜占庭将军问题，同时改进 BFT 的算法效率。

PBFT 共识算法主要分为三个阶段：Pre-Prepare 阶段(预准备)、Prepare 阶段(准备)、Commit 阶段(提交)。预准备阶段 Pre-Prepare 消息由主节点广播给副本节点，副本节点收到主节点的 Pre-Prepare 消息进行验证，验证通过后进入准备阶段副本节点会广播 Prepare 消息。当某一节点收到不少于 $2f$ 条来自不同节点的 Prepare 消息并验证通过时，该节点进入提交阶段并广播 Commit 消息。同样，当每个节点收到不少于 $2f+1$ 条消息并验证通过时，该节点共识完成，将区块写入本地账本。在网络总节点数为 N 的系统中，PBFT 共识算法的通信复杂度为 $O\left(N^2\right)$，其所能容忍的错误节点数 f 最大为 $\left(N-1\right)/3$。其共识流程如图 1 所示。

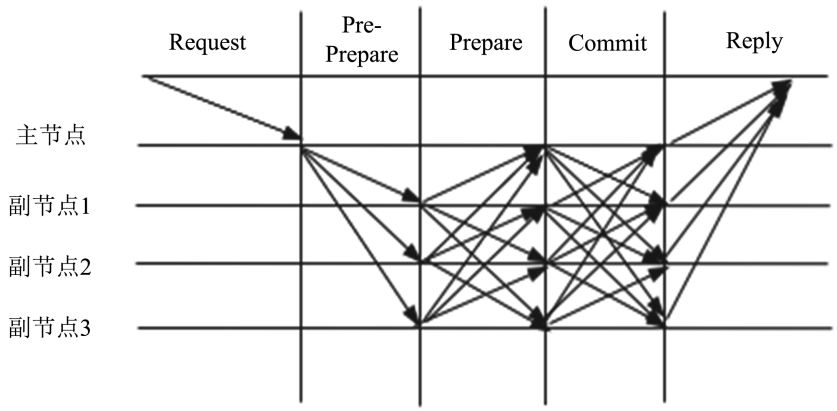

Figure 1. PBFT algorithm consensus process
图 1. PBFT 算法共识流程

PBFT 算法通过视图切换来为算法提供活性，当主节点发生宕机或者成为恶意节点，副本节点无法收到 $f+1$ 个相同消息时，系统会进入视图切换，更换主节点。

2.2. Raft 共识算法

Raft 是实现分布式共识的一种算法，该算法是由 Ongaro 等提出的，主要用来实现日志复制的一致性。Raft 共识算法有两个关键的工作机制：领导者的选举和账本复制。

在 Raft 共识算法中，节点有三种可能的角色：Follower (跟随者)、Candidate (候选人)、Leader (领导者)。Follwer 被动接受 Leader 发送的请求，所有节点刚开始的时候是处于 Follower 状态；Candidate 是 Follower 向 Leader 转换的中间状态；Leader 是负责和客户端交互以及日志复制，同一时刻最多只有 1 个 Leader 存在。在任一时刻，每个节点都处于这 3 种状态中的一种。

领导者选举工作机制：一开始，所有节点都是以 Follower 角色启动，同时启动选举定时器(随机时间)。如果节点没有感知到 Leader 的存在则该节点就会成为候选人，并且一直处于该状态，直到下列三种情况之一发生：该节点赢得选举、其他节点赢得选举、一段时间后没有任何一个节点赢得选举则计入下一轮任期选举。然后这个候选人就会向其他节点发送投票请求(Request Vote)，如果得到半数以上节点的同意，就成为 Leader。如果选举超时，还没有 Leader 选出，则进入下一任期，重新选举。完成 Leader 选举后，Leader 就会定时给其他节点发送心跳包(Heartbeat)，告诉其他节点 Leader 还在运行，同时重置这些节点的选举定时器。

账本复制工作机制：所有经过验证的交易在该任期内都由领导者打包生成区块。在收到超过半数节点回复后，领导者发送确认信息给跟随者，并将该区块作为账本上的下一个区块。跟随者收到领导者的确认信息后，将该区块作为本地账本上的下一个区块。

