异构分布式存储系统再生码数据修复的节点选择方案

钟凤艳 王 艳 李念爽
(华东交通大学软件学院 南昌 330013)

摘 要 近年来,海量数据的增长给现有的存储系统带来了严峻的挑战,包括存储成本和数据可靠性要求等。纠删码由于在相同的存储开销下可以提供更高的数据可靠性,得到了学术界和工业界的广泛关注。但由于纠删码的编码特性,让使用纠删码的存储系统在数据修复过程中增加了许多其他方面的额外开销,如计算、调度、传输、磁盘读写等。近年来对纠删码数据修复的研究都基于这样一个假定:分布式存储系统中各个节点是无差别的。然而,实际情况是,在大规模的数据中心中,设备替换、硬件故障等原因不仅会导致数据丢失,还会导致数据中心的各个存储节点的存储成本不同,从而使每个存储节点上所存储的数据量并不总是相等,这种现象被称为存储容量异构。存储容量异构场景下的修复过程面临供应节点的选择问题,需要设计一个节点选择策略来降低修复开销,提高存储系统的可靠性和可用性。鉴于实际数据修复过程中参与修复的节点对数据的传输成本不同,提出节点选择策略——树形拓扑修复算法,以降低整个修复过程中的修复成本。仿真结果表明,相对IFR码的固定节点选择策略,文中提出的树形选择策略在平均情况下可以进一步降低数据修复成本。

关键词 分布式存储系统,节点异构,再生码,数据修复
中图法分类号 TP309.3 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.006

Node Selection Scheme for Data Repair in Heterogeneous Distributed Storage Systems

ZHONG Feng-yan WANG Yan LI Nian-shuang
(School of Software, East China Jiaotong University, Nanchang 330013, China)

Abstract In recent years, the growth of massive data poses severe challenges to existing storage systems, including storage cost and data reliability requirements. Because erasure code can provide higher data reliability under the same storage overhead, it has been paid wide attention. However, the coding characteristics of erasure code increases the extra overhead for the storage system using erasure code, in the process of data repair, such as computing, scheduling, transmission, disk reading and writing, and so on. In recent years, the study of erasure code data recovery is based on the assumption that each node in the distributed storage system is indiscriminate. In a large scale of data center, however, equipment replacement, hardware failure and other reasons may not only cause data loss, but also lead to different storage cost of each storage node in the data center, so that the amount of data stored on each storage node is not always the same, this phenomenon is called storage capacity isomerism. The repair process under the heterogeneous storage capacity is faced with the selection of the providers. It is necessary to design a node selection strategy to make the repair cost lower, and improve the reliability and availability of the storage system. Based on the different transformation cost of nodes participating in repair in the actual repair process of data, this paper proposed a node selection strategy, namely tree topology repair algorithm, to reduce the cost of repair in the whole repair process. The simulation results show that the proposed tree selection strategy can further reduce the cost of data repair compared with the fixed node selection strategy of IFR code.

Keywords Distributed storage system, Node heterogeneity, Regenerating code, Data repair

1 引言

近年来,大规模分布式存储系统已被广泛应用于各大互联网公司的数据中心,例如 Microsoft[1], Haystack[2], facebook[3], TFS[4]。借助这些分布式存储系统,用户可以随时随地上传或下载数据,为了节省成本,分布式存储系统采用性价比较高的普通 PC 服务器作为存储服务器,这些存储服务器分散在不同的地理区域上,借助 Internet 技术互联,对外其作

到稿日期:2018-07-12 返修日期:2018-10-29 本文受国家自然科学基金项目(61402172),江西省教育厅项目(GJJ150509),教育部人文社科基金(15YJA860013)资助。

钟凤艳(1993—),女,硕士,主要研究方向为分布式存储,E-mail:1215396009@qq.com;**王 艳**(1982—),女,博士,副教授,主要研究方向为分布式存储、数据可靠性等,E-mail:wangyann@189.cn(通信作者);**李念爽**(1993—),男,硕士,主要研究方向为分布式存储。

为一个整体提供存储服务，而数据以分布式的形式保存于大量存储节点中，随着数据规模和节点数量的不断增加，节点失效成为了分布式存储系统中的常态[5-6]。为了避免因节点失效而造成数据丢失，分布式存储系统需要保存额外的冗余数据，通过下载一定量的冗余数据来解码丢失的数据，以保证系统中存储数据的可靠性和可用性。产生冗余数据的方式有副本和纠删码[7-8]。随着数据量的不断增加，副本冗余方式的存储开销越来越大，存储空间利用率极低，而纠删码可以权衡可靠性和存储空间，即在相同可靠性情况下，纠删码比副本方式更节省存储空间。例如，一个总存储容量为 30ZB 的集群，若采用三副本方式存储数据，则有效数据占 10ZB，冗余数据占 20ZB，存储效率为 33.3%；若使用(5,3)纠删码方式存储数据，则有效数据占 18ZB，冗余数据占 12ZB，存储效率为 60%。由此可见，纠删码方式明显提高了存储空间利用率。

