具有可信约束的分布式存储因果一致性模型

田俊峰[1,2]，张俊涛[1,2]，王彦骉[1,2]
（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）

摘 要：目前，关于分布式存储因果一致性的研究领域中鲜有考虑安全风险的成熟方案。在混合逻辑时钟和 HashGraph 的基础上，结合可信云平台中的可信云联盟技术，提出了具有可信约束的分布式存储因果一致性模型（CCT 模型）。CCT 模型在客户端、服务端分别设计了身份认证和一致性数据可信校验机制，并对云存储集群中数据副本之间同步数据的过程进行了安全约束。通过仿真实验验证，CCT 模型在造成了较小性能开销的前提下，能对客户端和服务端中身份签名伪造、非法第三方等安全风险进行识别并验证，为系统提供可信约束。

关键词：分布式存储；因果一致性；可信约束；混合逻辑时钟；HashGraph
中图分类号：TP309
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021091

Distributed storage causal consistency model with trusted constraint

TIAN Junfeng[1,2], ZHANG Juntao[1,2], WANG Yanbiao[1,2]
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

Abstract: At present, there are few mature solutions to consider security risks in the research field of distributed storage causal consistency. On the basis of hybrid logic clock and HashGraph, combined with trusted cloud alliance technology in trusted cloud platform, a distributed storage causal consistency model (CCT model) with trust constraints was proposed. The CCT model designed identity authentication and consistent data trust verification mechanism on the client side and the server side respectively, and imposed security constraints on the process of data synchronization between data replicas in the cloud storage cluster. Through the simulation experiment, CCT model can identify and verify the identity signature forgery, illegal third party and other security risks in the client and server, and provide the trusted constraint for the system on the premise of causing small performance cost.

Keywords: distributed storage, causal consistency, trusted constraint, hybrid logical clock, HashGraph

1 引言

随着互联网信息时代的高速发展，用户需求也变得多样化，具有高性能和可扩展性等优点的分布式云存储解决了这一困局。为了提供高速、可靠并且成本低的分布式存储服务，云服务提供商一般将服务器节点分布在不同的地理位置[1]。数据一致性是分布式云存储中的基本问题之一。在分布式系统中对各个节点间的数据进行同步，保持一致的状态即为数据一致性。

数据一致性按照强度大小可分为最终一致性、因果一致性与严格一致性。最终一致性的强度最弱，例如亚马逊的 Dynamo[2]存储平台。严格一致性是一种强一致性，指的是每个节点中的更新都在其他节点中立即可见[3]，对分布式系统的性能有极高的要求。因果一致性属于中间强度的数据一致性，只需

收稿日期：2020−12−15；修回日期：2021−03−24
通信作者：张俊涛，zhangjuntao7321@163.com
基金项目：国家自然科学基金资助项目（No.61802106）；河北省自然科学基金资助项目（No.F2016201244）
Foundation Items: The National Natural Science Foundation of China (No.61802106), The Natural Science Foundation of Hebei Province (No.F2016201244)

使每个节点中更新事件之间的因果顺序对其他节点可见。因果一致性模型不仅能为用户提供数据的及时更新，也能在因果依赖型的影响可见时为用户事件提供因果序保障。

随着信息技术的高速发展，现有的弱数据一致性平台越来越难以满足多样化的用户需求。强度更高、性能更好的因果一致性成为众多学者的研究对象。Didona 等[4]提出的 Okapi 是一种使用了混合逻辑时钟（HLC, hybrid logical clock）[5]和全局稳定向量（GSV, global stable vector）的因果一致性方案。该方案通过一个 GSV 来追踪当前分区最新条目的时间戳，并规定当所有节点都将更新信息同步完成后才允许用户查询，这导致其更新可见性时延很高。Roohitavaf 等[6]提出了一种使用 HLC 和数据中心稳定向量的因果一致性模型 CausalSpartan。该模型提出了全新的分区稳定向量，降低了更新可见性时延。但该模型是在理想情况下实现的，并未提出对第三方篡改等风险的应对策略。Bravo 等[7]提出的 Saturn 方案依靠元数据序列化标签来传递元数据的因果序，数据副本据此对元数据的因果一致性约束进行校验。此外，Saturn 还通过一组序列化器在彼此接近的 2 个数据中心之间快速地建立起元路径，加快标签在拓扑网络中的传播速度。该方案很好地解决了吞吐量和更新可见性时延之间的权衡问题，也能充分利用部分地理复制策略的优势。但序列化器运行在客户端的关键路径上，影响了系统的并发性，同时，该方案也并未考虑可能遇到的数据安全性问题。

随着云存储的不断普及与应用，安全问题已成为制约其进一步发展的重要因素[8]。分布式的系统部署、开放的网络环境、复杂的数据应用和众多的用户访问，都使大数据在机密性、完整性、可用性等方面面临更大的挑战。服务的高可用性是用户最关心的，为了提高系统的容灾能力，应对节点失败的情况，云服务平台往往会创建冗余资源。冗余数据在提高服务可用性的同时，扩大了数据的信任域，带来了一定的安全风险，需要云平台在用户访问控制以及数据存储过程的安全性方面付出更高的成本[9]，势必会对系统性能造成一定的影响。数据完整性避免了未经授权的用户访问数据，从而对数据进行篡改或其他非法操作[10]。身份认证能为用户和云服务提供商的身份真实性提供保证[11]，所以要在用户连接、访问和使用数据的过程中对用户的身份进行验证和确认[12]。针对现有基于公钥基础设施（PKI, public key infrastructure）的跨域身份认证机制存在信任路径长、证书验证效率低、域间信任路径构建复杂等问题，杨小东等[11]利用代理重签名技术提出了一种云环境下的跨域身份认证方案。可信计算是一种信息系统安全新技术，沈昌祥等[13]提出了以芯片为信任根、主板为平台、软件为核心、网络为纽带的应用成体系的可信计算体系框架。何欣枫等[14]对可信云平台的构建与可信虚拟化技术进行了详细的分类与介绍。刘川意等[15]提出了一种将虚拟可信平台模块（vTPM, virtual trusted platform module）和可信审计技术结合起来的用户可信运行环境构建与审计机制。田俊峰等[16]基于可信平台模块（TPM, trusted platform module）提出了基于 TPA 云联盟的数据完整性验证模型，改进了传统单节点 TPM 性能较低的缺点。

目前的数据因果一致性方案集中讨论基于拜占庭容错的建模方法对数据复制过程进行改进，旨在降低用户数据同步时间等方面性能，而这些因果一致性平台都是建立在理想环境下的，并未考虑实际云存储环境下的节点可信问题。现有的可信云平台节点管理、可信云平台节点可信认证等研究方案中鲜有支持数据一致性的相关方案。Roohitavaf 等[17]提出了一种分布式键值框架（DKVF, distributed key-value framework），并且集成了性能测试与底层数据存储接口，但并未考虑分布式存储环境中的安全风险，不能提供安全约束下的节点管理方案。

目前，数据一致性方面的研究都是基于理想环境的，并未考虑网络环境中的不安全因素[18]，原因在于若针对所有的元数据提供安全认证和完整性校验，势必会造成极大的性能开销。但现今网络中的安全风险已是不可忽略的热点问题，为了准确并高效地在实际云存储环境中进行身份认证、一致性元数据的完整性校验，保证数据因果一致性，本文在混合逻辑时钟和 HashGraph 的基础上，结合现有的可信云平台相关成果，提出了具有可信约束的分布式存储因果一致性模型（CCT, causal consistency model with trusted constraint）。本文的主要工作如下。

