doi:10.11959/j.issn.1000-436x.2017045

网络编码云存储系统差分数据更新方案

王龙江，陈越，严新成，黄恺翔
（解放军信息工程大学，河南 郑州 450001）

摘 要：针对基于网络编码云存储系统数据更新时通信开销过大的问题，提出一种差分数据更新方案，通过对文件中更新部分进行编码和压缩传输，降低了更新过程中的通信开销。设计实现了基于网络编码的云存储原型系统，在真实网络环境中进行部署和测试，实验结果表明，与现有更新方案相比，所提方案通信开销更小，具有更好的可扩展性。

关键词：网络编码；云存储；差分数据更新
中图分类号：TP391 **文献标识码**：A

Differential data update scheme on network-coding-based cloud storage system

WANG Long-jiang, CHEN Yue, YAN Xin-cheng, HUANG Kai-xiang
(PLA Information Engineering University, Zhengzhou 450001, China)

Abstract: In order to solve the problem that the communication overhead of date update was too large on network-coding-based cloud storage system, a new differential data update scheme was proposed. By encoding and compressing the updated part of file, the communication overhead was reduced significantly. A network-coding-based storage prototype system was designed and implemented, and update scheme was deployed in the real network settings. Experimental results show that the proposed scheme has less communication overhead and better scalability than the existing schemes.

Key words: network coding, cloud storage, differential data update

1 引言

随着大数据[1]时代的到来，海量数据的可靠存储成为一个亟待解决的重要问题，引起了学术界和产业界的广泛关注和研究。云存储技术[2]的兴起和发展，为大数据的存储和处理提供了一个良好的解决方案，越来越多的用户选择将自己的数据托管在云存储平台（如 Amazon S3[3]、Windows Azure[4]）上。现有的云存储平台普遍使用编码方式来组织和存储数据，通过增加数据冗余来增强系统的容错能力，保障数据安全性和服务可靠性。

基于网络编码的存储方案[5]将用户文件切分成块，经过编码分散存储在网络中不同的存储节点上，当其中部分节点失效时，可以通过存活节点进行快速修复，从而保证了数据的安全性和服务的连续性。目前的网络编码方案多用于存储静态数据，以文件的长期归档为其应用场景，数据的更新频次较低。这类编码方案优先考虑数据的冗余性和可靠性，相关研究多集中于如何减少存储开销和修复带宽，并不关心数据的更新效率。这类方案中，源文件的细小变化都将影响到大量与之相关的编码块，当用户需要更新数据时，需要对数据进行重新编码和上传，计算和通信开销太大，难以适应数据规模较大和数据更新密集型的应用场景，一定程度上限

收稿日期：2016-07-04；**修回日期**：2016-10-25
基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（No.2012CB315901）；河南省科技攻关计划基金资助项目(No.17210221001)
Foundation Items: The National Basic Research Program of China (973 Program) (No.2012CB315901), Key Technologies R & D Program of Henan Province (No.17210221001)

制了网络编码技术在云存储领域的应用和发展。目前的数据中心网络中，由于节点间频繁的数据交互，网络带宽已成为最紧俏的系统资源，传统的数据更新方式会占用大量的带宽资源，影响网络性能和服务质量。

先前关于云存储更新问题的研究多停留在理论层面，采用理论分析和模拟仿真的手段，本文第一次在真实的网络环境中，在分布式系统下通过实验手段对基于网络编码的云存储数据更新问题进行了探索和研究，并对现有的更新方案进行了验证和分析；提出了客户端编码压缩，存储端解压更新的数据更新机制，极大地减少了更新过程的带宽开销；针对差分数据的特点，设计了一种游程编码和霍夫曼编码相结合的压缩方法，具有较好的压缩效果；设计实现了基于代理的云存储原型系统，采用低耦合的分层结构，存储节点无需编码和译码能力，支持多样化的存储介质，具有良好的扩展性；在上述原型系统上实现和部署了本文方案、DeltaNC[5]这 2 种差分更新方案和 F-MSR[6]再生码方案，并在实际网络环境中进行了性能测试和对比分析，实验结果表明，本文方案比现有方案能够显著减少更新过程的带宽开销，一定程度解决了网络编码云存储系统数据更新困难的问题。

2　相关工作

在现有的云存储系统中，常见的编码方式有纠删码和网络编码 2 类。纠删码技术[7,8]通过增加数据冗余，提供一定程度的容错能力，在云存储领域得到了广泛的研究和应用。目前，大多数商用的云存储系统，如微软的 Azure[4]、Facebook 的数据仓库[9]均采用纠删码技术来组织和存储数据。RS 码[10]是一种典型的纠删码，于 20 世纪 50 年代被提出，起初被应用于通信领域，旨在提高通信质量，后来被逐步应用到存储系统中，成为磁盘阵列技术[11](如 RAID5、RAID6)的理论基础。RS 码具有良好的 MDS 性质[12]，即将一个文件切分成 k 个相同大小的数据块，编码成 n（$n>k$）个编码数据块，分散存储在网络中的数据节点上，任选其中 k 个数据块，即可正确译码，还原出原始文件。Cauchy RS 码[13]是一种改进的 RS 码，编码速度更快；RAID5 能够容忍一个节点的失效，RAID6 能容忍 2 个节点的失效，EVENODD 码[14]和 RDP 码[15]是 RAID6 的 2 个特例。

基于网络编码的存储方案比传统的多副本存储策略（如 GFS[16]、HDFS[17]）能够显著地降低存储开销，比纠删码方案节省修复带宽，因此得到了持续的关注和深入的研究，积累了许多优秀的研究成果。

网络编码最初应用于通信领域，用来提高网络的吞吐量和利用率，后被逐渐应用到存储系统中，旨在增强存储系统的容错能力，提高数据安全性和系统可靠性。Alshwede 等[18]首先提出网络编码理论，在网络中间节点上，对信息进行适当的编码处理再向下游转发，可获得更大的网络吞吐量，提高网络的利用率，改善传输性能，在多播通信和无线传感器网络中得到了广泛的研究和应用。文献[19]最早提出了一种基于网络编码的文件分发系统，通过网络编码来提高文件分发的效率；文献[20]首次将网络编码应用到分布式存储系统中，减少了存储节点修复过程的带宽消耗；文献[21]在文献[20]的基础上，进行了深入的理论分析，提出了再生码的思想，且证明了再生码能够达到最小的修复带宽，并给出最小带宽再生码（MBR）和最小存储再生码（MSR）的概念。

