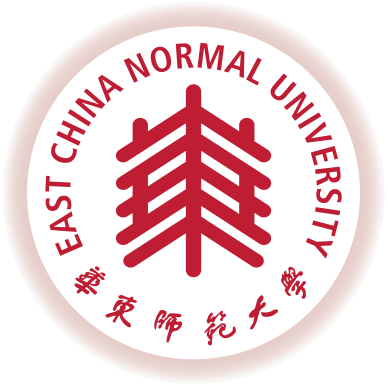
2022 届本科生学士学位论文　　学校代码：10269



華東師範大學

East China Normal University

本科生毕业论文

基于拜占庭容错的地理分布共识系统实现

Implementation of a Global Scale Consensus System Based on Byzantine Fault Tolerance

姓　　名：李昱鑫

学　　号：10174507132

学　　院：数据科学与工程学院

专　　业：数据科学与大数据技术

指导教师：张召

职　　称：教授

2022 年 5 月

华东师范大学学位论文诚信承诺

本毕业论文是本人在导师指导下独立完成的，内容真实、可靠。本人在撰写毕业论文过程中不存在请人代写、抄袭或者剽窃他人作品、伪造或者篡改数据以及其他学位论文作假行为。

本人清楚知道学位论文作假行为将会导致行为人受到不授予/撤销学位、开除学籍等处理（处分）决定。本人如果被查证在撰写本毕业论文过程中存在学位论文作假行为，愿意接受学校依法作出的处理（处分）决定。

承诺人签名：　　日期：　　年　　月　　日

华东师范大学学位论文使用授权说明

本论文的研究成果归华东师范大学所有，本论文的研究内容不得以其它单位的名义发表。本学位论文作者和指导教师完全了解华东师范大学有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权华东师范大学可以将论文的全部或部分内容编入有关数据库进行检索、交流，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

保密的毕业论文（设计）在解密后应遵守此规定。

作者签名：　　导师签名：　　日期：　　年　　月　　日

目录

基于拜占庭容错的地理分布共识系统实现

摘要：

随着区块链网络中副本规模的不断增大，并且在跨地理范围内分布，现有的拜占庭容错算法，如实用拜占庭容错共识协议 PBFT 等，其通信复杂度高，不能很好的支持跨地理范围部署副本。因此，需要对能够支持大规模跨地理范围部署副本的共识协议进行研究。地理分布共识算法通过利用网络拓扑信息将副本进行分组，在本地集群引入共识并行化，以及通过最小化集群全局间通信，实现了良好的可扩展性。通过实现地理分布共识算法并进行测量分析，得出以下结果：随着集群数量的增加，地理分布共识算法实现了良好的可扩展性；随着集群中副本数量的增加，尽管地理分布共识算法性能有所下降，但仍然优于实用拜占庭容错共识算法；随着批大小——单次共识决策中处理的客户端交易的数量增加，地理分布共识算法能够支持更大的批。通过结果表明，地理分布共识算法在多个集群的多个副本上分散共识，消除了由于任何单个主副本或集群的带宽和延迟造成的瓶颈，能够支持大规模跨地理范围部署副本。

关键词：区块链，共识算法，拜占庭容错，地理分布

Implementation of a Global Scale Consensus System Based on Byzantine Fault Tolerance

Abstract:

The scale of replicas in blockchain networks is constantly increasing, and replicas are deployed on a global scale. Unfortunately, existing Byzantine fault-tolerant consensus protocols, such as Practical Byzantine Fault-tolerant consensus protocol PBFT, are not designed to deal with geo-scale deployments in which many replicas spread across a geographically large area participate in consensus. Therefore, research is needed on consensus protocols that can support large-scale deployment of replicas across geographic scales. Geo-distributed Byzantine fault-tolerant consensus algorithms GeoBFT achieve good scalability by grouping replicas using network topology information, introducing consensus parallelism in local clusters, and by minimizing global inter-cluster communication. By implementing GeoBFT and conducting measurement analysis, the following results are obtained: First, as the number of clusters increases, the GeoBFT achieves good scalability. Second, as the number of replicas in the cluster increases, although the performance of GeoBFT decreases, it still outperforms PBFT. Third, as batch size - the number of client transactions processed in a single consensus decision - increases, GeoBFT is able to support larger batches.The results show that the geographically distributed consensus algorithm decentralizes consensus across multiple replicas of multiple clusters, eliminates bottlenecks due to bandwidth and latency of any single master replica or cluster, and is able to support large-scale global deployment of replicas.

Keywords: blockchain, consensus algorithm, Byzantine fault tolerance, geographic distribution, global scale

1、绪论

1.1 研究背景

共识协议是区块链系统的核心模块，直接决定了区块链在其分布式系统特性上的安全性与活性，目前学术界提出的共识协议主要分为四类：通过竞争记账权间接达成共识的 PoX 协议（PoW[1][2]，PoS[3]等），通过投票直接达成共识的 BFT 协议（PBFT[4][5]、HotStuff[6][7]等），通过选举代理人间接参与共识的代理协议以及舍弃传统线性链组织结构的 DAG 协议（SharPer[8]）。

在网络中节点规模较小时，现有的共识算法通常能时线较高的吞吐量和较低的时延。但随着网络中节点规模增大，并且这些节点分布在全球不同的地理位置，继续采用现有的共识算法系统的性能便会快速下降。相关测量结果表明，全局跨域网络的时延比本地局域网高 33-270 倍，全局跨域网络的吞吐量比本地局域网低 10-151 倍[9]，而现有的共识算法并不能感知这种地理分布导致的性能差异。

因此，如果要部署一个跨地域的区块链系统，其共识协议的设计必须要考虑地域分布特性，并能够感知出本地通信和全局通信，从而实现系统的最佳性能。除此之外，主流的共识协议如支持拜占庭容错的实用拜占庭容错协议 PBFT，其设计是集中式的，通过单一主节点协调所有的一致性决策，这与区块链去中心化的理念相悖，并且通信复杂度与节点数量的平方成正比[10]，扩展性较差。而其它的共识协议如联邦拜占庭协议 FBA、HotStuff 等同样具有中心化的领导者节点，也无法识别本地通信和跨域通信。

为了处理以上协议存在的问题，地理分布的共识协议需要考虑网络集群拓扑关系，识别出局域间通信和全局间通信。同时，也需要保证去中心化特性：任何单一节点或集群都不应该独自主导和协调全局的一致性共识决策——集中式设计会将全局带宽吞吐量和网络延迟限制在该单一节点或网络集群上。因此，具有地理分布感知的共识系统能够有效的提升在全球范围内部署的区块链系统的性能，减少网络延迟并有效提升带宽。

1.2 研究内容与方法

在实用拜占庭容错共识协议 PBFT 的基础上进行优化和改良，通过使集群中的副本节点能够感知网络拓扑关系，将在地理上接近的副本节点划分到同一个集群内，将副本之间两两通信改为集群内副本间通信和全局上的集群间通信，从而将 PBFT 中由单一主节点协调的集中式共识决策改为去中心化的共识决策。

首先，需要实现单一集群内部达成共识决策的实用拜占庭容错共识协议 PBFT。其次，需要实现支持集群间通信的优化全局共享协议。然后，为了支持优化的全局共享协议，需要实现一个新的远程视图更换（view change）协议，用于处理不同于单集群且更为复杂的主节点不响应或崩溃问题。最后，单个副本节点可以对接收到的已确定的交易进行排序与执行。

1.3 组织结构

本文第二节介绍了系统的需求分析和系统的基本流程；第三节对即与拜占庭容错的地理分布共识算法进行了叙述与证明；第四节对系统的网络场景和构建进行了详细描述；第五节给出了实验结果与分析，最后总结全文并展望。

