doi:10.11959/j.issn.1000-436x.2017106

云环境下基于安全网络编码的数据更新算法

吴昊[1]，赖成喆[1]，范九伦[1]，刘建华[2]
（1. 西安邮电大学通信与信息工程学院，陕西 西安 710121；2. 西安邮电大学信息中心，陕西 西安 710121）

摘 要：在云环境下进行数据存储时，利用安全网络编码技术可以很好地解决数据的隐私性和可靠性问题。但经过网络编码后的各个编码块通常具有很高的相关性，文件内容极少的变化都需要重新编码，极易造成数据泄露，同时严重消耗了系统资源。为此，提出一种网络编码云存储数据更新算法，存储节点只需要根据服务器发送的差值矩阵，更新部分编码块，就可以完成整个文件的更新。实验结果表明，所提算法和 RS 编码、Tornado 编码相比，在保证了数据安全的前提下大大提高了数据更新和数据重构的效率。

关键词：云存储；网络编码；数据更新；隐私保护；差值矩阵
中图分类号：TP393 **文献标识码**：A

Data update algorithm based on secure network coding in cloud environment

WU Hao[1], LAI Cheng-zhe[1], FAN Jiu-lun[1], LIU Jian-hua[2]
(1. School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China;
2. Information Centre, Xi'an University of Posts and Telecommunications, Xi'an 710121, China)

Abstract: In the cloud environment for data storage, the use of secure network coding technology can be a good solution to the data privacy and reliability issues. However, each coding block usually has a high correlation after network coding, very few updates to the file need to be re-encoded which is extremely easy to cause information leakage and serious consumption of system resources. To solve this problem, a network coding cloud storage data updating algorithm was proposed. Just by sending files change difference matrix, the storage node could update parts of the coding block accordingly which could complete the entire update files. Experimental results show that compared with RS coding and Tornado coding, the algorithm can not only ensure data security, but also greatly improve the efficiency of data update and data reconstruction.

Key words: cloud storage, network coding, data update, privacy protection, difference matrix

1 引言

随着互联网、移动互联网、物联网的发展和各种大数据应用规模的不断扩大，数据呈爆炸性增长的趋势并具有海量存储的特点。海量的信息对数据存储提出了巨大的挑战，传统的数据存储系统遇到了瓶颈，无法及时完成各项运作任务。而云存储系统可以使用户以较低廉的价格获取海量的存储能力，因此越来越受到用户的青睐。但云存储系统高度集中的计算资源又面临着严重安全挑战。用户将私有数据上传到云端的同时，也丧失了对数据的绝对控制权。调查显示，出于安全方面的考虑，多达70%的用户仍然不愿意将关键数据置于自身控制域之外。近年来，谷歌、百度网盘、360 云盘等国内外多家云服务商都曾出现过各种安全问题，并导致了严重的后果，导致很多厂商被迫关闭了云盘服务。安全技术的缺失已经成为云存储普及最重要的障碍。因此，云存储用户数据的隐私保护成为了亟

收稿日期：2016-12-16；**修回日期**：2017-04-06
基金项目：国家自然科学基金资助项目（No.61502386）；陕西省国际科技合作与交流计划基金资助项目（No.2015KW-010）
Foundation Items: The National Natural Science Foundation of China (No.61502386), The International Science and Technology Cooperation and Exchange Plan in Shaanxi Province (No.2015KW-010)

