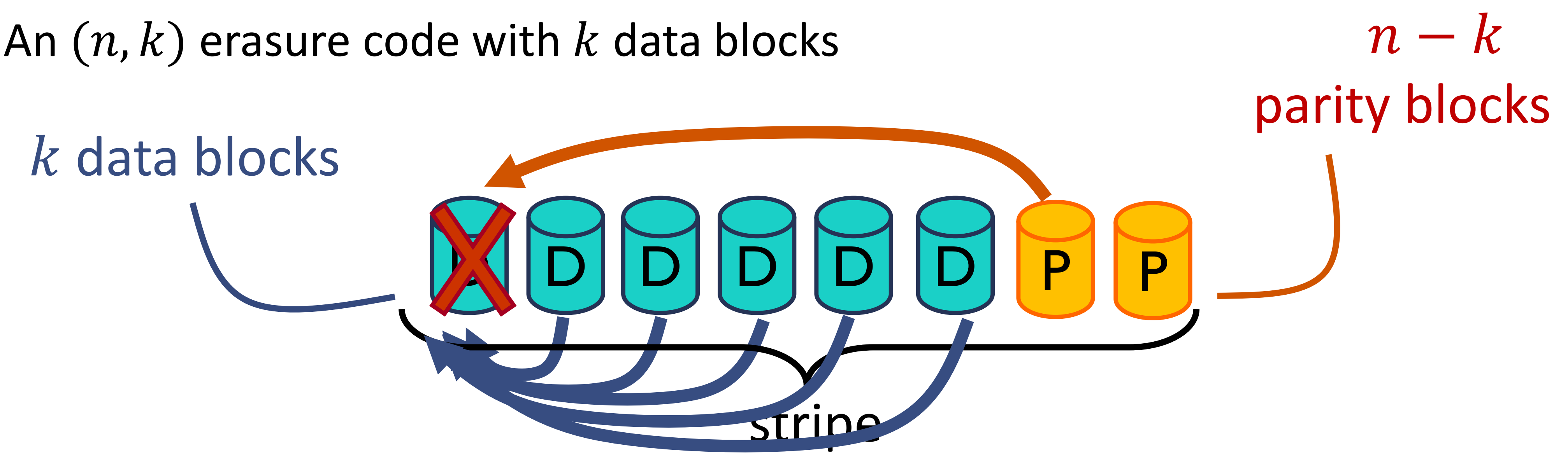
On Fault Tolerance, Locality, and Optimality in Locally Repairable Codes

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]