Raft 是一种强领导者算法，具有线性的复杂度，共识效率很高，但不具备拜占庭容错能力。

2.3. BLS 聚合签名

BLS 签名算法主要依赖于也称双线性映射(Bilinear maps)函数[13]，需要对配对函数有一定的理解。在这里做简单的介绍，定义配对函数 $e(P,Q)$，P 和 Q 为一条曲线(或两条不同的曲线)的两个点，配对函数对曲线上的运算满足分配律、交换律、结合律。假设要签署的消息为 m，G 为曲线上的一个生成点，私钥为 pk，则公钥为 $P = pk * G$。首先将消息 m 映射为曲线上的一个点 H，签名 $S = pk * G$，验证签名可以通过验证 $e(P,H)$ 是否等于 $e(G,S)$。

主节点收集各个副本节点发送的签名，再对验证通过的签名进行聚合，聚合后的签名大小与单个签名大小一致，仅需 33 字节，非常节约空间。对于签名的聚合操作，假设要聚合 100 个节点的签名，以 P_i 代表第 i 个节点的公钥，以 S_i 代表第 i 个节点的签名，每个签名都是不同节点的密钥对对同一个消息的签名。根据运算，聚合签名只是所有签名的总和，用 S 代表聚合后的聚合签名 $S = S_1 + S_2 + \cdots + S_{100}$，根据上面的证明，要对聚合签名进行验证，可以通过验证 $e(G,S)$ 是否等于 $e(P_1,H) * e(P_2,H) * \cdots * e(P_{100},H)$。

3. 本文共识

为了提供一种低时延、高吞吐量、通信开销小的共识算法，本文结合 PBFT 的拜占庭容错安全性与 Raft 高共识效率的优点，Raft 中引入监督节点，解决 Raft 共识中不能对抗拜占庭恶意行为的问题，提出一种结合 BLS 聚合签名的 Raft 集群拜占庭容错共识算法(ARBFT)。ARBFT 共识流程：首先启动节点生成，将节点进行预分组：确定分组数和选取监督节点，再在各个分组内通过 Raft 共识机制选出领导者组建委员会，委员会内有 PBFT 共识算法确定主节点，等待请求，开始共识，节点生成结束。

3.1. 分组策略

本文采用网络分片的方式对节点进行分组，引用文献[7]的分组方式，该分组方式通过对比 ELASTICO 协议[14]的网络分片机制和基于地理位置的分片机制[15]的优缺点，使用一致性 Hash 算法[16]设计思想对节点分组和选举监督节点，Hash 算法能够很好的解决分片不均问题，能使得组内领导者负载均衡。假设每 r 组分配监督节点，分组数为 k，监督节点数 s 需要满足

$$s \geq \begin{cases} \dfrac{k}{r} & k \bmod r = 0 \\ \dfrac{k}{r} & k \bmod r \neq 0 \end{cases} \tag{1}$$

假设分组数为4，每3组分配一个监督节点，则$s \geq 2$。通过对监督节点进行多次Hash运算，使其分配到不同的组中。监督节点需要保证匿名性，防止领导者的欺诈，可以在每组分配一个或多个相同的监督节点。表1使用每三组分配一个监督节点。

Table 1. Assign supervisory node
表1. 分配监督节点

分组方式	组号	分组节点	监督节点
每组含有一个监督节点	1	1, 7, 10	7
	2	2, 7, 11	7
	3	3, 7, 12	7
	4	4, 8, 13	8
	5	5, 8, 14	8
	6	6, 8, 15	8

3.2. 监督节点作用

Raft 算法加入监督节点使其具备抵抗拜占庭恶意节点的能力，能够防止领导者下发恶意信息破坏共识算法的一致性。如表1所示，每三组分配一个监督节点，监督节点只负责监督本组内领导者和传递消息的真实性，不参与领导者选举同时监督节点得保证匿名性，防止被领导者欺骗。监督节点会收到三个组内不同领导者的日志消息和签名，并对签名进行验证和比对内容，从而判断领导者是否作恶。如果发现某个领导节点作恶，监督节点需要打包消息$\langle P,t,m,i\rangle$发送给成员管理服务，成员管理服务需根据恶意节点的公钥等信息进行验证其是否为拜占庭恶意节点。其中P为消息标识，t为发现消息不一致时的时间，i为监督节点的编号，m为消息内容。

同时监督节点自身的安全性需要得到保证，有三种办法可以增加监督节点的安全性：第一，通过增加监督节点在组内的数量来解决监督节点收到攻击或者宕机的情况；第二，监督节点需要成员管理服务发送心跳消息作为存活证明，否则监督节点需要重选；第三，监督节点需要定期轮换，防止针对性攻击。

