引用格式：张洁薇, 王静, 杨红志, 等. 具有较低修复带宽开销的局部 Piggybacking 码构造 [J]. 电子科技大学学报, 2025, 54(4): 521-531.
ZHANG J W, WANG J, YANG H Z, et al. Construction of locally Piggybacking codes with lower repair bandwidth overhead[J]. Journal of University of Electronic Science and Technology of China, 2025, 54(4): 521-531.

具有较低修复带宽开销的局部 Piggybacking 码构造

张洁薇[1]，王 静[1*]，杨红志[1]，李 瞳[1]，毛祖权[1]，刘向阳[2]

（1. 长安大学 信息工程学院，西安 710064；2. 西北工业大学 电子信息学院，西安 710129）

摘要：目前针对 Piggybacking 码的研究主要聚焦在单个信息节点的快速修复上，校验节点故障的修复带宽率依旧较高，且对多节点故障修复尚未提出快速修复算法。为此，该文提出一种局部 Piggybacking 码的构造方案，在 Piggybacking 框架的基础上，增加 2 个局部校验子条带，并将数据块错位放置，信息节点分组按顺序捎带，以此降低故障节点的修复带宽开销和修复度。性能分析表明，该文所构造的局部 Piggybacking 码，其信息节点和校验节点的修复带宽率都有显著降低，且与现有的 Piggybacking 码相比，修复度性能也有明显改善。

关键词：分布式存储系统；Piggybacking 码；修复带宽开销；修复度
中图分类号：TN911.2 **文献标志码：**A DOI: 10.12178/1001-0548.2024150

Construction of locally Piggybacking codes with lower repair bandwidth overhead

ZHANG Jiewei[1], WANG Jing[1*], YANG Hongzhi[1], LI Tong[1], MAO Zuquan[1], and LIU Xiangyang[2]

(1. School of Information Engineering, Chang'an University, Xi'an 710064, China;
2. School of Electronics and lnformation, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract: The current research on Piggybacking codes mainly focuses on the fast repair of single systematic nodes, the repair bandwidth rates of the failed parity nodes remain higher, and fast repair algorithm has not been proposed for multi-node fault repair. For this reason, this paper proposes a construction scheme of locally Piggybacking codes. On the basis of Piggybacking framework, two local parity sub-strips are added, the data blocks are placed in a staggered manner, and the systematic nodes are grouped in sequential Piggybacking, hereby reducing the repair bandwidth overhead and repair degree of the failed nodes. Performance analysis shows that the locally Piggybacking codes constructed in this paper have prominently reduced the repair bandwidth rate of both the systematic nodes and parity nodes. Moreover, compared with existing Piggybacking codes, the performance of the repair degree also has a significant improvement.

Key words: distributed storage system; Piggybacking codes; repair bandwidth overhead; repair degree

随着云计算的发展，数据海量化已经成为一种趋势[1]，分布式存储系统因其在数据存储方面的可靠性和高效性，近年来已经被各大互联网公司广泛使用[2]。然而，随着系统存储规模的增大，存储节点故障导致的数据失效越来越常见。在此背景下如何提高分布式存储系统的可靠性，采用性能更好的容错技术成为目前的研究热点。大部分容错技术都是通过增加冗余数据来提高系统的可靠性，产生冗余的方式最常见的有“复制”和“纠删码”策略[3-5]。复制即存储数据的多个副本，最常见的为三副本复制，存储开销较大。相比于复制，纠删码策略有效降低了存储开销，但由于其 (n, k) 性质，发生节点故障时，需下载原文件大小的数据量，修复带宽开销过大。为了降低故障节点的修复带宽开销，寻找数据节点存储开销和修复带宽开销的最优权衡，2010 年文献 [6] 使用信息流图将网络编码引入分布

收稿日期：2024−06−25
基金项目：国家自然科学基金（62001059）；陕西省重点研发计划项目（2024GX-YBXM-068）
作者简介：张洁薇，主要从事分布式存储编码方面的研究。
*通信作者 E-mail: jingwang@chd.edu.cn

式存储系统中，提出了节点存储开销与修复带宽开销的最优权衡曲线，进而提出最小存储再生（minimum storage regenerating, MSR）码和最小带宽再生（minimum bandwidth regenerating, MBR）码。相比纠删码，再生码能够有效降低故障节点的修复带宽，但对于 MBR 码来说，高码率意味着指数级的子条带数，计算复杂度较高。

针对上述问题，构造能对故障节点进行精确修复、高码率、低复杂度、低修复带宽开销的存储编码成为了研究热点。2013 年，文献 [7] 提出了 Piggybacking 码，并于 2017 对其进行了进一步完善，其编码满足 RBT （repair-by-transfer）属性[8-9]，即修复故障节点期间下载的数据量与读取的数据量相同，同时能实现数据的精确修复。文献 [10] 提出一种可以高效修复信息节点的 Piggybacking 码，即 REPB （repair efficient Piggybacking）码，信息节点的平均修复带宽率最低可达到$2/(\sqrt{r}+1)$，但是子条带数不确定。文献 [11] 通过对信息节点进行分组，构造了一种子条带数等于奇偶校验节点数的Piggybacking 码（shangguan Piggybacking, SPB），降低了修复带宽开销。但是上述 REPB 和 SPB 都只能对信息节点进行高效修复。

2019 年，文献 [12] 提出一种“one-to-one”的 Piggybacking 码（one-to-one Piggybacking, OOP），文献 [13] 提出了一种均衡分配的 Piggybacking 码，在进一步降低信息节点修复带宽的同时，考虑了奇偶校验节点的有效修复，但校验节点的修复性能依旧不佳。文献 [9] 构造的 RSR-I*、RSR-II*和文献 [10] 构造的 REPB*通过对子条带进行整合，将前一个子条带中校验节点的数据块捎带到后一个子条带的校验节点中，虽然提高了校验节点的修复性能，但导致子条带数目成倍增长，影响到系统的 I/O 性能。2020 年，文献 [14] 通过牺牲容错能力，将信息节点和校验节点进行分层修复，提出双层 Piggybacking （double-layered Piggybacking, DPB）编码，文献 [15] 在 SPB 的基础上进一步对校验节点进行了捎带，但对故障节点的修复度都没有明显改善。2022 年，文献 [16] 将子条带数与校验节点数设为相等值，以此为基础，利用 OOP 未充分利用到的校验节点进行捎带，进一步降低了修复带宽开销。

针对目前 Piggybacking 码存在的局限性，本文在 Piggybacking 框架中加入 2 个局部校验子条带，构造了一种具有较低修复带宽开销的局部 Piggybacking 码。本文构造的局部 Piggybacking 码保留了 MDS 码的优点，加入局部特性，使得故障节点的修复带宽开销更小，同时构造 Piggyback 函数捎带对应的信息节点数据块，在保证其 MDS 容错性的同时，提出了一种多节点故障的快速修复算法。

