DOI: 10.3969/j.issn.1009-6868.2015.05.008
网络出版地址:http://www.cnki.net/kcms/detail/34.1228.TN.20150918.0918.002.html

大数据环境下的可靠存储技术思考

Reliable Storage Technologies for Big Data

中图分类号:TN929.1 文献标志码:A 文章编号:1009-6868 (2015) 05-0027-005

摘要: 针对分布式容错技术的研究,提出了两点关键要求:降低冗余开销、提高节点修复效率。分析目前主流的容错策略:复制、纠删码、再生码、基于局部可修复码,并认为这些容错策略存在不同程度的缺陷,因此设计出容错能力、计算效率及存储利用率更高的容错策略,仍是未来很长一段时间内值得深入研究的问题。

关键词: 大数据;可靠性;分布式存储;容错技术

Abstract: Two key requirements of fault tolerance technology are proposed in this paper: minimal storage overhead and maximum node recovery performance. Four main strategies for fault tolerance are analyzed: replication, erasure codes, regenerating codes and locally repairable codes. It is considered that these fault tolerance strategies have different defects. Designing a fault tolerance strategy with higher fault tolerance, better computational efficiency and memory utilization will still be a problem needs to be solved in the future.

Key words: big data; reliability; distributed storage; fault tolerance technology

李挥/LI Hui
张宇蒙/ZHANG Yumeng
陈俊/CHEN Jun
(北京大学深圳研究生院,广东 深圳 518055)
(Shenzhen Graduated School, Peking University, Shenzhen 518055, China)

- 纠删码的低效修复已经成为限制其广泛应用的瓶颈所在
- 再生码和局部可修复码通过适量增加存储开销,有效降低了纠删码的修复带宽
- 再生码编码策略的未来研究方向,应结合安全问题、网络拓扑和磁盘I/O复杂度进行设计

随着经济全球化的发展和科技改革的推进,网络覆盖面积不断加大,信息交互随之增强,全球数据正在以爆炸式的速度增长。国际数据公司(IDC)报告指出,从2010—2020年全球数据量将有50倍的增长,预测达到40 ZB数量级[1]。同时海量数据对存储系统提出了巨大的挑战,根据统计,数据存储的需求每年的增速在50%~62%之间。大规模分布式存储系统以其海量存储能力、高吞吐量、高可用性和低成本的突出优势成为存储海量数据的有效系统并被广泛使用。当前最主流的分布式系统是开源的Hadoop分布式文件系统(HDFS)[2],作为GFS[3]的一个开源实现,它被应用于众多大型企业,如Yahoo、Amazon、Facebook、eBay等。

收稿日期:2015-08-16
网络出版时间:2015-09-20

随着分布式存储系统的规模越来越大,为节省成本,存储节点大多采用廉价、可靠性差的设备,这直接导致节点故障越来越频繁。图1给出了Facebook部署的Hadoop集群的日节点失效数。集群共3 000个节点,涉及45 PB数据,平均每天有22个节点失效,最高的日节点失效超过100个[4]。如何有效保障数据可靠性成为了当前分布式存储系统首要关注的问题。

为了提供可靠的存储服务,分布式存储系统通过引入冗余信息来提高系统的容错能力。这种冗余存储的方式能够使系统容忍一定数量的节点故障[5-6],同时系统还需要一个良好的节点修复机制,在发生故障时能快速有效地修复失效数据,维持系统