3.3. 共识流程

主记账节点将客户端收到的客户消息后进行共识，共识流程如图2所示。ARBFT共识算法主要分为五个阶段：Pre-Prepare阶段、Prepare阶段、Commit阶段、组内共识阶段、Reply阶段。

Pre-Prepare阶段：主节点将从客户端收到Request消息，然后广播给各个副本节点进行共识，主节点广播一条$\langle Pre\text{-}Prepare,h,v,d,m,s\rangle$消息给其他节点，其中$h$代表区块高度，$v$代表视图编号，$d$代表$m$的摘要，即消息的哈希值，$m$则为消息的内容，$s$代表客户端对消息的数字签名。

Prepare阶段：所有副本节点收到主节点的消息后，首先会对该消息进行验证，检查摘要、高度、视图以及签名的合法性，为了防止主节点作恶，所有副本节点将会通过客户端公钥进行校验数字签名s，验证无误后，所有副本节点发送一条消息给主节点$\langle Prepare,h,v,d,i\rangle$。与PBFT算法有所不同，这里所有副本节点不会进行广播，而只是将签名后的消息发送给主节点，这种方式减少了通信，其中i为节点编号。

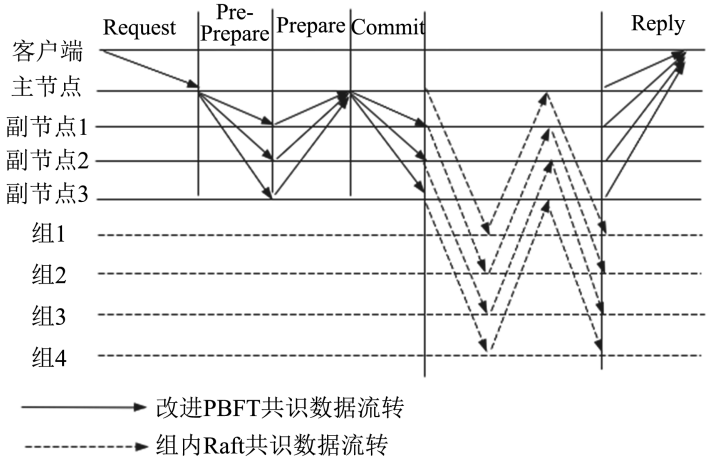

Figure 2. APBFT algorithm consensus process
图 2. APBFT 算法共识流程

Commit 阶段：主节点收到所有副本节点发送过来的 Prepare 消息后，会将每个节点进行验证，验证通过后收集起来，一旦主节点收到 $2f+1$ 个签名(包括自己的签名)并且验证通过的签名，将会将这些签名通过 BLS 签名聚合成一个签名，广播一条 $\langle Commit,h,v,d,aggrsignature,node\rangle$ 消息，这里的 *aggrsignature* 是聚合之后合成的签名，f 表示拜占庭节点个数，*node* 指的是所有参与该聚合签名的所有副节点的 *id* 列表，方便后面收到该消息的节点能利用参与签名的节点的公钥进行验证该签名是否正确，此时各节点收到主节点的聚合签名后，验证无误则将该区块链接到区块链的链尾完成同步。Commit 阶段的主要作用是由主节点收集签名并验证，结合 BLS 签名将验证通过的多个签名聚合成一个聚合签名，并将该聚合签名以及其他必要信息广播给其他所有副本节点验证，实现在不需要全广播通信的情况下各个副本节点能够根据参与签名的节点的公钥去验证该聚合签名的真实性。

组内共识阶段：副本节点验证 Commit 阶段主节点发送的消息，进行聚合签名验证。验证通过后，进入 Raft 共识阶段，副本节点包括主节点向组内节点广播 $\langle AppendLog,S,d,i\rangle$ 消息给组内 follower 节点，跟随者节点接收消息并反馈给主节点。主节点根据跟随节点反馈的结果，判断是否达成共识，如果收到消息个数达到大于等于 $n/2+1$，就认为达成共识并提交日志。S 代表领导者对消息的签名，n 表示组内节点的数量，i 表示节点的编号。

