云端数据异地容灾验证方案研究

周洪丞,杨 超,马建峰,张俊伟
(西安电子科技大学,陕西西安 710000)

摘 要: 云存储中,应用异地容灾备份的方式,可有效防止大规模停电和天灾发生情况下的数据丢失. 目前对于异地容灾能力的保障大多基于云存储服务提供商的合同约束,还没有高效且安全的数据异地容灾能力验证机制. 针对此问题,本文提出了一种对云端数据的异地容灾能力进行验证的方案——DPBDL(Data disaster-tolerant Proving Based on Different Location),其核心思想是使用时延与数据可恢复性验证结合的方法,对云端数据的异地容灾能力进行远程验证;并且,对其安全性和性能进行了理论分析与实际测试,分析与测试结果表明该方案能够达到可证明的安全强度,并能较好的判断云端数据的异地容灾能力.

关键词: 云存储; 异地容灾; 数据完整性验证
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2016)10-2485-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.10.029

A Study Off-Site Disaster Recovery Performance of Cloud Data

ZHOU Hong-cheng, YANG Chao, MA Jian-feng, ZHANG Jun-wei
(*Xidian University, Xi'an, Shaanxi* 710000, *China*)

Abstract: Implementation of file fault tolerance is the key to preventing data loss in cloud, especially the off-site disaster recovery, which can prevent large-scale power outage and natural disaster occurs from losing data. Currently disaster recovery guarantees for multi-cloud-based storage service provider agreement. To solve these problems, we propose a remote disaster recovery capability for cloud data to validate the program——DPBDL, whose theory is to use time delay and the proofs of retrievability to check the off-site data recovery of cloud data. Then, we analysis its security and performance theoretically and practically, and the analysis result shows that its security and be able to accurately determine the cloud data off-site disaster recovery capabilities.

Key words: cloud storage; off-site disaster recovery; data integrity verification

1 引言

当使用云存储时,用户可能会要求存储服务商在多个地理位置为其存储多个文件备份. 这样有多个优点:首先,可以在某个存储数据中心地区发生大规模停电或严重自然灾害时保证数据仍然可以被用户获得;其次,可以根据请求的地理位置选择距离最近的文件备份,并发送给用户.

据我们所知,目前还没有高效且安全的数据异地容灾能力验证机制. 所以,亟需一种数据的异地容灾验证方案,在损失未发生时,对云端的数据进行异地容灾验证.

早期,RSA 公司的 Juels 和 EMC 公司的 Kaliski 提出的基于岗哨的可恢复证明系统 POR[1]. 其基本思想是首先用对称加密体制将文件加密并用纠错码编码,然后在编码后文件的一些随机位置插入和文件数据不可区分的"岗哨";检查者在数据请求时要求服务器返回一些随机位置的岗哨. 然而,这种方案的缺点在于每次需要消耗掉一个岗哨,此外,在文件需要更新时,需要找出所有未使用的岗哨,然后重新编码. POR 方案只能进行有限的验证,因此不能很好满足异地容灾验证的要求.

几乎与 POR 同时,Ateniese 等人提出了可证明数据持有(Provable Data Possession, PDP)模型[2]. PDP 通过检查一个小的采样数据块完整性,来判断更大数据的

收稿日期:2015-02-15;修回日期:2015-08-31;责任编辑:孙瑶
基金项目:国家自然科学基金青年基金(No. 61303219, No. 61472310);陕西省自然科学基础研究计划(陕西省自然科学基金)(No. 2014JQ8295);中央高校基本科研业务费(No. JB140303)

完整性.但是,此方案计算开销较大.在数据异地容灾能力验证过程中,可能需要对较大的数据进行完整性验证,采用 PDP 方法会消耗过多的时间,因而并不适用.

后来,Bowers 等人[3]建立一个协议,允许用户验证他们的数据是否被复制存储在同一地理位置的多个磁盘上.其思想是,同时读取多个文件数据块,根据读取时间判断是否在同一磁盘上.但这种方案无法扩展到多个地理位置存储中.

接下来,Benson 等[4]提出了一种验证云端存储数据是否在声明的某几个地理位置的协议.在此协议中,用户指定哪些数据中心应该存储文件备份,他们使用基于传输时间的方案,使用多个地标,向多个数据中心进行数据请求,并假设已知所有数据中心的位置,验证者如果能够在某个指定时间内获得数据,可由此判定数据中心存储有数据备份.但是,此方案存在以下问题:

(1)不能验证容灾数据完整性.由于未使用数据可恢复性验证方案,客户端不能验证文件的完整性.

(2)数据中心准确位置难以确定.该方案需要知道所有数据中心的准确位置,但实际情况中,大多云存储服务提供商不会提供数据中心的准确位置.

(3)异地容灾验证结果误差大.此方案在进行异地容灾验证时,要求服务器将部分验证文件发回到地标进行验证,造成过大的传输延迟,因此不能准确判断地理位置.

近期,Watson 等人[5]提出方案,在很大的地理范围内使用地标服务器,并利用数据可恢复性算法来验证数据的完整性,从而判断数据被完整的存储在某个区域.方案不是确定文件被存储在多个位置,而是要确定某个地理位置确实存储有文件的完整备份.但是,这种方案需要借助数量众多的地理位置已知的验证服务器,且此方案的目的是验证某个确定地理位置存储有文件的完整备份,而非验证数据的异地容灾能力,其与本文针对的问题不尽相同.

关于验证云存储服务商是否达到承诺的存储容错问题,其他研究人员针对不同的侧面也进行了相关研究.例如,文献[6]提出一种基于约束的数据地理定位,使用包括拓扑感知模型的通用模型,并对 Amazon S3 上的数据进行测试.文献[7,8]讨论云环境下共享同一台物理机的两台虚拟机可能存在信息泄露,设计一种检测两份文件是否存在于同一个物理硬盘的方案.文献[9]研究如何验证是否存储了多个备份,其特点是无需知道文件的存储布局.文献[10]允许用户指定某个地理区域内禁止存放数据,且使用基于零知识证明和基于属性的加密.文献[11,12]使用了带有 GPS 的验证者,并对文件进行分块,加入随机字符并加密存储,将地理位置验证与云端文件认证相结合.文献[13]研究了如何在不可信的与存储服务器上对损坏的文件进行自我修复.同样,也有许多方案例如文献[14,15]等,都对云端数据的完整性验证的问题进行的分析和研究,以及一些关于云端数据确认性删除的研究[16].但以上文献研究的问题与本文研究的如何验证云存储服务商是否将文件存储在不同的地理位置从而保证云端数据的异地容灾能力不同,其提出的方案不能适用于本文针对的场景与问题.

针对上述问题,本文提出了数据的异地容灾能力验证 DPBDL(Data disaster-tolerant Proving Based on Different Location)方案.方案的基本思路是:首先,利用数据可恢复性验证方案验证数据是否完整的存储在某个服务器上;其次,利用地标服务器对云端数据进行数据请求时记录数据请求和响应的时间,利用时间差计算云端数据与验证服务器的距离[17],并利用此距离判断数据是否来源于不同的地点.

2 系统存储模型

本文考虑的应用场景包括如下四个角色,如图 1 所示,用户(User,U)、云存储服务提供商(Cloud Storage Service Provider,CSP)、云存储服务器(Cloud Storage Server,S)、验证地标(Landmark,L).