以上关于网络编码的研究多停留在理论层面，研究者多采用理论建模和仿真手段对编码方案进行分析和验证。Hu 等[22]提出了第一个基于网络编码的文件系统 NCFS，实现了 E-MBR 再生码方案[23,24]，并对纠删码和 E-MBR 的性能进行验证和分析；文献[25]设计开发了一个基于代理的多 CSP（云服务提供商）网络编码云存储系统，实现了支持功能性修复的最小存储再生码 F-MSR[5]，为网络编码的研究提供了实验平台。

在网络编码存储系统中，文件的存储、获取、修复问题得到了持续关注和大量研究，但针对文件更新问题的研究却相对较少。源数据和编码数据之间的强关联性导致了编码数据更新异常困难，用户文件的细小变化都将导致存储节点上与该文件相关的所有编码数据失效，若采用重新编码和存储的方法，将会带来大量的通信和计算开销。

为了降低数据更新时的通信和计算开销，文献[26]进行了关于纠删码的差分更新研究。Chesterfield 等[26]利用分层散列和纠删码的方法解决了无线广播信道的更新问题，该方案能够节省能耗和减少数据同步时间。Irmak 等[27]提出了一种基于开源工具 rsync 的文件同步方案，利用纠删码来减少通信开销。Mohammad 等[28]提出了一种差分更新模型 DUM，

在更新过程中只传输文件变动的部分，减少了更新时的数据传输量；文献[5]中提出文件差分更新方案DeltaNC，该方案给出了 DUM 的实现方法，通过剔除零元素的方法减少差分数据的规模，该方案要求存储节点具有编码和译码能力，不适合目前的生产环境，难以部署，缺乏实用性。

综上所述，传统数据更新方案通信开销和计算开销太大，现有的差分更新方案要求存储节点具有编码能力，扩展性差，难以部署和实现。本文针对这些问题，提出了一种新的差分更新方案，能进一步地减少更新过程的通信开销，存储节点无需编码能力，可支持多样化的存储介质，具有良好的扩展性，能够适应数据更新密集型的应用场景。

3 基于网络编码的云存储系统模型

基于网络编码的云存储系统中，通过对用户的原始文件进行切分和编码，生成一系列带有冗余信息的编码块，分散存储在云端的各数据节点上，提供一定程度的容错和抗毁能力。本节对现有的基于网络编码的云存储系统进行抽象和建模，讨论并分析了数据的存储、读取和更新过程。

3.1 模型架构

本文对实际网络编码存储系统进行了简化和抽象，得到了一个简化的云存储模型，如图 1 所示。该模型由客户端和云端数据节点 2 部分构成。客户端通过代理与云存储系统进行交互，代理程序负责用户数据的编码和译码、上传和下载，以及数据的修复和更新。数据节点分布于网络中，通过安全信道与客户端相连。数据节点支持不同类型的存储介质和存储系统，包括普通的 PC 机、商用服务器、RAID 类磁盘、NAS 存储设备等，甚至 CSP 或大型数据仓库。所涉及的符号及其含义如表 1 所示。

为了简化又不失一般性，本文对云存储系统做如下假设。

1) 每个用户文件被切分成 k 个原始数据块，通过编码生成 n 个编码块，原始数据块和编码块的大小均为 s byte，编码块被分散存储在 n 个数据节点上，每个数据节点仅存储一个编码块。

2) 所有的编码方案均符合 MDS 属性[4]，即从 n 个编码块中，任选 k 个进行译码能够完全还原出原始文件。

3) 编码矩阵均为可逆矩阵，保证编码数据可被正确译码。

4) 所有的运算遵循伽罗华域上的运算法则，符合其运算性质[29]。

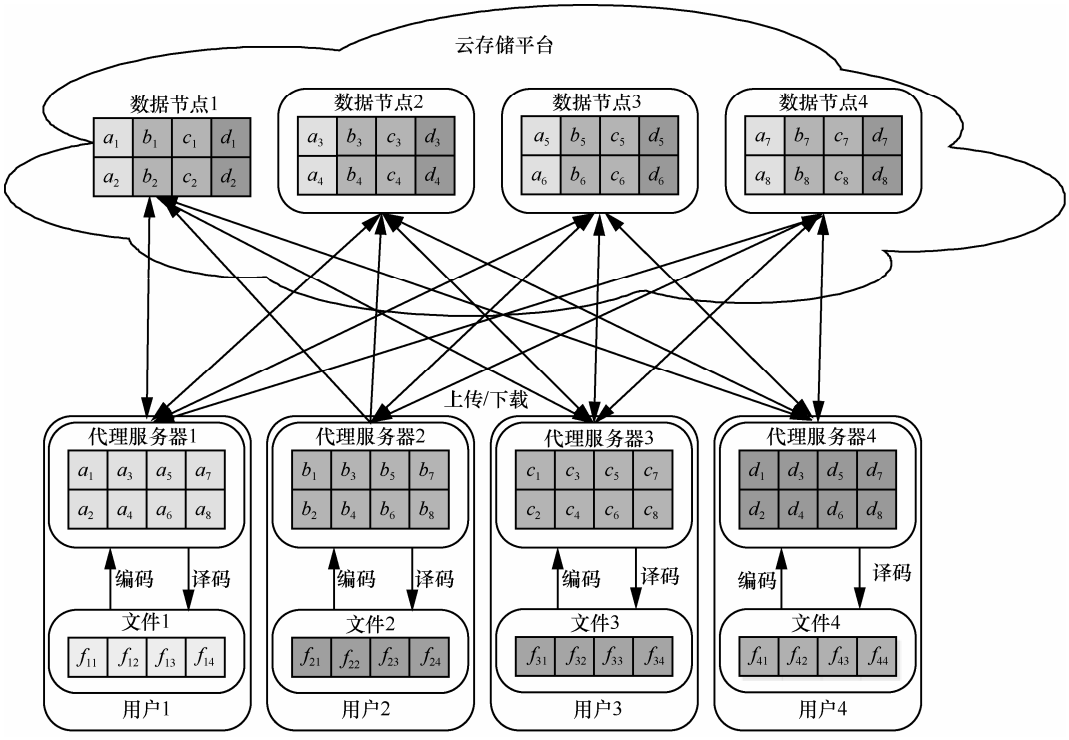


图 1 基于网络编码的云存储系统框架

表 1　符号与说明

符号	说明
$\boldsymbol{F}$	由文件 *file* 所有原始数据块组成，具有 k 个元素
$\boldsymbol{P}$	由文件 *file* 所有编码数据块组成，具有 n 个元素
s	每个数据块的大小，以字节为单位
k	原始数据块的数目
n	编码数据块的数目
$R=\frac{n}{k}$	冗余度，表示存储系统数据的冗余程度
$f_i(i=1,2,\cdots,k)$	文件 *file* 的第 i 个原始数据块
$p_j(j=1,2,\cdots,n)$	文件 *file* 的第 j 个编码数据块
GF(q)	元素个数为 q 的伽罗华域
$\boldsymbol{EM}=(\alpha_{i,j})_{n\times k}$	编码矩阵，是一个 $n\times k$ 的可逆矩阵，其中每个元素都来自 GF(q)
$\boldsymbol{EMV}_i$	编码块 p_i 对应的编码向量
$N_i(i=1,2,\cdots,n)$	第 i 个存储节点

