(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103688515 A
(43)申请公布日 2014.03.26

(21)申请号 201380001964.1
(22)申请日 2013.03.26
(85)PCT国际申请进入国家阶段日
2014.01.13
(86)PCT国际申请的申请数据
PCT/CN2013/073180 2013.03.26
(71)申请人 北京大学深圳研究生院
地址 518055 中国广东省深圳市南山区西丽镇丽水路深圳大学城北大校区
申请人 李挥
(72)发明人 李挥 侯韩旭 朱兵
(74)专利代理机构 深圳市科吉华烽知识产权事务所(普通合伙) 44248
代理人 胡玉
(51)Int.Cl.
H04L 29/08(2006.01)
G06F 11/14(2006.01)

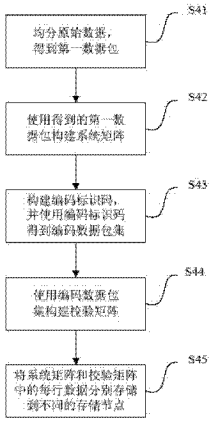
权利要求书3页 说明书15页 附图4页

(54)发明名称

一种最小带宽再生码的编码和存储节点修复方法

(57)摘要

本发明涉及一种最小带宽再生码的编码方法,包括如下步骤:将大小为B的原始数据平均分为k(k+1)/2个数据块,得到第一数据包;使用第一数据包构建尺寸为k×k的、对称的系统矩阵S;构建k个编码标识码,每个编码标识码包括k个元素;分别将所述系统矩阵的一列与编码标识码运算得到的编码数据包;分别选择n-k个不同的系统矩阵的列重复上述步骤,得到n-k个编码数据包集;以所述编码数据包集P_g的编码标识码编号g为其列号,构建大小为(n-k)×k的校验矩阵P;分别将所述系统矩阵和编码矩阵的每行存储到一个存储节点。本发明还涉及一种上述存储节点的修复方法。实施本发明的最小带宽再生码的编码和存储节点修复方法,具有以下有益效果:运算简单、开销小、修复带宽较小。



CN 103688515 A

1. 一种最小带宽再生码的编码方法，其特征在于，包括如下步骤：

A）将大小为 B 的原始数据平均分为 k（k+1）/2 个数据块，每个数据块大小为 L 比特，得到第一数据包；所述第一数据包表示为 $c_i=b_{i,1}b_{i,2}\ldots b_{i,L}$，i=1,2,...,k(k+1)/2；

B）使用所述第一数据包构建尺寸为 k×k 的、对称的系统矩阵 S；其中，按照其编号依次取得第一数据包，并将取得的第一数据包按照所述系统矩阵中元素所在列的顺序、逐行依次填入所述系统矩阵 S 的上三角中，得到所述系统矩阵 S 的上三角；

C）构建 k 个编码标识码，每个编码标识码包括 k 个元素；分别将所述系统矩阵的一列中的第一数据包按照一个标识编码中对应于该第一数据包编号的元素的值在该第一数据包的数据头或尾部加入设定数量的比特 0，得到 k 个第二数据包，运算所述 k 个第二数据包得到一个编码数据包；对所述系统矩阵中的该列使用不同的编码标识码重复上述步骤得到 k 个编码数据包；所述 k 个编码数据包按使用的编码数据包的编号排列而得到一个编码数据包集 $P_g=p_{g,1}p_{g,2}\ldots p_{g,k}$，其中，g=1,2,..,n-k，$p_{g,k}$ 是由所述第 g 个编码标识码和所述系统矩阵的第 k 列得到的编码数据包；分别选择 n-k 个不同的系统矩阵的列重复上述步骤，得到 n-k 个编码数据包集；

D）以所述编码数据包集 P_g 的编码标识码编号 g 为其列号，构建大小为 (n-k)×k 的校验矩阵 P；

E）分别将所述系统矩阵中的每一行包括的第一数据包存储到一个存储节点，得到 k 个系统节点；分别将所述校验矩阵中的每一行存储到一个存储节点，得到 n-k 个校验节点，所述 n 是存储节点总数。

2. 根据权利要求 1 所述的最小带宽再生码的编码方法，其特征在于，所述步骤 C）进一步包括如下步骤：

C1）得到 k 个编码标识码；

C2）取得一个编码标识码，选择所述系统矩阵的一列，对所选择的列的 k 个第一数据包分别依据该编码标识码中元素的最大值和该列中第一数据包所在行数对应的编码标识码元素值在该列第一数据包的数据头部或尾部分别添加设定数量的比特 0，得到 k 个第二数据包；对所述 k 个第二数据包进行运算，得到一个编码数据包；

C3）依次使用不同的编码标识码依次分别对所选择的系统矩阵的列重复步骤 C2），直到得到 n-k 个编码数据包；将得到的编码数据包依次排列得到一个编码数据包集；

C4）分别依次选择所述系统矩阵中 k 个不同的列并使用所述编码标识码重复步骤 C2）和 C3），得到 n-k 个编码数据包集。

3. 根据权利要求 2 所述的最小带宽再生码的编码方法，其特征在于，所述步骤 C1）进一步包括：

C11）判断 k 是否素数，如是，执行步骤 C12）；否则，执行步骤 C13）；

C12）按照 $(r_1^a,r_2^a,\ldots,r_k^a)=(0,a,2a,\ldots,(k-1)a)\bmod k$，a=1,2,...,n-k，分别将 a=1,2,...,n-k 带入数列 (0,a,2a,...,(k-1)a)，并对得到的数列中的元素分别取 k 的模，得到 n-k 个编码标识码；

C13）取大于 k 的最小素数 p，并按照 $(r_1^a,r_2^a,\ldots,r_k^a)=(a-1,2a-1,\ldots,ka-1)\bmod p$，a=1,2,...,n-k，分别将 a=1,2,...,n-k 带入数列 (a-1,2a-1,2a,...,ka-1)，并对得到的数列中的元素分别取 p 的模，得到 n-k 个编码标识码。

4. 根据权利要求 3 所述的最小带宽再生码的编码方法，其特征在于，所述步骤 C2）进一步包括：

C21）取得所述编码标识码中的最大值，即 $r_{max}=max(r_1^a, r_2^a, \ldots, r_k^a)$；

C22）在该系统矩阵被选择的列的第 y 个第一数据包的数据头部添加等于当前使用的编码标识码中第 y 个元素值的比特 0，而在该第一数据包的数据尾部添加 $r_{max}-r_y^a$ 个比特 0，得到一个第二数据包，其中，y=1, 2, . . . , k；依次分别对该列的 k-1 个第一数据包按照其在该列的行数取相同的 y 值并重复上述步骤，得到 k 个第二数据包；g 是被选择的系统矩阵的列，g 是 1, 2, . . , n-k 中的一个；

C23）将得到的 k 个第二数据包相加，得到由当前编码标识码产生的一个编码数据包 $p_{g,j}$，表示通过系统矩阵的第 g 列数据和第 j 编码标识码运算得到的编码数据包。

5. 根据权利要求 4 所述的最小带宽再生码的编码方法，其特征在于，所述步骤 C4）中进一步包括：

C41）选择所述步骤 C2）中编码标识码的相邻的下一个编码标识码；

C42）将所述取得的编码标识码作为当前使用的编码标识码，并重复步骤 C2）和 C3），直到所有的编码标识码均已使用。

6. 根据权利要求 5 所述的最小带宽再生码的编码方法，其特征在于，所述步骤 B）中进一步包括：

B1）将得到的第一数据包按照其编号取出，并按照所述系统矩阵 S 中元素所在列的顺序、逐行依次填入所述系统矩阵 S 的上三角部分，得到系统矩阵的上三角 $\begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ & c_{k+1} & \cdots & c_{2k-1} \\ & & \vdots & \vdots \\ & & & c_B \end{bmatrix}$；其中，B=k（k+1）/2；

B2）将上述步骤中得到上三角部分沿其对角线对折而得到该系统矩阵的下三角部分，系统矩阵表示为：$S=\begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_{k+1} & \cdots & c_{2k-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_k & c_{2k-1} & \cdots & c_B \end{bmatrix}$；

所述校验矩阵表示为：$P=\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{n-k} \end{bmatrix}=\begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n-k,1} & p_{n-k,2} & \cdots & p_{n-k,k} \end{bmatrix}$；其中，$P_1$ 到 P_{n-k} 是上述步骤中每次重复步骤 C3）时分别得到的编码数据包集；所述步骤 D）中，还包括如下步骤：将所述系统矩阵和所述校验矩阵排列为一个数据矩阵，并将该数据矩阵的每一行分别存储在各存储节点中；所述数据矩阵表示为：$M=\begin{bmatrix} S \\ P \end{bmatrix}$。

7. 一种修复如权利要求 1 所述的编码方法中存储节点的存储节点修复方法，其特征在

于，包括如下步骤：

I）确认存储节点失效，判断失效存储节点的类型是否系统节点，如是，执行下一步骤；否则，执行步骤 K）；

J）由剩余的每个正常的系统节点中下载该存储节点存储的第 f 个数据，即该系统节点位于系统矩阵第 f 列的数据，得到该失效节点中存储的 k-1 个数据，所述 f 是失效的系统节点位于系统矩阵的行数；f=1,2,...,k；选择该列数据对应的校验节点下载其存储的数据，使用所述由校验节点下载的数据和编码标识码运算，并结合所得到所述失效系统节点中存储的系统矩阵中一列的数据；得到所述失效系统节点中的全部数据；将得到的数据存储在新的存储节点并使存储节点取代失效的系统节点；

K）取得产生失效校验节点所存储编码数据包集对应的系统矩阵的列号，由所有系统节点中分别下载一个数据，所述下载的数据是系统矩阵中一个完整的、对应于所述取得列号的列；使用所有编码标识码对所述下载的数据进行编码，得到所述失效校验节点存储的数据，将其存储到新的存储节点并使其取代失效的校验节点。

8. 根据权利要求 7 所述的存储节点修复方法，其特征在于，所述步骤 J）进一步包括：

J1）取得失效系统节点在系统矩阵中的行数 f，对于剩余正常的每个系统节点，分别下载其位于系统矩阵的第 f 列的第一数据包；

J2）选择存储由所述系统矩阵第 f 列产生编码数据包集的校验节点下载其存储的编码数据包，使用下载的编码数据包和编码标识码进行编码运算的逆运算，得到所述系统矩阵第 f 列的第一数据包；