待解决的问题[1~4]。

网络编码的安全性优势为云存储应用的安全提供了一个很好的解决方案[5]。使用网络编码对原始信息进行编码操作，可以保证存储—转发过程中数据的隐私性和安全性，同时可以实现多播网络的最大吞吐量。文献[6]中使用基于纠删码的容错技术，论证了该方案在存储容量和存储代价方面比基于副本的容错技术更加具备优势。文献[7~9]提出了再生码（RC，regenerating code），使用该编码进行数据修复时所需的网络带宽与纠删码相比大大降低。文献[10]中提出的 RS（Reed-Solomon）再生纠删码继承了 RS 编码容多错的可靠性，又能实现容三错的高效性。文献[11]提出了一种安全编码——SRCS 编码，以保证在云存储这种高度开放的环境下,存储系统容错过程中数据的安全性。文献[12]提出了基于零空间的网络编码云存储数据完整性校验方案，通过计算原始信息的零空间生成验证向量，并将验证向量发送给独立的第三方验证节点完成数据验证。文献[13]将门限公钥加密技术与指数纠删码相融合，构造了一种同时具有机密性与容错性的安全云存储模型。这些方案多集中于解决数据修复问题,在一定程度上保护了云存储数据的隐私性、有效性、可靠性和完整性，但都忽略了数据更新的问题。经过网络编码后各个编码块具有很高的相关性，往往很小的内容变动必须对所有的编码块都进行更新，显然这是一个既费时又费力的过程。文献[14]对一些典型纠删码的更新效率进行了对比，其中，RS 码的更新效率很低，其他（如 EVENODD、RDP(row-diagonal parity)等）编码的更新效率也随着磁盘数目的增加而降低。文献[15]证明了 MDS（maximum distance separable）编码（达到最优存储利用率的编码）的更新效率不可能达到数学上的最优，更新效率和 MDS 特性不可兼得。同时，当用户在云盘上频繁进行数据更新时，黑客很容易窃取到用户的历史访问数据，进而对云服务实现“重放攻击”，极易造成用户隐私数据的泄露。因此，伴随着大数据管理和云存储服务以惊人的速度增长，存储节点上的数据更新非常频繁，如何安全、高效地进行文件内容的更新也是十分重要的问题。为了解决这个问题，提出了一种基于差值矩阵的网络编码云存储数据更新算法，通过该算法，存储节点是不需要不停地同步到最新的版本，而是仅通过发送文件的变化，选择性地进行编码块的更新，既实现了对数据更新时的用户隐私保护，又大大减少了系统进行更新的通信和计算成本。

2 数据编码

2.1 基于网络编码的云存储的优缺点

为了解决复制存储开销大的缺点,研究者尝试将网络编码引入云存储应用中。这些网络编码最常见的就是各类纠删码，包括 RS 码、喷泉码和 RC 码等。这些方法的一般过程如下。将每个文件分为 k 个原始块，同时提供 R 个文件的冗余副本，此会有 $n = Rk$ 个编码块分配在不同的存储节点中。当需要获取文件时，只要从存储节点随意下载任何 k 个不同的编码块即可，即符合 (n,k) 特性。当某些存储节点出现故障时，系统可以通过其他存储节点上的存储块修复被破坏的存储节点上的存储块。由于对数据采用了编码操作，可以用更少的数据存储量来达到与复制存储相同的可靠性，同时任意一个未解码的编码分组都不会泄露原始数据的信息，从而保护了数据的隐私性。因此在过去的云存储研究中，编码算法和数据保护算法得到了更多的关注。

在云存储系统中，数据是不断被更新的，系统需要连续不断地进行 I/O，因此，系统更新效率也是评价网络编码云存储系统性能最重要的标志之一。而在云存储系统中对数据采用编码操作后，数据更新效率低的缺点暴露无遗。由于编码块的相关性，某个文件需要更新时，通常整个系统的众多编码块都会受到影响，需要重新编码。另外，不同存储节点上的存储块可能因为不同步，造成版本不同，更新操作往往需要涉及众多存储节点，增加了系统的更新开销。因此，如何提高云存储系统的更新效率也成为云存储系统研究者关注的问题之一。

2.2 编码原理

为了提高网络编码云存储系统中的更新效率，本文提出了一种基于差值矩阵的网络编码云存储数据更新算法。该算法旨在进行文件更新时，寻找到尽可能少的需要更新的编码块，以降低系统的通信和计算成本。编码方法的伪代码如下。

假设每个文件在系统中的副本个数为 R，R 为大于或等于 1 的整数。

Begin

1) 输入原始文件 $\boldsymbol{F}$、k 个随机密钥 $[\varepsilon_1,\varepsilon_2,\cdots\varepsilon_k]$；