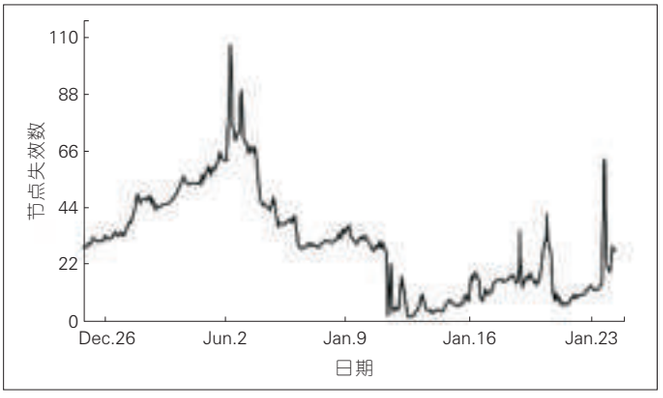


图1 Facebook日节点失效数

冗余度。

1 基于复制的容错技术

复制策略是引入冗余最简单的方法，其基本思想是为系统中的每一个数据对象都建立若干个相同的副本，并把这些副本分散存储在不同的节点上，当遇到某个数据损坏或失效而无法正常使用时，可通过访问最近的存储节点来获取与原件完全一致的数据备份，这样只要数据对象还有一个存活副本，分布式存储系统就可以一直正常运行。修复过程也十分简单高效，只要向所有存储副本的节点中最近的节点发出请求、下载并重新存储，即可恢复系统冗余度。复制策略存储方式简单，易于实现，故障修复容易，并且便于扩展。此外，存储的多个副本也可以均摊读文件时的负载，如通过为热点文件配置更高的副本数来支持高效的并发读操作。

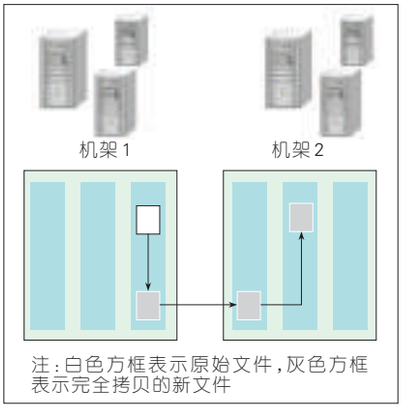
但是在节点数量庞大，存储结构复杂的大规模分布式系统中，要实现快速高效的容错技术，必须解决3个问题：副本数量的设置、副本的放置方式和副本的修复策略。

1.1 副本数量设置

设置副本数量一般有两种方式：一是静态设置，主流的分布式文件系统如HDFS[2]和GFS[3]都是采用3副本固定机制，这种方法操作简单，但灵活性差；二是动态设置副本数量，亚马逊分布式存储系统S3提供用户可以自行设定副本数的功能。另外，文献[7]提出一种动态的容错机制，系统根据数据的访问频率、出错概率、网络状况以及存储时间等动态因素决定副本数，同时动态地删除或添加副本，这种动态机制能大大增加存储空间的利用率、提高数据的获取性能，但动态决策方式会加大系统的处理开销。

1.2 副本放置策略

副本的放置策略不但影响分布式存储系统的容错性能，还关系到副本的存储效率和访问效率。HDFS采用的3副本放置策略，如图2所示[2]。3副本放置策略为：本地放一份，同机架内其他任一节点放一份，不同机架的任一节点放一份。同机架内存放两个副本，可减少机架间的数据传输，方便本地节点对于数据需求时的读取。若本地数据损坏，节点可以从同一机架内的相邻节点获取数据，读取速率快。而数据块存放在两个不同的机架中能避免机架故障导致的数据不可用。同时，为了降低整体的带宽消耗和读取延时，HDFS会尽量让读取程序读取离它最近的副本。如果在读取程序的同一个机架上有一个副本，那么就读取该副本。如果一个HDFS集群跨越多个数据中心，那么客户端也将首先读本地数据中心的副本。



注：白色方框表示原始文件，灰色方框表示完全拷贝的新文件

▲图2 HDFS的3副本放置策略

1.3 副本修复策略

容错技术的修复过程事实上就是恢复系统的冗余度，保证其在一定的可接受范围内。实际的存储系统采用的修复策略有两种：一种是“主动”修复策略[8]，一旦检测到一个副本失效立刻创建一个新副本；另一种是基于阈值的“惰性”修复策略，这种策略只有当备份数量小于某个阈值才进行修复，如Total Recall[9]。根据资源的访问频率，可以分为热门资源和冷门资源，热门资源一般采用主动修复，而访问量小的冷门资源则可以采用惰性修复策略，减少修复临时失效等不必要的开销。