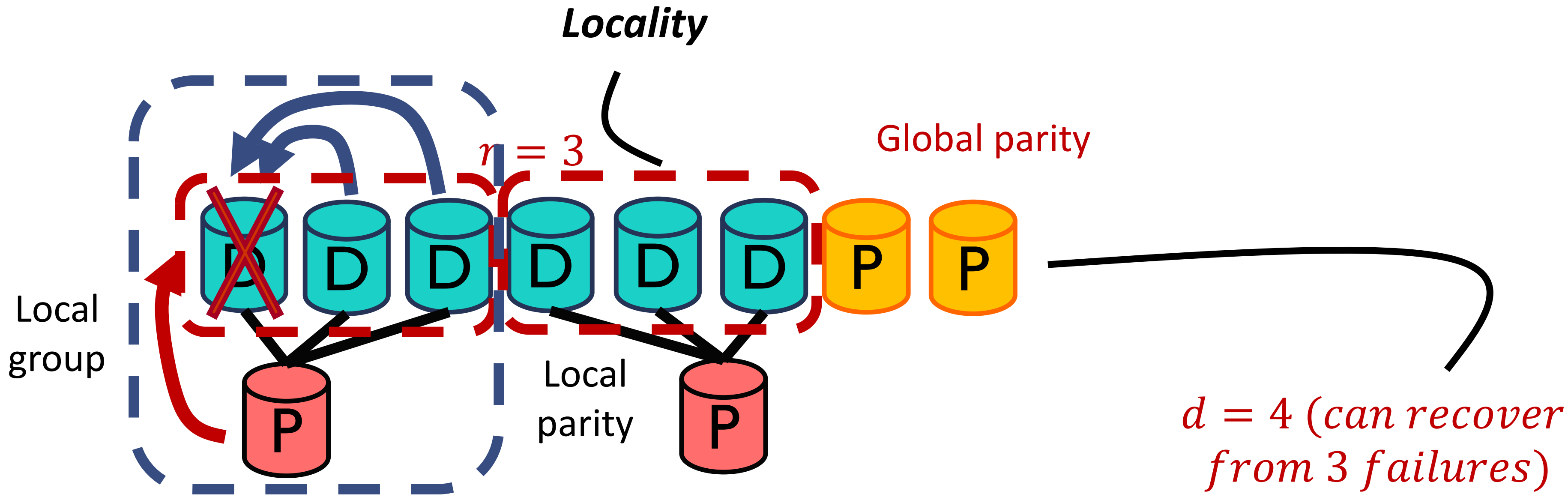
Availability with Reed-Solomon

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



- ✓ Low overhead
- ✓ Can recover from **at most** $n - k$ failures → minimal redundancy (MDS)
- ✗ Required reading k blocks for lost block recovery

Locally Repairable Codes (LRC)



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

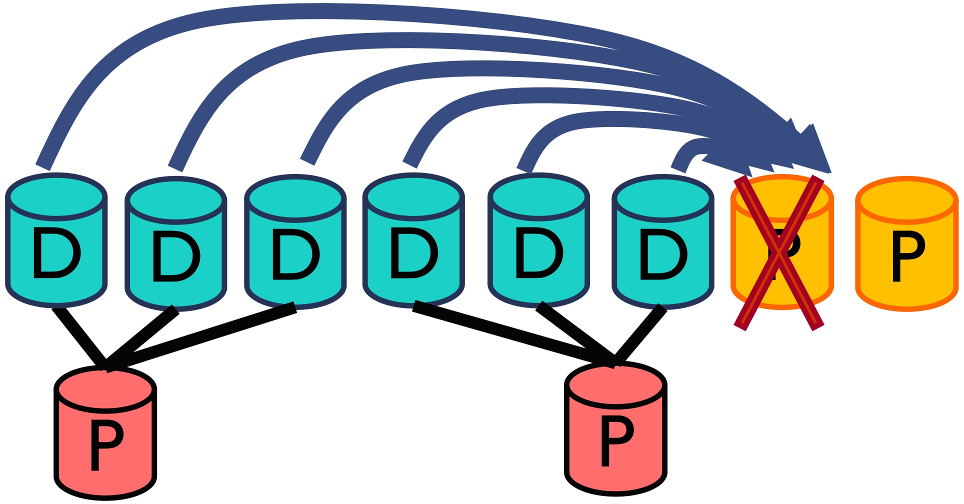
Distance
Minimum #failures that cause data loss

✗ 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**

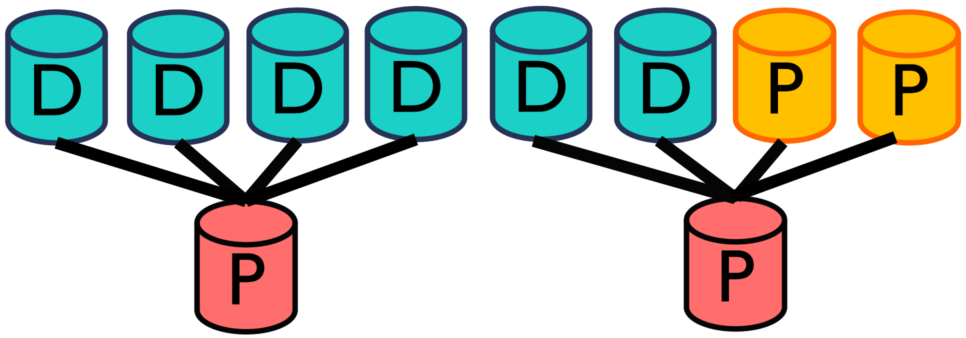
Recovery of global parity blocks

Optimal-LRC

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

Optimal $\boldsymbol{d}$ for a variety of combinations (but not for all…)

