计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(6):86−92 [doi: 10.15888/j.cnki.csa.008499]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于区块链的多代理联合去重方案①

张亚男, 陈卫卫, 付印金, 徐　堃

(陆军工程大学 指挥控制工程学院, 南京 210007)
通信作者: 陈卫卫, E-mail: njcww@qq.com

摘　要: 随着多云存储市场的快速发展, 越来越多的用户选择将数据存储在云上, 随之而来的是云环境中的重复数据也呈爆炸式增长. 由于云服务代理是相互独立的, 因此传统的数据去重只能消除代理本身管理的几个云服务器上的冗余数据. 为了进一步提高云环境中数据去重的力度, 本文提出了一种多代理联合去重方案. 通过区块链技术促成云服务代理间的合作, 并构建代理联盟, 将数据去重的范围从单个代理管理的云扩大到多代理管理的多云. 同时, 能够为用户、云服务代理和云服务提供商带来利益上的共赢. 实验表明, 多代理联合去重方案可以显著提高数据去重效果、节约网络带宽.

关键词: 多云存储; 数据去重; 区块链; 云服务代理; 云计算; 智能合约

引用格式: 张亚男,陈卫卫,付印金,徐堃.基于区块链的多代理联合去重方案.计算机系统应用,2022,31(6):86–92. http://www.c-s-a.org.cn/1003-3254/8499.html

Multi-brokerage Joint Deduplication Scheme Based on Blockchain

ZHANG Ya-Nan, CHEN Wei-Wei, FU Yin-Jin, XU Kun

(School of Command and Control Engineering, Army Engineering University, Nanjing 210007, China)

Abstract: With the rapid development of the multi-cloud storage market, more and more users choose to store data in the cloud, followed by the explosive growth of duplicate data in the cloud environment. Because cloud service brokerages are independent of each other, traditional data deduplication can only eliminate redundant data on several cloud servers managed by the brokerages themselves. To further improve the data deduplication in a cloud environment, this study proposes a multi-brokerage joint deduplication scheme. Through the blockchain technology to promote the cooperation between cloud service brokerages and the construction of a brokerage alliance, the scope of data deduplication is extended from single brokerage managed clouds to multiple brokerages managed clouds. At the same time, it can bring win-win benefits to users, cloud service brokerages and cloud service providers. Experiments show that the multi-brokerage joint deduplication scheme can significantly improve the data deduplication effect and save network bandwidth.

Key words: multi-cloud storage; data deduplication; Blockchain; cloud service brokerage (CSB); cloud computing; smart contract

近年来, 随着云存储技术的迅猛发展, 越来越多的企业和个人将数据存储在云上. 同时, 为避免传统的单一云存储服务带来的供应商锁定和服务中断导致数据丢失等问题, 多云存储技术应运而生, 用户可以通过云服务代理 (cloud service brokerage, CSB) 将数据存储在不用的云上以节约成本和降低风险. 根据 IDC 最新研究报告预测, 到 2025 年全球数据总量将从 2018 年的 33 ZB 增长到 175 ZB, 其中大约 49% 的数据将存储在云环境中[1]. 而据另一项调查显示, 云环境中近 75% 的外包数据是重复的副本[2], 浪费了大量存储资源. 为降低云服务提供商 (cloud service provider, CSP) 的存储成本, 云存储服务中普遍采用了数据去重 (data deduplication) 技术. 通过将云服务器中的重复数据删除, 仅保留一个副本以实现存储空间的高效利用.

① 基金项目: 江苏省自然科学基金 (BK20191327)
收稿时间: 2021-08-12; 修改时间: 2021-09-26; 采用时间: 2021-09-29; csa 在线出版时间: 2022-02-21

当前越来越多的 ICT (information and communication technology) 厂商加入云服务代理产业领域, 其中既有单纯的 CSP, 也有传统的 IT 厂商. CSB 通过对其管理的多云进行数据去重降低成本、提高服务质量和效益. 尽管如此, 由于 CSB 之间是隔离的, CSB 及为其提供存储服务的 CSP 实际上仍然是一座座“信息孤岛”, 数据去重的范围仅限于以单一代理为中心的云存储服务器群. 因此, 云服务器上仍然有大量的冗余数据.