2、系统分析

2.1 需求分析

区块链系统的核心共识算法都是拜占庭容错共识算法，它可以在系统中存在某些故障或恶意副本节点时提供活性与安全性保障。现有的区块链系统通常采用基于传统拜占庭容错共识的许可区块链设计[11][12][13]。这些许可区块链采用全复制模式，其中的每个副本都是已知的，并且每个副本都保存有所有链上数据的完整拷贝。相关测量结果表明，区域之间的通信比区域内部的通信成本高几个数量级，为了使许可区块链能够在跨地理范围内实现高性能部署副本，必须要区分本地通信和全局通信，并最小化全局通信。

因此，在跨地理范围共识协议中，必须要包含两个重要特性。其一是能够感知网络拓扑关系，并将处于同一个区域中的副本聚集在一起，并在这些副本内进行本地通信而不是全局通信。其二是分散共识，任何单一副本节点或集群都不应该负责协调所有的一致性决策，集中式设计会将吞吐量限制在主副本或单个集群的带宽和网络时延。而现有流行的共识算法不同时具备这两个条件。

在表2-1中，给出了跨地理分布共识算法和几种流行的拜占庭容错共识算法的复杂度分析对比[9][14]。

表 2-1　几种共识算法的通信复杂度（**z** 个集群，每个集群 **n** 个副本）

Table 2-1　Communication complexity of several consensus algorithms (z clusters, n replicas per cluster)

协议	决策数量	通信复杂度		中心化
		（本地）	（全局）	
地理分布共识算法	**z**	$\mathcal{O}(\mathbf{zn}^2)$	$\mathcal{O}(\mathbf{fz}^2)$	去中心化
↳单次决策	1	$\mathcal{O}(\mathbf{n}^2)$	$\mathcal{O}(\mathbf{fz})$	去中心化
Steward	1	$\mathcal{O}(2\mathbf{zn}^2)$	$\mathcal{O}(\mathbf{z}^2)$	去中心化
Pbft	1	$\mathcal{O}(2(\mathbf{zn})^2)$		中心化
Zyzzyva	1	$\mathcal{O}(\mathbf{zn})$		中心化
PoE	1	$\mathcal{O}((\mathbf{zn})^2)$		中心化
HotStuff	1	$\mathcal{O}(8(\mathbf{zn}))$		半去中心化

2.2 系统概要

跨地理分布共识算法将一个区域中的所有副本集中到一个集群内，并且让每个集群独立的做出一致性决策。然后通过全局共享协议以最小通信与其他的集群共享一致性决策，从而保证所有的集群能够得到相同的一致性决策。同样，也将每个客户端分配给其

所在区域的集群。这种集群划分有助于在跨地理规模部署副本节点中实现高吞吐量和良好的可扩展性。

在每一轮共识决策中，每个集群都能够对所属该集群的客户端请求达成一致性决策，并在通过全局共享阶段后对整个系统的该轮次的所有客户端请求达成共识。跨地理分布共识算法的每一轮次由图2-1中的三个步骤组成：本地复制、全局共享以及排序执行，将在下面进行详细介绍。

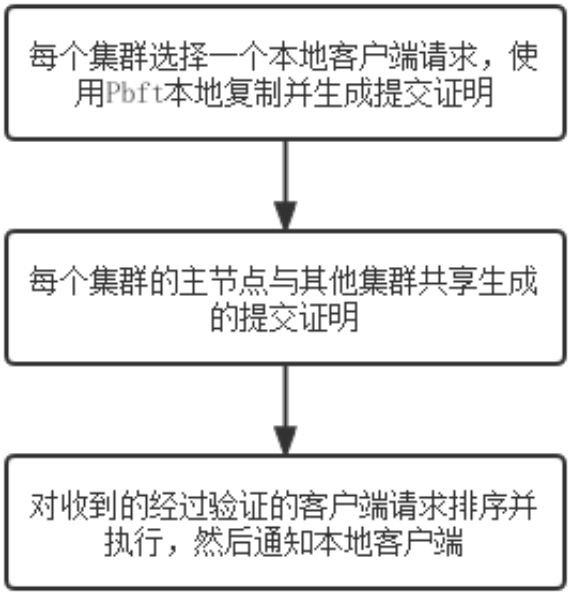


图 2-1　一轮地理分布共识算法的步骤

Figure 2-1　Steps in a round of GeoBFT

2.3 系统流程

首先在每一轮的开始，每个集群选择本地客户端的一个请求。然后每个集群使用实用拜占庭容错共识协议 Pbft 本地复制该事务。在本地复制阶段结束时，Pbft 保证每个无故障副本都可以通过生成提交证明以证明本地复制成功。

在全局共享阶段，每个集群都会与其他所有集群共享本地复制阶段的客户端请求和生成的提交证明，然后在本地集群内转发接收到的全局共享阶段消息。为了处理故障，需要使用一种新的远程视图更换协议。

在接收到其他所有集群本地复制阶段的客户端请求后，每个集群中的每个副本都可以确定性地对这些请求进行排序并执行。执行完成后，每个集群中的副本仅向本地客户端回复其请求的执行结果。

在图2-2中描述了由两个集群，每个集群 4 个副本组成的系统中一个轮次的地理分布共识算法。

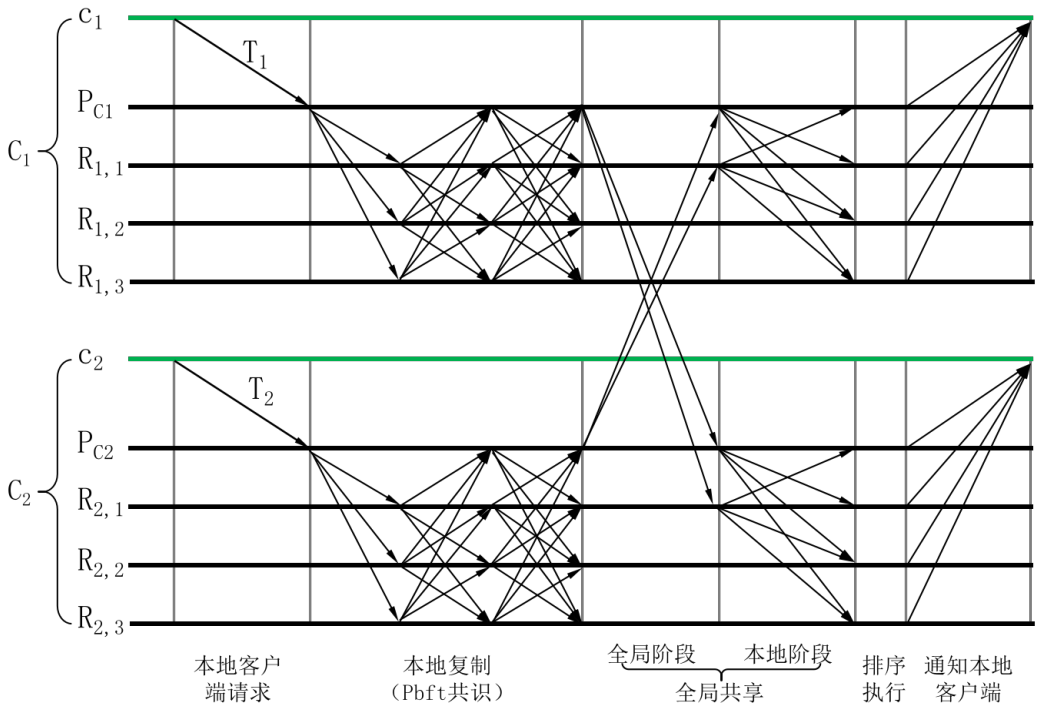


图 2-2　一轮地理分布共识算法的步骤

Figure 2-2　Steps in a round of GeoBFT

3、算法分析

3.1 概要

表 3-1　符号表 1

Table 3-1　Symbol table 1

符号	定义
R	副本
C	集群，$C=\{R_1,\cdots,R_n\}$
S	系统，$S=\{C_1,\cdots,C_z\}$
n	集群中的副本数量
z	系统中的集群数量
f	单个集群中最大容错数
$f(C_i)$	集群 C_i 中的无故障副本
$nf(C_i)$	集群 C_i 中的故障副本
P_C	集群 C 的主节点
$\langle m\rangle_u$	表示消息 m 由 u 签名