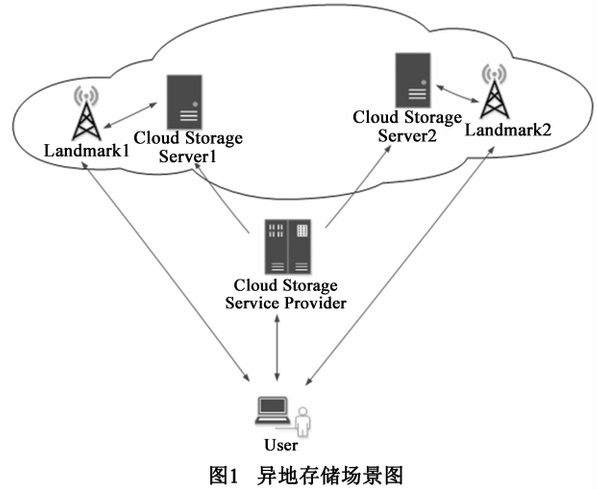

图1 异地存储场景图

用户 User 是文件拥有者,他将要在云端存储的文件进行对称加密,并添加冗余码,为其指定分节数并生成文件标签,最后将文件及文件标签一起上传至云存储服务提供商 CSP.

云存储服务商 CSP 接收到用户上传的文件和文件标签后,根据用户要求选择满足条件的云存储服务器 S,并根据云存储服务器特征码对文件标签进行重编码,最后将文件及重编码后的标签发送给对应的云存放服务器 S,将存储文件的存储服务器信息返回给用户.

云存储服务器 S 接收云存储服务提供商发送来的

文件及文件标签并对其进行存储,在接收到地标 L 的数据请求时,按照指定参数对请求进行响应,并将响应结果返回给发送请求的地标 L.

地标 L 为地理位置已知的可信服务器,其主要负责生成对云存储服务器 S 的数据请求,向 S 发送数据请求和接收 S 的响应,并对收到的响应进行验证,判断云存储服务器 S 是否确实按照协议要求在不同的地理位置完整存储用户的文件.

由于本论文的目的是要验证文件被存储在不同的地理位置,从而判断用户文件是否具有异地容灾能力.所以只需要判定数据是否存储在距离较远的多个云存储服务器即可,从而体现异地容灾能力.而将距离较近的服务器组合认为是一组整体的存储服务器,他们不具有异地容灾能力,也不需要进行异地性的区分.

3　DPBDL 方案设计型

DPBDL 方案主要包括两个部份:云端数据异地性验证和云端数据完整性验证.云端数据异地性验证完成对数据中心是否位于不同地理位置进行验证;云端数据完整性验证主要完成对云端数据的完整存储进行验证和判断的功能,最终实现数据的异地容灾能力验证功能.

DPBDL 方案特点:

(1)利用比较成熟的数据可恢复性验证方案 CPoR[18],在存储文件的同时,存储文件的标签,在进行文件的数据可恢复性验证的时候只需要传输少量的数据,即可对文件的可恢复性进行验证.从而避免了传输大量数据对判断存储服务器位置操作的判断误差.

(2)本方案利用地理位置已知的地标对云存储服务器的地理位置进行判定,结合 CPoR 方案,能够较准确地利用时延判断存储服务器的距离,并区分不同的存储服务器.

下面先给出云端数据异地性验证的子方案设计,然后将其与云端数据完整性验证子方案进行统一,并给出 DPBDL 整体方案的详细设计.

3.1　云端数据异地性验证子方案

我们假设云存储服务提供商 CSP 提供的满足用户要求的存储位置为 s_1 和 s_2. 首先,我们假设服务商并没有恶意,而只是为了节省费用而违反约定,从而没有按照约定在指定的不同位置进行数据存储.同时,我们做出如下假设:

假设 1　云存储服务提供商 CSP 提供的存储服务器位置都是已知的,并且,所有的数据都被存储在这些数据中心中.

假设 2　CSP 的不同数据中心之间没有专用的高速网络连接.

假设 3　对于每一个存储服务器,都有一个已知地理位置的地标 L,且地标 L 可以直接与所有数据中心进行相互通信以及存取数据.

位置确定具体方案如下:

(1)已知存储服务器 s_1 和 s_2 的大体方位,分别在 s_1 和 s_2 附近设置地标 L_1 和 L_2,两地标坐标分别为 (x_1, y_1) 和 (x_2, y_2). L_1 和 L_2 之间距离:$S=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$.

(2)测量两地标 L_1 和 L_2 之间的传输延迟,经过多次测量,得到延迟时间均值为 t. 因此,传输延迟可由计算得到 $v=\dfrac{S}{t}=\dfrac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{t}$.

(3)从 L_1 向 s_1 和 s_2 请求数据,延迟时间为 t_{11} 和 t_{21};从 L_2 向 s_1 和 s_2 请求数据,延迟时间为 t_{12} 和 t_{21}. 根据这些时间,可以计算得到数据中心距离 L_1 和 L_2 的距离为:

s_1 到 L_1 距离

$$r_{11}=v\cdot t_{11}=\frac{t_{11}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{t}$$

s_2 到 L_1 距离

$$r_{21}=v\cdot t_{21}=\frac{t_{21}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{t}$$

s_1 到 L_2 距离

$$r_{12}=v\cdot t_{12}=\frac{t_{12}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{t}$$

s_2 到 L_2 距离

$$r_{22}=v\cdot t_{22}=\frac{t_{22}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{t}$$

(4)通过测量的时间延迟得到 L_1、L_2 之间的距离,可以判断测得数据是否正常:

如果 $r_{11}+r_{22}>S$,可以判定最终画图结果得到的区域会有相交,如图 2 所示;

如果 $r_{11}+r_{12}<S$ 或 $r_{21}+r_{22}<S$,可以判定无法得到 s_1 或 s_2 的可能区域,因为会出现不相交区间,如图 2 所示;

其他情况,能得到两个不相交的区域,判定为 s_1 和 s_2 的可能区域,如图 3 所示.

(5)由此可以得到四个圆形方程,分别为:

$$\begin{aligned}(x-x_1)^2+(y-y_1)^2&=r_{11}^2\\(x-x_1)^2+(y-y_1)^2&=r_{21}^2\\(x-x_2)^2+(y-y_2)^2&=r_{12}^2\\(x-x_2)^2+(y-y_2)^2&=r_{22}^2\end{aligned}\tag{1}$$