不管是采用副本还是纠删码方式产生冗余数据，如果节点失效，存储系统需要在一个正常的存储节点(也叫新生节点)上修复所丢失的数据，通常称从节点失效到完成修复的过程为数据修复过程。相对于副本方式而言，纠删码方式能大大提高存储空间的利用率，但由于纠删码是通过编码的方式产生冗余数据，因此使用纠删码的存储系统在数据修复的过程中需要增加磁盘读写、编码、传输、解码等其他方面的额外开销。因为新生节点需要从多个可用节点(也叫供应节点)下载多个数据块来修复出一个数据块，文献[9]指出如果使用文献[10]中 Facebook 的数据统计，若约 50%的集群数据使用纠删码，则根据失效率可以得出修复流会将所有的链路带宽全部占用，从而严重地影响用户的体验效果。因此，优化纠删码数据修复机制具有非常重要的意义。

目前，不少科研工作者从不同方面优化纠删码数据修复机制。在减少数据修复带宽方面，Dimakis 等[11]通过分析信息流图推导出修复带宽开销和存储开销之间的权衡关系，并在文献[12]中首次将网络编码技术用于纠删码的数据修复，并提出了两种优化修复带宽开销的编码——OMMDS 编码和再生码(Regenerating Codes)。在减少磁盘开销方面，Wang 等[13]为了优化冗余数据修复过程中的读取量，通过归纳法给出了参数为$(n=k+2,k,r)$的精确修复 MDS 阵列码的构造方法。Shen 等[14]提出了一种编码感知数据布局(EDP)以减小在降级读过程中的磁盘开销。Ye 等[15]提出了一种 Hybrid-RC 码，该编码的特点是利用 MDS 码的优势计算部分数据块，并用校验数据块保持可靠性，使得修复失效节点时只需要一半的数据量，进而降低磁盘开销。王静等[16]提出了一种新的基于简单再生码的分段编码方案，使得修复失效节点时只需要增加少量存储开销，很大程度地优化了其磁盘读取的开销性能。

以上工作主要针对同构场景，每个存储节点的存储量是一致的，节点间的传输成本也是一致的，然而实际情况是不同区域的数据中心之间一般存在传输成本的差异性，不同存储设备的存储成本也存在差异性，传输成本的异构性对冗余数据修复性能的影响很大，存储成本的异构性影响每个存储节点的存储量，进而限制新生节点从供应节点下载的数据量和数据的完整性。Akhlaghi 等[17]首先考虑了比较简单的场景，假设系统中有两类节点 S_1 和 S_2，每类节点都对应不同的通信代价 C_1 和 C_1。根据再生码的要求，即修复时新生节点需要连接 d 个供应节点，假定从通信代价为 C_1 的节点中选取 d_1 个节点，从通信代价为 C_2 的通信节点里选取 $d_2=d-d_1$ 个节点。此时，修复一个新生节点所需的总修复代价 C_T 可以表示为：$C_T=(C_1d_1+C_2d_2)\beta$。其忽略了节点的存储成本，给出了修复代价和修复带宽的权衡曲线。Gerami 等[18]考虑了网络拓扑结构对数据修复过程的影响，通过信息流图的方法分析了满足最小割的条件，在此基础上将最优化通信代价问题转化成一个线性规划问题，给出了 4 种典型的网络模型下通信代价的下界。他们的研究的局限在于，使用了与典型再生码一样的假设，即新生节点从每个供应节点下载的数据量相同。李松涛等[19]提出了一种基于总体存储成本凸函数特性的自适应方法来实现缓存大小的自适应选择，以平衡系统缓存大小、数据存储成本、带宽成本和数据对象的访问时间。Du 等[20]针对分布式存储系统中每个节点存在计算能力和读写能力不同的场景，提出了一种数据放置方法，以降低存储成本并提高数据可靠性。Yu 等[21]假设系统中有两类节点 S_1 和 S_2，每类节点都对应不同的存储成本 C_1 和 C_2，每类节点对应的存储量为 α_1 和 α_2，修复数据时新生节点选择 d 个供应节点，从每个供应节点下载 β 数据量，此时修复带宽为 $d\beta$。他们在此基础上不考虑数据在链路上的传输成本，给出了存储成本和修复带宽的权衡曲线。Yu 等[22]同时考虑了节点的存储成本和节点间传输成本的差异性，构造了一种新的再生码，称为 MDS-IFR 码。该编码由外编码 MDS 码和内 IFR 两层编码组成，保留了无编码修复性质，即供应节点从磁盘读出相应的数据传递给新生节点，磁盘访问开销等于带宽开销。使用该编码的系统修复失效节点时磁盘开销和修复带宽能同时达到最小，然而这类编码的缺点在于存储开销较大，甚至比三副本的存储开销还大，因为该编码的思想是先对原始数据进行 MDS 编码，再把编码后产生的编码块分散地放置，每个编码块复制 ρ 次，因此该编码产生冗余数据的方式综合了纠删码方式和副本方式两种。以该文献构造 MDS-IFR 码的过程为例，假设原文件大小为 M，由 $k=4$ 个数据块组成，每个块的大小均为 $\frac{M}{k}$，原文件经过(7,4)MDS 码编码后产生了 7 个数据块，每个数据块复制 2 次，把这 14 个数据块分散地放置在 6 个节点上，所需的存储空间大小为 $3.5M$，存储效率为 28.6%，而如果用副本方式存储这 M 大小的数据量，所需存储空间大小为 $3M$，存储效率为 33.3%。

本文的主要工作是：在文献[22]的基础上，研究每个存储节点的存储容量不相等(也叫存储容量异构)时对数据修复过程的影响，通过建模分析，提出对失效节点的树形修复方案，考虑节点间不同链路上的传输成本因素，建立供应节点选择机制。具体来说，鉴于文献[22]所构造的编码存在存储效率低的问题，本文不是构造新的编码，而是依照现有再生码的构造方法，将原文件按照再生码的规则进行编码，考虑到节点的存储成本，把编码后产生的数据块进行合理的放置。由于 Li

