基于偏向稳定的分布式审计因果一致性模型

田俊峰[1,2]，杨乾宇[1,2]，Jitian Xiao[3]

（1. 河北大学网络空间安全与计算机学院，河北 保定 071002；
2. 河北省高可信信息系统重点实验室，河北 保定 071002；3. 伊迪斯科文大学科学学院，君达乐 WA6027）

摘 要：在分布式存储中，因果一致性由于易编程与性能权衡最佳而备受青睐。为解决现有因果一致性成果中矢量依赖跟踪损失吞吐量的问题，提出了基于偏向稳定的分布式审计因果一致性模型。在查询操作中使用组合矢量时间戳替代全矢量时间戳，减少系统管理与通信开销。同时，将分布式关联数组引入因果审计，分区协同审计细化数据依赖性，以减少虚假依赖条目数量。理论分析与仿真结果表明，所提模型可提升吞吐量 48.26%，降低更新响应时延 16.25%。

关键词：数据一致性；因果一致性；分布式存储；偏向稳定；因果审计

中图分类号：TN92

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023059

Distributed audit causal consistency model based on biased stability

TIAN Junfeng[1,2], YANG Qianyu[1,2], Jitian Xiao[3]

1. School of Cyber Security and Computer, Hebei University, Baoding 071002, China
2. Key Laboratory on High Trusted Information System in Hebei Province, Baoding 071002, China
3. School of Science, Edith Cowan University, Joondalup WA6027, Australia

Abstract: In the distributed storage, causal consistency is favored due to the best trade-off between ease of programming and performance. To address the problem of vector-dependent tracking loss of throughput in existing causal consistency results, a distributed audit causal consistency model based on biased stability was proposed. Combined vector timestamps were used instead of full vector timestamps in query operations to reduce system management and communication overhead. Meanwhile, the causal auditing was introduced with the help of distributed associative arrays, and data dependency was refined by partitioned cooperative auditing to reduce the number of false dependency entries. Theoretical analysis and simulation results show that proposed model improves throughput by 48.26% and reduces update response latency by 16.25%.

Keywords: data consistency, causal consistency, distributed storage, bias stability, causal audit

0 引言

在分布式存储系统中，服务供应商将数据副本分布在世界各地的不同节点上以获得更高的可伸缩性和可用性[1]。数据的分布式存储虽提高了系统可用性，却引发了副本间数据一致的问题。而数据之间是否一致是判断数据准确、安全、合规的基准，且其重要性随着对分布式数据及移动设备可访问性的增强而逐步增长。因此，如何保证数据一致性对分布式存储数据可信至关重要。

数据一致性一般分为强一致性与弱一致性。在理想状态下，更新过的数据对后续的访问即时可见，则称其为强一致性[2]。强一致性虽然有简单的语义，却引发了高时延和不允许网络分区现象。而弱一致性能在网络分区的情况下容忍客户端访问不到后续的部分或者全部更新。鉴于网络分区在分

收稿日期：2022–12–12；**修回日期：**2023–02–23

通信作者：杨乾宇，13753447065@163.com

基金项目：河北省自然科学基金资助项目（京津冀基础合作专项）（No.F2021201058）；河北省自然科学基金资助项目（No.F2021201049）

Foundation Items: The Natural Science Foundation of Hebei Province (Beijing-Tianjin-Hebei Basic Cooperation Special Project) (No.F2021201058), The Natural Science Foundation of Hebei Province (No.F2021201049)

布式环境下是必然存在的，因此，一些社交软件选择使用弱一致性实现数据可信。在众多弱一致性模型中，因果一致性[3]由于易编程性与性能权衡达到最佳点而广受关注[4]。

因果一致性所采用的性能权衡方式是在网络分区的情况下允许节点在相对顺序上存在分歧，仅在因果相关的操作顺序上要求全局共识[5]。实现全局共识的因果一致性模型大多通过依赖关系检验来验证因果序。依赖关系检验一般分为显式依赖检验与隐式依赖检验。显式依赖检验[6]仅跟踪拥有最新时间戳的依赖项的特性，在最坏的情况下，需要向所有的分区发送检查消息，牺牲了系统性能与可扩展性。隐式依赖检验允许客户端进行因果一致的读取任务而不需要显式依赖检验，因此被广泛应用于因果一致性模型中。隐式依赖检验一般通过标量依赖跟踪与矢量依赖跟踪 2 种方法实现。其中，标量依赖跟踪将数据项的因果关系汇总至单一时间戳，增加了虚假依赖条目，从而引发高时延。而矢量依赖跟踪通过增加元数据开销与分区重载次数降低时延，却损失了吞吐量。

针对矢量依赖跟踪损失吞吐量的问题，本文在实现完全地理复制的场景中引入偏向稳定向量的概念与因果审计的策略，提出了基于偏向稳定的分布式审计因果一致性（CCBAS, causal consistency based on biased audit stable distributed）模型。在 CCBAS 模型中，用户查询时采用组合标量时间戳与数据中心（DC）进行通信，以此提升吞吐量。在用户写入时，CCBAS 模型在矢量依赖跟踪的基础上增加因果审计，减少更新时延。因此，CCBAS 模型与现有模型相比，在吞吐量和更新时延上有更优的性能表现。本文贡献如下。

1) 提出 CCBAS 模型并实现偏向稳定向量（BSV, biased stable vector），BSV 使用组合向量代替查询操作中的全向量。因此，CCBAS 模型通过在查询操作中减少元数据开销，实现了吞吐量的提升。

2) CCBAS 模型采用因果审计细化数据依赖性，加快本地更新可见过程，降低系统更新响应时延。同时，CCBAS 模型通过在矢量依赖跟踪的基础上增加因果审计以减少分区重载次数与虚假依赖数量，达到降低更新可见时延的目的。

1　相关工作

分布式数据存储系统作为云计算基础设施的一部分，在网络服务中发挥着至关重要的作用。为满足系统严格的性能和可用性需求，数据存储通常采用地理复制的方式。地理复制通过将客户端请求转发至最近的数据副本来减少时延[7]，并通过将副本分布到世界各地来实现可用性。地理复制通常分为完全地理复制和部分地理复制 2 种类型[8]。完全地理复制规定节点存储的数据均相同，能为数据提供容灾保障和高效、稳定的数据存取服务。

在基于地理复制的分布式数据存储系统中，需要着重考虑的是如何有效保证数据副本之间的一致性[9]。Lamport[10]在 1978 年首次提出了因果一致性的概念，并使用逻辑时钟来解决分布式数据存储系统中区分事件发生的依赖检验问题。但由于逻辑时钟不能通过比较时钟大小推出事件发生的先后顺序，2014 年，Du 等[11]提出了一种基于物理时钟的因果一致性协议（GentleRain），该协议使用物理时间戳进行依赖检验，同时为多数据中心的一致复制提供了低时延。但是，在同步过程中，分区之间若存在时钟漂移，物理时钟会造成系统时延写入。针对物理时钟实际应用无法倒退造成的写入时延问题，Kulkarni 等[12]提出了一种时钟同步方法——混合逻辑时钟（HLC, hybrid logical clocks）。HLC 使用逻辑时钟与物理时钟的组合在实际时间写入的基础上提供因果一致性依据。但混合逻辑时钟和传统时钟方法一样，使用单一标量时间戳难以避免引入虚假依赖项。为解决上述问题，Roohitavaf 等[13]在 2017 年提出了一种使用混合逻辑时钟和数据中心稳定向量（DSV, data center stable vector）的因果一致性模型（CausalSpartan），使用向量技术实现矢量依赖检验，有效减少了虚假依赖项，降低了模型对更新响应时延和更新可见时延的影响。但该技术的依赖性检验逻辑复杂[14]，分区重载次数多，并不能最大限度地降低更新可见时延。

在因果依赖检验的基础上，设计副本与客户端之间通信协议的方案虽然降低了用户操作的等待时延，但却无法避免丢失更新、脏数据和幻读等风险。因此，部分学者结合事务的 ACID（atomicity-consistency-isolation-durability）特性对用户操作的数据因果一致性进行了研究。2015 年，Dziuma 等[15]提出了事务存储器的概念，规定用户只能读取已挂起事务写入的值。为了减少事务性存储系统开销，Zhang 等[16]提出不一致复制的事务应用程序协议（TAPIR），消除了复制协议中的一致性，提供了非一致性下的容错性，同时仍然为应用程序提供强一