Tamo and Barg, 2014



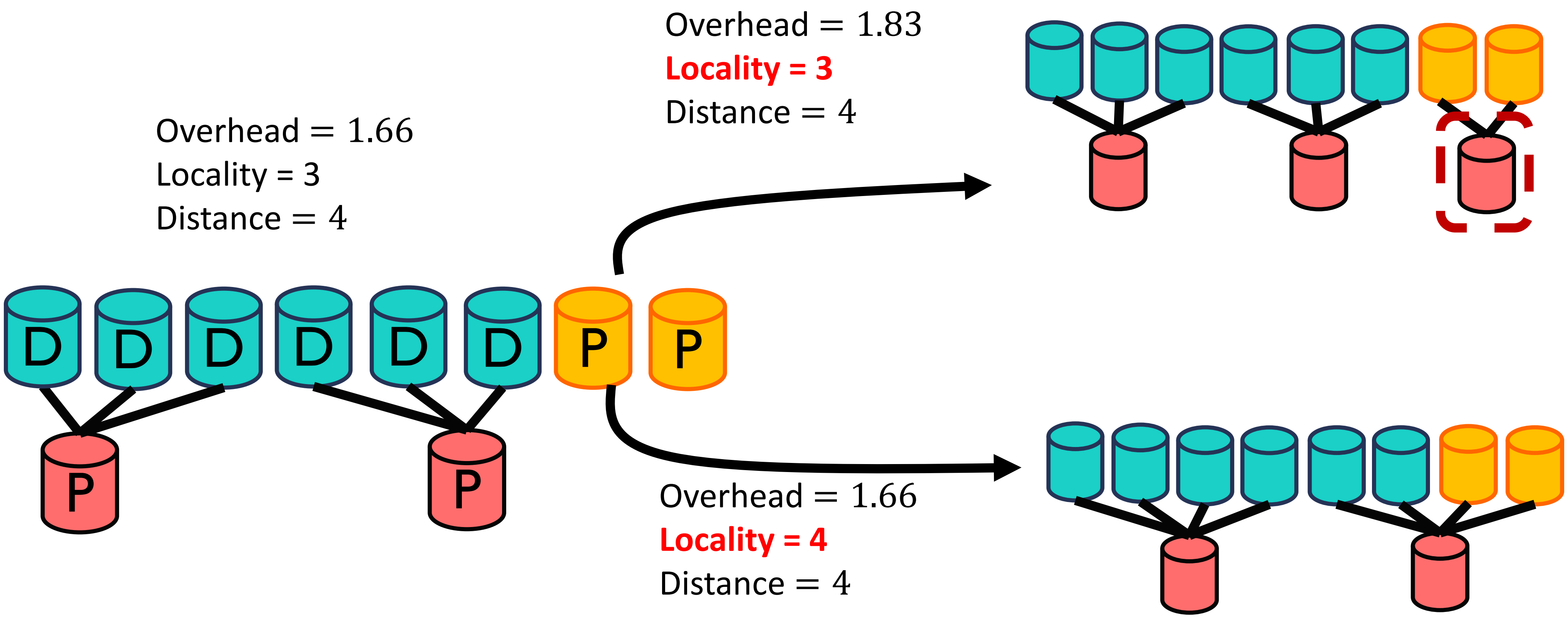
(10,6,4) Optimal-LRC

Optimal minimum distance (full-LRC)

$$d = n - k - \lceil {}^{k}/_{r} \rceil + 2$$

Gopalan et al. 2012

Which one is better?



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

<u>**Goal: Lay mathematical basis for comparison**</u>

→ **What's optimal in practice?**

Measuring repair costs

Previously:

Average repair cost (ARC) =

$$= \frac{\sum_{i=1}^{n} cost(b_i)}{n}$$

→ Doesn't address overhead

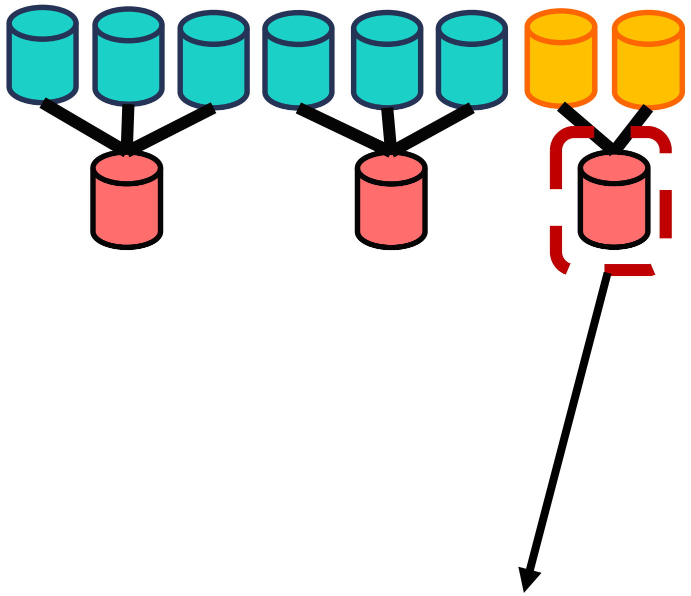
Our contribution:

Normalized repair cost (NRC) =

ARC X Overhead = $\frac{\sum_{i=1}^{n} cost(b_i)}{k}$

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

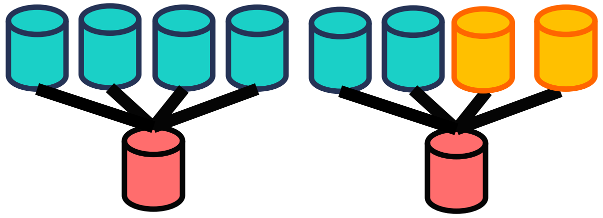
→ Useful for degraded read



Overhead: +16.6%
ARC: -24.1%
NRC: -16.6%
DC: 0%

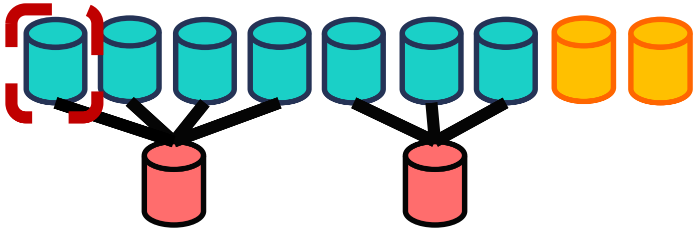
Our LRC extensions

Optimal-LRC



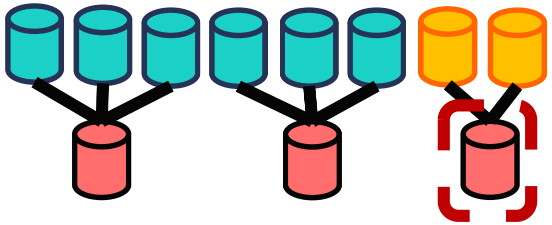
- New construction
- Achieves optimal $\boldsymbol{d}$