J3）由所述系统矩阵的行、列间的对应关系得到所述失效的系统节点存储的第一数据包。

9. 根据权利要求 8 所述的存储节点修复方法，其特征在于，所述步骤 K）进一步包括：

K1）确定失效的校验节点在编码矩阵中行数 e，e=1,2,..,n-k；取得 k 个编码标识码；

K2）分别下载 k 个系统节点的第 e 个第一数据包，得到所述系统矩阵的第 e 列数据；取得编码标识码中的最大值，即 $r_{max}=max(r_1^a, r_2^a, ..., r_n^a)$；

K3）对于得到的系统矩阵第 e 列数据，分别使用取得的编码标识码对其进行编码处理，得到存储在所述失效节点的数据。

10. 根据权利要求 9 所述的存储节点修复方法，其特征在于，所述步骤 K3）进一步包括：

K31）取得所述编码标识码中的最大值，即 $r_{max}=max(r_1^a, r_2^a, ..., r_k^a)$；

K32）在该系统矩阵被选择的列的第 y 个第一数据包的数据头部添加等于当前使用的编码标识码中第 y 个元素值的比特 0，而在该第一数据包的数据尾部添加 $r_{max}-r_y^a$ 个比特 0，得到一个第二数据包，其中，y=1,2,...,k；依次分别对该列的 k-1 个第一数据包按照其在该列的行数取相同的 y 值并重复上述步骤，得到 k 个第二数据包；g 是被选择的系统矩阵的列，g 是 1,2,..,n-k 中的一个；

K33）将得到的 k 个第二数据包相加，得到由当前编码标识码产生的一个编码数据包 $p_{g,j}$，表示通过系统矩阵的第 g 列数据和第 j 编码标识码运算得到的编码数据包。

一种最小带宽再生码的编码和存储节点修复方法

技术领域

[0001] 本发明涉及分布式存储领域，更具体地说，涉及一种最小带宽再生码的编码和存储节点修复方法。

背景技术

[0002] 随着计算机网络应用的迅速发展，网络信息数据量变得越来越大，海量信息存储变得尤为重要。传统意义的文件存储系统已经不能满足现有应用的大容量、高可靠性、高性能等方面的要求，分布式存储系统以其高效的可扩展性和高可用性成为存储海量数据的有效系统。然而在分布式存储系统中，存储数据的节点是不可靠的。为了能够由不可靠的存储节点提供可靠的存储服务，需要在存储系统中引入冗余。引入冗余最简单的方法就是对原始数据直接备份，直接备份虽然简单但是其存储效率和系统可靠性不高，而通过编码引入冗余的方法可以提高其存储效率。在目前的存储系统中，编码方法一般采用 MDS 码(Maximum Distance Separable 最大距离可分离)，MDS 码可以达到存储空间效率的最佳，一个 (n,k)MDS 纠错码需要将一个原始文件分成 k 个大小相等的模块，并通过线性编码生成 n 个互不相关的编码模块，由 n 个节点存储不同的模块，并满足 MDS 属性(n 个编码模块中任意 k 个就可重构原始文件)。这种编码技术在提供有效的网络存储冗余中占有重要的地位，特别适合存储大的文件以及档案数据备份应用。

[0003] 在分布式存储系统中，把大小为 B 的数据存储在 n 个存储节点中，每个存储节点存储的数据大小为 α 。数据接收者只需要连接并下载 n 个存储节点中的任意 k 个存储节点的数据即可恢复出原始数据 B，这一过程称为数据重建过程。RS (Reed - Solomon 里德 - 所罗门)码是满足 MDS 码特性的一种码字。当存储系统中的存储节点失效时，为了保持存储系统的冗余量，需要恢复该失效节点存储的数据并将该数据存储在新节点中，该过程称为修复过程。然而，在修复过程中，RS 码首先需要下载 k 个存储节点的数据并恢复出原始数据，之后为新节点编码出失效节点的存储数据。为了恢复一个存储节点的数据而解码出整个原始数据显然对传输带宽是一种浪费。

[0004] 然而，系统节点失效或者文件损耗，系统的冗余度会随着时间而逐渐减小，因此需要一种机制来保证系统的冗余。文献 [R.Rodrigues and B.Liskov, “High Availability in DHTs:Erasure Coding vs.Replication”,Workshop on Peer-to-Peer Systems(IPTPS)2005.] 中提出的 EC 码(Erasure Codes 纠错码)，该码在存储开销上是比较有效的，然而支持冗余恢复所需要的通信开销也比较大。图 1 表示只要系统中有效节点数 d ≥ k，就可以从现有节点中获得原始文件；图 2 表示恢复失效节点所存储内容的过程。从图 1 和图 2 中可以看出整个恢复过程是：1) 首先从系统中的 k 个存储节点中下载数据并重构原始文件；2) 由原始文件再重新编码出新的模块，存储在新节点上。该恢复过程表明修复任何一个失效节点所需要的网络负载至少为 k 个节点所存储的内容。

[0005] 同时，为了降低修复过程中所使用的带宽，文献 [A.G.Dimakis, P.G.Godfrey, M.J.Wainwright,K.Ramchandran,“Network coding for distributed storage systems”,

IEEE Proc. INFOCOM, Anchorage, Alaska, May2007.] 利用网络编码理论的思想提出了再生码 (RGC,Regenerating Codes),RGC 码也满足 MDS 码特性。再生码的修复过程中,新节点需要在剩下的存储节点中连接 d 个存储节点并分别从这 d 个存储节点中下载 β 大小的数据,所以 RGC 码的修复带宽为 dβ。同时给出了 RGC 码功能修复的模型并提出了 RGC 码的两类最佳码 :最小带宽再生码 (MSR, Minimum - storage Regenerating) 和最小修复带宽再生码 (MBR, Minimum - bandwidth Regenerating)。RGC 码的修复带宽优于 RS 码,但 RGC 的修复过程需要连接 d(d>k) 个存储节点 (d 称为修复节点)。另外,修复节点需要对其存储的数据执行随机线性网络编码操作。为了满足所有编码包是相互独立的,RGC 码的运算需要在一个较大的有限域内。

[0006] 专利 PCT/CN2012/083174 中提出了一种实用射影自修复码的编码、数据重构及修复方法。实用射影自修复码(PPSRC,Practical Projective Self-repairing Codes)同样具有自修复码的两个典型属性 :丢失的编码模块可从其他编码模块中下载少于整个文件的数据进行修复 ;丢失的编码模块从一个给定数的模块中修复,该给定数只与丢失了多少模块数有关,而与具体哪些模块丢失无关。这些属性使得修复一个丢失模块的负载比较低,另外由于系统中各节点地位相同、负载均衡使得在网络的不同位置,可以独立并发地修复不同丢失模块。

[0007] 该码字除了满足以上条件外还有以下特性 :当一个节点失效时,可以有(n - 1)/2 对修复节点可供选择 ;当有(n - 1)/2 个节点同时失效时,我们仍然可以使用剩下的(n+1)/2 个节点中的 2 两个节点来修复失效节点。

[0008] PPSRC 码的编码以及自修复过程仅涉及异或运算,并不像一般自修复码,其编码需要计算多项式相对较复杂,PPSRC 码的计算复杂度小于 PSRC 码(Projective Self-repairing Codes 射影自修复码)。同时,PPSRC 码的修复带宽和修复节点优于 MSR 码。PPSRC 码的冗余是可控的,适用于一般的存储系统,PPSRC 码的重建带宽达到最佳。

[0009] 总而言之,PPSRC 码有效地减少了数据存储节点,降低了系统数据存储的冗余度,很大程度上提高了实用自修复码的使用价值。

[0010] 然而,PPSRC 码也存在一定的不足之处。首先,PPSRC 码的编解码过程较为复杂,有限域及其子域的划分运算量相对较大,并且数据重构过程比较繁琐 ;其次,在 PPSRC 码中,编码模块是不可再分的,因此修复编码模块也必须是不可再分的。同时,PPSRC 码的整个编解码过程运算复杂度较高,冗余量虽然可控但其实还是相当大的。通常 PPSRC 码存储节点数选取非常大,对于相对小一些的文件来说就显得完全没有必要了。这些均增加了 PPSRC 码在实际分布式存储系统中实施难度,该射影自修复码通用性不强。

[0011] 通过属性——当任意一个模块所存储的信息是由两个其他模块的信息异或而得到的情况下,任意两个模块信息就可用来修复第三个模块,在文献 [A. Duminuco, E. Biersack, “Hierarchical Codes:How to Make Erasure Codes Attractive for Peer-to-Peer Storage Systems”, Peer-to-Peer Computing(P2P), 2008.] 中提出了一种 HC 码(Hierarchical Codes 分层码)。HC 码是一种迭代构造,从小的 EC 码开始逐渐构成一个大的编码,通过异或由 EC 码构造的子模块而产生。

[0012] 其主要思想是 :考虑一个大小为 s×k 的文件,将文件分成 s 个子群,每个子群包含 k 个未编码模块。在每个子群中使用一个 (n,k)EC 码来产生 n-k 个局部冗余编码模块。通

过编码计划进一步由所有的 s×k 个未编码模块来产生 r 个全局冗余编码模块。因此形成一个编码群，将 s×k 个未编码模块编码成（s×n+r）个编码模块。局部冗余模块可以用来修复子群中节点的失效，因此只需要访问少于整个文件大小的模块就可以进行修复；而全局冗余模块提供进一步修复保证，即当一个子群中失效的模块太多而不能自修复时可通过全局冗余模块进行修复。由于 HC 码中系统结构不对称，使得有些模块的地位或许比其他模块地位要高，使得很难做一个深入的恢复力分析（影响对编码有效性的理解）；在实际系统中如果利用该编码则需要更复杂的算法（不管是重构还是修复）；在 HC 码中不同编码模块的地位不同，因而修复丢失的模块所需要的模块数不仅仅取决于丢失的模块数，还与具体哪些模块丢失有关；同样地，重构原始文件所需要的模块数可能也因不同的丢失模块而不同。

[0013] 专利 PCT/CN2012/071177 中提出了一种 RGC 码，该方案中修复一个丢失的编码模块只需要一小部分的数据量，而不需要重构整个文件。RGC 码应用线性网络编码思想，利用 NC（Network Coding）属性（即最大流最小割）来改善修复一个编码模块所需要的开销，从网络信息论上可以证明用和丢失模块相同数据量的网络开销就可修复丢失模块。

