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

Oleg Kolosov¹, Itzhak Tamo¹, Gala Yadgar², Matan Liram², Alexander Barg³

Tel Aviv University¹, Technion², University Of Maryland³
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**: \( r = 3 \)
- **Distance**: \( d = 4 \) (can recover from 3 failures)

- **Non-MDS (non-optimal overhead)**
- **Fast recovery (good for degraded read)**

\[(n, k, r)\text{ LRC }\rightarrow (10,6,3)\text{ Azure-LRC}\]

- 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

Overhead = 1.66
Locality = 4
Distance = 4

→ There is no mathematical framework for comparison of existing LRC approaches

Goal: Lay mathematical basis for comparison

→ What’s optimal in practice?
Measuring repair costs

Previously:

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

→ Doesn’t address overhead

Our contribution:

Normalized repair cost (NRC) =
\[ \text{ARC} \times \text{Overhead} = \sum_{i=1}^{n} \frac{\text{cost}(b_i)}{\kappa} \]

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

→ 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

NRC and Degraded cost
Durability & repair cost

Want to maximize $d$ and minimize NRC

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

→ 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