3.2　数据存储

用户端文件需经过切分和编码才能交付给云存储系统，单个文件的存储需要经过以下几个步骤。

1) 文件切分。将文件 *file* 切分成 k 个大小为 s byte 的数据块，记作 $\boldsymbol{F}=\{f_1,f_2,\cdots,f_k\}$，将 $\boldsymbol{F}$ 交给代理程序进行编码处理。

2) 数据编码。代理程序将原始数据块 $\boldsymbol{F}$ 编码成 $n=Rk$ 个编码数据块，记作 $\boldsymbol{P}=\{p_1,p_2,\cdots,p_n\}$。

代理程序首先构造一个编码矩阵 $\boldsymbol{EM}=(\alpha_{i,j})_{n\times k}$，其元素均从伽罗华域 GF($q$)（$q$ 一般取 2^8、2^{16}、2^{32}）中随机选取出来的。为了确保编码数据能被正确译码，要求编码矩阵 $\boldsymbol{EM}$ 必须是可逆的，编码过程可表示为

$$\boldsymbol{P}=\boldsymbol{EM}\cdot\boldsymbol{F} \tag{1}$$

编码矩阵 $\boldsymbol{EM}$ 的每个行向量，对应于一个编码块，称为编码向量，记作 $\boldsymbol{EMV}_i=[\alpha_{i,1},\alpha_{i,2},\cdots,\alpha_{i,k}]$，$i=1,2,\cdots,n$，编码向量之间线性无关。

对于任意编码块 p_i 可表示为

$$p_i=\boldsymbol{EMV}_i\cdot\boldsymbol{F}=\sum_{j=1}^{k}\alpha_{i,j}f_j \tag{2}$$

对于不同的编码存储方案，其编码参数（包括原始块和编码块的数目、数据块的大小、数据节点的数目等）、编码方式和编码块的放置策略会有所差异。任何一种具体的 MDS 码均可用一个五元组 (n,k,d,m,c) 来表示，其中，n 表示一个存储单元内的数据块总数，k 表示还原出原始数据所需的最小数据块数目，d 表示修复一个数据块所需的最小数据块数目，m 表示原始数据块的数目，c 表示编码块的数目，本文中 $n=c$，$k=m$，由于本文不讨论修复过程，参数 d 不考虑。

3) 数据分发存储。代理程序将编码块 p_i 及其编码向量 $\boldsymbol{EMV}_i$ 打包，根据放置策略发送给数据节点 N_i 进行存储，同时，客户端会为每个用户建立和维护一个索引表，每个文件占用一个表项，存储有文件元数据（包括文件名、文件属性等）和编码参数（包括编码方案、编码矩阵等），以及数据节点的索引号和地址。

3.3　数据读取

当用户需要读取云端的某个文件时，首先向本地的代理程序发出读取请求，代理程序采用就近策略从云端数据节点上下载 k 个编码数据块，通过译码恢复出目标文件。假设这 k 个编码块组成的集合为 $\boldsymbol{P}_k$，$\boldsymbol{P}_k$ 为 $\boldsymbol{P}$ 的真子集，其所对应的编码向量组成的可逆矩阵为 $\boldsymbol{DM}$,有 $\boldsymbol{DM}\cdot\boldsymbol{F}=\boldsymbol{P}_k$，因此，译码过程可表示为

$$\boldsymbol{F}=\boldsymbol{DM}^{-1}\cdot\boldsymbol{P}_k \tag{3}$$

3.4　数据更新

在现有的基于网络编码的存储方案中，由于源文件和编码数据之间存在强关联性，用户文件的细微变化，都将扩散到所有的编码块中。若采用重新编码存储的方式，将造成大量计算和通信开销，资源浪费严重。数据更新困难问题导致现有网络编码云存储系统不能适用于更新密集型应用，只限于存储冷数据集，难以提供灵活多样的存储服务。本文提出了一种网络编码云存储系统差分数据更新方案，只对变动的数据部分进行更新，可极大地减少更新带宽，一定程度上解决了网络编码云存储系统更新困难的问题。

4　方案设计

4.1　差分更新原理

本文方案要求用户端在数据更新之前保留旧版本的文件数据或记录下文件的更新操作。假设用户端的旧文件为 $\boldsymbol{F}=[f_1,f_2,\cdots,f_k]^{\mathrm{T}}$，需要更新为 $\boldsymbol{F}'=[f_1',f_2',\cdots,f_k']^{\mathrm{T}}$。

定义 1　差分文件 $\boldsymbol{\Delta}$ 用来表示 $\boldsymbol{F}'$ 和 $\boldsymbol{F}$ 之间的差异性，为 2 个文件内容的异或结果，即 $\boldsymbol{\Delta}=\boldsymbol{F}'-\boldsymbol{F}=[\delta_1,\delta_2,\cdots,\delta_k]^{\mathrm{T}}$，其分量 $\delta_i(i=1,2,\cdots,k)$ 表示差分

文件块。

差分文件与原文件的大小相同，对于 2 个文件版本内容相同的比特为 0，内容不同的比特为 1。

定理 1 假设文件 $\boldsymbol{F}$ 的编码矩阵为 $\boldsymbol{EM}$，2 个文件版本使用相同的编码矩阵，新文件 $\boldsymbol{F}'$ 编码后生成的编码块集合为 $\boldsymbol{P}'$，有

$$\boldsymbol{P}' = \boldsymbol{P} + \boldsymbol{EM} \cdot \boldsymbol{\Delta} \tag{4}$$

证明 新的编码块可表示为

$$\begin{aligned} \boldsymbol{P}' &= \boldsymbol{EM} \cdot \boldsymbol{F}' \\ &= \boldsymbol{EM} \cdot (\boldsymbol{F} + \boldsymbol{\Delta}) \\ &= \boldsymbol{EM} \cdot \boldsymbol{F} + \boldsymbol{EM} \cdot \boldsymbol{\Delta} \\ &= \boldsymbol{P} + \boldsymbol{EM} \cdot \boldsymbol{\Delta} \end{aligned}$$

证毕。

定义 2 式(4)中 $\boldsymbol{EM} \cdot \boldsymbol{\Delta}$ 表示 2 个版本的编码数据之间的差异，将其定义为差分编码文件，表示为 $\boldsymbol{\Delta}_c$，即 $\boldsymbol{\Delta}_c = \boldsymbol{EM} \cdot \boldsymbol{\Delta}$。