等[23]指出可以构造树形拓扑过程来修复失效节点，供应节点可以对本身所存储的数据块和接收到的数据块进行预编码，再把编码结果向上传给其父节点，最终新生节点修复出相应的丢失的数据块。另外，单个节点失效的情况超过 98%，比多个节点同时失效的情况更普遍[24]。因此，本文关注单个节点失效的修复问题，通过构造树形拓扑来修复失效节点，对于原数据获取问题，数据收集节点(DC 节点)连接指定的恢复集来恢复原文件，而不是任意的 k 个节点。

本文提出的基于存储异构的数据放置及数据修复算法主要有以下两个优势：

1)存储效率高。把原文件划分成更多的块，因为总的编码块数目 B 不变，k 越大，系统存储这个原文件所需的存储空间就越小。

2)数据修复成本低。通过利用网络中不同链路上的传输成本，提出了一种存储容量自适应的树形再生过程，充分利用传输成本低的链路，避开传输成本高的链路，从而降低修复成本。

2　示例分析

下面使用一个简单的例子来说明本文的核心工作。假设一个分布式系统由 $n=6$ 个节点组成，节点 $i(i=1,\cdots,6)$ 的存储成本用 s_i 表示，相邻节点 $i,j(i\neq j)$ 的传输成本用 C_{ij} 表示，各个节点的存储成本和节点间的传输成本已在图 1 中标出，Node1－Node6 分别代表分布式存储系统中的 6 个节点。MDS-IFR 码的编码步骤如下：把原文件划分成 4 个数据块，分别用 F_1-F_4 表示，每个数据块的大小为 $\frac{M}{4}$，经过(7，4) MDS 码编码后产生 7 个数据块，分别用 F_1-F_7 表示，每个数据块都复制 2 份，将这 14 个数据块按照图 2 所示的方式放置在各个节点上，Node1 存储 F_1 和 F_7，Node2 存储 F_1 和 F_2，Node3 存储 F_2 和 F_3，Node4 存储 F_3，Node5 存储 F_4-F_6，Node6 存储 F_4-F_7。在这种分配方式下，新生节点可以通过连接指定的 1 个或 2 个节点修复丢失的数据，所需连接的节点数目取决于具体的失效节点，例如 Node1 失效时，新生节点连接 2 个供应节点完成修复操作，而 Node4 失效时，新生节点仅需连接 1 个节点就可以完成修复操作。至于数据恢复，数据收集节点可以从 10 个包含 $m=2$ 个节点的恢复集合中恢复原文件，分别是 $R_1=\{1,3\}$，$R_2=\{1,5\}$，$R_3=\{1,6\}$，$R_4=\{2,5\}$，$R_5=\{2,6\}$，$R_6=\{3,5\}$，$R_7=\{3,6\}$，$R_8=\{4,5\}$，$R_9=\{4,6\}$，$R_{10}=\{5,6\}$。这里，我们把总存储空间和原文件的比值定义为存储开销。本示例中，系统存储这个原文件需要的总存储空间为原文件的 3.5 倍大小。如果用(14，7)RS 码进行编码，首先把原文件划分为 7 个数据块，每个数据块的大小为 $\frac{M}{7}$，经过编码后产生 14 个数据块，将这 14 个数据块按照图 3 所示的方式放置在各个节点上，Node1 存储 F_1 和 F_2，Node2 存储 F_3 和 F_4，Node3 存储 F_5 和 F_6，Node4 存储 F_7，Node5 存储 F_8-F_{10}，Node6 存储 $F_{11}-F_{14}$。新生节点可以通过构造一个 (r,d) 再生树修复出丢失的数据，其中 r 表示供应节点的数目，d 表示可供选择的节点数目。在这个例子中，假如 Node1 失效，新生节点通过构造一棵(2，5)再生树修复 Node1 上的数据，再生过程如图 4 所示。在树形拓扑中，Node5 接收到 Node3 传来的 F_3 后，紧接着与自己的编码块编码出 F_1 和 F_2，把这两个编码块向上传给新生节点，从而完成修复操作。此时，修复 Node1 的修复成本是 3，相比于 MDS-IFR 码编码方式，修复 Node1 所需的开销减小 25%。数据收集节点可以从 13 个包含 $m=3$ 个节点的恢复集合中恢复原文件，分别是 $R_1=\{1,2,5\}$，$R_2=\{1,2,6\}$，$R_3=\{1,3,5\}$，$R_4=\{1,3,6\}$，$R_5=\{1,4,6\}$，$R_6=\{1,5,6\}$，$R_7=\{2,3,5\}$，$R_8=\{2,3,6\}$，$R_9=\{2,4,6\}$，$R_{10}=\{2,5,6\}$，$R_{11}=\{3,4,6\}$，$R_{12}=\{3,5,6\}$，$R_{13}=\{4,5,6\}$。系统存储这个原文件需要的总存储空间为原文件大小的 2 倍。由示例分析可知，与 MDS-IFR 编码相比，再生码能在不增加修复开销的情况下降低存储开销。本文的主要贡献是降低存储开销，并据此优化数据修复机制。