Azure-LRC



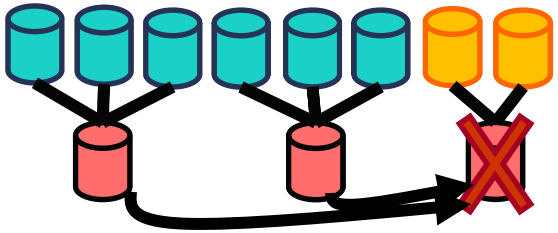
- Removed division constraints

Azure-LRC+1



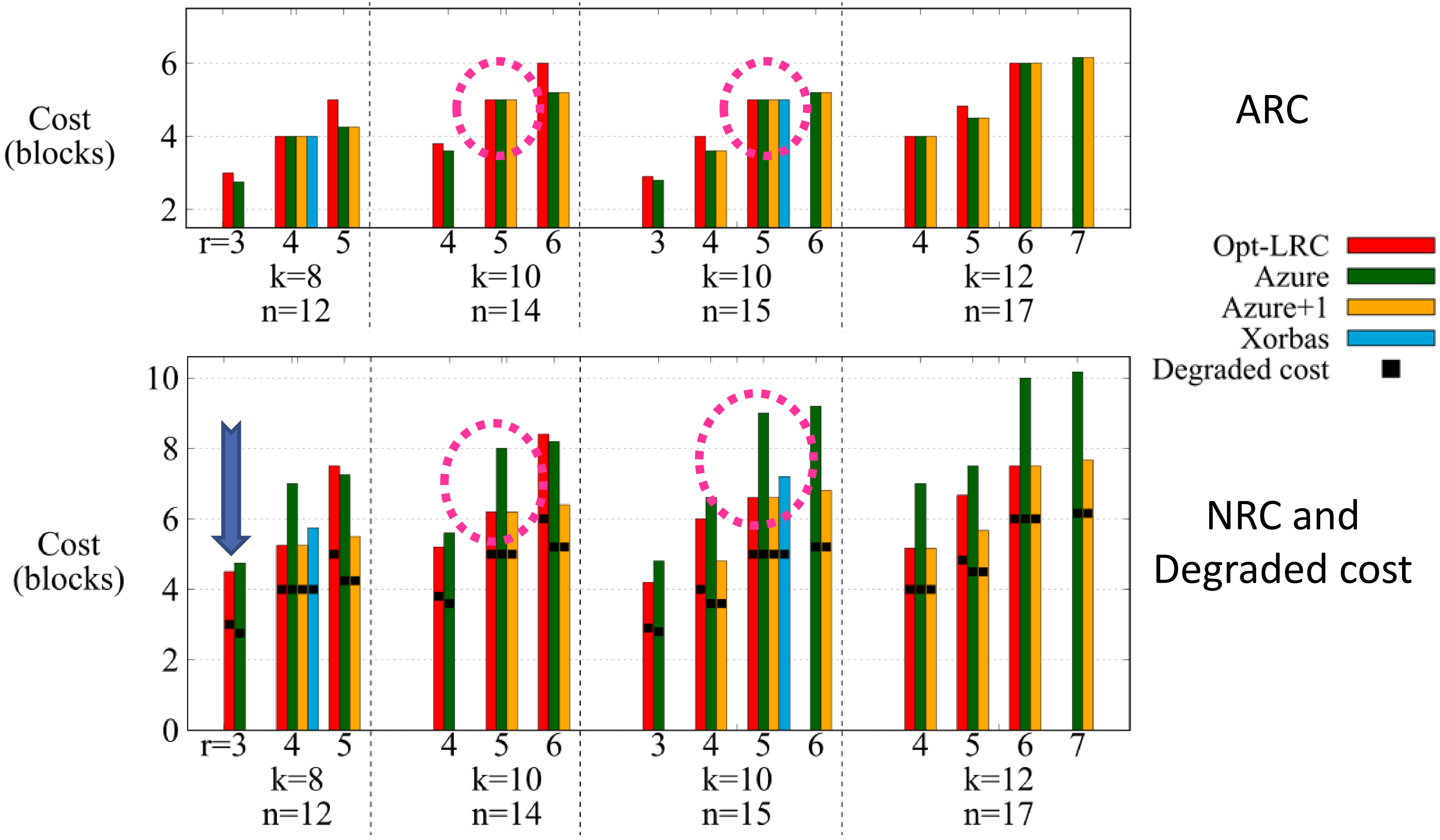
- Full-LRC extension of Azure-LRC

Xorbas



- A trivial extension

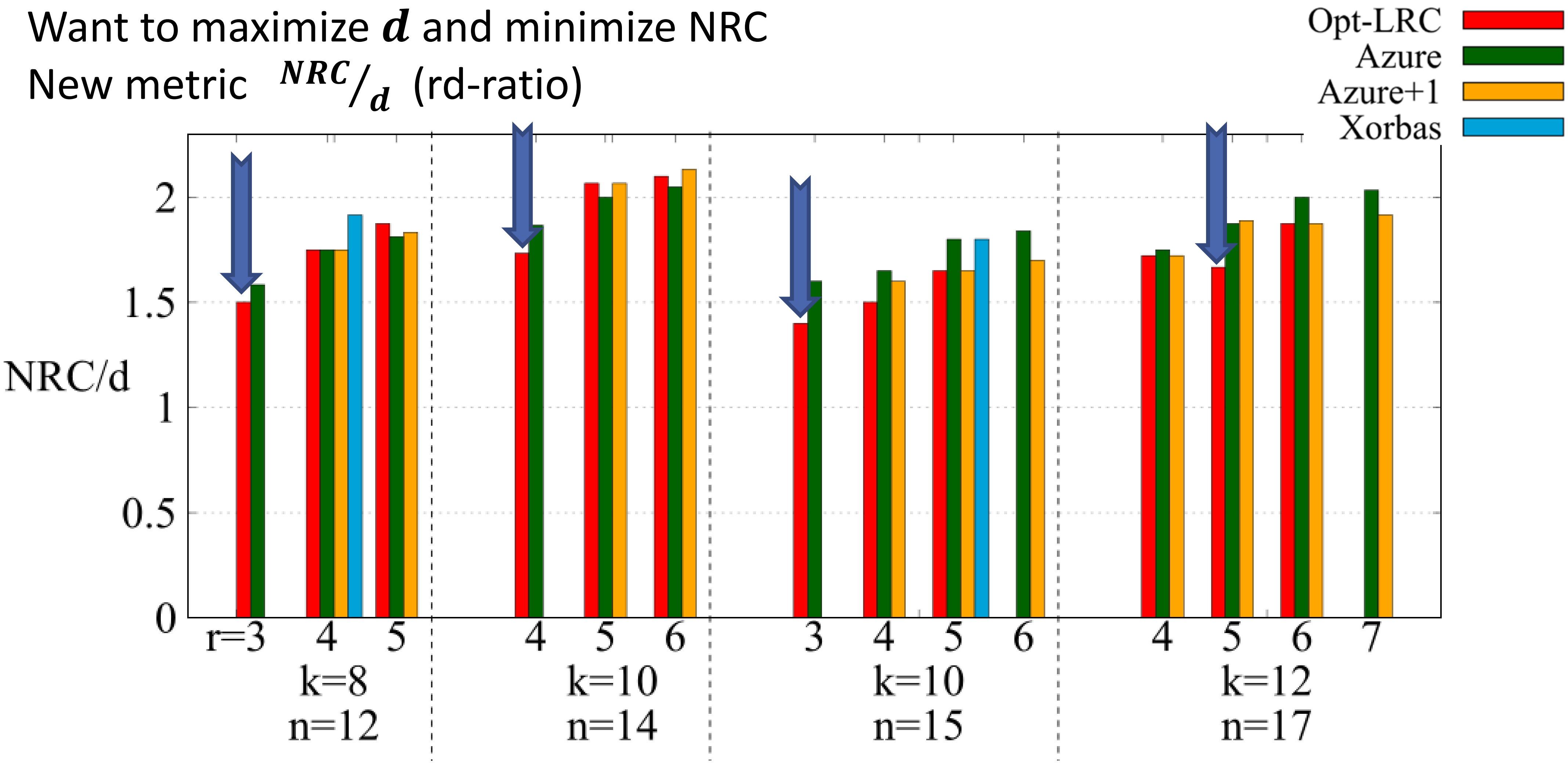
Which construction is best for my system?



Durability & repair cost

