纠删码存储系统修复算法及其发展趋势研究综述

M202173667-余祺

摘　要　伴随着现实世界中数据的爆炸增长，纠删码存储已经成为了成为了副本存储的有效替代方法，因为在保障相同的可靠性情况下，纠删码能大幅减少存储开销。然而，纠删码在修复失效的块时会生成大量网络流量，导致修复时间不乐观。为了解决这一网络 I/O 放大问题，学术界主要从三个方面深入研究：1）构建具有低修复流量的纠删码体系；2）设计有效的修复算法以减少修复时间；3）使用机器学习算法对可能出现的故障进行预测，从而提前重存数据。本文将按论文发表顺序分别讨论以下修复算法：1）局部并行修复技术（Partial-Parallel-Repair，PPR）；2）修复流水线技术（Repair Pipelining，RP）；3）并行流水线树（Parallel Pipeline Tree，PPT）。并会讨论针对于全节点修复的框架 RepairBoost 以及纠删码的发展趋势——大条带，以及如何处理大条带情形下被进一步放大的网络 I/O。

1　引言

伴随着网络世界的复杂化，人们越来越依赖网络，从而导致了大量的数据被生产出来。根据研究，在世界范围内，有约 90%的数据于最近两年产生。这些数据不会马上消失，而是流向各式存储系统中，等待使用者的后续访问。伴随大量数据的产生，相应的故障问题也愈发凸显。使用者一定希望在要使用指定数据时能及时访问它们，因此，数据的可靠性需要得到保障。

比较传统的维护数据可靠性的方案是复制，许多实际的存储系统（例如 Ceph，HDFS，Swift 等）存放原始数据及它们的两个副本，这会大幅增加存储成本。近年来，纠删码（Erasure Code）以其低存储开销、高可靠性的特点被当作是数据复制的替代方案。其中，Reed-Solomon 码是使用最广泛的纠删码，因为它在给定的存储开销下提供了最大的可靠性，并且可以灵活选择确定可实现可靠性的编码参数。

Reed-Solomon 码（以下简称 RS 码）的编码参数一般为（k，m），意思是通过 k 个原始的数据块编码得到 m 个冗余块（编码参数也可以是（n，k），其中 n = k + m），这（k + m）个块的集合视为一个条带（Stripe），当（k + m）个块中的任何 k 个块都足以重建原始数据时，编码完成。例如，在 RS（4，2）码中，将总大小为 256MB 大小的数据分为四个块，每个块大小为 64MB；然后，使用这四个大小为 64MB 的块进行计算，得到两个大小同样为 64MB 的冗余块。同样地，若要在三重复制系统下得到冗余，则这四个 64MB 的块均要被复制一遍。综上所述，RS（4，2）编码系统花费 1.5 倍初始存储空间来保证存储系统的可靠性；而三重复制系统花费 3 倍的初始存储空间才能容忍相同数量的同时发生的故障。

然而，虽然纠删码系统不需要太多的存储开销去维持数据可靠性，但是纠删码系统在面临数据块失效时，需要多个幸存块来恢复丢失块。以 RS（4，2）系统为例，若有一个大小为 64MB 的数据块丢失，则需要 4 个幸存块去恢复丢失块。这意味着，相比于三重复制系统，RS（4，2）纠删码系统将修复时的网络 I/O 放大了 4 倍（块大小相同时）。因此，为了缓解纠删码中的网络 I/O 放大问题，研究者们提出了各种修复算法，用以减少纠删码系统修复过程中的网络传输时间，甚至有的修复算法理论上能将网络传输时间减少到 O（1）（指单块修复时间）。显然，

当纠删码的修复时间在可控范围之内时，纠删码存储系统的可靠性会大大增加。

另一方面，为了进一步减少冗余，进而减少成本，最新的生产系统考虑采用大条带纠删码系统。所谓大条带，指的是对于编码参数（k，m），当 m 保持 3 或 4 而 k 较大（一般指超过 20）时，纠删码系统的条带环境。比如，VAST 系统考虑（n，k）=（154，150）的设置，仅产生 1.027 倍的冗余。大条带所带来的高存储效率对于冷分布式存储系统和热分布式存储系统均有一定的吸引力。大条带能使冷分布式存储系统以极低的存储成本实现长久存储（冷存储系统中的数据需要长期存储但很少访问）；同时，大条带能显著减少热分布式存储系统较为昂贵的硬件成本。

