一种基于网络编码的云存储系统

刘宴涛[1] 刘 珩[2]

(渤海大学工学院 辽宁 锦州 121000)[1] (北京理工大学信息与电子学院 北京 100081)[2]

摘 要 存储空间、修复带宽和更新带宽是云存储系统的 3 个重要指标,系统设计往往需要在这些性能度量之间取折衷。为了降低存储空间、修复带宽、更新带宽以及系统复杂度,文中提出了一种基于网络编码的云存储系统。该系统结构为 $m*n$ 数据阵列的形式,n 列表示 n 个存储节点,其中 k 个节点用于存储原始数据,称为系统部分;另外$(n-k)$个节点用于存储校验字符,称为非系统部分。数据阵列的 m 行对应 m 个系统形式的(n,k)最大距离可分(MDS)码,每个源数据符号只参与它所在行的编码,不参与其他行的编码,这种系统结构大幅降低了编译码的复杂度。该系统可以承受最多$(n-k)$个节点的失效,此外,当单节点失效时,由于使用了系统形式的 MDS 码,可以使用干扰对齐技术进一步缩减修复带宽。与现有的某些云存储系统相比,该系统明显降低了存储空间、修复带宽和更新带宽等资源消耗,性能得到大幅提升。

关键词 网络编码,云存储,最大距离可分码,干扰对齐

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.12.047

Cloud Storage System Based on Network Coding

LIU Yan-tao[1] LIU Heng[2]

(College of Engineering, Bohai University, Jinzhou, Liaoning 121000, China)[1]

(School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China)[2]

Abstract Storage, repair bandwidth and update bandwidth are three performance metrics for a cloud storage system. System design needs to make trade-off among them. To decrease the consumption on storage, repair bandwidth update bandwidth and system complexity, this paper proposed a network coding based cloud storage system. This system is in the form of an $m*n$ data array. The n columns stand for n storage nodes, which are comprised of two parts, one is systematic part which stores source symbols, and the other is nonsystematic part which stores parity symbols. The m rows of the data array stand for the number of $m(n,k)$ systematic Maximum Distance Separable (MDS) code. Any source symbol is only involved into encoding within the unique row in which it locates and is not used by other rows. Such a structure significantly decreases the complexity of encoding and decoding. The functionality of the system is still available even in front of the failures of less than $(n-k)$ nodes. Moreover, by using interference alignment, systematic MDS code is beneficial to further reduce repair bandwidth in case of single node failures. Compared to some existing cloud storage schemes, the system greatly reduces the resource consumption on storage space, update bandwidth and repair bandwidth, so its performance is improved significantly.

Keywords Network coding, Cloud storage, MDS code, Interference alignment

1 引言

云存储系统和服务(如 Dropbox、微软的 OneDrive、亚马逊的 S3)极大地方便了网络用户在任何时间和任何地点存取及管理数据。与传统数据中心采用的集中式的数据存储和管理方式不同的是,云存储系统采用网络存储模式,即把由原始数据衍生出的多个副本存储在分布于网络各处的多个节点中,用户可以通过调取某些节点上的存储内容来恢复和重建原始数据,因此数据存储系统的架构逐渐由中心式过渡到了扁平式。对于一个数据存储系统而言,其最基本的要求是数据保存的可靠性,这往往需要通过增加数据冗余来实现。增加数据冗余最简单的实现方式是在多个存储节点保存原始数据的多份拷贝,但简单的拷贝存储会造成巨大的存储浪费,图 1 给出了简单的复制存储和奇偶校验码存储两种策略,图中的两种码都占用了 n 个数据块的存储空间。从存储效率上看,复制码只存储了 1 个数据块,奇偶校验码存储了 $n-1$ 个数据块,可见后者的存储效率远远高于前者。然而,对于图 1(b)所示的这种基于编码的存储系统,在提高存储效率的同

到稿日期:2017-11-07 返修日期:2018-02-28 本文受国家自然科学基金(61471045),辽宁省自然科学基金项目(20170540008)资助。

刘宴涛(1975-),男,博士,副教授,主要研究方向为 Ad hoc 网络、网络编码和网络仿真等;刘 珩(1982-),女,博士,副教授,主要研究方向为 Ad hoc 网络、传感器网络、分布式系统和网络编码等,E-mail:lhengzzt@bit.edu.cn(通信作者)。

时带来了另一个问题，即当某个存储节点损坏时，需要启动一个自动修复过程，通过向依然存活的节点请求数据来计算和恢复损毁的数据并存储到一个新建节点上，这被称为系统的修复问题。从修复功能上看，图 1(a)中的复制码能修复任何 $n-1$ 以内的数据块，所需修复带宽(即修复过程中所需的数据块数目)仅为 1 个数据块；奇偶校验码只能修复 1 个数据块，所需的修复带宽为 $n-1$ 个数据块，如果有 2 个以上数据块损坏，系统将崩溃无法自行修复。由此可见，复制存储和简单的奇偶校验码存储代表了分布式存储的两个极端，即极低的存储效率伴随极高的修复性能(复制码)和极高的存储效率伴随极低的修复性能(奇偶校验码)。最大距离可分(MDS-Maximum Distance Separable)码实现了二者的折衷，以 Reed-Solomon 编码方式为例，(n,k)-MDS 码把 k 个源数据块编码成 n 个码数据块，从其中任意取出 k 个码数据块即可恢复全部的码数据块和源数据块，因此其码率和修复带宽分别为 k/n 和 k。复制存储、奇偶校验码和 MDS 码的性能比较如图 2 所示，由此可见，相对于简单的复制存储而言，存储码是更为高效的增加数据冗余的方法[1-2]。存储编码可以分为冗余磁盘阵列、删除码、网络编码，以及这些编码方法的结合等。

