On Fault Tolerance, Locality, and Optimality in Locally Repairable Codes

Oleg Kolosov$^1$, Itzhak Tamo$^1$, Gala Yadgar$^2$, Matan Liram$^2$, Alexander Barg$^3$

Tel Aviv University$^1$, Technion$^2$, University Of Maryland$^3$
Availability with Reed-Solomon

An \((n, k)\) erasure code with \(k\) data blocks

- \(k\) data blocks
- \(n - k\) parity blocks

- Low overhead
- Can recover from at most \(n - k\) failures \(\Rightarrow\) minimal redundancy (MDS)
- Required reading \(k\) blocks for lost block recovery
Locally Repairable Codes (LRC)

Locality

Global parity

Local group

Local parity

$(n, k, r)$ LRC → $(10, 6, 3)$ Azure-LRC

Distance

Minimum #failures that cause data loss

d = 4 (can recover from 3 failures)

Non-MDS (non-optimal overhead)

Fast recovery (good for degraded read)

Huang et al. 2012
Huang et al. 2013
Sathiamoorthy et al. 2013
Node failure and reconstruction

× Non-MDS (non-optimal overhead)
✓ Fast recovery
× Slow recovery of global parity
Recovery of global parity blocks

**Optimal-LRC**

Full-LRC (vs. data-LRC)  [also *information-symbol locality vs. all-symbol locality*]

Optimal $d$ for a variety of combinations (but not for all...)

Optimal minimum distance (full-LRC)

\[ d = n - k - \left\lceil \frac{k}{r} \right\rceil + 2 \]

Gopalan et al. 2012

Tamo and Barg, 2014
Which one is better?

Overhead = 1.66
Locality = 3
Distance = 4

Overhead = 1.83
Locality = 3
Distance = 4

→ There is no mathematical framework for comparison of existing LRC approaches
→ What’s optimal in practice?
Goals and methodology

Goal: Lay mathematical basis for comparison

- Define parameters for comparison
- Compare codes in all sets of configurations – extend LRCs
- Compare for $9 \leq n \leq 19$ and overhead $\leq 2$
- Validate in a real system
Measuring repair costs

Previously:

Average repair cost (ARC) = \( \frac{\sum_{i=1}^{n} cost(b_i)}{n} \)

→ Doesn’t address overhead

Our contribution:

Normalized repair cost (NRC) = \( \frac{\sum_{i=1}^{n} cost(b_i)}{\kappa} \)

Degraded cost (DC) = \( \frac{\sum_{i=1}^{k} cost(b_i)}{\kappa} \)

→ Useful for degraded read

Overhead: +16.6%
ARC: -24.1%
NRC: -16.6%
DC: 0%
Our LRC extensions

**Optimal-LRC**
- New construction
- Achieves optimal $d$

**Azure-LRC**
- Removed division constraints

**Azure-LRC+1**
- Full-LRC extension of Azure-LRC

**Xorbas**
- A trivial extension
Which construction is best for my system?

ARC

Cost (blocks)

NRC and Degraded cost

Cost (blocks)
Want to maximize $d$ and minimize NRC

New metric $\frac{NRC}{d}$ (rd-ratio)

$\rightarrow$ Optimal-LRC is best for fixed $(n, k)$
System level evaluation setup

**Goals:**
- Validate NRC accuracy
- Evaluate NRC abilities of estimation
- Compare LRCs

**Platform:**
- Ceph – a distributed open-source object-based storage system
- Amazon EC2

**Methods:**
- Utilize Ceph LRC plugin for Azure-LRC
- Implement Optimal-LRC
- Simulate failure and measure
Predicting repair time?

→ NRC can’t predict accurately – but it can predict a trend
→ Overall, full-LRCs outperform data-LRC

**Also validated on**(in the paper): 
→ Various storage types 
→ Various network architectures 
→ Application workloads
Summary

• **First systematic comparison of LRCs**
  - Defined theoretical framework for comparison of LRCs
  - Validated on a real system

• **Generalized known LRC codes**

Conclusions

→ ARC is limited – we introduced NRC
→ There is no one optimal code (theory vs. practice)
→ Optimize repair cost ≠ optimize degraded cost

Our Ceph implementation can be found here:

Thank you