1 基础知识

1.1 Piggybacking 框架

Piggybacking 框架是一种在现有码的基础上，以现有码为基础码，将部分子条带的数据块经过线性组合形成 Piggyback 函数，之后将 Piggyback 函数嵌入，以设计具有高码率、低子条带数、低修复带宽开销的 Piggybacking 码。基于存储效率、计算复杂度等多方面考虑，现有的 Piggybacking 码通常选取 MDS 码为基础码，同时为了保证其 MDS 属性，在捎带时只选取子条带之前的数据块形成 Piggyback 函数。图 1 为采用 MDS 码为基础码的 Piggybacking 框架中校验节点结构，其中，$\boldsymbol{a}_i(1\leqslant i\leqslant\alpha)$代表第 i 个子条带上存储的信息数据块，$\boldsymbol{P}_j^{\mathrm{T}}\boldsymbol{a}_i(1\leqslant j\leqslant r)$为校验数据块，$g_{i,j}(\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_{i-1})$ $(i\neq 1)$为捎带的 Piggyback 函数。

节点k+1	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_1$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_2+g_{2,1}(\boldsymbol{a}_1)$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_3+g_{3,1}(\boldsymbol{a}_1,\boldsymbol{a}_2)$	…	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_\alpha+g_{\alpha,1}(\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_{\alpha-1})$
节点k+2	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_1$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_2+g_{2,2}(\boldsymbol{a}_1)$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_3+g_{3,2}(\boldsymbol{a}_1,\boldsymbol{a}_2)$	…	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_\alpha+g_{\alpha,2}(\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_{\alpha-1})$
⋮	⋮	⋮	⋮	⋮	⋮
节点k+r	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_1$	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_2+g_{2,r}(\boldsymbol{a}_1)$	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_3+g_{3,r}(\boldsymbol{a}_1,\boldsymbol{a}_2)$	…	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_\alpha+g_{\alpha,r}(\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_{\alpha-1})$

图 1 Piggybacking 框架中校验节点结构

当出现故障节点时，通过 MDS 码译码以及其中嵌入的 Piggyback 块进行修复即可。这一修复方式通过将部分 MDS 译码替换为 Piggyback 块译码减少了修复带宽开销。具体的衡量指标包括信息节

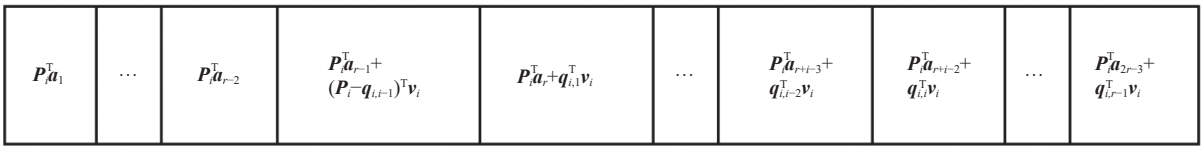
点平均修复带宽率γ^{sys}，校验节点平均修复带宽率γ^{par}，信息节点平均修复度率$\Im^{\mathrm{sys}}$以及校验节点平均修复度率$\Im^{\mathrm{par}}$。定义B_i为节点i故障时需要下载的数据块数目，定义η_i为节点i故障时需要连接的节点数目，通用公式表示如下：

$$\gamma^{\mathrm{sys}}=\frac{\sum_{i=1}^{k}B_i}{kk\alpha}\quad \gamma^{\mathrm{par}}=\frac{\sum_{i=k+1}^{k+r}B_i}{rk\alpha},$$

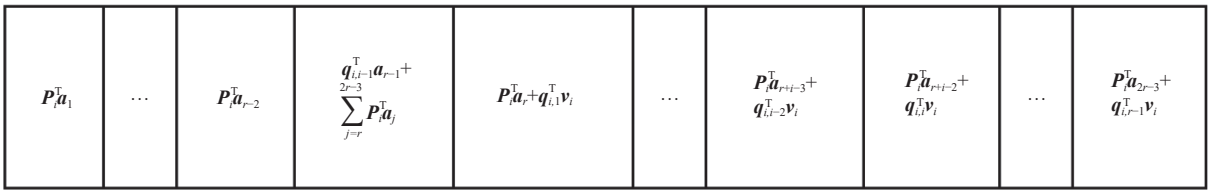
$$\Im^{\mathrm{sys}}=\frac{\sum_{i=1}^{k}\eta_i}{kk}\quad \Im^{\mathrm{par}}=\frac{\sum_{i=k+1}^{k+r}\eta_i}{rk}$$

1.2　RSR-II 码

RSR-II 码以 MDS 码为基本码[9]，其中信息节点的个数 k 可以取任意值，校验节点的个数$r\geqslant 3$，子条带的个数$\alpha=2r-3$，$\boldsymbol{P}_i^{\mathrm{T}}\boldsymbol{a}_n$为校验数据块$(1\leqslant n\leqslant\alpha)$，由 k 个信息数据块进行 MDS 编码生成。RSR-II 码的基本思想为将前$(r-1)$个子条带中信息节点的数据块捎带到后$(r-1)$个子条带校验节点的数据块中，其中$\boldsymbol{q}_{i,j}$, $\boldsymbol{v}_i$和$\hat{\boldsymbol{v}}_i$, $(2\leqslant i\leqslant r, 1\leqslant j\leqslant r-1)$共同构成 Piggyback 函数，且$\sum_{j=1}^{r-1}\boldsymbol{q}_{i,j}=\boldsymbol{P}_i, \forall i\in\{2,3,\cdots r\}$，校验节点$i\ (2\leqslant i\leqslant r)$的存储情况如图 2a 所示。将所有信息节点的前$(r-1)$个子条带中的数据块分为大小相等的集合，分别形成 Piggyback 函数，捎带到校验节点$i\ (2\leqslant i\leqslant r)$的后$(r-1)$个子条带中，同时满足每个子条带只能捎带之前子条带中的数据块，以保证 MDS 属性。在后$(r-1)$个校验节点中进行线性变换，用第$(r-1)$个子条带的数据块减去最后$(r-2)$个子条带中的数据块，得到如图 2b 所示的 RSR-II 编码结构。