1) 客户端操作中，客户端将客户端可信签名（TSc, trusted signature of client）随相应的请求信息

一起发送给服务端。当收到服务端返回的消息后，客户端首先对其中的服务端可信签名（TSs, trusted signature of service）进行验证，若验证成功，则依据响应消息更新本地依赖集。最后借鉴 HashGraph 共识机制对收到的数据可信证据（TPS, trusted proof of data series）在本地进行验证。

2) 服务端操作中，在收到客户端请求后，服务端对 TSc 进行验证，若结果为不可信，则拒绝处理该请求。若验证为可信状态，则将响应结果与对应的 TSs 一起发送到客户端。节点内部、节点间同步数据的过程中，将 TSs 作为节点稳定状态向量的一部分内容进行同步，从而为数据因果一致性协议提供最大程度的可信保障。

3) 采用部分地理复制策略，即当前数据中心只存储完整数据集的任意子集[19]，借鉴区块链技术 HashGraph 中的共识机制，各个节点使用混合逻辑时钟随机地向其他节点同步最新的记录与可信状态。每个节点的可信状态具有一定期限，节点内部的分区共享当前节点的可信状态。

2　背景知识

2.1　可信云平台

云计算的可信问题是云服务广泛推广面临的一个关键问题。构建可信云平台是保障云计算安全的基础之一。可信云平台利用底层的 TPM 作为可信任根，构建可信任链，通过平台进行验证，保证节点处于安全可信状态[15]。可信云平台的核心之一是可信虚拟化。可信云平台通过 vTPM 模拟硬件 TPM 的功能，实现不同虚拟机对硬件 TPM 的共享，同时保证了虚拟机之间的相对独立，提高了安全性。目前，我国已经构建起相对完整的可行计算体系，并提出了可信密码模块（TCM, trusted cryptography module）标准[14]。

可信云平台基于 TPM 技术，提供安全内存的读写 TSS_Pcr()、非易失性（NV, non-volatile）存储的安全认证 TSS_Quote() 以及安全信息的认证 TSS_Sig() 等安全机制。其中 TSS_Pcr() 和 TSS_Quote()可为当前云存储环境提供可信认证，包含存储状态、服务端口以及通信地址等内容，确保了客户端及服务端身份的真实性。实际分布式存储环境中，在云服务商为用户提供元数据的查询、读取等服务的过程中还存在数据被篡改的安全隐患。针对此风险提出的可信云平台技术基于 TPM 提供的可信度量机制，使用安全信息的认证机制 TSS_Sig()可为同步后的数据提供完整性签名，使客户端在收到服务端响应后对数据进行完整性验证，解决了非安全环境中数据完整性验证的问题。

可信云平台的可信认证与数据加密机制在最大程度上保障了实际环境中用户身份与数据的可信度。但由于其节点动态管理的性能瓶颈与可信证据难以统一认证等问题，在实际分布式存储环境中 TPM 极难普遍采用。田俊峰等[20]提出的可信云联盟方案是一种可信云平台管理模型。该模型在 TPM 的基础上设计了数据保护与可信认证、可信证据度量等组件，并包含 vTPM，提供了安全存储、可信度量等接口。该云联盟技术解决了不同位置 TPM 之间的信任问题，即在 TPM 的基础上设置信任根（rTPM, root TPM），将可信证据的统一度量问题简化为 rTPM 之间对可信证据的传递。

2.2　因果一致性原理

数据一致性一般指在分布式存储中，位于不同地理位置的用户读取到的数据保持一致。因果一致性作为一种中间强度的一致性，是目前数据一致性的重要方案之一。相比于最终一致性难以及时更新数据的缺点和严格一致性对性能开销的高要求，因果一致性在保证数据能及时同步的前提下，为用户提供保障因果序的存储服务。

因果一致性是基于用户操作之间的前后关系定义的，该关系的定义如下。

定义 1　用户操作的前后关系。a 和 b 定义为 2 个操作，当且仅当至少以下条件之一成立时，a 在 b 之前发生成立。

1) a、b 在同一线程中执行，并且 a 在 b 之前提交。

2) a 和 b 在不同线程中执行，a 是 PUT(k,v)操作，b 是 GET(k)操作，b 返回由 a 写入的值。

3) 存在操作 c，使 $a \rightarrow c$ 并且 $c \rightarrow b$ 成立。

a 与 b 之间的这种关系可表示为 $a \rightarrow b$，也称为 a 因果依赖于 b。

定义 2　因果关系。如果用 X 和 Y 分别表示键 x 和 y 的值，若 PUT(x,X)→PUT(y,Y)成立，则 X 与 Y 为因果关系，可表示为 X dep Y。

定义 3　可见性。如果客户端 c 的 GET(k)操作结果为 v'，并且 v'满足 v'≠v 或者┐(v dep v')，那么键 k 的值 v 对客户端 c 是满足可见性的。

定义 4 因果一致性。假设数据中心中存在 2 个任意的键 x 和 y，X 为键 x 的值，Y 为键 y 的值，在 X dep Y 的情况下，如果 X、Y 对客户端 c 可见，那么该数据中心满足“因果一致性”。

在实际云环境中，不同地理位置的节点中的数据存储方案一般分为 2 种，一种为完全地理复制，即分布式系统中每个数据中心都存储完整的数据集；另一种为部分地理复制，即当前数据中心只存储完整数据集的任意子集。

本文提出的 CCT 模型使用部分地理复制方案，基于可信机制为用户的因果一致性元数据提供可信约束，用户在当前节点写入数据之后，当前的更新状态也会与其余节点进行同步，同步过程中除了可信证据的传递，还包括节点稳定状态同步与时钟同步。

2.3 HashGraph

区块链是一种非常优秀的分布式数据存储技术，包含 2 种核心思想：分布式账本存储与共识机制，解决了分布式节点中交易的信任和安全问题，可实现更广泛意义上的安全多方计算[21]。HashGraph 是区块链的一种改进方案，提供分布式账本和共识机制的数据存储，主要采用八卦同步和虚拟投票的方式对新产生的交易进行存储和同步。

在 HashGraph 中，共识机制是通过不同事件间的八卦同步实现的。如图 1 所示，用户 A 将发生的事件随机与相邻的其余用户进行交流，将其所有的事件历史告诉用户 B。同样 B 也执行相同操作，其余成员也是如此。这样，如果环境中产生一个新的事件，该信息会以指数级的速度在整个环境中传播，直到所有成员同步完成该事件。

此外，用户不仅会同步新发生的事件，还会确切地知道其他用户什么时候同步了该事件。当所有成员都有 HashGraph 的副本时，即达成了最新状态的共识。

当所有节点都收到用户 A 发生的事件后，当前环境下记录事件历史的账单就完成了同步。这是因为 HashGraph 所有副本都同步完事件历史后，任意副本（例如用户 D）都知道“其他所有副本都已同步此事件”，即形成了全网共识。HashGraph 最终达成的效果就是所有用户并没有聚到一起进行表决，也能公认当前事件的完成，从而达成共识。