国内外学者对多云的重删管理进行了深入研究. MetaStorage[3] 作为一种地理分布式代理架构, 将协调器组件部署在每个代理上, 无须中央控制, 避免了单点故障, 但是需要协调器频繁交换数据信息, 管理难度大, 并且该方案默认代理属于同一个组织. ClouDedup[4] 是一种基于访问控制机制的去重方案, 但同时该方案也引入了第三方密钥服务器管理数据块密钥. 缺点是选择一个所有代理普遍信任的第三方本身具有一定难度, 同时也额外增加了存储成本. CloudShare[5] 是一种基于区块链技术的云端数据去重方案, 使用区块链将多个云服务提供商联合在一起, 跨用户进行数据去重. 其最大缺点是无法解决不同云的数据存储格式和接口不同带来的影响.

本文提出了一种基于区块链技术的多代理联合去重方案 (Blockchain based multi-brokerage deduplicating alliance, BMBDA), 将松散的 CSB 组织起来形成一个联盟, 并为联盟内的 CSB 提供不同云上所存储的数据的全局一致视图 (全局索引表), 处于联盟内的 CSB 成员通过全局视图判断待上传数据在云上是否已有重复副本. 如果没有, CSB 则将数据上传至云上, 同时更新对应的索引. 反之, 则仅需添加这次上传的数据索引, 将地址指针指向云上的唯一副本. 通过对多个云上的数据联合去重, 可以进一步节约云存储空间以降低成本. 同时, 用户不必真正上传冗余副本, 可以节省带宽并提高用户体验.

1 相关工作

1.1 基于代理的多云存储系统

多云存储, 是指依靠代理将多个供应商的不同云存储服务虚拟化联合管理, 为用户提供统一的存取接口. 多云存储系统按照布局差异可以分为集中式代理架构和分布式代理架构两种[6]. 其主要工作是, 首先对用户上传的数据进行预处理, 如分块、加密、去重等. 然后根据用户需求, 综合考量网络延迟、存取费用、安全性等因素计算出最佳策略. 最后将用户数据上传, 同时更新索引表. 通过将用户的数据合理部署在公有云、私有云或混合云上, 进一步降低存储成本, 此外还有避免单点故障导致数据丢失、保持高可用的业务连续性、提高云服务的安全可靠性等优点. 据 IBM 商业价值研究院预计, 到 2021 年将有 98% 的企业采用多云架构[7]. 随着多云存储领域的快速发展, CSB 厂商推出了大量云存储管理产品, 如表 1 所示.

表 1 CSB 各类多云管理产品对比

CSB	产 品	提供存储服务的CSP	CSP类型
Google	Anthos	AWS、Azure	公有云
IBM	Cloud Paks	AWS、Azure、IBM Cloud	公有云
MultCloud	MultCloud	Dropbox、OneDrive、AmazonS3、百度网盘等	公有云
行云管家	行云管家云管平台	阿里云、腾讯云、OpenStack、VMware等	公有云、私有云
FIT2CLOUD	cloudExplorer	百度云、腾讯云、华为云、OpenStack等	公有云、私有云
贝斯平	OpsNow	AWS、阿里云、华为云、本地IDC等	公有云、私有云

尽管云存储市场上存在很多的多云管理产品, 但是尚未发现一款能够实现将所有云存储服务 (涵盖公有云、私有云和混合云) 统一管理的产品. CSB 可以将数据去重技术应用在为自己提供存储服务的云上以消除冗余数据, 但是对于用户通过其他 CSB 存储在另外云上的重复数据则无法做到进一步的去重.

1.2 区块链

区块链 (Blockchain)[8] 的概念来源于中本聪在 2008 年提出的数字加密货币系统——比特币 (Bitcoin). 区块链的本质是一种开放的去中心化的分布式数据库, 由一系列数据区块按照产生时间有序链接而成. 每个区块由两部分构成, 区块头负责记录 Merkle 根哈希、父哈希值、时间戳、计算难度以及随机数等信息, 区块体中则包含交易计数器以及详细的交易数据, 如图 1.