(6)首先考虑根据画图得到数据中心 s_1 的可能位置.计算两圆相交区域中心点距 L_1 距离应该为 r_{11} −

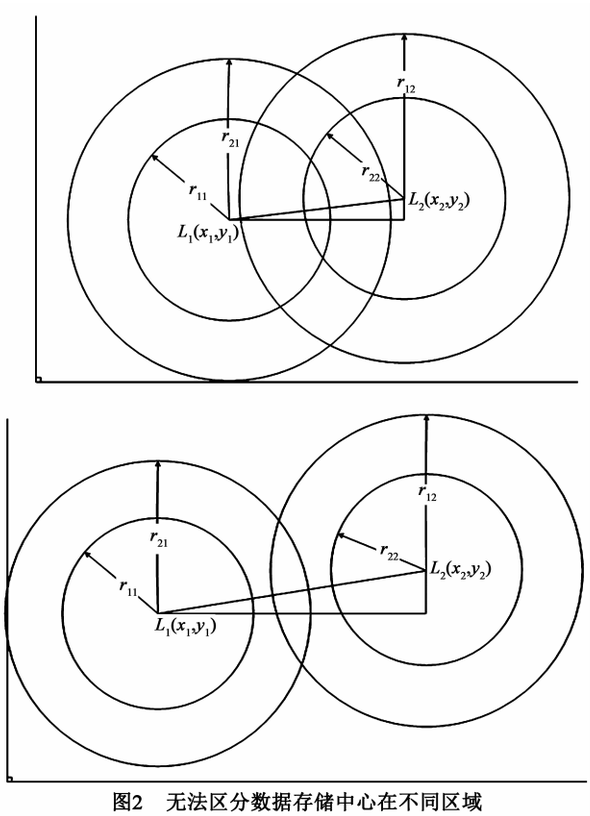

图2 无法区分数据存储中心在不同区域

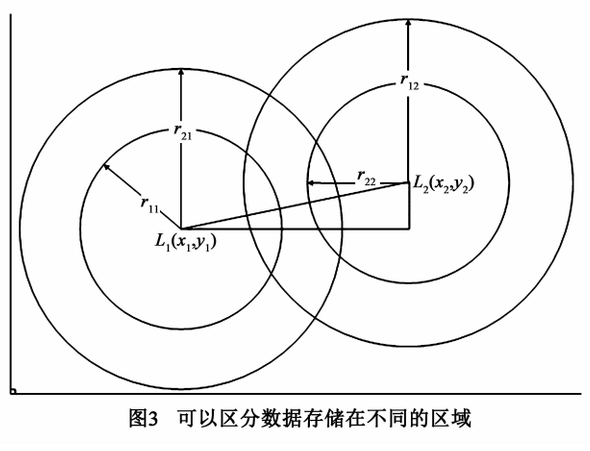

图3 可以区分数据存储在不同的区域

$(r_{11}+r_{12}-S)/2=(S+r_{11}-r_{12})/2$. 根据三角形相似原则得到公式

$$\frac{S+r_{11}-r_{12}}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}=\frac{y_{c1}-y_1}{y_2-y_1},$$

以及$$\frac{S+r_{11}-r_{12}}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}=\frac{x_{c1}-x_1}{x_2-x_1}.$$

计算其坐标为

$$(x_{c1},y_{c1})=\left(x_1+\frac{(S+r_{11}-r_{12})*(x_2-x_1)}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}},\right.$$
$$\left.y_1+\frac{(S+r_{11}-r_{12})*(y_2-y_1)}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}\right).$$

同理,可得到计算 s_2 中心位置的公式如下

$$\frac{S+r_{21}-r_{22}}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}=\frac{y_{c2}-y_1}{y_2-y_1},$$

以及$$\frac{S+r_{21}-r_{22}}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}=\frac{x_{c2}-x_1}{x_2-x_1}.$$

计算其坐标为

$$(x_{c2},y_{c2})=\left(x_1+\frac{(S+r_{21}-r_{22})*(x_2-x_1)}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}},\right.$$
$$\left.y_1+\frac{(S+r_{21}-r_{22})*(y_2-y_1)}{2\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}\right).$$

如果一组数据库在位置上彼此非常接近,我们可以将在某个地理区域内的数据中心分为一个组,并且验证在这个区域内至少存在一份数据备份,这也是用户想要达到的目的.

3.2 DPBDL 方案详细设计

DPBDL 方案的流程如图 4 所示.

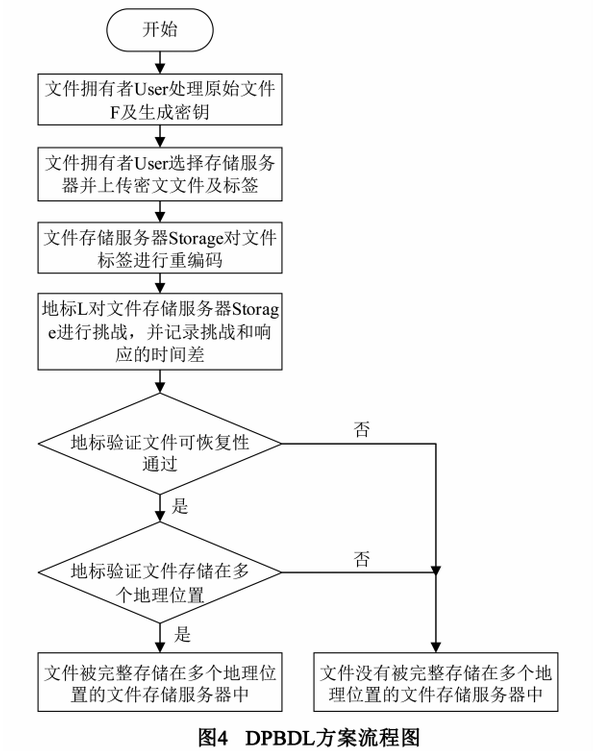

图4 DPBDL方案流程图

其详细设计步骤如下:

步骤 1 利用加盐数据$\{\alpha_1,\alpha_2,\cdots,\alpha_s\}$和加密后的文件 F',分别计算密文文件 F'每一块文件对应的标签 σ_i,其计算公式如下:

$$\sigma_i\leftarrow f_{k_{prf}}(i)+\sum_{j=1}^{s}\alpha_j m_{i,j} \tag{2}$$

步骤 2　文件拥有者 User 根据需求选择一个存储服务器集合 C,将密文文件 F' 及其标签 σ_i 一起上传云储存服务商,云存储服务商将密文文件 F' 及其标签 σ_i 发送至服务器集合 C 中的每一个文件存储服务器.

步骤 3　云存储服务商 Provider 为每个文件存储服务器 Storage 分配一个唯一的服务器标记 ρ,并利用 ρ 对文件标签 σ_i 进行重编码,计算过程表示如下.

$$\sigma_{\rho,i} \leftarrow \sigma_i + h_{sk_r}(\rho) \tag{3}$$

其中,sk_r 是文件标签进行重编码时使用的密钥,ρ 是存储服务器的标签,$h_{sk_r}(\rho)$ 是以 ρ 为输入的哈希算法,σ_i 是文件上传者生成的第 i 块文件的文件标签,$\sigma_{\rho,i}$ 是存储服务器 ρ 对文件标签 σ_i 进行重编码后得到的新文件标签.

步骤 4　地标服务器 L 对文件存储服务器 Storage 进行数据请求.

(a)用户根据文件存储服务器的位置,在每个存储服务器的最近位置设置一个地标服务器 L,根据两地标服务器之间的距离 Len,测量两个地标服务器之间时延为 t,得到两地标服务器间数据传输速率 $V=\dfrac{Len}{t}$;

(b)地标服务器 L 使用伪随机数生成方案,生成一组数据请求 Q,发送给某个文件存储服务器 Storage,并由存储服务器计算标签响应值和文件块响应值,标签的响应值 σ_ρ 计算如下:

$$\sigma_\rho \leftarrow \sum_{(i,v_i)\in Q} v_i\,\sigma_{\rho,i} \tag{4}$$

文件块响应值 μ_j 的计算公式如下:

$$\mu_j \leftarrow \sum_{(i,v_i)\in Q} v_i m_{i,j} \tag{5}$$

文件存储服务器 Storage 将计算得到的标签响应值 σ_ρ 和文件响应值$\{\mu\}$发送给地标服务器 L,地标服务器 L 接收响应并记录时间为 t_{re}.