2) $\boldsymbol{F}$ 被分成 k 块，$\boldsymbol{F}=[f_1,f_2,\cdots,f_k]^{\mathrm{T}}$；

3) 编码系数矩阵 $\boldsymbol{P}=\begin{bmatrix}\varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_k\\ \varepsilon_1^2 & \varepsilon_2^2 & \cdots & \varepsilon_k^2\\ \vdots & \vdots & \ddots & \vdots\\ \varepsilon_1^n & \varepsilon_2^n & \cdots & \varepsilon_k^n\end{bmatrix}$；

4) 编码 $\boldsymbol{C}=\boldsymbol{PF}$；

5) 将编码块和相应的编码系数平均分配在系统中存储节点中；

End

编码操作如图 1 所示。经过这样的网络编码，每个存储节点上存储的编码块本身都是原始块的线性组合。因此只要获得任何 k 个线性无关的编码块 $\boldsymbol{C}=[c_1,c_2,\cdots,c_k]$，就可以使用式(1)恢复原始文件 $\boldsymbol{F}$。也就是说，该编码方法和 RS 编码一样，同样符合 (n,k) 特性。

$$\boldsymbol{F}=\boldsymbol{P}^{-1}\boldsymbol{C}^{\mathrm{T}} \tag{1}$$

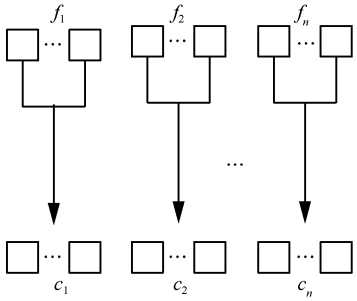

图 1　编码操作

另外，每个存储节点可以从服务器或临近的存储节点直接下载编码块。当需要向临近节点提供一个编码块时，可将本身存储的编码块的线性组合作为新的编码块。也就是说，如果有存储节点发生故障，可以通过完好节点的编码块生成新的编码块，它们仍然是原始块的线性组合。

3　数据更新

3.1　数据更新原理

当文件更新时，用户程序需要将更新的部分或整个全新文件传送到服务器，服务器使用类似版本控制系统的方式管理存储文件的更新，每次更新指定一个版本号 v 即可，具体方法如下。

假设有某个文件 $\boldsymbol{F}$，版本号为 v，它所有原始数据块为 $\boldsymbol{F}^v=[f_1^v,f_2^v,\cdots,f_k^v]$，版本号为 $v+m$ 的同一文件的原始数据块为 $\boldsymbol{F}^{v+m}=[f_1^{v+m},f_2^{v+m},\cdots,f_k^{v+m}]$，$\delta_i^{v+m}$ 表示第 i 个数据块的在 2 个版本之间的差值，即

$$\delta_{i,j}^{v,v+m}=b_{i,j}^{v+m}-b_{i,j}^{v},1\leqslant j\leqslant s \tag{2}$$

可以得到

$$f_i^{v+m}=f_i^v+\delta_i^{v+m} \tag{3}$$

取 $\boldsymbol{\Delta}^{v,v+m}=[\delta_1^{v,v+m},\delta_2^{v,v+m},\cdots,\delta_k^{v,v+m}]^{\mathrm{T}}$ 作为差值矩阵，由于每个存储节点上存储的编码块本身都是原始块的线性组合，可得

$$c_{i,j}^{v+m}=c_{i,j}^{v}+\sum\varepsilon_{i,j}\delta_{i,j}^{v,v+m} \tag{4}$$

因此，只需要将差值矩阵 $\boldsymbol{\Delta}^{v,v+m}$ 传输到每个存储节点，就可以根据式(4)完成存储编码块的更新，而无需传输完整的更新文件，这样可以大大减少文件更新时系统的通信开销。