然而，大条带的修复成本问题会被进一步放大。因此，缓解纠删码系统中的修复瓶颈是一个有实际意义的研究课题。而设计有效的修复算法也是主要的研究方向之一。

2 原理和优势

PPR 算法：局部并行修复技术（Partial-Parallel-Repair，以下简称 PPR），是一种将重建操作细分并调度的修复技术，可以在 O（lg（n））时间内完成。PPR 将整个修复操作分为一组局部操作，然后安排这些局部操作在多个服务器上并行执行。PPR 减轻了网络容量和磁盘读取的压力。图 fig1.jpg 可以进一步比较传统修复手段和 PPR 的区别。传统修复手段将所有数据路由到一个节点然后进行修复，示例中是路由到 S7 中。这会导致 S7 处下载链路的拥挤，影响到了修复性能，需要花费 4 个时隙完成一个完整的 RS（3，2）系统修复过程。而 PPR 将修复任务细分，并将细小任务并行执行，共需要 3 个时隙，并且 S7 处的链路没有拥塞。

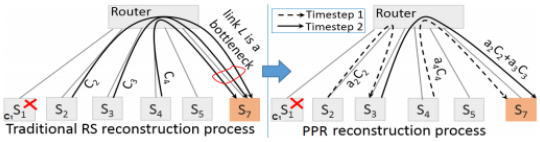


fig 1.jpg

PPR 算法减少了修复时间，但带宽分布仍未完全平衡。因此，在 PPR 之后，有学者提出了修复流水线技术（Repair Pipelining，以下简称 RP）。

RP 算法：具体来说，RP 将 k 个 helper 和 1 个 requester 安排为一个线性路径，即 N1→N2→…→Nk→R。在较高级别上，为了修复丢失的块 B*，N1 将 a1B1 发送给 N2。然后，N2 将 a1B1 与它自己的块 B2 合并，并将 a1B1 + a2B2 发送到 N3。重复该过程，最后 Nk 向 R 发送合并结果 B*。整个修复会导致跨越 k 个不同链路的 k 个传输。因此，没有瓶颈链接。但是，这种简单的方法未充分利用带宽资源，因为每个时隙中只有一个块级传输。整个修复仍然需要 k 个时隙，与常规修复相同。因此，修复流水线将块的修复分解为一组称为 S1，S2，…，Ss 的 s 个小型固定大小单元的修复。它通过线性路径对每个片的修复进行流水线传输，并且链路上每个片级别的传输仅占用 1/s 时隙。fig2.jpg 显示了在 k = 4 和 s = 6 时修复流水线如何工作。经过分析，整个修复时间由 (s-1+k)/s=1+(k-1)/s 时隙组成，因此，当 s 趋于无穷时，修复时间趋近于单个时隙。

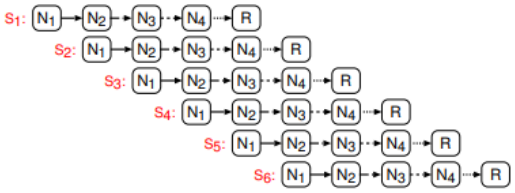


fig 2.jpg

虽然 RP 算法进一步均衡了带宽，减少了修复时间，但是无论是 RP 算法还是 PPR 算法，考虑的都是均匀网络，这两种算法并没有考虑在非均匀网络的情况下如何优化。并行流水线树（Parallel Pipeline Tree，以下简称 PPT）在非均匀网络的环境下对 RP 算法进行了优化，使算法能适应异构网络环境。

PPT 算法：针对于异构网络环境，RP 算法中的修复流水线会被修复链路中的瓶颈带宽所限制，进而影响到整个修复流水线的修复效率。PPT 设计了一种算法，使得修复流水线可以绕开瓶颈链路，从而提高流水线性能。