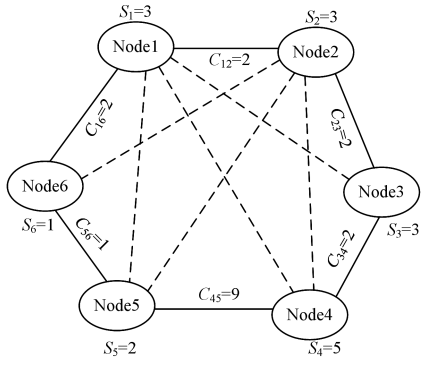

图 1　节点拓扑结构

Fig. 1　Node topology

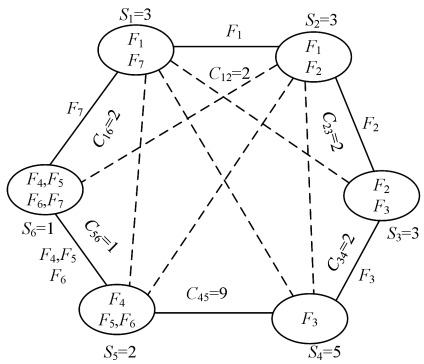

图 2　IFR 码构造实例[22]

Fig. 2　Example of constructing MDS-IFR code[22]

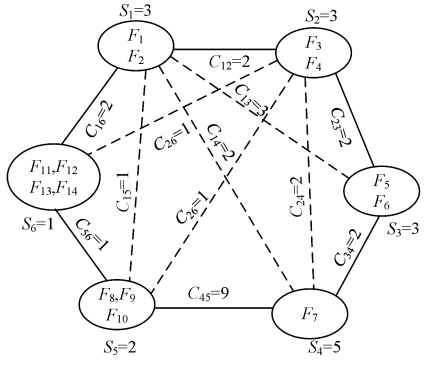

图 3　再生码构造实例

Fig. 3　Example of constructing regenerating code

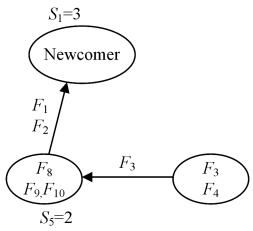

图 4 树形再生过程

Fig. 4 Tree regeneration process

3 对存储成本和通信成本建模

本节对存储成本和通信成本进行建模，并在存储节点存储成本异构的场景下设计数据分配方案，在树形再生拓扑结构中，结合存储节点传输成本的不同，提出存储容量异构场景下供应节点的选择方案。

3.1 存储成本和通信成本建模

一个再生码存储系统通常用(n,k,d,α)来描述，n 表示系统中存储节点的总数。假设原文件大小是 MBit，再生码编码过程如下：把原文件平均分成 k 个块，每个块的大小为 $\alpha(\alpha=\frac{M}{k})$ Bit，把这 k 个块编码成 n 个冗余块，然后分散地存储到 n 个不同的存储节点上，每个存储节点存储 αBit，其中编码过程要保证 MDS 性质，即数据收集节点(DC 节点)通过访问任意 k 个节点下载不小于 M 大小的数据量就能够重建出原文件。当一个节点失效后，新生节点通过访问任意 d 个供应节点可以修复出相应的丢失的数据。上述再生码的数据分配假设每个节点的存储量和存储成本是一样的，修复过程假设每条链路的传输成本是一样的，从每个供应节点的下载量也是一样的。然而，实际的分布式存储系统由于地理位置、设备更新等原因导致存储节点和网络链路异构，这意味着每个存储节点的存储容量以及相应的存储成本也不总是相同，通常我们把每个节点的存储量不同称为存储容量异构；另外，从带宽、通信成本、传输速度等角度来看，每对节点之间的链路也存在不同的特性。对于这样的异构分布式存储系统，我们使用带权无向图$\widetilde{G}=(V,\widetilde{w})$代表异构分布式存储系统的存储网络模型，其中 V 代表节点集，$\widetilde{w}$代表节点间的边集。用 s_i 表示存储节点 $i\in V$ 的存储成本，存储成本指节点存储一个数据块的成本。用$\widetilde{c}_{ij}$表示任意一对节点 $i,j(i\neq j)$的传输成本，传输成本指一个数据块在一条链路上传输所需的成本，称之为单跳成本，所有节点间的单跳成本用矩阵$\widetilde{C}=[\widetilde{c}_{ij}]$表示。

我们把每个存储节点的存储量用一组标号 $x_1,x_2,\cdots,x_n$ 表示(其中 x_i 表示节点 i 存储块的数目)，并分配相应的权值因子(存储成本)$s_1,s_2,\cdots,s_n$，则总的存储成本可以定义为：

$$c_s=\sum_{i=1}^{n}x_is_i \tag{1}$$

由式(1)可知，系统存储成本 c_s 的值是由各个节点的存储量及其存储成本决定的。为了降低系统存储成本，各个节点的存储成本值在考虑数据分配时是一个很重要的参考因素。

