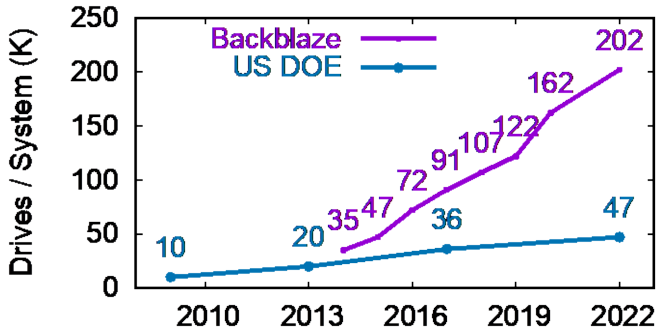
# Analysis and Design Considerations of Multi-level Erasure Coding in Hierarchical Data Centers

**Meng Wang**, Jiajun Mao, Rajdeep Rana, John Bent, Serkay Olmez,
Garrett Wilson Ransom, Anjus George, Jun Li, and Haryadi S. Gunawi

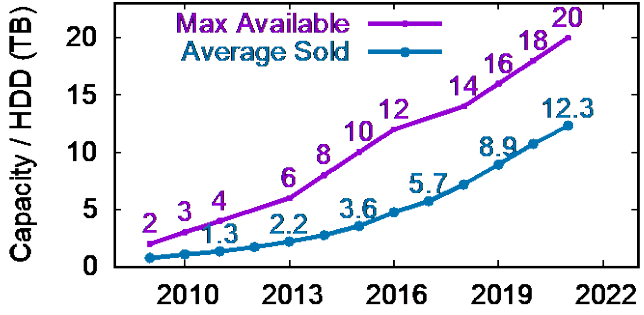
# Introduction

- We store data in disks. Unfortunately, disks fail!




<u>Growing number</u> of disks in data centers

- More disk failures



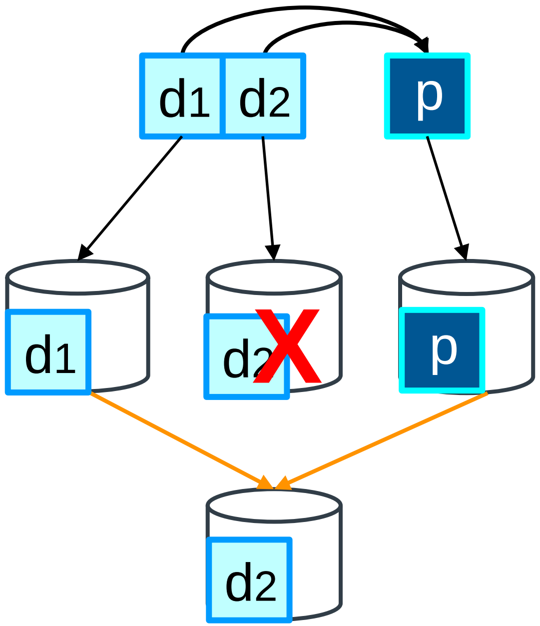

<u>Larger</u> disk capacity

- Longer rebuild time

**Better data protection approach** is needed!

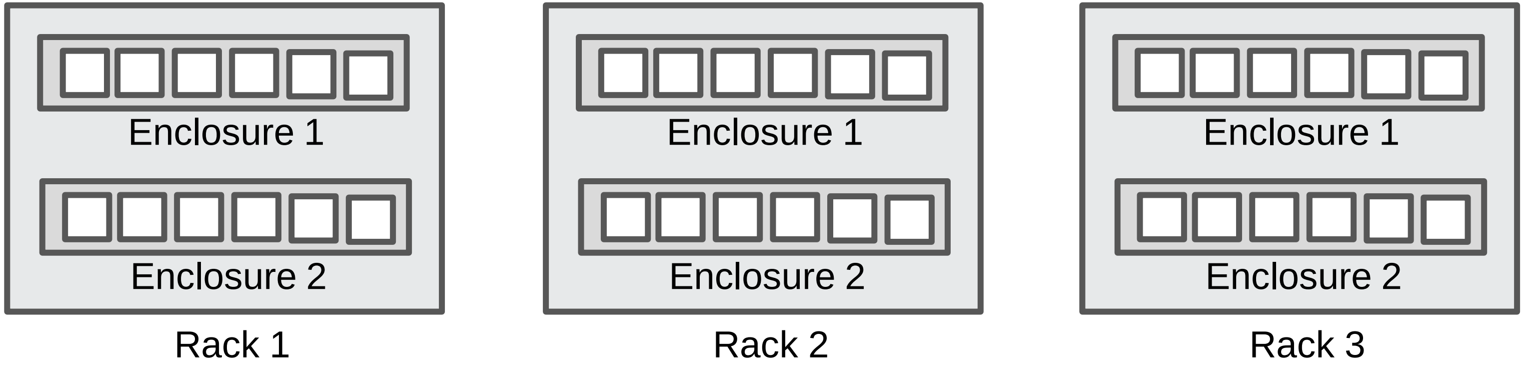
# Existing Solutions

- **Erasure Coding (EC)**
  - (K+P)
    - Data is split into K data chunks
    - P parities are computed
    - Stripe: every (K+P) chunks
  - Example: *2+1*
    - Tolerate any single failure
    - 1.5x storage
  - What if you want to tolerate more failures?
    - More parities!
      - 4+2
      - 6+3