3.2　编码块的不同步问题的解决

当需要进行文件更新时，利用差值矩阵的传输大大减少了系统的通信成本。但实际应用过程中，由于传输错误或某些存储节点的自身问题未能及时更新，存储节点之间不能达到完美的同步，就可能造成不同存储节点存储的文件版本各不相同的情况。如果仍然采用上面的方法，每 2 个存储节点之间差值矩阵可能都不尽相同，要将这些差值矩阵都进行存储，必然会给系统的存储和数据传输带来巨大的压力。为了减少系统功耗，需要进一步减少差值矩阵的传输，采用以下方法。每个存储节点除了存储最近版本（即最近更新版本）的编码块之外，还需存储一个初始版本的编码块作为基准编码块，假设该版本号为 1，系统每次传输的都是新版本与初始版本的差值矩阵为 $\boldsymbol{\Delta}^{1,v+m}$，这时，无论每个存储节点是否同步，都可以根据式(5)完成更新。

$$c_{i,j}^{v+m}=c_{i,j}^{1}+\sum\varepsilon_{i,j}\delta_{i,j}^{1,v+m} \tag{5}$$

同样地，每个存储节点只需要存储当前版本与初始版本之间的差值矩阵 $\boldsymbol{\Delta}^{1,v}$，而不需要存储完整的初始版本编码块。这时每个节点的编码块的更新需要以下 3 个步骤。

1) 根据存储节点上存储的版本为 v 的编码块和差值矩阵 $\boldsymbol{\Delta}^{1,v}$，根据式(6)计算出初始版本编码块。

$$c_{i,j}^{1}=c_{i,j}^{v}-\sum\varepsilon_{i,j}\delta_{i,j}^{1,v} \tag{6}$$

2) 根据式(5)计算更新版本 $v+m$ 的编码。

3) 用 $c_{i,j}^{v+m}$ 和 $\boldsymbol{\Delta}^{1,v+m}$ 分别代替 $c_{i,j}^{v}$ 和 $\boldsymbol{\Delta}^{1,v}$，即可完成编码块的更新。

3.3　差值矩阵的压缩

通常情况下，对文件的更新只是小范围的变

动，大部分文件块和编码块没有变动，因此差值矩阵 $\boldsymbol{\Delta}^{1,v+m}$ 中的元素 $\delta_{i,j}^{1,v+m}$ 大部分取值为 0，完整存储这个差值矩阵显然会造成存储空间的浪费，因此，可以对差值矩阵进行压缩，压缩方法如下。用一个向量 $\boldsymbol{U}$ 代替 $\boldsymbol{\Delta}^{1,v+m}$ 的传输，$\boldsymbol{U}$ 中的每一个元素都是 $\delta_{i,j}^{1,v+m}\neq 0$ 时的序列 (i,j)，表示第 i 个文件块的第 j 个字节有更新。这时的更新只需读取向量 $\boldsymbol{U}$ 所有取值，只有在这个向量表中存在的编码块才进行如 3.3 节所述的更新，其余编码块保持不变，算法如式(7)所示。这样，每次更新只需对部分编码块进行更新，大大减少了系统的负担。同样地，可以对存储节点上存储的 $\boldsymbol{\Delta}^{1,v}$ 也用一个向量 $\boldsymbol{V}$ 代替，$\boldsymbol{V}$ 中的每一个元素都是 $\delta_{i,j}^{1,v}\neq 0$ 时的序列 (i,j)，这样就可以进一步减少系统的存储开销。

$$c_{i,j}^{v+m}=\begin{cases}c_{i,j}^{v},(i,j)\notin \boldsymbol{U}\\ c_{i,j}^{v}-\sum\varepsilon_{i,j}\delta_{i,j}^{1,v}+\sum\varepsilon_{i,j}\delta_{i,j}^{1,v+m},(i,j)\in \boldsymbol{U}\end{cases} \tag{7}$$

经过这样的压缩，不仅减少了传输差值矩阵 $\boldsymbol{\Delta}^{1,v+m}$ 的通信开销，同时减少了每个存储节点上存储差值矩阵 $\boldsymbol{\Delta}^{1,v}$ 所需的存储开销，大大节省了系统的资源。