Reply 阶段：Raft 各个组内完成共识，回复客户端。

4. 实验结果与分析

为了验证本文改进后的 ARBFT 算法的有效性和可靠性，本节将从通信开销、共识时延、吞吐量三个方面对比其他主流算法的表现。本次实验在一台 Intel Xeon Cooper Lake (3.4 GHz/3.8Ghz)，32vCPU，64 G 内存，centOS8.2 64 位的云服务器上进行实验，采用多机器多节点模拟共识进行。文献[17]通过实验证明结合 BLS 去改进 PBFT 共识算法不仅不会造成整个网络耗时的增加，而且有效的改进 PBFT 的通信开销大的问题，有效地提高了共识效率，解决了 PBFT 算法随着节点增加的情况性能下降的问题，有利于提升整个网络的可扩展性。因此 BLS 聚合签名并不会带来额外的开销，且有利于维持较高的吞吐量以及一定程度上减少共识时延。

4.1. 通信开销

PBFT 共识算法需要两两节点进行通信，通信量为 $O\left(N^2\right)$ (其中 N 为节点数)，Raft 的通信量为 $O(N)$。

ARBFT 相比于经典 PBFT 通信量 $O\left(N^2\right)$ 和 RBFT 通信量 $O(N/k)+O\left(k^2\right)$ 下降到 $O(N/k)+O(k)$ (其中 k 代表分组数)。ARBFT 和 RBFT 算法分组数 k 和 N 节点总数需满足

$$\begin{cases} k=3f+1 & f\neq 0 \\ N=k(2f+1) & f\neq 0 \end{cases} \tag{2}$$

其中 f 为拜占庭节点个数，从 1 开始递增。同时 ARBFT 使用 BLS 聚合签名算法，减小了消息摘要的大小。图 3 为 ARBFT 共识与 PBFT 共识和 RBFT 共识通信开销比较。

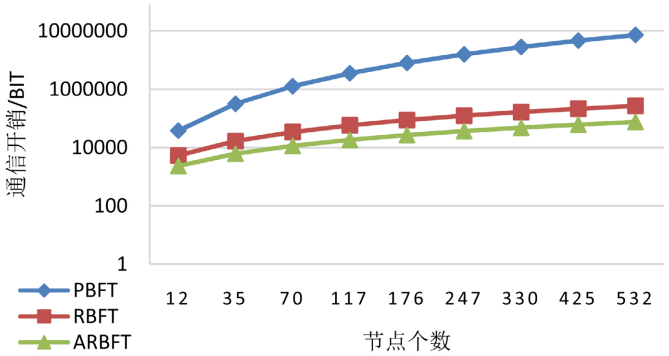

Figure 3. ARBFT and PBFT, RBFT algorithm communication overhead comparison
图 3. ARBFT 和 PBFT、RBFT 算法通信开销对比

从图 3 中看出 ARBFT 共识机制的开销远小于经典 PBFT 和 RBFT 共识机制通信的开销，如当网络节点数量为 176 时，经典 PBFT 的通信开销为 7.94×10^6 bit，RBFT 的通信开销为 8.72×10^4 bit，ARBFT 的通信开销为 2.66×10^4 bit。通信开销分别降低了 99.7%和 69.5%，且随着网络节点个数的增加，PBFT 和 RBFT 比 ARBFT 的通信开销越大。

4.2. 共识时延

算法的共识时延迟是指主节点打包完区块并发起网络共识到确认区块完成共识的时间间隔。本次实验分别在 20、40、60、80、100 个节点参与共识的情况下，分别记录在相同网络规模下 ARBFT 算法和对比算法 PBFT 和 RBFT 的数据，其中 RBFT 和 ARBFT 的分组数固定为 4 组，节点数增加但分组数不变。图 4 为 ARBFT 和 PBFT、RBFT 算法的共识时延对比。

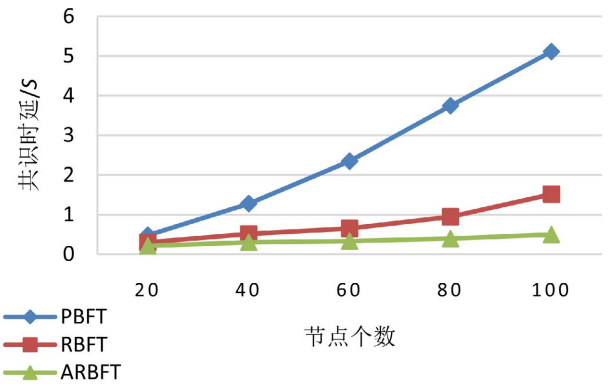

Figure 4. ARBFT and PBFT, RBFT algorithm delay comparison
图 4. ARBFT 和 PBFT、RBFT 算法共识时延对比

