一种均衡分配的修复校验节点的 Piggybacks 捎带设计

周 悦,李贵洋,韩鸿宇,李 慧,胡金平
(四川师范大学计算机科学学院,四川成都 610101)

摘 要: 针对最初的减少校验节点修复带宽的 Piggybacks 捎带设计存在的问题,提出了一种均衡分配的 Piggybacks 捎带设计(Balanced-Allocation Piggybacks Adding,BAPA). 首先,通过分析给出了新的 Piggybacks 捎带规则,在此基础上得出了能进一步减少校验节点修复带宽的 Piggybacking 设计 BARSR-I 和 BARSR-II. 然后,给出了 BARSR-I 和 BARSR-II 中校验节点的修复过程以及平均修复带宽率的推导值. 最后,给出了 BARSR-I 和 BARSR-II 下的编码复杂度和修复复杂度. 通过与现有的 Piggybacking 设计对比分析表明,BARSR-I 和 BARSR-II 能有效的减少校验节点的修复带宽.

关键词: 分布式存储系统; Piggybacking 框架; 校验节点; RSR-I; RSR-II; 平均修复带宽率
中图分类号: TP302.7 **文献标识码:** A **文章编号:** 0372-2112 (2021)04-0812-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20200056

A Balanced-Allocation Piggybacks Adding Design for Repairing Parity Nodes

ZHOU Yue,LI Gui-yang,HAN Hong-yu,LI Hui,HU Jin-ping
(*Department of Computer Science,Sichuan Normal University,Chengdu,Sichuan* 610101,*China*)

Abstract: To address the problem of the original Piggybacks adding design which reduces the repair bandwidth of the parity nodes,we propose a balanced-allocation Piggybacks adding design (BAPA). Firstly,the new piggybacks adding rules are given through analysis,and on this base,the piggybacking design BARSR-I and BARSR-II which can further reduce the repair bandwidth of parity nodes,are given. Then,the repair process and theoretical derivation value of average repair bandwidth rate of parity nodes in BARSR-I and BARSR-II are given. Finally,the encoding complexity and repair complexity under BARSR-I and BARSR-II are given. The comparison analysis with the existing Piggybacking design shows that BARSR-I and BARSR-II can effectively reduce the repair bandwidth of the parity nodes.

Key words: distributed storage systems;piggybacking framework;parity nodes;RSR-I;RSR-II;average repair bandwidth rate

1 引言

2013 年,Rashmi 等人[1,2]提出了 Piggybacking 框架的概念,在不添加额外存储空间的情况下,不仅可以保留纠删码已有的优良性质,还能有效减少修复失效节点期间的修复带宽,并给出了几种具体的 Piggybacking 设计. 随后,其他的 Piggybacking 设计研究成果被相继提出[3~6].

文献[2]中的 Piggybacking 设计主要针对减少系统节点的修复带宽,为了进一步减少 RSR-I 和 RSR-II 中校验节点的修复带宽,Rashmi 等人提出了将实例数进行翻倍的 Piggybacking 设计,得到了能同时减少系统节点和校验节点修复带宽的 ORSR-I 和 ORSR-II. 值得指出的是,在实例数翻倍的 Piggybacking 设计中,最核心的是跨实例间如何捎带 Piggybacks,通过分析 RSR-I 和 RSR-II 中系统节点的修复过程发现,Rashmi 等人提出的 Piggybacks 捎带设计存在只考虑到了部分校验节点可以捎带 Piggybacks,并且捎带规则单一的问题. 针对此问题,本文在实例数翻倍的 Piggybacking 设计中捎带 Piggybacks 时,提出了一种均衡分配的 Piggybacks 捎带设计(Balanced-Allocation Piggybacks Adding,BAPA),得出了能同时减少系统节点和校验节点修复带宽的 Pig-

收稿日期:2020-01-02;修回日期:2020-10-10;责任编辑:李勇锋
基金项目:国家自然科学基金(No. 61701331)

gybacking 设计 BARSR-I 和 BARSR-II.