步骤 5　地标服务器 L 判断文件的完整性和地理位置.

(a)地标服务器 L 使用收到的文件响应值$\{\mu\}$,得到结果标签 σ_L,其计算公式如下:

$$\sigma_L \leftarrow \sum_{j=1}^{s} \alpha_j \mu_j + \sum_{(i,v_i)\in Q} v_i\left(f_{k_{prf}}(i) + h_{sk_r}(\rho)\right) \tag{6}$$

(b)地标服务器 L 验证结果标签 σ_L 与收到的标签响应值 σ_ρ 是否相同,根据发送数据请求时间 t_{ch} 和接收响应的时间 t_{re},计算存储服务器与地标服务器 L 的距离:$r=V*(t_{re}-t_{ch})$;

(c)文件拥有者 User 分别以所述的地标服务器 L、L'的位置为圆心,以 r 与 r'为半径作圆,用该两圆的交汇区域作为存储服务器 Storage 的测量位置.

步骤 6　文件拥有者 User 按照步骤 5,计算服务器集合 C 中所有存储服务器的测量位置,判断所有存储服务器测量位置是否满足 User 对地理位置的要求.

4　DPBDL 方案安全性分析

假设协议实例涉及一个分为 n 块的文件,文件有秘密值$\{\alpha_1,\alpha_2,\cdots,\alpha_s\}$,包含分区$\{m_{ij}\}$,并且由 St 产生的块签名是$\{\sigma_i\}$. 假设 $Q=\{(i,v_i)\}$是引起数据请求者退出的查询,敌手对查询的响应是$\mu_1',\mu_2',\cdots,\mu_s'$以及 σ'. 期望的响应,即应该从诚实的证明者处获得的响应为 μ_1,$\mu_2,\cdots,\mu_s$ 和 σ,其中 $\sigma\leftarrow\sum_{(i,v_i)\in Q} v_i\sigma_i$ 而 $\mu_i\leftarrow\sum_{(i,v_i)\in Q} v_i m_{i,j}$,$1\leqslant j\leqslant s$. 如果敌手的响应满足验证者,即如果 $\sigma'\leftarrow\sum_{(i,v_i)\in Q} v_i r_{k_{prf},i}+\sum_{j=1}^{s}\alpha_j u_j'$,其中 $r_{k_{prf},i}$ 是替代 $f_{k_{prf}}(i)$的随机值,但是至少存在一个 j 使得 $\mu_j'\neq\mu_j$,数据请求者退出.(如果对于所有的 j,都有 $\mu_j'=\mu_j$,但是 $\sigma'\neq\sigma$,验证等式不可能成立,所以不必担心这种情况.)

方案的正确性,期待得 σ 值以及$\{\mu_j\}$也能够满足验证等式,我们有 $\sigma\leftarrow\sum_{(i,v_i)\in Q} r_{k_{prf},i}+\sum_{j=1}^{s}\alpha_j u_j$. 让 $\Delta\sigma\overset{\text{def}}{=}\sigma'-\sigma$ 而 $\Delta\mu_j\overset{\text{def}}{=}\mu_j'-\mu_j$,$1\leqslant j\leqslant s$,并且从 σ'中减去 σ 验证等式,我们得到:

$$\Delta\sigma = \sum_{j=1}^{s}\alpha_j\Delta\mu_j \tag{7}$$

敌手第一次执行协议,$\{\alpha_1,\alpha_2,\cdots,\alpha_s\}$在它看来是相互独立的:他们不再被加密到标签中,他们唯一出现是在计算 $\sigma_i\leftarrow r_{k_{prf},i}+\sum_{j=1}^{s}\alpha_j m_{ij}$;但是随机值 $r_{k_{prf},i}$替代 $f_{k_{prf}}(i)$意味着 σ_i 是独立于$\{\alpha_1,\alpha_2,\cdots,\alpha_s\}$的. 因此,他选择的$\{\Delta\mu_j\}$和 $\Delta\sigma$ 退出的概率是 $1/p$,继续执行的概率是 $1-1/p$.

我们看到数据请求者在攻击者 A 的 q 次协议执行中退出的概率为:

$$1-\left(1-\frac{1}{p}\right)\left(1-\frac{1}{p-1}\right)\cdots\left(1-\frac{1}{p-q+1}\right) \tag{8}$$

这个概率是可以忽略的.　证毕

5　DPBDL 方案性能分析与评估

据我们所知,目前还没有类似针对云端数据的异地容灾能力进行验证的可行方案,因此本文与现有的方案没有可比性,所以本文的测试方案只是分别对云端数据的异地存储验证 DPBDL 方案进行测试及 DPBDL 方案中各参数对验证的影响进行测试和分析.

5.1　DPBDL 方案性能的理论分析

5.1.1　DPBDL 方案的整体性能理论分析

DPBDL 数据完整性验证子方案的性能:采用 CPOR[11],其在初始化阶段的计算开销为 $O(ns)$,在响应阶段的计算开销为 $O(s)$,验证阶段的计算开销为

$O(s)$,传输开销为 $O(s)$,存储开销为 $O(n)$,其中 n 为文件的分块数,s 为文件的分节数.

总体看来,DPBDL 方案的整体计算开销为 $O(ns)$,方案的计算开销是很小的,即此方案是可行的.

5.1.2 初始化阶段与参数选择的关系

在计算标签过程中使用的是在某个整数域内的计算,因此得到的标签大小不会超过计算域中的最大数.设原文件大小为 m,指定的分节数为 s,每一小节的文件大小为 λ. 计算得到的文件块数以及标签数为:$n=\frac{m}{s\times\lambda}$.

计算中选用 λ 位的素数 p,而计算中的数据都选择整数域 Z_p 中的数据. 因此,生成的标签数量与文件大小 m 成正比,而与文件块的分节数成反比. 可知每个标签数据的长度应为 λ,标签数量为 n. 因此,整个文件的标签大小应该为 $n\times\lambda=\frac{m}{s\times\lambda}\times\lambda=\frac{m}{s}$. 由此可见,整个文件的标签大小与文件大小 m 成正比,与分节数 s 成反比.

每个文件块的标签计算需要 s 次乘法和 s 次加法,假设每次加法需要处理时间为 t_1,每次乘法需要的计算时间是 t_2. 则可以计算每个文件标签计算时间为 $s\times(t_1+t_2)$,而所有文件标签计算时间之和为

$$n\times s\times(t_1+t_2)=\frac{m}{s\times\lambda}\times S\times(t_1+t_2)=\frac{m}{\lambda}\times(t_1+t_2)$$

即,文件的所有标签计算时间应该是与文件的分节数无关的,仅与文件大小成正比.

5.1.3 数据请求和验证阶段与参数选择的关系

生成响应的过程中,响应数据大小与数据块分节数即每节大小相关. 设数据分节数为 s,每一小节的文件大小为 λ. 生成的响应数据大小应为$(s+1)\times\lambda$,在分节数和每节数据大小固定的情况下,计算得到的响应数据大小是相同的.