2 基于纠删码的容错技术

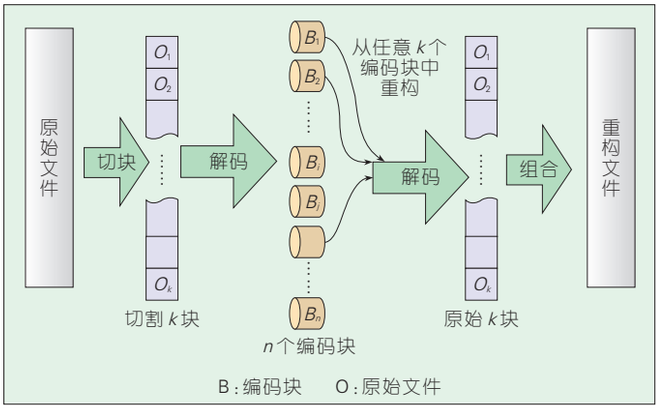
纠删码起源于通信传输领域，由于其数学特性，被逐渐应用于大规模存储系统中，特别是分布式存储环境，实现数据的冗余保护。相较于复制策略，纠删码技术在相同可靠性条件下可以最小化冗余存储，学术界和工业界已将纠删码广泛应用于分布式文件系统。例如卡耐基梅隆大学研究的DiskReduce[10]、Facebook的HDFS-RAID[11]、谷歌的Colossus[12]、微软的Azure[13]存储系统均采用了纠删码并实现了更经济的可靠性。

2.1 纠删码基本原理

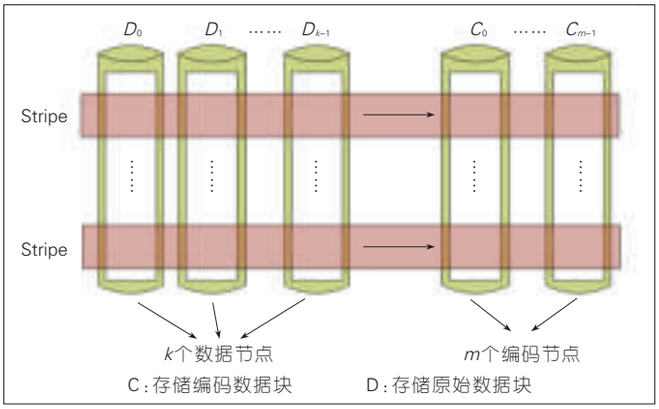
纠删码的基本原理如图3所示，存储原始文件O，首先将其切分成k个数据块，记为$O_1, O_2, \cdots, O_k$，然后编码生成n个编码块，记为$B_1, B_2, \cdots, B_n$，$n>k$，最后将这n个编码块按照一定的放置规则分别存储在不同的节点上。编码过程中生成了冗余数据，当系统中有存储节点失效时，只要留下足够的编码块就可以利用这些剩余的编码块恢复出丢失的数据，维持系统的冗余度。若n个编码块中任意k个块即可重构原始文件O，则这种纠删码满足最大距离可分特性（MDS）[14]，在可靠性和冗余的权衡上达到最优，最常用的编码方法是RS码[15]。

2.2 基于纠删码的分布式存储模型

在分布式存储系统中，数据分布在多个相互关联的存储节点上，通常情况下，映射生成的编码块需要存储在不同的节点上。图4给出了一种基于纠删码的分布式存储模型[16]，假设系统中含有n个存储节点，其中k个是数据节点，m个是编码节点，即满足$n = k + m$。k个数据节点存储原始数据块，标记为$D_0, D_1, \cdots, D_{k-1}$；$m$个



◀图 3
纠删码原理示意



◀图 4
基于纠删码的分布式存储模型

编码节点存储编码数据块，标记为 $C_0, C_1, \cdots, C_{m-1}$。纠删码算法需要将原始文件切割成 k 等份后依次存储在 k 个数据节点中，并将编码生成的 m 份放入 m 个编码节点。当存储大文件时，需要对原始文件进行二次切割，即每次从文件中读取指定大小的数据量进行编码，我们将一次编码过程中涉及的原始数据和编码数据称为一个 stripe[16]。一个 stripe 独立地构成一个编码的信息集合，不同 stripe 之间相互无关。但是，逻辑上的 stripe 与实际物理节点的对应关系并不是恒定不变的，可以通过 stripe 的轮转实现数据存储负载均衡。