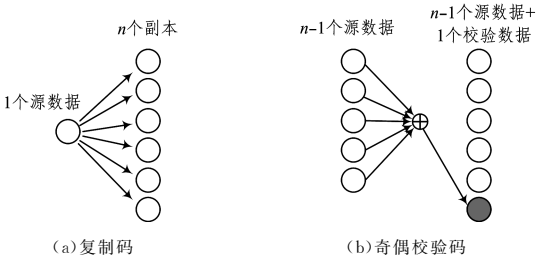
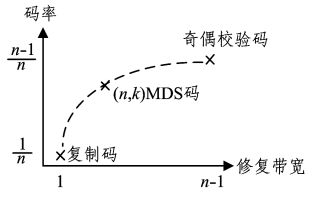


图 1 复制码和奇偶校验码

Fig. 1 Replica code and parity check code



注：左上角(即高码率、低修复带宽)为最优性能区

图 2 3 种存储码的性能比较

Fig. 2 Performance comparison of three storage codes

云存储系统由于需要存储数据冗余以及在节点间的传输数据，不可避免地要消耗一些网络资源，其中存储空间、更新带宽和修复带宽是评价一个云存储系统性能的 3 个重要指标。对于一个基于存储编码的云存储系统而言，即使改变一个原始数据块，也会造成系统中编码数据块的过时，因此系统需要启动一个更新过程来计算、传输和更新这些数据，称在更新过程中传输的数据符号数为更新带宽，称归一化到更新一个原始数据符号所需的更新带宽为归一化更新带宽。此外，对于一个云存储系统，不可避免地存在着存储节点的损坏、毁伤、宕机或者离开网络，从而造成数据的丢失，此时系统需要发起一个修复过程，一方面重新选取或建立新的存储节点，另一方面通过向依然存活的节点请求数据来计算和恢复新建立的存储节点上应该存储的数据。修复过程传输的数据符号数被称为修复带宽，归一化到修复一个原始数据符号所需的修复带宽被称为归一化修复带宽。由此可见，更新带宽和修复带宽度量了一个云存储系统在更新过程和修复过程中的计算和通信代价。

显然，一个高效的云存储系统应该对应着较低的存储、更新带宽和修复带宽的消耗，同时还要兼顾系统复杂性的考量，然而图 2 所示的 3 种存储方式中不存在这样一种在全部性能指标上都表现最优的编码策略。在实际应用中，必须根据实际需求在众多性能度量中进行折衷，建立最符合实际需要的编码存储系统，这一直是云存储系统设计者们孜孜以求的目标。Dimakis 等[1-2]深入探讨了云存储系统的修复问题，提出了一种基于网络编码的云存储编码策略，其被称为再生码，这种码应用网络编码技术降低了修复带宽。自此之后，再生码一直是存储码的热点研究课题。文献[1-2]把修复问题等效为网络组播传输问题，提出了功能性修复和精确修复两种修复策略，应用信息流图的方法确定了存储和修复带宽之间的定量关系，推导了这两种性能指标的限并提出达到这两种性能限的两种码——最小存储再生码和最小带宽再生码，讨论了如何在两者之间取得折衷。然而，如果把一个基于网络编码的云存储系统看作一个数据阵列，那么上述编码策略既使用了行内编码也使用了行间编码，从而加剧了系统设计和编译码的复杂度；此外，对于多节点失效的问题，其只能逐一修复，不能同时修复。Acedański 等[3]比较了无编码的随机存储、传统删除码和随机线性网络编码 3 种存储码的有效性问题，建立了译码概率和存储、带宽之间的函数关系，通过计算和比较证明了随机线性网络编码相比于删除码具有同样优越的性能，同时大大减少了存储消耗。

除了文献[1-3]展开的理论研究外，许多研究者努力寻找或设计了基于网络编码的实用且高效的云存储系统。Zakerinasab 等[4-5]提出了一种减小更新带宽的网络编码方法，其基本思想是通过保持编码系数不变，使得更新后的新码字是更新前老码字加上一个简单的码字差函数，更新过程仅仅需要传输该差函数即可，极大地降低了更新带宽的消耗。类似地，王龙江等[6]提出了一种基于网络编码的云存储系统的差分数据更新方案，基于数据更新往往集中在文件的局部、更新比例小的特点，结合游程码和霍夫曼码等压缩码对差分编码块进行压缩，从而大幅减小了更新带宽。吴昊等[7]综合考虑了更新过程中存在的数据泄漏隐患，提出了一种基于差值矩阵的更新算法，寻找尽可能少的需要更新的编码块，在降低计算和通信成本的同时提高了安全性。Wu 等[8]提出了一种基于干扰对齐(interference alignment)技术的降低修复带宽的方法。干扰对齐技术[9]最初是一种通信信号设计技术，其基本思想是通过对多个发射信号的联合设计，使得在接收端的非目的接收机处产生人为的叠加衰落，同时又保持目的接收机对其想接收信号的灵敏度和区别度。采用与干扰对齐类似的原理，文献[8]通过对多变量线性方程组进行处理，减少了所需要的变量个数，从而降低了修复带宽消耗。黄倩[10]探讨了干扰对齐技术在云存储系统修复过程中的应用，提出了一种基于干扰对齐的多节点最小存储精确再生码。针对 NCCloud 系统[11]需要做大量矩阵计算导致计算量大的问题，张俊