为了描述地理分布共识算法，过程中使用的符号如表3-1中所示。

在这里将能够进行地理感知并分组的地理分布共识算法表示为一组集群 C_i 的集合 S。对每个集群 C_i，都包含 **n** 个副本，其中最多有 **f** 个故障副本，并且 $\mathbf{n}>3\mathbf{f}$。

地理分布共识算法采用和 Steward 相同的故障模型，在系统 S 中最多可容纳 **fz** 个故障副本，并且每个集群中的故障副本不超过 **f** 个。这与 Pbft 等拜占庭容错共识算法不同。在这些算法中，都最多可容纳 $\lfloor \mathbf{zn}/3\rfloor=\lfloor \mathbf{z}(3\mathbf{f}+j)/3\rfloor=\mathbf{fz}+\lfloor \mathbf{z}j/3\rfloor$ 个副本。假设 $\mathbf{n}=16$，$\mathbf{f}=5$，$\mathbf{z}=6$，那么地理分布共识算法可容纳 $\mathbf{fz}=30$ 个故障副本，而 Pbft 可容纳 $\lfloor \mathbf{zn}/3\rfloor=32$ 个故障副本。

下面定义地理分布共识算法所提供的共识：

定义 3.1　设 S 是 R 上的一个系统，任何共识协议的单次运行都需要满足以下两个要求：

可用性：R 中每个副本都执行一个事务。

一致性：所有无故障副本执行相同的事务。

3.2 本地复制

地理分布共识算法的第一步是本地复制阶段。在这个阶段，每个集群都会独立的从本地客户端中选择一个请求去执行。在本地复制阶段，需要依赖于实用拜占庭容错共识

协议 Pbft。Pbft 是一种三阶段共识协议，通过主节点 P_C 协调并达成一致性共识决策，在图3-1描述了一轮 Pbft 共识协议的过程。下面详细介绍这几个步骤。

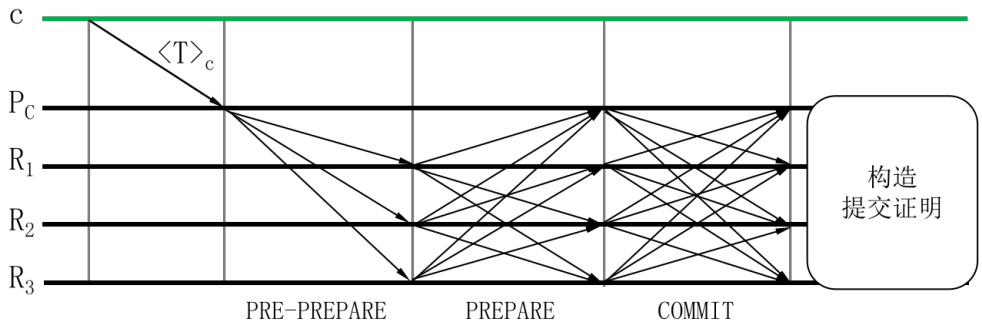


图 3-1　单个集群中一轮 PBFT 共识算法的步骤

Figure 3-1　Steps in a round of Pbft within a cluster C

首先，主节点 P_C 接收来自客户端的请求 $\langle T \rangle_c$，表示事务 T 由客户端 $c \in clients(C)$ 签名。

然后，主节点 P_C 广播一条 Pre-Prepare 消息将该请求提议给所有副本以发起对该请求的复制。当一个副本节点接收到 Pre-Prepare 消息后，将会进入下一阶段。当集群中有至少 $\mathbf{n - 2f > f}$ 个无故障副本接收到 Pre-Prepare 消息后，则对于该请求的提议成功。

当副本接收到一条 Pre-Prepare 消息后，副本会进入 PREPARE 阶段，并且通过广播一条支持 Pre-Prepare 消息 m 的 Prepare 消息作为对 Pre-Prepare 消息的响应。广播完之后，副本将会等待直到接收到 $\mathbf{n - f}$ 条支持消息 m 的 Prepare 消息。这表明系统中至少有 $\mathbf{n - 2f > f}$ 个非故障副本支持消息 m，此时将会进入下一阶段。

最后，接收到 n-f 条 Prepare 消息后，副本会进入 COMMIT 阶段，并且通过广播一条支持消息 m 的 Commit 消息。当副本接收到 n-f 条支持消息 m 的 Commit 消息后，能够保证最终所有的副本都可以提交请求 $\langle T \rangle_c$。

在本地复制阶段的最后，每个无故障副本都能够构造一个提交证明 $[\langle T \rangle_c, \rho]_R$ 来证明该次提交的确发生了。其中，提交证明 $[\langle T \rangle_c, \rho]_R$ 由客户端请求 $\langle T \rangle_c$ 和 $\mathbf{n - f > 2f}$ 个相同的支持 $\langle T \rangle_c$ 的 Commit 消息组成，并且这些 Commit 消息由不同的副本节点签名。

引理 3.1　[4][15] 设 S 是一个系统，并且 $C \in S$ 是一个满足 $\mathbf{n > 3f}$ 的集群，那么有

可用性：如果存在可靠的有限延迟通信，并且一个副本 $R \in C$ 能够构造一个提交证明 $[\langle T \rangle_c, \rho]_R$，那么所有的无故障副本 $R' \in nf(C)$ 最终都能构造一个提交证明 $[\langle T' \rangle_{c'}, \rho]_{R'}$。

一致性：如果副本 R_1，$R_2 \in C$ 能够分别构造提交证明 $[\langle T_1 \rangle_{c_1}, \rho]_{R_1}$ 和 $[\langle T_2 \rangle_{c_2}, \rho]_{R_2}$，那么有 $T_1 = T_2$ 且 $C_1 = C_2$。

证明 Pbft 能够保证一致性成立。当存在可靠的有限延迟通信时，为了保证系统可用性，Pbft 使用了视图更换和检查点技术。如果主副本出现故障，视图更换协议保证无故障副本能够发现主副本故障，并触发主副本更换直到找到无故障主副本。检查点协议则能够保证系统从故障和恶意行为中恢复到一致性状态，从而保证系统的可用性。

3.3 全局共享

当一个本地集群完成本地复制阶段都后，会进入全局共享阶段，与其他所有的集群共享在本地复制阶段提交的客户端请求和生成的提交证明 $[\langle T\rangle_c, \rho]_C$。全局共享阶段需要在集群间通信，该阶段需要在保障可靠检测到发送方故障的能力的前提下，最小化集群间通信数量。

假设 S 是一个系统，其中包含两个集群 C_1、$C_2 \in$S。在全局共享阶段，集群 C_1 的主节点 P_{C_1} 向集群 C_2 发送消息 $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$，消息 m 包含了在第 ρ 轮共识中本地集群提交的客户端请求和对于该请求的提交证明。为了能够保证能够检测到发送方故障并同时最小化通信，集群 C_1 需要向集群 C_2 发送 $\mathbf{f}+1$ 条消息 m。