致性的分布式事务。TAPIR 在没有对分布式事务集中协调的情况下对其排序。与传统系统相比，TAPIR 能提供更低的时延和更高的吞吐量。Cure 是 Akkoorath 等[17]于 2016 年提出的一种通过一个交互式的事务接口来保证因果一致性的模型，在基于地理复制的分布式数据存储系统中保持高度可用性。此外，Cure 利用向量实现跟踪因果一致性的同时降低更新可见时延。但协调器时间戳的选取规则会导致死锁状态产生。为解决 Cure 产生的死锁问题，2018 年，Spirovska 等[18]提出实现非阻塞读取操作的事务因果一致性模型（Wren），对事务的执行、依赖性跟踪和稳定性提出了新协议。Wren 通过提供一种由 2 个标量时间戳组成的新型快照和客户端缓存实现无阻塞读取操作。2 个标量时间戳中，一个时间戳跟踪本地项的依赖关系，另一个时间戳跟踪远程项的依赖关系。与之前的设计相比，2 个标量时间戳提高了系统可扩展性与资源利用率。但是，由于远程时间戳选取问题，快照易受慢速副本的影响，同时 Wren 的写入操作缓存在客户端中，增加了客户端负载，对客户端要求较高。2019 年，Spirovska 等[19]提出的 PaRis 模型将无死锁应用于部分地理复制，但在 PaRis 模型中使用所有分区最小值作为快照时间戳，无法避免因标量时间戳导致的高更新可见时延。除此之外，PaRis 数据仍然向所有数据中心同步，产生了不必要的消息冗余。为在元数据通信和更新可见性之间进行合理的权衡，Roohitavaf 等[20]在 2018 年对原有的 CausalSpartan 模型进行改进，提出了支持事务只读一致性的 CausalSpartanX 模型。与文献[11]等的模型不同，CausalSpartanX 实现事务性只读操作，其性能优点是不会受到慢速不相关分区（即一个不承载事务请求的任何密钥的分区）的影响，但由于使用向量元数据引入了损失吞吐量的问题。为解决这个问题，Kakwani 等[21]在 2020 年提出的 Orion 模型针对客户端获取服务器快照的方式进行改进。其在客户端使用预测读取时间戳机制取代依赖服务器的稳定向量。在最好的情况下，Orion 模型只需要一轮通信；但如果客户端预测错误，Orion 则仍需要两轮通信。与此同时，Orion 模型预测机制位于客户端，对客户端的要求较高且存在预测成功率低的问题。表 1 将不同因果一致性模型主要属性进行对比，其中，M 为数据中心数目。在使用标量时间戳的一致性模型中，依赖元数据被整合到单一时间戳进行因果关系验证，从而导致存在大量无因果相关性的更新（即虚假依赖），使对应操作时延增加。而在改进的 Wren 模型和 PaRis 模型中，虽然本地更新和远程更新分离，但是时间戳的选取规则仍会导致大量虚假依赖，因果依赖关系未得到有效细化。

2 基础知识

在分布式数据存储系统中，多个计算机节点通过网络互连对外提供整体存储服务。然而，随着互联网中用户资源访问的数量日益增多，若仅有一份数据，网络容易出现拥塞，而处理能力有限的节点也会因为访问数量太大导致宕机。因此，为提升系统的性能和可靠性，需要为每个数据进行备份。

2.1 完全地理复制

地理复制旨在将数据副本分布在客户端附近来减少操作时延，并通过多数据中心存储提高可用性。分布式数据存储，即存储设备分布在不同的地理位置且存储数据选择就近存储，以减轻带宽压力。当某个副本内的信息发生更新后，及时向处于其他节点的副本同步，维持系统中所有副本之间的一致性。在分布式数据存储中，当数据集被完全复制到 M 个数据中心（即副本）时即完全地理复制。基于完全地理复制的分布式模型如图 1 所示，其中每个数据中心被划分为 N 个分区。本文假设数据中心内部不会发生网络故障。因此，数据中心内的分

表 1 不同因果一致性模型主要属性对比

模型	一致性	死锁	时钟方法	时间戳	时延	依赖元数据
GentleRain	因果一致性	×	物理	标量	大	1
CausalSpartan	因果一致性	×	混合逻辑	矢量	较小	M
Cure	事务因果一致性	√	混合逻辑	矢量	较小	M
Wren	事务因果一致性	×	混合逻辑	标量	较大	2
PaRis	因果一致性	×	混合逻辑	标量	大	2
CausalSpartanX	因果一致性	×	混合逻辑	矢量	较小	M
Orion	因果一致性	×	混合逻辑	矢量	较小	M

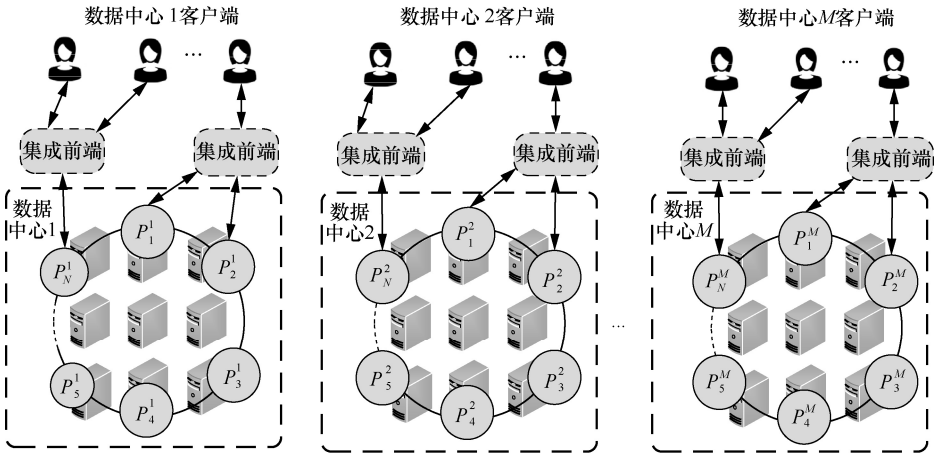


图 1　基于完全地理复制的分布式模型

区总是可以相互通信的。

2.2　因果一致性

因果一致性是基于 2 个事件之间发生的 happens-before 关系[10]而实现的。因果一致性模型是顺序一致性的淡化，与顺序一致性[22]不同的是，它按潜在因果关系来区分并排序各事件。对于 2 个操作 a 和 b，b 因果依赖于 a，记作 $a \rightarrow b$，当且仅当至少有以下一个条件成立。

1) a 和 b 是一个执行线程中的操作，a 发生在 b 之前。

2) a 是写入操作，b 是读取操作，b 读取 a 写入的值。

3) 存在其他操作 c，其中 $a \rightarrow c$，$c \rightarrow b$。

用小写字母表示键，如 x，用相应的大写字母表示键的某个版本，如 X。如果 X 的读取因果依赖于 Y 的写入，则称 X 依赖于 Y。

定义 1　因果序。假设 v_1 是键 k_1 对应的版本，v_2 是键 k_2 对应的版本，如果 $\mathrm{PUT}(k_2,v_2) \rightarrow \mathrm{PUT}(k_1,v_1)$ 成立，那么认为 v_1 因果依赖于 v_2，或者 v_2 是 v_1 的一个因果依赖项，记作 $v_1 \,\mathrm{dep}\, v_2$。

定义 2　数据项稳定。连接在本地数据中心的客户端，需要等待数据项所有依赖项可见时，才能读取数据项的当前更新。即当该项的所有因果依赖项都应用于本地数据中心中并可见时，该项才会稳定[23]。

定义 3　更新响应时延。因用户写入时需要验证因果序，则一个项目 d 从客户端发出请求至接收到项目 d 稳定的时间差称为更新响应时延。

定义 4　更新可见时延。一个项目 d 从在原始数据中心中被创建到项目 d 在远程数据中心中可见的时间差称为更新可见时延。需要注意的是，要使键值远程可见，数据中心需要访问元数据，并检查其所有因果依赖关系之前也已在本地可见。