峰[12]提出了一种基于预计算修复矩阵的云存储修复策略，其使用近似相同的修复数据块对不同的受损节点进行修复。谢垂益等[13]提出了一种基于伽罗华域 GF(2^s)应用随机线性网络编码的云存储系统设计方案，其关键点在于解码时不是随机选取节点读取编码数据，而是优先选择那些使用了单位向量作为编码系数的节点，从而降低了解码的复杂度。Xu 等[14]探讨了存储码在车际网络中的修复问题，为了计算最小修复带宽，其另辟蹊径，采用了如下信息论的方法：k 个源数据块通过(n,k)-MDS 码编码为 n 个码数据块并存储在 n 辆车中，假设由于某辆车离开网络造成了数据块 X 丢失，其信息量等于 α，从其他车辆下载数据块的集合记为 Y，则可修复的下载方案应该满足 $I(X;Y)=\alpha$，进一步从所有可修复的下载方案中确定最小修复带宽。然而，需要说明的是，文献[14]能够计算互信息 $I(X;Y)$的一个前提是源数据块 X 必须是在空间 F_2^s 中均匀分布的，这种依赖信源概率分布的分析方法存在很大的局限性。上述研究在讨论云存储系统的更新或修复问题时，大部分是基于均匀对等的网络架构加以分析，但在实际应用中，很多网络并不是全连通的，这导致节点之间的通信代价并不是均匀分布的，因此网络修复问题与网络拓扑紧密相关。Sipos 等[15]把网络感知的思想应用于网络修复中，通过定义某种代价函数来反映网络节点之间当前的连通状态，从而确定最优修复方案。

Elyasi 等[16]提出了一种基于行列式码的精确修复再生码的构造方法，该再生码的参数为(n,k,d,α,β,M)，其中，n 表示节点数，任意的 k 个节点都可以重建整个系统，在修复过程中一共使用 d 个帮助节点，α 和 β 分别表示每个节点存储的符号数和在修复过程中从每个帮助节点中请求的符号数，M 表示源文件总的符号数。文献[16]构造的再生码存储系统的参数需要满足 $d=k$，且有：

$$(\alpha,\beta,M)=\left(\binom{k}{i},\binom{k-1}{i-1},i\binom{k+1}{i+1}\right)$$

其中，i 是介于 1 和 k 之间的整数。该码依靠一个非常复杂的编码过程，具体包括两个步骤：1)预编码，即把原始的 M 个符号编码为一组 $k\binom{k}{i}$ 个中间符号，并把这些中间符号按某种特定顺序排列成一个特殊的矩阵；2)将该矩阵与一个范德姆型的生成矩阵相乘即可得到各个节点存储的码符号。由于对应于任意 k 个节点的范德姆矩阵的子矩阵都是可逆的，因此依靠任意的 k 个节点足以恢复全部数据。然而，文献[16]并不是最小存储码，也就是说并不满足 $\alpha=M/k$，而且该码也不是系统形式的。Wang 等[17]提出了一种系统形式的最小存储再生码的构造方法，该系统参数为(n,k,α)，其中 n,k,α 分别表示节点数、系统节点数和每个节点存储的符号数，且这 3 个参数需要满足 $n=k+r,k=(r+1)\log_r^s$。Wang 等证明这种修复码的生成矩阵需要满足一定的子空间属性，基于这种子空间属性，既可以确定生成矩阵实现编码，也可以在某个系统节点失效时，计算从剩余的$(n-1)$个节点请求的数据，从而完成对某个系统节点的修复。这种构造方法具有较高的参数灵活性，而且实现了最小存储功能，但其缺陷是基于子空间的计算复杂度高，而且无法修复多重节点失效。

Papailiopoulos 等[18]提出了一种基于 MDS 码的轻量的再生码策略。从功能上看，(n,k)-MDS 码是一种删除码策略，其假设原始数据包含 k 个数据块，通过线性编码后得到 n 个编码码字或码块，从这 n 个码块中任意取出 k 个即可恢复出原始的 k 个数据块；从数学上看，MDS 码编码矩阵的任意 k 列都是线性无关的。文献[18]的编码策略把 $2k$ 个源数据符号 $\beta_{11},\cdots,\beta_{1k}$ 和 $\beta_{21},\cdots,\beta_{2k}$ 分别通过两个(n,k)-MDS 编码器进行编码(见图 3)，并把编码后输出的码字 $c_1,\cdots,c_n$ 和 $d_1,\cdots,d_n$，以及它们的和 $s_i=c_i+d_i(i=1,\cdots,n)$分别存储在 n 个存储节点中，其排列顺序如图 4 所示。在该系统中，如果有一个节点失效，存储于其中的 3 个数据字符可以通过调取其他节点存储的数据字符来恢复。例如，假设节点 1 失效，丢失的 3 个符号 c_1,d_2 和 s_3 可以通过符号 d_1,s_1,c_2,s_2,c_3,d_3 计算得到。如果把图 4 看作一个 3 行 n 列的符号阵列，则这个策略的一个明显优势在于针对每个源数据符号的编译码操作都被限制在同一行内，也就是行内编码。与这种方式相对应的是文献[1-2]采用的行间编码方式，即来自多个行的符号被混合纠缠在一起参与编码。从这个意义上讲，文献[18]中的方法比文献[1-2]中的方法更具简单性和易操作性。

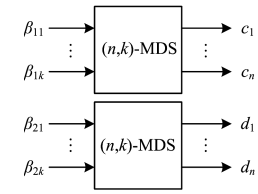


图 3　文献[18]的编码策略使用了两个 MDS 编码器

Fig. 3　Two MDS encoders used by coding strategy of ref. [18]

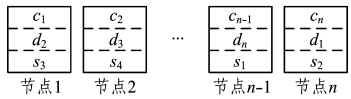


图 4　文献[18]的编码策略中的符号排列

Fig. 4　Arrangement of code symbols in coding strategy of ref. [18]