2　BAPA 设计

2.1　BAPA 原则

文献[2]中修复校验节点的捎带设计(OPA)只考虑到某些实例的第一个校验节点未参与任意系统节点的修复过程. 但是,通过反复确认任意系统节点的修复过程,发现了以下值得改进的问题.

问题 1　在任意系统节点的修复过程中,还存在其他的校验节点也未参与修复. 于是,可以同样利用将实例数翻倍后继续向这些校验节点中捎带 Piggybacks,来进一步减少校验节点的修复带宽.

问题 2　在 OPA 设计中,部分子实例中任意的校验节点修复都需要下载该子实例中全部的系统节点数据,并没有减少该子实例中校验节点的修复带宽. 所以,可以考虑将这些子实例中的校验节点数据作为 Piggybacks 捎带到后面的实例中子实例的校验节点上,以此来减少校验节点的修复带宽.

问题 3　由问题 1 和问题 2 可以看出,后面的实例中子实例的某些校验节点可以捎带 Piggybacks,而前面的实例中子实例的校验节点需要作为 Piggybacks 被捎带. 于是,这就产生了一个如何分配的问题,即如何捎带 Piggybacks 才能进一步减少所有校验节点的平均修复带宽率.

基于上面的问题,可以将 Piggybacks 的捎带规则看作一个数据分配问题. 可描述为:假设一共有 $s(s\geqslant 2)$ 个数据需要被分配,每个数据都可以被分配到 $t(1\leqslant t\leqslant s)$ 个集合中的一个,当某个集合中的某一个数据丢失时,需要下载该集合中其他所有未丢失的数据来修复丢失数据,修复代价即为数据的下载量,用符号 P 表示,则如何分配 s 个数据到 t 个集合中才能使得所有数据的平均修复代价最小. 基于该描述可以提出以下结论.

定理 1　s 个数据被分配到的集合数越多,平均修复代价越小.

证明　当 s 个数据可被分配的集合数为 1 时,此时的平均修复代价为:

$$P_1=\frac{s(s-1)}{s}$$

当 s 个数据可被分配的个集合数为 2 时,则将 s 个数据分配到这两个集合中的分配方式一共有 2^s 种,其中考虑修复代价最高的一种方式,即将其中的 1 个数据分配到第一个集合中,将剩余的 $s-1$ 数据分配到第二个集合中,此时的平均修复代价为:

$$P_2=\frac{(s-1)(s-2)+1\times 0}{s}$$

并且有 $P_2-P_1=\frac{2(1-s)}{s}<0$.

依此类推,当 s 个数据可被分配的集合数为 $t(t>2)$ 时,则将 s 个数据分配到这 t 个集合中的分配方式一共有 t^s 种,其中考虑修复代价最高的一种方式,即在前 $t-1$ 个集合中的每个集合中只分配一个数据,并将剩余的数据都分配到第 t 个集合中,此时的平均修复代价为:

$$\begin{aligned}P_t&=\frac{(t-1)\times 1\times 0+1\times[s-(t-1)][s-(t-1)-1]}{s}\\&=\frac{s^2-2st+t^2+s-t}{s}\\&=\frac{(s-t)^2+(s-t)}{s}\end{aligned}$$

由此可得出,当数据量 s 不变时,随着已分配数据的集合数 t 的增加,数据的平均修复代价逐渐减小,证明完毕.

定理 2　s 个数据被分配到所有 t 个集合中时,分配越均衡,其平均修复代价越小.

证明　假设 s 个数据不能被平均分配到 t 个集合中时,有 $s/t=p$,$s\%t=q$,即每个集合中至少有 p 个数据. 当余下的 q 个数据全部被分配到一个集合中时,平均修复代价为:

$$\begin{aligned}P_t^1&=\frac{(t-1)p(p-1)+1\times(p+q)(p+q-1)}{s}\\&=\frac{tp^2-tp+2pq+q^2-q}{s}\end{aligned}$$

当余下的 q 个数据被均分到 q 个集合中时,平均修复代价为:

$$\begin{aligned}P_t^2&=\frac{q(p+1)p+(t-q)p(p-1)}{s}\\&=\frac{2pq+tp^2-tp}{s}\end{aligned}$$

并有 $P_t^2-P_t^1=\frac{q(1-q)}{s}$,由于 $q\geqslant 1$,故 $\frac{q(1-q)}{s}\leqslant 0$,即当数据分配越均衡时,平均修复代价越小,证明完毕.