# EC at Scale

- A large-scale data center is usually hierarchical
  - Racks
    - Enclosures
      - Disks
- How to deploy EC in a large-scale data center?

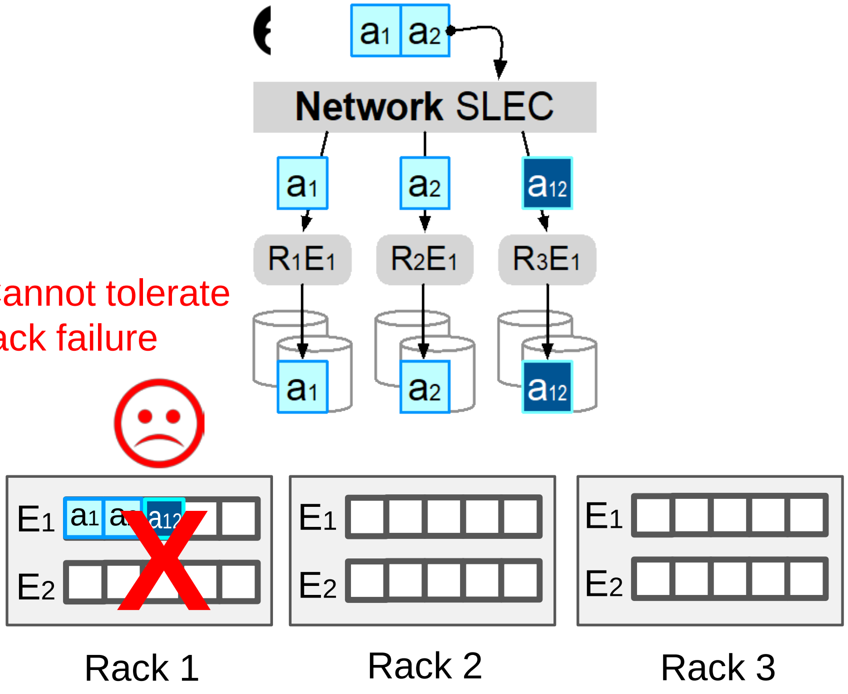


# Single-level Erasure Coding

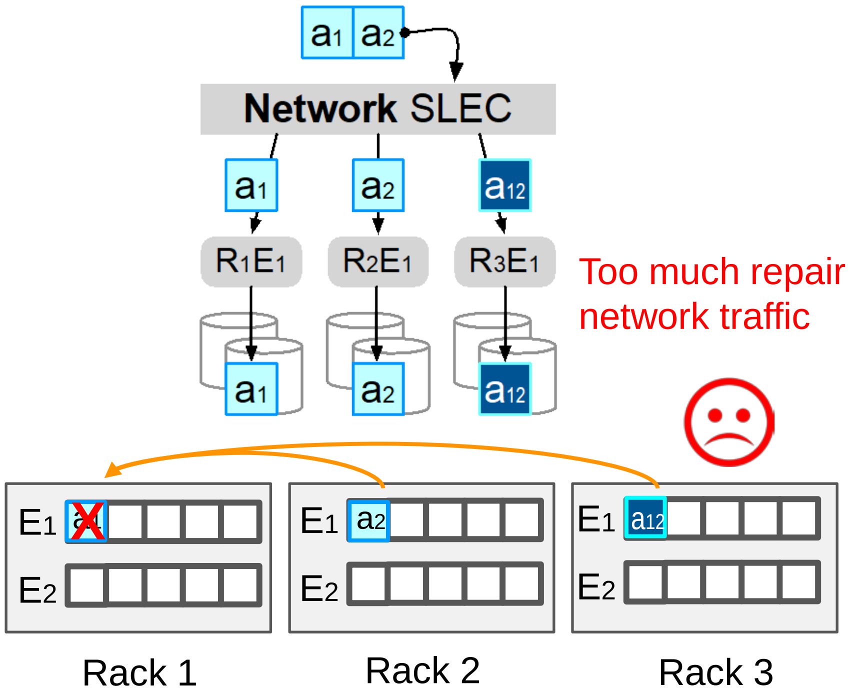
$E_1$: Enclosure 1
$R_1$: Rack 1

- **SLEC:** Single-level Erasure Coding

Cannot tolerate rack failure

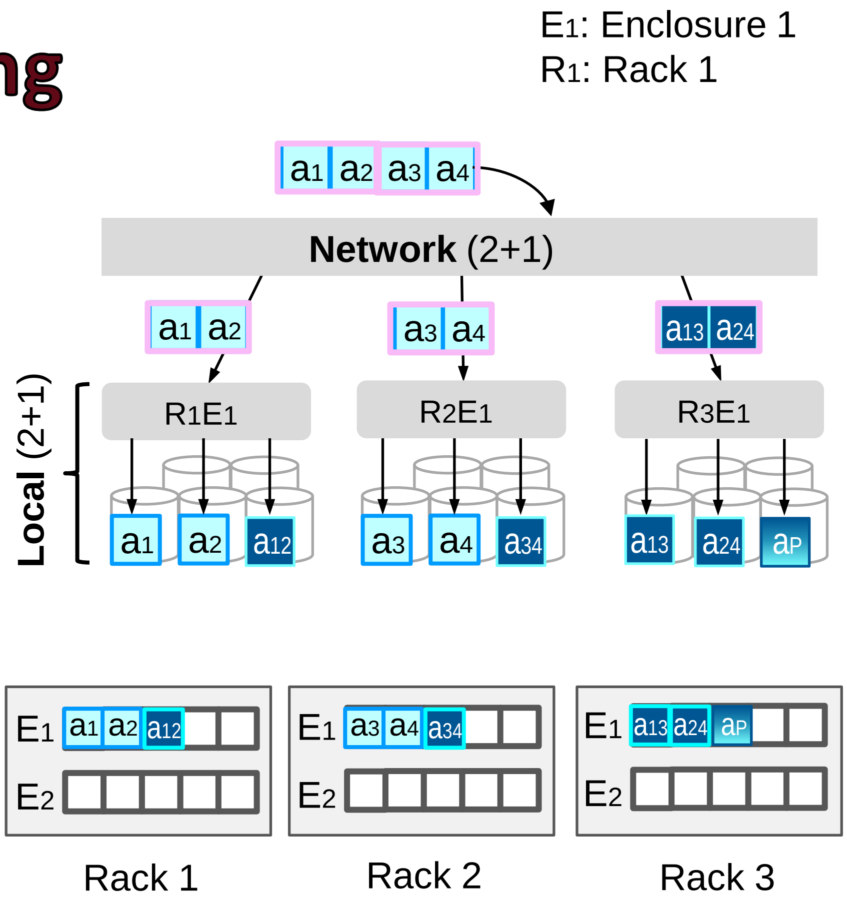



Too much repair network traffic



# Multi-level Erasure Coding

- **MLEC:** Multi-level Erasure Coding
  - Example: (2+1)/(2+1)
- Why MLEC?
  - Repair most failures locally
  - Can tolerate rack failures
  - Stackable and easy to deploy
  - Configurable



# Multi-level Erasure Coding

- MLEC has seen large deployments in practice
  - LANL MarFS
  - Scality ARTESCA