a. RSR-II添加Piggyback函数



b. RSR-II进行可逆线性变换

图 2　RSR-II 编码结构

2　局部 Piggybacking 码的构造

2.1　具有较低修复带宽开销局部 Piggybacking 码的一般结构

图 3 为局部 Piggybacking 码的一般结构图，包括 A、B、C 共 3 个区域，其中区域 A 为信息数据块区域，区域 B 为全局校验块区域，区域 C 为局部检验块区域。将区域 A 中的数据块通过 Piggyback 函数形成 Piggyback 块，嵌入到区域 B，区域 C 由区域 A 和区域 B 中的数据块通过分组生成局部校验块。设$a_{m,n}(1\leqslant m\leqslant k, 1\leqslant n\leqslant\alpha)$代表第 m 个节点的第 n 个子条带中存储的数据块，$\boldsymbol{a}_h=(a_{1,h},a_{2,h},\cdots,a_{k,h})^{\mathrm{T}}$，$1\leqslant h\leqslant 2r-1$为子条带 h 中存储的信息数据块，$\boldsymbol{P}_1,\boldsymbol{P}_2,\cdots,\boldsymbol{P}_r$表示 r 个长度为 k 的 MDS 编码向量。局部 Piggybacking 码的总条带数$\alpha=2r-1, r\geqslant 3$，其具体编码步骤如下：1）预留两个空闲子条带作为局部校验条带，即第 r-1 个和第 $2r$−1 个子条带。将信息节点的前 r−2 个子条带和第 r+1 到第 $2r$−2 个子条带分为两个局部组，生成局部校验块，分别存储在第 r−1 和第 $2r$−1 个子条带中。将第 r 个子条带的信息符号分为 2 部分，第 1 部分为前$\lceil k/2\rceil$个信息节点，第 2 部分为其余信息节点，分别生成局部校验块存放在第 1 个校验节点的第 r−1 个子条带和第 $2r$−1 个子条带，即图 3 中x_1和x_2所在位置。2）将信息节点的 2 个局部组以及对应的局部校验条带内的数据块进行循环错位放置，之后将前 r−2 个子条带中的信息数据块按照捎带规则捎带到后 r−1 个校验节点的第 r 到第 $2r$−2 个子条带中。3）具体捎带规则如下。

① 将信息节点分为 t+1 个不相交的集合A_x，前 t 个集合的大小为 r−1，第 t+1 个集合的大小为$k-t(r-1)$，即：

$$|A_x|=r-1\quad x=1,2,\cdots,t$$

$$|A_x| = y, x = t+1 \quad k = t(r-1)+y$$

②定义长度为k的向量$\{\boldsymbol{q}_{j,h}\}_{j=1,h=1}^{r-2,r-2}$和$\{\boldsymbol{g}_{j,h}\}_{j=r-1,h=1}^{r-1,r-2}$，其中，$j$代表集合$A_x$中第$j$个信息节点，$h$代表第$h$个子条带，即：

$$\boldsymbol{q}_{j,h} = [\underbrace{0\cdots01}_{j-1}\underbrace{0\cdots01}_{r-2}\underbrace{0\cdots01}_{r-2}0\cdots]^{\mathrm{T}}$$

$$\boldsymbol{g}_{j,h} = [\underbrace{0\cdots01}_{r-2}\underbrace{0\cdots01}_{r-2}\underbrace{0\cdots01}_{r-2}0\cdots]^{\mathrm{T}}$$

③定义 Piggyback 函数$f_{j,h} = \boldsymbol{q}_{j,h}^{\mathrm{T}}\boldsymbol{a}_h (1 \leqslant j,h \leqslant r-2)$，将$f_{j,h}$附加到第$j$+2 个校验节点的第$h$+$r$个子条带。令 Piggyback 函数$f_h = \boldsymbol{g}_{j,h}^{\mathrm{T}}\boldsymbol{a}_h$，将$f_h$附加到第$h$+2 个校验节点的第$r$个子条带。