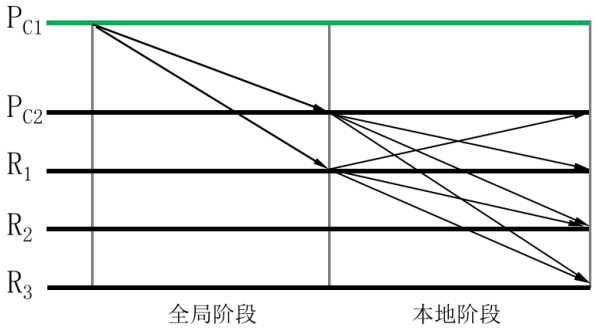


图 3-2　全局共享阶段的两个子阶段

Figure 3-2　Two sub-phases of the global sharing

如图3-2所示，全局共享阶段包含两个子阶段。首先是一个全局阶段，在这个阶段主节点 P_{C_1} 将消息 m 发送到集群 C_2 的 $\mathbf{f}+1$ 个副本。然后是一个本地阶段，这个阶段每个接收到正确的消息 m 的副本将消息 m 广播给本地集群的所有副本。

命题 3.1　设 S 是一个系统，C_1、$C_2 \in S$ 是两个集群，$m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$ 表示集群 C_1 在全局共享阶段发送给集群 C_2 的消息。有：

可用性：如果主节点 P_{C_1} 无故障并且存在可靠的有限延迟通信，那么 C_2 中的每个副本最终都将会接收到消息 m。

一致性：如果集群 C_2 中的副本接收到两条来自集群 C_1 的消息 $m_1 = (\langle T_1\rangle_{c_1}, [\langle T_1\rangle_{c_1}, \rho]_{C_1})$ 和 $m_2 = (\langle T_2\rangle_{c_2}, [\langle T_2\rangle_{c_2}, \rho]_{C_1})$，那么一定有 $T_1 = T_2$，$C_1 = C_2$。

证明 首先是可用性证明。如果主节点 P_{C_1} 无故障，那么集群 C_2 中的 $\mathbf{f}+1$ 个副本将会接收到消息 m。因为集群 C_2 中至多有 $\mathbf{f}$ 个拜占庭副本，因此在接收到消息 m 的副本中至少有一个副本是无故障副本，该副本会将消息 m 转发给集群 C_2 中的所有副本，保证了系统的可用性。

然后是一致性证明。提交证明 $[\langle T\rangle_c,\rho]_C$ 中包含了 $\mathbf{n}-\mathbf{f}>2\mathbf{f}$ 个副本签名的 Commit 消息，因此提交证明不可能被故障副本伪造，所以 C_2 中副本接收到的消息 m 的完整性能够很容易验证。而**引理 3.1** 排除了其他任何消息 $m'=(\langle T'\rangle_{c'},[\langle T'\rangle_{c'},\rho]_{C_1})$ 的存在，保证了系统的一致性。

3.4 远程视图更换

在以上的过程中，存在两种情况下集群 C_2 中的副本无法接收到 C_1 中的消息 m：一种是 P_{C_1} 是故障副本并且没有发送消息 m 到集群 C_2 中的 $\mathbf{f}+1$ 个副本；另一种是通信不可靠导致消息超出了最大时延或者丢失。在这两种情况中，都会触发集群 C_2 中的无故障副本启动远程视图更换以使得集群 C_1 替换主节点 P_{C_1}。在图3-3中给出了远程视图更换中的消息传递过程。

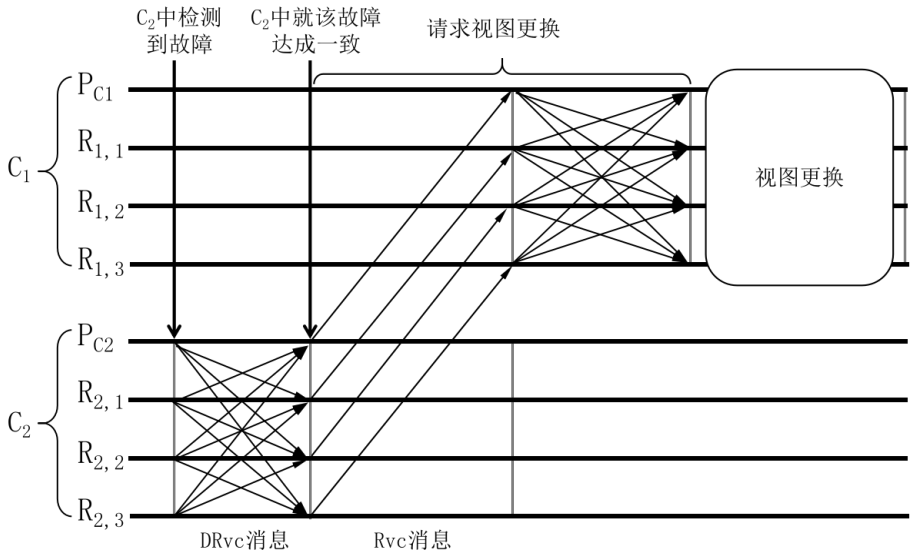


图 3-3 远程视图更换

Figure 3-3 Remote view change

以 C_1 中的主节点没有或无法向 C_2 中的副本发送消息 m 为例。首先，集群 C_2 中的副本需要能够检测到集群 C_1 中的故障。因此，在第 ρ 轮的开始，集群 C_2 中的每个副本 R 都为集群 C_1 设置一个定时器，并等待来自集群 C_1 的有效消息 m。如果定时器超时并且 R 仍然没有接收到消息 m，那么 R 检测到第 ρ 轮中集群 C_1 的故障。

当 $R\in C_2$ 检测到集群 C_1 的故障后，R 会在集群 C_2 中启动与其他副本就该故障达成一致的过程。R 会广播消息 $DRvc(C_1,\rho,v_1)$ 到集群 C_2 中的所有副本。然后，R 等待

接收到 $\mathbf{n}-\mathbf{f}>2\mathbf{f}$ 个不同副本签名的相同的 $DRvc(C_1,\rho,v_1)$ 消息后，能够保证集群 C_2 中的副本就集群 C_1 有故障达成一致。最后，R 向集群 C_1 中对应的副本 $Q\in C_1$ 发送消息 $\langle Rvc(C_1,\rho,v_1)\rangle_R$ 来强制更换集群 C_1 的主节点 P_{C_1}。

如果其他副本 $R'\in C_2$ 接收到了集群 C_1 的消息 m，则 R' 会用消息 m 响应消息 $DRvc(C_1,\rho,v_1)$。这使得 R 能够在无法就集群 C_1 有故障达成一致的情况下进行恢复。除此之外，如果 $R'\in C_2$ 没有接收到消息 m 并且定时器也没有超时，那么当 R' 接收到 $\mathbf{f}+1$ 条 $DRvc(C_1,\rho,v_1)$ 消息时便能够认为集群 C_1 发生了故障，因为这 $\mathbf{f}+1$ 条消息中至少有一条来自无故障副本，并且该副本成功的检测到了集群 C_1 故障。

当副本 $Q\in C_1$ 接收到视图更换请求 $\langle Rvc(C_1,\rho,v_1)\rangle_R$ 并经过验证后，Q 会广播该消息给集群 C_1 中的所有副本。当副本 Q 接收到 $\mathbf{f}+1$ 条不同副本签名的完全相同的 $Rvc(C_1,\rho,v_1)$ 消息后，它可以确定至少有一条远程视图更换请求来自集群 C_2 中的一个无故障副本。当集群 C_1 中没有正在进行的本地视图更换以及这是第一次接收到来自于集群 C_2 的对视图 v_1 的更换请求时，Q 会检测主节点 P_{C_1} 的故障。