区块链系统工作过程大致如下: 首先, 交易参与者将交易信息发布到 P2P 网络, 矿工节点收到交易信息, 经验证无误后将其放入交易池. 然后, 矿工节点将当前目标难度值 T 与 hashMerkleRoot 等字段组成区块头, 并在区块头中加入不同的随机值, 计算区块头哈希值, 直到此哈希值小于或等于目标值 (此过程被称作“挖

矿”). 最后, 成功计算到结果的矿工将交易池中的交易进行打包作为区块体, 与区块头封装在一起向全网广播, 由其他节点进行验证. 一旦被集体接受就无法改变, 这使得交易是不可篡改的, 并且是可信和可审计的.

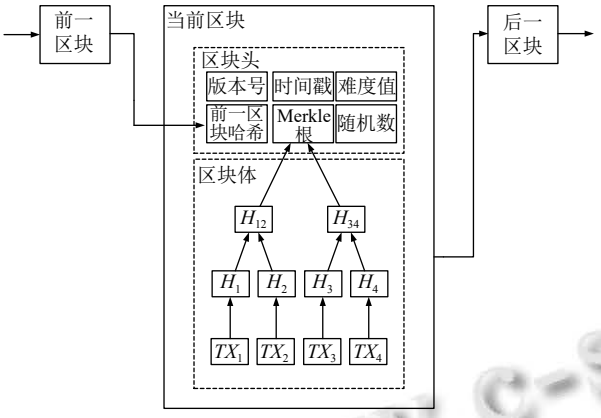

图 1 区块链结构

智能合约 (smart contract) 丰富了区块链技术的应用形式, 这个术语最早由跨领域法律学者 Szabo 在 1995 年提出, 将其定义为“一套以数字形式定义的承诺, 包括合约参与方可以在上面执行这些承诺的协议”[9]. 狭义上的智能合约一般是指部署在区块链上可以自动执行的代码, 以太坊 (Ethereum)[10] 就是一个运行智能合约的分布式平台, 这些智能合约运行在众多以太坊虚拟机上. 以太坊的出现使区块链应用不再局限于数字货币[11]. 区块链技术作为颠覆性的创新技术, 因其去中心化、不可篡改、安全性等特性, 近年来已逐步应用在到金融、版权、资产和医疗等各种领域中.

1.3 数据去重

数据去重, 也称重复数据删除, 是指在满足数据冗余度要求的前提下, 只传输和存储一个数据对象副本, 其他重复副本由指向该对象数据的指针替换, 从而降低存储空间和传输带宽开销[12]. 在云环境中数据去重的一般工作流程大致为: 首先, 服务器对已存储的数据进行哈希计算, 得到它们对应的签名值. 然后, 将这些签名值与用户待上传数据的签名值进行匹配计算, 以此判断与原存储数据是否重复. 最后, 如果重复则删除相同的数据, 且该用户无须再次上传, 并将指向原存储数据的指针返回给用户以便下次访问; 如果不重复则存储用户上传的数据, 并将指针返回给用户. 数据去重按照去重粒度可以分为文件级、块级和字节级去重, 按照工作域可以分为客户端、代理端和服务器端去重. 通过采用数据去重技术, 可以为备份系统节约 83% 的存储空间, 为主存系统节约 68% 的存储空间, 为固态硬盘节约 28% 的存储空间, 为云虚拟机中通用数据的存储节约高达 80% 的空间[13].

采用数据去重技术仅存储一个数据副本会降低安全性, 因此在数据去重中广泛应用了纠删码技术 (erasure coding, EC), 通过增加少量冗余增加系统的可靠性[14]. 其安全性可以用式 (1) 表示:

$$n = k + m \tag{1}$$

其中, 变量k表示原始数据被切分成块的数量, m表示通过编码算法额外添加的编码块的数量, n表示经过编码后创建的数据块的总数量. 对于任意m'个数据块丢失, 只要$m' < m$, 就仍然能通过剩下的$k - m'$个数据块还原完整文件数据.

2 方案设计

BMBDA 的目标是将各自独立的 CSB 联合起来, 提供跨更多云的重删服务, 如图 2 所示. 为此, 采用区块链技术解决 CSB 互不信任的问题, 保证共享数据可信且不被恶意篡改, 为联合去重提供基础.