为了避免响应数据的大小不同对响应传输时间造成影响,从而影响对存储文件距离的估算,我们将响应数据的前 1024 位作为响应标记首先返回至地标,再将响应数据整体返回并与响应标记对比. 因此,响应标记的返回时间应该与文件大小和分节数等无关,而只与云端存储文件与地标的距离有关.

此方案中,我们设共有 n 个数据块,每次数据请求 q 块数据,需要数据请求的数据块数为 c 块,我们需要对存储服务器进行 t 次数据请求来达到 c 块的覆盖率. 我们设每次数据请求都是相互独立的,一个数据块在 t 次数据请求中都未被抽到的概率为$\left(1-\frac{q}{n}\right)^t$,$n$ 块数据中的 c 块被数据请求的概率如下所示:

$$n\times\left(1-\left(1-\frac{q}{n}\right)^t\right)\geqslant c$$

$$\left(1-\frac{q}{n}\right)^t\leqslant\frac{n-c}{n} \tag{9}$$

可得:

$$t\geqslant\log_{(1-q/n)}\left(\frac{n-c}{n}\right) \tag{10}$$

n、q 和 c 都已知的情况下,可以得到 t 的计算式为:

$$t=\left|\log_{(1-q/n)}\left(\frac{n-c}{n}\right)\right| \tag{11}$$

这里我们假设数据块数 $n=1000$,每次数据请求是数据块数为 20,如果需要数据请求的数据块总数为 459 块[19~21],代入上式计算得到 $t=31$,即需要 31 次测试可以在概率上覆盖 459 块数据块.

5.2 DPBDL 方案实际测试与结果分析

5.2.1 测试场景

为了测试方案对异地存储文件的性能,我们租用 4 台云服务器,服务器具体参数如表 1 所示.

表 1 云服务器参数

主机	A	B	C	D
服务商	阿里云	阿里云	迅速互联	美橙互联
地理位置	北京	杭州	北京	上海
CPU	Xeon 单核 2.4GHz	Xeon 单核 2.4GHz	Xeon 单核 2.4GHz	Xeon 双核 2.4GHz
内存	512MB	512MB	512MB	1GB
硬盘	20G	20G	20G	70G
带宽	1Mbps	1Mbps	1Mbps	5Mbps
OS	ubuntu 12.04	ubuntu 12.04	CentOS 5.8	ubuntu 12.04
实验语言	Python2.7	Python2.7	Python2.7	Python2.7

5.2.2 测试方案

在使用 DPBDL 方案对云端数据异地容灾能力进行测试的过程中,我们需要云端数据异地性以及云端数据完整性验证的参数选择进行测试.

首先,我们分别使用云端主机 A、B、C 和 D 对数据的异地性进行测试;接着对数据可恢复性验证过程中的各个参数进行测试分析,分别包括不同分节数、不同文件大小和不同数据请求组数,具体方案分为如下四点.

(1)数据的异地性测试.

(*a*)学习阶段:使用主机 A 和 B 作为存储服务器和地标服务器,并分别进行 100 组测试;

(*b*)测量时间:使用主机 A 和主机 B 作为地标验证服务器,主机 C 和主机 D 作为数据存储服务器,分别进行 100 组测试,并选择 100 组测试结果的中值作为每两

个主机之间的延迟时间.

(2)对于固定大小的文件,分别将文件分节数设置为 100、200、400、800、1600、3200 和 6400,并验证产生数据标签的大小、计算标签的时间、响应数据的大小、响应数据的计算时间、响应传输时间以及验证时间.

(3)针对不同大小的文件,采用相同的分节数,文件大小包括 100KB、1MB、10MB、100MB 和 1GB 的文件,分别测试响应数据的大小和验证时间.

(4)针对相同大小的文件和相同的分节数,采用不同的数据请求组数,分别测量响应大小、数据请求时间、响应计算时间、传输时间以及验证时间.

5.2.3　测试结果与分析

(1)数据异地性验证方案准确性测试

(*a*)学习阶段:结果如图 5 所示.

为了学习阶段的准确性,我们分别进行了 100 组测试,分别得到测试阶段时间的最低值、最高值、平均值和中值. 计算得到,L_1 对 L_2 进行测试的时间最小值为 34194μs,中值为 34364μs,L_2 对 L_1 进行测试的时间最小值为 34098μs,中值为 34328μs,我们使用测量结果中的两个中值的平均值 34346μs 作为测试结果,如图 6 所示.

组号	L1-L2	L2-L1	组号	L1-L2	L2-L1	组号	L1-L2	L2-L1
1	34412	34465	35	34204	34336	68	34379	34334
2	34196	34205	36	34444	34334	69	34346	34319
3	34428	34238	37	34348	34426	70	34444	34246
4	34346	34418	38	34496	34346	71	34362	34466
5	34398	34290	39	34352	34324	72	34388	34368
6	34320	34331	40	34487	34479	73	34269	34257
7	34331	34218	41	34370	34375	74	34355	34495
8	34291	34275	42	34345	34317	75	34304	34467
9	34499	34303	43	34363	34368	76	34267	34322
10	34426	34374	44	34343	34361	77	34290	34376
11	34338	34398	45	34410	34288	78	34415	34278
12	34540	34364	46	34463	34262	79	34356	34537
13	34612	34227	47	34382	34221	80	34284	34331
14	34539	34374	48	34468	34340	81	34231	34220
15	34469	34376	49	34361	34259	82	34397	34466
16	34396	34469	50	34419	34400	83	34312	34313
17	34318	34326	51	34424	34325	84	34252	34280
18	34448	34250	52	34314	34486	85	34263	34433
19	34454	34285	53	34289	34218	86	34435	34098
20	34495	34393	54	34450	34310	87	34315	34308
21	34210	34337	55	34325	34252	88	34368	34299
22	34194	34261	56	34229	34296	89	34401	34397
23	34459	34476	57	34372	34441	90	34452	34237
24	34374	34393	58	34353	34240	91	34262	34342
25	34452	34198	59	34368	34252	92	34347	34292
26	34360	34319	60	34342	34204	93	34252	34206
27	34221	34525	61	34397	34309	94	34258	34203
28	34467	34396	62	34293	34470	95	34420	34436
29	34434	34301	63	34360	34271	96	34303	34351
30	34413	34263	64	34308	34321	97	34379	34329
31	34275	34286	65	34592	34192	98	34365	34381
32	34417	34390	66	34405	34396	99	34411	34331
33	34213	34469	67	34254	34413	100	34240	34390
34	34427	34271						

图5　数据请求阶段测试时间

由图 6 可以看出,使用 L_2 对 L_1 进行数据请求的时间无论最大值、最小值或平均值都小于 L_1 对 L_2 的数据请求. 经测试发现,两个方向的路由不完全相同,因此造成了一定的差异,但由于差异很小,可以忽略不计.

(*b*)测量阶段:结果如图 7 所示.