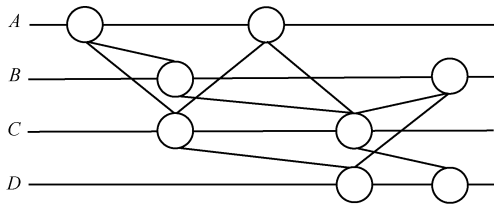


图 1 相邻事件之间的八卦同步

2.4 混合逻辑时钟

传统的因果一致性协议采用单调的物理时钟作为判断因果序的依据，例如 GentleRain 模型[22]，在物理时钟的基础上定义了一个单向向量，即全局稳定时间（GTS, global stable time）。该向量使用数据条目的更新时间戳作为判断因果序的依据。若采用物理时钟，各个节点间的时钟无法在同一段时间内完全达成一致，从而产生时钟漂移，无法准确判断用户事件的因果序。在实际应用中，物理时钟是单调递增的，不能倒退。假设在一个由 2 个数据中心 A 和 B 组成的系统中，2 个数据中心的 GST 为 5，这意味着 2 个数据中心时间戳小于 5 的所有值都能被用户查询到。假设数据中心 A 在时间戳为 5 时更新了一条数据 Item_1，而此时数据中心 B 与数据中心 A 的时钟存在偏差，向 A 同步一条时间戳为 6 的数据 Item_2。由于该数据的时间戳高于 GST，因此 Item_2 不能被用户查询到，即该分布式系统无法为用户提供准确的因果一致性数据的读写服务。

逻辑时钟仅对定义 2 中的因果关系有要求，即只记录客户端发生事件的前后顺序，与事件发生的具体物理时间不相关。

HLC 结合了逻辑时钟和物理时钟两者的优势。事件 E 的 HLC 时间戳表示为 E.hlc，该时间戳是一个元组<$l.E$, $c.E$>。第一个元素 $l.E$ 是当前事件发生的时刻，即事件 E 发生时本地服务器物理时间。第二个元素 $c.E$ 是一个因果关系计数器，用于捕捉事件之间的因果序。通过 $c.E$，可以在时钟漂移造成 2 个事件 $l.E$ 相同的情况下，能够正确判断事件之间的因果序。

算法 1 混合逻辑时钟

输入 事件 E，物理时间 pt

输出 时间戳

Upon sending E message or event by client process

1) $l'.E = l.E$

2) $l.E = \max(l'.E, \text{pt}.E)$

/*判别最新时间*/

3) if　($l.E == l'.E$)　$c.E = c.E + 1$

/*追踪因果序*/

4) else　$c.E = 0$

5) end if

6) print Timestamp: $<l,c>.E$

Upon receiving message m by client process

1) $l'.E = l.E$

2) $l.E = \max(l'.E; l.m; \text{pt}.E)$

/*$l.m$ 是收到的消息中的时钟信息*/

3) if ($l.E == l'.E == l.m$) then $c.E = \max(c.E; c.m) + 1$

4) elseif　($l.E == l'.E$) then $c.E = c.E + 1$

5) elseif　($l.E == l.m$) then $c.E := c.m + 1$

6) else　$c.E = 0$

7) end if

8) print Timestamp: $<l,c>.E$

算法 1 是 HLC 的计算过程，分别为客户端产生事件时的时钟定义方法与接收事件之后对 2 个时间戳元素进行检查的时钟同步方法。若 2 个事件 m 和 n 的 $l.E$ 相同，但 m 在 n 之前发生，混合逻辑时钟则将 $c.m$ 设置为高于 $c.n$ 的值，从而为事件 m、n 提供因果序判别依据。

3　具有可信约束的 CCT 协议

本文提出的可信约束下的 CCT 协议结合了可信云平台中的可信认证机制，不仅对节点与客户端的可信身份签名 TS 进行验证，而且将节点可信证据的收集、管理与数据因果一致性协议相结合，为用户一致性元数据的安全存储与操作提供了保障。图 2 为 CCT 模型的实体流程。

此外，CCT 协议借鉴 HashGraph 共识机制，将用户每次的存、取操作均看作一次事件，每次处理用户的存、取请求之后都基于可信机制为该操作数据和依赖集分配一个 TPS，作为事件历史的签名信息。分布式存储副本在处理完用户请求或在固定时间间隔内未处理任何请求的情况下，均随机与其连接的其他副本同步最新状态，最后各副本之间未聚合在一起进行投票，但都快速地达成了最新状态的共识。

CCT 协议中对因果一致性操作提供的可信机制不仅包括客户端以及服务端的身份认证，还包括数据因果一致性元数据的完整性验证，即服务端处理 PUT、GET 请求后均计算元数据和更新后依赖集的数据可信证据并发送到客户端，以验证操作和结果的完整性和可信性。

分布式环境中，数据一致性协议仅靠服务端提供的可信机制，难以提供可信证据的传递验证，进而无法保证整体分布式环境的可信状态。基于可信认证的 CCT 因果一致性协议中不仅包括了客户端与服务端的身份认证与一致性数据完整性校验机制，还对服务端之间的数据同步过程进行了安全约束。

CCT 协议中客户端与服务端之间的写入（PUT）操作和查询（GET）操作的具体过程如图 3(a)和图 3(b)