与复制策略相比，纠删码策略可以有效地降低维持可靠性所需的存储开销，提供令人满意的存储效率[5]。

2.3 纠删码技术的缺陷

然而，基于纠删码的容错技术未能在实际的大规模分布式存储系统中真正应用，除了其结构较复制策略复杂外，纠删码本身在数据恢复时存在致命的缺陷。在基于纠删码的分布式存储系统中，当一个节点失效时，为维持系统冗余度，新节点需要首先从 k 个节点中下载全部数据恢复出原始文件，再重新编码生成失效的数据，这个过程中传输的数据量是失效数据的 k 倍。当节点在网络中分布较分散时，节点的修复需要消耗大量的网络带宽。这一缺陷在普通分布式系统中已有制约，在大数据环境下，数据量和存储节点在成倍甚至几何级增长时更为明显。同时，需要的下载量太大势必会导致节点修复过程变慢，对于不断发生故障的分布式存储系统来说，节点的修复速率直接影响到系统可靠性。如果修复速率过慢，甚至赶不上节点发生故障的速度，那么系统将无法维持其可靠性。据 Facebook 在 HotStorage '13 上发布的论文指出，纠删码的低效修复已经成为限制其广泛应用的瓶颈所在[4]。

针对纠删码的修复问题，Rodrigues 提出了一种混合策略[5]：采用纠删码的同时维护一个副本，从而有效减少修复带宽。然而，这种混合策略节省带宽有限，但存储开销大，同时使得系统设计复杂化。Dimakis 创造性地将网络编码应用于分布式存储，提出再生码的概念[17]，显著降低了修复带宽。

3 基于再生码的容错技术

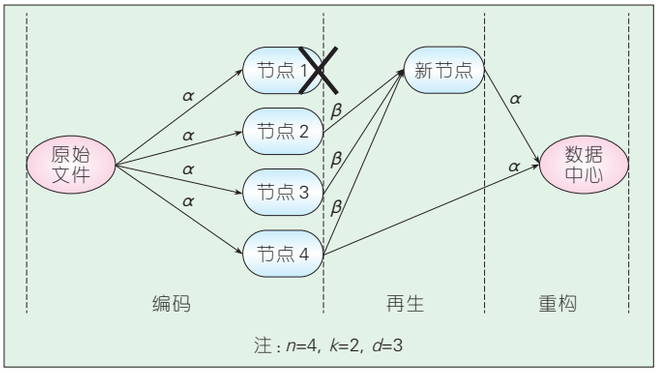
3.1 再生码的基本原理

再生码的描述如下：将原始文件编码后存储到 n 个节点中，每个节点存储大小为 α。当一个节点失效时，新节点连接剩余 $n-1$ 个节点中的 d 个节点 ($k \leqslant d \leqslant n-1$)，从每个节点下载大小为 β ($\beta \leqslant \alpha$) 的数据进行修复，即修复带宽为 $\gamma=d\times\beta$。再生码的参数集可表示为 $\{n, k, d, \alpha, \beta, B\}$，其平均修复带宽 γ 小于文件大小 B。再生码的编码、再生及重构过程如图 5 所示。

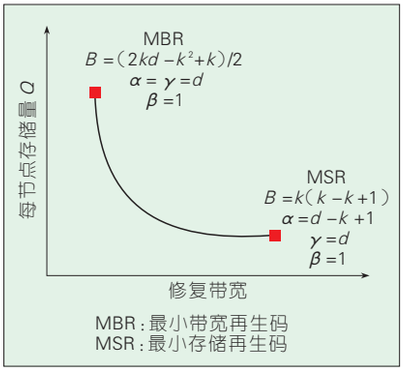
随着每个节点的存储量的提高，节点修复时需要下载的数据量将降低，通过在信息流图上求最小割界的方法，给出了节点修复带宽消耗的下界曲线，而再生码正是在存储开销 α 和修复带宽 γ 的最优曲线上。如图 6 所示，最优曲线上存在两个极值点，分别代表最优存储效应和最小修复带宽效应，达到这两个极值点的编码称为最小存储再生码（MSR）和最小带宽再生码（MBR），已有一些明确的编码实现[18]。理论上，当 $d=n-1$ 时，再生码的修复带宽达到最小值。