因此定理 1 可记作

$$\boldsymbol{P}' = \boldsymbol{P} + \boldsymbol{\Delta}_c \tag{5}$$

定理 2 对于数据节点 N_i，假设其上的编码数据块为 p_i，需要更新为 p_i'，编码向量为 $\boldsymbol{EMV}_i$，有

$$p_i' = p_i + \boldsymbol{EMV}_i \cdot \boldsymbol{\Delta} \tag{6}$$

证明 对于节点 N_i，新文件 $\boldsymbol{F}'$ 的编码块

$$\begin{aligned} p_i' &= \boldsymbol{EMV}_i \cdot \boldsymbol{F}' \\ &= \boldsymbol{EMV}_i \cdot (\boldsymbol{F} + \boldsymbol{\Delta}) \\ &= \boldsymbol{EMV}_i \cdot \boldsymbol{F} + \boldsymbol{EMV}_i \cdot \boldsymbol{\Delta} \\ &= p_i + \boldsymbol{EMV}_i \cdot \boldsymbol{\Delta} \end{aligned}$$

证毕。

定义 3 式(6)中的 $\boldsymbol{EMV}_i \cdot \boldsymbol{\Delta}$ 表示 2 个版本的单个编码块间的差异性，将定义为差分编码块 $\delta_{ci} = \boldsymbol{EMV}_i \cdot \boldsymbol{\Delta} = \sum_{j=1}^{k} \alpha_{i,j} \delta_j$。

因此，差分编码文件 $\boldsymbol{\Delta}_c$ 可表示为 $\boldsymbol{\Delta}_c = [\delta_{c1}, \delta_{c2}, \cdots, \delta_{cn}]$，定理 2 可写作

$$p_i' = p_i + \delta_{ci} \tag{7}$$

根据式(5)和式(7)可知，要对文件进行更新，需要计算差分编码文件 $\boldsymbol{\Delta}_c$，将其分量即差分编码块 δ_{ci} 发送到相应的存储节点 N_i 进行更新。

4.2 数据压缩传输

在云存储系统中，大部分更新操作往往集中于文件局部，更新比例较小，这使差分文件中存在大量的零元素。差分文件经过编码生成了一系列的差分编码块，其中的 0-1 布局被打乱重组，差分编码块中出现了许多连续的重复元素。为了进一步减少更新带宽，本文根据差分编码块的冗余特征，设计了一种游程编码和霍夫曼编码相结合的压缩方法，对差分编码块进行压缩传输。整个压缩过程分为 2 部分：首先使用游程编码，压缩数据中的连续重复数据，减少输入数据规模；然后利用霍夫曼编码作进一步的压缩，可达到较好的压缩效果。

4.2.1 游程编码

游程编码的主要思想是把连续重复的信息用长度和重复的信息单元来表示，易于实现和部署，具有较快的压缩和解压速率，对于含有大量重复信息的数据，能达到很高的压缩比。

针对差分编码块的数据冗余特征，本文设计了一种新的游程编码方案。该方案中，对于非连续重复出现的普通字节，将其直接放入压缩序列；对于单一字节重复出现的序列，进行编码生成一个压缩码写入压缩序列。压缩码的长度由该序列长度决定，介于 2 byte 和 5 byte 之间。压缩码的格式如图 2 所示，其首部字段为 FF，起分隔标识作用。为了防止产生歧义，用 FFF0 来表示单个出现的 FF。

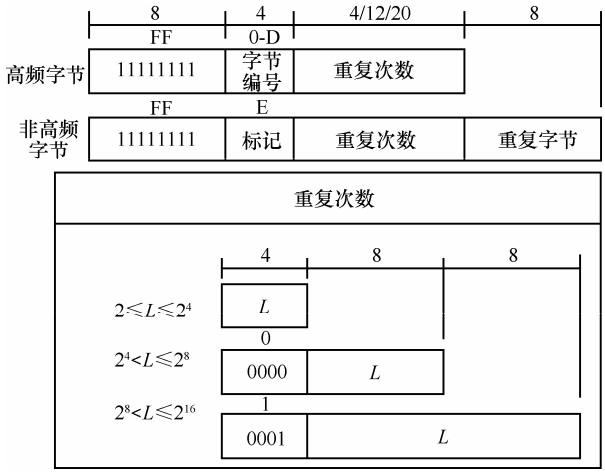


图 2 压缩码格式

生成压缩码的具体过程如下。

1) 对原码序列以字节为单位进行频数统计，找出频数最多的前 $k(k \leqslant 14)$ byte 作为高频字节，对其进行编号，通常 k 取 4 或 5。

2) 对于高频字节，字节编码字段表示其编号；对于非高频字节，标记字段写入 E（1110），起标记作用，用于区分。

3) 写入重复次数，假设连续出现的次数为 L，当 $2 \leqslant L \leqslant 2^4$ 时，用重复次数字段的前 4 位来表示 L；当 $2^4 < L \leqslant 2^8$ 时，将重复次数字段的前 4 位置为 0（0000），用 1 byte 表示 L；当 $2^8 < L \leqslant 2^{16}$ 时，将重复次数字段的前 4 位写入 1（0001），用 2 byte 表示 L；差分编码块的大小一般不会大于 512 MB (即 2^9 byte)，因此不考虑 $L > 2^{16}$ 的情况。

4) 对于非高频字节，需要将重复字节追加到压缩码最后。

如图 3 所示，对于一个 605 byte 的原码序列，假定了 4 个高频字节为 00、12、34、56，分别编号为 0、1、2、3，使用游程编码对原码序列进行压缩，得到了 24 byte 压缩码，压缩比约为 25.2:1。

4.2.2　霍夫曼编码

为了进一步减少数据规模，可使用霍夫曼编码对数据进行二次压缩。霍夫曼编码的原理是依据字符出现的频率对其进行编码，使用短码表示出现频率高的码字，使用长码表示出现频率低的码字，从而达到压缩的目的。

具体压缩过程如下。

1) 以 $\frac{1}{2}$ 字节为一个码字，统计输入串中各码字出现的频率。

2) 以码字的出现频率为权值，使用霍夫曼树对其进行编码。

3) 使用霍夫曼编码替换原码中对应的码字，写入到压缩数据中。

对图 3 压缩得到的 24 byte 数据进行霍夫曼编码，得到其霍夫曼编码树，如图 4 所示，通过二次压缩，可将 24 byte 压缩至不到 16 byte，压缩比为 3:2。综合 2 个压缩过程，整个数据压缩过程的压缩比为 37.8:1。