3.2 数据分配

在实际的分布式存储系统中，存储节点的存储成本可以预先测出，在获得各个节点的存储成本后，将编码产生的 B 个编码块合理地放置在 n 个不同的节点上，使得总的存储成本最优。在最优分配下，原文件的恢复满足 DC 节点从包含 m 个节点的 w 个恢复集中任意选择一个恢复集就能获取原文件的要求。为解决存储成本优化的问题，可以将该问题转化成一个线性规划问题，即最小化存储成本问题，可以描述为：

$$\text{Minnimize } c_s \triangleq\sum_{i=1}^{n}x_is_i \tag{2}$$

各个节点的储存量满足以下约束条件：

$$x_1+x_2+\cdots+x_n=B \tag{3}$$

$$R_i\subset\Psi,i=1,2,\cdots,w \tag{4}$$

$$x_j\in R_i,j=1,2,\cdots,m \tag{5}$$

$$\sum_{j=1}^{m}x_j\geqslant M \tag{6}$$

式(2)中，x_i 表示节点 i 的存储量，s_i 表示节点 i 的存储成本。式(4)－式(6)表示按照再生码恢复原文件的要求，DC 节点选择任意一个包含 m 个节点的恢复集 $R_i\subset\Psi(i=1,2,\cdots,w)$，连接该恢复集中的每个节点 $j(j=1,2,\cdots,m)$并下载 x_j 数据量，使得 DC 节点接收到的总数据量不少于 M 即可获取原文件。

上文给出了冗余数据分配的一般性方案，下文将给出一个例子来说明在特定编码方式下数据分配的具体方案。仍以图 1 为例，经过(7,14)RS 编码后产生 14 个冗余数据块，将这 14 个块分散地放置在 6 个节点上，那么就有 $A_{14}^{6}\times14^{8}$ 种分配结果。我们采用专业的数学工具 Matlab，使用其中的函数 conv()可以快速得到分配结果，遍历每一种分配结果，根据约束条件(3)－(6)，剔除不满足约束条件的分配结果，再次遍历剩下的满足约束条件的分配结果，计算每种分配结果下的 c_s 值，舍弃较大的 c_s 值对应的分配结果，直至找到最小的 c_s 值对应的分配结果，经过多次筛选后，最终得到图 3 所示的分配结果。

3.3 数据修复

在找出最优数据分配方案后，考虑原文件恢复问题以及节点失效的修复问题。我们先定义数据修复要求和恢复要求。数据修复要求：如果系统中不超过 $n-k$ 个节点同时失效，新生节点通过连接 r 个节点可以精确修复出任意失效节点上的数据，r 也称为节点的修复度，每个节点的修复度可以不一样，r 值取决于具体失效的节点。数据恢复要求：一个集合包含 w 个子集，用 $\Psi\triangleq\{\mathcal{R}_1,\mathcal{R}_2,\cdots,\mathcal{R}_w\}$表示这个集合，集合中的子集也称为恢复集，每个恢复集包含 m 个节点，DC 节点通过选择该集合中的任意一个恢复集可以获取原文件。

对于原文件恢复问题，传统再生码的做法是从 n 个节点中任意选择 k 个节点，分别下载相等数据量恢复原文件，对应恢复集的个数为 $w=\binom{k}{n}$。作为特例，我们限制 DC 节点仅选取一部分恢复集来获取原文件，虽然每个恢复集的基数都相等，但是不要求从每个节点下载的数据量一样，只需满足从一个恢复集 $\mathcal{R}_j\in\Psi$ 中的 m 个节点下载的数据总量不少于原文件大小 M 即可。

对于失效数据的修复问题，在实际的分布式存储系统中，一个数据块可以通过多跳的方式传送到目的节点。假设一个

数据块从节点 i 经过多跳后传送到节点 j，这个过程所需的传输成本用 c_{ij} 表示，称 c_{ij} 为传输成本，c_{ij} 的值为所经链路单跳成本之和。如果节点 i 和节点 j 之间存在多条路径，那么 c_{ij} 的值为最小费用路径的成本，所有节点间的传输成本用矩阵 $C=[c_{ij}]$ 表示。我们可以利用 Johnson 算法[25] 找出每对节点的传输成本，该值在数据修复过程中关于供应节点的选择问题上是一个不可忽略的参考因素。

假设节点 i 向其上级节点传输的数据量为 β_i，节点 i 把 β_i 数据量传输到节点 j 的通信成本为：

$$C_{ij}=\beta_i c_{ij} \tag{7}$$

由式(7)可知，降低通信成本的方法是减小传输量或者选择成本较小的路径。我们从两方面来降低通信成本，即在数据传输的过程中允许中间节点进行预编码以降低传输量，同时各节点可以选择传输成本较小的路径传输数据。按照灵活再生码[26]，一个确定的树形修复拓扑过程中从节点 $i(i\in V)$ 输出的数据量为：

$$\beta_i^*=\begin{cases}\dfrac{c_i M}{k\sum\limits_{r=1}^{d-k+1}c_r}, & \text{if } 1\leqslant i\leqslant d-k+1\\ \dfrac{c_{d-k+1}M}{k\sum\limits_{r=1}^{d-k+1}c_r}, & \text{if } d-k+1<i\leqslant d\end{cases} \tag{8}$$

其中，供应节点按照传输成本的大小排序，$c_1\leqslant c_2\leqslant\cdots\leqslant c_d$。在修复失效节点的过程中，新生节点需要从剩下的 $n-1$ 个可用节点中选择 r 个节点作为供应节点，数据从每个叶子节点 i 经过一系列中间节点处理并传输到新生节点后，整个修复过程的通信成本为：

$$c_r=\min\{\beta_i^* c_{ij}+\beta_j^* c_{jk}+\cdots+\beta_m^* c_{mn}\} \tag{9}$$