3.2 再生码技术的瓶颈及前景

虽然理论上再生码可以达到最优的存储开销和修复带宽，但由于它依赖于复杂的参数和晦涩难懂的数学理论，其实现方式非常复杂。现有的再生码大多在有限域 $GF(2^w)$ 上进行域元素的多项式运算[18]。计算机处理中，加法较为简单，但乘法和除法



◀图5 再生码编码、再生及重构过程



▲图6 “存储—带宽”的最优曲线

却非常复杂，甚至需要借助离散对数运算和查表才能实现。这使得再生码的编解码计算开销大，无法适应存储系统对计算效率的要求。很多研究都表明，设计一种结构简单、计算复杂度低的策略至关重要。文献[19]中分析了3种再生码：随机线性网络码（RL）[20]、精确线性码（EL）[21]和生成矩阵码（PM）[22]。其中，PM码利用一种紧凑的表示方式和高效的编解码算法大大提高了编解码速率，然而与纠删码相比，PM码仍需要更长的计算时间。

再生码作为对纠删码的改进，具有很好的理论支撑。但目前提出的大多数再生码、编解码复杂度较高且码率较底。如何提出码率较高并且复杂度低的编码策略就很有意义。深圳市融合网络技术实验室在该领域进行了深入研究，并取得了一定的研究成果：1）提出BASIC[23]编码框架，利用一种新颖的卷积形式来表示编码运算过程，可以将有限域运算转化为GF(2)内简单的移位和异或操作；2）提出一种改进的Zig-Zag编码[24]，采用移位和异或的Zig-Zag解码算法，避免解码时所需要的复杂计算，达到了最低的编解码复杂度。这些编码都可以应用在再生码的构造上，以更好地实现码率较高并且复杂度低的编码。

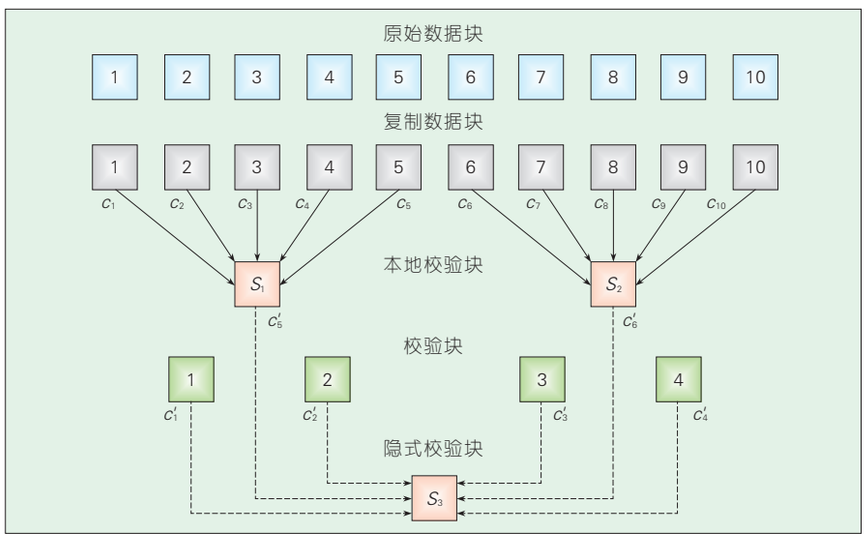
对于再生码编码策略的未来研究方向，应结合安全问题、网络拓扑和磁盘输入输出（I/O）复杂度进行设计，从而使再生码更为实用。

4 基于局部可修复码的容错技术

除再生码外，局部可修复码技术（LRC）[25]可以通过增加本地数据实现修复带宽的降低。文献[25]给出了修复局部性r、编码距离d、每个节点的存储大小α以及存储编码长度n之间的权衡。Facebook在HDFS中实现了LRC技术[4]，微软也在Azure上添加了LRC技术[26]。