具体来说，如 fig3.jpg 所示，（a）图表示 RP 中流水线方向，此时流水线的瓶颈链路为 London->Tokyo2，瓶颈带宽为 31Mb/s；修改路由方式，使得 London，Tokyo2 节点均向 California 节点发送数据，瓶颈带宽为 34.5Mb/s，成功绕过（a）中的瓶颈链路。

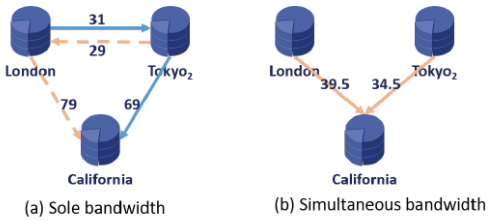


fig 3.jpg

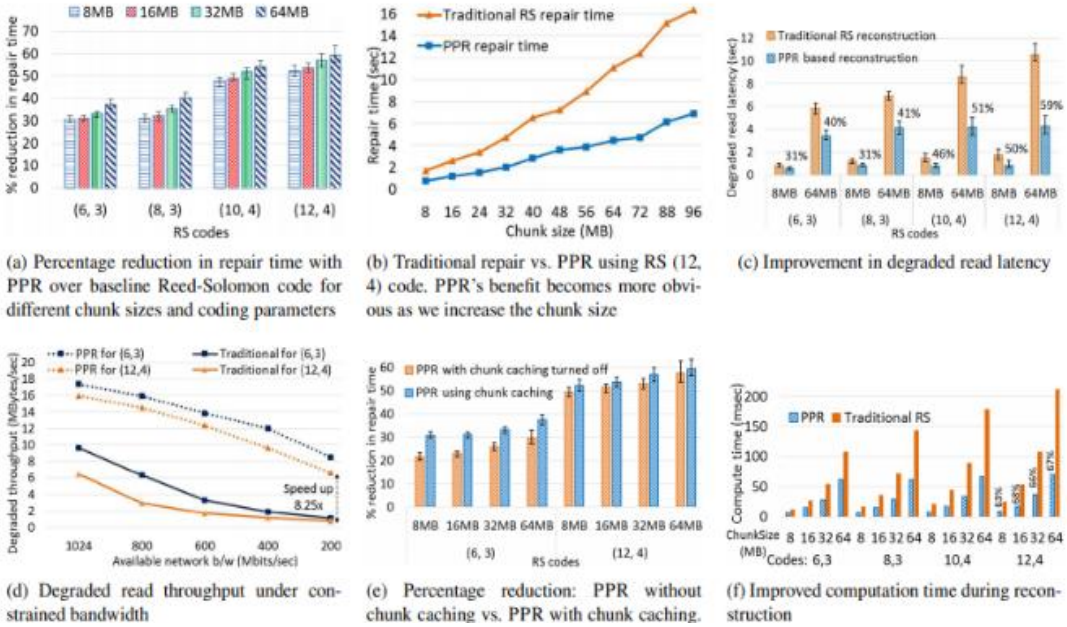
RepairBoost：RepairBoost 是一个修复框架，针对全节点修复。所谓全节点修复，指的是某个节点宕机后，需要将该节点上的数据块全部修复出来。RepairBoost 背后的主要思想是通过修复有向无环图(RDAG)制定单块修复，该图表征网络上的数据路由以及请求修复的块之间的依赖关系。然后将一个 RDAG 分解为多个修复任务，每个修复任务执行数据上传和下载以方便修复。RepairBoost 通过以下两个步骤来平衡全节点修复中的修复流量和带宽利用率：(i)将多个 RDAG 的修复任务仔细分派到相应的节点，以平衡整体上传下载修复流量；(ii)协调修复任务的执行顺序，使可用上传和下载带宽的利用率达到饱和。RepairBoost 将全节点修复的所有修复任务统一组织调度，一定程度上使得各个块之间的修复耦合更紧密，并且通过任务调度减少了修复时间、均衡了节点负载。

3 研究进展

针对纠删码系统中修复 I/O 放大问题，本文调查了第 2 节中的几个算法。这些算法没有修改纠删码结构，没有增加存储开销，也没有减少修复 I/O 流量，而是通过调度、流水线等技术，减少修复时间，缓解修复给纠删码系统网络所带来的压力。