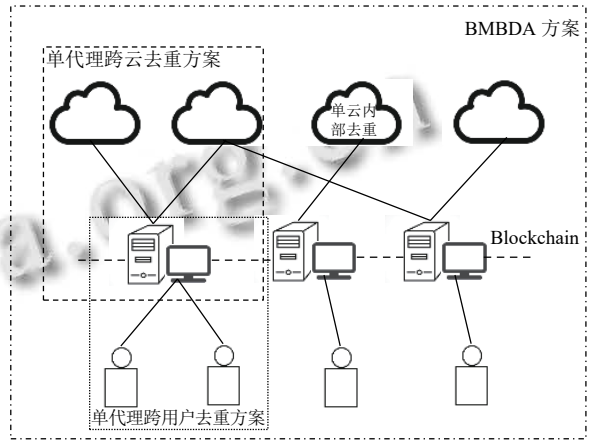

图 2 不同去重方案对比

2.1 可行性分析

一旦 CSB 加入联盟并积极工作, 它可以通过更大范围的重删消除冗余数据, 从而为 CSP 节省存储资源, 尤其是降低了边缘建设成本. 对于 CSB 来说, 通过数据去重可以减少上传的数据, 进而节约了云服务器的租用成本. 同时, 存储一份数据还可能得到额外的收入. 例如代理 a 已上传过数据 A, 当代理 b 要上传相同的数据 A 时, 代理 b 向代理 a 支付少量费用以换取指向

已存储数据 A 的地址指针, 同时无须上传整个数据 A, 这对于代理 a 和代理 b 来说是双赢的. 合作可以最大限度地提高利润, 也为用户带来更为快捷的上传体验, 因此认为 CSB 倾向于合作.

CSB 基于自身的利益考虑, 在商业环境中无法做到完全互信, 使用区块链可以很好地解决这些问题. 首先, 区块链是一个分布式的系统, 无须任何一个共同信任的中央机构即可达成共识. 其次, 区块链可以将数据快速同步至各节点, 为联盟中的 CSB 提供当前数据的强一致性全局视图. 最后, 区块链一旦达成共识几乎不可能被篡改, 且是可审计的. CSB 篡改数据将会被发现, 而且无法抵赖.

2.2 BMBDA 模型

为应对上述威胁, BMBDA 模型采用了区块链技术, 以便多个 CSB 开展合作, 而无须依赖可值得信任的中心. 与传统方案相比, 可以节约更多的存储成本, 更少的带宽占用. BMBDA 模型可以简单划分为核心工作层和文件处理层, 如图 3 所示.

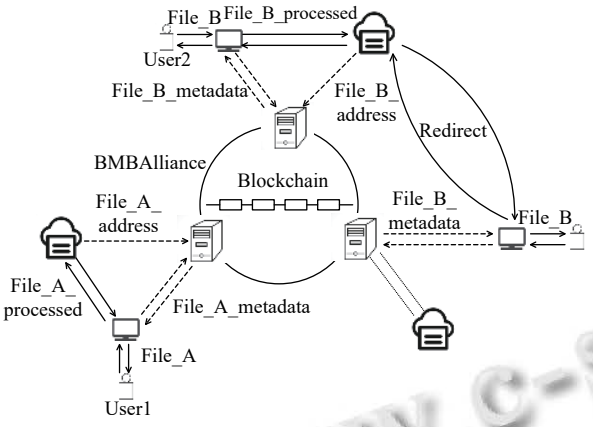

图 3 BMBDA 模型

BMBDA 模型核心工作层由 CSB 和区块链系统组成, 区块链采用以太坊框架. CSB 将文件上传、下载、删除等元数据写入区块链, 通过区块链同步至各 CSB, CSB 读取新区块内容并更新全局索引表. 通过引入区块链, 可以保证全局索引的强一致性, 使得联盟内的 CSB 在每个时刻对整个系统存储的文件都具有一致的全局视图. 当用户通过不同 CSB 上传相同文件副本时, 由最先执行上传操作 (也是实际存储文件副本) 的 CSB 返回给后执行上传操作的 CSB 该副本存储在云上的地址, 而后者则仅需向前者支付少量费用. 区块链的特殊结构使得新区块一旦被集体接受便几乎无法更改, 保证 CSB 联盟的可靠性.

文件处理层包括用户端和 CSP, 主要工作为文件的上传和下载. 其中用户端的主要执行文件哈希、通过纠删码技术分块等数据预处理任务, 以及对云上下载的数据进行读取恢复. CSP 端主要负责文件的存储, 并将存储地址通过 CSB 返回给用户.