4.2.3　联合压缩技术

整个数据压缩过程如图 5 所示。在联合压缩方法中，游程编码是对数据的预处理，旨在减少数据中连续码字带来的冗余，使霍夫曼编码更好地发挥作用。该压缩方法结合了 2 种压缩算法的优点，以实现更大的压缩比，若输入原码中连续码字较多时则第一次压缩起主要作用，若连续码字不多则第二次压缩能弥补其不足，达到更高的压缩比。这种混合压缩算法比单纯的游程编码压缩率更高，比单纯的霍夫曼编码更节省时间。解压缩的算法过程是先进行霍夫曼解压缩再进行游程解压缩，由于这 2 种编码都是可逆的，解压缩即压缩的逆过程，在此不再赘述。

图 5　整个数据压缩过程

4.3　数据更新方案

综上所述，文件的更新过程可以描述如下。

1) 客户端计算 2 个文件版本的差分文件 $\boldsymbol{\Delta} = \boldsymbol{F}' - \boldsymbol{F}$；

2) 使用先前的编码矩阵 $\boldsymbol{EM}$，对差分文件进行

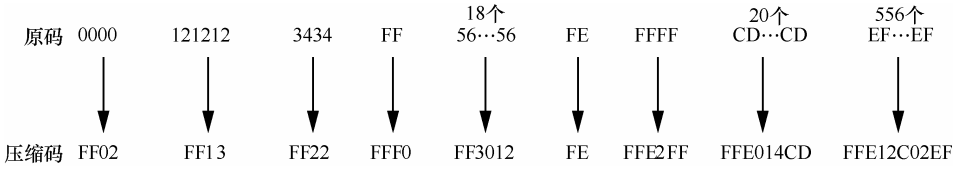


图 3　游程编码示例

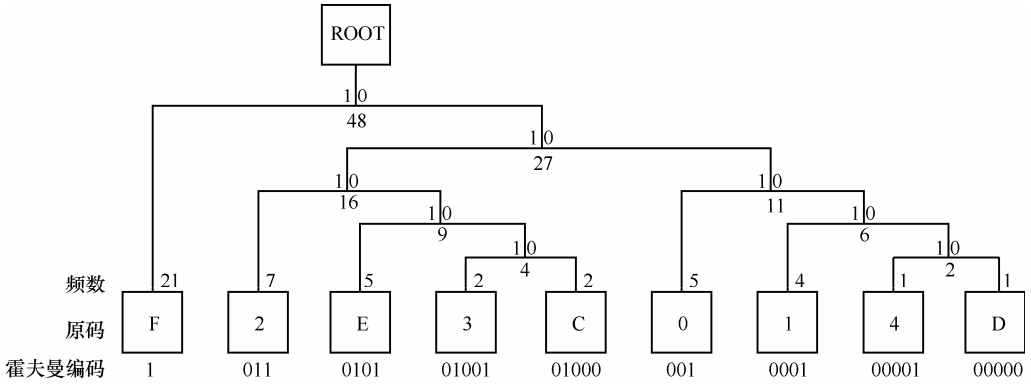


图 4　霍夫曼编码树

切分编码，生成差分编码文件 $\boldsymbol{\Delta}_c = \boldsymbol{EM} \cdot \boldsymbol{\Delta}$；

3) 差分编码文件 $\boldsymbol{\Delta}_c = [\delta_{c1}, \delta_{c2}, \cdots, \delta_{cn}]$ 由 n 个差分编码块组成，分别对这 n 个差分编码块进行压缩，生成 n 个压缩的差分编码块 $\delta_{cz1}, \delta_{cz2}, \cdots, \delta_{czn}$；

4) 客户端将将压缩后差分编码块 δ_{czi} 按先前的放置策略发送给数据节点 N_i；

5) 数据节点 N_i 将收到的数据 δ_{czi}，进行解压缩得到差分编码块 δ_{ci}；

6) 差分编码块与原编码块进行异或运算，$p_i' = p_i + \delta_{ci}$，得到新的编码块，更新完成。

文件的更新过程如图 6 所示。

5 系统实现与性能测试

本文设计实现了一个基于网络编码的云存储原型系统，在真实的网络环境下实现部署了本文方案，并进行了相关的性能测试，与现有的更新方案进行了对比分析。

5.1 系统的实现

原型验证系统设计架构如图 7 所示。整个系统采用基于代理的结构，用户通过在终端运行代理程序与存储节点进行交互。代理程序负责用户数据的处理和维护，对上层用户和应用程序提供统一的标准数据访问接口，屏蔽下层的实现细节，对下兼容不同类型的存储介质和存储系统，能够适应不同的存储和网络设置。

代理端采用分层的结构设计，分为文件系统层、编码层和存储层 3 个功能层次。

文件系统层负责与用户和应用程序进行交互，对上提供标准的数据访问接口（包括读、写、删除和更新等操作接口），负责文件信息的管理和维护，在文件更新时，计算 2 个文件版本之间的差分文件。文件系统层是基于开源的文件系统框架 FUSE[30]开发实现的，是一个应用层的文件系统。

编码层负责对数据进行切分和编码，包括编码器、译码器、编码管理和配置模块，实现了几种常用的网络编码方案，具有良好的扩展性。编码器和解码器对外提供了编程接口，用户可以利用这些接口实现自定义的编码方案，替换已有的编码方案，或建立自己的编码方案库。

存储层负责数据块的存取访问，将编码数据块进行压缩，分发给云端存储节点进行存储；可根据文件系统层的数据块访问请求，从存储节点上下载相应的编码块。该层是一个虚拟存储层，没有具体