其中，节点 $i,j,\cdots,m\in V$，β_m^* 表示节点 m 传输到新生节点的数据量，c_{mn} 是节点 m 到新生节点的传输成本。

3.4　树形修复模型

再生码的提出是为了优化修复带宽，它要求从每个供应节点获取同样多的数据来完成修复过程，因此再生码并不能同时达到优化修复成本的目的。Shah 等[27] 提出弹性修复策略，依据链路传输成本的大小下载不等量数据，传输成本低的链路传输较多数据量，反之，传输较少数据量。Li 等[28] 提出了基于随机线性码的树形修复拓扑再生过程，根据各条边上的权值构造最优再生树，并作为树形修复过程。该再生树覆盖 1 个新生节点和 r 个供应节点，新生节点为根节点，供应节点分为叶子节点和非叶子节点，叶子节点将自己所存储的编码块编码后向上传输给其父节点，非叶子节点接收到其所有子节点的数据后与自己的编码块编码，然后将编码后的结果继续向其父节点传输，如此逐级上传直到根节点收到所有子节点的数据，并对收到的数据进行线性组合生成相应的丢失的编码块，最终完成失效数据的修复。

基于上述研究，我们希望同时利用弹性修复策略和树形再生策略的思想，在构造树形修复拓扑结构时，使用迭代法动态地替换当前通信成本较大的链路，使得构造出的树形修复过程的总通信成本最低。选择供应节点的核心思想是：如果节点 i 将数据传给节点 j，节点 i,j 之间存在多条路径，每条路径上的传输成本不同，那么选择不同的路径产生的成本也不同。为了降低通信成本，对于任意节点 i,j 之间的链路，计算其传输成本，选择较低成本的链路进行数据传输。首先，按照一般星形拓扑的修复算法选择 r 个供应节点，也就是先计算除失效节点外的 $n-1$ 个节点到新生节点的传输成本，求得 $n-1$ 个节点的通信成本 $C_{in}(i=1,2,\cdots,n-1)$，将 C_{in} 按照从小到大的顺序编号，取编号为 $1,2,\cdots,r$ 的节点作为供应节点；然后，依次遍历图中其他可以选择的链路，选择所有覆盖 r 个供应节点的链路，计算每条链路上的传输成本，逐个比较当前的最大传输成本，如果某条链路上的传输成本小于当前传输成本，则替换当前具有较大传输成本的供应节点。节点存储量异构场景下的树形选择供应节点算法如算法 1 所示。

算法 1　树形供应节点选择算法

```
1. P,β*,C←Ø
2. /* P 存放选出来的供应节点 */
3. If (1≤i≤d−k+1) then
4.  β_i* ←β_i* = c_i M / k Σ_{r=1}^{d−k+1} c_r
5. Else if (d−k+1<i≤d) then
6.  β_i* ←β_i* = c_{d−k+1} M / k Σ_{r=1}^{d−k+1} c_r
7. End if
8. /* 节点 i 所需输出的数据量 */
9.  C_ij=β_i* × c_ij
10. /* 计算每条链路的传输成本 */
11. for each link in remeaning Links
12. if(MinLinkCost+link.cost<MaxLinkCost) then
13. remove MaxLink
14. add MinLink.add(link)
15. end if
16. end for
17. return bottleneckLink
```

与在指定的供应节点集合中选择 d 个节点作为供应节点相比，树形修复中所用的修复成本较少，因为在一轮修复过程中，存在供应节点链路替换的机制，把传输成本较高的链路替换为传输成本较低的链路，供应节点的选择相对灵活。我们将从指定的供应节点集合中选择的 d 个节点作为供应节点的机制称为固定选择机制。相比固定选择供应节点机制，该树形拓扑修复算法可以达到全局最优，找到更小的瓶颈链路，所需要的时间复杂度和空间复杂度均为 $O(d(n-d-1))$。

4　实验结果

实验所使用的分布式存储系统集群由 19 个节点组成，其中 1 个节点作为 NameNode 节点，其余 18 个节点作为 DataNode 节点，NameNode 节点同时运行 RaidNode 进程，每个节点配置 2 个 8 核 Intel(R)Core(TM)i7-6500U CPU @ 2.50 GHz 处理器、64 GB 内存、1 TB 硬盘和千兆网卡。所有节点运行在 64 位 CentOS7 系统和 jdk-8u181-linux-x64 上。

事实上，在真实的分布式存储系统中，每个存储节点会因设备更新或替换成其他类型的设备而导致数据存储成本不同，并且处于不同地理区域的存储节点之间进行通信使得节

点间传输数据的成本不同。本节考虑到存储节点的存储成本和传输成本的差异性，通过模拟产生每个存储节点的存储成本值 $s=[3,3,3,5,2,1,2,2,1,1,2,2,5,1,1,1,1,3]$ 和单跳成本值 $\widetilde{C}=[\widetilde{c}_{ij}]$，其对应值都为整数，取值范围为[1,50]。将第 2 节提出的算法与固定选择机制进行修复成本测试对比。为了使结果相对稳定，取 100 次重复实验结果的平均值。

为了分析节点存储容量异构对数据修复过程的影响，我们测试了存储容量异构对修复成本的影响。如图 5 所示，假设原文件大小 $M=30$ MB，n 分别取 8,9,10,11,12,13,14,15，在 8 种不同网络大小的情况下，当一个存储节点失效后，分别测试本文提出的 RS 码的树形修复策略和 IFR 码的固定选择策略的平均修复成本，同时作为对比，加入了传统的 RS 码修复策略，称之为 RS 码的星形修复策略。从图 5 可以看出，在节点存储容量异构的场景下，随着存储网络 n 的增加，存储节点的数量增大，3 种策略下的修复成本逐渐下降，因为 n 越大，在总存储量不变的情况下，单个节点的存储量会降低，修复时所需传输的数据量也将降低，随着 n 的增加，RS 码的树形修复策略比星形修复策略的下降程度更为明显，因为供应节点可选的范围变得越来越多，可以选择的链路也越来越多，选中通信成本低的链路的概率也会增大。同样的原因，RS 码的树形修复策略和 IFR 码的固定选择策略之间的差距也逐渐缩小，甚至优于 IFR 码的固定选择策略，由此可见，利用存储容量异构弥补了节点数量增加的不足。