根据定理 1 和定理 2 的结论,本文提出了 BAPA 设计,有 $k+1\leqslant i\leqslant k+r$,$2\leqslant j\leqslant m$,其中 k 为系统节点个数,r 为校验节点个数,m 为实例的翻倍数,则 BAPA 设计中提出了以下规则.

规则 1　对于实例 j,只能将实例 $j-1$ 中的数据作为 Piggybacks 捎带到实例 j 中.

规则 2　实例 j 的节点 i 上捎带的实例 $j-1$ 的子实例中的数据不能与节点 i 位于同一个节点上.

规则 3　实例 j 的节点 i 上捎带的实例 $j-1$ 的多个子实例中的数据不能位于同一节点上.

基于 BAPA 设计,本文得出了能同时减少系统节点

和校验节点修复带宽的 Piggybacking 设计 BARSR-I 和 BARSR-II.

2.2 校验节点的修复

Piggybacking 设计 BARSR-I 将 RSR-I 中的实例数翻倍后为 $m(m\geqslant 2)$,每个实例中含有 2 个子实例,所有子实例依次统一编号后,其编码过程为:将子实例$\{i,i+1\}$,$i\in\{1,3,\cdots,2m-3\}$中的所有校验节点上的数据根据 BAPA 设计尽量均衡的捎带到子实例 $i+2$ 的前 $r-1$ 个校验节点上.

当发生校验节点失效时,失效节点上的数据可分为三类进行修复,修复过程如下:

Step1 失效校验节点中既不作为 Piggybacks 也没有捎带 Piggybacks 的数据,需要通过下载该子实例和其前一个子实例中全部系统节点上的数据来计算恢复;

Step2 失效校验节点中被捎带有 Piggybacks 的数据,除需要下载该子实例中全部系统节点上的数据外,还需要下载捎带到该数据中的各项 Piggybacks 来计算恢复;

Step3 失效校验节点中作为 Piggybacks 的数据,除需下载含有该 Piggybacks 的数据外,还需要下载与该 Piggybacks 数据一同捎带的其他 Piggybacks 数据来计算恢复,修复完成.

为使校验节点的修复更具一般性,假设 $2r$ 不能被 $r-1$整除,定义三个变量为:

$$t_l=\lfloor\frac{2r}{r-1}\rfloor,\ t_h=\lceil\frac{2r}{r-1}\rceil,\ t=2r-(r-1)t_l$$

由修复过程可得,将修复过程推广到 m 取任意其他值时,原始数据总量为 $2mk$,则 BAPA 设计下 BARSR-I 中校验节点的平均修复带宽率如式(1)所示.

$$\gamma_{\mathrm{BARSR-I}}^{par}=\frac{mrk+[tt_h^2+tt_h+(3r-tt_h-t-1)t_l](m-1)}{2mkr} \tag{1}$$

通过分析 BARSR-I 可以发现,只有当校验节点数 $r\geqslant 4$时,BARSR-I 才能将前面的子实例中的 $2r$ 个数据全部捎带到后面的子实例中的前 $r-1$ 个校验节点上. 当 $r=2$ 时,其校验节点的平均修复带宽率为:$\frac{3mk+m+k-1}{4mk}$. 当 $r=3$ 时,其校验节点的平均修复带宽率为:$\frac{3mk+(m-1)(2k+8)+4}{6mk}$.

Piggybacking 设计 BARSR-II 将 RSR-II 中的实例数翻倍后为 $m(m\geqslant 2)$,每个实例中含有 $2r-3(r\geqslant 3)$个子实例,所有子实例依次统一编号后,其编码过程为:对 $i\in\{1,2r-2,\cdots,(2r-3)m-4r+7\}$,将子实例$\{i,i+1,\cdots,i+2r-4\}$中的校验节点上的数据根据 BAPA 设计尽量多的尽量均衡的捎带到子实例$\{i+2r-3,i+2r-2,\cdots,i+3r-5\}$的全部校验节点以及子实例 $i+3r-4$ 的第一个校验节点上.