文献[4]给出了LRC技术的一种实现：如图7，原始文件被等分成10个数据块，通过RS编码生成4个冗余块，图中显示为4个绿色方框。为降低修复带宽，在RS码基础上进行二次编码产生3个额外的冗余块，标记为S_1，S_2和S_3，图中显示为3个橙色的方框。S_1是由前5个数据块编码产生，S_2是由后5个数据块编码产生，这两个由局部原始数据块编码产生的冗余块称为本地校验块。而S_3则是由4个冗余校验块编码产生，称为隐式校验块。实际存储中，我们将10个原始数据块标记为c_1，c_2，...，c_{10}，将7个冗余块标记为c_1’，c_2’，...，c_7’，存放在7个不同的节点。当1个数据块丢失时，只需要1个额外的冗余块和4个数据块即可修复失效数据，与传统纠删码相比，修复带宽降低了大概一半。

从图7可以看出LRC技术以额外的14%存储开销为代价，降低RS码的修复带宽。但其编码方式仍是RS码，因此编码效率没有提高。另外，LRC编码不满足MDS特性，系统还需要增加额外信息标示二次编码数



▲图7 基于LRC容错技术的实现过程

据。当修复一个节点故障时，LRC具有很好的修复局部性，但修复两个或两个以上的节点故障时就需要连接k个节点，修复带宽与纠删码相同，仍是失效数据的k倍。随着存储系统规模变得越来越大，出现两个或者多个故障的几率也随之增大。

除此之外，针对大数据存储系统中的容错修复问题，我们不断对存储编码的构造方式进行改进[27-28]，以获得更低的冗余开销和更高效的修复性能。

5 结束语

介绍了大数据环境下的可靠存储技术，并针对分布式存储系统，介绍多种容错策略及相关技术。基于复制的容错技术冗余度大，性能提升艰难，很多研究者将目光聚集于基于纠删码的容错技术。而再生码和局部可修复码通过适量增加存储开销，有效降低了纠删码的修复带宽。

这些容错策略在容错能力、计算效率、存储利用率等方面都存在不同程度的缺陷，如何平衡这些影响系统可靠性的因素，设计出容错能力、计算效率及存储利用率更高的容错策略，仍是未来很长一段时间内值得不断深入研究的问题。