2.3　查询放大

随着服务器的物理时钟漂移，操作等待的发生率和数量都会增加。这种敏感性问题在实践中会变得更严重，因为单个终端用户请求通常会转换为许多可能具有因果关系的内部查询，这种现象称为查询放大[13]。在这种情况下，任何内部查询中的任何时延都会增加终端用户感知的最终响应时间，并影响用户最终体验。尤其在云存储服务供应商为成千上万个用户提供并发服务时，查询放大的现象更加明显。

2.4　last-writer-wins 冲突解决策略

基于时钟同步协议，相同键值的并发写入导致数据同步的不一致，即冲突问题。由于混合逻辑时钟与物理时钟非常接近，因此在并发更新的情况下，为了避免由于并行冲突导致的数据中心不一致问题，一致性模型通常使用 last-writer-wins 函数来处理操作之间的冲突。last-write-wins 规则细化为根据给定更新的时间戳，将更新进行排序，高时间戳更新成为优胜者。在时间戳一致的情形下，通过对应项的原始数据中心 id 以解决冲突。与此同时，模型使用 last-writer-wins 函数处理冲突可以很容易地集成其他机制来实现状态收敛。

3　CCBAS 模型

CCBAS 模型所实现的通信协议是在分布式存储环境中稳定运行的，目的是在满足因果一致性的要求下为用户提供快速高效的数据服务。本节介绍 CCBAS 模型的设计与实现。传统上，隐式依赖跟踪

使用全局时间戳机制进行因果序验证。这避免了显式依赖跟踪由于全分区检验带来的系统开销与可扩展性限制。在隐式依赖跟踪的基础上，CCBAS 模型将元数据进行分离操作。旨在基于隐式依赖跟踪进行元数据序列化从而降低更新插入操作受同一时间不同依赖关系影响的程度。与此同时，CCBAS 模型在使用全局时间戳机制的基础上，通过偏向稳定向量与因果审计相结合来提升系统性能。

CCBAS 模型的 GET 与 PUT 操作匹配不同的元数据策略，由于矢量时间戳用于在插入操作中表征因果依赖性，在 GET 操作元数据不再携带全向量矢量时间戳，减少了元数据大小，以此提升系统性能。与此同时，PUT 操作在矢量时间戳的基础上，与因果审计相结合，进一步细化因果依赖性。每个数据中心维护属于自己的分布式关联数组（DAA, distributed associative array），DAA 仅存储本地数据中心增量更新的元数据。DAA 对数据元数据进行因果审计，数据项本身仍为更新后即刻传播。数据中心之间通过 Gossip 协议共享 DAA，这使所有 DAA 以指数级速度达到一致。CCBAS 模型简要架构如图 2 所示。

3.1　元数据

对于每一个写入操作，分区生成该更新的唯一标识符 *u*.id。在一个会话中，客户端存储 *u*.id 捕获依赖性。该标识符由源副本 sr、键值、更新时间戳、依赖集（DS, dependency set）和稳定远程向量（SRV, stable remote vector）组成。SRV 为一个向量时间戳，包含了系统中所有数据中心的时钟条目，若在数据中心 i 中的 SRV[j]=t，则意味着所有在时间 t 之前写入数据中心 j 的更新已经在数据中心 i 中可见。客户端存储分布式依赖集（DDS, distributed dependency set）中，每个数据中心 i 仅有一个 h 条目，指向客户端获取到该数据中心的最新稳定值。客户端同时存储偏向稳定向量（BSV, biased stable vector），即<local, D-mst>，在进行 GET 操作时使用 BSV 封装元数据。

3.2　具体操作

本节介绍 CCBAS 模型执行的具体操作。基于指定的键 k，对客户端和服务器如何执行 GET/PUT 操作进行了描述，对模型如何计算 SRV、审计和发送心跳消息进行了描述，对 CCBAS 模型的简要操作进行了描述。在进行 GET/PUT 操作时，CCBAS 模型不再单一地使用标量或矢量时间戳，而是针对不同的操作类型匹配不同的元数据策略。在进行 GET 操作时，客户端发送 BSV 以表示 GET 操作的因果依赖关系，对应分区接收到 GET 请求后，查询并更新本地依赖关系，将符合因果依赖的更新与时钟关系反馈给对应客户端。客户端接收到来自对应分区的

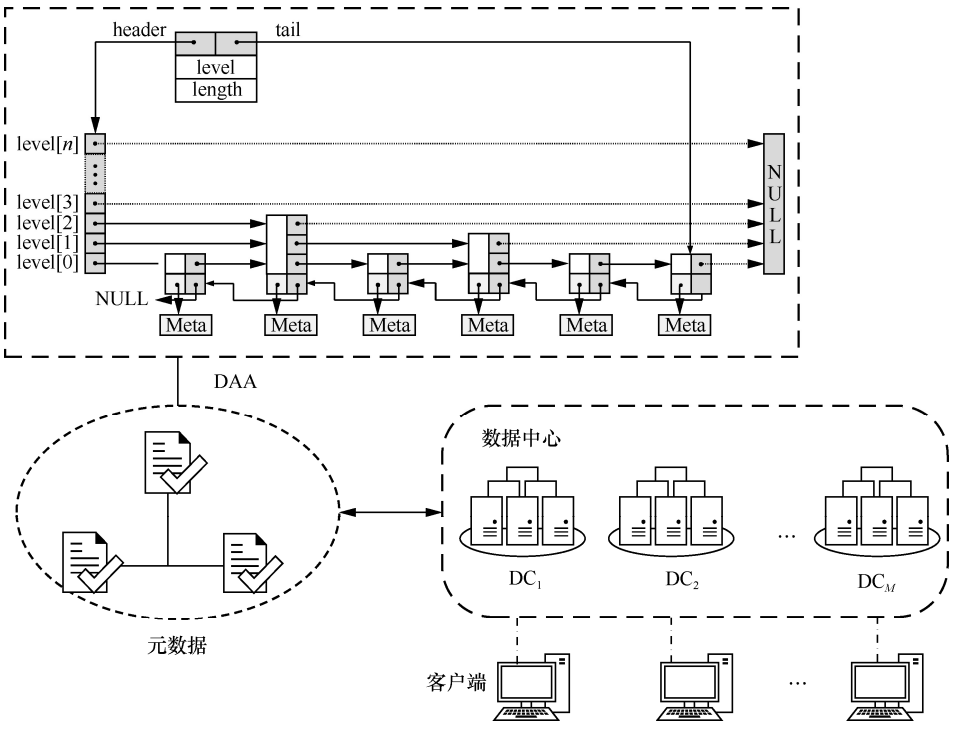


图 2　CCBAS 模型简要架构

更新后更新 BSV，以保证后续操作的因果性。在 PUT 操作中，CCBAS 模型将元数据与数据分离，本地数据插入完成后，将元数据传递至 DAA 中，使数据在远端分区进行数据同步时，接收到已序列化的依赖关系，从而减少分区重载次数。

在模型运行过程中，每个分区访问一个单调递增的物理时钟 Clock_n^m。相关模型符号及定义如表 2 所示。

表 2　　相关模型符号及定义

符号	定义
N	分区数
M	每个分区副本数
P_n^m	第 m 个副本的第 n 个分区
$\mathbf{DDS}_c$	客户端 c 的分布式依赖向量集
$\mathbf{SRV}_n^m$	P_n^m 的稳定远程向量
$\mathbf{SRV}_c$	客户端 c 的稳定远程向量
Clock_n^m	P_n^m 的物理时钟
$\mathbf{SVV}_n^m$	P_n^m 的稳定版本向量
sr	源副本
dt	时间戳
dds	分布式依赖集
ds	依赖集
BSV	偏向稳定向量
D-mst	偏向时间戳

3.2.1　GET<*k*>操作

与传统因果一致性协议[11,18,20]不同，客户端 c 发出 GET<*k*> 请求前，元数据封装选择 $\mathbf{BSV}\text{-}<\text{local},\text{D-mst}>$。首先，分区 P_n^m 收到请求后，用 **BSV** 更新对应时间戳，确保安装的快照至少与客户端 c 目前提供的快照一样。然后，分区 P_n^m 选择键 k 稳定的最新版本 v。分区 P_n^m 用读取到的 v 的更新时间戳 dt 和源副本 sr 来更新 ds（算法 2 的步骤 4)）。P_n^m 将读取到的值 v、$\mathbf{SRV}_n^m$、D-mst 通过 GETREPLY 消息返回给客户端 c。客户端 c 接收到 GETREPLY 消息后更新 $\mathbf{SRV}_c$、$\mathbf{DDS}_c$、**BSV**，以此确保操作的因果序关系。