设 S 是一个系统，C_1、$C_2\in S$ 是两个集群，$m=(\langle T\rangle_c,[\langle T\rangle_c,\rho]_{C_1})$ 表示第 ρ 轮 C_1 需要发送给 C_2 的消息。如果存在可靠的有限延迟通信，那么 C_2 中的每个副本要么接收到消息 m，要么 C_1 执行本地视图更换。在第 ρ 轮，集群中的每个副本将会接收到一组 z 个消息的集合 $\{(\langle T_i\rangle_{c_i},[\langle T_i\rangle_{c_i},\rho]_{C_i})\mid(1\leqslant i\leqslant\mathbf{z})\wedge(c_i\in clients(C_i))\}$。这些集合都包含相同的客户端请求。

3.5 排序与执行

当副本选择了要执行的客户端请求并且接收到了其他集群的所有客户端请求，便可以进入最后一步：排序并执行这些客户端请求。

在本地复制阶段和全局共享阶段结束后，每个副本在第 ρ 轮会收到相同的含有 z 个客户端请求的集合 $S_\rho=\{\langle T_i\rangle_{c_i}\mid(1\leqslant i\leqslant\mathbf{z})\wedge(c_i\in clients(C_i))\}$。地理分布共识算法可以在集群上使用预定义的顺序，例如按照 $[T_1,\cdots,T_2]$ 的顺序执行客户端请求。执行完成后，每个副本将会通知其本地客户端。当客户端收到 $\mathbf{f}+1$ 个相同的响应之后，可以保证其中至少有一条来自于一个无故障副本。

设 S 是一组副本上的一个系统，S 中的每个集群 C_i 都满足 $\mathbf{n}>3\mathbf{f}$。如果存在可靠的有限延迟通信，地理分布共识算法可以保证 S 中的每个无故障副本在一个轮次中执行相同的 $\mathbf{z}$ 个事务。

4、设计实现

4.1 使用环境

为了测试地理分布共识算法的性能，使用 Go 语言对地理分布共识算法和实用拜占庭容错算法进行了模拟仿真。实验环境为 Intel i5-8250U CPU 和 8GB 内存，操作系统为 64 位 Ubuntu 20.04.4 LTS，Go 语言版本 go1.17.9 linux/amd64。

表 4-1　符号表 2

Table 4-1　Symbol table 2

符号	定义
$\langle T\rangle_c/\langle REQUEST,o,t,c\rangle_c$	由客户端签名的请求消息
$\langle\langle PRE-PREPARE,v,n,d\rangle_{P_C},m\rangle$	由主节点 P_C 签名的 Pre-Prepare 消息
$\langle PREPARE,v,n,d,i\rangle_i$	由节点 i 签名的 Prepare 消息
$\langle COMMIT,v,n,d,i\rangle_i$	由节点 i 签名的 Commit 消息
$[\langle T\rangle_c,\rho]_C$	集群 C 生成的提交证明
$DRvc(C_i,\rho,v)$	集群内就集群 C_i 故障达成一致的消息
$Rvc(C_i,\rho,v)$	远程视图更换请求
$\langle REPLY,v,t,c,i,r>_i$	由节点 i 签名的 Reply 消息
$\langle VIEW-CHANGE,v+1,n,c,p,i\rangle_i$	由节点 i 签名的视图更换请求
$\langle NEW-VIEW,v+1,V,O\rangle_{P_C}$	由主节点 P_C 签名的新视图消息
o	客户端发起的请求执行的操作
t	时间戳
c	客户端 id
ρ	当前轮次
v	当前视图号
n	请求序列号
d	消息 m 的摘要
r	请求的执行结果
V	主节点 P_C 发送和接收的视图更换消息集合
O	一组 Pre-Prepare 消息的集合

4.2 功能模块

为了实现算法的仿真模拟，系统在 1 到 6 台机器上采用多进程方式部署，节点总数量根据不同的测试需求分别有 16、28、40、52、60 个。除此之外，节点之间的通信采用

远程过程调用（Remote Procedure Call，RPC）实现。

RPC 是一种计算机通信协议，它允许一台计算机上的程序调用另一台计算机上的子程序，而不需要额外的为这个交互过程编程。Golang 中已经有官方提供的库，因此调用起来十分方便。

除此之外，Golang 中的 goroutine 很适用于并发编程。协程是一种轻量级用户态线程，不存在上下文切换问题，因此效率比较高。Goalng 中使用 goroutine 只需要函数前添加一个 go 关键字即可为该函数创建一个 goroutine。在下述的流程中，每个 **event** 都可以看作一个 goroutine。

表4-1中是本节所使用的消息符号说明。

4.2.1 客户端

Send Request (Used by the client $c \in clients(C_k)$) **:**

1: Send Request $\langle T \rangle_c$ to P_{C_k}.

Receive Reply (Used by the client $c \in clients(C_k)$) **:**

2: **event** Receive $\langle REPLY, v, t, c, i, r \rangle_{R_i}$ from $R_i \in C_k$ **do**

3: **if** $|\langle REPLY, v, t, c, i, r \rangle_{R_i}| = \mathbf{f} + 1$ **then**

4: Statistics latency and throughput.

图 4-1　客户端流程

Figure 4-1　Client process

客户端由图4-1中所描述的两个流程组成。客户端向本地集群的主节点发送请求并等待直到收到 $\mathbf{f}+1$ 条对该请求的签名正确且结果相同的回复之后，才能够把 r 作为正确的执行结果。因为故障副本数不超过 $\mathbf{f}$ 个，所以 $\mathbf{f}+1$ 条一致的响应消息一定能够保证执行结果是正确的。

4.2.2 服务端 \ 副本节点

在副本节点的本地复制阶段包括图4-2中的四个流程。主节点在接收到本地客户端请求后，会为该消息分配一个序号 n，然后广播 Pre-Prepare 消息并将客户端请求也广播到集群中。客户端请求本身可以不包含在 Pre-Prepare 消息中，这样可以使 Pre-Prepare 消息足够小，有利于优化消息传输的速率。当接收到 Pre-Prepare 消息后，集群中的副本进入准备阶段并广播 Prepare 消息。同时，副本会将接收到的 Pre-Prepare 消息和 Prepare 消息保存到消息日志。接收到足够的正确的 Prepare 消息后，副本进入提交阶段，并广播 Commit 消息到集群中并将其和接收到的 Commit 消息一并保存到消息日志中。

Pre-Prepare Phase (Used by the replica P_{C_k} of C_k) **:**

1: **event** P_{C_k} receives $\langle T \rangle_c$ from $c \in clients(C_k)$ **do**

2: Broadcast $\langle \langle PRE-PREPARE, v, n, d \rangle_{P_C}, m \rangle$ to all replicas $R_i \in C_k$.

Prepare Phase (Used by the replicas $R_i \in C_k$) **:**

3: **event** Receive $\langle \langle PRE-PREPARE, v, n, d \rangle_{P_C}, m \rangle$ from P_{C_k} of C_k **do**

4: Broadcast $\langle PREPARE, v, n, d, i \rangle_i$ to all replicas $R_i \in C_k$.

Commit Phase (Used by the replicas $R_i \in C_k$) **:**

5: **event** Receive $\langle PREPARE, v, n, d, i \rangle_i$ from $R_i \in C_k$ **do**

6: **if** $|\langle PREPARE, v, n, d, i \rangle_i| = \mathbf{n} - \mathbf{f} > 2\mathbf{f}$ **then**

7: Broadcast $\langle COMMIT, v, n, d, i \rangle_i$ to all replicas $R_i \in C_k$.