2.3 威胁模型

本文的方案中, 处于联盟内的 CSB 共同维护一个全局视图, 因此需要考虑以下因素.

(1) CSB 在对外宣称自己将某个文件副本上传至云端后, 多个用户可能通过不同的 CSB 上传相同文件副本, 后者通过全局视图得知该副本已被上传过, 此时只需要以少量费用向前者换取文件实际存储地址即可. 假如前者为节省存储开销, 擅自将该副本在云端删除, 同时拒绝承认其恶意行为, 当其他用户想要下载该文件副本时, 将会失败.

(2) 竞争对手可能与恶意 CSB 勾结, 即使该 CSB 从未帮助任何用户上传过某个文件, 也可以在拥有全局视图的情况下假装自己也上传过该数据的相同副本, 进而窃取其他用户存储的数据.

3 数据去重

3.1 索引表设计

为便于实现数据的全局去重, 避免错误删除造成用户数据丢失, 文本专门设计了全局索引表, 用于存放用户通过代理上传文件的元数据信息. CSB 除维护自身的本地索引表外, 还共同维护一张全局索引表, 如图 4 所示. CSB 将用户上传、删除文件的元数据信息写入区块, 经区块链达成共识并产生新块后, 所有 CSB 同步读取新区块的内容并更新全局索引表. 全局索引表以文件指纹值作为行键, 存储了实际执行存储操作的代理、所有上传相同副本的代理和上传相同副本的用户信息. 当用户删除自己上传的文件时, CSB 将其对应的本地索引条目删除, 同时将用户删除文件副本的元数据信息写入区块, 经区块链同步至各 CSB, 删除全局索引表中该文件副本条目下用户及对应 CSB 的信息. 如果当前仅有该用户上传过文件副本, 则删除全局索引表中文件副本对应的整行条目.

3.2 去重算法设计

针对本文所提出的方案, 设计了多代理联合去重

算法. 首先, 计算文件签名值. 然后 CSB 在本地索引中比较签名值, 查找是否存储过相同副本. 如果存储过, 则仅需更新本地和全局索引表. 反之, 则需要比较全局索引中文件的签名值, 查找是否有其他 CSB 上传过相同副本. 如果其他 CSB 上传过相同副本, 则该 CSB 支付少量费用从实际执行存储的 CSB 获取文件副本在云中的存储地址, 同时更新本地和全局索引. 为避免 CSB 在拥有全局视图的情况下非法窃取用户数据, 在此过程中实际执行存储操作的 CSB 会对执行上传操作的 CSB 进行随机提问, 如算法 1 中第 12–15 行所示. 最后, 如果所有 CSB 都未上传过该文件副本, 则该 CSB 上传该副本, 并在本地和全局索引中增加对应条目. 算法如算法 1 (算法符号标记及含义详见表 2).

File_finggerprint (文件指纹)	CSB_save_name (实际执行存储操作的云代理)	CSB_put_name (所有上传相同副本的云代理)	User_put_name (所有上传相同副本的用户)

(a) Global_index_table (全局索引表)

User_name 用户名	File_name 文件名	File_finggerprint 文件指纹	File_address 地址指针

(b) Local_index_table (本地索引表)

图 4 索引表设计

算法 1. 多代理联合去重算法

输入: File
输出: Update Global_index_table, Local_index_table, upload file

```
for each file k do
 fp= hash(k) //计算文件 k 的签名值
 rl= search the fp in Local_index_table //在本地索引中查找k的签名值
 if rl==true then
  update the record in Local_index_table //更新本地索引
  update the record in Global_index_table //更新全局索引
 else
  rg=search the fp in Global_index_table //在全局索引中查找 k 的签名值
  if rg==true then
   fpn_CSBSave= SHA(k+nonce) //加入随机数与文件合并哈希
   the CSB_save send nonce to the CSB_put //返回随机数
   fpn_CSBPut=SHA(k+nonce) //计算随机数与文件哈希
   if fpn_CSBPut==fpn_CSBSave then //判断是否合法上传
    update the record in Global_index_table //更新全局索引
    the CSB_save send file_address of k to the CSB_put //返回文件 k 在云中的存储地址
  else
   the CSB upload the file k to cloud //代理将用户文件 k 上传至云
   insert a new record in Local_index_table //更新本地索引
   insert a new record in Global_index_table //更新全局索引
end
```