算法 1　客户端 GET<*k*>操作

输入　键 k，$\mathbf{BSV}_c$

输出　值 v，ds，$\mathbf{SRV}_n^m$，D-mst

1) function GET< *k*>
2) 发送<GETREQ k, $\mathbf{BSV}_c$ > to P_n^m
3) 接收<GETREPLY v, ds, $\mathbf{SRV}_n^m$, D-mst >
4) $\mathbf{BSV}_c$ local← max（$\mathbf{SRV}.i$, $\mathbf{BSV}_c$ local）
5) if D-mst>= $\mathbf{SRV}_c.m$//保持客户端因果序 $\mathbf{BSV}_c$ mst← D-mst*/
6) $\mathbf{SRV}_c \leftarrow \max(\mathbf{SRV}_c, \mathbf{SRV}_n^m)$ //根据数据中心最新稳定状态更新客户端状态
7) for each <i, h>∈dds
8) $\mathbf{DDS}_c$←maxDS(i,h, $\mathbf{DDS}_c$) //更新依赖集
9) return v

算法 2　服务器端 GET<*k*>操作

输入　键 k，$\mathbf{BSV}_c$

输出　值 v，ds，$\mathbf{SRV}_n^m$，D-mst^m

1) Upon receive<GETREQ k, $\mathbf{BSV}_c$ > from c do
2) $\mathbf{SRV}_n^m$ ←max（$\mathbf{BSV}_c$, $\mathbf{SRV}_n^m$ ）
3) 验证当前数据中心因果依赖关系 <dt, sr > $.d$.sr = m, or for any <i, h> ∈ d.ds, h ≤ $\mathbf{SRV}_n^m$ [i]
4) ds ← maxDS（ds,{<sr, dt>}）
5) 发送 <GETREPLY $d.v$, ds, $\mathbf{SRV}_n^m$, D-mst^m> to c

3.2.2　PUT(*k*,*v*)操作

客户端 c 将更新对应键值的存储分区 P_n^m。除键和值外，请求还包括客户端时钟 Clock_c。为确保时间戳单调递增，当 P_n^m 接收到请求时，首先通过获取 P_n^m 当前最大时间戳和 DDS_c 之间的最大值来计算更新时间戳（算法 4 的步骤 2)）。在完成键值对的插入后，将包含时间戳 dt 的标识信息异步发送到 DAA，DAA 对其进行统一因果审计后再发送至对应分区，而数据本身仍直接发送，这使更新在满足一致性要求的基础上减少了分区重载次数。在完成键值对插入后，分区向客户端返回标识信息，以更新 $\mathbf{DDS}_c$，完成键值对插入。

在 CCBAS 模型中，一个数据中心成功创建新条目后，该数据中心会通过 replicate message 把新条目发送到其他数据中心。同时将发送 replicate message 置于本地插入操作之前，以确保具有更小时间戳的版本被更快地复制。分区 P_n^m 收到来自分区 P_n^i 的<Replicate d>消息后，就把新条目添加到本地包含键 $d.k$ 的条目链中。

算法 3　客户端 PUT<*k*,*v*>操作

输入　键 k，值 v

输出　dt，sr

1) function PUT < k, v>//PUT 操作

2) 发送 <PUTREQ k, v, $\mathbf{DDS}_c$ > to P_n^m
3) 接收<PUTREPLY dt, sr>
4) $\mathbf{DDS}_c$ ← maxDS(sr, dt, $\mathbf{DDS}_c$)
//依赖集更新
5) function maxDS (数据中心 id i, 时间戳 h, dependency set ds)//依赖集更新操作
6) if ∃<i, h'> ∈ ds
ds ← ds − <i, h'>
ds ← <i, max(h, h')>
else ds ← ds ∪ {i, h}
7) return ds

算法 4 服务器端 **PUT<*k*, *v*>操作**

输入 键 k，值 v，dds

输出 dt，m，ds

1) Upon receive <PUTREQ k,v,dds>from c do
2) $\mathbf{SRV}_n^m$ ←maxDS（ $\mathbf{SRV}_n^m$, dds）
3) dt←max value in {dds.values∪{ $\mathbf{SRV}_n^m$ [m]}}
4) 更新 HCL(dt)
5) 新建项目 d
6) $d.k$←k
7) $d.v$←v
8) d.dt← $\mathbf{SVV}_n^m$ [m]//本地最新物理时间戳
9) d.sr ← m
10) d.ds ←ds
11) $d.p$←d.ds
12) 发送<PUTREPLY d.dt,m,d.ds> to CCBAS
13) for each P_n^i， i∈{0,⋯,m−1}， i≠m do
i. 发送<REPLICATE $d.p$> to P_n^i
14) 插入键值对<k,d>
15) 发送<PUTREPLY d.dt, m> to client
16) Upon receive <REPLICATE $d.p$>from P_n^i do//数据同步操作
17) 等待接收 d.ds from DAA^i
18) 远程数据中心插入<k,d>
19) $\mathbf{SVV}_n^m$ [i]←d.dt//已收到 P_n^i 时间戳小于 dt 的所有更新

3.3 数据同步策略

在 CCBAS 模型中，本地数据中心写入成功后触发数据中心间同步，其将数据与元数据分离，以此减少 DAA 存储开销。远程数据中心接收到元数据后，将完整数据项根据因果序插入本地的版本链。插入完成后，分区更新对应稳定版本向量 $\mathbf{SVV}_n^m[i]$（算法 4 的步骤 19)）。与稳定远程向量不同，稳定版本向量为数据中心内部分区之间当前更新状态。当 $k \neq m$ 时，$\mathbf{SVV}_n^m[k]$是分区 P_n^m 收到来自 P_k^m 更新的最新时间戳；当 k=m 时，$\mathbf{SVV}_n^m[m]$ 是对应分区 P_n^m 当前物理时间戳。

但在实际环境中，会存在一种特殊情况：由于未触发更新操作， P_n^m 不会向其对应副本发送 replicate message。因此，对应 **SVV** 和 **SRV** 的第 m 项无法单调递增。为了避免这种情况发生，规定若一个分区在Δt 时间内未发生更新，则将其当前 Clock$_n^m$ 发送给它的副本（算法 5 的步骤 13)）。

算法 5 审计策略、心跳与全局稳定策略

审计策略

输入 dt，m， SRV$_n^m$

输出 DAA

1) Upon receive<PUTREPLY d.dt, m, $\mathbf{SRV}_n^m$ > from P_n^m
2) list ← list ∪ <k, $\mathbf{SRV}_n^m$ >
3) Upon every Δt do//DAA 同步
4) 发送 DAA^i to DAA^j ($j = i$)
5) DAA^i ← ϕ

心跳与全局稳定策略

输入 无

输出 心跳同步

6) Upon every Δt do//全局稳定过程后触发计算 D-mst
7) $\mathbf{SRV}_n^m$ ← max($\mathbf{SRV}_n^m$,entry-wise $\min_{j=1}^{N}(\mathbf{SVV}_j^m)$)
8) D-mst^m ←middle($\mathbf{SRV}^m$)//近似中位数选取
9) Upon every Δt // Δt 时间内未接收数据同步请求
10) if there has not been any replicate message in the past Δ time updateClockOnHeartbeat()
11) for each server P_n^j, j ∈ {0,⋯ ,M−1}, $k \neq m$ do send<HEARTBEAT $\mathbf{SVV}_n^m$ [m]> to P_n^j
12) Upon receiving <HEARTBEAT hlc> from P_n^j do//响应心跳消息
13) $\mathbf{SVV}_n^m$ [j] ←Clock$_n^m$