PPR 算法假设的网络环境是具有类似于胖树的拓扑结构，每个级别获得近似完整的二等分带宽，数据中心中任何两台服务器之间的网络容量相似。PPR 通过增加修复操作的并行度，减少网络拥塞。且由于大多数实际使用的纠删码是线性相关的，基于 PPR 的修复可以很容易地应用在这些纠删码系统上面。经过计算，PPR 理论上将网络传输时间减少了$\frac{k}{\lceil(\log_2(k+1))\rceil}$，其中 k 为编码参数。

图 fig4.jpg（a）说明了针对不同编码参数的 PPR 与传统修复技术相比，修复时间减少



(a) Percentage reduction in repair time with PPR over baseline Reed-Solomon code for different chunk sizes and coding parameters

(b) Traditional repair vs. PPR using RS (12, 4) code. PPR's benefit becomes more obvious as we increase the chunk size

(c) Improvement in degraded read latency

(d) Degraded read throughput under constrained bandwidth

(e) Percentage reduction: PPR without chunk caching vs. PPR with chunk caching.

(f) Improved computation time during reconstruction

fig 4.jpg

的百分比。可以看到，随着 k 的增加或单块大小的增加，修复性能增加的更多。图（b）进一步说明了这一点。fig4.jpg（c）说明了 PPR 在降低降级读延迟方面的表现，可以看到相比于传统修复降级读延迟大幅下降。PPR 不仅减少了修复时间，而且减少了参与修复节点的最大传输数据量。从 fig4.jpg(d)中可以看出，当可用带宽下降时，应用 PPR 系统的吞吐量远大于应用传统修复方案系统的吞吐量。此外，PPR 的缓存机制减少了磁盘 I/O 时间，fig4.jpg（e）指出块缓存对于较低的 k 作用更大。最后，PPR 所引入的分布计算原数据块方案几乎带来太多额外的开销，从 fig4.jpg（f）中可以看出，相比于传统修复，PPR 利用其并行性还可以大幅加快计算时间。

总的来说，PPR 相比于传统修复各方面都进行了较多优化，此外 PPR 还提出了名为 m-PPR 的用于处理并发故障的调度算法，以减少争取共享资源的冲突。PPR 改善了传统修复带宽使用分布严重倾斜的缺点，通过并行技术来减少修复时间。尽管如此，PPR 对于带宽的使用分布并未完全平衡，靠近待修复节点的中间节点的下行链路仍承载了更多的修复流量。针对此结论，学术界又提出了 RP 算法，以完全平衡带宽的使用。

RP 算法使用的拓扑结构为一条流水线链路，通过数据块的进一步分片将修复分解为更小的修复单元，然后通过流水线链路对每个修复单元进行流水线传输。若假设将 1 个数据块分为 s 个数据片，整个修复过程的理论修复时间为$t(1+\frac{k-1}{s})$，其中 t 为单块传输时间，k 为编码参数，s 为切片数。当 s 趋于无穷时，修复时间收敛为单块传输时间。测试结果表明（fig5.jpg（a）），当切片大小越大时，修复时间减少越多。图（b）显示了单块修复时间与块大小的关系，可以看出块大小增加时，相比于传统算法和 PPR 算法，RP 算法的增加并不显著。图（c）显示了 RP 算法的单块修复时间几乎不会随 k 变化而增加。图（d）显示 RP 算法在全节点修复同样具有优势。

PPR 算法和 RP 算法在均匀网络的情景下，通过并行、流水线等技术将纠删码系统的修复时间在传统修复手段的基础上大幅减少。但是，实际的分布式系统中网络一般是异构的。因此，有必要考虑在非均匀网络中如何进行纠删码的修复工作，PPT 算法就是基于以上考虑提出的。

学者观察到，在一个三角形网路中，当两条链路带宽中较小的那个值的一半仍大于第三条链路的带宽时，可以选择那两条链路并行传输代替第三条链路。基于以上观察，PPT 被提出了。PPT 通过枚举获得瓶颈最小的路由方案，并在此基础上进行流水线传输，从而最大限度减少修复时间。