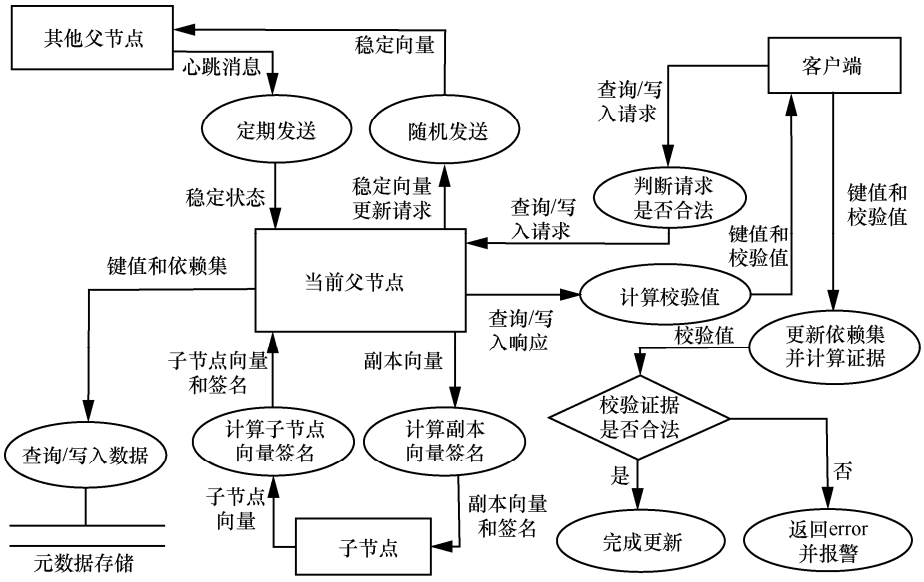


图 2　CCT 模型的实体流程

所示，相对应的算法详见算法 2 和算法 3。

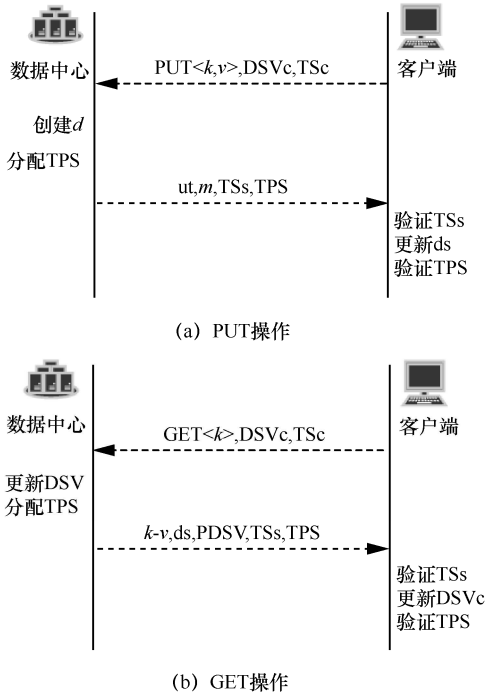


图 3　CCT 协议中的 PUT 和 GET 操作

CCT 模型基于可信云平台对客户端与服务端之间的通信进行身份认证与因果一致性数据完整性校验；同时对服务端之间的消息同步进行安全约束；通过使用 HLC 与服务端部分稳定向量（PDSV, part data stable vector），减少了客户端与服务端之间的消息传播的体量；同时借鉴 HashGroup 共识机制的消息传播方式，加快了消息同步速度。因此，CCT 模型在为数据因果一致性系统规避安全风险的同时并没有造成很高的性能开销。相比于未考虑安全约束的因果一致性方案如 Okapi、CausalSpartan 等，CCT 模型在环境中存在不可信第三方时，可以为数据因果一致性操作提供安全保障，防止数据被篡改；与使用数据标签提供安全校验的 Saturn 模型相比，大大降低了更新可见性时延，提高了系统性能。

3.1　客户端中的可信机制

算法 2 是客户端中的 GET 操作和 PUT 操作的算法。其中客户端首次运行时先对自身运行环境、可信存储空间做出认证，结合可信云平台中的可信签名机制，分配身份签名 TSc，作为服务端处理请求的依据。当客户端中存在不安全因素使可信存储空间或运行环境存在风险时，可信认证失败，用户可依据错误信息对客户端进行检查。

算法 2　客户端 c 中的可信约束

GET 操作

输入　键 k，DSVc，TSc

输出　值 v ，ds，PDSV，TSs，TPS

1) TSc = TSS_Quote(Client c,Pcr,IP,Port);

/*根据客户端、可信存储以及通信条件分配客户端可信签名*/

2) send <GETREQ k,DSVc,TSc> to server;

/*发送 GET 请求*/

3) receive <GETREPLY v,ds,PDSV,TSs,TPS>;

4) if (check(TSs))

/*验证服务端可信签名*/

5)　DSVc ← max(DSVc,PDSV);

/*更新客户端状态*/

6)　for each <i,h>∈ds

7)　　DSc ← updateDS(i,h,ds,PDSV);

/*更新客户端依赖集*/

8)　　if (TPC(DSc) ↔ TPS)　return v ;

/*校验可信证据*/

9)　　else　return error;

10)　end if

11)　end for

12) else　return error;

13) end if

PUT 操作

输入　键 k，值 v，TSc，DSVc

输出　ut, m, TSs, TPS

1) TSc = TSS_Quote(Client c,Pcr,IP,Port);

/*根据客户端、可信存储以及通信条件分配客户端可信签名*/

2) send <PUTREQ k,v,TSc,DSVc> to server;

/*发送 PUT 请求*/

3) receive <PUTREPLY ut,m,TSs,TPS>;

4) if (check(TSs))

/*验证服务端可信签名*/

5)　DSc ← updateDS(m,ut);

/*更新客户端依赖集*/

6)　if (TPC (DSc) ↔ TPS)　return success

/*校验可信证据*/

7)　else　return error;

8)　end if

9) else　return error;

10) end if

在 GET 过程中，若客户端分配可信签名成功，用户将客户端可信签名 TSc 与查询请求一起向服务端发送。查询请求中还包含了客户端数据稳定向量（DSVc, data stable vector of client），用来与节点同步最新状态。客户端收到服务端返回的响应消息之后，先对服务端签名 TSs 进行验证。验证通过后，根据其中的服务端部分稳定向量更新自身依赖关系集（DSc, dependency series client）。然后根据更新之后的依赖关系集计算数据可信证据，并与收到的服务端数据 TPS 进行验证。若验证成功说明当前操作来源为已连接且认证为可信的服务节点，且基于可信认证机制验证的数据满足可信约束下的完整性、数据因果一致性，否则返回错误信息，由客户端排查错误并重新发起申请。

客户端中的 PUT 过程与 GET 过程相似，客户端将自身可信身份签名 TSc 封装进 PUT 请求消息发送给服务端。客户端在收到服务端响应后同样首先结合 TPM 2.0 可信认证机制校验服务端的身份签名 TSs，然后根据响应消息中为数据条目分配的存储位置 m 以及时间戳 ut 更新本地依赖关系集。最后计算依赖关系集的数据可信证据，与服务端发来的数据 TPS 进行校验，若校验失败则返回错误信息并且驳回该请求，进行重新认证与发送。

3.2　服务端中的可信机制

服务端操作一般处理过程包含 GET、PUT 这 2 个操作，GET 过程如算法 3 所示。首先生成 TSs。在处理客户端 GET 请求之前，集群对客户端可信签名 TSc 进行校验，若身份认证通过则依据客户端稳定状态 DSVc 更新副本的 DSV。其次根据该查询请求中的键在安全存储中检索对应键的最新值并更新依赖关系集 ds。再次计算 k-v 条目和依赖关系集的数据 TPS。最后将查询到的值 v、依赖关系集 ds、服务端部分稳定向量 PDSV、服务端可信签名 TSs 以及数据可信证据 TPS 打包成为响应消息发送到客户端。因为客户端与服务端运行环境中的可信约束是基于可信云联盟技术的，所以无论是客户端还是服务端，均可对收到的可信身份签名、数据可信证据进行校验。

算法 3　服务端 p_n^m 中的 GET 和 PUT 操作

GET 操作

输入　键 k，DSVc，TSc

输出　值 v，ds，PDSV，TSs，TPS

1) Upon receive < GETREQ k,DSVc,TSc>

2)　TSs = TSS_Quote(p_n^m,Pcr,IP,Port);

/*根据分区、可信存储以及通信条件分配服务端可信签名*/

3)　if Check(TSc)

/*检查客户端可信签名*/

4)　　$DSV_n^m \leftarrow \max(DSV_n^m, DSVc)$;

5)　　obtain latest v from v chain of key k;

6)　　ds ← updateDS(d.sr,d.ut,d.ds);

/*更新依赖关系集*/

7)　　TPS = TSS_Sig(k,v,DS);

/*根据数据、更新后依赖集分配数据可信证据*/

8)　　send <GETREPLY d.v,ds,PDSV, TSs, TPS> to client

9)　else return error;

10) end if

PUT 操作

输入　键 k，值 v，DSVc，TSc

输出　ut，m，TSs，TPS

1) Upon receive < PUTREQ k,v,DSVc,TSc>

2)　TSs = TSS_Quote(p_n^m,Pcr,IP,Port);

/*根据分区、可信存储以及通信条件分配服务端可信签名*/

3)　if Check(TSc)

4)　　dt ← max value in ds;

5)　　updateHCL(dt) and Create new item d;

/*分配空间*/

6)　　$d.k \leftarrow <k,v,VV_n^m[m],m,ds>$;

7)　　insert d to key chain of k;

/*将更新条目插入存储链中*/

8)　　TPS = TSS_Sig(k,v,ds);

/*根据数据、更新后依赖集分配数据可信证据*/

9)　　send <PUTREPLY d.ut,m,TSs,TPS> to client;

10)　else return error;