④最后，将除第 1 个以外校验节点的前r−2 个子条带和第r+1 到第 2r−2 个子条带分别作为 1 个局部组，生成局部校验块，局部组内部整体进行错位放置。

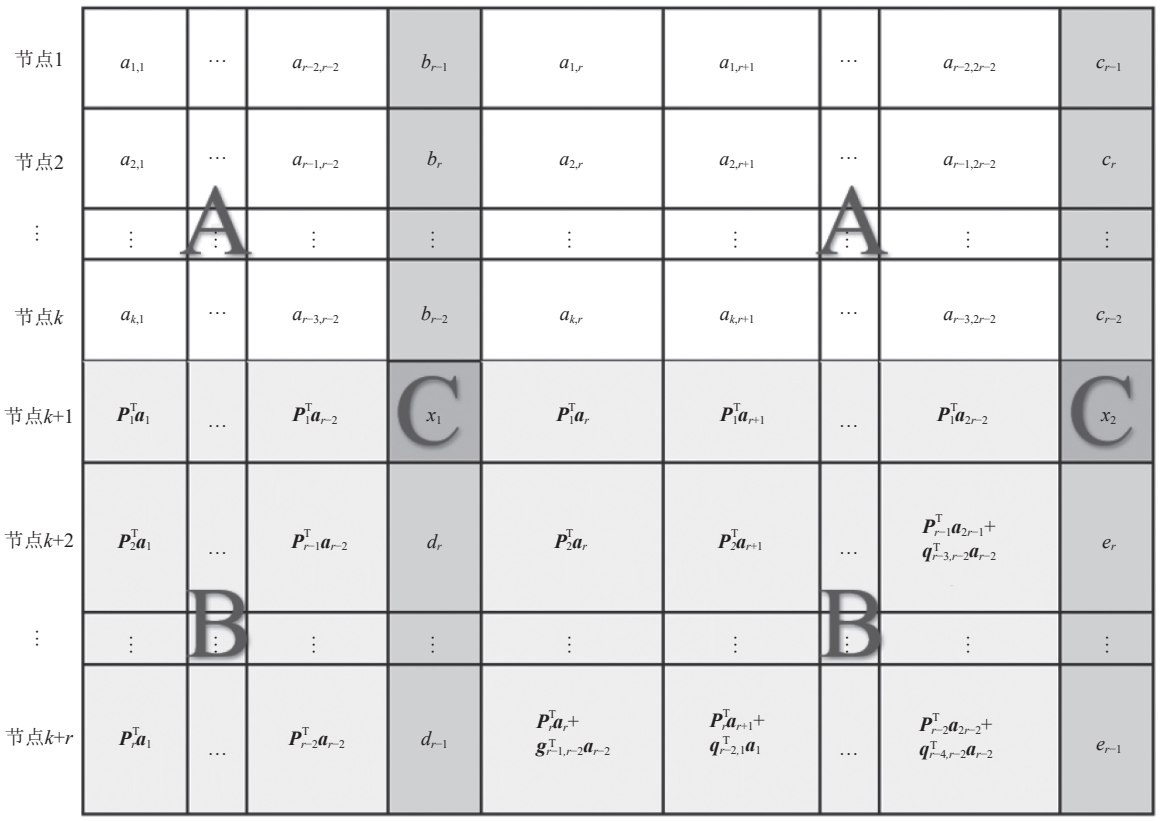


图 3 局部 Piggybacking 码的一般结构

图 3 为具有较低修复带宽开销的局部 Piggybacking 码的一般性结构图。以 (15, 10) MDS 码作为基本码，子条带数目$\alpha = 2r-1 = 9$为例。将信息节点和校验节点分为 3 个区域，进行编码生成局部校验块，按照上述捎带规则在多个区域间对信息符号进行捎带。具体操作如下：首先，将信息节点除预留条带外的前 3 个条带和后 3 个条带作为两个局部组，将信息符号按行生成局部校验块，存入预留的 2 个局部校验条带中。将第 5 个条带的信息符号分为两部分，分别生成局部校验块存入x_1和x_2所在位置。其次，将 2 个局部组和对应的局部校验条带内的所有数据块进行循环错位放置，之后将 10 个信息节点分为 3 组，分别为{1,2,3,4}，{5,6,7,8}，{9,10}。将第 1,2,3 条带与第 6,7,8 条带一一对应。将每组的第 1 个节点的信息符号捎带到第 3 个校验节点中，依次向剩余节点中捎带信息符号，然后将每组的第 4 个信息节点的信息符号捎带到后 3 个校验节点的第 5 个子条带中。最后，将第 2,3,4,5 个校验节点中 1,2,3 条带及 6,7,8 条带分为 2 个局部校验组生成局部校验块，存入剩余空闲子条带中，再将 2 个局部校验组内所有数据块及生成的局部校验块进行循环错位放置。最终构造如图 4 所示。

以下是该局部 Piggybacking 码的故障节点修复过程：1）单信息节点故障：若节点 1 发生故障，除第r个子条带之外的信息块可通过连接节点

2,3,4,8,9,10，从这些节点中读取并下载下标相同的数据块，共 24 个数据块进行修复。第r个子条带的信息块d_1可通过下载数据块d_2,d_3,d_4,d_5,x_1进行修复，并额外连接了节点 5 和 11。节点 1 的修复一共下载 29 个数据块，连接 8 个节点。2）单校验节点故障：若节点 14 发生故障，前 4 个数据块和后 4 个数据块通过连接节点 12,13,15 下载 24 个数据块进行修复，第 5 个数据块通过下载第r个子条带的所有信息数据块以及被嵌入到该故障节点的第 5 个数据块中 Piggyback 块所涉及的数据块共 12 个数据块进行修复。修复过程中共连接 13 个节点。3）多节点故障：若节点 1,3 同时故障，因节点 1 和 3 在一个局部组中存在 2 个下标相同的数据块，所以先下载第 5,6,7 个子条带中未失效的信息数据块和未嵌入 Piggyback 块的校验数据块，之后通过节点中第 6 和第 7 个子条带的 Piggyback 块修复部分失效数据块，其余数据块利用局部校验条带进行修复。

节点1	a_1	b_2	c_3	u_4	d_1	e_1	f_2	g_3	v_4
节点2	a_2	b_3	c_4	u_5	d_2	e_2	f_3	g_4	v_5
节点3	a_3	b_4	c_5	u_6	d_3	e_3	f_4	g_5	v_6
节点4	a_4	b_5	c_6	u_7	d_4	e_4	f_5	g_6	v_7
节点5	a_5	b_6	c_7	u_8	d_5	e_5	f_6	g_7	v_8
节点6	a_6	b_7	c_8	u_9	d_6	e_6	f_7	g_8	v_9
节点7	a_7	b_8	c_9	u_{10}	d_7	e_7	f_8	g_9	v_{10}
节点8	a_8	b_9	c_{10}	u_1	d_8	e_8	f_9	g_{10}	v_1
节点9	a_9	b_{10}	c_1	u_2	d_9	e_9	f_{10}	g_1	v_2
节点10	a_{10}	b_1	c_2	u_3	d_{10}	e_{10}	f_1	g_2	v_3
节点11	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{b}$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{c}$	x_1	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{d}$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{e}$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{f}$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{g}$	x_2
节点12	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}$	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{b}$	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{c}$	y_5	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{d}$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{e}$	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{f}+b_2+b_6+b_{10}$	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{g}+c_4+c_8+c_2$	z_5
节点13	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{a}$	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{b}$	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{c}$	y_2	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{d}+a_4+a_8$	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{e}+a_1+a_5+a_9$	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{f}+b_3+b_7+b_1$	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{g}+c_5+c_9$	z_2
节点14	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{a}$	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{b}$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{c}$	y_3	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{d}+b_5+b_9$	$\boldsymbol{P}_4^{\mathrm{T}}\boldsymbol{e}+a_2+a_6+a_{10}$	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{f}+b_4+b_8$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{g}$	z_3
节点15	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{a}$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{b}$	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{c}$	y_4	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{d}+c_6+c_{10}$	$\boldsymbol{P}_5^{\mathrm{T}}\boldsymbol{e}+a_3+a_7$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{f}$	$\boldsymbol{P}_3^{\mathrm{T}}\boldsymbol{g}+c_3+c_7+c_1$	z_4

图 4　(15,10) 局部 Piggybacking 码

2.2　特殊结构

图 5 为具有较低修复带宽开销的局部 Piggybacking 码的特殊结构图，即当 k 可以整除 $r-1$ 时，此时令 $t+1=k/(r-1)$，$|A_x|=r-1, x=1,2,\cdots,t+1$。

与局部 Piggybacking 码的一般结构相比，此构

造第 1 步对信息节点进行分组，第 2 步生成局部校验块，其余构造步骤与一般结构相同，如图 5 所示。在此结构中，每个集合的数据块和局部校验块在该集合进行错位循环放置。当节点故障时，仅需要在故障节点所在集合进行修复，无须连接其他集合的节点，可以在一定程度上减少信息节点的修复度及多个信息节点故障的修复复杂度。

节点1	$a_{1,1}$	…	$a_{r-2,r-2}$	b_{r-1}	$a_{1,r}$	$a_{1,r+1}$	…	$a_{r-2,2r-2}$	c_{r-1}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
节点$r-1$	$a_{r-1,1}$	…	$a_{r-3,r-2}$	b_{r-2}	$a_{r-1,r}$	$a_{r-1,r+1}$	…	$a_{r-3,2r-2}$	c_{r-2}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
节点k	$a_{k,1}$	…	$a_{k-2,r-2}$	b_{k-3}	$a_{k,r}$	$a_{k,r+1}$	…	$a_{k-2,2r-2}$	c_{k-3}
节点$k+1$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_1$	…	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_{r-2}$	x_1	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_r$	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_{r+1}$	…	$\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{a}_{2r-2}$	x_2
节点$k+2$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_1$	…	$\boldsymbol{P}_{r-1}^{\mathrm{T}}\boldsymbol{a}_{r-2}$	d_r	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_r$	$\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{a}_{r+1}$	…	$\boldsymbol{P}_{r-1}^{\mathrm{T}}\boldsymbol{a}_{2r-2}+$ $\boldsymbol{q}_{r-3,r-2}^{\mathrm{T}}\boldsymbol{a}_{r-2}$	e_r
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
节点$k+r$	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_1$	…	$\boldsymbol{P}_{r-2}^{\mathrm{T}}\boldsymbol{a}_{r-2}$	d_{r-1}	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_r+$ $\boldsymbol{g}_{r-1,r-2}^{\mathrm{T}}\boldsymbol{a}_{r-2}$	$\boldsymbol{P}_r^{\mathrm{T}}\boldsymbol{a}_{r+1}+$ $\boldsymbol{q}_{r-2,1}^{\mathrm{T}}\boldsymbol{a}_1$	…	$\boldsymbol{P}_{r-2}^{\mathrm{T}}\boldsymbol{a}_{2r-2}+$ $\boldsymbol{q}_{r-4,r-2}^{\mathrm{T}}\boldsymbol{a}_{r-2}$	e_{r-1}