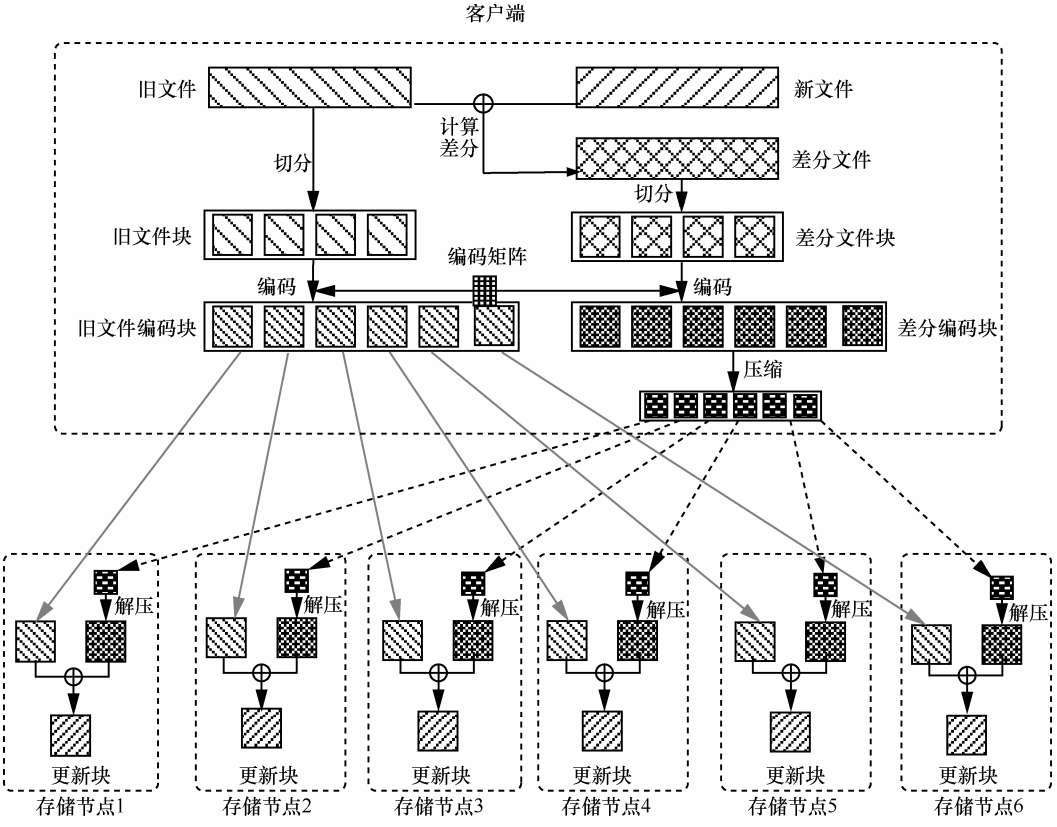


图 6 文件更新过程

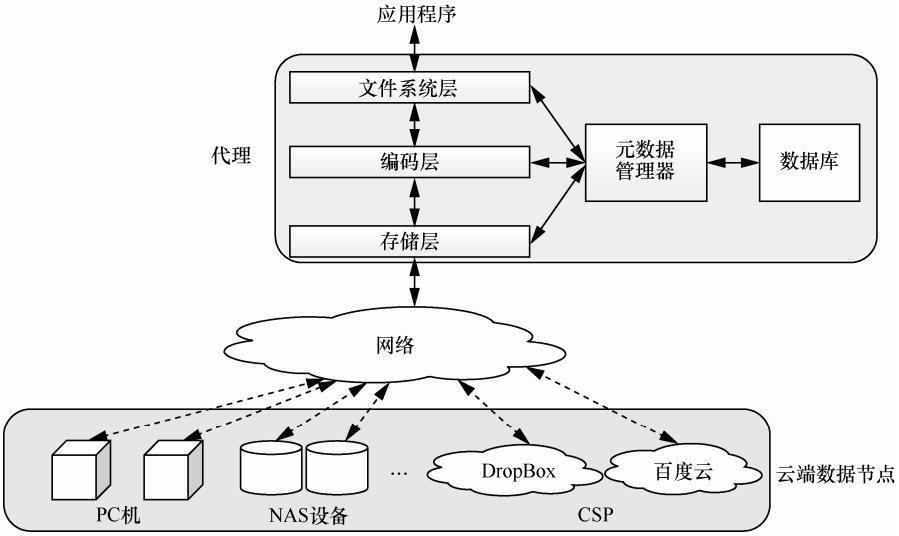


图 7　原型验证系统设计架构

的存储实现，对下层的存储介质进行了抽象，可兼容和支持不同类型的存储介质和分布式存储系统，对上提供统一的访问接口，存储实现细节对上层完全透明。

元数据管理器负责对文件和数据块进行管理和维护，它将文件元数据（包括文件名、文件属性、读写权限等）、编码参数（包括编码方案、编码矩阵等）和数据块元数据（包括数据块索引、存储节点 IP 地址等）存储在数据库中，为各个层所共享使用。

客户端通过安全信道与存储节点相连，存储节点支持多样化的存储介质和存储系统，可以是普通的 PC 机、廉价的商用服务器、NAS 设备，也可以是云服务提供商或大型的数据仓库等。每个存储节点上运行一个更新守护进程，负责接收差分编码块，对差分编码块进行解压缩和完整性校验，完成文件的更新。

本文在该原型系统上实现部署了再生码方案 F-MSR[6]，在 F-MSR 的基础上实现的本文方案和差分更新方案 DeltaNC[5]，并进行了一系列相关的性能测试。

5.2　性能测试

本文利用局域网环境和 OpenStack 来模拟真实的云存储环境，实验拓扑和相关配置如图 8 所示。客户机运行代理程序，6 个存储节点由 OpenStack 虚拟而来，每个节点上运行一个更新守护进程，负责对用户数据进行更新。存储节点和客户机通过局域网连接，为了简化代理程序的存储层，存储节点通过网络文件系统 NFS 挂载在客户机目录下，这样客户主机可以直接通过访问本地目录来访问存储节点。本文设计和进行了相关实验，测量和分析了文件大小、更新比例和冗余度 3 个因素对更新性能的影响，并与现有的方案进行了对比分析。

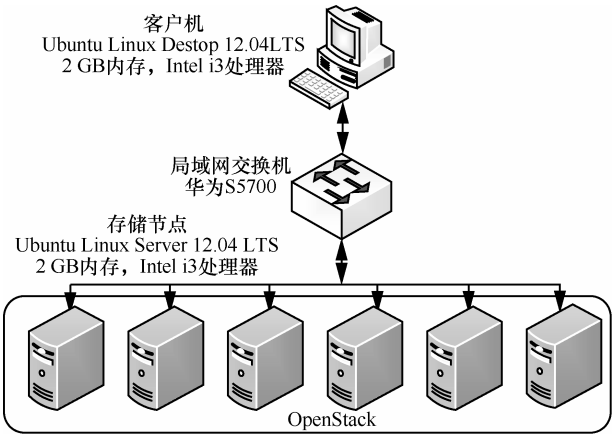


图 8　实验拓扑和相关配置

5.2.1　文件大小对更新性能的影响

本文使用编码参数 $k=4$，$n=6$，每次对文件内容做 20%的更新，分别使用本文方案、DeltaNC 和 F-MSR 3 种方案对单一文件进行更新，测量了 3 种方案随着文件大小指数增长过程中的更新带宽和所用时间。

更新过程的带宽开销如图 9 所示。假设数据对象的大小为 M，DeltaNC 方案需要为每个存储节点传输一份压缩后的差分文件，假设其压缩率为 $\alpha(0\leqslant\alpha\leqslant1)$，则其更新过程的带宽开销为 $nM\alpha$ 。