11)　end if

12)　for each server p_n^k,$k \in \{0,\cdots,M-1\}$,$k \neq m$ do

13)　　send <REPLICATE d,TS> to p_n^k;

/*向已连接副本发送数据更新消息*/

14)　end for

算法 3 中还包含了服务端的 PUT 操作过程。首先生成 TSs 并对 TSc 进行验证，不仅能保障身份可信，更能对客户端请求写入数据进行安全认证。若客户端身份满足可信约束，则为该请求创建存储空间，分配键值链空间，并更新本地稳定状态和依赖

关系集 ds。最后计算数据 TPS 并将响应消息发送到客户端，其中内容包括时间戳 ut、分配的存储空间 m、依赖集数据可信证据 TPS 和服务端身份可信签名 TSs。此外，服务端处理完客户端的 PUT 请求后还要与其余分区共享当前最近写入状态，即向相邻分区发送复制消息(Replicate d)，参照 HLC、HashGraph 共识机制，由数据中心根分区随机与其余数据中心同步最新状态，详见 3.3 节。

3.3 服务端之间的同步机制

基于可信认证的 CCT 因果一致性协议中包括了客户端与服务端的可信认证过程，其中节点间同步的过程如图 4 所示。针对实际云环境中分布式数据的存取情况进行了调查与分析，其中用户的存取请求大多集中在 11:00—23:00，CCT 模型将服务运行周期设定为 24 h，即在凌晨 2 点为系统重新分配认证证据，在接下来的可信周期之内节点与客户端能够在彼此可信的前提下，提供数据一致性元数据的存取操作。

图 4 为 CCT 模型中针对节点间状态进行更新的过程，其满足因果一致性的可信约束体现在两方面。一方面为不同地点之间的副本更新步骤中，基于可信云联盟机制运行为每个副本设置了 DSV 和 TSs。其中 DSV 不仅是判断客户端存取请求的时钟状态是否落后于当前副本最新状态的判断依据，还是对副本之间最新状态进行同步的依据。另一方面则为副本内部的数据存储分区在处理存取请求时，也基于树形拓扑结构与数据中心根分区共享彼此的分区可信稳定分量（PV, partion vector），该分量的生成依据为副本的稳定状态，每次副本在分布式环境中与其副本同步稳定状态后均将最新条目时间戳和可信副本链扩散到内部分区中，数据分区则以此为依据对自身存储状态进行更新。

算法 4 服务端 p_n^m 中的 HEARTBEAT 消息

输入 无

输出 心跳消息

1) Upon every θ time

2) $\mathrm{DSV}_n^m \leftarrow$ entry-wise $\min_{j=1}^{N}(\mathrm{PV}_j^m)$;

/*固定时间间隔后均依据 PV 更新 DSV*/

3) Upon every Δ time

4) if there has not been any replicate message in the past Δ time

5) update HCL();

6) end if

7) for each server $p_n^k, t = \mathrm{random}(M-1), k \neq m$ do

8) send <HEARTBEAT DSV_n^m, TSs> to p_n^t;

/*随机与其余副本更新心跳消息*/

9) end for

4 仿真实验与结果分析

CCT 模型基于 Berkeley DB 设计了底层数据存储和检索方式并通过 Java 实现。本实验在项目组开发的 HBU-Cluster 平台基础上，将不同地点的集群副本在本地进行了模拟。HBU-Cluster 平台是项目组开发的一个分布式键值存储管理框架，使用 Google 的 Protocol Buffer 将数据因果一致性协议结构化并集成 Yahoo 的 YCSB 基准测试模块作为性能测试工具。

HBU-Cluster 平台还实现了 GentleRain 与 CausalSpartan 协议，并提供了基准测试结果。为了模拟实际分布式环境中不稳定的用户请求情况，实验将测试负载均设置为共 1 000 个操作，对不同模型处理不同并发请求的可信约束性能开销进行了测试与统计。实验基于具有自主知识产权的可信云平台 YF-I，规格如下：运行 NeoKylin Linux Trusted OS V6(x64)，CPU*12:Intel Xeon E5-2603，1.6 GHz，16 GB 内存，240 GB 存储。

4.1 存在不可信节点的情况

CCT 模型中服务端对存储节点的存储状态和运行机制进行了安全认证，项目组在 CCT 模型的基础上，为分布式存储环境设置了 5 个存储副本，

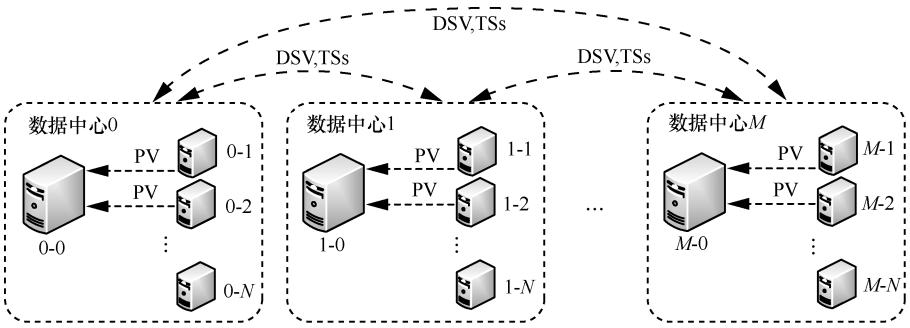


图 4 CCT 模型中各数据中心间的同步

每个副本均分配一个分区，副本距离则在本地的可信云服务器中进行模拟，与本地的首节点位置的距离分别模拟为 100 km、500 km、1 000 km、1 500 km，以期模拟实际云环境中云服务商将数据副本分布在不同区域的情况。其中为末节点手动分配身份可信签名，供其余节点校验身份签名，在不干扰客户端存取操作的前提下进行验证与测试。

为了模拟不可信环境中的风险，例如木马等非法第三方对因果一致性数据依赖集进行篡改，本文实验同时在末节点设置了不同比例的元数据条目修改与验证机制，供 HBU-Cluster 平台对结果进行验证与测试。为了使结果更清晰地展现，本节在客户端模拟了不同并发请求的情况，例如低并发环境设置吞吐负载为 5~10 个/s，同步心跳间隔为 0.1 s；而高并发环境模拟条件降低副本同步心跳间隔为 0.01 s，同时将吞吐负载增加至 50~100 个/s。在处理完服务端 PUT 操作后，将该条目对应依赖集的随机一个条目篡改为 0~9 的随机数。为了避免偶然性误差，实验结果均是 3 次完成 1 000 个存取操作的平均值，最后分别针对不同的篡改比例进行数据完整性校验，结果如图 5 所示。