然而，仔细观察图 3 和图 4 可以发现，文献[18]的编码策略仍存在一些问题：1)图 3 所示的 MDS 编码器并不是系统形式的，因此其译码过程更复杂；2)图 4 所示的 3 行 n 列的符号阵列较标准的 MDS 编码器的输出多占用了 n 个符号的存储空间，即存储 $2k$ 个源符号需要 $3n$ 个码符号的存储空间，因此其码率等于$(2k/3n)$；3)假设图 4 中某个节点失效，为了重建该节点上存储的 3 个符号，需要调用其他节点的 6 个符号，也就是说修复带宽为 6 个符号(或者等价地，归一化修复带宽为 3 个符号)，这一带宽大于文献[1-2]所给出的再生码最小修复带宽；4)对应于单个源数据符号的归一化更新带宽为 $2n$ 个码符号，因此该系统的文件更新过程非常复杂；5)该系统的修复功能只能承受最多 1 个节点的失效，如果多于一个节点失效，则丢失的数据可能无法恢复，整个系统将崩溃。为了解决文献[18]存在的上述问题，同时保留其行内编码简单、轻量的优势，本文提出了一种新的基于网络编码的云存储策略。

2　基于网络编码的云存储系统

本文提出的系统对图 3 加以扩展，采用 m 个(n,k)-MDS

编码器。具体地，假设源数据由取自伽罗华域 GF(q)的 mk 个源数据符号构成，把这些源数据符号分成 m 组，每组包含 k 个符号，记为 $\beta_{i1},\cdots,\beta_{ik}$($i=1,\cdots,m$)。把每组源数据符号都送入一个($n,k$)系统形式的 MDS 编码器，以第 i 组为例，编码器的输入即 $\beta_{i1},\cdots,\beta_{ik}$；编码器的输出由两部分构成，第一部分是码字的系统部分，即 k 个源数据符号 $\beta_{i1},\cdots,\beta_{ik}$，第二部分是码字的非系统部分，包括($n-k$)个校验符号，记为 $c_{i1},\cdots,c_{i(n-k)}$。根据 MDS 编码器的编码规则，每个校验符号都由源符号的线性组合产生，即：

$$c_{ij}=\sum_{h=1}^{k} l_{ijh}\beta_{ih} \tag{1}$$

其中，β_{ih}($1\leqslant i\leqslant m,1\leqslant h\leqslant k$)表示第 i 个 MDS 编码器的第 h 个输入的源符号；c_{ij}($1\leqslant j\leqslant n-k$)表示第 i 个 MDS 编码器的第 j 个输出的校验符号；l_{ijh} 表示编码系数。第 i 个编码器的生成矩阵如下：

$$G_i=\begin{bmatrix} 1 & \cdots & 0 & l_{i11} & \cdots & l_{i(n-k)1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & l_{i1k} & \cdots & l_{i(n-k)k} \end{bmatrix} \tag{2}$$

G_i 的任意 k 列都是线性无关的，这种码属于 Reed-Solomon 码，其特点是根据其输出码字中任意 k 个码符号都可以恢复全部源数据符号 $\beta_{i1},\cdots,\beta_{ik}$。第 i 个编码器的矩阵计算如下：

$$(\beta_{i1},\cdots,\beta_{ik},c_{i1},\cdots,c_{i(n-k)})=(\beta_{i1},\cdots,\beta_{ik})\begin{bmatrix} 1 & \cdots & 0 & l_{i11} & \cdots & l_{i(n-k)1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & l_{i1k} & \cdots & l_{i(n-k)k} \end{bmatrix} \tag{3}$$

按照式(3)为每一组源数据符号编码完成后，将全部 m 个编码器输出的码符号按照图 5 所示的阵列结构排列，即得本系统最终的 n 个存储节点，以下简称图 5 所示的阵列结构为(n,k,m)码。

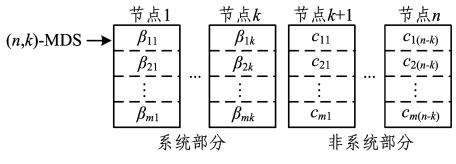


图 5 (n,k,m)码的阵列结构

Fig. 5 Array structure of (n,k,m) code

需要强调的是，图 5 中没有使用符号的行间交叉引用，每个源数据符号 β_{ij} 的编码都被限制在该符号所位于的行中，没有被其他编码器所使用。这样的系统结构具有两方面的优势：1)这种行内编码方式大幅降低了编译码的复杂度；2)图 5 所示的存储系统的文件更新过程更为简单和轻量，针对某个源符号 β_{ij} 的改变，只需要更新第 i 个编码器输出符号中的 β_{ij} 和($n-k$)个校验符号$\{c_{i1},\cdots,c_{i(n-k)}\}$即可，系统中其他符号都保持不变，因此其对应于单个源符号的更新带宽为($n-k+1$)个符号。最后，需要说明的是在本文的策略中，由于编码系数保持不变，因此文献[4-5]提出的减小更新带宽的方法依然可用。

接下来，考查图 5 所示的系统修复问题，可以将看到，基于网络编码允许在中间节点处对存储的数据进行线性运算的特点，图 5 所示的(n,k,m)码系统的修复带宽低于图 4 所示的系统。根据 MDS 码的特点，任何少于或等于($n-k$)个符号的丢失都可以通过从剩余的存活符号中任意选择 k 个符号加以恢复，以下简称这种方法为常规修复方法，以区别基于干扰对齐的修复方法。应用常规修复方法，图 5 所示的系统最多能承受($n-k$)个节点的失效，只要还有 k 个节点存活，整个系统就是可恢复的，而图 4 所示的系统最多只能修复单个节点失效的情况。

在应用常规方法修复多个节点失效时，图 5 的修复带宽等于 k 个节点，或等效的 km 个符号。然而在实际应用中，最常发生的是单节点失效的情况，因此有必要重点讨论图 5 所示的系统中单节点失效的修复问题。在单节点失效的情况下，利用系统 MDS 码的属性和干扰对齐技术，可以进一步降低图 5 所示的云存储系统的修复带宽。