3.4　全局稳定

由于运行全局稳定程序的代价高，因果一致性模型周期性运行全局稳定过程[24]。如图 3 所示，在 CCBAS 模型中，为减少消息冗余选择传播树进行分区稳定信息交换与广播。每隔单位时间Δt，数据中心内的分区共享它们的稳定版本向量。并将已知稳定版本向量的最小值计算为$\mathbf{SRV}_n^m$（算法 5 的步骤 7)）。分区计算完成后与其他对等分区共享，计算出数据中心稳定远程向量。稳定远程向量是向量时间戳，包含系统中所有数据中心的可见时钟条目。计算完成后将稳定远程向量推回到树中。若在数据中心 i 中，$\mathbf{SRV}[j]=t$ 等同于数据中心 j 中时间戳小于 t 的更新已经本地可见。

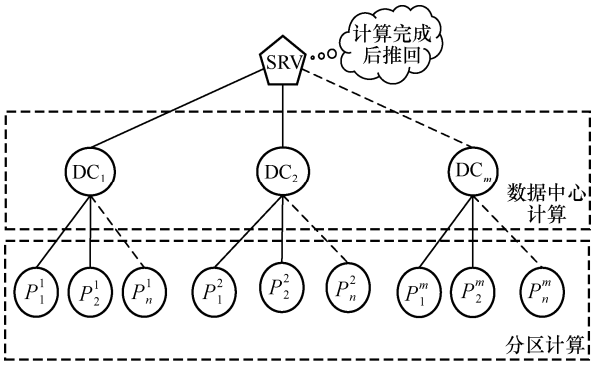


图 3　全局稳定框架

3.5　资源效率与因果分析

如图 4 所示，为客户端不同的操作适配不同的元数据策略，不仅可以有效减少 GET/PUT 操作的复杂度，还可以进一步细化数据间的因果依赖。

在 GET<*k*>操作中，相比于使用全向量进行读取操作的一致性模型，CCBAS 模型的读取操作使用部分组合向量在满足一致性要求的基础上有效降低了操作过程中的通信开销。同时部分组合向量通过时间复杂度为 $O(n)$的线性查找（BFPRT）算法来尽可能降低对系统性能的影响。

在 PUT<*k*,*v*>操作中，CCBAS 模型对元数据通过 DAA 进行针对性的依赖细化。在 DAA 中对相同键更新排序后采用 Gossip 协议，耗费时间极短，使更新操作实现分区协同审计。CCBAS 模型在向量技术的基础上增加因果审计策略减少了依赖检验的成本。这是因为在审计策略中，分区协同审计减少了虚假依赖数量，同时降低了因分区重载而导致因果关系验证开销，加快了数据收敛。

4　CCBAS 因果一致性实现

本节介绍 CCBAS 模型实现因果一致性的过程中所使用的因果审计策略与偏向稳定向量的具体设计。

4.1　因果审计策略

在分布式数据存储系统中，由于多个用户对于资源的访问是同时进行的，执行这些操作难免会违反一致性。与以往的因果一致性协议不同，CCBAS 模型在全局稳定过程的基础上仅对具有因果关系

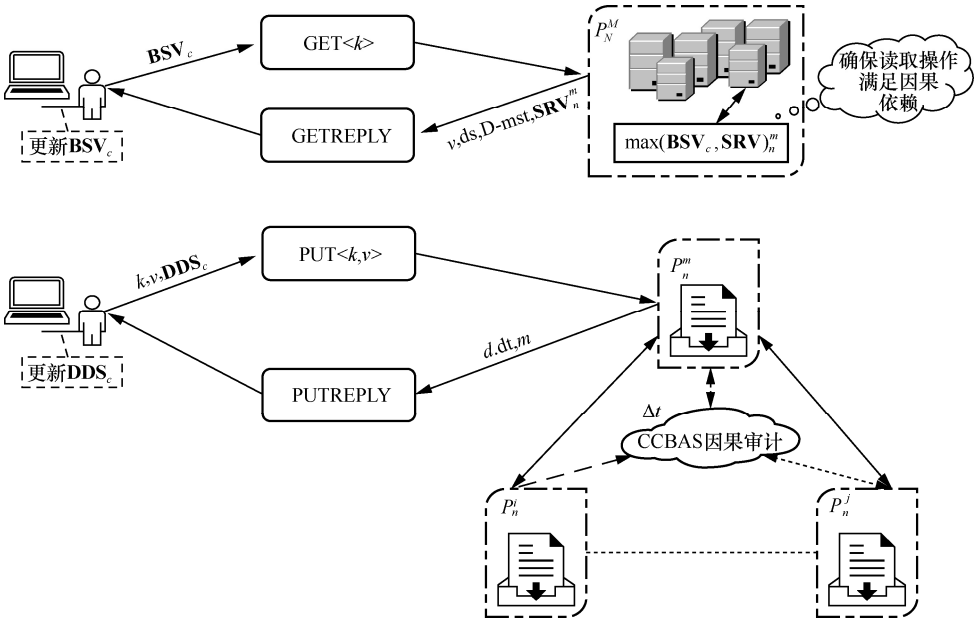


图 4　CCEAS 模型简要操作

的元数据进行审计后进行同步。通过因果审计，使本地数据中心的更新以更符合因果序的方式进行排序，且 CCBAS 模型接收的更新时间戳不会发生时钟倒退（分区内部的更新审计由分区自行判定）。

该审计策略分析相同用户的前一写入操作与使用相同资源的其他用户写入操作。将具备因果序列的写入操作按照键值、键值提交时间进行双重排序后，将审计结果注册在由 2 个哈希表组成的 DAA 中。DAA 的审计实现是借鉴跳跃表的思想对元数据进行操作。跳跃表的审计效率可以和平衡树相媲美，但是平衡树的插入和删除操作可能引发子树的调整，逻辑复杂，而跳跃表的插入和删除只需要修改相邻节点的指针。DAA 审计细化的操作因果序为

$$(o_1 = w(x)a) \wedge (o_2 = w(x)b) \tag{1}$$

$$(c_1 = w(x)a) \wedge (c_2 = w(y)b) \tag{2}$$

在式(1)中，对来自资源 x 的值 a 进行 o_1 的写入操作，然后对来自资源 x 的值 b 进行 o_2 的写入操作，这表示对相同资源的连续写入操作审计。则分区 p_j 上的资源 x 上写入 b 之前，已在分区 p_j 上的资源 x 上写入 a；在式(2)中，对于相同客户端对 x 进行写入操作后，对 y 值进行写入操作。则在分区 p_j 上的 y 值写入之前，已经在分区 p_i 上的资源 x 写入 a。

DAA 收到不同分区或非因果序的相同时间戳数据后，可以按任何顺序对其进行处理。同时为减少系统开销，CCBAS 模型周期性地将 DAA 结果采用 Gossip 协议，随机选择其他节点进行 DAA 结果同步，使 DAA 最快可以在 $O(\mathrm{lb}N)$的时间复杂度内收敛到一致。CCBAS 模型中因果审计流程如图 5 所示。

4.2 偏向稳定向量

CCBAS 模型使用 BSV，即<local, D-mst>进行读取操作。BSV 由 2 个标量时间戳组合而成，其中，local 为本地时间戳，D-mst 为偏向时间戳。通过本地与远程时间戳分离，BSV 使本地更新在无验证同步情况下先行。这避免了矢量依赖跟踪所造成的吞吐量损失与单一标量时间戳导致的高时延。

为避免实时计算 D-mst 导致的时延开销，CCBAS 模型周期性地根据数据中心稳定结果进行深拷贝与中值投票的组合操作计算偏向时间戳。且因 D-mst 对慢速副本等同步异常状态的不敏感性使偏向时间戳相比于 GentleRain 和 Wren 等模型中的数据同步更具有代表性（屏蔽慢速副本等同步异常状况）。其中，D-mst 由数据中心 id 与其稳定时间戳 2 个组件组成。

D-mst 的选取方式是 CCBAS 模型通过中值投票选取客户端已知的全局稳定时间戳近似中位数。该投票方式使用分治思想，将数据分成等份数据组，寻找分组中位数。当数据中心数目小于 5 时，使用插入排序选取其中位数作为偏向时间戳；当数据中心数目大于 5 时，在组内使用插入排序找到组内中位数。最终使用互递归的方法得出近似中位数。此算法借鉴 BFPRT 算法，将其数据中心稳定时间戳划分为$\frac{n}{5}$个大小相同组，多余部分舍去。通过寻找组内中位数可确保至少有$\frac{3n}{10}$个数比近似中位数大，使最终结果靠近中位数，最坏情况近似中值处于向量中的$\frac{3n}{10}$或$\frac{7n}{10}$部分。寻找偏向时间戳的时间复杂度为

$$T(n) \leqslant T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) \tag{3}$$