图 5 局部 Piggybacking 码的特殊结构

以 (13, 8) MDS 码作为基本码，子条带数目 $\alpha=2r-1=9$为例，将 8 个信息节点分为 2 组，分别为{1,2,3,4}，{5,6,7,8}。之后，将 2 个局部组和对应局部校验条带内的所有数据块进行循环错位放置。在 $k=8,r=5$的一般结构中，信息节点 1 故障，修复时需连接节点 2,3,4,6,7,8,9 进行数据块的读取和下载，修复度为 7；而在特殊结构中，当信息节点 1 故障时，仅需连接节点 2,3,4,9 即可修复故障节点，修复度仅为 4。

3 修复分析

3.1 单节点故障修复

局部 Piggybacking 码在单个信息节点故障时，统一利用局部校验块进行修复，在单个校验节点故障时，利用 MDS 性质和局部校验块进行混合修复。信息节点 i（$1\leqslant i\leqslant k$）故障的修复算法如下：1）下载信息节点中除第 r 个子条带以外的所有条带中与故障信息节点 i 下标相同的数据块，通过局部校验块进行修复；2）若$i\leqslant\left\lceil\frac{k}{2}\right\rceil$，需下载第 r 个子条带前$\left\lceil\frac{k}{2}\right\rceil$个数据块中未失效的数据块及$x_1$，通过异或修复信息节点 i 的第 r 个条带中失效的数据块；否则下载后$k-\left\lceil\frac{k}{2}\right\rceil$个数据块中未失效数据块及$x_2$通过异或进行修复。

定理 1 局部 Piggybacking 码在单信息节点故障时的平均修复带宽率为：

$$\gamma^{\mathrm{sys}}=\frac{2k(r-2)(r-1)+\left\lceil\frac{k}{2}\right\rceil^2+\left\lfloor\frac{k}{2}\right\rfloor^2}{k^2(2r-3)} \tag{1}$$

证明：设第 i 个信息节点故障需要下载的数据块为 B_i，若 $i \leqslant \left\lceil \frac{k}{2} \right\rceil$，需下载 $2(r-2)(r-1)+\left\lceil \frac{k}{2} \right\rceil$ 个数据块，否则需下载 $2(r-2)(r-1)+\left\lfloor \frac{k}{2} \right\rfloor$ 个数据块。共 k 个信息节点，原始数据块总量为 $k(2r-3)$，所以：

$$\gamma^{\text{sys}} = \frac{\sum_{i=1}^{k} B_i}{k^2(2r-3)} = \frac{\left[2(r-2)(r-1)+\left\lceil \frac{k}{2} \right\rceil\right]\left\lceil \frac{k}{2} \right\rceil + \left[2(r-2)(r-1)+\left\lfloor \frac{k}{2} \right\rfloor\right]\left\lfloor \frac{k}{2} \right\rfloor}{k^2(2r-3)} = \frac{2k(r-2)(r-1)+\left\lceil \frac{k}{2} \right\rceil^2+\left\lfloor \frac{k}{2} \right\rfloor^2}{k^2(2r-3)}。$$

假设校验节点故障，修复算法如下：1）第 1 个校验节点故障，通过 MDS 性质修复，即下载所有原始信息数据块修复；2）其余校验节点故障，利用校验节点的局部校验块和 MDS 属性进行混合修复。

定理 2　局部 Piggybacking 码在单校验节点故障时的平均修复带宽率为：

$$\gamma^{\text{par}} = \frac{2k(r-1)+[2(r-1)^2+k+m](r-2)}{kr(2r-3)} \tag{2}$$

式中，$m = \begin{cases} t+1, & k=(t+1)(r-1) \\ t, & k \neq (t+1)(r-1) \end{cases}$。

证明：设第 i 个校验节点，即第 $k+i$ 个节点故障需要下载的数据块为 B_i。节点 k+1 故障，根据 MDS 性质，需要下载 $k(2r-3)$ 个数据块。其余校验节点故障，第 r 个条带以外的数据块利用局部校验块进行修复，一个节点需下载 $2(r-1)(r-2)$ 个数据块进行修复，共 r–1 个节点。第 r 个条带的修复需根据 MDS 性质及 Piggybacking 框架的性质进行修复，共需下载 $k+(k+m)(r-2)$ 个数据块，其中 $m = \begin{cases} t+1, & k=(t+1)(r-1) \\ t, & k \neq (t+1)(r-1) \end{cases}$。所以：$\gamma^{\text{par}} = \frac{\sum_{i=k+1}^{k+r} B_i}{kr(2r-3)} = \frac{k(2r-3)+2(r-1)^2(r-2)+k+(k+m)(r-2)}{kr(2r-3)} = \frac{2k(r-1)+[2(r-1)^2+k+m](r-2)}{kr(2r-3)}$。

3.2　多节点故障修复

考虑到存储系统中除单节点故障外，2 节点故障占比较大，3 个或更多节点故障占比较小[17]，所以本小节主要分析 2 节点故障的快速修复。对于多节点故障，当故障节点数大于等于 3 时，使用 MDS 码译码修复。当故障节点数为 2 时，分为以下 3 种情况：1）2 个故障节点都为校验节点时，使用 MDS 码译码；2）2 个故障节点为 1 个信息节点和 1 个校验节点时，修复方式同单节点修复方式；3）2 个故障节点都为信息节点时，若 2 个故障节点不存在下标相同的数据块，采用局部校验块修复；若 2 个故障节点在 1 个局部组中存在 j 个下标相同的数据块，$j \in \{1,2,\cdots,r-2\}$，具体修复方式如下：①先下载第 r 到第 $r+j$ 个子条带中未失效信息数据块以及 2 个未附加其他信息符号的校验块，设相同下标位于前 j 个子条带的故障节点为故障节点 1。②优先修复故障节点 2 的第 1 个失效数据块，之后故障节点 1 的前 j 个失效数据块利用 Piggyback 函数进行修复。③剩余失效数据块利用局部校验块进行修复。

定理 3　设 $Z = C_k^2$，$S_1 = \{[k-2(r-2)-1]k\}/2$，$j \in \{1,2,\cdots,r-2\}$，$k=(r-1)t+y, 0<y \leqslant r-1$，$Q_1 = 4(r-1)(r-2)+k$，$Q_2 = (j+1)k+2(r-1-j)(r-2-j)+j(t+1)+j(r-3)+2(r-1-j)(r-2)$。