为简化描述,本文仅对 $r=3$ 时的 BARSR-II 进行讨论,对于 r 取其他值的情况可以类推. 当发生校验节点失效时,失效节点上的数据同样分为三类进行修复,修复过程除第一步中需要下载其前 $2r-4$ 个子实例中全部系统节点上的数据来计算恢复不同外,其余修复步骤与 BARSR-I 相同.

从 BARSR-II 中可以看出,前 $2r-3$ 个子实例中的第一个校验节点上总会存在 $r-2$ 个数据因不能满足捎带设计而不会被捎带到其他子实例中,即只有 $2r^2-4r+2$ 个数据能被捎带到 r^2-2r+1 个校验节点上,每一个校验节点上捎带的 Piggybacks 数据量为 2. 由修复算法可得,将修复过程推广到 m 取任意其他值时,原始数据总量为$(2r-3)mk$,则 BAPA 设计下 BARSR-II 中校验节点的平均修复带宽率如式(2)所示.

$$\gamma_{\mathrm{BARSR-II}}^{par}=\frac{[(2r-3)rk+(6r^2-12r+6)+(r-2)k](m-1)}{(2r-3)mkr} \tag{2}$$

2.3 编码和修复复杂度

考虑到元素在底层有限域 $GF(2^v)$ 上进行基本算数,为简化表述,令一次加法运算的复杂度为 $O(v)$,一次乘法运算的复杂度为 $O(v^2)$[3].

就编码复杂度而言,在 BARSR-I 中,其首先要分别在 $2m$ 个子实例中校验数据. 然后,根据 RSR-I 的编码构造对系统节点进行捎带. 最后,根据 BARSR-I 中的捎带规则对校验节点进行捎带. 综上,BARSR-I 的编码复杂度为:

$$C_E^I=O((2mrk+mr-m)v^2+(2mrk-mr-m+mtt_h+mt_l-tt_h-t_l)v)$$

同理可得 BARSR-II 的编码复杂度为:

$$C_E^{II}=O((r^3+(2m-3)r^2+3r(1-m)+1)kv^2+((2mr^2-3r)(k-1)+(3r^2-9r+6)+(m-1)(2r^2-4r+2))v)$$

就修复复杂度而言,在 BARSR-I 中,当发生校验节点失效时,根据其修复过程,失效数据分为三类,分别根据其修复规则进行解码修复. 综上,BARSR-I 的修复复杂度为:

$$C_R^I=O(2mkv^2+(2m(k-1)+(m-1)t_l+4t_h)v)$$

同理可得 BARSR-II 的修复复杂度为:

$$C_R^{II}=O((2r-3)mkv^2+((2r-3)m(k-1)+4(m-1)(r-1))v)$$

3 对比分析

在近年的研究成果中,同样以系统型 MDS 码作为基础码的 Piggybacking 设计的还有文献[4]中的 CSGG、

文献[5]中的 REPB 以及文献[6]中的 OOP. 表 1 中给出了本文提出的 BARSR-I 和 BARSR-II 与其他 Piggybacking 设计的对比.