5 实验测试与分析

为了公平比较，CCBAS 的开发测试环境为 Intel（R） Core（TM） i5-4570,CPU 3.20 GHz，16GB 内存。云服务器均为运行 Ubantu16.0.4 的 Aliyun ecs.t6-c1m1.large 实例，2.5 GHz vCPU，2 GB 内存，40 GB 存储空间。

为测试协议性能，实验使用了 DKVF[25]平台来布置和管理分布式副本节点，同时与使用标量依赖的最新模型 CausalSpartanX、Wren、GentleRain、POCC 进行实验对比。DKVF 平台使用 Java 编程语

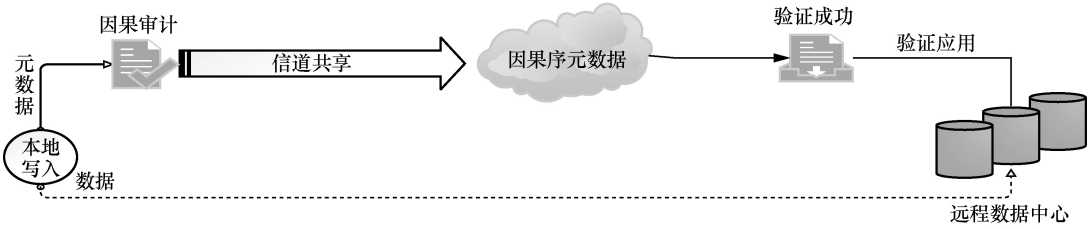


图 5 CCBAS 模型中因果审计流程

言和 Berkeley db 数据库。该数据库支持完整 ACID 事务语义，同时也提供 NoSQL 中简单的数据库编程接口[26]。

5.1　系统吞吐量

本节将评估 CCBAS 模型对于吞吐量性能的提升。将 CCBAS 模型运行在与 NTP（network time protocol）同步的不同物理机器、相同操作数的情况下进行交互操作。通过与 CausalSpartanX、GentleRain、Wren、POCC 进行对比测试吞吐量，计算 CCBAS 模型对吞吐量的优化。同时，测试仅使用 BSV 的 CCBAS-m 对于吞吐量的提升。

本节从 3 个方面对 CCBAS 模型进行吞吐量测试。

1) 设置数据中心数量为 3，分区数目为 4，客户端随机读取或更新从各个分区中随机选择的键，观察系统中的吞吐量变化情况，系统吞吐量如图 6 所示。在同等条件下，CCBAS 与 CausalSpartanX 相比吞吐量提高了 47.74%。仅使用偏向稳定向量的 CCBAS-m 模型与 CausalSpartanX 相比吞吐量提升了 36.29%。在进行 GET 操作时，CCBAS 模型在元数据中进行封装时通过深拷贝与中值算法提取部分向量代替全向量时间戳，在保证可见性的基础上减少了元数据大小，在同等条件下具有更好的吞吐量表现。在进行 PUT 操作时，与现存一致性模型不同，CCBAS 模型传递序列化的元数据，使对应更新进行插入时访问数据分区次数降低，加快因果依赖关系收敛，从而提升数据中心的处理速率。而 Wren 模型对数据中心进行周期性写入，而非实时性写入，增加了客户端负载损失系统吞吐量。与 Wren 模型相比，CCBAS 模型系统吞吐量提升 23.27%。通过 CCBAS 与 CCBAS-m 对比可知，通过协同审计排序细化数据依赖性，简化因果序验证可以进一步提升吞吐量。

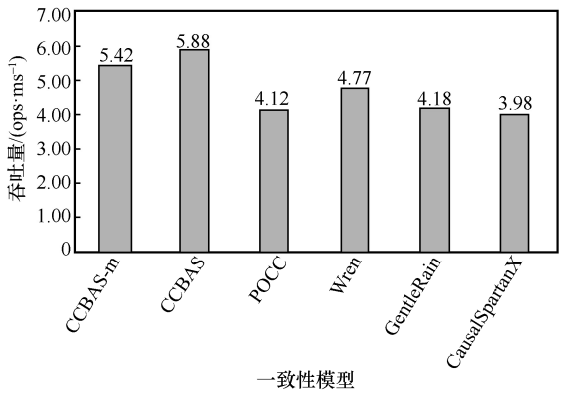


图 6　系统吞吐量

2) 为体现在不同工作负载 GET:PUT 中 CCBAS 模型的性能，本节使用工作负载 GET:PUT 比率为 95:5、80:20、50:50 对 CCBAS 模型吞吐量进一步测试。其中，50:50 的工作负载对应于基于 Internet 服务的更新较多的 YCSB（Yahoo cloud serving benchmark）工作负载，95:5 的工作负载对应于读取较多的 YCSB 工作负载。图 7 表明，在 3 种负载情况下，CCBAS 模型与 CCBAS-m 模型的性能提升较优。这体现了使用有限长度的偏向稳定向量可以有效降低元数据管理与通信开销，从而有效提升吞吐量。与此同时，随着写入负载的增大，CCBAS 模型中因审计操作提升的吞吐量从 95:5 的 3.98%提升至 50:50 的 19.95%。

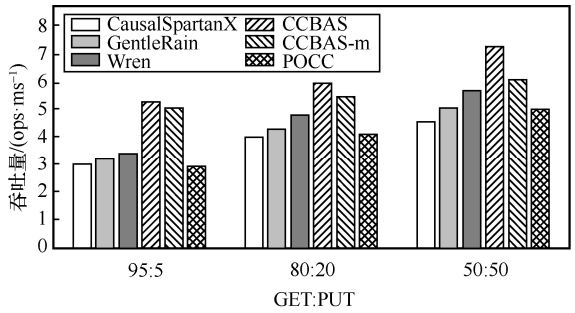


图 7　GET:PUT 对吞吐量的影响

3) 为对 CCBAS 模型进行吞吐量可扩展性分析，本节分别对数据中心数量和分区数目进行测试，观察 CCBAS 模型的可扩展性，结果如图 8 所示。在每个数据中心 2 个分区的情况下测试数据中心数目对吞吐量的影响，结果如图 8(a)所示。从图 8(a)可知，随着数据中心数量的增加，系统并发处理能力增强，吞吐量呈上升趋势。在数据中心数量为 6 时，CCBAS 模型与 CausalSpartanX 模型相比吞吐量提升了 44.4%。这是由于 CCBAS 在通过因果审计的方法加快写入操作的同时，通过组合向量取代全向量，降低元数据管理与通信的开销从而提升吞吐量。在同等条件下，CCBAS 模型与 Wren 模型相比吞吐量提升了 36.13%。这是因为 Wren 模型降低了元数据的管理开销，但在写入操作中使用单一远程标量时间戳，虚假依赖条目数量激增，增加了因果序的验证成本。而在 CCBAS 模型中，因果协同审计的方法使更新本地序列化后同步，减少了分区重载次数以提升系统性能。

在数据中心数量为 3 的情况下，测试分区数目对吞吐量的影响，结果如图 8(b)所示。从图 8(b)可知，在同等条件下，CCBAS 模型与 CausalSpartanX

模型相比吞吐量提升了 52.96%，CCBAS-m 模型与 CausalSpartanX 模型相比吞吐量提升了 39.6%。因 Wren 模型增加客户端存储开销，且经测量不同分区下客户端缓存平均读取成功率仅为 15.38%，增加其 GET 操作开销。除此之外，Wren 其仍使用标量时间戳验证因果序，增加 PUT 操作开销。故与 Wren 模型相比，CCBAS 模型吞吐量提升了 20.43%。