首先，考虑图 5 中某个系统节点(即节点 1 至节点 k)的失效问题，不失一般性，假设节点 1 失效，符号$\{\beta_{11},\cdots,\beta_{m1}\}$丢失。为了恢复这些符号，取出节点 2 至节点 $k-1$ 上存储的全部源数据符号，并用 Λ 表示这组符号集合：

$$\Lambda=\{(\beta_{12},\cdots,\beta_{m2}),\cdots,(\beta_{1(k-1)},\cdots,\beta_{m(k-1)})\} \tag{4}$$

此外，对节点 k 中存储的 m 个符号$\{\beta_{1k},\cdots,\beta_{mk}\}$取和，可以得到一个“和”符号 θ：

$$\theta=\sum_{i=1}^{m}\beta_{ik} \tag{5}$$

利用节点 $k+1$ 至节点 n 这些非系统节点上存储的校验符号 c_{ij}($1\leqslant i\leqslant m,1\leqslant j\leqslant n-k$)，以及 Λ,θ 和$\{\beta_{11},\cdots,\beta_{m1}\}$，可以构造出一组由 $n-k$ 个线性方程构成的方程组：

$$\begin{cases} \sum_{i=1}^{m}\alpha_{i1}c_{i1}=f_1(\beta_{11},\cdots,\beta_{m1},\Lambda,\theta) \\ \quad\vdots \\ \sum_{i=1}^{m}\alpha_{i(n-k)}c_{i(n-k)}=f_{n-k}(\beta_{11},\cdots,\beta_{m1},\Lambda,\theta) \end{cases} \tag{6}$$

其中，α_{ij}($1\leqslant i\leqslant m,1\leqslant j\leqslant n-k$)表示线性方程组中为校验符号 c_{ij} 分配的乘法系数，该系数的分配策略是使$\{\beta_{1k},\cdots,\beta_{mk}\}$能够被合并得到 θ。对应于($n-k$)个非系统节点，方程组(6)一共包括($n-k$)个线性方程，其中只有符号$\{\beta_{11},\cdots,\beta_{m1}\}$未知，因此当 $m\leqslant n-k$ 时，通过求解式(6)即可解析出丢失的符号$\{\beta_{11},\cdots,\beta_{m1}\}$。在上述修复过程中，需要从剩余的存储节点请求传输的符号，包括 Λ 中的 $m(k-2)$个符号、和符号 θ 以及式(6)中的($n-k$)个和符号，因此总的修复带宽等于 $1+k(m-1)-2m+n$ 个符号。

其次，考虑图 5 中某个非系统节点(即节点 $k+1$ 至节点 n)的失效问题。根据 Reed-Solomon 码的特点，对于任意一个(n,k)-MDS 编码器的输出符号，即图 5 中任意一行符号，其中任意一个符号都可以唯一地由剩余的 $n-1$ 个符号中的 k 个符号所确定，且无论这些符号是位于系统部分还是非系统部分。因此，我们可以把图 5 中任意的 k 个节点看作是系统部分，剩余的($n-k$)个节点看作是非系统部分。换句话说，就修复问题而言，系统部分和非系统部分是可以互换的，因此上面提到的干扰对齐方法对于单个非系统节点的失效修复依然是可用的。

本节最后重新考查式(6)，有两点需要加以说明：1)当 $m=n-k$ 时，式(6)中全部 $n-k$ 个线性方程必须线性无关，否

则不足以求解 m 个未知变量，为此，充当符号字母集的有限域 GF(q)要足够大才行；2)通过式(6)能够求解丢失的源符号的条件是 $m\leqslant n-k$，这就给我们一个提示，对于如图 6 所示的(7,3,2)码，当两个节点失效时，基于干扰对齐的修复方法也是有效的，而且此时修复带宽等于 5 个符号，即从节点 3 至节点 7 提取出的 5 个和符号，低于常规修复方法所需要的 $km=6$ 个符号，但图 6 中的码在提高修复性能的同时不可避免地带来了码率上的损失。

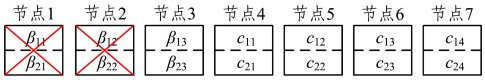


图 6　(7,3,2)码的阵列结构

Fig. 6　Array structure of (7,3,2) code

3　样例与比较

本节给出一个(n,k,m)码的具体样例，该样例来源于实验室环境下的一个小规模的云存储演示系统，其目的主要是解释应用干扰对齐方法修复单节点失效的操作过程，并把本文提出的(n,k,m)码与文献[18]的方法进行性能比较。该系统如图 7 所示，这是一个定义在伽罗华域 GF(7)上的(5,3,2)码的阵列结构，其中两个系统形式的(5,3)-MDS 码的编码矩阵分别为：

$$G_1=\begin{pmatrix}1&0&0&1&1\\0&1&0&1&2\\0&0&1&1&3\end{pmatrix},G_2=\begin{pmatrix}1&0&0&1&1\\0&1&0&1&3\\0&0&1&3&4\end{pmatrix}$$

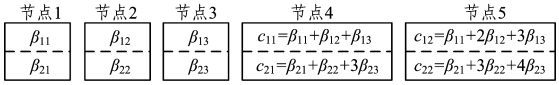


图 7　一个基于(5,3,2)码的云存储系统

Fig. 7　Cloud storage system based on (5,3,2) code

该系统为了保存 6 个原始数据符号，占用了 10 个码符号的存储空间，而文献[18]所提出的图 4 中一个采用(5,3)-MDS 码的系统需要 15 个符号的存储空间。

下面考查系统的修复问题，不难发现，在图 7 中，当发生 2 个节点失效而导致 4 个符号丢失时，应用常规方法调用剩余 3 个存活节点中存储的 6 个符号并求解线性方程组，系统能够修复最多 2 个节点的失效问题，此时对应的修复带宽为 6 个符号。而文献[18]的系统中，当出现 2 个节点失效时，系统是无法修复的。