[0014] RGC 码主要思想还是利用 MDS 属性，当网络中一些存储节点失效，也就相当于存储数据丢失，需要从现有有效节点中下载信息来使得丢失的数据修复丢失的数据模块，并将其存储在新的节点上。随着时间的推移，很多原始节点可能都会失效，一些再生的新节点可以在自身再重新执行再生过程，继而生成更多的新节点。因此再生过程需要确保两点：1）失效的节点间是相互独立的，再生过程可以循环递推；2）任意 k 个节点就足够恢复原始文件。

[0015] 图 2 描述了当一个节点失效后的再生过程。分布式系统中 n 个存储节点各自存储 α 个数据，当有一个节点失效，新节点通过从其他 d ≥ k 个存活节点中下载数据并用于节点再生，每个节点的下载量为 β，每个存储节点 i 通过一对节点 X^i_{in}, X^i_{out} 来表示，这对节点通过一个容量为该节点的存储量（即 α）的边连接。再生过程通过一个信息流图描述，X_{in} 从系统中任意 d 个可用节点中各自收集 β 个数据，通过 $X_{in} \xrightarrow{\alpha} X_{out}$ 在 X_{out} 中存储 α 个数据，任何一个接收者都可以访问 X_{out}。从信源到信宿的最大信息流是由图中最小割集决定，当信宿要重构原始文件时，这个流的大小不能低于原始文件的大小。

[0016] 每个节点存储量 α 和再生一个节点所需要的带宽 γ 之间存在一个折中，因此又引入最小带宽再生码（MBR）和最小带宽再生码（MSR）。对于最小存储点可以知道每个节点至少存储 M/k 比特，因此可推出 MSR 码中 $(\alpha_{MSR}, \gamma_{MSR}) = (\frac{M}{k}, \frac{Md}{k(d-k+1)})$，当 d 取最大值即一个新来者同时和所有存活的 n-1 个节点通信时，修复带宽 γ_{MSR} 最小即 $\gamma^{\min}_{MSR} = \frac{M}{k} \cdot \frac{n-1}{n-k}$。而 MBR 码拥有最小修复带宽，可以推出当 d=n-1 时，获得最小修复负载 $(\alpha^{\min}_{MBR}, \gamma^{\min}_{MBR}) = (\frac{M}{k} \cdot \frac{2n-2}{2n-k-1}, \frac{M}{k} \cdot \frac{2n-2}{2n-k-1})$。

[0017] 对于节点失效修复问题，考虑了三种修复模型：精确修复：失效的模块需要正确构造，恢复的信息和丢失的一样（核心技术为干扰队列和 NC）；功能修复：新产生的模块可以包含不同于丢失节点的数据，只要修复的系统支持 MDS 码属性（核心技术为 NC）；系统部分精确修复：是介于精确修复和功能修复之间的一个混合修复模型，在这个混合模型中，对于

系统节点(存储未编码数据)要求必须精确恢复,即恢复的信息和失效节点所存储的信息一样,对于非系统节点(存储编码模块),则不需要精确修复,只需要功能修复使得恢复的信息能够满则 MDS 码属性(核心技术为干扰队列和 NC)。

[0018] 为了使 RGC 码运用到实际的分布式系统中,即使不是最优情况也至少需要从 k 个节点下载数据才能修复丢失模块,因此即使修复过程所需要的数据传输量比较低,RGC 码也需要高的协议负载和系统设计(NC 技术)复杂度来实现。另外 RGC 码中未考虑工程解决方法,如懒修复过程,因此不能避免临时失效所带来的修复负载。最后基于 NC 的 RGC 码的编解码实现所需要的计算开销比较大,比传统的 EC 码要高一个阶数。

发明内容

[0019] 本发明要解决的技术问题在于,针对现有技术的上述运算复杂、开销大、修复带宽较大的缺陷,提供一种运算简单、开销小、修复带宽较小的最小带宽再生码的编码和存储节点修复方法。

[0020] 本发明解决其技术问题所采用的技术方案是:构造一种最小带宽再生码的编码方法,包括如下步骤:

[0021] A)将大小为 B 的原始数据平均分为 k (k+1)/2 个数据块,每个数据块大小为 L 比特,得到第一数据包;所述第一数据包表示为 $c_i=b_{i,1}b_{i,2}...b_{i,L}$, i=1,2,...,k(k+1)/2;

[0022] B)使用所述第一数据包构建尺寸为 k×k 的、对称的系统矩阵 S;其中,按照其编号依次取得第一数据包,并将取得的第一数据包按照所述系统矩阵中元素所在列的顺序、逐行依次填入所述系统矩阵 S 的上三角中,得到所述系统矩阵 S 的上三角;

[0023] C)构建 k 个编码标识码,每个编码标识码包括 k 个元素;分别将所述系统矩阵的一列中的第一数据包按照一个标识编码中对应于该第一数据包编号的元素的值在该第一数据包的数据头或尾部加入设定数量的比特 0,得到 k 个第二数据包,运算所述 k 个第二数据包得到一个编码数据包;对所述系统矩阵中的该列使用不同的编码标识码重复上述步骤得到 k 个编码数据包;所述 k 个编码数据包按使用的编码数据包的编号排列而得到一个编码数据包集 $P_g=p_{g,1}p_{g,2}...p_{g,k}$,其中,g=1,2,..,n-k,$p_{g,k}$ 是由所述第 k 个编码标识码和所述系统矩阵的第 g 列得到的编码数据包;分别选择 n-k 个不同的系统矩阵的列重复上述步骤,得到 n-k 个编码数据包集;

[0024] D)构建大小为 (n-k)×k 的校验矩阵 P,所述校验矩阵 P 的各行为依次排列的所述编码数据包集 P_g;

[0025] E)分别将所述系统矩阵中的每一行包括的第一数据包存储到一个存储节点,得到 k 个系统节点;分别将所述校验矩阵中的每一行存储到一个存储节点,得到 n-k 个校验节点,所述 n 是存储节点总数。

[0026] 更进一步地,所述步骤 C)进一步包括如下步骤:

[0027] C1)得到 k 个编码标识码;

[0028] C2)取得一个编码标识码,选择所述系统矩阵的一列,对所选择的列的 k 个第一数据包分别依据该编码标识码中元素的最大值和该列中第一数据包所在行数对应的编码标识码元素值在该列第一数据包的数据头部或尾部分别添加设定数量的比特 0,得到 k 个第二数据包;对所述 k 个第二数据包进行运算,得到一个编码数据包;

[0029] C3)依次使用不同的编码标识码依次分别对所选择的系统矩阵的列重复步骤C2)，直到得到n-k个编码数据包；将得到的编码数据包依次排列得到一个编码数据包集；

[0030] C4)分别依次选择所述系统矩阵中k个不同的列并使用所述编码标识码重复步骤C2)和C3)，得到n-k个编码数据包集。

[0031] 更进一步地，所述步骤C1)进一步包括：

[0032] C11)判断k是否素数，如是，执行步骤C12)；否则，执行步骤C13)；

[0033] C12)按照$(r_1^a, r_2^a, \ldots, r_k^a)=(0, a, 2a, \ldots, (k-1)a) \bmod k$, a=1, 2, ..., n-k，分别将a=1, 2, ..., n-k带入数列(0, a, 2a, ..., (k-1)a)，并对得到的数列中的元素分别取k的模，得到n-k个编码标识码；

[0034] C13)取大于k的最小素数p，并按照$(r_1^a, r_2^a, \ldots, r_k^a)=(a-1, 2a-1, \ldots, ka-1) \bmod p$, a=1, 2, ..., n-k，分别将a=1, 2, ..., n-k带入数列(a-1, 2a-1, 2a, ..., ka-1)，并对得到的数列中的元素分别取p的模，得到n-k个编码标识码。

[0035] 更进一步地，所述步骤C2)进一步包括：

[0036] C21)取得所述编码标识码中的最大值，即$r_{max}=\max(r_1^a, r_2^a, \ldots, r_k^a)$；

[0037] C22)在该系统矩阵被选择的列的第y个第一数据包的数据头部添加等于当前使用的编码标识码中第y个元素值的比特0，而在该第一数据包的数据尾部添加$r_{max}-r_y^a$个比特0，得到一个第二数据包，其中，y=1, 2, ..., k；依次分别对该列的k-1个第一数据包按照其在该列的行数取相同的y值并重复上述步骤，得到k个第二数据包；g是被选择的系统矩阵的列，g是1, 2, .., n-k中的一个；

[0038] C23)将得到的k个第二数据包相加，得到由当前编码标识码产生的一个编码数据包$p_{g,j}$，表示通过系统矩阵的第g列数据和第j编码标识码运算得到的编码数据包。

[0039] 更进一步地，所述步骤C4)中进一步包括：

[0040] C41)选择所述步骤C2)中编码标识码的相邻的下一个编码标识码；

[0041] C42)将所述取得的编码标识码作为当前使用的编码标识码，并重复步骤C2)和C3)，直到所有的编码标识码均已使用。

[0042] 更进一步地，所述步骤B)中进一步包括：

[0043] B1)将得到的第一数据包按照其编号取出，并按照所述系统矩阵S中元素所在列的顺序、逐行依次填入所述系统矩阵S的上三角部分，得到系统矩阵的上三角

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ & c_{k+1} & \cdots & c_{2k-1} \\ & & \vdots & \vdots \\ & & & c_B \end{bmatrix}$$；其中，B=k（k+1）/2；

[0044] B2)将上述步骤中得到上三角部分沿其对角线对折而得到该系统矩阵的下三角部分，系统矩阵表示为：$S=\begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_{k+1} & \cdots & c_{2k-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_k & c_{2k-1} & \cdots & c_B \end{bmatrix}$；

[0045] 所述校验矩阵表示为：$P=\begin{bmatrix}P_1\\P_2\\\vdots\\P_{n-k}\end{bmatrix}=\begin{bmatrix}p_{1,1}&p_{1,2}&\cdots&p_{1,k}\\p_{2,1}&p_{2,2}&\cdots&p_{2,k}\\\vdots&\vdots&\vdots&\vdots\\p_{n-k,1}&p_{n-k,2}&\cdots&p_{n-k,k}\end{bmatrix}$；其中，$P_1$ 到 P_{n-k} 是上述步骤中每次重复步骤 C3）时分别得到的编码数据包集；所述步骤 D）中，还包括如下步骤：将所述系统矩阵和所述校验矩阵排列为一个数据矩阵，并将该数据矩阵的每一行分别存储在各存储节点中；所述数据矩阵表示为：$M=\begin{bmatrix}S\\P\end{bmatrix}$。