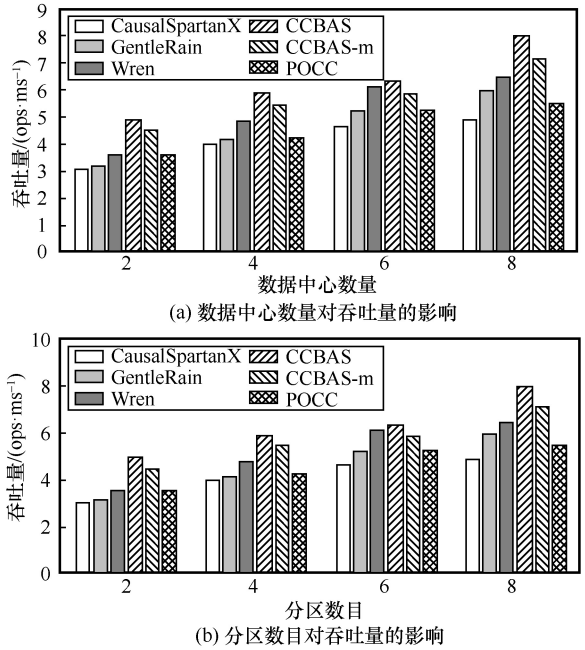


图 8 CCBAS 模型吞吐量可扩展性测试

5.2 更新响应时延

更新响应时延是分布式数据存储的重要方面。在因果一致性模型中，数据中心应用更新前需要访问元数据，检查因果关系后应用。更新响应时延是数据中心接收到本地更新到本地更新应用的时间差值。

本节将服务器运行在不同的物理机上，在不引入任何人为的时钟漂移的情况下，记录 6 个客户端以循环方式向服务器发送 1 000 个请求时更新平均响应时延的变化情况，分区数目对更新响应时延的影响如图 9 所示。从图 9 可知，更新响应时延会随着分区数目的增加而增加。其中，与以 GentleRain 为代表的物理时钟因果一致性模型相比，Wren 和 CCBAS 受分区数目影响较小，证明了采用混合逻辑时钟可以有效避免不同物理机时钟漂移的影响。同时，GentleRain 和 Wren 模型分别使用单一标量依赖跟踪和组合标量依赖跟踪验证因果序。尽管 Wren 模型通过本地与远程依赖分离的方式能大幅减少更新响应时延，但是因采用周期性轮循进行写入，相比于采用审计矢量因果检验的 CCBAS 模型，Wren 模型的更新响应时延提高了 79.02%；与以 CausalSpartanX 为代表的矢量依赖跟踪相比，尽管 CCBAS 模型因审计操作造成了 7.62%的读取响应时延，但 CCBAS 模型也因审计操作简化了因果序验证，同时序列化的因果关系进一步细化跟踪因果性使更新响应时延进一步降低。CCBAS 模型更新响应时延相比于 CausalSpartanX 平均降低了 16.25%。

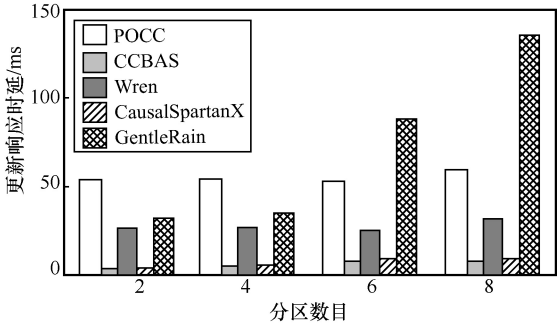


图 9 分区数目对更新响应时延的影响

5.3 查询放大

查询放大因子定义为单个请求生成的内部查询的数量。为了研究实际环境中不同的查询放大因子对更新响应时延的影响，本节实验在本地数据中设定 12 个服务端运行相同实验，目的是针对查询放大情况对请求操作时延进行精准分析。

查询放大因子对更新响应时延的影响如图 10 所示。图 10 反映了用户请求受查询放大情况的影响程度，查询放大情况越严重，更新响应时延越明显，因为每个最终请求都包含了几百个甚至更多的内部查询。从图 10(a)可知，随着查询放大因子的增加，GentleRain 的更新响应时延增长速度明显高于其余两者。在 GentleRain 中使用单一标量依赖导致虚假依赖条目更易受到查询放大因子影响，而 Wren 模型因远程依赖的时间戳选取问题，更新响应时延仍高于 CCBAS 模型。

CCBAS 模型和 CausalSpartanX 模型通过用矢量依赖跟踪来解决查询放大问题。CCBAS 模型引入审计机制减少分区载入次数，使查询对应更新因果应用的操作次数降低，从而减少内部查询对更新响应时延的影响。结果表明 CCBAS 模型在查询放大较严重的环境中减少写入时延是可行的。如图 10(b)所示，在同样条件下，CCBAS 模

型的更新响应时延比 CausalSpartanX 平均降低了 15.35%。

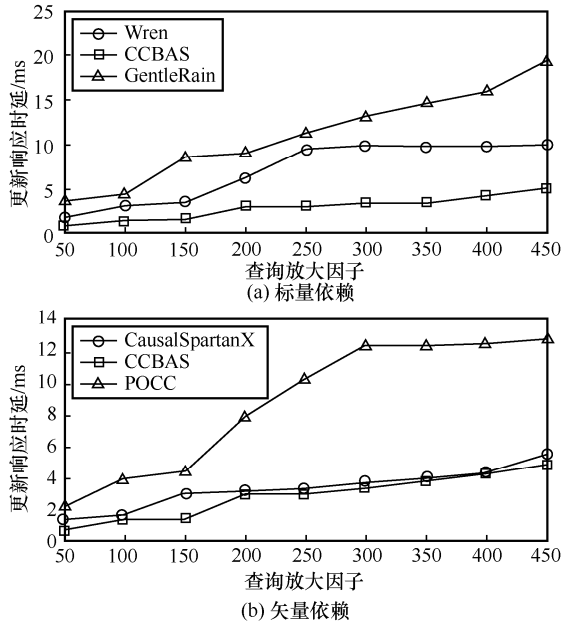


图 10　查询放大因子对更新响应时延的影响

5.4　远程更新可见时延

远程更新可见时延是对应数据项在本地数据中心可见与远程数据中心可读的时间差值。为了测量 CCBAS 模型对于更新可见时延的优化，分别与不同依赖跟踪策略的模型 Wren、GentleRian、CausalSpartanX 进行对比。以 GentleRain 为代表的单一标量依赖跟踪将因果相关性汇总到单个标量时间戳不可避免地增加了虚假相关性数量，同时最小可见性时延受到最远数据中心的传输时延影响，对应更新的可见性大幅度降低。Wren 的标量依赖检验机制将本地与远程数据中心分离，减少本地更新应用时延但是无法避免虚假依赖性与慢速副本影响。与标量时间戳不同，以 CausalSpartanX 为代表的矢量依赖跟踪检查因果相关性较复杂，分区重载次数多，故依赖性检查费时。且由于元数据因果依赖条目多，增加了存储与计算开销。尽管 CCBAS 与 Causal SpartanX 模型都依赖矢量检验，但是 CCBAS 模型在审计结果中验证因果序时，远端数据中心接收到已序列化的元数据，进而进行因果验证时分区重载次数降低，从而降低系统开销，提升数据远程可见性。

本节实验过程中设置的分布式数据存储系统由 3 个数据中心组成。其中，2 个数据中心设置在固定位置（河北保定），第三个数据中心根据距离远近的原则来设置，分别设在北京、上海、新加坡。实验通过多次测试数据包在数据中心之间的往返，统计往返时延（RTT, round-trip time），统计结果如表 3 所示。从表 3 可知，数据中心往返时延与距离成正比。

表 3　　数据中心往返时延

数据中心位置	RTT/ms
保定	0.187
北京	15.308
上海	34.689
新加坡	83.936

RTT 反映了网络中数据传输状态，与更新可见时延呈正相关。由于不同地理位置导致数据中心存在一定程度时钟漂移，实验所得更新可见时延为近似值。不同模型的更新可见时延对比如图 11 所示。