Generate Commit Certificate (Used by the replica P_{C_k} of C_k) **:**

8: **event** Receive $\langle COMMIT, v, n, d, i \rangle_i$ from $R_i \in C_k$ **do**

9: **if** $|\langle COMMIT, v, n, d, i \rangle_i| = \mathbf{n} - \mathbf{f} > 2\mathbf{f}$ **then**

10: Generate commit certificate $[\langle T \rangle_c, \rho]_C$.

图 4-2　副本节点流程（本地复制阶段）

Figure 4-2　Replicas process (Local replication phase)

当 Prepare 消息经过验证并且接收到至少 $2\mathbf{f}+1$ 个 Commit 消息（包括自己）和对应的 Pre-prepare 消息时，能够保证集群中的副本最终都会提交该客户端请求。此时，可以构造提交证明 $[\langle T \rangle_c, \rho]_C$ 并进入下一阶段——全局共享阶段。

全局共享阶段由图4-3中的两个子阶段构成。以两个集群之间的为例，假设存在两个集群 C_1 和 C_2，集群 C_1 向 C_2 发送消息 $m = (\langle T \rangle_c, [\langle T \rangle_c, \rho]_{C_1})$。在全局阶段，集群 C_1 的主节点 P_{C_1} 从集群 C_2 中选择 $\mathbf{f}+1$ 个副本并发向它们送消息 m，能够保证至少有一个无故障副本可以接收到消息 m。当集群 C_2 中的副本接收到消息 m 后在集群 C_2 中进行广播。

当集群中的副本接收到一组 $\mathbf{z}$ 个消息的集合 $\{(\langle T_i \rangle_{c_i}, [\langle T_i \rangle_{c_i}, \rho]_{C_i} | (1 \leqslant i \leqslant \mathbf{z}) \wedge (c_i \in clients(C_i))\}$ 后，副本将会向本地客户端发送请求的执行结果。注意，这里的副本仅向本地客户端发送结果（图4-4）。

Global Phase (Used by the replica P_{C_1} of C_1) **:**

1: **event** Generated commit certificate $[\langle T\rangle_c, \rho]_C$ **do**

2: Choose $\mathbf{f}+1$ replicas in C_2.

3: Send $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$ to the $\mathbf{f}+1$ replicas in C_2.

Local Phase (Used by the replicas $R_i \in C_2$) **:**

4: **event** Receive $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$ from P_{C_1} of C_1 **do**

5: Broadcast $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$ to all replicas $R_i \in C_2$.

图 4-3 副本节点流程（全局共享阶段）

Figure 4-3 Replicas process (Global sharing phase)

Reply to the client (Used by the replica $R \in C_k$) **:**

1: **event** Receive message $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_i})$ **do**

2: **if** $|\{(\langle T_i\rangle_{c_i}, [\langle T_i\rangle_{c_i}, \rho]_{C_i} | (1 \leqslant i \leqslant \mathbf{z}) \wedge (c_i \in clients(C_i))\}| = \mathbf{z}$ **then**

3: Broadcast $\langle REPLY, v, t, c, i, r >_i$ to local client $c \in clients(C_k)$.

图 4-4 副本节点流程（回复客户端）

Figure 4-4 Replicas process (Reply to the client)

4.2.3 远程视图更换

远程视图更换包括图4-5中的两个阶段。以两个集群为例，假设集群 C_1 是故障集群。如果集群 $R \in C_2$ 检测到了集群 C_1 的故障，那么 R 在集群 C_2 中广播 $\langle DRvc(C_i, \rho, v)\rangle_R$ 消息。当一个集群收到 $\langle DRvc(C_i, \rho, v)\rangle_R$ 消息时，如果该节点在此前接收到了集群 C_1 正常传来的消息 m，那么该节点用消息 m 响应消息 $\langle DRvc(C_i, \rho, v)\rangle_R$；如果接收到了 $\mathbf{f}+1$ 条 $\langle DRvc(C_i, \rho, v)\rangle_R$ 消息并且自身没有探测到集群 C_1 故障，那么此时可认为集群 C_1 故障；如果接收到了 $\mathbf{n}-\mathbf{f}$ 条 $\langle DRvc(C_i, \rho, v)\rangle_R$ 消息，那么在集群 C_2 中就集群 C_1 故障达成一致，此时发送 $\langle Rvc(C_i, \rho, v)\rangle_R$ 消息以强制更换集群 C_1 的主节点。

对于集群 C_1，当其中的副本接收到 $\langle Rvc(C_i, \rho, v)\rangle_R$ 消息时，在集群内广播该消息；当接收到来自集群 C_2 的 $\mathbf{f}+1$ 条 $\langle Rvc(C_i, \rho, v)\rangle_R$ 消息时，可认为主节点 P_{C_1} 故障并进行本地视图更换。

4.2.4 本地视图更换

当本地集群中的某个副本为本地集群主节点设置的定时器超时，或者接收到足够且正确的远程视图更换阶段消息 $\langle Rvc(C_i, \rho, v)\rangle_{R_i}$ 时，可以触发本地的视图更换。首先会

Initiation Phase (Used by the replica $R \in C_2$) **:**

event Detect failure of C_1 in round ρ **do**
  Broadcast $\langle DRvc(C_i, \rho, v)\rangle_R$ to all replicas in C_2.
event Receive $\langle DRvc(C_i, \rho, v)\rangle_{R_i}$ from $R_i \in C_2$ **do**
  if R received $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$ from $Q \in C_1$ **then**
    Send $m = (\langle T\rangle_c, [\langle T\rangle_c, \rho]_{C_1})$ to R_i.
  if $|\{\langle DRvc(C_i, \rho, v)\rangle_{R_i} \mid 1 \leqslant i \leqslant \mathbf{n}| = \mathbf{f} + 1$ **then**
    Detect failure of C_1.
  if $|\{\langle DRvc(C_i, \rho, v)\rangle_{R_i} \mid 1 \leqslant i \leqslant \mathbf{n}| = \mathbf{n} - \mathbf{f}$ **then**
    Send $\langle Rvc(C_i, \rho, v)\rangle_R$ to $Q \in C_1$, id(Q)=id(R).

Response Phase (Used by the replicas $Q \in C_1$) **:**

event Receive $\langle Rvc(C_i, \rho, v)\rangle_{R_i}$ from C_2 **do**
  Broadcast $\langle Rvc(C_i, \rho, v)\rangle_{R_i}$ to all replicas in C_1.
  if $|\{\langle Rvc(C_i, \rho, v)\rangle_{R_i} | (1 \leqslant i \leqslant \mathbf{n}) \wedge (R_i \in C_2)\}| = \mathbf{f} + 1$ **then**
    Detect failure of P_{C_1}.

图 4-5　副本节点流程（远程视图更换阶段）

Figure 4-5　Replicas process (Global view change phase)

Local View Change (Used by the replica R of C_k) **:**

event Detect failure of P_{C_k} **do**
  $v := v + 1$.
  Broadcast $\langle VIEW - CHANGE, v + 1, n, c, p, i\rangle_R$ to all replicas in C_k.
event Receive $\langle VIEW - CHANGE, v + 1, n, c, p, i\rangle_R$ from $R \in C_k$ **do**
  if $|\langle VIEW - CHANGE, v + 1, n, c, p, i\rangle_R| = 2\mathbf{f}$ **then**
    Broadcast $\langle NEW - VIEW, v + 1, V, O\rangle_{P_C}$ to all replicas $R \in C_k$.