此外，当发生一个系统节点失效时，以节点 1 失效为例，β_{11} 和 β_{21} 丢失，根据前述方法，我们有 $\Lambda=\{\beta_{12},\beta_{22}\}$，$\theta=\beta_{13}+\beta_{23}$，并可以构造出：

$$\begin{cases}f_1=3c_{11}+c_{21}=3\beta_{11}+\beta_{21}+3\beta_{12}+\beta_{22}+3\theta\\f_2=4c_{12}+3c_{22}=4\beta_{11}+3\beta_{21}+\beta_{12}+2\beta_{22}+5\theta\end{cases}\tag{7}$$

在式(7)中，β_{12} 和 β_{22} 是已知的，θ，f_1 和 f_2 可以分别在存储节点 3,4,5 处通过计算得到并传输给新建节点。因此，通过求解式(7)，可以解析出 β_{11} 和 β_{21} 的值，从而重建节点 1。

在图 7 中，当一个非系统节点失效时，例如节点 4 失效时，c_{11} 和 c_{21} 丢失。通过简单的线性运算即可把节点 4 和节点 3 进行置换(见图 8)。采用干扰对齐的方法可得 $\Lambda=\{\beta_{11},\beta_{21}\}$，$\theta=\beta_{12}+\beta_{22}$，以及

$$\begin{cases}f_1=\beta_{13}+3\beta_{23}=c_{11}+c_{21}+6\beta_{11}+6\beta_{21}+6\theta\\f_2=2c_{12}+3c_{22}=6c_{11}+c_{21}+3\beta_{11}+6\beta_{21}+5\theta\end{cases}\tag{8}$$

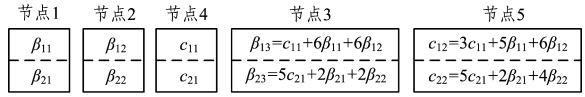


图 8　图 7 系统中节点 3 和节点 4 的置换

Fig. 8　Exchange roles of node 3 and node 4 in Fig. 7

由此，根据符号 β_{11}，β_{21}，θ，f_1 和 f_2 的值，通过求解式(8)，就可以恢复 c_{11} 和 c_{21} 的值，从而重建节点 4，因此这种干扰对齐的方法对于修复单个系统节点和非系统节点是等效的。最后，在上述单节点失效的修复过程中，系统的修复带宽为 5 个符号，即 Λ 中的两个符号、θ、f_1 以及 f_2。作为比较，图 4 所示的系统中当发生单个节点失效时，其修复带宽为 6 个符号。

接下来比较图 7 和图 4 所示系统的更新性能。当发生单个符号的更新时，例如当 β_{11} 发生变化时，图 7 中只需要修改或更新 β_{11}，c_{11}，c_{12} 这 3 个符号，其他符号不变，因此图 7 系统中单符号更新对应的归一化更新带宽为 3 个符号，而图 4 所示的系统的更新带宽为 10 个符号。

由于图 4 所示的阵列只使用了 2 行存储原始数据，因此将本文提出的(n,k,$m=2$)码与文献[18]的编码策略进行性能比较，结果列于表 1，同时以参数(n,k)=(5,3)为例，计算具体参数并列于表 1 中。可见在全部性能指标上，本文提出的系统都明显优于文献[18]中的系统。

表 1　文献[18]和(n,k,2)码的指标参数的比较

Table 1　Performance comparison between ref. [18] and (n,k,2) code

	文献[18]		(n,k,2)码	
	一般(n,k)码	(5,3)码	一般(n,k,2)码	(5,3,2)码
存储空间	$3n$ 个符号	15 个符号	$2n$ 个符号	10 个符号
码率	$2k/3n$	2/5	k/n	3/5
单节点失效的修复带宽	6 个符号	6 个符号	$n+k-3$ 个符号	5 个符号
归一化修复带宽	3 个符号	3 个符号	$(n+k-3)/2$ 个符号	2.5 个符号
两节点失效的修复带宽	不可修复	不可修复	$2k$ 个符号	6 个符号
单符号更新的归一化更新带宽	$2n$ 个符号	10 个符号	$n-k+1$ 个符号	3 个符号

文献[1]使用(n,k,d,α,β,M)六元组来描述再生码。其中，M 表示原始数据文件大小，这些源数据被编码成码数据存储在 n 个节点中，每个节点存储 α 个字符，依靠 n 个节点中任意 k 个节点可以恢复整个文件，当某个节点失效时，系统向 d 个帮助节点请求数据恢复丢失的存储数据，每个帮助节点提供 β 个字符。由该六元组定义的再生码的存储空间为 αn，单节点失效的修复带宽为 βd，另外，参数为 $\alpha=M/k$ 的码是最小存储码。文献[16]提出了一种再生码，其六元组参数为：

$$\left(n,k,d=k,\alpha=\binom{k}{i},\beta=\binom{k-1}{i-1},M=i\binom{k-1}{i-1}\right)\tag{9}$$

其中，i 取介于 1 和 k 之间的整数值。该码采用两阶段编码过

程:第一阶段为预编码,生成校验字符并把原始数据字符和校验字符排列在一个数据阵列中;第二阶段将该数据阵列与一个生成矩阵相乘,这种编码乘方式的复杂度较高,参数配置灵活性较差,而且不是系统码。文献[17]提出了一种系统形式的最小存储再生码,以 α 和 r 分别表示每个节点存储的符号数(即列长)和校验节点的个数,对于任意的 α 和 r,该码的六元组参数为:

$$(n=k+r,k=(r+1)\log_r,d=n-1,\alpha,\beta=\alpha/r,M=\alpha k) \tag{10}$$

文献[18]对应的参数为 $(n,k,4,3,\beta,2k)$,对应不同的帮助节点,β 可能取 1 或 2。本文提出的 (n,k,m) 码对应的六元组参数为 (n,k,d,m,β,mk),其中 d 和 β 根据修复方式(常规方式和干扰对齐)和帮助节点的不同可以取不同值。