Want to maximize $\boldsymbol{d}$ and minimize NRC

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



→ Optimal-LRC is best for fixed $(\boldsymbol{n}, \boldsymbol{k})$

System level evaluation setup

Goals:

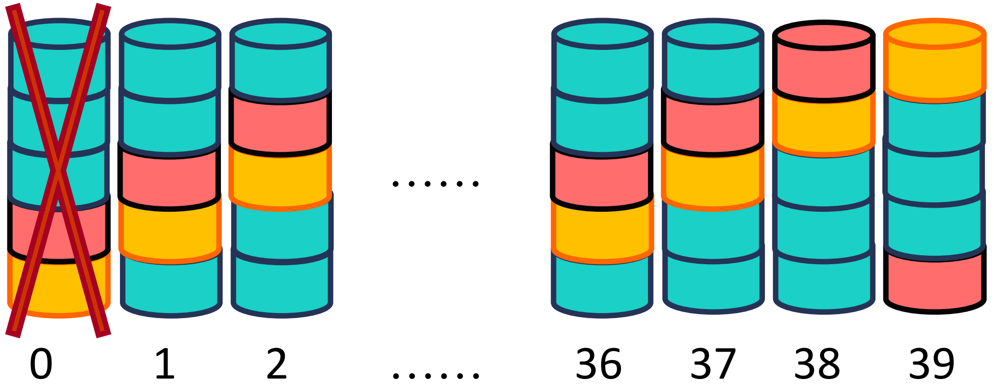
- Validate NRC accuracy
- Evaluate NRC abilities of estimation
- Compare LRCs