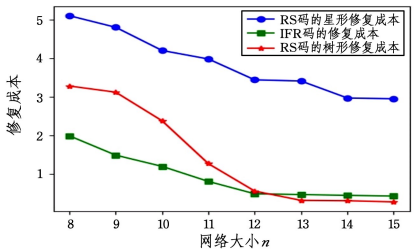

图 5 3 种选择策略在不同网络大小下的修复成本对比

Fig. 5 Repair cost comparison of three selection strategies under different network sizes

如图 6 所示，给定一种 $(n,k,d)=(6,k,3)$ 的 MDS 码，编码模式为 MBR 码，原文件大小为 M，我们使用 5 个不同的 M 值，分别为 50 MB，100 MB，150 MB，200 MB，250 MB，节点的数量为 6。

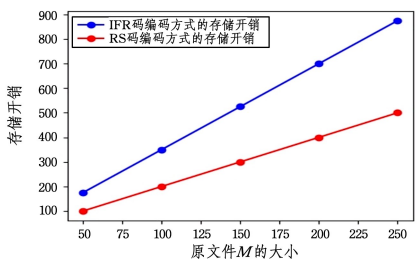

图 6 IFR 码与 RS 码编码方式的存储开销对比

Fig. 6 Comparison of storage overhead of IFR code and RS code

在原文件大小不同的情况下，MDF-IFR 码 $(n=6,k=4,d=4,\rho=2)$ 将原文件划分成 $k=4$ 个块，编码生成 7 个块，每个块复制 2 次，共 14 个块；RS 码 $(n=6,k=7,d=2)$ 将原文件划分成 $k=7$ 个块，编码生成 14 个块。测试 MDS-IFR 码和 RS 码两种产生冗余数据策略的存储开销。从图 6 可以看出，系统存储大小相等的原文件时，RS 码比 MDS-IFR 码所需的存储开销更小，随着原文件的不断增大，两者开销的差距越来越明显。

修复过程中的计算开销是指恢复失效盘上每一个数据块所需的平均异或运算次数。为了分析数据修复过程的计算开销，我们测试 RS 树形修复策略、RS 星形修复策略和 MDS-IFR 码三者的计算开销，定义指标开销比来比较这些编码方案，R_{code2}^{code1} 表示一个编码 $code1$ 修复的计算开销与另外一种编码 $code2$ 的计算开销的比值，表 1 列出了 RS 星形修复策略 RS-星形、MDS-IFR 码和 RS-树形修复策略的计算开销比。由表 1 可知，相比再生码，MDS-IFR 码修复的计算开销更低，RS 树形修复策略 RS-树形的计算开销比其他两种编码的计算开销都要高，RS 树形修复策略 RS-树形的计算开销比 MDS-IFR 码和 RS 星形修复策略 RS-星形的计算开销更高。

表 1 3 种编码在修复度作业下的计算开销比

Table 1 Computation overhead ratio of three codes under repair degree

修复度数	9	16	17	32
$R_{MDS\text{-}IFR}^{RS\text{-}星形}$	7.2	0.12	3.8	0.03
$R_{MDS\text{-}IFR}^{RS\text{-}树形}$	462.2	7.6	244.7	1.97
$R_{RS\text{-}星形}^{RS\text{-}树形}$	8	0.5	16	0.5

结束语 考虑到真实的分布式存储系统存在存储节点存储容量和传输成本的差异性，本文分析了节点的存储容量异构对纠删码数据修复的影响，结合传输成本异构的场景，建立数据树形修复拓扑模型，对于供应节点的选择问题，提出数据修复过程中供应节点的选择策略，并通过仿真实验比较了 RS 码的星形修复策略、IFR 码的固定选择策略与 RS 码的树形修复策略三者的修复成本。结果表明，通过考虑实际的存储节点存储容量异构场景和传输成本异构场景，本文提出的树形修复算法在某种程度上可以进一步降低失效数据的修复成本，从而提高整个存储网络的可靠性。

本文主要关注系统的平均修复成本，所提出的树形修复方式并不是在任何时候都比 IFR 码的修复方式更好，但是平均情况会比 IFR 码更好；另外，该树形修复方式是基于 RS 码提出的，一定程度上可以降低修复成本，但是要以较高的计算开销为代价，而如何降低计算开销是接下来需要解决的问题。

参 考 文 献

[1] HUANG C, SIMITCI H, XU Y, et al. Erasure coding in windows azure storage[C]// Usenix Conference on Technical Conference. USENIX Association, 2012: 2.

[2] BEAVER D, KUMAR S, LI H C, et al. Finding a needle in Haystack: facebook's photo storage[C]// Usenix Conference on Operating Systems Design and Implementation. USENIX Association, 2010: 47-60.

[3] RASHMI K V, BORTHAKUR D, BORTHAKUR D, et al. A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers[C]// ACM Conference on SIGCOMM. ACM, 2014: 331-342.