3.4 文件更新流程

综合以上算法，系统文件更新的伪代码如下。

Begin

1) 输入更新文件 $\boldsymbol{F}^{v+m}$,版本号为 $v+m$；

2) if $v=1$

3) $V=0$；

4) else

5) $\boldsymbol{\Delta}^{1,v}=V$，$\boldsymbol{\Delta}^{1,v+m}=\boldsymbol{F}^{v+m}-\boldsymbol{F}^{1}$；

6) $\boldsymbol{U}$ =compress($\boldsymbol{\Delta}^{1,v+m}$);

7) 向每个存储节点发送更新内容向量 $\boldsymbol{U}$；

8) if $(i,j)\in \boldsymbol{U}$

9) Begin

10) if $(i,j)\in \boldsymbol{V}$

11) $c_{i,j}^{1}=c_{i,j}^{v}-\sum\varepsilon_{i,j}\delta_{i,j}^{1,v}$；

12) if$(i,j)\in \boldsymbol{U}$

13) $c_{i,j}^{v+m}=c_{i,j}^{1}+\sum\varepsilon_{i,j}\delta_{i,j}^{1,v+m}$；

14) $c_{i,j}^{v}=c_{i,j}^{v+m}$；

15) $\boldsymbol{V}=\boldsymbol{U}$；

16) End

End

4 算法性能分析

通常评价一种网络编码算法的性能指标有以下几个。

1) 存储利用率：存储的数据量与所有的信息量之比，即系统的冗余度。

2) 计算效率：即算法的计算速度，包括编解码速度、数据更新速度和数据重构速度 3 个方面。

在云存储系统中，由于存储介质的数目众多，存储利用率不是关注的焦点。因此，本文对算法性能评价重点聚焦在容错能力和计算效率上。

4.1 实验测试环境

实验采用开源的 Hadoop 分布式文件系统作为测试平台，实现了对输入和输出文件的分块、编解码处理、存储管理和差错控制等功能。Hadoop 系统负责按需求将文件划分成多个数据块，并将它们存储在不同的存储节点中，存储节点根据客户端的位置排列成一个有序的节点云。存储节点云由若干台主机组成，一台服务器作为存储服务的管理服务器，用户通过客户端对系统进行访问和操作，网络拓扑结构如图 2 所示。云存储管理服务器的 CPU 为 Xeon X3430 2.4 GHz、主频 2.4 GHz、内存 8 GB、硬盘 260 GB，存储节点主机的 CPU 为 Core 2 DuoT7200、主频 2.0 GHz、内存 4 GB、硬盘 500 GB。实验分别从容错能力、编解码速度、数据更新速度和数据重构速度 4 个方面将本文算法与其他 2 种编码算法（RS 编码算法和 Tornado 编码算法）进行了比较和分析。

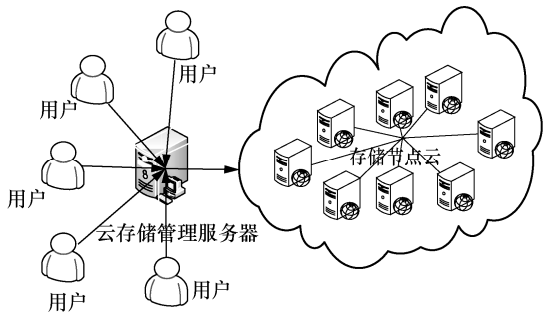

图 2 实验环境网络拓扑结构

4.2 编解码速度

实验选择了 5 个文件，大小分别为 1 MB、5 MB、10 MB、50 MB、100 MB，每个文件都分成 100 块，副本数均为 3，因此，每个文件的数据块数都为 300。实验记录了每个文件编码和解码所花费的时

间，测试结果如图 3 所示。可以看到，随着数据文件的增大，RS 码的编码速率迅速下降。Tornado 码的编解码过程中只需进行简单的二进制异或运算，能够以线性时间进行编码和解码，大幅提升了编解码的速度。本文算法的编解码也是对原始数据块进行线性运算，因此编解码时间随数据大小线性增长，速度接近 Tornado 码，对处理大数据文件有很大优势。