表 2 算法符号标记及含义

符号标记	含义
fp	文件k签名值
Local_index_table	本地索引表
Global_index_table	全局索引表
CSB_save	实际执行文件k存储的CSB
CSB_put	正在为用户存储文件k的CSB
nonce	随机数 (与文件合并哈希后作为提问问题)
fpn_CSBSave	实际执行文件k存储的CSB得到的标准答案
fpn_CSBPut	正在为用户存储文件k的CSB的回答
file_address	文件k在云中的存储地址

4 安全性分析

针对上文提到的威胁模型, 及其可能带来的问题进行安全性分析:

首先, 由于区块链的特殊结构使得篡改区块内容几乎不可能, 因此 CSB 一旦将某个文件副本上传至云, 并通过区块链同步到各 CSB, 他的行为将被所有节点记录下来. 由于区块链可追溯的特性, 尽管实际执行存储的 CSB 可能为节约成本删除云端文件, 并且不将该行为写入区块, 但是可以被很容易发现并且无法抵赖. 由于恶意行为造成的损失远大于合法行为带来的收益, 并且总会被发现, 因此 CSB 几乎不可能恶意删除存储在云中的文件.

其次, 尽管联盟内的所有 CSB 都拥有存储文件的全局视图, 但是 CSB 仍然无法通过其他 CSB 获取其自身从未上传过的文件. 因为全局索引表是通过区块链同步而来, 具有较强的一致性, 虽然 CSB 可以修改自己存储的全局索引表, 但是无法篡改其他 CSB 上存储的全局索引表, 因此企图通过篡改全局视图窃取用户数据的办法是行不通的. 另外在提问环节, 实际存储文件副本的 CSB 随机产生一个随机数, 与文件合并计算 SHA-256 或者 MD5 值, 然后将随机数返回给要上传相同文件副本的 CSB. 加密算法的特性使得即使原始数据哪怕有 1 bit 不同, 也几乎不可能产生相同的哈希值. 由于恶意 CSB 和竞争对手没有整个文件数据, 因此几乎不可能正确回答问题.

5 仿真实验

5.1 实验环境

实验代码采用 Python 语言编写, 测试环境部署在 Windows 10 操作系统上, 硬件环境为 Intel(R) Core(TM) i5-10200HCPU @2.40GHz 处理器, 8 GB 内存容量. 数

据集采用搜狗实验室中的全网新闻数据 2012 版, 从中随机选取 14 000 篇新闻进行后续实验. 实验模拟了多个用户通过 3 个云代理向不同的云服务器存储文件, 区块链采用 PoW 共识算法. 使用 Python 语言的 multiprocessing 工具包, 建立 4 个进程, 进程 1–3 分别对应代理 1–3, 协助用户存储文件, 各个进程之间相互独立. 进程 4 模拟了区块链挖矿过程, 前 3 个进程分别与第 4 个进程交互, 以同步更新全局视图.

5.2 结果分析

(1) 去重效果比较

去重率是衡量系统好坏的一个重要指标, 去重率越高就越能够节省云存储空间, 提高带宽利用率, 降低云服务代理存储开销. 其计算公式如下:

$$去重率=\left(1-\frac{实际存储在云上的文件数}{所有上传的文件数}\right)\times 100\% \quad (2)$$

如图 5 所示, 对于 a、b、c 三个代理, $SUMx$表示用户通过不同代理上传的文件总数, $DUPx$表示每个代理上传文件中重复的个数. 经过单代理内部去重后, 每个代理实际上像云服务器上存储的文件数为:

$$DdSUMx=SUMx-DUPx \quad (3)$$

因此, 对于n个代理组成的系统, 整个系统的去重率可以表示为:

$$DeDupRate=1-\frac{\sum DdSUMx}{\sum SUMx} \quad (4)$$

使用多代理联合去重方案, 可以对经过单代理内部去重的数据进行二次去重, $DUPxyz$表示这些文件中仍然重复的个数. 因此, 经联合去重后, 整个系统的去重率为:

$$DeDupRate'=1-\frac{\sum DdSUMx-DUPxyz}{\sum SUMx} \quad (5)$$

由式 (5) 可以看出, 系统去重率和$DUPxyz$有关, 理想情况下, $DUPxyz$越大, 系统去重率越大. 当$DUPxyz=0$时, 系统去重率达到下界, 此时经单代理内部去重后, 代理之间向云上存储的数据没有任何一个是重复的. 当经过单代理内部去重后仍然存在重复数据, 且所有重复数据都被检测到并去重, 此时去重率达到上界.