[4] 杨传辉. 大规模分布式存储系统：原理解析与架构实战[M]. 北京：机械工业出版社，2013.

[5] ZOU S C, WANG H Q, FENG G S, et al. Multi-strategy Trust Evolution Model for Cognitive relay Network Based on Moran Process[J]. Journal of Chinese Computer Systems, 2014, 35(10): 2209-2214. (in Chinese)
邹世辰，王慧强，冯光升，等. 基于 Moran 过程的认知中继网络多策略信任演化模型[J]. 小型微型计算机系统，2014，35(10)：2209-2214.

[6] SCHROEDER B, GIBSON G. Disk failures in the real world: What does an mttf of 1000000 hours mean to you? [C]// Proc. of USENIX FAST. 2007.

[7] SONG Y, ZHANG F, SHAO Y, et al. Energy efficiency optimization of cognitive relay network based on cooperative spectrum sensing[J]. The Journal of China Universities of Posts and Telecommunications, 2015, 22(3): 26-34.

[8] BANG J, LEE J, KIM S, et al. An Efficient Relay Selection Strategy for Random Cognitive Relay Networks [J]. IEEE Transactions on Wireless Commsunications, 2015, 14(3): 1555-1566.

[9] BORTHAKUR D. HDFS architecture guide [OL]. [2013-08-04]. http://hadoop. apache. org/docs/r1. 2. 1/hdfs_design. html.

[10] ASTERIS M, ASTERIS M, PAPAILIOPOULOS D, et al. XORing elephants: novel erasure codes for big data[J]. Proceedings of the VLDB Endowment, 2013, 6(5): 325-336.

[11] 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.

[12] DIMAKIS A G, BRIGHTEN P, WAINWRIGHT M J, et al. The Benefits of Network Coding for Peer-to-Peer Storage Systems [C]// Proceedings of the International Conference on Computer Communications (INFOCOM). 2007.

[13] WANG Y, YIN X, WANG X. Two New Classes of Two-Parity MDS Array Codes With Optimal Repair[J]. IEEE Communications Letters, 2016, 20(7): 1293-1296.

[14] SHEN Z, LEE P P C, SHU J, et al. Encoding-Aware Data Placement for Efficient Degraded Reads in XOR-Coded Storage Systems[C]// Reliable Distributed Systems. IEEE, 2016: 239-248.

[15] YE L, FENG D, HU Y, et al. Hybrid-RC: Flexible Erasure Codes with Optimized Recovery Performance and Low Storage Overhead[C]// Reliable Distributed Systems. IEEE, 2017: 124-133.

[16] WANG J, LUO W, OUYANG M S, et al. Segmentation Coding Scheme Based on Simple Regenerating Codes [J]. Computer Science, 2016, 43(8): 148-153. (in Chinese)
王静，罗威，欧阳明生，等. 基于简单再生码的分段编码方案[J]. 计算机科学，2016，43(8)：148-153.

[17] AKHLAGHI S, KIANI A, GHANAVATI M R. Cost-bandwidth tradeoff in distributed storage systems [J]. Computer Communications, 2010, 33(17): 2105-2115.

[18] GERAMI M, XIAO M, SKOGLUND M. Optimal-cost repair in multi-hop distributed storage systems[C]// IEEE International Symposium on Information Theory Proceedings. IEEE Xplore, 2012: 1437-1441.

[19] LI S T, JIN X. Low-cost cloud storage scheme based on hybrid strategy[J]. Journal of Computer Applications, 2014, 34(10): 2800-2805. (in Chinese)
李松涛，金欣. 基于混合策略的低成本云存储方案[J]. 计算机应用，2014，34(10)：2800-2805.

[20] DU Y, XIONG R, JIN J, et al. A Cost-Efficient Data Placement Algorithm with High Reliability in Hadoop[C]// International Conference on Advanced Cloud & Big Data. IEEE Computer Society, 2017: 100-105.

[21] YU Q, SHUM K W, CHI W S. Minimization of Storage Cost in Distributed Storage Systems with Repair Consideration[C]// Global Telecommunications Conference. IEEE, 2012: 1-5.

[22] YU Q, CHI W S, CHAN T H. Irregular Fractional Repetition Code Optimization for Heterogeneous Cloud Storage[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(5): 1048-1060.

[23] LI J, YANG S, WANG X, et al. Tree-structured data regeneration in distributed storage systems with regenerating codes[C]// Proc of the IEEE INFOCOM. Piscataway, NJ: IEEE, 2010: 1-9.

[24] XIA M, SAXENA M, BLAUM M, et al. A tale of two erasure codes in HDFS[C]// Usenix Conference on File and Storage Technologies. USENIX Association, 2015: 213-226.

[25] CORMEN T H, LEISERSON C E, RIVEST R L, et al. Introduction to Algorithms (Second Edition)[M]. DBLP, 2001.

[26] GONG Q, WANG J, WANG Y, et al. Topology-Aware Node Selection for Data Regeneration in Heterogeneous Distributed Storage Systems[J]. arXiv preprint arXiv: 1506. 05579, 2015.

[27] SHAH N B, RASHMI K V, KUMAR P V. A flexible class of regenerating codes for distributed storage[C]// IEEE International Symposium on Information Theory. IEEE, 1947: 1943-1947.

[28] LI J, YANG S, WANG X, et al. Tree-structured data regeneration with network coding in distributed storage systems[C]// International Workshop on Quality of Service. IEEE, 2009: 2892-2900.