致谢：

感谢北京大学先进网络技术实验室博士研究生侯韩旭对于研究的理论支持。

参考文献

[1] 潘凯西. 大数据支持企业转型与创新[R]. 北京: 大数据重构企业智慧发布会, 2014
[2] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, May 3–7, 2010, Lake Tahoe, Nevada, USA, 2010: 1–10
[3] Ghemawat S, Gobioff H, and Leung S T. The Google file system [J]. ACM SIGOPS operating systems review, 2003, 37(5): 29–43
[4] Sathiamoorthy M, Asteris M, Papailiopoulos D, et al. Xoring elephants: Novel erasure codes for big data[J]. VLDB Endowment, 2013, 6(5): 325–336
[5] Weatherspoon H, and Kubiatowicz J D. Erasure coding vs. replication: A quantitative comparison [M]. Peer–to–Peer Systems. Germany: Springer Berlin Heidelberg, 2002: 328–337
[6] Rodrigues R, Liskov B. High availability in DHTs: Erasure coding vs. replication [M]. Peer–to–Peer Systems IV. Germany: Springer Berlin Heidelberg, 2005, pp. 226–239
[7] Wu L, et al. A Dynamic Data Fault–Tolerance Mechanism for Cloud Storage [C]//Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on, Xi'an, China, 2013: 95–99
[8] Sit E, et al. Proactive Replication for Data Durability [C]//5th International workshop on Peer–To–Peer Systems (IPTPS 2006), Santa Barbara, USA, 2006
[9] Bhagwan R, Tati K, Cheng Y, et al. Total Recall: System Support for Automated Availability Management [C]//1st Symposium on Networked Systems Design and Implementation (NSDI 2004), San Francisco, USA, 2004: 25–25
[10] Fan B, Tantisiriroj W, Xiao L, et al. DiskReduce: RAID for data–intensive scalable computing [C]//Proceedings of the 4th Annual Workshop on Petascale Data Storage. ACM, 2009: 6–10
[11] Saving capacity with HDFS RAID [EB/OL]. [2014–06–05]. https://code.facebook.com/posts/536638663113101/saving–capacity–with–hdfs–raid/.
[12] Colossus [EB/OL].//http://www.quora.com/Colossus–Google–GFS2
[13] Calder B, Wang J, Ogus A, et al. Windows Azure Storage: a highly available cloud storage service with strong consistency [C]// Proceedings of the Twenty–Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 2011: 143–157
[14] Blaum M, et al. On lowest density MDS codes [J]. Information Theory, IEEE Transactions on, 1999, 45(1): 46–59
[15] Stephen B W, and Vijay K B. Reed–Solomon codes and their applications [M]. USA: John Wiley & Sons, 1999
[16] Xianghong L, and Jiwu S. Summary of research for erasure code in storage system [J]. Journal of Computer Research and Development, 2012, 49(1): 1–11
[17] Dimakis A G, Godfrey P B, Wu Y, et al. Network coding for distributed storage systems [J]. Information Theory, IEEE Transactions on, 2010, 56(9): 4539–4551
[18] Dimakis A G, Ramchandran K, Wu Y, et al. A survey on network codes for distributed storage [J]. Proceedings of the IEEE, 2011, 99(3): 476–489
[19] Jiekak S, Kermarrec A M, Le Scouarnec N, et al. Regenerating codes: A system perspective [J]. ACM SIGOPS Operating Systems Review, 2013, 47(2): 23–32
[20] Ho T, M é dard M, Koetter R, et al. A random linear network coding approach to multicast [J]. Information Theory, IEEE Transactions on, 2006, 52(10): 4413–4430
[21] Suh C, Ramchandran K. Exact–repair MDS code construction using interference alignment [J]. Information Theory, IEEE Transactions on, 2011, 57(3): 1425–1442. doi: DOI: 10.1109/TIT.2011.2105003
[22] Rashmi K V, Shah N B, Kumar P V. Optimal exact–regenerating codes for distributed storage at the MSR and MBR points via a product–matrix construction [J]. Information Theory, IEEE Transactions on, 2011, 57(8): 5227–5239
[23] Hou H, Shum K W, Chen M, et al. BASIC regenerating code: Binary addition and shift for exact repair[C]//Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE, Istanbul, Turkey, 2013: 1621–1625
[24] Chen J, Li H, Hou H, et al. A new Zigzag MDS code with optimal encoding and efficient decoding[C]//Big Data (Big Data), 2014 IEEE International Conference on. IEEE, Washington DC, USA, 2014: 1–6
[25] Papailiopoulos D S, Dimakis A G. Locally repairable codes [C]//information theory 体系，proceedings (ISIT), 2012 IEEE international symposium on. IEEE, Cambridge, USA, 2012: 2771–2775
[26] Huang C, Simitci H, Xu Y, et al. Erasure Coding in Windows Azure Storage[C]// USENIX Annual Technical Conference, BOSTON, MA, 2012: 15–26
[27] Zhu B, Shum K, Li H. Heterogeneity–Aware Codes with Uncoded Repair for Distributed Storage Systems[J]. Communications Letters, IEEE, 2015,19(6):901–904. doi: 10.1109/LCOMM.2015.2415495
[28] Zhu B, Shum K, Li H, et al. General fractional repetition codes for distributed storage systems [J]. Communications Letters, IEEE, 2015, 18(4): 660–663. doi: 10.1109/LCOMM.2014.030114.132694

作者简介

李挥，北京大学教授、博士生导师，广东省粤教云计划总体组专家，广东省计算机学会大数据专委会副主任委员，广东省云计算行业协会副会长等；研究领域为未来网络大数据、分布式存储系统及网络编码理论；近5年在国内外核心刊物和会议上发表论文50多篇，其中SCI和EI收录45篇。

张宇蒙，北京大学先进网络技术实验室在读硕士研究生；研究领域为分布式存储系统及网络编码理论。

陈俊，北京大学先进网络技术实验室在读硕士研究生；研究领域为分布式存储系统及网络编码理论。