由图 7 可以看出无论 L_1 或 L_2 对存储服务器 S_1 进行数据请求时,都会有一些测试时间远大于其他值,由

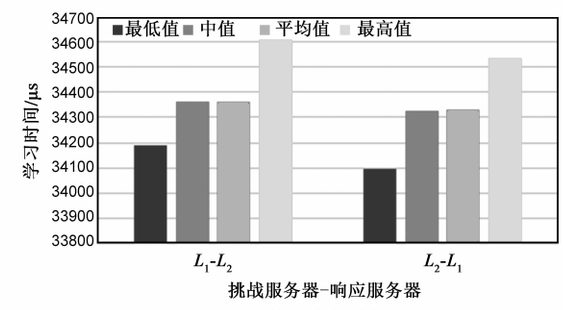

图6　学习阶段时间

组号	L1-S1	L1-S2	L2-S1	L2-S2	组号	L1-S1	L1-S2	L2-S1	L2-S2	组号	L1-S1	L1-S2	L2-S1	L2-S2
1	97524	30581	33935	6679	35	3562	30391	34632	6630	68	5186	30359	37322	6534
2	102157	30450	41165	6538	36	3440	30516	34113	6458	69	6262	30419	34009	6466
3	3546	30338	39807	6475	37	3464	30296	33667	6529	70	65309	30415	34840	6503
4	3368	30394	33073	6499	38	4874	30353	36710	6502	71	3321	30373	41588	6528
5	3403	30350	34024	6686	39	69245	30338	88073	6453	72	3532	30359	33730	6510
6	3688	30388	33760	6515	40	4552	30338	34106	6827	73	3343	30431	36380	6490
7	6056	30400	33438	6548	41	3393	30471	34397	6606	74	4210	30407	34872	6511
8	3526	30418	35083	6487	42	3770	30280	34936	6564	75	3545	30357	32671	6716
9	5803	30511	46567	6493	43	3296	30476	32840	6623	76	3313	30387	49475	6518
10	3429	30306	33310	6481	44	5648	30504	85313	6482	77	3937	30394	33228	6521
11	3978	30414	34108	6506	45	3316	30384	33503	6529	78	3429	30307	33630	6443
12	3583	30314	33615	6473	46	3442	30391	33972	6516	79	6437	30418	34132	6612
13	4237	30447	33572	6511	47	88631	30367	33317	6508	80	3808	30306	33147	6501
14	67315	30358	74088	6428	48	3973	30329	34176	6609	81	3984	30436	33880	6586
15	3965	30320	33897	6491	49	3527	30430	33132	6633	82	3474	30398	33917	6475
16	8823	30329	33733	6522	50	3387	30368	39242	6555	83	3413	30358	34614	6483
17	3445	30411	33710	6488	51	11618	30531	33310	6535	84	3503	30251	32633	6486
18	10903	30391	37767	6444	52	3783	30390	34438	6684	85	3581	30313	35010	6538
19	4771	30282	69547	6556	53	5099	30351	32993	6542	86	5342	30271	33181	6556
20	7162	30247	34323	6479	54	3810	30454	35452	6444	87	3351	30352	33567	6504
21	3901	30361	33358	6498	55	9777	30425	39039	6624	88	68432	30392	36908	6532
22	3242	30416	33246	6533	56	3405	30303	33424	6498	89	3583	30381	33359	6502
23	3671	30268	33107	6501	57	14303	30383	33147	6481	90	4182	30390	33685	6565
24	3476	30310	33741	6488	58	55607	30312	35423	6561	91	3336	30463	34840	6495
25	59522	30370	32827	6505	59	6350	30431	34764	6493	92	4702	30314	36648	6533
26	4679	30357	37891	6495	60	3553	30206	33741	6505	93	4129	30296	33005	6493
27	3822	30315	33356	6482	61	3429	30424	34297	6555	94	3621	30295	32916	6507
28	4104	30330	37526	6517	62	3740	30347	40186	6595	95	3390	30343	43482	6493
29	3507	30312	32829	6598	63	9021	30370	33241	6483	96	3705	30511	105871	6598
30	3944	30452	33089	6481	64	3410	30474	35496	6520	97	3604	30407	93476	6573
31	5998	30312	33978	6534	65	3706	30438	34146	6509	98	3524	30182	33387	6499
32	6551	30340	33805	6484	66	3240	30461	33871	6469	99	3349	30317	34096	6559
33	3844	30400	34238	6556	67	3357	30402	32886	6536	100	8002	30321	33652	6557
34	3597	30394	34244	6540										

图7　测试阶段时间

此我们推测,迅速互联所提供的服务器网络情况并不稳定.

我们在百度地图上测量北京与杭州之间的距离为 1130km. 根据之前学习阶段和测试阶段的时间,可以在百度地图上画出相应的距离半径. 根据测试阶段得到的时间,可以计算出地标与存储服务器之间的距离,分别得到 L_1 – S_1 距离为 117.3km,L_1 – S_2 距离为 998.5km,L_2 – S_1 距离为 1101.2km,L_2 – S_2 距离为 214km. 在地图上分别以 L_1 和 L_2 为圆心,以相应距离为半径画圆,得到图 8.

由图 8 可以看出,有北京和杭州的地标服务器发出了数据请求,将位于北京和上海的存储服务器进行数据请求的时间,转换为在地图上的显示距离. 根据此测试时间计算得到的存储服务器位于相距很远的不同地理位置.

我们可以使用之前得到的公式,计算出两个存储服务器之间的距离,我们得到 r_{11} = 117.3km,r_{12} =

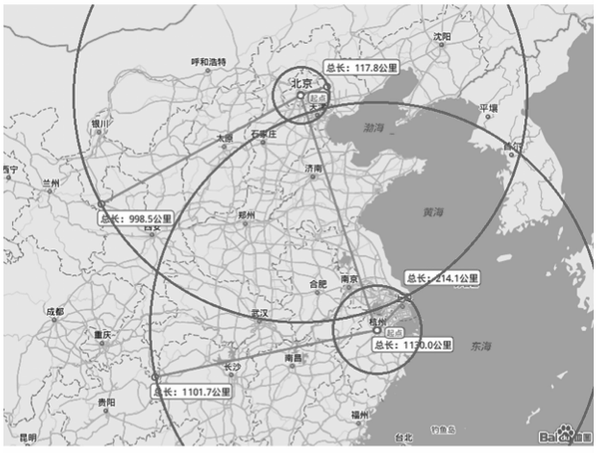

图8 基于测试时间的地理范围

998.5km，r_{21} = 1101.2km，r_{22} = 214km.

(2)分节数对方案性能的影响

随着分节数 s 的增大，在可恢复性证明中的性能变化. 如图 9 ~ 12 所示.