由图 6 可以看出, 本文所提方案能够明显降低存储空间. 使用传统的单代理内部去重方案在代理 1、代理 2、代理 3 上的去重率分别为 2.97%、4.68%、9.08%, 而使用 BMBDA 方案在不同代理上的去重率分别可以达到 4.33%、9.90%、12.68%, 明显高于前一方案, 并且随着用户上传文件数量的增加, 去重率越来越高. 使用 BMBDA 方案后, 整个系统的去重率从 5.99% 提高到了 9.90%, 相应地, 上传文件副本产生的网络带宽开销也随之降低. 因此本文所提出方案可以显著提高去重效果, 为云服务代理提供更大的收益和更好的服务质量. 出于利益最大化考虑, 越来越多的 CSB 会加入代理联盟, 这也进一步佐证了本方案的可行性.

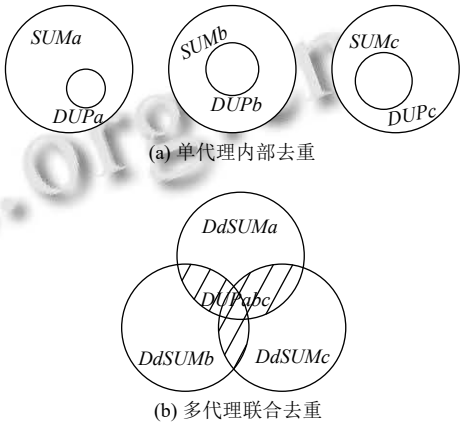

图 5 文件去重示意图

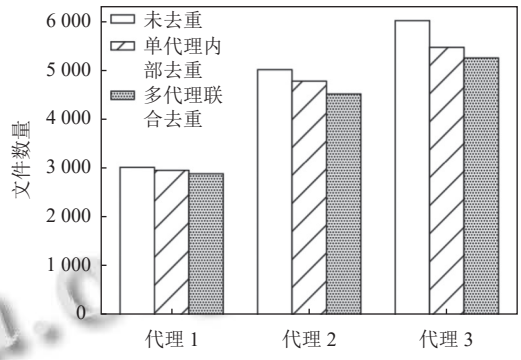

图 6 去重效果对比

(2) 性能评估

区块链的特殊结构和共识机制使得其具有效率不高的缺点, 可能影响本文所提方案的性能, 因此设计实验对文件以不同速率上传的情况进行分析. 本文适当调整了 PoW 算法“挖矿”难度, 将出块时间设置为 3 s.

由图 7 可以看出, 当文件上传速率高于 1 时, 去重率趋于平稳. 因为仿真区块链每 3 s 生成一个区块, 每个区块包含 7 笔“交易” (文件上传) 信息, 当每个代理都以 0.8 个/s 的速率上传文件时, 区块链就刚好可以将所有交易打包生成区块, 而不会存在大量交易排队等待的现象. 在文件传输速率高于 1 时, 系统去重率仍然会有极小变化, 这是因为可能存在不同用户几乎同时上传相同副本

的情况, 同时又发生在打包一个区块时段内, 然而此时全局索引表上还没有该重复副本的元数据信息.

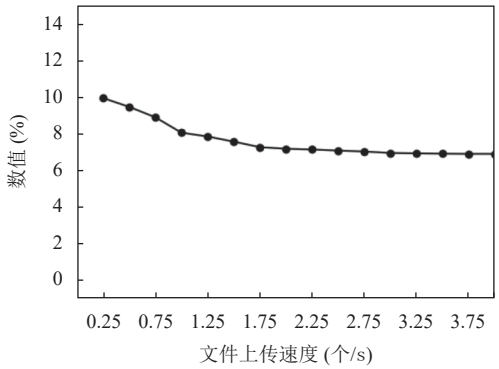

图 7 不同上传速率对系统去重的影响

综上, 虽然区块链性能瓶颈对高上传速率下的去重率有一定影响, 但并不妨碍其使用. 并且即使在文件上传速率很高时, BMBDA 方案也显示出了较好的数据去重能力, 明显好于现有的单代理数据去重方案.