表 1　Piggybacking 设计对比表

Piggybacking 设计	子实例数	系统节点的平均修复带宽率	校验节点的平均修复带宽率
RSR-I	$2(r\geqslant 2)$	$\geqslant\frac{r+1}{2r}$	1
RSR-II	$2r-3(r\geqslant 3)$	$\geqslant\frac{r-1}{2r-3}$	1
ORSR-I	$2m$	$\geqslant\frac{r+1}{2r}$	$>\frac{1}{r}+\frac{(r-1)(k+r-1)}{2kr}$
ORSR-II	$(2r-3)m$	$\geqslant\frac{r-1}{2r-3}$	$>\frac{1}{r}+\frac{(r-1)(k+r-1)}{2kr}$
CSGG	r	$\approx\frac{\sqrt{2r-1}}{r}$	1
REPB	$s+p$	$\geqslant\frac{2}{\sqrt{r}+1}$	1
OOP	$\sqrt{r-1}+r-1$	$\geqslant\frac{2\sqrt{r-1}+1}{2\sqrt{r-1}+r}$	$\geqslant\frac{\sqrt{r-1}+1}{r}+\frac{(r-1)^2-\sqrt{(r-1)^3}}{kr}$
BARSR-I	$2m$	$\geqslant\frac{r+1}{2r}$	$\geqslant\frac{1}{2}+\frac{3r-1}{2k(r-1)}$
BARSR-II	$(2r-3)m$	$\geqslant\frac{r-1}{2r-3}$	$\geqslant\frac{1}{2}+\frac{6r^2-12r+6+kr-2k}{2kr(2r-3)}$

从表 1 中可以得出,BARSR-I 和 ORSR-I 的子实例数相同,是 RSR-I 中子实例数的任意倍数;BARSR-II 和 ORSR-II 的子实例数相同,是 RSR-II 中子实例数的任意倍数;在 $r\geqslant 4$ 时,CSGG 的子实例数多于 RSR-I,却少于 RSR-II;而 REPB 中的子实例数由 s 和 p 两个部分组成,并且未给出其确切的取值;对于 OOP,其子实例数总是多于 RSR-I,少于 RSR-II.

在以上 Piggybacking 设计中,都对基础码的系统节点的平均修复带宽率进行了不同程度的降低. 但是,对于校验节点而言,RSR-I、RSR-II、CSGG 和 REPB 并未进行考虑,其校验节点的平均修复带宽率都为 1.

在文献[2]中,ORSR-I 和 ORSR-II 下校验节点的平均修复带宽率如式(3)和(4)所示,并指出,随着实例翻倍数 m 的不断增大,ORSR-I 和 ORSR-II 中校验节点的平均修复带宽率会越来越小. 因此,可给出 ORSR-I 和 ORSR-II 中校验节点的平均修复带宽率的最小值如式(5)和(6)所示.

$$\gamma_{\mathrm{ORSR-I}}^{par}=\frac{2mk+(r-1)[(m+1)k+(m-1)(r-1)]}{2mkr} \tag{3}$$

$$\gamma_{\mathrm{ORSR-II}}^{par}=\frac{(2r-3)mk+(r-1)(m+1)(r-2)k}{(2r-3)mkr}+\frac{(r-1)(m-1)(r-2)(r-1)}{(2r-3)mkr}+\frac{(r-1)\left(\left\lceil\frac{m}{2}\right\rceil k+\left\lfloor\frac{m}{2}\right\rfloor(r-1)\right)}{(2r-3)mkr} \tag{4}$$

$$\min(\gamma_{\mathrm{ORSR-I}}^{par})=\lim_{m\to+\infty}\gamma_{\mathrm{ORSR-I}}^{par}=\frac{1}{r}+\frac{(r-1)(k+r-1)}{2kr} \tag{5}$$

$$\min(\gamma_{\mathrm{ORSR-II}}^{par})=\lim_{m\to+\infty}\gamma_{\mathrm{ORSR-II}}^{par}=\frac{1}{r}+\frac{(r-1)(k+r-1)}{2kr} \tag{6}$$

从式(5)和(6)中可以看出,当实例翻倍数趋于无穷时,ORSR-I 和 ORSR-II 中校验节点的平均修复带宽率具有相同的最小值. 因此,在后续表达中,有

$$\min(\gamma_{\mathrm{ORSR}*}^{par})\triangleq\min(\gamma_{\mathrm{ORSR-I}}^{par})=\min(\gamma_{\mathrm{ORSR-II}}^{par})$$

文献[6]中,OOP 中校验节点的平均修复带宽率的最小值如式(7)所示.

$$\min(\gamma_{\mathrm{OOP}}^{par})=\frac{\sqrt{r-1}+1}{r}+\frac{(r-1)^2-\sqrt{(r-1)^3}}{kr} \tag{7}$$