[0046] 本发明还涉及一种修复存储上述编码数据存储节点的方法，包括如下步骤：

[0047] I）确认存储节点失效，判断失效存储节点的类型是否系统节点，如是，执行下一步骤；否则，执行步骤 K）；

[0048] J）由剩余的每个正常的系统节点中下载该存储节点存储的第 f 个数据，即该系统节点位于系统矩阵第 f 列的数据，得到该失效节点中存储的 k-1 个数据，所述 f 是失效的系统节点位于系统矩阵的行数；f=1,2,...,k；选择该列数据对应的校验节点下载其存储的数据，使用所述由校验节点下载的数据和编码标识码运算，并结合所得到所述失效系统节点中存储的系统矩阵中一列的数据；得到所述失效系统节点中的全部数据；将得到的数据存储在新的存储节点并使存储节点取代失效的系统节点；

[0049] K）取得产生失效校验节点所存储编码数据包集对应的系统矩阵的列号，由所有系统节点中分别下载一个数据，所述下载的数据是系统矩阵中一个完整的、对应于所述取得列号的列；使用所有编码标识码对所述下载的数据进行编码，得到所述失效校验节点存储的数据，将其存储到新的存储节点并使其取代失效的校验节点。

[0050] 更进一步地，所述步骤 J）进一步包括：

[0051] J1）取得失效系统节点在系统矩阵中的行数 f，对于剩余正常的每个系统节点，分别下载其位于系统矩阵的第 f 列的第一数据包；

[0052] J2）选择存储由所述系统矩阵第 f 列产生编码数据包集的校验节点下载其存储的编码数据包，使用下载的编码数据包和编码标识码进行编码运算的逆运算，得到所述系统矩阵第 f 列的第一数据包；

[0053] J3）由所述系统矩阵的行、列间的对应关系得到所述失效的系统节点存储的第一数据包。

[0054] 更进一步地，所述步骤 K）进一步包括：

[0055] K1）确定失效的校验节点在编码矩阵中行数 e，e=1,2,..,n-k；取得 k 个编码标识码；

[0056] K2）分别下载 k 个系统节点的第 e 个第一数据包，得到所述系统矩阵的第 e 列数据；取得编码标识码中的最大值，即 $r_{max}=\max(r_1^a,r_2^a,...,r_n^a)$；

[0057] K3）对于得到的系统矩阵第 e 列数据，分别使用取得的编码标识码对其进行编码处理，得到存储在所述失效节点的数据。

[0058] 更进一步地，所述步骤 K3）进一步包括：

[0059] K31）取得所述编码标识码中的最大值，即 $r_{max}=\max(r_1^a,r_2^a,...,r_k^a)$；

[0060] K32）在该系统矩阵被选择的列的第 y 个第一数据包的数据头部添加等于当前使用的编码标识码中第 y 个元素值的比特 0，而在该第一数据包的数据尾部添加 $r_{max}-r_y^a$ 个比特 0，得到一个第二数据包，其中，y=1,2,...,k ；依次分别对该列的 k-1 个第一数据包按照其在该列的行数取相同的 y 值并重复上述步骤，得到 k 个第二数据包 ；g 是被选择的系统矩阵的列，g 是 1,2,..,n-k 中的一个 ；

[0061] K33）将得到的 k 个第二数据包相加，得到由当前编码标识码产生的一个编码数据包 $p_{g,j}$，表示通过系统矩阵的第 g 列数据和第 j 编码标识码运算得到的编码数据包。

[0062] 实施本发明的最小带宽再生码的编码和存储节点修复方法，具有以下有益效果 ：传统 RGC 码的构造基于有限域 GF(q)，编解码过程中设计到的有限域加法、减法以及乘法 ；有限域的运算虽然理论研究比较成熟，但实际运用起来比较繁琐、时间消耗大 ；而在本实施例中，由于减小了编解码过程中计算复杂度，以简单易于实施的异或运算取代了有限域复杂的运算。其编解码的运算仅仅限于快速的异或运算（即上述的相加），大大提高了节点修复及数据块再生的速率 ；本实施例中的二进制最小带宽再生码（BMBR Binary Minimum-bandwidth Regenerating）不仅降低了系统运算复杂度，同时可以保证节点修复过程中所消耗的带宽是最小的（即原始文件大小），并不消耗多余的带宽。在带宽资源越来越宝贵的今天，上述 BMBR 码带来的裨益是显然的。BMBR 码可以保证 ：丢失的编码块可以直接下载其他编码模块的若干子集进行修复 ；丢失的编码块可以通过固定数目的编码模块进行修复，该固定数目只与系统丢失了多少模块数有关，而与具体哪些模块丢失无关。同时，BMBR 码修复后的节点存储的数据和失效节点是完全一致的，也就是精确修复，很大程度上减少了系统操作复杂度因此。综上，其运算简单、开销小、修复带宽较小。

附图说明

[0063] 图 1 是现有技术中 EC 码的数据重构示意图 ；

[0064] 图 2 是现有技术中 EC 码的失效存储节点修复示意图 ；

[0065] 图 3 是现有技术中 RGC 码的数据重构示意图 ；

[0066] 图 4 是本发明最小带宽再生码的编码和存储节点修复方法实施例中编码方法的流程图 ；

[0067] 图 5 是所述实施例中编码数据包的取得流程图 ；

[0068] 图 6 是所述是实施例中编码标识码的取得流程图 ；

[0069] 图 7 是所述实施例中存储节点的修复流程图 ；

[0070] 图 8 是所述实施例中编码数据包的取得示意图。

具体实施方式

[0071] 下面将结合附图对本发明实施例作进一步说明。

[0072] 如图 4 所示，在本发明最小带宽再生码的编码和存储节点修复方法实施例中，该最小带宽再生码的编码方法包括如下步骤 ：

[0073] 步骤 S41 均分原始数据，得到第一数据包 ：在本步骤中，将大小为 B 的原始数据平均分为 k（k+1）/2 个数据块，每个数据块大小为 L 比特，得到第一数据包 ；所述第一数据包表示为 $c_i=b_{i,1}b_{i,2}...b_{i,L}$，i=1,2,...,k(k+1)/2 ；在本实施例中，为了简单起见，作为一个例

子，设上述得到数据块的大小为 1 比特，这样，每个第一数据包中包括了一个比特的数据，即 $c_i=b_{i,1}$；这样，在本实施例中，上述参数 B 和 k（k+1）/2 在数量上是相等的，即 B=k（k+1）/2。

[0074] 步骤 S42 使用得到的第一数据包构建系统矩阵：在本步骤中，使用上述步骤中得到的第一数据包构建尺寸为 k×k 的、对称的系统矩阵 S；其中，按照其编号依次取得第一数据包，并将取得的第一数据包按照系统矩阵 S 中元素所在列的顺序、逐行依次填入系统矩阵 S 的上三角中，得到系统矩阵 S 的上三角。也就是说，在本步骤中，通过使用上述步骤中得到的第一数据包，得到一个系统矩阵 S。在本实施例中，该系统矩阵是对称矩阵，即以该矩阵的对角线为轴，轴两侧的矩阵元素是对称的。为此，在本实施例中，采用了先得到该系统矩阵的上三角，然后再将其对折，得到整个系统矩阵的方法。得到系统矩阵上三角的方法为：按照第一数据包的编号（即在其表达式中的下标 i）将其取出，将 i=1 的第一数据包放置在系统矩阵第 1 行中的第 1 列元素的位置上，将 i=2 的第一数据包放置在系统矩阵第 1 行中的第 2 列元素的位置上，将 i=3 的第一数据包放置在系统矩阵第 1 行中的第 3 列元素的位置上，以此类推，放置到第 k 个第一数据包时，得到系统数据包的第 1 行；由于是构建系统矩阵的上三角，所以，第 i=k+1 个第一数据包放置到该系统矩阵 S 的第 2 行第 2 列元素的位置，按照上述方法，第 k+2 个第一数据包放置在该系统矩阵 S 的第 2 行第 3 列，……，到第 2 行第 k 列时，放置的是第 2k-1 个第一数据包。总之，在构建上述上三角时，系统矩阵 S 的第 1 行按照上述方法填入 k 个第一数据包（c_1 到 c_k），第 2 行填入 k-1 个数据包（c_{k+1} 到 c_{2k-1}），第 3 行填入 k-2 个第一数据包（c_{2k} 到 c_{3k-2}），……，第 k 行填入 1 个第一数据包（c_B），于是，系统矩阵 S 的上三角共使用 k×(k-1)×(k-2)×(k-3)×L×2×1=k(k-1)/2，刚好将上述步骤中得到的第一数据包用完。得到系统矩阵的上三角后，将其沿对角线对折，得到系统矩阵 S。也就是说，在本实施例中，将得到的第一数据包按照其编号取出，并按照系统矩阵 S 中元素所在列的顺序、逐行依次填入所述系统矩阵 S 的上三角部分，得到系统矩阵 S 的上三角部分为 $\begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ & c_{k+1} & \cdots & c_{2k-1} \\ & & \vdots & \vdots \\ & & & c_B \end{bmatrix}$；其中，B=k（k+1）/2；然后，将上述步骤中得到上三角部分沿其对角线对折而得到该系统矩阵 S 的下三角部分，于是，系统矩阵表示为：

$$S=\begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_{k+1} & \cdots & c_{2k-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_k & c_{2k-1} & \cdots & c_B \end{bmatrix}。$$

[0075] 步骤 S43 构建编码标识码，并使用编码标识码得到编码数据包集：在本步骤中，构建 k 个编码标识码，每个编码标识码包括 k 个元素（编码标识码的构建方法在稍后详述）；之后，分别将系统矩阵 S 的一列中的第一数据包按照一个标识编码中对应于该第一数据包编号的元素的值在该第一数据包的数据头或尾部加入设定数量的比特 0，得到 k 个第二数据包，运算所述 k 个第二数据包得到一个编码数据包；对系统矩阵 S 中的该列使用不同的编码标识码重复上述步骤得到 k 个编码数据包；所述 k 个编码数据包按使用的编码数据包的编