6 总结与展望

针对多云环境下的数据冗余问题, 本文提出了一种基于区块链的多代理联合去重方案, 通过区块链将云服务代理联合起来, 进行覆盖更多用户和云的去重工作. 为促成这种联合引入了区块链技术而非受信任的第三方, 区块链不可篡改的特性也有助于提高系统的安全稳定性. 通过联合去重, 进一步降低了存储空间, 节约了网络带宽, 同时实现了用户、云服务代理和云服务提供商的共赢. 该方案特别适用于处于起步阶段的云服务代理商, 他们由于初期投入成本受限往往只能代理某个地区个别云服务商的存储产品, 同时由于合作能够降低存储开销并提高用户存储体验, 因此市场竞争力也大为提高. 例如, 很多用户可能将某部流行电影存到云盘上以备今后观看, 使用本文方案此过程可能仅需几秒.

未来工作应着重解决两个问题: 首先是区块链性能瓶颈问题, 导致文件元数据同步时间长、去重效率不高. 其次是负载均衡问题, 例如集中下载某个副本可能造成实际存储该副本的云提供商服务瘫痪. 针对以上问题, 一是可以通过提高区块链中的块存储容量, 提高文件元数据同步的平均时间. 例如 Bitcoin-NG[15] 技术可以在不改变块容量的基础上, 引入微区块实现辅助扩容. 二是通过选择更加合适的共识算法, 进一步提高区块链的出块速度. 三是使用纠删码技术对文件分块, 合理分配将块文件存储到多个云上, 同时避免单点存储造成性能瓶颈和安全性下降.

参考文献

1 Reinsel D, Gantz J, Rydning J. Data age 2025: The digitization of the world from edge to core. IDC White Paper#US44413318, 2018. 1–28.
2 Gantz J, Reinsel D. The digital universe decade—Are you ready. IDC White Paper, 2010. 1–16.
3 Bermbach D, Klems M, Tai S, *et al*. Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs. Proceedings of the IEEE 4th International Conference on Cloud Computing. Washington, DC: IEEE, 2011. 452–459.
4 Puzio P, Molva R, Onen M, *et al*. ClouDedup: Secure deduplication with encrypted data for cloud storage. Proceedings of the IEEE 5th International Conference on Cloud Computing Technology and Science. Bristol: IEEE, 2013. 363–370.
5 Li YD, Zhu LH, Shen M, *et al*. CloudShare: Towards a cost-efficient and privacy-preserving alliance cloud using permissioned blockchains. Proceedings of the 9th International Conference on Mobile Networks and Management. Melbourne: Springer, 2017. 339–352.
6 鲍禹含, 付印金, 陈卫卫. 多云存储关键技术研究进展. 计算机工程, 2020, 46(10): 18–32, 40.
7 IBM. 新一代的混合云管理能力. https://www.ibm.com/downloads/cas/DBVDGRVM?mhsrc=ibmsearch_a&mhq=新一代的混合云管理能力.
8 Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. https://nakamotoinstitute.org/bitcoin/. (2008-10-31).
9 Szabo N. Smart contracts. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/Szabo.best.vwh.net/smart.contracts.html.
10 Ethereum Whitepaper. A next-generation smart contract and decentralized application platform. https://ethereum.org/zh/whitepaper. (2021-11-05).
11 欧阳丽炜, 王帅, 袁勇, 等. 智能合约: 架构及进展. 自动化学报, 2019, 45(3): 445–457.
12 付印金, 肖侬, 刘芳. 重复数据删除关键技术研究进展. 计算机研究与发展, 2012, 49(1): 12–30.
13 Paulo J, Pereira J. A survey and classification of storage deduplication systems. ACM Computing Surveys, 2014, 47(1): 11.
14 敖莉, 舒继武, 李明强. 重复数据删除技术. 软件学报, 2010, 21(5): 916–929.
15 Eyal I, Gencer AE, Sirer EG, *et al*. Bitcoin-NG: A scalable blockchain protocol. Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2016. 45–59.