本文中,随着 m 的不断增大,BARSR-I 和 BARSR-II 中校验节点的平均修复带宽率随着 m 的不断增大而逐渐升高. 因此,当 $m=2$ 时,代入式(1)和(2)可得 BARSR-I 和 BARSR-II 的校验节点可以取得平均修复带宽率的最小值如式(8)和(9)所示.

$$\min(\gamma_{\mathrm{BARSR-I}}^{par})=\frac{1}{2}+\frac{3r-1}{2k(r-1)} \tag{8}$$

$$\min(\gamma_{\mathrm{BARSR-II}}^{par})=\frac{1}{2}+\frac{6r^2-12r+6+kr-2k}{2kr(2r-3)} \tag{9}$$

下面将对 ORSR $*$、OOP、BARSR-I 以及 BARSR-II 的校验节点的平均修复带宽率的最小值进行理论对比分析. 图 1 中给出了当 $m=2, r=4$,系统节点数取值范围为 $k\in\{10,15,\cdots,50\}$ 时,$\min(\gamma_{\mathrm{ORSR}*}^{par})$、$\min(\gamma_{\mathrm{OOP}}^{par})$、$\min(\gamma_{\mathrm{BARSR-I}}^{par})$以及 $\min(\gamma_{\mathrm{BARSR-II}}^{par})$在不同系统节点取值下的对比图. 从图 1 中可以得出,随着系统节点数 k 的取值不断增大,$\min(\gamma_{\mathrm{ORSR}*}^{par})$、$\min(\gamma_{\mathrm{OOP}}^{par})$、$\min(\gamma_{\mathrm{BARSR-I}}^{par})$以及 $\min(\gamma_{\mathrm{BARSR-II}}^{par})$的值都在逐渐减小;无论 k 取何值,$\min(\gamma_{\mathrm{BARSR-I}}^{par})$和 $\min(\gamma_{\mathrm{BARSR-II}}^{par})$的值都小于 $\min(\gamma_{\mathrm{ORSR}*}^{par})$和 $\min(\gamma_{\mathrm{OOP}}^{par})$,并且 $\min(\gamma_{\mathrm{BARSR-I}}^{par})$最低.

图 2 中给出了当 $m=2, k=50$,校验节点数取值范围为 $r\in\{4,5,\cdots,12\}$ 时,$\min(\gamma_{\mathrm{ORSR}*}^{par})$、$\min(\gamma_{\mathrm{OOP}}^{par})$、$\min(\gamma_{\mathrm{BARSR-I}}^{par})$以及 $\min(\gamma_{\mathrm{BARSR-II}}^{par})$在不同校验节点取值下