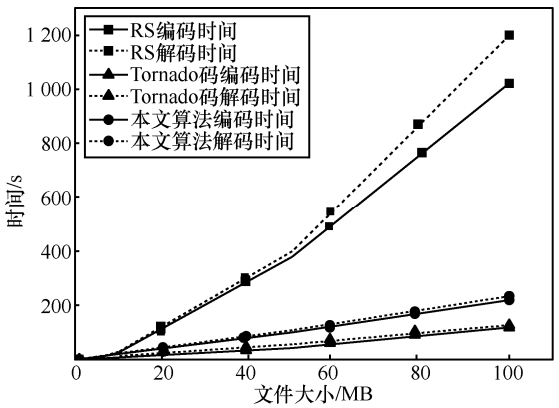

图 3　编解码速度实验

4.3　数据更新速度

选择一个大小为 10 MB 的文件作为初始文件，文件分块方式与编解码实验相同。不断改变更新的文件块数，对 3 种算法的更新时间进行了比较，如图 4 所示。比较结果显示，RS 编码算法和 Tornado 编码算法在进行数据更新时，不管更新的原始数据块有多少，几乎所有的编码块都需要重新编码，文件更新相当于重新编码，更新效率较低。而本文算法只需根据更新向量对部分编码块进行更新，更新算法也是线性运算，故更新速度也与需要更新的原始数据块数目成正比。虽然本文算法进行文件更新时增加了差值向量计算和传输，但当原始数据文件较大、编码块较多，而需要更新的编码块较少时，只要网络带宽足够，计算和传输差值向量的时间远远小于编码的时间，对系统整体更新效率影响不大。在进行文件更新时，每个存储节点可以根据更新向量同时进行更新，实现了存储节点上编码块的分布式更新，相对于另外 2 种算法的集中式更新，本文算法可以有效分担管理服务器的压力，提高工作效率。由此可见，在进行大数据文件更新时，使用本文算法与可以大大减少数据更新消耗的时间。

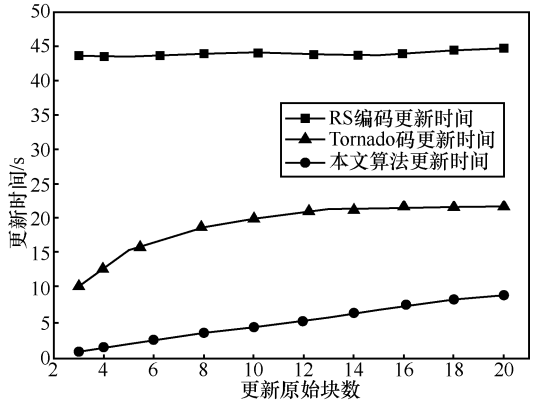

图 4　数据更新速度实验

4.4　数据重构速度

仍然选择大小为 10 MB 的文件作为初始文件，文件分块方式也与上面的实验相同，将所有编码块平均分配在 30 个不同的存储节点上。不断增加故障节点的数目，将 3 种算法的数据重构速度进行比较，结果如图 5 所示。由于 RS 编码的重构过程首先需要对矩阵求逆，然后才能通过矩阵乘法运算恢复出错的元素，而矩阵的求逆运算和乘法运算都非常复杂，因此 RS 编码的重构效率非常低。相比较而言，Tornado 编码只需进行线性运算即可对恢复出错的节点，数据重构速度大大提高，而本文算法中如果有存储节点发生故障，只需将完好节点的编码块经过线性运算，即可生成新的编码块，它们仍然是原始块的线性组合，数据重构的速度进一步提高。

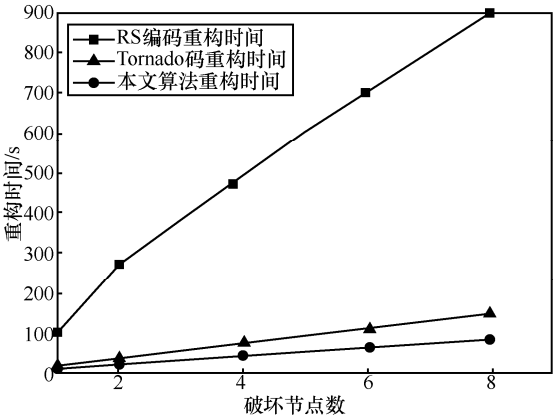

图 5　数据重构速度实验

5　算法安全性和通信开销分析

5.1　算法安全性分析

算法安全性分析如表 1 所示。

表 1 算法安全分析

编码算法	解码所需文件块数	数据更新时传输的内容
RS 编码	k	所有新的编码块
Tornado 码	$k(1+\varepsilon)$	所有新的编码块
本文算法	k	差值向量 $\boldsymbol{U}$