本文方案在客户端为每个存储节点生成差分编码块，经过压缩后传输给对应的存储节点，假设差分编码块的平均压缩率 $\beta(0 \leqslant \beta \leqslant 1)$，则总的更新带宽开销为 $\frac{Mn\beta}{n-2}$，由于 $\beta < \alpha$，所以本文方案的传输带宽小于 DeltaNC 方案传输带宽的 $\frac{1}{n-2}$，可明显缓解数据更新过程中的网络传输压力。

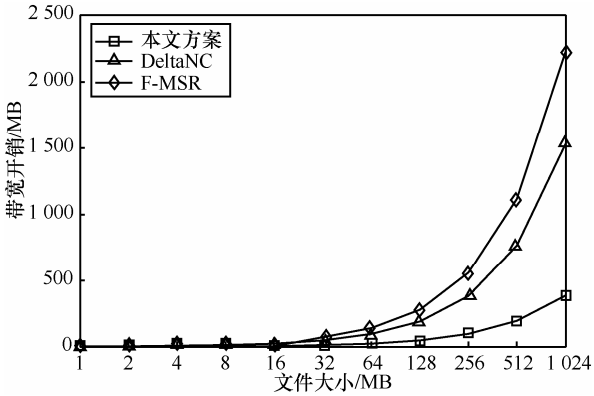


图 9 文件大小对更新带宽开销的影响

更新过程中的时间开销如图 10 所示，本文方案的更新时间小于其他 2 种方案。更新时间由 3 部分组成：客户端处理时间 T_c、数据传输时间 T_t 和存储节点处理时间 T_n。本文方案在客户端需要对数据进行编码和压缩，比较耗时，但传输时间和数据节点处理时间相对较少，整体更新时间略小于 DeltaNC 方案。

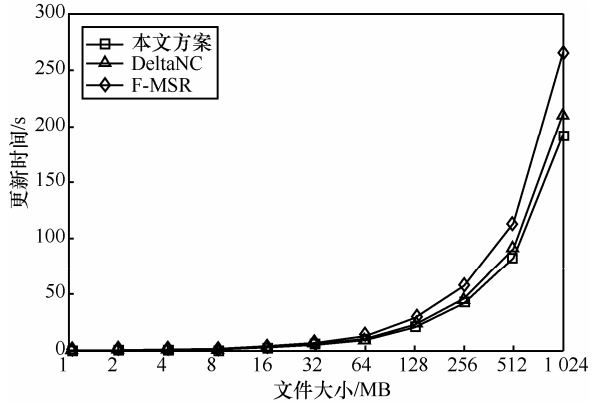


图 10 文件大小对更新时间的影响

5.2.2 更新比例对更新性能的影响

本文还测量和分析了文件更新比例对更新性能的影响。实验选取了一个大小为 16 MB 的文件，不断增加其更新比例，测量和统计每次实验中 3 种方案的更新带宽和时间开销，并进行了对比分析。

更新比例对通信开销的影响如图 11 所示。由于 F-MSR 方案不是差分更新方案，其更新带宽为编码数据大小，与更新比例无关。本文方案比 DeltaNC 方案通信开销更小，当更新比例较小时，可节省大量的带宽资源，缓解网络传输压力。

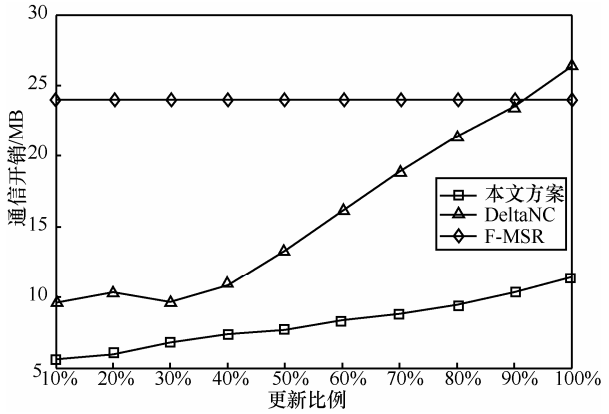


图 11 更新比例对更新通信开销的影响

更新时间随更新比例的变化如图 12 所示。F-MSR 方案更新时间不受更新比例的影响，在相同更新比例下，本文方案的更新时间比 DeltaNC 更少。从图 12 可知，当更新比例较小时，本文方案能够节省大量时间，更能适应更新密集型的应用场景。

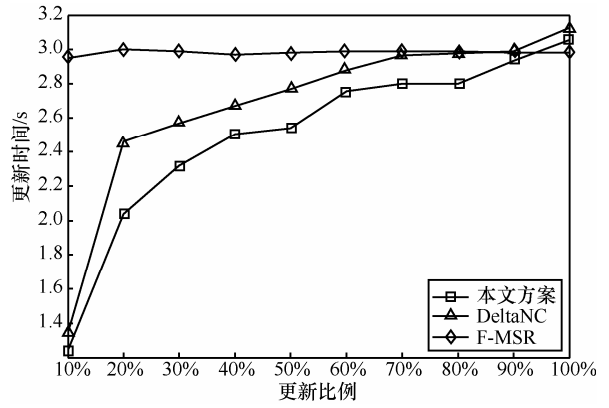


图 12 更新比例对更新时间的影响

5.2.3 冗余度对更新性能的影响

收敛时间定义为从用户提交更新操作到所有的存储节点完成数据更新所用的时间，本文在 100 Mbit/s 带宽的局域网内对 3 种方案的收敛时间进行了测试。

实验选取一个大小为 4 MB 的文件，每次更新其 20%的内容，通过不断增加其编码数据块的数目 n 来增加存储系统的冗余度 R，分别测量了 3 种方

案在不同的冗余度下的收敛时间，实验结果如图 13 所示。从图 13 可知，本文方案的收敛时间比另外 2 种方案有了较大提升，能够适应大规模数据节点的应用场景。

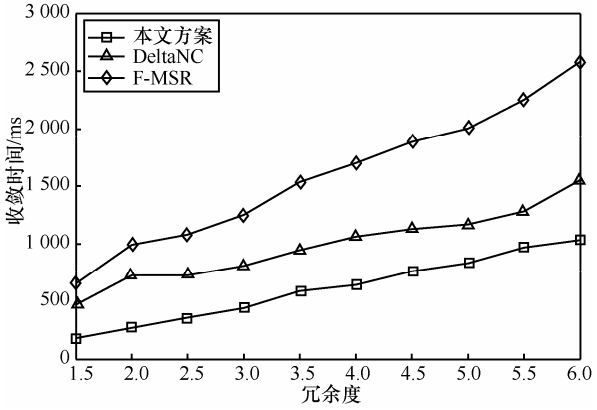


图 13　冗余度对收敛时间的影响

6　结束语

本文针对现有的网络编码云存储系统数据更新困难，更新过程通信开销过大的问题，提出了一种新的差分数据更新方案，通过对文件中更新部分进行网络编码和压缩传输，显著降低了更新过程中的通信开销；第一次在真实的网络环境下运用实验手段对网络编码云存储数据更新问题进行探索和研究；设计了一种游程编码和霍夫曼编码相结合的压缩方法，具有较好的压缩效果；本文方案不要求存储节点具有编码和译码能力，简化了云存储平台的设计，可支持多样化的存储介质；设计实现了基于代理的云存储原型系统，采用低耦合的分层结构，具有良好的扩展性；在真实网络下进行了实验部署，实验结果表明，与现有方案相比，本文方案具有更小的通信开销和更快的更新速率，能够较好地适应数据更新密集型的应用场景，对缓解数据中心网络的传输压力，改善网络性能具有重要意义。