经过测试，从 fig6.jpg（a）可以看出在图中的几种纠删码配置下，相比于 PPR 和 RP，PPT 拥有更短的修复时间。从（b）可以

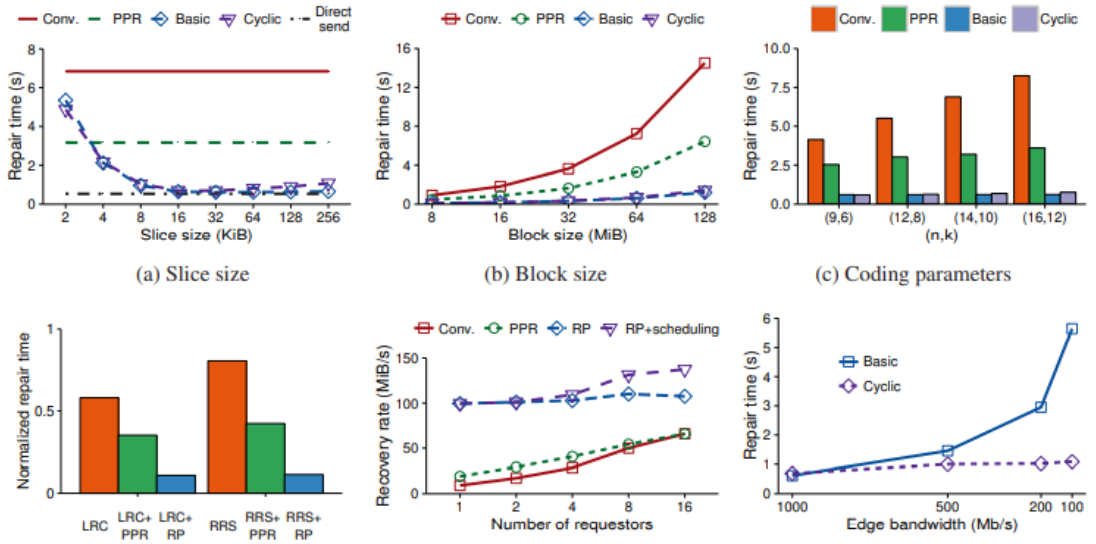
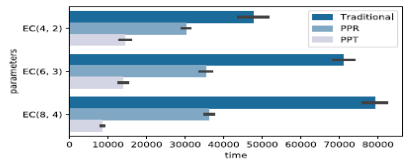
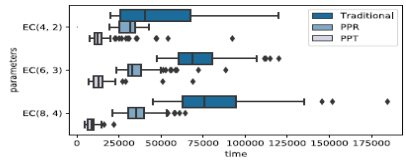


fig 5.jpg



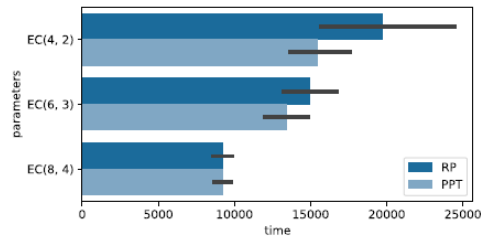
(a) The average recovery time with PPT over traditional technique and PPR.



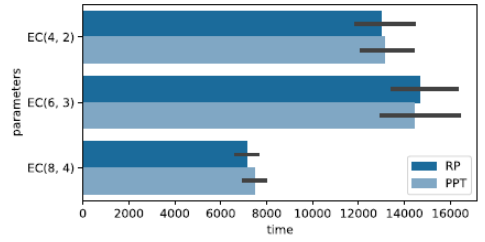
(b) The maximum, minimum, median and interquartile range of recovery time with PPT over traditional technique and PPR.