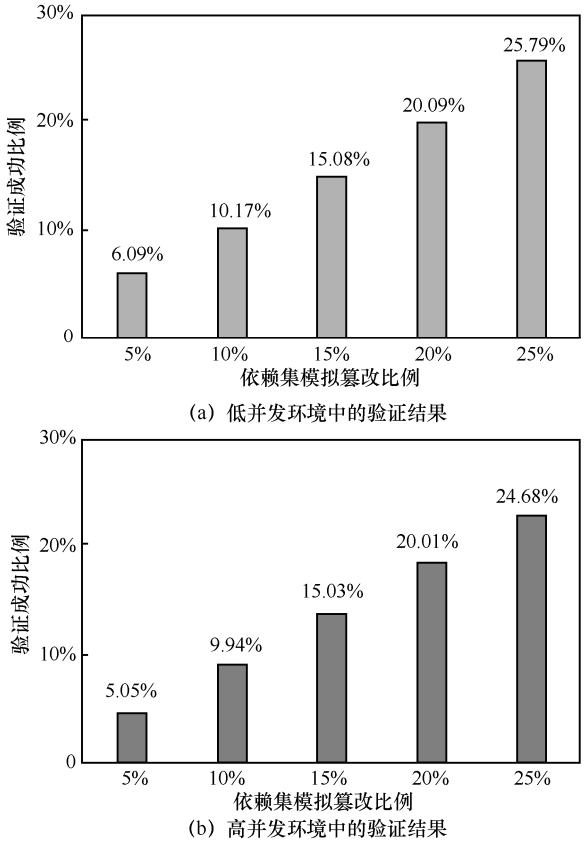


图 5　不同并发环境下模拟依赖集篡改后可信验证的结果

图 5 结果显示，在不可信节点对数据依赖集进行篡改后，5 个不同比例均全部验证了对应比例的数据风险。针对实际云环境中数据篡改的风险，CCT 模型针对 25%以下的数据篡改风险只返回风险信息，超过该阈值后则拒绝提供一致性数据存取服务。图 5(a)结果中低负载情况下由于副本间同步更新间隔相对较长，即心跳间隔较大，造成数据依赖集模拟修改后，后续操作查询结果中存在依赖集验证失败的情况，因此每个篡改比例的验证结果均存在 0.08%~1.09%的误检测率，平均误检测率为 0.44%。

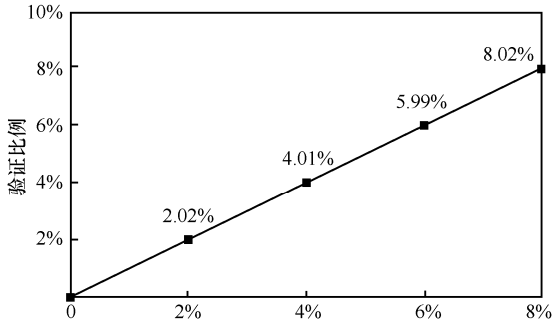
图 5(b)对实际存储环境中普遍存在的高负载处理情况进行了模拟，除了降低副本同步心跳间隔和提高负载，其余条件并未进行改变，基于此条件在客户端对随机写入的数据进行查询同时进行验证。结果显示，相比于在低并发环境中，CCT 模型在高并发环境下有更好的性能表现，全部验证出了不可信副本对一致性元数据的篡改，而误检测率范围降低到了 0.01%~0.32%。平均误检测率为 0.094%，说明本节中 CCT 模型在服务端设计可信约束的方案具有可行性。实验中的高并发条件模拟了云服务提供商满足不同用户需求和应对一些常见风险的手段，即提升系统性能，在因果一致性服务处理过程中表现为更快的副本更新速度、更高的性能开销，而 CCT 在提升系统性能的基础上相比低性能条件并未造成更高的开销，且能提供更高效的一致性数据存取服务。

4.2　客户端身份重新认证的情况

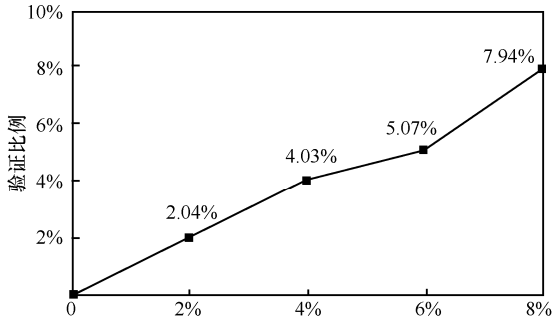
CCT 模型基于可信云联盟技术为因果一致性服务提供了身份签名和数据可信认证机制，但数据一致性存储模型是针对客户端请求提供服务的，不仅在服务端中提供了安全保障，对于客户端中的身份签名伪造、运行环境风险也提出了安全约束。实验基于 HBU-Cluster 平台在本地可信云服务器中部署了 6 个数据副本，包含 8 个分区，并在可信云联盟技术基础上分别设置 5、30、200 个客户端，每个测试结果均将分布式存储集群运行 2 天，可信周期设置为 24 h，并将集群重启时间设置在凌晨 2 点。为了模拟实际云环境中社交应用、Web 应用存取 NoSQL 型数据的条件，实验将同步心跳机制设置为 0.05 s，负载设置为 50 个/s。

为了对 CCT 模型中客户端安全风险的安全约束进行验证与测试，实验将客户端中数据存取请求

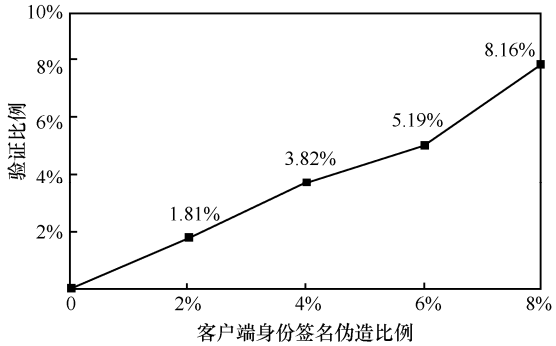
的发送方式设定为循环方式，并在 11:00—23:00 进行客户端单位吞吐量为 100~500 个/s 的大量随机存取，其余时间则为 10~100 个/s。客户端中身份签名认证机制停止服务阈值为 10%，即如果超过 10%的客户端请求可信验证失败，则对该客户端进行报警处理并拒绝服务。实验在停止服务阈值之下为客户端请求随机设计了 2%、4%、6%、8%的身份伪造，对集群将客户端伪造签名重新认证的结果进行统计和分析，结果如图 6 所示。



(a) 5个客户端下客户端身份验证表现



(b) 30个客户端下客户端身份验证表现



(c) 200个客户端下客户端身份验证表现

图 6 不同客户端数目环境中客户端身份验证表现

图 6 结果显示，CCT 模型能将客户端身份签名伪造风险完全识别，但由于通信环境的复杂性与可信云联盟技术对可信证据的验证性能缺陷，造成模型中依然存在一定的误检率。图 6(a)为使用 5 个客户端以循环方式向服务端发送 PUT、GET 请求并验证签名、统计的结果。在为低数量级客户端提供因果一致性数据存取服务时，CCT 模型有着较好的表现，其平均误检率为 0.015%。图 6(b)将客户端数量增加了 5 倍，在同样条件下进行测试的结果略差于图 6(a)中结果，其平均误检测率为 0.05%。图 6(c)为模拟实际环境中较高客户端请求的情况，客户端规模为 200 个，其余条件不变的情况下对身份篡改的验证比例进行统计，平均误检率为 0.155%。

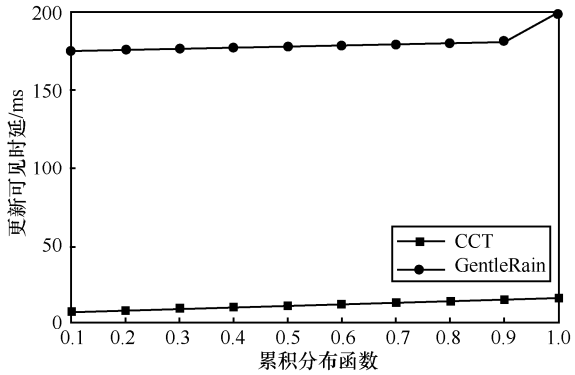
实验基于可信云平台中的可信机制，对 CCT 模型中客户端一方的可信约束进行了仿真测试，相比于 CausalSpartan、Okapi 等未提供安全约束的传统因果一致性方案，实验结果证明了 CCT 模型中提出的可信约束能对客户端中身份签名伪造、非法第三方等安全风险进行识别并验证，同时证明 CCT 模型在客户端提出可信约束的方案是可行的。虽然局限于现有实际通信环境的复杂性和可信云平台技术的性能瓶颈，仿真实验结果存在一定的误检率，随着客户端连接数、处理用户请求的并发量的提高，CCT 模型平均误检测率的递增趋势是比较稳定的。安全约束的设计难免带来一定的性能开销，CCT 模型中由于提供安全约束带来的 PUT 时延、更新可见性时延相关分析在 4.3 节进行了讨论。