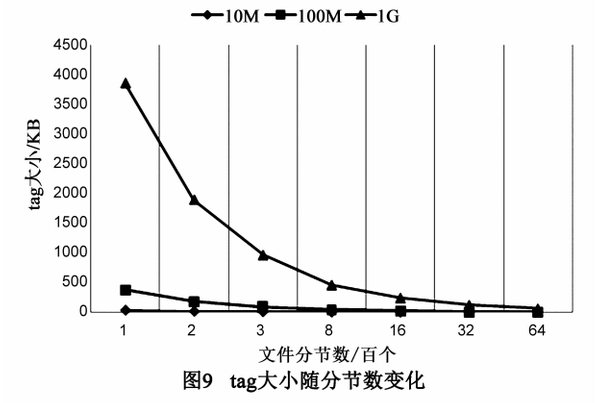

图9 tag大小随分节数变化

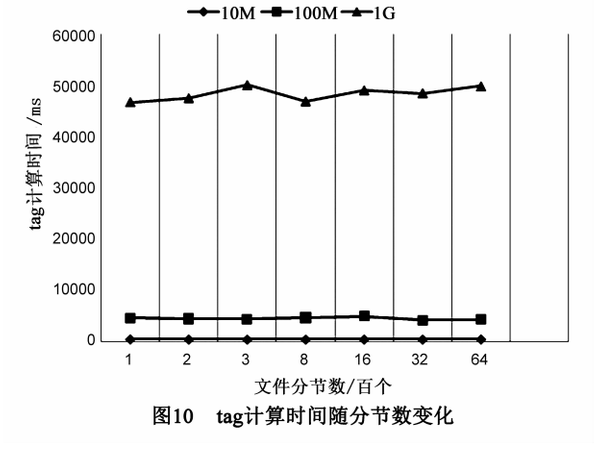

图10 tag计算时间随分节数变化

图 9 产生的原因是，随着分节数 s 的增多，分块数 n 是在不断减小的，每块文件生成的 tag 大小是固定的 λ 位，所以生成的 tag 总大小是会随着减小的.

图 10 产生的原因是，对于不同的分块和分节方案，都需要对每一节文件进行处理，而且在计算过程中是在 Z_p 域内进行的，不会因为参与计算的数据块数增多而是数据量变得很大从而减慢计算速度.

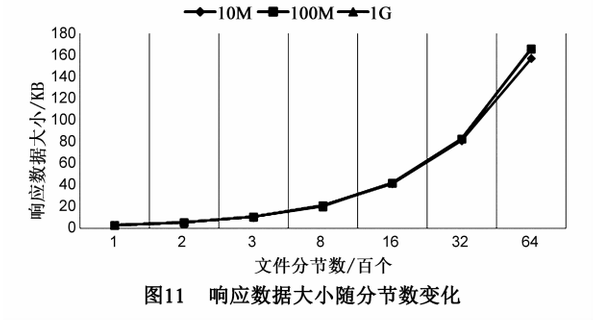

图11 响应数据大小随分节数变化

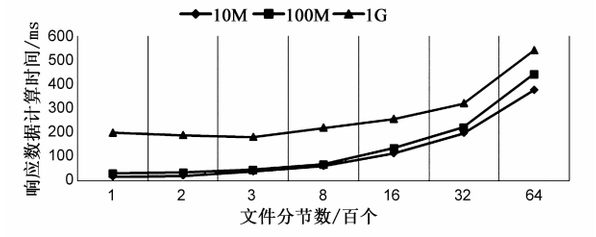

图12 响应数据计算时间随分节数变化

图 11 产生的响应数量是与文件的分节数相关联的，即响应块数量与分节数相同，而与文件大小及分块数量 n 无关.

图 12 原因是在进行计算前的准备阶段和结束阶段花费了一定的时间，而进行计算响应的时间基本上是成倍增长的.

对不同分节数情况下响应的传输时间进行测量. 结果如图 13 所示.

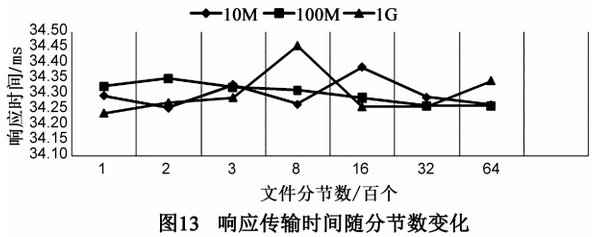

图13 响应传输时间随分节数变化

验证地理位置的响应使用的是响应结果的前 1024 字节. 在接收完验证地理位置的信息后再传输完整的验证结果并进行比对. 如图 14 所示.

由图 14 可以看出，对于不同的文件大小应该选择的文件分节数不完全相同.

(3)文件大小对方案性能的影响

随着文件大小的增加，tag 大小、计算 tag 时间等数据都是增加的. 响应大小基本保持不变原因是响应大小是与文件分节数相关的，在采用相同分节数的情况

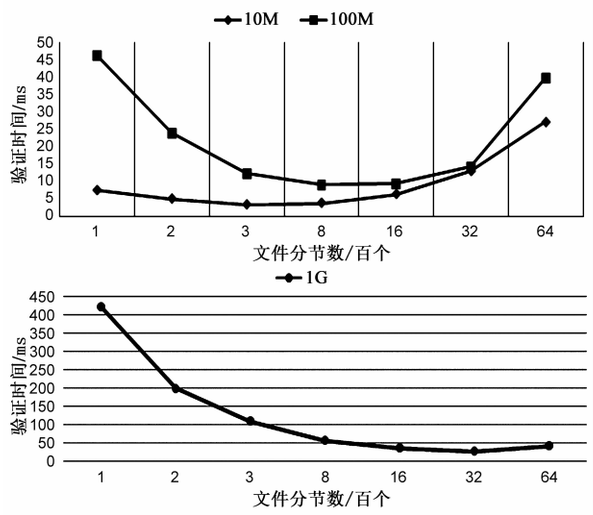

图14　验证时间随文件分节数变化

下,数据请求大小基本保持不变,如图 15 所示.

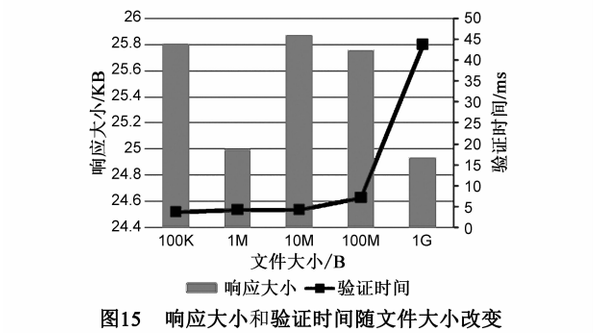

图15　响应大小和验证时间随文件大小改变

图 15 从侧面证明了需要针对不同的文件大小选择不同的分节数,否则在文件大小增大时可能会使验证时间急剧增大.

(4)数据请求数对方案性能的影响

具体结果如图 16 所示. 图 16 产生的原因是,数据请求数量增多,计算响应时参与计算的文件块数增多,因此对应的计算时间也会增加.

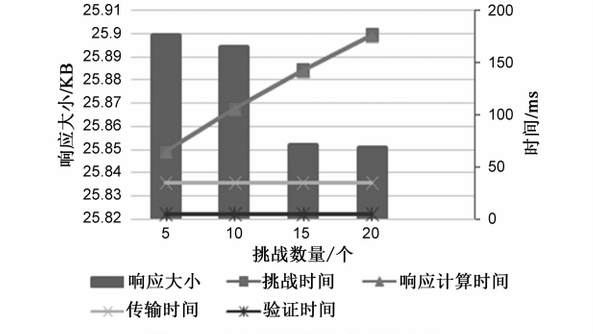

图16　响应随数据请求组数变化

6　结论

云计算越来越受到当今学术界和企业界的关注. 在发生大规模停电和天灾时要防止数据丢失,其关键是实现云端数据的异地备份容灾. 在现有的云存储环境下,还没有一种对云端数据的异地备份进行验证的可行方法. 针对该问题,本文提出了一种基于数据可恢复性验证的数据多地理位置验证方案. 该方案中,用户不需要下载数据,就可以对数据的多地理位置及数据完整性进行检查. 性能分析和测试证明该方案具有较低的计算、存储和传输负载. 下一步研究的重点是在进行申请第三方公开性验证时,如何确保数据的机密性,或者不需要引入可信第三方就可以实现公开性验证,以及如何对动态数据的持有性进行多副本的验证等[22~24].