fig 6.jpg

看出，相比于传统修复和 PPR，PPT 的总体表现更优。fig7.jpg 显示，在带宽差异较大的非均匀网络中，PPT 表现比 RP 好。但是当带宽差异较小时，即使在非均匀网络中，RP 的表现反而会更好。PPT 的作者认为这微小的差异是由网络环境的波动引起的。但是，这可能是由于在带宽差异较小的网络中，PPT 计算得到的路由链路与 RP 相同，PPT 额外付出了计算时间，最终导致了 PPT 的表现略逊一筹。总的来说，PPT 的提出是对 PPR 和 RP 的一种补充，因为在真实数据中心中，节点之间的网络应该是异构的，这就不可避免地会导致带宽差异。PPT 针对非均匀网络，提出了绕开瓶颈链路的方案，有效提升了修复效率。但是，PPT 路由方案生成时间较慢，



(a) The average recovery time in non-uniform network with big bandwidth gap.



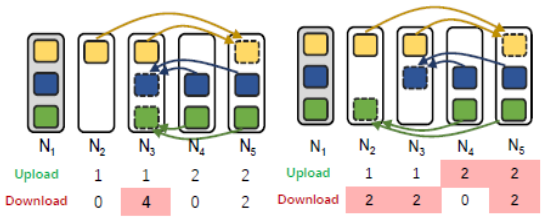
(b) The average recovery time in non-uniform network with small bandwidth gap.

fig7.jpg

不适合应用在编码参数较大的纠删码系统中。因为可能会出现减少修复时间的收益弥补不了计算路由方案带来的负担。

此外，上述算法主要关注单块修复，而当全节点故障（一个节点中的所有块均丢失）必须同时处理多个块的修复。因此，现有的修复算法的部署和全节点修复的需求之间的耦合并不紧密。且导致了以下局限：1）这些算法没有专门利用全双工传输来使可用带宽饱和，见图 fig8.jpg；2）没有针对全节点修复的数据块传输调度，导致了带宽未被充分利用，见图 fig9.jpg；3）现有修复算法不能弹性组合不同的修复算法，并使它们协作以满足不同的可靠性要求；4）缺乏全节点修复的通用框架。因此，针对全节点修复，RepairBoost 被提出。RepairBoost 基于以下几点设计：1）修复抽象，RepairBoost 将修复过程抽象成修复有向无环图（简称 RDAG）；2）修复流量均衡，RepairBoost 通过将节点与 RDAG 中的顶点按照一定的规则对应来均衡每个节点的修复流量；3）传输调度，RepairBoost 将全节点修复中的调度抽象成一个最大流的问题，通过解决最大流问题来生成调度方案。经过测试，RepairBoost 能有效提升全节点修复的性能。

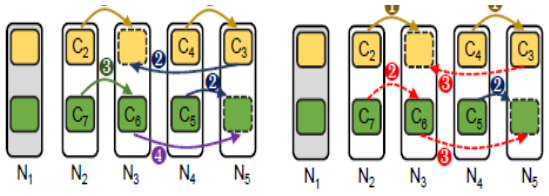
以上算法的研究对象均是小条带（一个



(a) Unbalanced repair solutions (b) Balanced repair solutions

Figure 3: L1 (Failing to utilize the full duplex transmission). The repair time is determined by the most loaded node (with the red color in the background).

fig 8.jpg



(a) Repair using four timeslots (b) Repair using three timeslots

Figure 4: L2 (Failing to fully utilize the bandwidth at each timeslot). The transmission scheduling affects the bandwidth utilization.