4.3 可信约束造成的性能开销

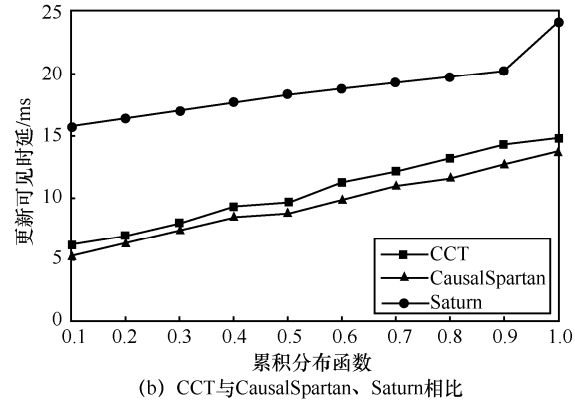
可信云平台在可信芯片 TPM2.0 的基础上提供安全策略管理、数据认证以及非对称秘钥协商等安全机制，在服务启动、运行过程中均对身份认证、数据加密有着高标准的约束。基于可信云联盟技术的 CCT 模型提出了可信身份签名、数据可信证据等安全约束，并将性能开销控制在了较小的程度。本节参考文献[6]中实验环境（Intel Xeon E5-2680、3.75 GB 内存并运行 Ubuntu 14.04）和文献[7]中实验环境（Intel Xeon E5-2680、8 GB 内存并运行 Ubuntu 12.04），基于 HBU-Cluster 平台将实验环境进行了设计和复现，最终针对 CCT 模型中 PUT 时延和更新可见性时延进行了分析，并与不包含安全约束的方案如 GentleRain 和 CausalSpartan，以及使用序列化标签进行因果一致性安全校验的方案 Saturn 进行了对比。

实验在本地可信服务器部署了 7 个副本，共包含 7 个分区，副本同步心跳间隔为 0.05 s，客户端规模为 2 个，客户端单位负载设置为 50 个/s。其中模拟 FaceBook 数据集，将随机键值大小设置为 2 B，

并进行读写比例为 1:1 的实验和统计。数据副本地理位置均设置在本地，但实际存储环境中云服务商往往键副本部署在世界各地，以满足不同区域用户的需求。为了使 CCT 模型的性能开销更加明确，实验在本地手动设置了副本间的时钟时延，即对分区间地理位置差异造成的通信时延进行模拟，并在分区间不同通信时延的条件下对更新可见性时延进行统计和分析。4.1 节和 4.2 节实验已对服务端安全约束、客户端的安全机制进行了验证与测试，因此本节只在可信云服务器环境中对安全约束进行性能开销的测试和分析，并对未对身份伪造或数据篡改风险进行模拟，实验将操作数设置为 1 000 个，每个数据均是 3 次实验所取的平均值，其中累积分布函数（CDF, cumulative distribution function）指的是与主控节点通信时延小于当前最大时延阈值的分区比例，结果如图 7 所示。



(a) CCT与GentleRain相比



(b) CCT与CausalSpartan、Saturn相比

图 7　分区间时延阈值为 10 ms 时不同模型的性能开销

实验中通过在分区间设置不同通信时延，模拟不同地理位置之间服务器的通信开销，对不同数据因果一致性方案的性能开销进行了分析。图 7(a)为 CCT 模型与 GentleRain 之间的对比，结果显示 CCT 模型明显优于 GentleRain，因为使用混合逻辑时钟能避免时钟漂移、网络时钟协议（NTP, network time protocol）同步失败带来的安全风险，结合 HashGraph 共识机制，分区稳定状态在拓扑树中快速形成共识。图 7(b)结果为参照文献[7]中数据，并在低通信时延环境下将 GentleRain 方案复现并测试，同时对比 CausalSpartan，可见 CCT 模型的更新可见时延比 Saturn 平均降低了 45.03%，比 CausalSpartan 平均增加了 10.86%。

Saturn 由于在客户端请求发送的关键路径上使用标签序列化为客户端和服务端提供双向追踪依赖集的因果序的依据，并且在处理客户端请求过程中通过构建一棵优化的结构树的方式降低通信开销，有一定的性能瓶颈。CCT 模型基于可信云联盟技术，提供可信约束的前提下并未局限于 TPM 的性能瓶颈，且结合 HLC 同步方法有效降低了时钟时延和通信时延造成的操作处理时延。因此 CCT 模型在较理想的环境中性能开销明显低于 Saturn，证明 3.1 节中结合 HashGraph 对可信证据的同步机制进行优化的方案是可行的。

在 CausalSpartan 方案中，同样使用分区稳定向量标识分区最新状态，并且都依据 HLC 判别用户存取数据事件的因果序。虽然 CCT 模型中对集群拓扑结构中数据的同步方式结合 HashGraph 共识机制设计了优化方案，对 PUT 和 GET 的事件使用 Gossip about Gossip 进行同步的方式降低了分区间数据、状态同步过程的时间开销，但是客户端与服务端之间的身份签名、数据可信证据验证机制消耗了一定性能，因而在低时延环境中 CCT 模型的更新可见性时延也是略高于 CausalSpartan。

最后一部分实验模拟了相同副本和分区规模情况下的高通信时延环境，为了使数据对比 Saturn 模型具有权威性，分区时延阈值设置为 107 ms，每个对比数据均为进行 1 000 个操作负载后 3 次结果的平均值。图 8 结果表明，在分区间通信最低为 10 ms，按比例递增至全分区通信时延为 107 ms 的情况中，CCT 模型的性能表现明显优于 Saturn、GentleRain，比未提供安全约束的 CausalSpartan 略高。

具体看来，Saturn 在高通信时延环境中的更新可见时延接近于使用物理时钟标量追踪因果序的 GentleRain，通信时延接近 107 ms 的分区占总数比例为 10%~80%时，两者更新可见时延非常接近，但在高于 80%时，GentleRain 模型严格依赖物理时钟同步的缺陷就非常明显了。Saturn 模型在所有比

例环境中表现相对稳定，因为该方案使用序列化的元数据标签判别因果序并提供安全约束、构建一棵优化的结构树的方式在分区规模增大时具有较明显的性能颈。CCT 模型采用混合逻辑时钟判别事件之间的因果序，在分区间时钟漂移、通信时延较高的情况下均有非常优秀的表现，同样包含安全约束的前提下在高通信时延环境中的更新可见时延比 Saturn 平均降低了 32.31%，且比 GentleRain 平均降低 32.92%。而在模拟实际云环境的分区间高通信时延条件下，与低通信时延条件下情况类似，CCT 模型的更新可见时延比 CausalSpartan 模型平均增加了 10.54%。