号排列而得到一个编码数据包集 $P_g=p_{g,1}p_{g,2}...p_{g,k}$，其中，g=1,2,..,n-k，$p_{g,k}$ 是由所述第 g 个编码标识码和系统矩阵 S 的第 k 列得到的编码数据包；然后，分别选择 n-k 个不同的系统矩阵 S 的列重复上述步骤，得到 n-k 个编码数据包集。也就是说，在本实施例中，选择系统矩阵 S 中的一列，先使用得到的 n-k 个编码标识码中的一个（例如，编号为 1 编码标识码）对其进行添加比特 0 的处理，得到 k 个第二数据包，对得到的 k 个第二数据包进行异或运算，得到一个编码数据包；按照编码标识码的编号依次使用不同的编码标识码分别对该列进行相同的处理，共得到 k 个编码数据包；将这些得到的编码数据包按照其处理时使用的编码标识码依次排列，得到一个编码数据包集。分别再选择上述系统矩阵 S 中 n-k-1 个不同的列进行上述步骤，共得到 n-k 个编码数据包集。值得一提的是，在本实施例中，通常可以由系统矩阵 S 的第一列开始，一直选择到第 n-k 列来运行上述步骤。

[0076] 步骤 S44 使用编码数据包集构建校验矩阵：在本步骤中，构建大小为 (n-k)×k 的校验矩阵 P，校验矩阵 P 的各行为依次排列的所述编码数据包集 P_g；即该校验矩阵 P 的每行是一个上述得到的编码数据包集；每行的第 1 列是使用第 1 个编码标识码得到的编码数据包，第 2 列是使用第 2 个编码标识码得到的编码数据包，以此类推，第 k 列是使用第 k 个编码标识码得到的编码数据包；校验矩阵 P 中不同的行是选择系统矩阵 S 不同的列而得到的编码数据包集；通常，校验矩阵 P 的行是按照别选择的系统矩阵 S 的列号顺序排列的；例如，由系统矩阵 S 的第一列得到的编码数据包集是校验矩阵 P 的第一行，由系统矩阵 S 的第二列得到的编码数据包集是校验矩阵 P 的第二行，由系统矩阵 S 的第三列得到的编码数据包集是校验矩阵 P 的第三行，并以此类推。

[0077] 步骤 S45 将系统矩阵和校验矩阵中的每行数据分别存储到不同的存储节点：在本步骤中，分别将系统矩阵 S 中的每一行包括的第一数据包存储到一个存储节点，得到 k 个系统节点；分别将校验矩阵 P 中的每一行存储到一个存储节点，得到 n-k 个校验节点，以实现数据编码及存储。其中，n 是存储节点总数。

[0078] 在本实施例中的步骤 S43 中，其编码数据包的取得具体包括如下步骤：

[0079] 步骤 S51 得到编码标识码：在本步骤中，得到 k 个编码标识码（取得编码标识码的具体步骤在稍后详述）。每个编码标识码中包括 k 个数值（或元素），这些数值指示出该编码标识码用于与系统矩阵 S 的列运算产生第二数据包时，应该在与该数值（编码识别码中的数值）位置对应的第一数据包的数据头部添加比特 0 的个数。

[0080] 步骤 S52 按照所述编码标识码对所述每个第一数据包在其数据头部或尾部添加设定数量的比特 0，得到 k 个第二数据包：在本步骤中，首先选择一个系统矩阵 S 的列，例如，系统矩阵 S 的第一列；然后，选择一个编码识别码，例如，包括了 k 个元素的第一个编码识别码，使用该第一个编码识别码对该选择的列进行处理，得到 k 个第二数据包。处理过程如下：取得该第一列的第一个第一数据包，在其前面加该编码识别码中第一个元素值的比特 0；得到所有编码识别码中元素值的最大值，将其与上述编码识别码中第一个元素值相减，得到一个数值；在该列第一个第一数据包的尾部增加上述相减得到数值个数的比特 0；得到一个第二数据包；其中，在该数据包的数据尾部加入 $r_{max}-r_i^a$ 个比特 0，其中，$r_{max}=max(r_1^a, r_2^a, ..., r_n^a)$，为所有编码标识码中元素值的最大值，是事先求得的，通常其最大值为 k-1；r_i^a 是该第一数据包在本次操作中对应的编码标识码的元素值。如此，得到一个（例如，系统矩阵 S 第 1 列第 1 行的第一数据包对应的）第二数据包；在上述选择的系统矩阵

S 的列中，分别对其中位于不同行的第一数据包重复上述步骤，得到 k 个第二数据包。

[0081] 步骤 S53 对所述 k 个第二数据包进行运算，得到其编码数据包：在本步骤中，如上所述，在选择的系统矩阵列上，使用一个编码识别码，可以得到 k 个第二数据包，在本步骤中，将上述得到的 k 可第二数据包运算，得到一个编码数据包。值得一提的是，在本步骤中，运算或相加都是指将这些数据包彼此相互异或。对上述选择的系统矩阵 S 的列分别使用剩余的 k-1 个编码标识码重复上述步骤 S53 和 S53，得到 k 个编码数据包，这 k 个编码数据包是在相同的系统矩阵 S 的列上，使用不同的编码标识码而得到的。将得到 k 个编码数据包按其使用的编码标识码序号排列为一行，得到一个编码数据包集。

[0082] 步骤 S54 得到 n-k 个编码数据包集：选择不同的系统矩阵 S 的列，重复上述步骤 S52 到步骤 S53，直到得到 n-k 个编码数据包集；所述 n-k 个编码数据包集构成冗余符号。将这些得到的编码数据包集按照其产生时选择的系统矩阵 S 的列号（通常，顺序选择系统矩阵 S 的第 1、2、3、…、n-k 列）排列为一列，即可得到编码矩阵 P。

[0083] 此外，在本实施例中，编码数据包的取得过程请参见图 8。图 8 从一个侧面表明了第一数据包、第二数据包及编码数据包之间的变换（转换）关系。

[0084] 图 6 示出了在本实施例中得到编码标识码的具体步骤，包括：

[0085] 步骤 S61 判断 k 是否为素数，如果是，执行步骤 S62；否则，执行步骤 S63；在本实施例中，这个 k 就是将原始数据平均分配为 k（k+1）/2 部分的 k，是依据原始数据的大小事先设定的。

[0086] 步骤 S62 按照 $(r_1^a, r_2^a, \ldots, r_k^a)=(0, a, 2a, \ldots, (n-1)a) \bmod k, a=1, 2, \ldots, k$，得到 k 个编码标识码：在本实施例中，按照上述记载，分别将 a=1,2,...,k 带入数列 (0,a,2a,...,(n-1)a)，并分别对得到的数列中的元素分别取 k 的模，得到 k 个编码标识码。每个 a 分别代入数列并对其进行求模处理，得到一个编码标识码，其中，a 的数值就是该编码标识码的序号。

[0087] 步骤 S63 取大于 k 的最小素数 p，并按照 $(r_1^a, r_2^a, \ldots, r_k^a)=(a-1, 2a-1, \ldots, ka-1) \bmod p, a=1, 2, \ldots, p-1$，得到 k 个编码标识码：分别将 a=1,2,...,p-1 带入数列 (a-1,2a-1,2a,...,ka-1)，并分别对得到的数列中的元素分别取 p 的模，得到 k 个编码标识码。每个 a 分别代入数列并对其进行求模处理，得到一个编码标识码，其中，a 的数值就是该编码标识码的序号。

[0088] 实际的分布式存储系统中，节点经常会发生失效。这时需要引入新的节点，替换失效的节点以保证系统冗余维持在一定的范围内。这一过程称为节点再生。在本实施例中的最小带宽码中再生一个失效的节点，并且最小化所需的修复带宽，可以通过如下方式进行：

[0089] 步骤 S71 确认节点失效：在本步骤中，确认有存储节点失效。

[0090] 步骤 S72 判断该失效节点的类型：在本步骤中，判断失效节点的类型，就本实施例而言，存储节点的类型包括两种：一种是系统节点，一个系统节点存储系统矩阵 S 中的一行数据，共有 k 个系统节点；一种是校验节点，一个校验节点存储所述校验矩阵 P 中的一行数据，共有 n-k 个校验节点；在本步骤中，判断该失效节点是系统节点还是校验节点，如果是系统节点，还要判断该失效的存储节点在系统节点中的编号，例如，是第 f 个系统节点失效，此时，f 在 1 到 k 之间取值，然后执行步骤 S73；如果是校验节点，还要判断该失效的存

储节点在校验节点中的编号，例如，是第 i 个校验节点失效，此时，i 在 1 到 n-k 之间取值，然后执行步骤 S74。值得一提的是，在本步骤的判断方法是结合实际的节点部署而得的。例如，在本实施例中，选择 n 个存储节点，前 k 个是系统节点，后面的 n-k 个节点为校验节点。由于系统在分配存储节点或分配存储节点所存储的数据时会记录相关节点分配信息，通常是以元数据的形式存储。这里的系统，可以理解为一个服务器，它负责调度、管理以及所述节点失效判断。所以，可以由系统中得到一个存储节点是系统节点还是校验节点的信息，同时，也可以得到该节点对应的原始编码的列。

[0091] 步骤 S73 选择剩余的 k-1 个系统节点，分别下载其中的第 f 个数据，并选择对应的校验节点下载其数据，得到失效节点所存储的数据：在本步骤中，由剩余的每个正常的系统节点中下载该存储节点存储的第 f 个数据，即该系统节点位于系统矩阵第 f 列的数据（缺少一个失效节点的第 f 个数据），得到该失效节点中存储的 k-1 个数据（在本实施例中，设发现一个失效节点就立即进行修复），其中，f 是失效的系统节点位于系统矩阵的行数；f=1,2,...,k；选择该列数据对应的校验节点下载其存储的数据（也就是由系统矩阵的、上述下载的列与编码标识码运算得到的校验数据），使用所述由校验节点下载的数据和编码标识码运算，并结合所得到失效系统节点中存储的系统矩阵中一列（即上述下载了 k-1 个数据的列）的已下载数据；得到失效系统节点中的全部数据；将得到的数据存储在新的存储节点并使存储节点取代失效的系统节点。在本步骤中，由于可以由上述步骤中所述的系统得到失效节点的相关编码信息，对于每一个失效的校验节点，总是可以找到其对应的原始编码的列数据。

[0092] 也就是说，在本步骤中，取得失效系统节点在系统矩阵中的行数 f，对于剩余正常的每个系统节点，分别下载其位于系统矩阵的第 f 列的第一数据包；选择存储由所述系统矩阵第 f 列产生编码数据包集的校验节点下载其存储的编码数据包，使用下载的编码数据包和编码标识码进行编码运算的逆运算，得到所述系统矩阵第 f 列的第一数据包；由所述系统矩阵的行、列间的对应关系得到所述失效的系统节点存储的第一数据包。