参考文献

[1] Juels A, Kaliski Jr B S. PORs: Proofs of retrievability for large files[A]. Proceedings of the 14th ACM Conference on Computer and Communications Security[C]. USA: ACM, 2007. 584 – 597.

[2] Erway C, Küpçü A, Papamanthou C, et al. Dynamic provable data possession[A]. Proceedings of the 16th ACM Conference on Computer and Communications Security[C]. USA: ACM, 2009. 213 – 222.

[3] Bowers K D, van Dijk M, Juels A, et al. How to tell if your cloud files are vulnerable to drive crashes[A]. Proceedings of the 18th ACM Conference on Computer and Communications Security[C]. USA: ACM, 2011. 501 – 514.

[4] Benson K, Dowsley R, Shacham H. Do you know where your cloud files are? [A]. Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop[C]. USA: ACM, 2011. 73 – 82.

[5] Watson G J, Safavi-Naini R, Alimomeni M, et al. LoSt: location based storage[A]. Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop[C]. USA: ACM, 2012. 59 – 70.

[6] Gondree M, Peterson Z N J. Geolocation of data in the cloud[A]. Proceedings of the Third ACM Conference on Data and Application Security and Privacy[C]. USA: ACM, 2013. 25 – 36.

[7] Wang Z, Sun K, Jajodia S, et al. Disk storage isolation and verification in cloud[A]. IEEE Global Communications Conference (GLOBECOM)[C]. USA: IEEE, 2012. 771 – 776.

[8] Wang Z, Sun K, Jing J, et al. Verification of data redundancy in cloud storage[A]. Proceedings of the 2013 International Workshop on Security in Cloud Computing[C]. USA: ACM, 2013. 11 – 18.

[9] Wang Z, Sun K, Jajodia S, et al. Terracheck: Verification of dedicated cloud storage[A]. Data and Applications Security

and Privacy XXVII [M]. Berlin Heidelberg: Springer, 2013. 113 – 127.

[10] Noman A, Adams C. Providing a data location assurance service for cloud storage environments[J]. Journal of Mobile Multimedia, 2012, 8(4): 265 – 286.

[11] Albeshri A, Boyd C, Nieto J G. Geoproof: proofs of geographic location for cloud computing environment[A]. Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)[C]. USA: IEEE, 2012. 506 – 514.

[12] Albeshri A, Boyd C, Nieto J G. Enhanced GeoProof: improved geographic assurance for data in the cloud[J]. International Journal of Information Security, 2014, 13(2): 191 – 198.

[13] Chen B, Curtmola R. Towards self-repairing replication-based storage systems using untrusted clouds[A]. Proceedings of the Third ACM Conference on Data and Application Security and Privacy[C]. USA: ACM, 2013. 377 – 388.

[14] BOWERS K D, JUELS A, OPREA A. HAIL: Ahigt-aviailability and integrity layer for cloud storage[A]. Proceedings of the 16th ACM Conference on Computer and Communications Security[C]. USA: ACM, 2009. 67 – 76.

[15] 周恩光,李舟军,郭华,贾仰理. 一个改进的云存储数据完整性验证方案[J]. 电子学报, 2014, 42 (1): 150 – 154.
ZHOU En-guang, LI Zhou-jun, GUO Hua, JIA Yang-li. An improved data integrity verification scheme in cloud storage system[J]. Acta Electronica Sinica, 2014, 42 (1): 150 – 154. (in Chinese)

[16] 王丽娜,任正伟,余荣威,韩凤,董永峰. 一种适于云存储的数据确定性删除方法[J]. 电子学报, 2012, 40(2): 266 – 272.
WANG Li-na, REN Zheng-wei, YU Rong-wei, HAN Feng, DONG Yong-feng. A data assured deletion approach adapted for cloud storage[J]. Acta Electronica Sinica, 2012, 40(2): 266 – 272. (in Chinese)

[17] 沈钢,魏震,蔡云泽,许晓鸣,何星,张卫东. 一种实时以太网介质访问控制协议的时延性能分析[J]. 电子学报, 2003, 31(2): 175 – 179.
SHEN Gang, WEI Zhen, CAI Yun-ze, XU Xiao-ming, HE Xing, ZHANG Wei-dong. Delivery delay analysis of a real-time ethernet MAC protocol[J]. Acta Electronica Sinica, 2003, 31(2): 175 – 179. (in Chinese)

[18] Shacham H, Waters B. Compact proofs of retrievability [A]. Advances in Cryptology-ASIACRYPT[M]. Berlin Heidelberg: Springer, 2008. 90 – 107.

[19] Curtmola R, Khan O, Burns R, et al. MR-PDP: Multiple-replica provable data possession[A]. Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS08)[C]. USA: IEEE, 2008. 411 – 420.

[20] 李超零,陈越,谭鹏许,杨刚. 基于同态 hash 的数据多副本持有性证明方案[J]. 计算机应用研究, 2013, 30(1): 265 – 269.
Li Chao-ling, Chen Yue, Tan Peng-xu, Yang Gang. Multiple-replica provable data possession based on homomorphic hash[J]. Application Research of Computers, 2013, 30(1): 265 – 269. (in Chinese)

[21] Yang C, Ren J, Ma J. Provable ownership of file in de-duplication cloud storage[A]. Proceedings of Global Communications Conference (GLOBECOM)[C]. USA: IEEE, 2013. 695 – 700.

[22] Erway C, Kupcu A, Papamanthou C, et al. Dynamic provable data possession[A]. Proceedings of the 16th ACM Conference on Computer and Communications Security [C]. USA: ACM, 2009. 213 – 222.

[23] 李超零,陈越,谭鹏许,杨刚. 一种高效的动态数据持有性证明方案[J]. 小型微型计算机系统, 2013, 34(11): 2461 – 2466.
LI Chao-ling, CHEN Yue, TAN Peng-xu, LI Min, YANG Gang. An efficient provable data possession scheme with data dynamics[J]. Journal of Chinese Computer Systems, 2013, 34(11): 2461 – 2466. (in Chinese)

[24] 胡德敏,余星. 一种基于同态标签的动态云存储数据完整性验证方法[J]. 计算机应用研究, 2014, 31(5): 1362 – 1365, 1395.
HU De-min, YU Xing. Dynamic cloud storage data integrity verifying method based on homomorphic tags[J]. Application Research of Computers, 2014, 31(5): 1362 – 1365, 1395. (in Chinese)

作者简介

周洪丞 男, 1989 年生, 山东威海人. 西安电子科技大学硕士, 主要研究方向为云计算和存储安全.

杨 超 男, 1979 年生, 陕西西安人. 西安电子科技大学副教授, 主要研究方向为密码学与网络安全、云计算及移动智能计算安全.
E-mail: chaoyang@ mail. xidian. edu. cn