在对编码算法介绍中已经证明，本文编码方法和 RS 编码一样符合 (n,k) 特性，也就是说，二者的容错能力相同。因此，若攻击者无法获得 k 个线性无关的数据分块，就不能获取完整的文件。而 Tornado 码是基于纠删码的前向纠错码，必须要获得 $k(1+\varepsilon)$ 个节点才能完成解码。另外，Tornado 编码前的 m 个原始分块也保留在编码后的分块中，这就造成当攻击者攻陷的节点数目即使没有超过 $n-r$，也能够获得关于原始文件的部分信息。因此，从安全性方面看，本文编码方法和 RS 编码较 Tornado 码更适合对安全要求较高的云存储环境。

本文算法将用户更新的文件直接转换成差值向量发送给云管理服务器，可以保证在不泄露完整更新文件的情况下完成文件更新，从而保证了数据更新的隐私性。而另外 2 种方法需要对更新文件重新编码并在系统中传输，大大增加了数据块被泄露的可能性。

5.2 通信开销分析

本文方案的云管理服务器只需根据用户新旧版本文件的变化计算差值向量，通过对原来的编码块与差值向量线性组合就可以得到新的编码块，不需要用户重新生成，大大减少更新过程的通信开销。对于每次的数据更新过程，其主要的通信开销是计算差值向量的开销和从管理服务器将差值向量发送给存储节点的开销，这个开销是由文件内容更新的比例决定的。

图 6 给出了不同更新比例下的通信开销。在实际测试中发现，差值向量的计算过程随文件更新比例基本是线性增长，而差值向量传输过程中的通信数据较小，消耗的通信时间变化不大。在实际测试过程中，当更新比例小于 20%时，通信开销不超过 5 000 ms，这相对于大数据文件的编码和存储过程所消耗的时间是可以忽略不计的。

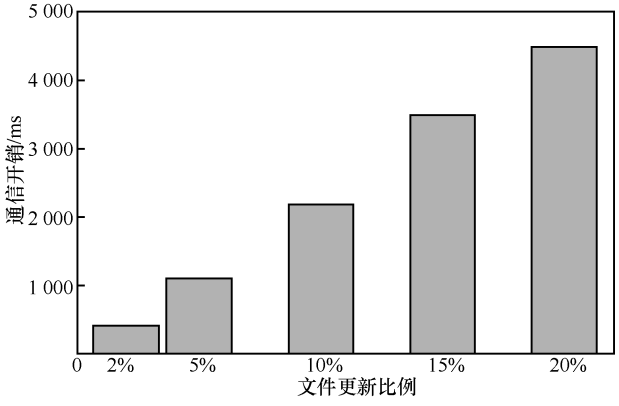

图 6 不同更新比例下的通信开销

6 结束语

基于网络编码的云存储方案大多着重于解决云存储应用中的数据安全问题，而忽略了网络编码对大数据文件更新所带来的难题。本文设计了一种基于差值矩阵的网络编码云存储数据更新算法，通过该算法进行文件更新时，存储节点不需要不停地同步到最新的版本，而是通过传输文件中变更的部分，存储节点根据需求更新部分编码块实现文件的整体更新。实验表明，该方案有效地解决了云存储系统文件的更新问题，同时提升了编解码效率和数据重构效率，存储消耗少，是一种行之有效的网络编码云存储方案。

参考文献：

[1] 谭霜, 贾焰, 韩伟红. 云存储中的数据完整性证明研究及进展[J]. 计算机学报, 2015, 38(1): 164-176.
TAN S, JIA Y, HAN W H. Research and development of provable data integrity in cloud storage[J]. Chinese Journal of Computers, 2015, 38(1): 164-176.