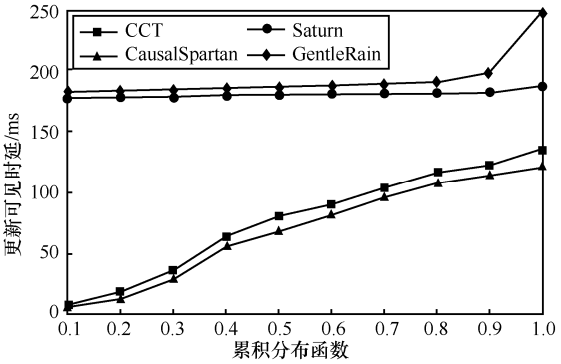


图 8 分区间时延阈值为 107 ms 时不同模型的性能开销

5 结束语

云服务商在为用户提供因果一致性数据存取服务时，采用分布式服务器在降低运营成本的同时也可满足用户的多样化需求，但针对分布式服务器、客户端中的身份签名伪造、数据窃听或篡改等安全风险鲜有考虑。CCT 模型基于 Java 实现，结合混合逻辑时钟和 HashGraph 共识机制的优化方案，并使用分区稳定向量标识分区最先状态，基于可信云平台中可信认证机制，在服务端和客户端设计了身份签名可信验证、一致性元数据完整性验证等可信约束。通过仿真实验表明，CCT 模型针对安全风险能全部识别并记录，存在较低的误检测率和较小的性能开销。相比于使用标签序列化保障数据安全的方案 Saturn，更新可见性时延明显降低。实验结果表明，CCT 模型中设计的可信约束是具有可行性的。相比于未提供安全约束的 CausalSpartan，CCT 模型的更新可见时延也仅增加了 10%左右。在实际分布式数据存储环境中，考虑到用户因果一致性敏感数据的安全性，结合可信云平台、可信认证机制设计安全约束所造成的较小性能开销是可以接受的。

参考文献：

[1] 崔勇, 宋健, 缪葱葱, 等. 移动云计算研究进展与趋势[J]. 计算机学报, 2017, 40(2): 273-295.
CUI Y, SONG J, MIAO C C, et al. Mobile cloud computing research progress and trends[J]. Chinese Journal of Computers, 2017, 40(2): 273-295.

[2] BALEGAS V, CHENG L, NAJAFZADEH M. Geo-replication: fast if possible, consistent if necessary[J]. IEEE Data Engineering Bulletin, Special Issue on Data Consistency Across Research Communities, 2016, 39(1): 6-12.

[3] DECANDIA G, HASTORUN D, JAMPANI M, et al. Dynamo[J]. ACM SIGOPS Operating Systems Review, 2007, 41(6): 205-220.

[4] DIDONA D, SPIROVSKA K, ZWAENEPOEL W. Okapi: causally consistent geo-replication made faster, cheaper and more available[J]. arXiv Preprint, arXiv:1702.04263, 2017.

[5] KULKARNI S S, DEMIRBAS M, MADAPPA D, et al. Logical physical clocks[M]. Cham: Springer International Publishing, 2014: 17-32.

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

[7] BRAVO M, RODRIGUES L, PETER V R. Saturn: a distributed metadata service for causal consistency[C]//Proceedings of the Twelfth European Conference on Computer Systems. New York: ACM Press, 2017: 111-126.

[8] 张玉清, 王晓菲, 刘雪峰, 等. 云计算环境安全综述[J]. 软件学报, 2016, 27(6): 1328-1348.
ZHANG Y Q, WANG X F, LIU X F, et al. Survey on cloud computing security[J]. Journal of Software, 2016, 27(6): 1328-1348.

[9] 张晓丽, 杨家海, 孙晓晴, 等. 分布式云的研究进展综述[J]. 软件学报, 2018, 29(7): 2116-2132.
ZHANG X L, YANG J H, SUN X Q, et al. Survey of geo-distributed cloud research progress[J]. Journal of Software, 2018, 29(7): 2116-2132.

[10] MAHALAKSHMI B, SUSEENDRAN G. An analysis of cloud computing issues on data integrity, privacy and its current solutions[M]. Singapore: Springer Singapore, 2018: 467-482.

[11] 杨小东, 安发英, 杨平, 等. 云环境下基于代理重签名的跨域身份认证方案[J]. 计算机学报, 2019, 42(4): 756-771.
YANG X D, AN F Y, YANG P, et al. Cross-domain authentication scheme based on proxy Re-signature in cloud environment[J]. Chinese Journal of Computers, 2019, 42(4): 756-771.

[12] KUMAR P R, RAJ P H, JELCIANA P. Exploring data security issues and solutions in cloud computing[J]. Procedia Computer Science, 2018, 125: 691-697.

[13] 沈昌祥, 公备. 基于国产密码体系的可信计算体系框架[J]. 密码学报, 2015, 2(5): 381-389.
SHEN C X, GONG B. The innovation of trusted computing based on the domestic cryptography[J]. Journal of Cryptologic Research, 2015, 2(5): 381-389.

[14] 何欣枫, 田俊峰, 刘凡鸣. 可信云平台技术综述[J]. 通信学报,

2019, 40(2): 154-163.
HE X F, TIAN J F, LIU F M. Survey on trusted cloud platform technology[J]. Journal on Communications, 2019, 40(2): 154-163.

[15] 刘川意, 王国峰, 林杰, 等. 可信的云计算运行环境构建和审计[J]. 计算机学报, 2016, 39(2): 339-350.
LIU C Y, WANG G F, LIN J, et al. Practical construction and audit for trusted cloud execution environment[J]. Chinese Journal of Computers, 2016, 39(2): 339-350.

[16] 田俊峰, 李天乐. 基于 TPA 云联盟的数据完整性验证模型[J]. 通信学报, 2018, 39(8): 113-124.
TIAN J F, LI T L. Data integrity verification based on model cloud federation of TPA[J]. Journal on Communications, 2018, 39(8): 113-124.

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

[18] 田俊峰, 王彦翉, 何欣枫, 等. 数据因果一致性研究综述[J]. 通信学报, 2020, 41(3): 154-167.
TIAN J F, WANG Y B, HE X F, et al. Survey on the causal consistency of data[J]. Journal on Communications, 2020, 41(3): 154-167.

[19] SPIROVSKA K, DIDONA D, ZWAENEPOEL W. PaRiS: causally consistent transactions with non-blocking reads and partial replication[J]. arXiv Preprint, arXiv: 1902.09327, 2019.

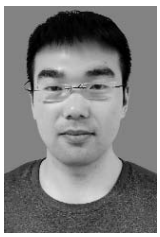
[20] 田俊峰, 常方舒. 基于 TPM 联盟的可信云平台管理模型[J]. 通信学报, 2016, 37(2): 1-10.
TIAN J F, CHANG F S. Trusted cloud platform management model based on TPM alliance[J]. Journal on Communications, 2016, 37(2): 1-10.

[21] 刘明达, 拾以娟, 陈左宁. 基于区块链的分布式可信网络连接架构[J]. 软件学报, 2019, 30(8): 2314-2336.
LIU M D, SHI Y J, CHEN Z N. Distributed trusted network connection architecture based on blockchain[J]. Journal of Software, 2019, 30(8): 2314-2336.

[22] 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.

[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。



张俊涛（1995− ），男，河北保定人，河北大学硕士生，主要研究方向为信息安全与数据一致性。

王彦翉（1994− ），男，河北邢台人，河北大学硕士生，主要研究方向为信息安全与数据一致性。