Platform:

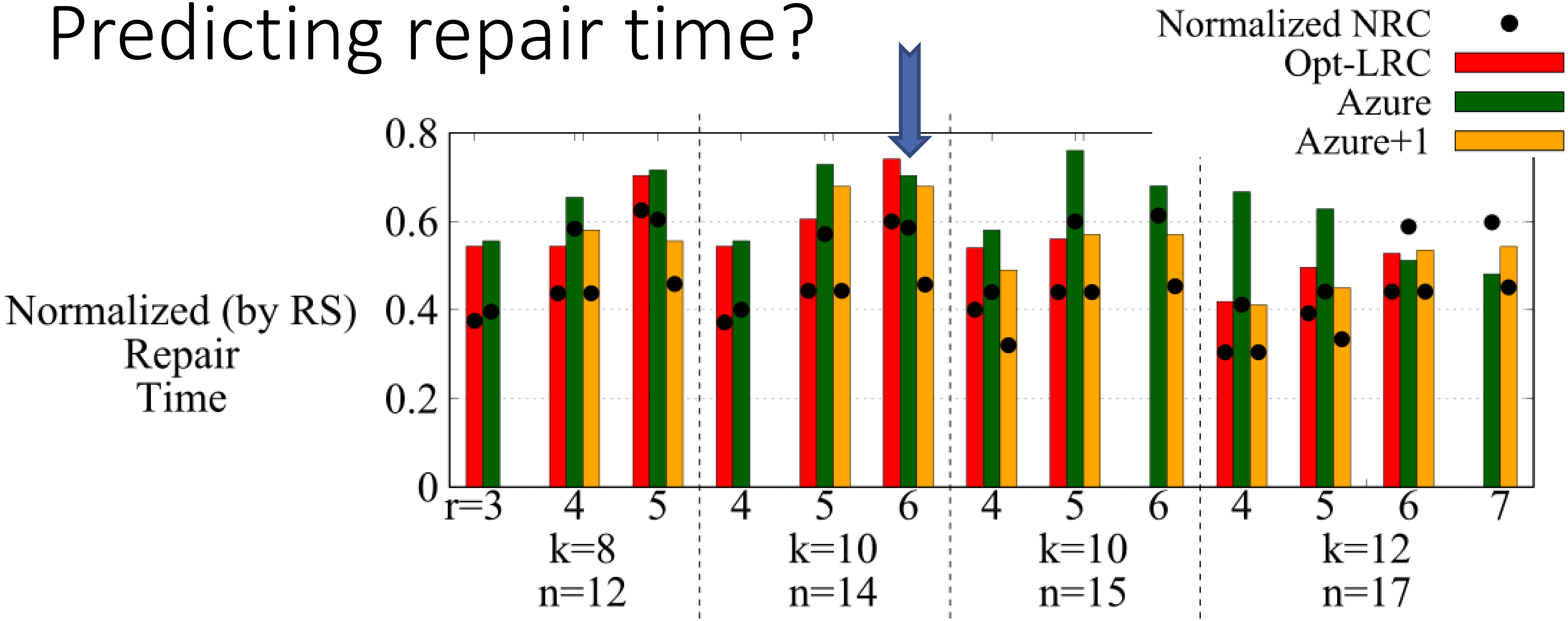
- Ceph – a distributed open-source object-based storage system
- Amazon EC2

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

- **<u>First systematic comparison of LRCs</u>**
  - o Defined theoretical framework for comparison of LRCs
  - o Validated on a real system
- **<u>Generalized known LRC codes</u>**

Conclusions

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

Thank you

Our Ceph implementation can be found here:
https://github.com/olekol33/optlrc2018/blob/master/src/erasure-code/optlrc