[0093] 步骤 S74 取得失效节点对应的系统矩阵列号，下载该列数据，编码得到失效校验节点存储的数据：在本步骤中，取得产生失效校验节点所存储编码数据包集对应的系统矩阵的列号 e，e=1,2,..,n-k（该校验节点存储的编码数据包集是由系统矩阵的该列数据和编码识别码运算而得到的），由所有系统节点中分别下载一个数据，下载的数据是系统矩阵中一个完整的、对应于取得列号的列（及系统矩阵的第 e 列）；使用所有编码标识码分别对下载的列数据进行编码（与编码时的运算相同），得到所述失效校验节点存储的数据，将其存储到新的存储节点并使其取代失效的校验节点。

[0094] 步骤 S75 节点修复完成：在执行完上述步骤 S73 或步骤 S74 之后，执行本步骤，使得到的新的存储节点代替上述步骤中检测到的失效节点，完成节点的修复。

[0095] 也就是说，对于校验节点失效而言，需要确定失效的校验节点在编码矩阵中行数 e，e=1,2,..,n-k；并取得 k 个编码标识码（编码识别码是事先已知并存储的）；然后，分别下载 k 个系统节点的第 e 个第一数据包，得到系统矩阵的第 e 列数据；对于得到的系统矩阵第 e 列数据，分别使用取得的编码标识码对其进行编码处理，得到存储在所述失效节点的数据。

[0096] 编码时，取得所述编码标识码中的最大值，即 $r_{max}=max(r_1^a, r_2^a, ..., r_k^a)$；在该系统

矩阵被选择的列的第 y 个第一数据包的数据头部添加等于当前使用的编码标识码中第 y 个元素值的比特 0，而在该第一数据包的数据尾部添加 $r_{max}-r_y^a$ 个比特 0，得到一个第二数据包，其中，y=1,2,...,k；依次分别对该列的 k-1 个第一数据包按照其在该列的行数取相同的 y 值并重复上述步骤，得到 k 个第二数据包；g 是被选择的系统矩阵的列，g 是 1,2,..,n-k 中的一个；将得到的 k 个第二数据包相加（也就是将其异或），得到由当前编码标识码产生的一个编码数据包 $p_{g,j}$，表示通过系统矩阵的第 g 列数据和第 j 编码标识码运算得到的编码数据包。

[0097] 在本实施例中，对于 k 个原始的数据包（长度为 L 比特），不妨记为 $c_i=b_{i,1}b_{i,2}...b_{i,L}$, i=1,2,...,k。难点在于成功找到 n-k 个独立的编码包，使得 n 个数据包（包括数据包和编码包）中的任意 k 个数据包是线性独立的。一般情况下，我们把满足以上条件的数据包称为 (n,k) 独立。

[0098] 例如，取一个文件 $B=\{c_1,c_2\}$，包含两个数据包 c_1、c_2。明显可以看出，运用异或编码，存在三个线性独立的数据包 $\{c_1,c_2,c_1\oplus c_2\}$。然而，这并不能满足分布式存储系统的要求。如果我们在数据包 c_1 头部添加一个比特“0”，在数据包 c_2 尾部添加一个比特“0”。记变动后的数据包为 $c_i(r_i)$，其中 r_i 是在数据包 c_i 头部添加的比特数。就上述三个数据包而言，变动后的数据包和编码包是线性独立的。

[0099] 一般来讲，k 个原始的数据包（长度为 L 比特），不妨记为 $c_i=b_{i,1}b_{i,2}...b_{i,L}$, i=1,2,...,k，编码包 ya 通过如下方式给出：$y_a=c_1(r_1)\oplus c_2(r_2)\oplus\cdots\oplus c_k(r_k)$。每个数据包 c_i 头部总共添加的冗余比特数目为 $r_{max}=\max\{r_1,r_2,...,r_k\}$。编码块 y_a 唯一的标识符（即编码标识码）为 $ID_a=(r_1^a,r_2^a,...,r_k^a)$。可以看出，在数据包 c_i 头部添加的 r_i 冗余比特等效于操作 $c_i(r_i)=2^{r_{max}-r_i}c_i$。

[0100] 如果 k 是任意素数 k，编码块 y_a 唯一的标识符（即编码标识码）可以通过如下方式得到，即：$ID=(r_1^a,r_2^a,...,r_k^a)=(0,a,2a,...,(k-1)a)\bmod k$, a=1,2,...,k. 通过上述编码方式编码出的 n 个数据包 $\{c_1,c_2,...,c_k,y_1,y_2...,y_{n-k}\}$ 是线性独立的。例如，当 k=5，编码标识相应地为 $ID_1=(0,1,2,3,4)_1$, $ID_2=(0,2,4,1,3)_2$, $ID_3=(0,3,1,4,2)_3$, $ID_4=(0,4,3,2,1)_4$, $ID_5=(0,0,0,0,0)_5$.

[0101] 如果 k 不是素数，而是一个正整数 k，可以选择最小的素数 p，并且满足 p>k。此时编码标识可以表示为：

[0102] $(r_1^a,r_2^a,...,r_k^a)=(a-1,2a-1,...,ka-1)\bmod p$, a=1,2,...,p-1.

[0103] $(r_1^p,r_2^p,...,r_{k-1}^p)=(0,0,...,0)$.

[0104] 例如，当 k=4 时，取 p=5，编码标识相应地为 $ID_1=(0,1,2,3)_1$, $ID_2=(1,3,0,2)_2$, $ID_3=(2,0,3,1)_3$, $ID_4=(3,2,1,0)_4$, $ID_5=(0,0,0,0)_5$.

[0105] 综上所述，对于任意正整数 k：如果 k 是素数的话，通过在 k 个原始数据包头前添加（p-1）比特数据（p 是素数，且 p>k），我们可以构造出（n+k，k）线性独立的数据包。如果 k 不是素数的话，同样可以构造出（n+k，k）线性独立的数据包，只是此时在每个原始数据包添加（p-2）比特数据。

[0106] 通常，参数为（n，k，d）的 MBR 码包含 n 个节点，记为 $\{N_1,N_2,...,N_n\}$。BMBR 码应用于包含 n 个节点的系统中，每个节点存储 k 个数据块。任意 k 个节点存储的数据（$\{s_i\}$

$_{i=1,2,\ldots,k}$)涵盖了文件的所有原始数据，通常也将这种编码称为系统码。剩下的 n -k 个节点，通常称之为校验节点，存储的是编码后的数据块。

[0107] 将大小为 B 的文件等分成 k(k+1)/2 份，每份大小为 L 比特。记 S 为 (k×k) 对称的系统矩阵，该矩阵的上三角数据来自集合 $\{c_i\}_{i=1,2,\ldots,B}$。因为 S 严格对称矩阵，所以相应地可以构造出完整的系统矩阵 S：

[0108] $$S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_k \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_{k+1} & \cdots & c_{2k-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_k & c_{2k-1} & \cdots & c_B \end{bmatrix}$$

[0109] 同理，记校验节点存储的编码块的矩阵为 P，其具体形式为：

[0110] $$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{n-k} \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n-k,1} & p_{n-k,2} & \cdots & p_{n-k,k} \end{bmatrix}$$

[0111] 其中，$p_{i,j}$ 是标识符为 (0,i,2i,...,(k-1)i)$_i$modk 的数据包相异或后的编码包。比如说，当 k=3 时，$S = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_2 & c_4 & c_5 \\ c_3 & c_5 & c_6 \end{bmatrix}$。取 L=3 比特的话，数据包文件相应地就可以定义为：$M = \begin{bmatrix} S \\ P \end{bmatrix}$。

[0112] 当一个系统节点 S_i(i=1,2,...,k) 失效了，需要引入新的节点替换它，此时可以从任意 k 个节点中分别下载一个数据包进行修复。具体地做法是，所有选中的节点将其第 i 块数据包传输给该新引入的节点。如果校验节点 P_i(i=1,2,...,n-k) 失效了，同样需要引入新的节点进行替换，此时可以从每个系统节点中分别下载一个数据包并进行相应的编码。编码的过程采用失效节点的标识符，将编码成功后的数据块传送给新引入的节点。由于编码过程所采用的标识符与失效节点的标识符一样，明显可以看出，修复后的节点存储的数据和失效节点是完全一致的。

[0113] 无论是系统节点还是校验节点失效，BMBR 码总是可以实现失效节点的精确修复，同时满足最小割所规定的带宽界限。因此，就修复带宽而言，BMBR 的修复过程是最优的。

[0114] BMBR 码同样是一种 MBR 码，满足所有 MDS 码的特性。也就是说，从任意 k 个节点下载数据就可以恢复出原始数据 B。通常，再生过程中下载所需要的带宽为 k^2，这显然不是最优的。下面我们将给出最优再生过程，可以使得修复过程中下载所需要的带宽最小。

[0115] 最优再生过程如下：数据采集者(DC)可以选择下载数据矩阵 M 第一列的任意 k 个数据，第二列任意(k - 1)数据，第三列任意(k - 2)个数据，直到第 k 列下载一个数据。从 BMBR 码的构造过程可以知道，矩阵 M 的任意一列任意 k 个数据是相互独立的。同时，矩阵

S 是对称的，可以看出我们选择的只是矩阵 S 的下三角数据的以及矩阵 P 的数据。因此，DC 可以获得 B 个线性独立的数据包，最终解码出原始的文件。

[0116] 由以上再生过程可以看出，整个再生过程 DC 下载的数据总量为原始数据大小 B，达到了理论上最优的修复带宽。

[0117] BMBR 码性能评估时，主要分析比较本专利所提出的 BMBR 码与传统 RGC、RS 码在编码、解码以及修复过程中的计算复杂度。

[0118] 编码计算复杂度：

[0119] 对于 BMBR 码，系统总共有(n-k)个校验节点，每个校验节点存储 k 个编码数据包，每个数据包是 k 个原始数据包通过异或运算得到。因此，编码计算复杂度为 k(n-k)(k-1)异或运算。

[0120] 对于 RGC（基于 GF(q)），同样系统有(n-k)个校验节点，每个校验节点存储 k 个编码数据包。不同的是，编码包是通过 k 个原始数据包在有限域 GF(q) 选择相应多项式系数，进行异或运算得到的。因而，传统 RGC 编码计算复杂度为 k（n-k）（k-1）的异或运算，同时 $k^2(n-k)$ 的有限域 GF(q) 上的乘法运算。

[0121] 对于 RS 码，原始文件大小为 B=k(k+1)/2，每个节点仅仅存储一个数据包。通常为了存储大小为 B 的文件，需要存储 (k+1)/2 倍的 RS(n,k) 码所需的数据量。RS 码编码过程和 RGC 相似，因此其计算复杂度为 k(k+1)(n-k)/2 的有限域乘法运算，(k-1)(k+1)(n-k)/2 的异或运算。