下面比较分析这 4 种码的性能。首先,从存储和码率上看,4 种码占用的存储空间和码率分别为 αn 和 $M/\alpha n$。其次,文献[16-17]的单节点修复带宽都为 βd,文献[18]恒定为 6 个符号,(n,k,m) 码采用干扰对齐时单节点修复带宽等于 $1+k(m-1)-2m+n$ 个符号。最后,从更新性能上看,由于文献[16]采用的是非系统码形式,任何一个原始符号的更新都会造成全部码符号的改变,并且其码字阵列的维数是 $(n*\alpha)$,因此文献[16]的更新带宽为 αn;文献[17]采用了系统码的形式,某个源符号的更新只会带来校验符号的变化,但需要说明的是文献[17]使用了行间引用,即某一行的校验符号不仅取决于该行的源符号,还受其他行源符号的影响,因此其单符号更新带宽等于 $1+(n-k)\alpha$;文献[18]和 (n,k,m) 码的更新带宽则分别等于 $2n$ 和 $n-k+1$ 个符号,可见本文提出的 (n,k,m) 码的更新性能是最优的。

基于上面的分析,使这些码基于同一参数 $(n,k)=(5,3)$,将其六元组、码率、存储空间、修复带宽和更新带宽列于表 2。可见,文献[17]和本文提出的 (n,k,m) 码具有明显优势,这两种码不仅实现了最小存储码的功能,而且都是系统码的形式。此外,由于文献[16]严格规定了文件大小、节点存储的符号数和在修复过程中帮助节点传输的符号数,因此参数灵活性较差。文献[17]通过调整 α 和 r 的取值可以实现较灵活的参数配置,但文献[17]的单节点修复过程必须从全部剩余的 $n-1$ 个节点请求数据,因此无法修复多节点失效,这是一个缺点。最后,从编码的复杂度上看,文献[16]的两阶段编码过程和文献[17]的基于子空间的编码和修复过程的复杂度高于文献[18]和本文提出的 (n,k,m) 码。综上所述,对文献[16-18]和本文提出的 (n,k,m) 码的性能,从参数配置的灵活性、占用的存储空间、系统复杂度、码结构、修复带宽和更新带宽等几方面加以定性比较,结果如表 3 所列。

表 2 基于参数 $(n,k)=(5,3)$ 的 4 种存储码的性能比较

Table 2 Performance comparison among four storage codes based on $(n,k)=(5,3)$

	(n,k,d,α,β,M)	码率 $M/\alpha n$	存储空间 αn	单节点修复带宽	单符号更新带宽
文献[16]	(5,3,3,3,2,8)	8/15	15	6	15
文献[17]	(5,3,4,2,1,6)	3/5	10	4	5
文献[18]	(5,3,4,3,β,6)	2/5	15	6	10
(n,k,m)	(5,3,d,2,β,6)	3/5	10	5	3

表 3 4 种存储码的性能比较

Table 3 Performance comparison among four storage codes

	参数灵活性	存储空间	复杂度	码结构形式	修复带宽	更新带宽
文献[16]	差	差	高	非系统形式	高	高
文献[17]	好	最小存储码	高	系统形式	低	较高
文献[18]	好	差	低	非系统形式	高	高
(n,k,m)	好	最小存储码	低	系统形式	低	低

结束语 本文提出了一种基于网络编码的云存储系统架构。该系统的关键点在于使用了系统的 (n,k)-MDS 码,此外由于采用行内编码而非行间编码策略,因此该系统保留了文献[18]的简单性,与文献[1]的编码策略相比,编译码的复杂度大幅降低。此外,通过详细的比较分析,基于(5,3,2)码的样例系统证明了相比于文献[18],所提系统在存储空间、更新带宽和修复带宽等性能参数指标上更优。尤其是在单节点失效的情况下,该系统由于可以使用干扰对齐方法,因此进一步减少了单节点失效所需要的修复带宽。表 2 和表 3 对 4 种存储码的分析和比较说明了本文提出的 (n,k,m) 码具有较好的系统性能。但正如前面所提到的,不存在一个在全部性能指标上都最优的存储码,我们只能根据实际需求在各个性能度量上取折衷,同时还要兼顾系统的复杂性。未来的研究工作可以从以下几个方向加以展开:1)Dimakis 等[1]虽然指出了存储和修复带宽之间存在折衷关系,但是在具体的参数设置下,如何设计再生码实现功能修复或精确修复还是一个开放问题,目前只有几种参数配置被解决[19-20],而且现有的再生码系统的参数配置灵活性差,约束条件多,因此有必要研究具有较高参数灵活性的再生码设计方案;2)与单节点修复问题相比,云存储系统的多重修复问题[21-23]更具挑战性,本文所提出的 (n,k,m) 码虽然可以用于多重修复,但是它是以码率损失为代价的,如何设计更高效的多重修复系统是一个很有价值的方向;3)把云存储系统与网络安全问题相结合,在被动攻击或主动攻击背景下讨论安全云存储系统的设计问题[24-26]。

参 考 文 献

[1] 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.

[2] DIMAKIS A G,RAMCHANDRAN K,WU Y,et al. A Survey on Network Codes for Distributed Storage [J]. Proceedings of the IEEE,2011,99(3):476-489.

[3] ACEDANSKI S,DEB S,MEDARD M,et al. How Good Is Random Linear Coding Based Distributed Networked Storage [C]// The 1st Workshop on Network Coding,Theory and Applications. Riva del Garda:IEEE Press,2005:1-6.

[4] ZAKERINASAB M R,WANG M. An Update Model for Network Coding in Cloud Storage Systems [C]// The 50th Annual Allerton Conference on Communication,Control,and Computing. Monticello:IEEE Press,2012:1158-1165.