实际的网络编码方案比较复杂，更新过程中出错概率较大，本文方案缺乏对文件的一致性检查和错误处理机制，如何增强本文方案的安全性和可靠性将是下一步研究的重点。

参考文献：

[1] HASHEM I A T, YAQOOB I, ANUAR N B, et al. The rise of “big data” on cloud computing: review and open research issues[J]. Information Systems, 2014, 47(47): 98-115.

[2] JIEHUI J U, JIYI W U, JIANQING F U, et al. A survey on cloud storage[J]. Journal of Computers, 2011, 6(8): 1764-1771.

[3] Amazon. What is Amazon S3?[Z]. Amazon Simple Storage Service Developer Guide. 2006.

[4] LI H. Introducing windows azure[Z]. David Chappell & Associates White Paper, 2010.

[5] ZAKERINASAB M R, WANG M. DeltaNC: efficient file updates for network-coding-based cloud storage systems[C]//IEEE International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems. 2013: 360-364.

[6] HU Y, LEE P P C, SHUM K W. Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems[C]//IEEE INFOCOM. 2012: 2355-2363.

[7] 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 and Development, 2012, 49(1): 1-11.

[8] LI J, LI B. Erasure coding for cloud storage systems: a survey[J]. Tsinghua Science & Technology, 2013, 18(3): 259-272.

[9] RASHMI K V, SHAH N B, GU D, et al. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: a study on the facebook warehouse cluster[J]. Usenix Hotstorage, 2013:8.

[10] REED I S, SOLOMON G. Polynomial codes over certain finite fields[J]. Journal of the Society for Industrial & Applied Mathematics, 1960, 8(2): 300-304.

[11] PLANK, JAMES S. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems[J]. Software—practice & Experience, 1996, 27(9):995-1012.

[12] DIMAKIS A G, GODFREY P B, WU Y, et al. Network coding for distributed storage systems[J]. IEEE Transactions on Information Theory, 2007, 56(9): 2000-2008.

[13] BLOMER J, KALFANE M, KARP R, et al. An XOR-based erasure-resilient coding scheme[C]//Proc Acm Sigcomm. 1999.

[14] BLAUM M, BRADY J, BRUCK J, et al. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures[J]. IEEE Transactions on Computers, 1995, 44(2):192-202.

[15] CORBETT P, ENGLISH B, GOEL A, et al. Row-diagonal parity for double disk failure correction[C]//3rd USENIX Symposium on File and Storage Technologies (FAST'04), 2014:1-14.

[16] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system[J]. ACM Sigops Operating Systems Review, 2003, 37(5):29-43.

[17] DHRUBA B. HDFS architecture guide[Z]. Hadoop, 2008.

[18] AHLSWEDE R, CAI N, LI S Y R, et al. Network information flow[J]. Fundamenta Informaticae, 2006, 72(1-3): 167-180.

[19] GKANTSIDIS C, RODRIGUEZ P R. Network coding for large scale content distribution[C]//Infocom, Joint Conference of the IEEE Com-

puter & Communications Societies. 2005: 2235-2245.

[20] DIMAKIS A G, GODFREY P B, WAINWRIGHT M J, et al. Network coding for distributed storage systems[C]//IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications. 2007: 2000-2008.

[21] DIMAKIS A G, GODFREY P B, WU Y, et al. Network coding for distributed storage systems[J]. IEEE Transactions on Information Theory, 2010, 56(9): 4539-4551.

[22] HU Y, YU C M, LI Y K, et al. NCFS: on the practicality and extensibility of a network-coding-based distributed file system[C]// International Symposium on Network Coding. 2011: 1-6.

[23] RASHMI K V, SHAH N B, KUMAR P V. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction[J]. IEEE Transactions on Information Theory, 2011, 57(57): 5227-5239.

[24] RASHMI K V, SHAH N B, KUMAR P V, et al. Explicit construction of optimal exact regenerating codes for distributed storage[C]//47th Annual Allerton Conference on Communication, Control, and Computing, 2009: 1243-1249.

[25] CHEN H C H, HU Y, LEE P P C, et al. NCCloud: a network-coding-based storage system in a cloud-of-clouds[J]. IEEE Transactions on Computers, 2014, 63(1): 31-44.

[26] CHESTERFIELD J, RODRIGUEZ P. DeltaCast: efficient file reconciliation in wireless broadcast systems[C]//International Conference on Mobile Systems, Applications, and Services. 2005: 93-106.

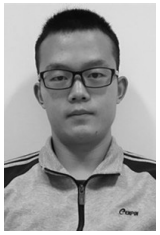
[27] IRMAK U, MIHAYLOV S, SUEL T. Improved single-round protocols for remote file synchronization[J]. Proceedings-IEEE INFOCOM, 2005, 3: 1665-1676.

[28] MOHAMMAD R Z, WANG M. An update model for network coding in cloud storage systems[C]//Communication, Control, and Computing. 2012:1158-1165.

[29] ZHANG M, NEMAT A B, BLISS D E. Galois field multiplication system and method: US, US7526518[P]. 2009.

[30] RATH N, SZEREDI M, et al. FUSE[DB]. 2016-10-04.

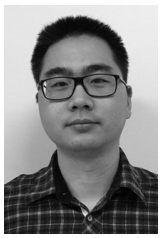
作者简介：



王龙江（1991-），男，陕西商洛人，解放军信息工程大学硕士生，主要研究方向为网络信息安全、云存储安全等。

陈越（1965-），男，河南开封人，博士，解放军信息工程大学教授、博士生导师，主要研究方向为网络与信息安全。

严新成（1991-），男，河南信阳人，解放军信息工程大学硕士生，主要研究方向为网络信息安全、基于机密的云数据访问控制。



黄恺翔（1987-），男，河南新乡人，解放军信息工程大学博士生，主要研究方向为网络与信息安全等。