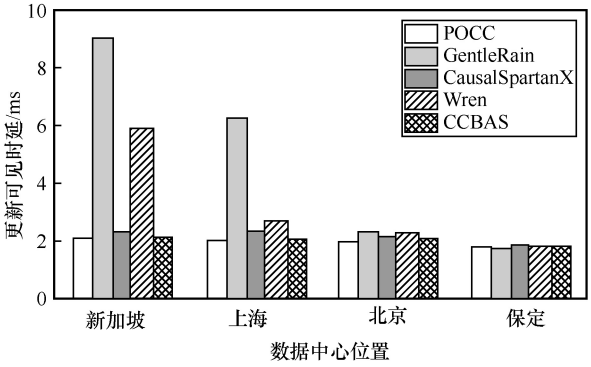


图 11　不同模型的远程更新可见时延对比

从图 11 可知，GentleRain 和 Wren 的更新可见时延明显高于其他模型。此外，随着 RTT 的增加，GentleRain 和 Wren 的更新可见时延呈明显递增趋势，这说明使用标量依赖易受网络传输时延的影响．但 CausalSpartanX 和 CCBAS 并未受到影响，这是因为使用矢量依赖跟踪本地最新的键值条目，有效降低了数据中心与慢副本、客户端与服务端之间存在通信时延造成的更新可见时延。同时，当远程数据中心位于上海时，CCBAS 模型的更新可见时延比 CausalSpartanX 低 12.56%。CCBAS 模型在矢量依赖的基础上分区协同审计，排除具有相同时间戳的无因果相关性的更新对依赖跟踪的影响，优化了系统的更新可见

性。与不运行全局稳定协议的 POCC 模型相比，CCBAS 模型的远程更新可见时延略高。但在 POCC 模型中，为满足一致性需求使用操作间留白降低阻塞概率，在实际应用中影响了吞吐量等性能指标的实现。

6 结束语

为解决矢量依赖跟踪导致的低吞吐量等问题，本文提出基于偏向稳定的分布式审计因果一致性模型 CCBAS。CCBAS 模型通过偏向稳定向量减小元数据大小，降低分区载入次数，实现了吞吐量的提升。此外，CCBAS 模型通过在矢量依赖跟踪的基础上增加因果审计和分区协同审计，减少元数据因果序验证开销。在满足数据因果一致性的基础上，CCBAS 模型降低了 GET/PUT 操作的复杂度，提升系统在短时间内大量并发请求的响应速度。实验结果表明，减少元数据通信与验证开销可实现在吞吐量、更新响应时延与更新可见时延上的优化。

参考文献：

[1] ALDIN H N S, DELDARI H, MOATTAR M H, et al. Strict timed causal consistency as a hybrid consistency model in the cloud environment[J]. Future Generation Computer Systems, 2020, 105(C): 259-274.

[2] MOHAMED B S, AHMED B, CONSTANTIN E. Robustness against transactional causal consistency[J]. Logical Methods in Computer Science, 2021, 17(1): 3-56.

[3] AHAMAD M, NEIGER G, BURNS J E, et al. Causal memory: definitions, implementation, and programming[J]. Distributed Computing, 1995, 9(1): 37-49.

[4] CADAMBE V, LYU S H. CausalEC: a causally consistent data storage algorithm based on cross-object erasure coding[J]. arXiv Preprint, arXiv: 2102.13310, 2021

[5] CAMILLERI C, VELLA J G, NEZVAL V. ThespisTRX causally-consistent read transactions[J]. International Journal of Information Technology and Web Engineering, 2020, 15(1): 1-16.

[6] DU J Q, ELNIKETY S, ROY A, et al. Orbe: scalable causal consistency using dependency matrices and physical clocks[C]//Proceedings of the 4th annual Symposium on Cloud Computing. New York: ACM Press, 2013: 1-14.

[7] DIEGO D, RACHID G, WANG J J, et al. Causal consistency and latency optimality: friend or foe?[J]. Proceedings of the VLDB Endowment, 2018, 11(11): 1618-1632.

[8] XIANG Z L, VAIDYA N H. Global stabilization for causally consistent partial replication[C]//Proceedings of the 21st International Conference on Distributed Computing and Networking. New York: ACM Press, 2020: 1-30.

[9] SPIROVSKA K, DIDONA D, ZWAENEPOEL W. Optimistic causal consistency for geo-replicated key-value stores[C]//Proceedings of 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2017: 2626-2629.

[10] LAMPORT L. Time, clocks, and the ordering of events in a distributed system[J]. Communications of the ACM, 1978, 21(7): 558-565.

[11] DU J Q, IORGULESCU C, ROY A, et al. GentleRain: cheap and scalable causal consistency with physical clocks[C]//Proceedings of the ACM Symposium on Cloud Computing. New York: ACM Press, 2014: 1-13.

[12] KULKARNI S S, DEMIRBAS M, MADAPPA D, et al. Logical physical clocks[C]//International Conference on Principles of Distributed Systems. Berlin: Springer, 2014: 17-32.

[13] ROOHITAVAF M, DEMIRBAS M, KULKARNI S. CausalSpartan: causal consistency for distributed data stores using hybrid logical clocks[C]//Proceedings of 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). Piscataway: IEEE Press, 2017: 184-193.

[14] GUNAWARDHANA C, BRAVO M, RODRIGUES L. Unobtrusive deferred update stabilization for efficient geo-replication[C]//Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference. New York: ACM Press, 2017: 83-95.

[15] DZIUMA D, FATOUROU P, KANELLOU E. Consistency for transactional memory computing[C]//Transactional Memory. Foundations, Algorithms, Tools, and Applications. Berlin: Springer, 2015: 3-31.

[16] ZHANG I, SHARMA N K, SZEKERES A, et al. Building consistent transactions with inconsistent replication[C]//Proceedings of the 25th Symposium on Operating Systems Principles. New York: ACM Press, 2015: 263-278.

[17] AKKOORATH D D, TOMSIC A Z, BRAVO M, et al. Cure: strong semantics meets high availability and low latency[C]//Proceedings of 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2016: 405-414.

[18] SPIROVSKA K, DIDONA D, ZWAENEPOEL W. Wren: nonblocking reads in a partitioned transactional causally consistent data store[C]//Proceedings of 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway: IEEE Press, 2018: 1-12.

[19] SPIROVSKA K, DIDONA D, ZWAENEPOEL W. PaRis: causally consistent transactions with non-blocking reads and partial replication[C]//Proceedings of 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2019: 304-316.

[20] ROOHITAVAF M, DEMIRBAS M, KULKARNI S. CausalSpartanX: causal consistency and non-blocking read-only transactions[J]. arXiv Preprint, arXiv: 1812.07123, 2018.

[21] KAKWANI D, NASRE R. Orion: time estimated causally consistent key-value store[C]//Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data. New York: ACM Press, 2020: 6.

[22] LAMPORT. How to make a multiprocessor computer that correctly executes multiprocess programs[J]. IEEE Transactions on Computers, 1979, C-28(9): 690-691.

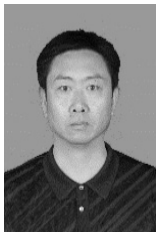
[23] XIANG Z L, VAIDYA N H. Partially replicated causally consistent shared memory: lower bounds and an algorithm[C]//Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. New York: ACM Press, 2019: 425-434.

[24] XIANG Z L, VAIDYA N H. Brief announcement: partially replicated causally consistent shared memory[C]//Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing. New York: ACM Press, 2018: 273-275.

[25] ROOHITAVAF M, KULKARNI S. DKVF: a framework for rapid prototyping and evaluating distributed key-value stores[C]// Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. New York: ACM Press, 2018: 912-915.

[26] 马文龙, 朱好晴, 蒋德钧, 等. Key-value 型 NoSQL 本地存储系统研究[J]. 计算机学报, 2018, 41(8): 1722-1751.
MA W L, ZHU Y Q, JIANG D J, et al. A survey on local key-value store of NoSQL system[J]. Chinese Journal of Computers, 2018, 41(8): 1722-1751.

[作者简介]



田俊峰（1965− ），男，河北保定人，博士，河北大学教授、博士生导师，主要研究方向为信息安全与分布式计算。

杨乾宇（1998− ），女，山西长治人，河北大学硕士生，主要研究方向为信息安全、数据一致性。

Jitian Xiao（1958− ），男，河北邯郸人，博士，伊迪斯科文大学教授，主要研究方向为数据库应用、数据处理、数据挖掘、大数据分析和企业数据安全。