[0122] 修复计算复杂度：

[0123] 对于 BMBR 码的修复过程，如果系统中同时有系统节点和校验节点失效的话，系统节点可以理解为优先级高于校验节点。也就是说，系统先修复系统节点然后在修复校验节点。为修复一个系统节点，至少需要一个校验节点、至多需要 k 个校验节点，因而修复一个系统节点的计算复杂度为至少 (k-1)、至多 k(k-1) 的异或运算。修复一个校验节点需要 k 个系统节点，则校验节点的修复计算复杂度为 k(k-1) 的异或运算。

[0124] 为了修复 RGC 的一个节点，k 个协助节点将 k 个数据包汇集于新引入的节点，通过运算该节点通过运算将 k 个数据包再生成之前失效的数据包。所以，整个修复过程的计算复杂度至少为 $2k^2$ 的有限域乘法运算、(2k(k-1)) 的异或运算。

[0125] 而对于 RS 码，修复一个失效的节点需要下载原始文件大小的数据量以重建原始文件，再编码生成失效节点存储的数据包。修复过程的计算复杂度为 $(k^2(k+1)/2+k)$ 的有限域乘法运算、$(k^2(k+1)/2+k-1)$ 的异或运算。

[0126] 解码计算复杂度：

[0127] 为了恢复出原始文件，BMBR 码需要 k(k-1)(k+1)/2 的异或运算。相似地，RGC 的解码复杂度为 k^3 的有限域乘法运算、k^3 的异或运算。RS 码的解码运算复杂度为 $k^2(k+1)/2$ 的有限域乘法运算、$k^2(k+1)/2$ 的异或运算。

[0128] 总结 BMBR 码与传统 RGC、RS 码在编码、解码以及修复过程中的计算复杂度，如下表所示，其中，X 代表异或运算，M 代表有限域乘法运算：

[0129]

Codes	Encoding Computation	Repairing Computation	Decoding Computation
BMBR	$k(k-1)(n-k)\cdot X$	$k(k-1)\cdot X$	$k(k^2-1)/2\cdot X$
RS	$(k^2-1)(n-k)/2\cdot X$ $(k^2+k)(n-k)/2\cdot M$	$(k^2(k+1)/2+k-1)\cdot X$ $(k^2(k+1)/2+k)\cdot M$	$k^2(k+1)/2\cdot X$ $k^2(k+1)/2\cdot M$
RGC	$k(k-1)(n-k)\cdot X$ $k^2(n-k)\cdot M$	$2k(k-1)\cdot X$ $2k^2\cdot M$	$k^3\cdot X$ $k^3\cdot M$

[0130] 本实施例中的最小带宽再生码(BMBR)相比传统 RGC 码,最大的优势在于其大大减小了编解码过程中计算复杂度,以简单易于实施的异或运算取代了有限域复杂的运算。传统 RGC 码的构造基于有限域 GF(q),编解码过程中设计到的有限域加法、减法以及乘法。有限域的运算虽然理论研究比较成熟,但实际运用起来比较繁琐、时间消耗大,明显不能符合当今分布式存储系统快速可靠的设计指标。二进制最小带宽再生码则不同,编解码的运算仅仅限于快速的异或运算,大大提高了节点修复及数据块再生的速率,在实际的分布式存储系统中具有很高的应用价值和发展潜力。

[0131] 本实施例中最小带宽再生码不仅降低了系统运算复杂度,同时可以保证节点修复过程中所消耗的带宽是最小的(即原始文件大小),并不消耗多余的带宽。在带宽资源越来越宝贵的今天,BMBR 码带来的裨益是显然的。BMBR 码可以保证 :丢失的编码块可以直接下载其他编码模块的若干子集进行修复 ;丢失的编码块可以通过固定数目的编码模块进行修复,该固定数目只与系统丢失了多少模块数有关,而与具体哪些模块丢失无关。同时,BMBR 码修复后的节点存储的数据和失效节点是完全一致的,也就是精确修复,很大程度上减少了系统操作复杂度(如元数据更新、更新后的数据广播等)。

[0132] 以上所述实施例仅表达了本发明的几种实施方式,其描述较为具体和详细,但并不能因此而理解为对本发明专利范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本发明构思的前提下,还可以做出若干变形和改进,这些都属于本发明的保护范围。因此,本发明专利的保护范围应以所附权利要求为准。

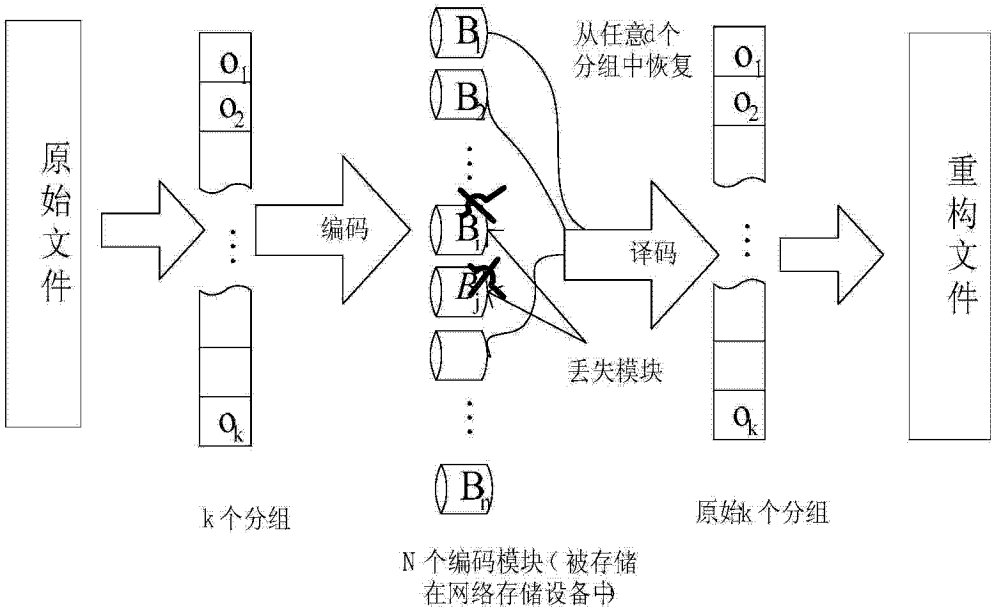

原始文件
编码
从任意d个分组中恢复
译码
丢失模块
重构文件
k 个分组
原始k 个分组
N 个编码模块（被存储在网络存储设备中）

图 1

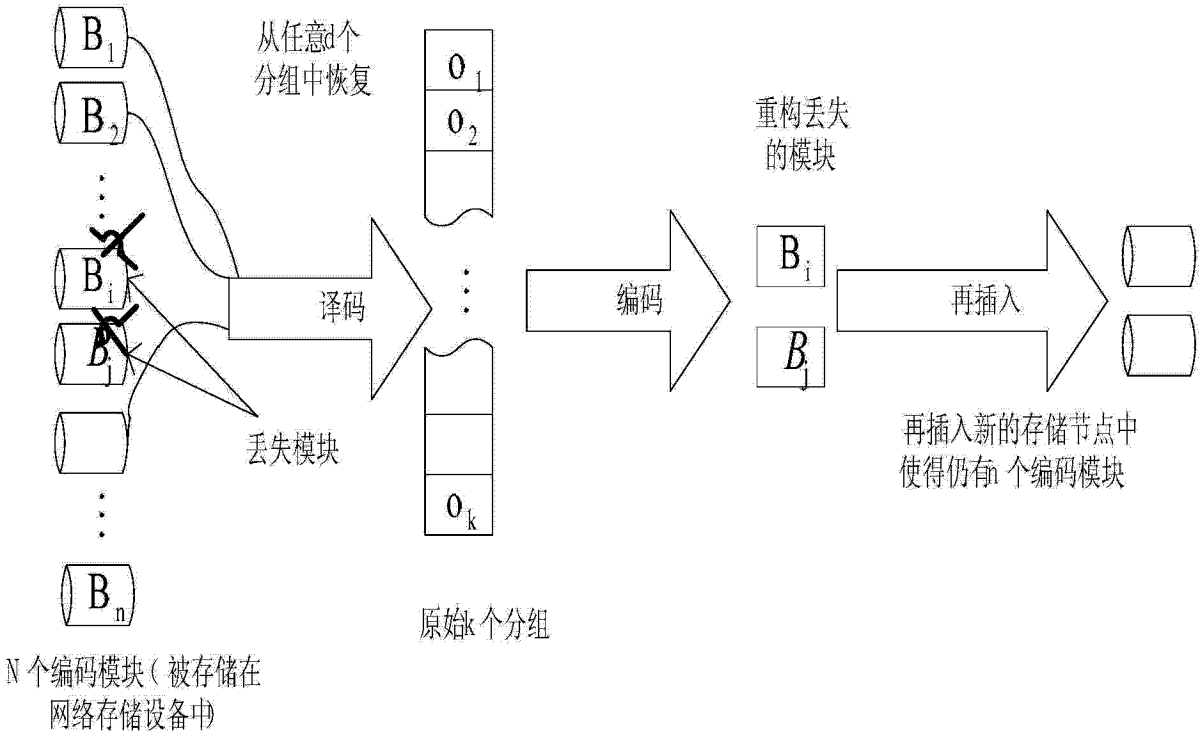

从任意d个分组中恢复
译码
丢失模块
重构丢失的模块
编码
再插入
再插入新的存储节点中使得仍有n 个编码模块
原始k 个分组
N 个编码模块（被存储在网络存储设备中）