则 2 信息节点故障的最大修复带宽率：

$$\gamma_{\max}^{\text{sys}} = \frac{Q_1 S_1 + Q_2(Z-S_1)}{Zk(2r-3)} \tag{3}$$

证明：设两信息节点故障，共有 Z（$Z = C_k^2$）种可能。

1）2 信息节点无下标相同数据块，共有 S_1 种可能，此时修复 2 个故障信息节点的数据下载量为：$Q_1 = 4(r-1)(r-2)+k$。

2）2 信息节点中含有 $2j$ 个数据块下标相同，$j \in \{1,2,\cdots,r-2\}$，共有 $Z-S_1$ 种可能。因具体修复过程中利用 MDS 修复的数据块和利用局部校验修复的数据块在下载过程中存在对某些数据块的重复下载，而实际情况中不需要对下载过的数据块重复下载，所以本节不考虑重复情况，给出最大修复带宽：令 $k=(r-1)t+y, 0<y \leqslant r-1, Q_2 = (j+1)k+2(r-1-j)(r-2-j)+j(t+1)+j(r-3)+2(r-1-j)(r-2)$，所以：$\gamma_{\max}^{\text{sys}} = \frac{Q_1 S_1 + Q_2(Z-S_1)}{Zk(2r-3)}$。

3.3　平均修复度率

对于单信息节点故障，考虑第 r 个子条带的修复方式与其他子条带的修复方式不同，在 $r \neq 3$ 且 $k \geqslant 2(r-1)$ 的情况下，得到信息节点的修复度率如下。

定理 4　局部 Piggybacking 码信息节点的修复

度率为：

$$\mathfrak{J}^{\text{sys}}=\begin{cases}\dfrac{2(k+r-2)(r-1)+k\left(\dfrac{k}{2}-2r+2\right)}{k^2} & k\text{为偶数且}\dfrac{k}{2}\geqslant 2(r-2)+1\\ \dfrac{2\left[\left(3r-5+\dfrac{k}{2}\right)\left(\dfrac{k}{2}-r+2\right)+\left(2r-\dfrac{k}{2}-4\right)(2r-3)\right]}{k^2} & k\text{为偶数且}\dfrac{k}{2}<2(r-2)+1\\ \dfrac{2(r-1)(k+r-2)+\dfrac{k+1}{2}\left(\dfrac{k+1}{2}-2r+2\right)+\dfrac{k-1}{2}\left(\dfrac{k-1}{2}-2r+2\right)}{k^2} & k\text{为奇数且}\lfloor\dfrac{k}{2}\rfloor\geqslant 2(r-2)+1\\ \dfrac{(k+6r-9)(k-2r+5)+(k+6r-11)(k-2r+3)+(4r-6)(8r-2k-16)}{4k^2} & k\text{为奇数且}\lceil\dfrac{k}{2}\rceil\leqslant 2(r-2)+1\end{cases}\tag{4}$$

证明：当 k 为偶数且 $\frac{k}{2}\geqslant 2(r-2)+1$ 时，因结构中存在第 r 个子条带信息数据块失效时的修复方式与其他子条带不同，将信息节点平均分为 2 部分（奇数时 2 部分的修复度最大最小值不一样），在信息节点中，存在修复度最小为 $\frac{k}{2}$，修复度最大为 $\frac{k}{2}+r-2$，整个过程一共包含 $4\left[\left(\frac{k}{2}+r-2\right)-\left(\frac{k}{2}\right)+1\right]$ 个节点。所以：$\mathfrak{J}^{\text{sys}}=\frac{\left(\frac{k}{2}+\frac{k}{2}+r-2\right)\left(\frac{k}{2}+r-2-\frac{k}{2}+1\right)2}{k^2}+\frac{2\left[\frac{k}{2}-2\left(\frac{k}{2}+r-2-\frac{k}{2}+1\right)\right]\frac{k}{2}}{k^2}=\frac{2(k+r-2)(r-1)+k\left(\frac{k}{2}-2r+2\right)}{k^2}$。

当 k 为偶数且 $\frac{k}{2}<2(r-2)+1$ 时，在信息节点中，存在修复度最小值为 $2(r-2)+1$，修复度最大值为 $\frac{k}{2}+r-2$，节点的修复度变化情况和上述情况相同，整个过程包含 $4\left[\left(\frac{k}{2}+r-2\right)-(2(r-2)+1)+1\right]$ 个节点。所以：$\mathfrak{J}^{\text{sys}}=\frac{2\left[2(r-2)+1+\frac{k}{2}+r-2\right]\left[\left(\frac{k}{2}+r-2\right)-[2(r-2)+1]+1\right]}{k^2}+\frac{2\left[\frac{k}{2}-2\left[\frac{k}{2}+r-2-[2(r-2)+1]+1\right]\right][2(r-2)+1]}{k^2}=\frac{2\left[\left(3r-5+\frac{k}{2}\right)\left(\frac{k}{2}-r+2\right)+\left(2r-\frac{k}{2}-4\right)(2r-3)\right]}{k^2}$。$k$ 为奇数的情况证明过程与上述同理。

定理 5 局部 Piggybacking 码校验节点的修复度率为：

$$\mathfrak{J}^{\text{par}}=\frac{k+(r-1)(k+r-2)}{kr}\tag{5}$$

证明：第 1 个校验节点故障，需连接前 k 个信息节点，下载所有信息数据块。其余节点，除第 r 个条带以外的数据块，访问除第 1 个校验节点以及故障节点以外的 $r-2$ 个校验节点，第 r 个条带中的数据块恢复需要访问前 k 个信息节点，所以局部 Piggybacking 码校验节点的修复度率为 $\mathfrak{J}^{\text{par}}=\frac{k+(r-1)(k+r-2)}{kr}$。

4 性能比较

4.1 存储开销

假设文件大小为 M，将 M 分为 αk 个未经编码的原始信息数据块（其中 α 为子条带数，k 为节点个数），所以框架中的每个数据块的大小均为 $M/\alpha k$，RSR-II 码、SPB 码等现有的 Piggybacking 码的存储开销为 $\alpha_1=M/k$。本文构造的局部 Piggybacking 码的存储开销为 $\alpha_2=\frac{(2r-1)M}{(2r-3)k}$，所以相比现有的几种 Piggybacking 编码，本文所构造的局部 Piggybacking 码的存储开销的增加率为 $2/2r-3$。图 6 为本文构造的局部 Piggybacking 码的存储开销增加率与子条带数目之间的关系。在仿真过程中，取 M=2 000 Mb，k=100, $10\leqslant\alpha\leqslant 50$。通过图 6 可以看出，因在本文构造中，一个显著的特点是引入的局部校验子条带数不随 k、r 的变化而变化，所以随着子条带数的增大，增加的局部校验块与原始信息数据块的比值越来越小。当子条带数目为 50 时，存储开销的增加率为 0.04，趋近于 0。所以条带数增加到一定量时，局部校验子条带带来的存储开销变得极其微小，几乎可以忽略不计，从而在改善修复带宽开销的同时，尽可能地保持了存储效率。