[5] ZAKERINASAB M R,WANG M. DeltaNC:Efficient File Updates for Network-Coding-Based Cloud Storage Systems [C]// IEEE 21st International Symposium on Modelling,Analysis and Simulation of Computer and Telecommunication Systems. San Francisco:IEEE Press,2013:360-364.

(下转第 312 页)

[D]. Jinan:Shandong University,2014. (in Chinese)
常征. 基于 RNA 测序技术的转录组从头拼接算法研究[D]. 济南:山东大学,2014.

[14] ZHENG C,LI G,LIU J,et al. Bridger:a new framework for de novo transcriptome assembly using RNA-seq data [J]. Genome Biology,2015,16(1):30.

[15] XIONG X J. Introduction to NCBI's SRA database [J]. Chemistry of Life,2010(6):959-963. (in Chinese)
熊筱晶. NCBI 高通量测序数据库 SRA 介绍[J]. 生命的化学,2010(6):959-963.

(上接第 298 页)

[6] WANG L J,CHEN Y,YAN X C,et al. Differential Data Update Scheme on Network-Coding-Based Cloud Storage System [J]. Journal on Communications,2017,38(3):154-164. (in Chinese)
王龙江,陈越,严新成,等. 网络编码云存储系统差分数据更新方案[J]. 通信学报,2017,38(3):154-164.

[7] WU H,LAI C Z,FAN J L,et al. Data Update Algorithm Based on Secure Network Coding in Cloud Environment [J]. Journal on Communications,2017,35(5):121-127. (in Chinese)
吴昊,赖成喆,范九伦,等. 云环境下基于安全网络编码的数据更新方法[J]. 通信学报,2017,35(5):121-127.

[8] WU Y,DIMAKIS A G. Reducing Repair Traffic for Erasure Coding-Based Storage via Interference Alignment [C]// 2009 IEEE International Symposium on Information Theory. Seoul:IEEE Press,2009:2276-2280.

[9] CADAMBE V R,JAFAR S A. Interference Alignment and the Degrees of Freedom for the K User Interference Channel [J]. IEEE Transactions on Information Theory,2008,54(8):3425-3441.

[10] HUANG Q. Efficient Repair Schemes in Cloud Storage Based on Interference Alignment [D]. Chongqing:Chongqing University of Posts and Telecommunications,2014. (in Chinese)
黄倩. 基于干扰对齐的高效云存储修复方法研究[D]. 重庆:重庆邮电大学,2014.

[11] 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.

[12] ZHANG J F. Cloud Storage Schemes Based on Network Coding [D]. Changsha:Central South University,2014. (in Chinese)
张俊峰. 基于网络编码云存储方案研究 [D]. 长沙:中南大学,2014.

[13] XIE C Y,JIA Z T,QING S H,et al. Semi-Random Linear Network Coding for Cloud Storage Redundancy [J]. Journal of Beijing University of Posts and Telecommunications,2013,36(3):30-34. (in Chinese)
谢垂益,贾忠田,卿斯汉,等. 适用于云存储冗余的半随机线性网络编码[J]. 北京邮电大学学报,2013,36(3):30-34.

[14] XU Y,HE Q,LUO Y. Optimal Repair for Distributed Storage Codes in Vehicular Networks [C]// 2016 IEEE 83rd Vehicular Technology Conference. Nanjing:IEEE Press,2016:1-5.

[15] SIPOS M,GAHM J,VENKAT N,et al. Erasure Coded Storage on A Changing Network:the Untold Story [C]// 2016 IEEE Global Communications Conference (GLOBECOM). Washington DC:IEEE Press,2016:1-6.

[16] ELYASI M,MOHAJER S. Determinant Coding:A Novel Framework for Exact-Repair Regenerating Codes [J]. IEEE Transactions on Information Theory,2016,62(12):6683-6697.

[17] WANG Z,TAMO I,BRUCK J. Explicit Minimum Storage Regenerating Codes [J]. IEEE Transactions on Information Theory,2016,62(8):4466-4479.

[18] PAPAILIOPOULOS D S,LUO J,DIMAKIS A G,et al. Simple Regenerating Codes:Network Coding for Cloud Storage [OL]. http://arXiv.org/pdf/1109.0264.pdf.

[19] PRAKASH N,KRISHNAN M N. The Storage-Repair-Bandwidth Trade off of Exact Repair Linear Regenerating Codes for the Case $d=k=n-1$ [C]// IEEE International Symposium on Information Theory (ISIT). Hong Kong:IEEE Press,2015:859-863.

[20] LI J,TANG X,XIANG W. A New Construction of $(k+2,k)$ Minimal Storage Regenerating Code over F_3 with Optimal Access Property for All Nodes [J]. IEEE Communications Letters,2016,20(7):1289-1292.

[21] WANG Z,TAMO I,BRUCK J. Optimal Rebuilding of Multiple Erasures in MDS Codes [J]. IEEE Transactions on Information Theory,2017,63(2):1084-1101.

[22] LI R,LIN J,LEE P P C. Enabling Concurrent Failure Recovery for Regenerating-Coding-Based Storage Systems:from Theory to Practice [J]. IEEE Transactions on Computers,2015,64(7):1898-1911.

[23] ZHANG H,LI H,SHUM K W,et al. Concurrent Regenerating Codes [J]. IET Communications,2017,11(3):362-369.

[24] KUMAR S,ROSNES E,AMAT A G I. Secure Repairable Fountain Codes [J]. IEEE Communications Letters,2016,20(8):1491-1494.

[25] AGARWAL A,MAZUMDAR A. Security in Locally Repairable Storage [J]. IEEE Transactions on Information Theory,2016,62(11):6204-6217.

[26] CHEN Y,WANG L,LIAO C. Eavesdropping Prevention for Network Coding Encrypted Cloud Storage Systems [J]. IEEE Transactions on Parallel and Distributed Systems,2016,27(8):2261-2273.