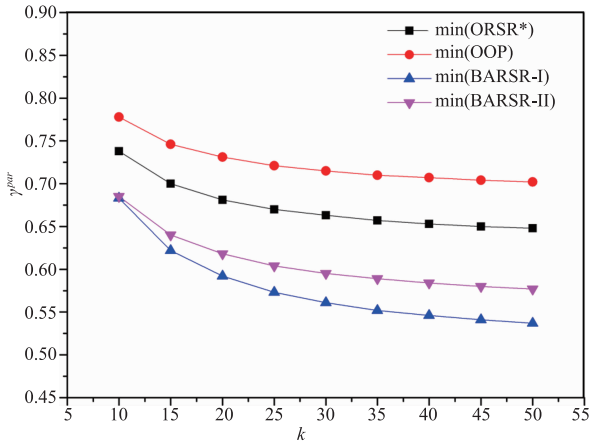


图1 不同k取值下对比图

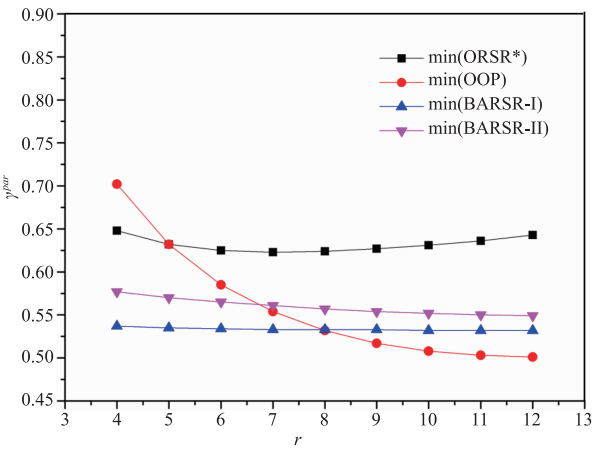


图2 不同r取值下对比图

的对比图. 从图 2 中可以得出,随着校验节点数 r 的取值不断增大,$\min(\gamma^{par}_{\mathrm{ORSR}*})$的值降低后又升高,而$\min(\gamma^{par}_{\mathrm{OOP}})$、$\min(\gamma^{par}_{\mathrm{BARSR-I}})$以及 $\min(\gamma^{par}_{\mathrm{BARSR-II}})$的值都在逐渐减小;无论 r 取何值,$\min(\gamma^{par}_{\mathrm{BARSR-I}})$和 $\min(\gamma^{par}_{\mathrm{BARSR-II}})$的值都小于 $\min(\gamma^{par}_{\mathrm{ORSR}*})$,并且 $\min(\gamma^{par}_{\mathrm{BARSR-I}})$最低;在 $r<7$ 时,$\min(\gamma^{par}_{\mathrm{BARSR-II}})$的值小于 $\min(\gamma^{par}_{\mathrm{OOP}})$;在 $r<8$ 时,$\min(\gamma^{par}_{\mathrm{BARSR-I}})$的值小于 $\min(\gamma^{par}_{\mathrm{OOP}})$.

4 总结

针对减少校验节点的修复带宽问题,本文提出了一种均衡分配的 Piggybacks 捎带设计 BAPA,得出了能同时减少系统节点和校验节点修复带宽的 Piggybacking 设计 BARSR-I 和 BARSR-II,通过与现有的其他 Piggybacking 设计进行对比分析证明,BAPA 设计下的 BARSR-I 和 BARSR-II 能有效的减少校验节点的修复带宽.

参考文献

[1] Rashmi K V, Shah N B, Ramchandran K. A piggybacking design framework for read-and download-efficient distributed storage codes[A]. IEEE International Symposium on Information Theory[C]. Istanbul, Turkey: IEEE, 2013. 331 – 335.

[2] Rashmi K V, Shah N B, Ramchandran K. A piggybacking design framework for read-and download-efficient distributed storage codes[J]. IEEE Transactions on Information Theory, 2017, 63(9): 5802 – 5820.

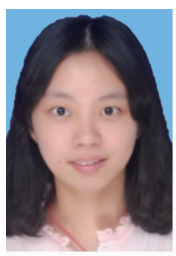
[3] Kumar S, Amat A G I, Andriyanova I, et al. A family of erasure correcting codes with low repair bandwidth and low repair complexity[A]. IEEE Global Communications Conference[C]. San Diego, USA: IEEE, 2015. 1 – 6.

[4] Shangguan C, Ge G. A new piggybacking design for systematic MDS storage codes[J]. Designs Codes & Cryptography, 2019, 87(12): 2753 – 2770.

[5] Yuan S, Huang Q, Wang Z. Generalized piggybacking codes for distributed storage systems[A]. Global Communications Conference[C]. Washington, USA, 2016. 1 – 6.

[6] Li G Y, et al. An efficient one-to-one piggybacking design for distributed storage systems[J]. IEEE Transactions on Communications, 2019, 67(12): 8193 – 8205.

作者简介



周 悦 女,1993 年出生于四川眉山. 现为四川师范大学计算机科学学院硕士研究生,研究方向为分布式存储与纠删码.
E-mail: yzhou916@foxmail.com

李贵洋(通信作者) 男,1975 年出生四川宜宾,现为四川师范大学计算机科学学院副教授,研究生导师,研究方向为分布式存储与纠删码、机器学习.
E-mail: gyli@sicnu.edu.cn