从图 4 可以看出，随着网络节点数量的增加，共识时延逐渐增大。其中，PBFT 算法增长最快，RBFT 算法其次，ARBFT 算法增长缓慢。相对来说 ARBFT 算法完成一个区块的共识所需的时间并不会随着节点的增多而急剧增大，共识时延增长比较稳定，而 PBFT 算法共识时延随着节点的增多而急剧增大，RBFT 算法效率跟 ARBFT 算法相比还有差距。ARBFT 利用 BLS 聚合签名对各个副本节点的签名进行聚合，并将该聚合签名以及其他必要信息广播给其他所有副本节点验证，优化了达成共识的过程，减少了通信的次数以及开销，共识时延降低。因此，ARBFT 共识算法在节点规模扩大时仍能保证高共识效率。

4.3. 吞吐量

吞吐量是指一个单位时间内系统处理的交易量，在区块链中，吞吐量表现为交易数量与处理对应交易时间的比值，即 $\mathrm{TPS}=\frac{M}{T}$。节点并发度是影响 TPS 的另一因素，节点并发度越高，交易量越大。实验过程分别针对 12 节点、16 节点、20 节点、24 节点的情况对数据吞吐量进行测试，RBFT 和 ARBFT 共识算法分组数为 4，节点数增加分组数不增加。图 5 为 ARBFT 和 PBFT 算法吞吐量测试的结果。根据图 5 可以发现，ARBFT 算法和 RBFT 算法的吞吐量随着节点数的增多而上升，ARBFT 上升的趋势比 RBFT 高，交易的吞吐量维持在较高水平，通过增加网络节点对其影响不大。而 PBFT 算法的吞吐量随着节点数的增加，吞吐量快速下滑。

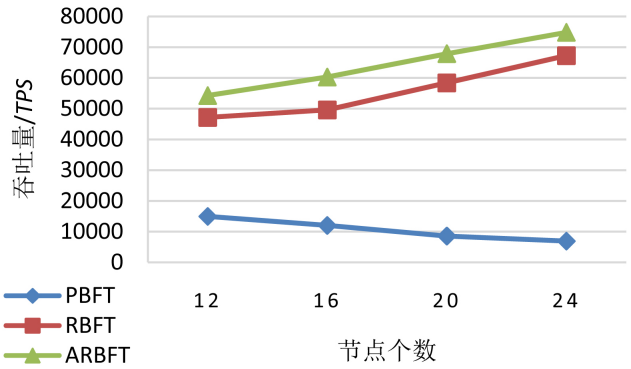

Figure 5. ARBFT and PBFT, RBFT algorithm throughput comparison
图 5. ARBFT 和 PBFT、RBFT 算法吞吐量对比

关于 ARBFT 算法保持高可拓展性的原因在于：一方面 RBFT 算法保留了 Raft 算法高效的共识逻辑，另一方面具有低通信复杂度。如果只增加组内节点数量，相比于共识时延，增加节点数量显著的增加了交易并发量。在同等网络规模情况下，ARBFT 的 TPS 约为经典 PBFT 的 300%~1000%，同时 TPS 优于 RBFT。因此，ARBFT 更适用于对 TPS 要求更高的联盟链应用场景。

5. 结束语

本文提出一种结合 BLS 聚合签名的 Raft 集群实用拜占庭容错共识机制——ARBFT。所提出的共识机制很好利用 Raft 共识效率高和 PBFT 拜占庭容错性好的特点。ARBFT 算法利用 BLS 聚合签名对各个副本节点的签名进行聚合，并将该聚合签名以及其他必要信息广播给其他所有副本节点验证，优化了达成共识的过程，通过实验验证，该改进方式大大降低了共识时延，随着网络节点数量的增加，ARBFT 算法明显优于 PBFT 和 RBFT 算法。同时 ARBFT 算法将原本 PBFT 算法的通信复杂度 $O\left(N^2\right)$ 和 RBFT 算法通信复杂度 $O(N/k)+O\left(k^2\right)$ 降低为 $O(N/k)+O(k)$，减少的网络通信量，提升了通信效率。通过与 PBFT、RBFT 共识机制对比实验，ARBFT 共识算法在共识时延、通信开销、吞吐量等方面具有更好的性能，能