图 2

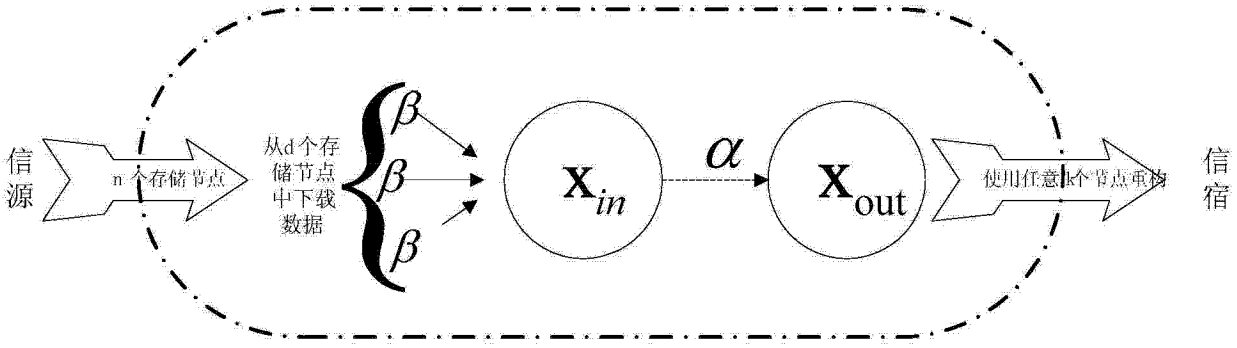

信源
n 个存储节点
从d个存储节点中下载数据
β
β
β
x_{in}
α
x_{out}
使用任意k个节点重构
信宿

图 3

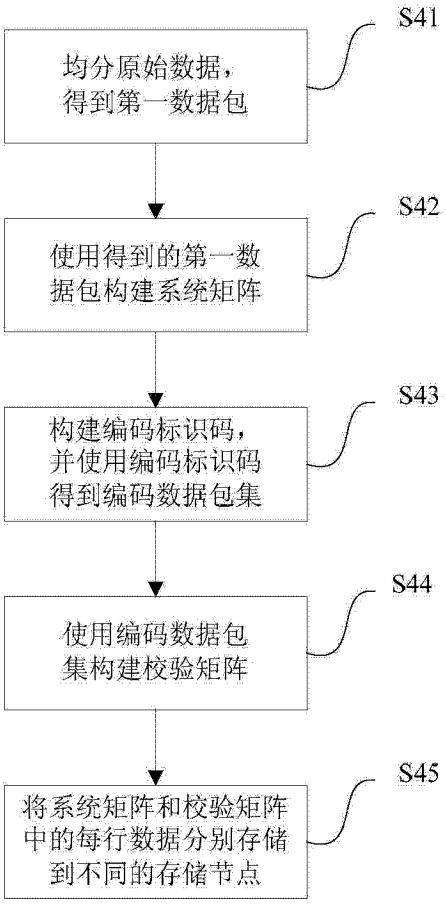

S41
均分原始数据，得到第一数据包
S42
使用得到的第一数据包构建系统矩阵
S43
构建编码标识码，并使用编码标识码得到编码数据包集
S44
使用编码数据包集构建校验矩阵
S45
将系统矩阵和校验矩阵中的每行数据分别存储到不同的存储节点

图 4

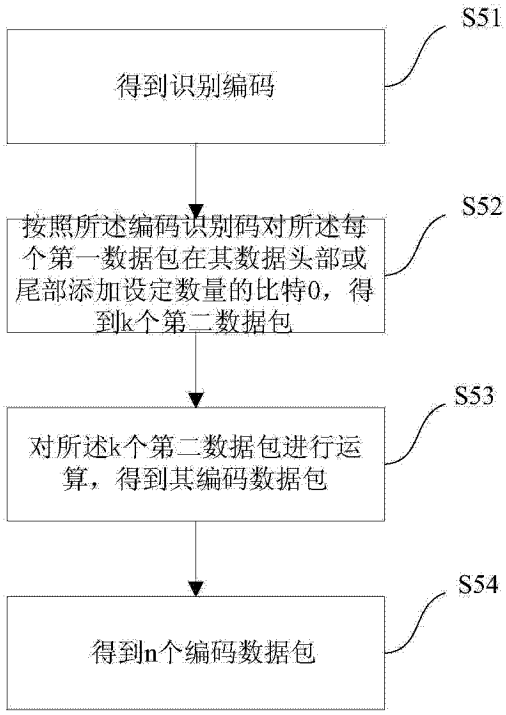

S51
得到识别编码
S52
按照所述编码识别码对所述每个第一数据包在其数据头部或尾部添加设定数量的比特0，得到k个第二数据包
S53
对所述k个第二数据包进行运算，得到其编码数据包
S54
得到n个编码数据包

图 5

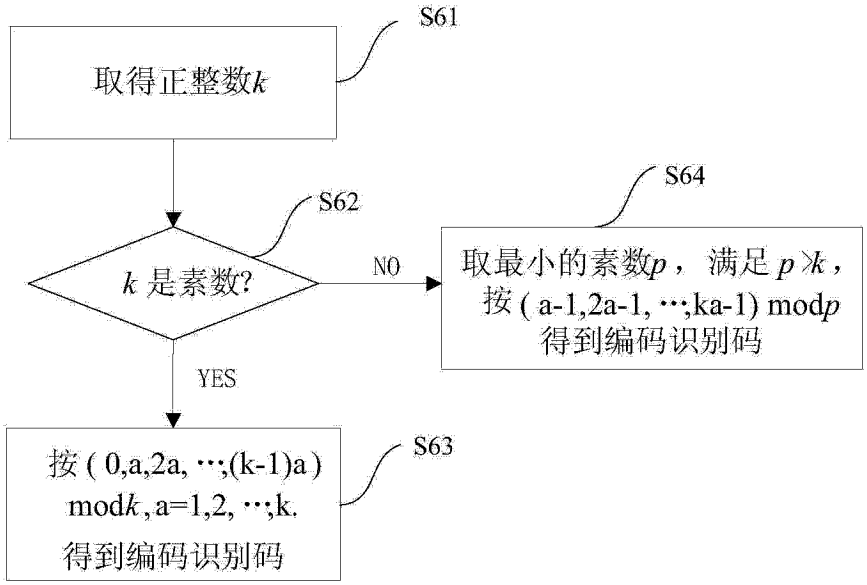

S61
取得正整数k
S62
k 是素数?
NO
S64
取最小的素数p，满足 p>k，
按 (a-1,2a-1, …;ka-1) modp
得到编码识别码
YES
按 (0,a,2a, …;(k-1)a)
modk, a=1,2, …;k.
得到编码识别码
S63

图 6

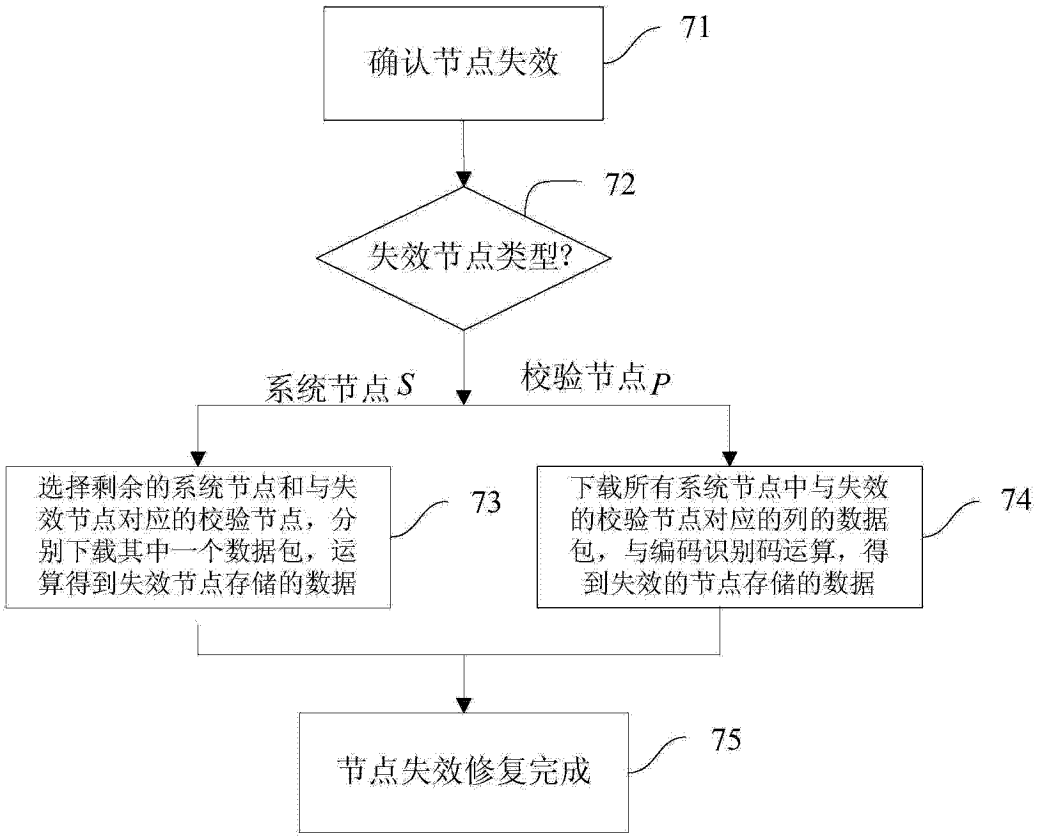

确认节点失效
71
72
失效节点类型?
系统节点 S
校验节点 P
选择剩余的系统节点和与失效节点对应的校验节点，分别下载其中一个数据包，运算得到失效节点存储的数据
73
下载所有系统节点中与失效的校验节点对应的列的数据包，与编码识别码运算，得到失效的节点存储的数据
74
节点失效修复完成
75

图 7

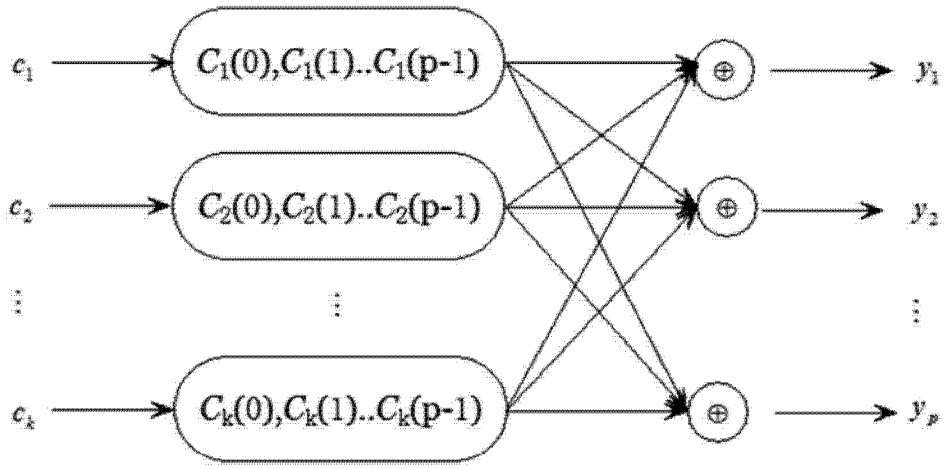


图 8