图 4-6　副本节点流程（本地视图更换阶段）

Figure 4-6　Replicas process (Local view change phase)

在本地集群中广播视图更换消息 $\langle VIEW-CHANGE, v+1, n, c, p, i\rangle_R$。当新视图中的主节点接收到 **2f** 条有效的来自不同节点相同的视图更换消息时，会在本地集群中广播新视图消息 $\langle NEW-VIEW, v+1, V, O\rangle_{P_C}$。副本节点接收到有效的新视图消息后，主节点和副本都将进入新视图 $v+1$（图4-6）。

5、测试评估

为了对地理分布共识算法 GeoBFT 和实用拜占庭容错算法 Pbft 进行对比分析，首先使用 Golang 实现了两种算法。然后在腾讯云上进行了部署，使用了六台 4 核 8GB 内存内存的主机，每台主机上运行一个客户端进程和多个服务端副本节点进程。每次测试每个客户端都会发送 1000 条请求。为了模拟真实环境中的网络延迟，参考了现有的不同区域的网络延迟数据[9]并在配置文件中设置分属不同集群的副本节点之间的网络延迟。

5.1 集群数量的影响

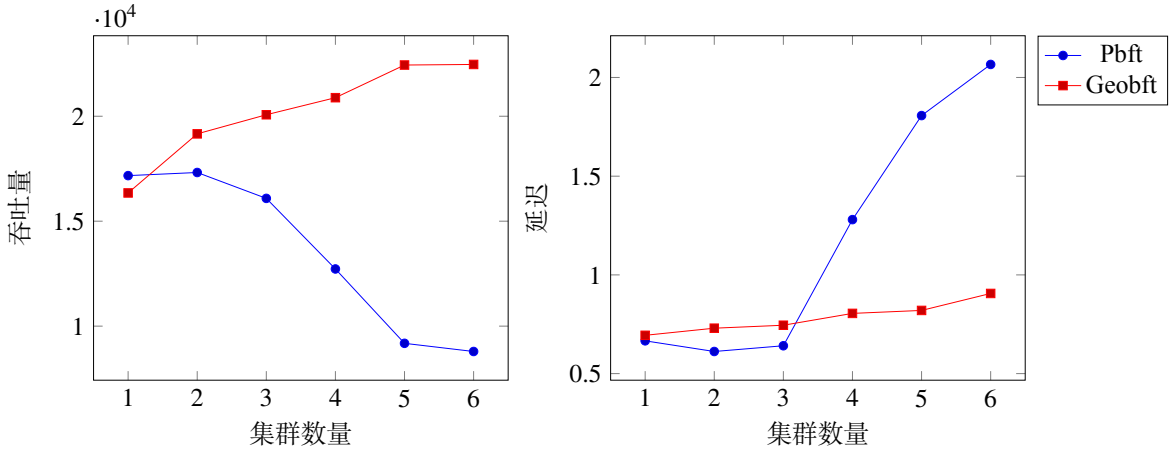


图 5-1　集群数量的影响

Figure 5-1　Impact of the number of clusters

为了测试集群数量对于性能的影响，首先保证系统中副本总数不变，集群数量从 1 增加到 6。在实验过程中，对于每种情况下的数据，按相同配置运行三次取平均值作为最终的结果。在两种算法中，集群数量对于系统性能的影响如图5-1所示。

可以看到当集群数量较少的时候，Pbft 能够维持比较好的性能。在单个集群上运行时，地理分布共识算法由于全局共享阶段的存在造成额外开销，所以性能相对比较低一点。随着集群数量的增多，全局通信成为系统性能的瓶颈，此时 Pbft 算法的性能快速下降，而地理分布共识算法能够实现较好的扩展性。这是因为相对于 Pbft 算法中不论本地或者全局，都需要副本之间的两两通信，而地理分布共识算法由于在集群间只需要传递 $\mathbf{f}+1$ 条消息，远少于 Pbft 所需的全局间通信数量。因此，地理分布共识算法的性能大约是 Pbft 的 2.5 倍。

5.2 集群中副本数量的影响

每个集群中副本数量对于性能的影响如图5-2所示。为了测量集群中副本数量对于性能的影响，需要保证集群总数不变，集群中副本的数量分别选取 $\mathbf{f}=1$，2，3，4，5，所以单个集群中副本数量 $\mathbf{n}=3\mathbf{f}+1=4$，7，10，13，16。

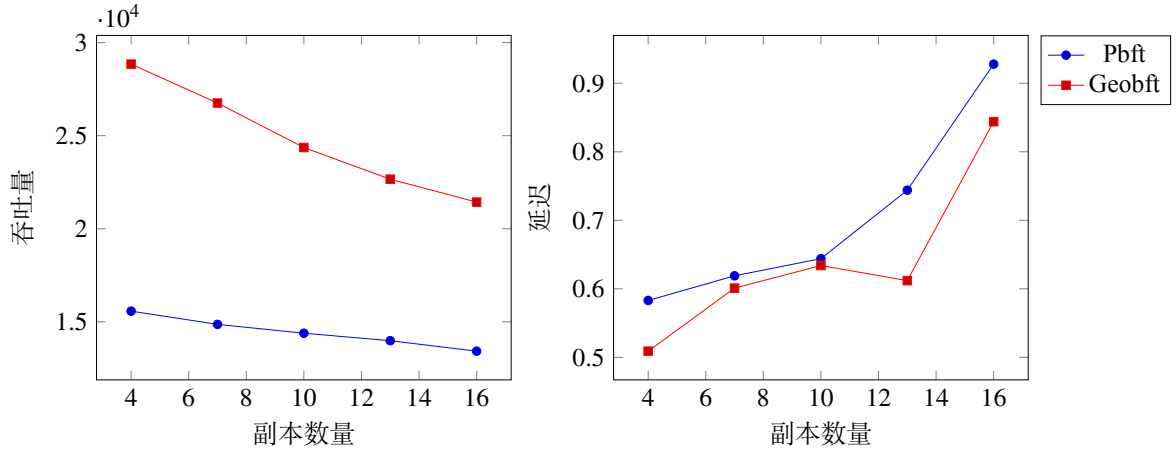


图 5-2　集群中副本数量的影响

Figure 5-2　Impact of the number of replicas in the cluster

可以看到，增加集群中的副本数量对于 Pbft 的影响并不是很大。Pbft 中的通信瓶颈在于主节点与其他集群的单个副本节点之间的远程通信速度哦，仅仅增加副本数量并不会改变远程通信速度，其性能的下降是由于系统中副本节点增多带来的通信次数增多导致。而地理分布共识算法中，随着单个集群中副本节点的增多，在全局共享阶段需要构造的提交证明也会变大。因此会造成全局共享阶段本地主节点与其他集群间副本节点的消息传递时间增长，从而导致系统的性能下降。

5.3 批大小的影响

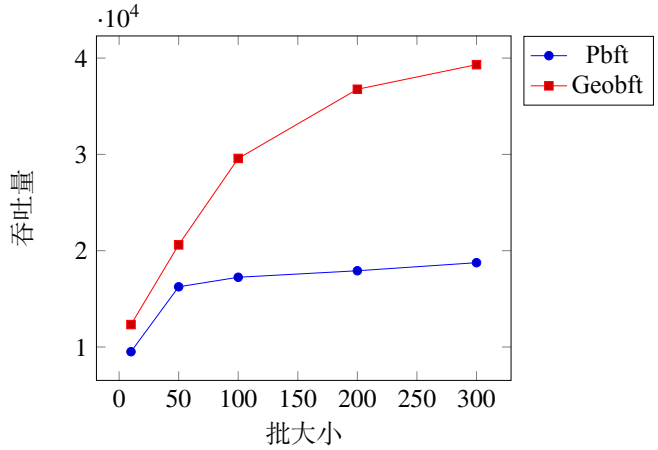


图 5-3　批大小的影响

Figure 5-3　Impact of Batch size

批大小定义为单次共识决策中处理的客户端请求的数量。为了测量批大小对性能的影响，需要保证集群数量和单个集群中的副本数量不变。批大小分别取 10，50，100，200 和 300，测量结果如图5-3所示。