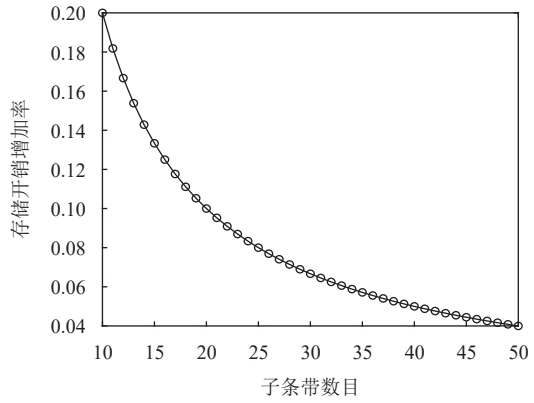


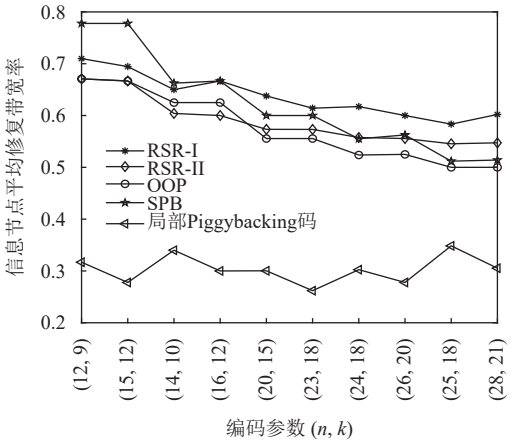
图 6　存储开销增加率

4.2　修复带宽率

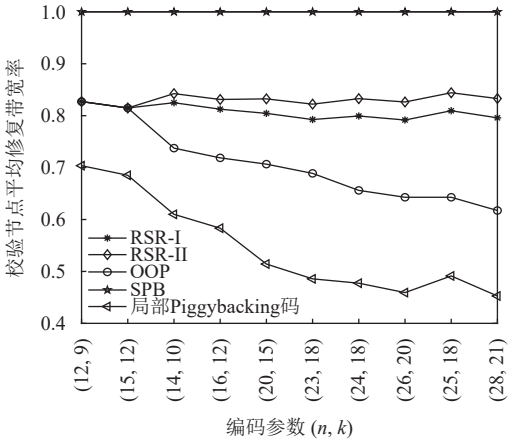
针对信息节点故障时的平均修复带宽率γ^{sys}和校验节点故障的平均修复带宽率γ^{par}，将本文构造的局部 Piggybacking 码与现有的 Piggybacking 码进行比较分析。为了更好地进行对比，设置 OOP 的条带数为$\lfloor \sqrt{r-1}+r-1 \rfloor$，在进行信息节点的平均修复带宽率对比时，设置 RSR-I 的子条带数为 2，RSR-II 的子条带数为$2r-3$，SPB 的子条带数为r。这样的设置旨在精选出 Piggybacking 码在性能上表现出色的情况，作为基准，与本文构造的 Piggybacking 码进行全方位的对比。RSR-I 和 RSR-II 信息节点的修复带宽率不随其结构的增加而改变，但其校验节点的修复带宽率会发生变化，所以在对比校验节点的平均修复带宽率时，设置 RSR-I 的子条带数为 6，RSR-II 的子条带数为$2(2r-3)$，而 SPB 并未对校验节点进行捎带，所以 SPB 校验节点平均修复带宽率为 1。

图 7 给出了节点的平均修复带宽率，可以看出，本文构造的局部 Piggybacking 码同时显著降低了信息节点和校验节点的修复带宽率，这是因为本文构造引入了局部修复组的思想。由图 7a 可以看出，本文构造的局部 Piggybacking 码在节点取值个数较小时，修复带宽率就相对较低。这是因为在信息节点修复过程中仅使用局部校验块进行修复。由图 7b 可知，本文构造的校验节点修复带宽率整体随节点个数的增加而降低。这是因为校验节点修复时虽然利用了局部的思想，但第 r 个子条带中的数据块的修复依旧取决于 MDS 译码和 Piggyback 函数。

图 8 给出了本结构在双信息节点故障时的修复带宽率。现有的 Piggybacking 算法并未提出 2 信息节点故障时的快速修复算法，默认其 2 节点故障修复时采用 MDS 修复，即修复带宽率为 1。本结构下的双信息节点采用顺序修复，在$r \neq 3$且$k > 2(r-1)$的情况下，修复过程利用 MDS 译码、Piggybacking 译码以及局部修复相结合。图 8 给出了双信息节点不存在数据块下标相同情况下的修复带宽率 Q1，及存在部分下标相同情况下的修复带宽率 Q2。为了便于比较，这里只选取编码参数 $j=r-2$ 时 Q2 的边界曲线。由图 8 可以看出，相比现有 Piggybacking 算法采用 MDS 译码，本文提出的局部 Piggybacking 码显著改善了双信息节点的修复带宽率性能。



a. 信息节点的平均修复带宽率



b. 校验节点的平均修复带宽率

图 7　平均修复带宽率

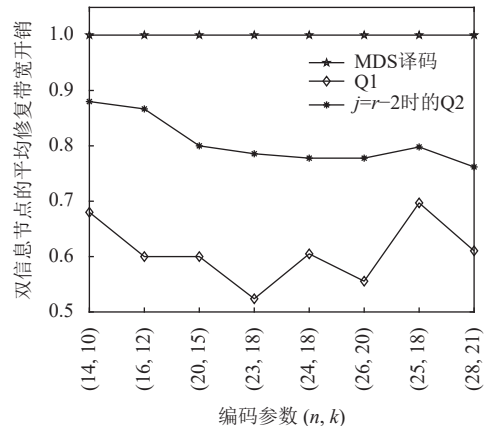


图 8　双信息节点修复带宽率

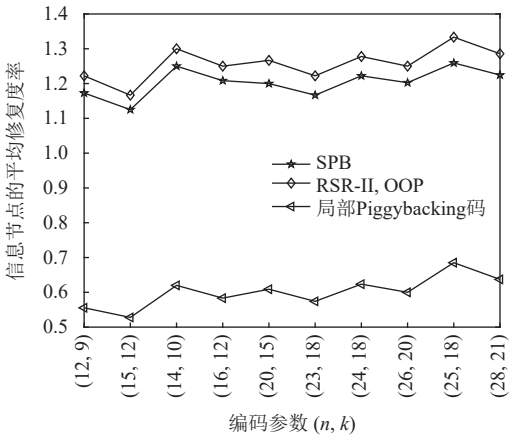
4.3 修复度率

表 1 给出了现有的几种 Piggybacking 码和本文构造的局部 Piggybacking 码关于子条带数和修复度率的对比，其中局部 Piggybacking 码信息节点修复度率由定理 4 给出。从表 1 可以看出，现有的 Piggybacking 码信息节点的平均修复度率与 k 成反比，与 r 成正比。