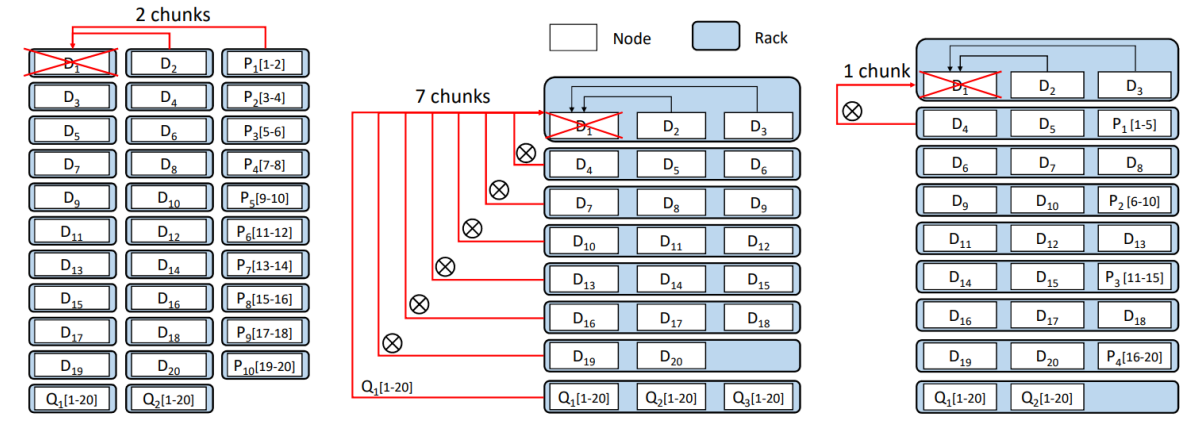
fig 9.jpg

条带中的块数小于 20）。然而，最新的云存储系统考虑通过大条带纠删码存储来进一步减少存储开销。根据研究，对于最新的生产系统所使用的参数（n，k），可容忍的故障数目通常为 3 或 4，而条带的大小 n 不超过 20（适当大小的 n 可以减少纠删码在维修时付出的代价）。然而，当编码参数（n，k）较大的同时保持可容忍故障的数目时（即 m = n - k 的取值仍为 3 或 4），可以进一步减少编码时所带来的冗余。例如，VAST 考虑（n，k）=（154，150）的设置，仅产生 1.027 倍的冗余。大条带所带来的高存储效率对于冷分布式存储系统和热分布式存储系统均有一定的吸引力。大条带能使冷分布式存储系统以极低的存储成本实现长久存储（冷存储系统中的数据需要长期存储但很少访问）；同时，大条带能显著减少热分布式存储系统较为昂贵的硬件成本。

虽然大条带可以进一步减少纠删码系统的冗余度，减少存储开销，但与此同时纠删码存储系统的修复问题被进一步放大。因此，对于大条带的研究尤其是其修复、更新等 I/O 放大问题逐渐成为纠删码研究的主流方向之一。胡烯翀教授在 FAST21 中提出了 ECWide，利用校验局部性和拓扑局部性来缓解大条带纠删码存储系统的修复问题。如 fig10.jpg 所示。校验局部性指的是在 RS 码的基础上加上局部校验块以减少修复流量。拓扑局部性是通过减少跨机架传输的流量来减少修复时间，因为跨机架传输的速度远远慢于机架内部传输。ECWide 利用组合局部性来减少修复时所带来的成本，提高了大条带在实际系统中的可用性。

4 总结与展望

在互联网高速发展的今天，数据呈爆炸式增长。因而存储系统往往部署在大量存储节点上，这导致了故障普遍发生。因此，如何容错以提高系统的可靠性成为了关键性的问题。主流的容错技术有：副本技术和纠删码技术。相比于副本技术，纠删码技术可以在保障相同容错的情况下具有更低的存储开销。然而，纠删码存在修复 I/O 放大问题。以 RS（k，m）纠删码系统为例，需要 k 个幸存块来修复单个块。因此，如何有效地进行修复成为了纠删码系统的主要研究方向之一。PPR、RP、PPT 等修复算法的提出指示了可以从并行、流水线等方向来解决纠删码系统中的修复 I/O 放大问题。同时，RepairBoost 利用修复有向无环图、最大流问题来解决全节点修复，弥补了学术界在全节点修复中研究的不足。综上，学术界目前面对纠删码系统的修复问题已经有了比较成熟的方案。但是，最新的云存储系统又考虑通过大条带以进一步减少存储开销。然而，大条带进一步放大了修复 I/O，已有的修复算法已不适用。因此，ECWide 被针对性提出，利用组合局部性来减少修复流量。然而，目前仍没有基于大条带的节点修复算法来解决，这也是未来学术界的研究方向之一。



(a) Parity locality: (32, 20, 2) Azure-LRC　(b) Topology locality: (23, 20, 8) TL　(c) Combined locality: (26, 20, 5, 9) CL

fig 10.jpg