[2] 傅颖勋, 罗圣美, 舒继武. 安全云存储系统与关键技术综述[J]. 计算机研究与发展, 2013, 50(1):136-145.
FU Y X, LUO S M, SHU J W. Survey of secure cloud storage system and key technologies[J]. Journal of Computer Research & Development, 2013, 50(1):136-145.

[3] 李晖, 孙文海, 李凤华, 等. 公共云存储服务数据安全及隐私保护技术综述[J]. 计算机研究与发展, 2014, 51(7):1397-1409.
LI H, SUN W H, LI F H, et al. Secure and privacy-preserving data storage service in public cloud[J]. Journal of Computer Research & Development, 2014, 51(7):1397-1409.

[4] 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 .

[5] DIMAKIS A G, BRIGHTEN G P, WU Y, et al. Network coding for distributed storage systems[J]. IEEE Transactions on Information Theory, 2007, 56(9):4539 - 4551.

[6] LIN H, TZENG W. A secure decentralized erasure code for distributed networked storage[J]. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(11): 1586-1594.

[7] WU Y N. Existence and construction of capacity-achieving network codes for distributed storage[J]. IEEE Journal on Selected Areas in Communications, 2010, 28 (2): 277-288.

[8] DIMAKIS A G, ALEXANDROS G. A survey on network codes for distributed storage[J]. Computing Research Repository, 2011, 99 (3): 476-489.

[9] KUBIATOWICZ H W A J D. Erasure coding vs. replication: a quantitative comparison[J]. Lecture Notes in Computer Science, 2002, 2429: 328-337.

[10] 鄢喜爱, 张大方, 杨金民, 等. 面向云存储容错系统的 RS 再生码[J]. 通信学报, 2016, 37(10):65-74.
YAN X A, ZHANG D F, YANG J M, et al. RS regenerating codes for cloud storage fault-tolerant system[J]. Journal on Communications, 2016, 37(10):65-74.

[11] 谭鹏许, 陈越, 兰巨龙, 等. 用于云存储的安全容错编码[J]. 通信学报, 2014, 35(3):109-115.
TAN P X, CHEN Y, LAN J L, et al. Secure fault-tolerant code for cloud storage[J]. Journal on Communications, 2014, 35(3):109-115.

[12] 王伟平, 张俊峰, 王建新. 基于零空间的网络编码云存储完整性校验方案[J]. 清华大学学报(自然科学版), 2016(1):83-88.
WANG W P, ZHANG J F, WANG J X. Data integrity check based on null space for network coding based cloud storage[J]. Journal of Tsinghua University(Science and Technology), 2016(1): 83-88.

[13] 徐剑, 李坚, 韩健, 等. 融合门限公钥加密和纠删码的安全云存储模型[J]. 软件学报, 2016, 27(6):1463-1474.
XU J, LI J, HAN J, et al. Secure cloud storage model based on threshold public key encryption and erasure codes over exponents[J]. Journal of Software, 2016, 27(6): 1463-1474.

[14] 罗象宏,舒继武. 存储系统中的纠删码研究综述[J]. 计算机研究与发展, 2012, 49(1):1-11.
LUO X H, SHU J W. Summary of research for erasure code in storage system[J]. Journal of Computer Research & Development, 2012, 49(1): 1-11.

[15] PLANK J S. The RAID-6 liberation codes[C]//The 6th USENIX Conference on File and Storage Technologies(FAST'08). 2008:97-110.

作者简介：

吴昊（1981-），男，江苏武进人，西安邮电大学讲师，主要研究方向为信息安全。

赖成喆（1985-），男，陕西汉中人，博士，西安邮电大学副教授，主要研究方向为信息安全。

范九伦（1964-），男，河南温县人，博士，西安邮电大学教授，主要研究方向为信号处理和信息安全。

刘建华（1963-），男，陕西宝鸡人，西安邮电大学高级工程师，主要研究方向为信息安全。