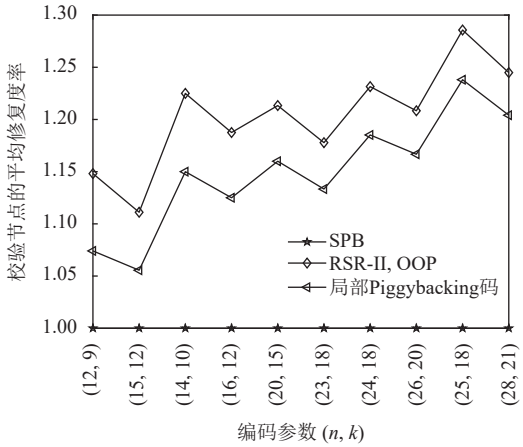
表 1 修复度率对比

Piggybacking编码	子条带数α	修复度率	
		信息节点$\mathfrak{I}^{\mathrm{sys}}$	校验节点$\mathfrak{I}^{\mathrm{par}}$
RSR-II	$(2r-3)\tau$	$\frac{k+r-1}{k}$	$\frac{k+(r-1)(k+r-1)}{rk}$
OOP	$\sqrt{r-1}+r-1$	$\frac{k+r-1}{k}$	$\frac{k+(r-1)(k+r-1)}{rk}$
SPB	r	$\frac{(k+r)k-\sum_{l=1}^{L} l\varphi_l}{k^2}$	1
局部Piggybacking码	$2r-1$	$\mathfrak{I}^{\mathrm{sys}}$	$\frac{k+(r-1)(k+r-2)}{kr}$

图 9 给出了信息节点和校验节点平均修复度率的仿真结果。与 SPB、RSR-II 以及 OOP 相比，本文所构造的局部 Piggybacking 码信息节点的平均修复度率显著降低，且最高降低了 61.7%。SPB 码的校验节点修复带宽率恒为 1，而本文所构造的局部 Piggybacking 码因校验节点的第 r 个子条带并未生成可以进行局部修复的校验块，所以每个校验节点故障修复时都需要连接所有的信息节点，以至于修复度率大于 SPB 码的修复度率，但与 RSR-II 码和 OOP 相比，其修复度率更低。



a. 信息节点的平均修复度率



b. 校验节点的平均修复度率

图 9 平均修复度率

5 结束语

分布式存储系统中 Piggybacking 编码作为一种实用方法，显著降低了故障节点的修复带宽开销。但现有 Piggybacking 码大部分聚焦在信息节点的快速修复，而校验节点及多节点故障的修复依旧存在修复带宽开销较大的问题。为此，本文引入局部修复的思想，增加 2 个局部校验条带，同时通过对信息节点进行分组，采用局部组内按序匹配的方式进行捎带，构造的局部 Piggybacking 码对信息节点的修复度率和修复带宽率都具有显著改善，且同时降低了校验节点的修复带宽率。

参 考 文 献

[1] SIDDIQA A, KARIM A, GANI A. Big data storage technologies: A survey[J]. Frontiers of Information Technology and Electronic Engineering, 2017, 18(8): 1040-1070.

[2] YAVARI E, ESMAEILI M. Locally repairable codes: Joint sequential-parallel repair for multiple node failures[J]. IEEE Transactions on Information Theory, 2019, 66(1): 222-232.

[3] LI J, LI B C. Demand-aware erasure coding for distributed storage systems[J]. IEEE Transactions on Cloud Computing, 2021, 9(2): 532-545.

[4] LEE O T, KUMAR S D M, CHANDRAN P. Erasure coded storage systems for cloud storage−challenges and opportunities[C]//IEEE International Conference on Data Science and Engineering (ICDSE). Cochin: IEEE, 2016: 1-7.

[5] PAPAILIOPOULOS D S, DIMAKIS A G, CADAMBE V R. Repair optimal erasure codes through hadamard designs[J]. IEEE Transactions on Information Theory, 2013, 59(5): 3021-3037.

[6] DIMAKIS A G, GODFREY P B, WU Y N, et al. Network coding for distributed storage systems[J]. IEEE

Transactions on Information Theory, 2010, 56(9): 4539-4551.

[7] RASHMI K V, SHAH N B, RAMCHANDRAN K. A piggybacking design framework for read-and download-efficient distributed storage codes[C]//Proceedings of IEEE International Symposium on Information Theory. Istanbul: IEEE, 2013: 331-335.

[8] SHAH N B, RASHMI K V, KUMAR P V, et al. Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth trade off[J]. IEEE Transactions on Information Theory, 2011, 58(3): 1837-1852.

[9] RASHMI K V, SHAH N B, RAMCHANDRAN K. A piggybacking design framework for readand download-efficient distributed storage codes[J]. IEEE Transactions on Information Theory, 2017, 63(9): 5802-5820.

[10] YUAN S, HUANG Q, WANG Z L. A repair-efficient coding for distributed storage systems under piggybacking framework[J]. IEEE Transactions on Communications, 2018, 66(8): 3245-3254.

[11] SHANGGUAN C, GE G N. A new piggybacking design for systematic MDS storage codes[J]. Designs, Codes and Cryptography, 2019, 87(12): 2753-2770.

[12] LI G Y, LIN X, TANG X H. An efficient one-to-one piggybacking design for distributed storage systems[J]. IEEE Transactions on Communications, 2019, 67(12): 8193-8205.

[13] 周悦, 李贵洋, 韩鸿宇, 等. 一种均衡分配的修复校验节点的 Piggybacks 捎带设计[J]. 电子学报, 2021, 49(4): 812-816.
ZHOU Y, LI G Y, HAN H Y, et al. A balanced-allocation Piggybacks adding design for repairing parity nodes[J]. Acta Electronica Sinica, 2021, 49(4): 812-816.

[14] SUN R, LI X, ZHANG L, et al. Distributed storage codes based on double-layered piggybacking framework[J]. IEEE Access, 2020, 8: 150447-150464.

[15] SUN R, ZHANG L, LIU J W. A new piggybacking design with low-repair bandwidth and complexity[J]. IEEE Communications Letters, 2021, 25(7): 2099-2103.

[16] SHI H, HOU H X, HAN Y S, et al. New piggybacking codes with lower repair bandwidth for any single-node failure[C]//The 2022 IEEE International Symposium on Information Theory (ISIT). Espoo: IEEE, 2022: 2601-2606.

[17] 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[C]//Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems. San Jose: USENIX Association, 2013: 1.

编　辑　刘飞阳