由于系统中集群数量与单个集群中的副本数量都是固定的，因此系统通信复杂度也是固定的，只需要考虑批大小对于吞吐量的影响即可，测量如图5-3所示。在 Pbft 中，由于存在中心化的主节点，系统的带宽限制在该单一主节点上。随着批大小的增大，主节点带宽成为系统性能瓶颈。而地理分布共识算法中，每个集群内都有本地主节点，在整个系统中分散了共识，消除了单个主节点带宽的限制，从而能够支持更大的批，性能约为 Pbft 的两倍。

6、总结展望

本文实现并对比分析了地理分布共识算法和实用拜占庭容错共识算法的性能。地理分布共识算法通过利用网络拓扑关系，将本地副本分组到同一个集群中，通过分散共识消除了实用拜占庭容错共识算法中由于单个主节点集中式协调共识决策造成的性能瓶颈。除此之外，在保证能够可靠探测到故障的情况下，最大限度的减少了全局间通信。通过对比地理分布共识算法和 Pbft 共识算法在几个方面的性能，可以看出，地理分布共识算法拥有比 Pbft 更高的性能。因此，地理分布共识算法能够支持区块链系统的跨地理规模部署。尽管如此，地理分布共识算法中仍然有可以改进的地方。在全局共享阶段，所生成的提交证明的大小会影响该阶段的系统性能，在下一阶段可以尝试使用阈值签名[16]通过生成单个恒定大小的阈值签名来解决该问题。另一方面，在利用网络拓扑信息划分集群上，如何更合理的将节点划分到一个集群中，能够更好的提升系统的性能也有待展开研究。除此之外，包括地理分布共识算法在内的拜占庭容错算法目前都仅支持在许可区块链系统中使用，如何让其也能够在非许可链中使用也是未来的一个研究方向。

参考文献

[1] DWORK C, NAOR M. Pricing via processing or combatting junk mail[C]//Annual international cryptology conference. 1992: 139-147.

[2] JAKOBSSON M, JUELS A. Proofs of work and bread pudding protocols[G]//Secure information networks. Springer, 1999: 258-272.

[3] KING S, NADAL S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. Self-published paper, August, 2012, 19(1).

[4] CASTRO M, LISKOV B, et al. Practical byzantine fault tolerance[C]//OsDI: vol. 99: 1999. 1999: 173-186.

[5] LI W, FENG C, ZHANG L, et al. A scalable multi-layer pbft consensus for blockchain[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 32(5): 1146-1160.

[6] YIN M, MALKHI D, REITER M K, et al. HotStuff: BFT consensus in the lens of blockchain[J]. ArXiv preprint arXiv:1803.05069, 2018.

[7] YIN M, MALKHI D, REITER M K, et al. Hotstuff: Bft consensus with linearity and responsiveness[C]//Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. 2019: 347-356.

[8] AMIRI M J, AGRAWAL D, EL ABBADI A. Sharper: Sharding permissioned blockchains over network clusters[C]//Proceedings of the 2021 International Conference on Management of Data. 2021: 76-88.

[9] GUPTA S, RAHNAMA S, HELLINGS J, et al. Resilientdb: Global scale resilient blockchain fabric[J]. ArXiv preprint arXiv:2002.00160, 2020.

[10] 宋双杰. 主流区块链共识机制的简介与比较——区块链技术引卷之三[R]. 2018.

[11] HERLIHY M. Blockchains from a distributed computing perspective[J]. Communications of the ACM, 2019, 62(2): 78-85.

[12] ÖZSU M T, VALDURIEZ P. Principles of distributed database systems[M]. Springer, 1999.

[13] TEL G. Introduction to distributed algorithms[M]. Cambridge university press, 2000.

[14] 陆歌皓, 谢莉红, 李析禹. 区块链共识算法对比研究[J]. 计算机科学, 2020, 47(s1): 332-339.

[15] CASTRO M, LISKOV B. Practical Byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems (TOCS), 2002, 20(4): 398-461.

[16] SHOUP V. Practical threshold signatures[C]//International Conference on the Theory and Applications of Cryptographic Techniques. 2000: 207-220.

[17] 刘炜, 阮敏捷, 佘维, 等. 面向物联网的 pbft 优化共识算法[J]. 计算机科学, 2021, 48(11): 151-158.

[18] 王同贺, 华昊辰, 曹军威. 共识边缘计算及其在能源互联网中的应用[J]. 电力建设, 2021, 42(02): 116-125.

附录

致谢

五年时间，转眼间匆匆而过。

犹记得 2017 年 6 月 8 号下午考完英语，收拾好东西坐在回家的车上，看着校园里熙熙攘攘的人群，一个个大包小包提着东西，就此别离，突然间流下眼泪来，吓得我妈以为我没考好赶紧安慰我。那时的高中校园，高一高二的校园，也是如华师大这般，郁郁葱葱。高三的校园虽没了绿树成荫，但回想起高三，浮现的总是阳光洒落在走廊里背书的同学身上的画面。

对于在这里生活了五年之久的华师大，也有着一些自己与她的独特联系吧。文史楼前消失的那棵树，丽娃停车场角落里不见了的海宝，大活旁边的小假山，丽虹桥底系着的小船，那只蹭了我一裤子毛的三花，以及为数不多但很漂亮的槭树，当然，还有华师大里可爱的人们。我不长于交际，更多的时候喜欢坐在旁边，坐在远处，看着人来人往。但，偶尔又会离开旁观者的位置，在另一个毕业季的晚上，与打印店小哥和一个俄罗斯女生畅谈到深夜。细细想来，平淡枯燥在期末赶 ddl 的生活中也不乏一些回想起来嘴角上扬的片段。不过，我现在最想做的，是能够到草坪上打滚！

好了，回到正题。对于即将完成的毕业论文，十分感谢在选题，开题，实验和写作中给予帮助的范维学长和张召老师。范维学长虽然有一点 shy，但是喜欢跑步和邓丽君的男生，应该自有一方天地吧！有一位豆友，也喜欢邓丽君，是一位有着自己人格魅力的低调自律不张扬有血有肉有温度的男生。张召老师是一位认真负责的老师，在疫情初始网课期间由于一些生活问题也和张召老师谈过几次话，而后在考研选择研究生导师的谈话中也能体会到张召老师的关心。当然，我们院的每位老师都很关心同学们的学习以及日常生活！在这里还要感谢一下辅导员杨兴龙老师，在疫情隔离期间麻烦了杨老师好多事情，谢谢杨老师！除此之外，还要感谢孙老师（感觉称呼孙秋实同学似乎有点遥远的距离感，还是写孙老师吧哈哈哈）提供的 LaTeX 模板，让我能够免于 Word 调格式的痛苦，感谢！

除了毕业论文之外，由于疫情封校的原故，在准备考研复试环境上也是一波三折。因此，十分感谢借给我电脑用的梁辉哥和吴佳威学妹以及借给我手机支架的于佩民小朋友，感谢他们让我能够顺利完成考研复试！

五年的时间里，在这里认识了不少的同学，也有几位好友可以时不时出来一起吃个饭。希望已经毕业的他们不要受到疫情的过多干扰，也希望疫情能够早点过去，恢复疫情前的生活状态！

现在的我，与五年前的我，总归是不太一样的。对于这五年来的学习生活时光，不能说很满意，也不算太差。前几天看到一本书的书评中有一句话：“……为荒废过许多青春时间而悲哀。但还好学无止境，受用于此书，自当勤勉”。

学无止境，受用于此，自当勤勉。

二〇二二年四月二十三日，华东师范大学中北三馆