有效突破制约当前联盟链落地的性能瓶颈，有助于推动联盟链的发展。在未来的研究中可以对 ARBFT 算法进行进一步优化监督节点的监督策略，提高容错性，增加节点动态加入和退出功能，使其能更适用于联盟链。

基金项目

广东省重点领域研发计划项目(2019B010139002)；广州市科技计划项目(201902020007、202007010004)。

参考文献

[1] Di Pierro, M. (2017) What Is the Blockchain? *Computing in Science& Engineering*, **19**, 92-95. https://doi.org/10.1109/MCSE.2017.3421554

[2] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. http://www.bitcoin.org/bit-coin.pdf

[3] 刘懿中, 刘建伟, 喻辉. 区块链共识机制研究: 典型方案对比[J]. 中兴通信技术, 2018, 24(6): 2-7.

[4] 黄步添, 蔡亮. 区块链解密: 构建基于信用的下一代互联网[M]. 北京: 清华大学出版社, 2018: 43.

[5] Castro, M. and Liskov, B. (1999) Practical Byzantine Fault Tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, New Orleans, 22-25 February 1999, 173-186.

[6] Androulaki, E., Barger, A., Bortnikov, V., *et al*. (2018) Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the Thirteenth EuroSys Conference*, Porto, 23-26 April 2018, Article No. 30. https://doi.org/10.1145/3190508.3190538

[7] 黄冬艳, 李浪, 陈斌, 等. RBFT: 基于 Raft 集群的拜占庭容错共识机制[J]. 通信学报, 2021, 42(3): 209-219.

[8] Ongaro, D. and Ousterhout, J. (2014) In Search of an Understandable Consensus Algorithm. *Proceedings of USENIX ATC*'14: 2014 *USENIX Annual Technical Conference*, Philadelphia, 19-20 June 2014, 305-320.

[9] Wang, R., Zhang, L., Xu, Q., *et al*. (2019) K-Bucket Based Raft-Like Consensus Algorithm for Permissioned Blockchain. 2019 *IEEE* 25*th International Conference on Parallel and Distributed Systems*, Tianjin, 4-6 December 2019, 996-999. https://doi.org/10.1109/ICPADS47876.2019.00152

[10] Aublin, P.L., Mokhtar, S.B. and Quéma, V. (2013) RBFT: Redundant Byzantine Fault Tolerance. *Proceedings of International Conference on Distributed Computing Systems*, Philadelphia, 8-11 July 2013, 297-306. https://doi.org/10.1109/ICDCS.2013.53

[11] Zhan, H. and Zhao, W. (2021) Concurrent Byzantine Fault Tolerance for Software-Transaction-Memory Based Applications. *International Journal of Future Computer and Communication*, **1**, 47-50. https://doi.org/10.7763/IJFCC.2012.V1.14

[12] Gao, S., Yu, T., Zhu, J., *et al*. (2019) T-PBFT: An Eigen Trust-Based Practical Byzantine Fault Tolerance Consensus Algorithm. *China Communications*, **16**, 111-123. https://doi.org/10.23919/JCC.2019.12.008

[13] Kurosawa, K. (2017) Advances in Cryptology. *ASIACRYPT* 2007: 13*th International Conference on the Theory and Application of Cryptology and Information Security*, Kuching, 2-6 December 2007, 4833-4839. https://doi.org/10.1007/978-3-540-76900-2

[14] Luu, L., Narayanan, V., Zheng, C., *et al*. A Secure Sharding Protocol for Open Blockchains. *Proceedings of the* 2016 *ACM SIGSAC Conference*, New York, 24-28 October 2016, 17-30. https://doi.org/10.1145/2976749.2978389

[15] Hyunkyung, Y., Jongchoul, Y. and Sunme, K. (2018) The Blockchain for Domain Based Static Sharding. 2018 17*th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*/12*th IEEE International Conference on Big Data Science and Engineering*, New York, 1-3 August 2018, 1689-1692.

[16] Li, Q., Sun, Z., He, R. and Tan, T. (2017) Deep Supervised Discrete Hashing. 31*st Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 2482-2491.

[17] 陈佳伟, 冼祥斌, 杨振国, 刘文印. 结合 BLS 聚合签名改进实用拜占庭容错共识算法[J]. 计算机应用研究, 2021, 38(7): 1952-1955+1962.