- Many research questions remain **unanswered**!

What are the possible chunk placement schemes for MLEC at scale?

What are their pros/cons in terms of performance and durability?

What are the types of failure modes an MLEC system can face?

Can we optimize repair methods to improve the performance/durability?

# MLEC at Scale

- Our work: *Comprehensive design considerations and analysis of MLEC at scale*

| | |
|---|---|
| **Chunk placement schemes** | C/C, C/D, D/C, D/D |
| **Failure modes** | Single disk failure, Catastrophic local failure |
| **Repair methods** | RALL, RFCO, RHYB, RMIN |
| **Analysis** | Performance, durability |
| **Comparison** | Vs. SLEC, LRC, … |

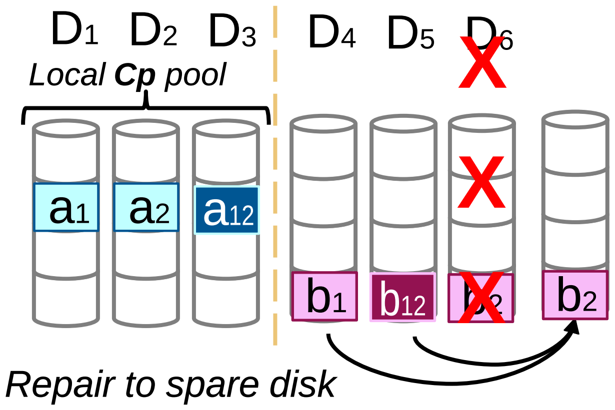
- Introduction
- MLEC Overview
- MLEC Design and Analysis
  - Chunk Placement Schemes
  - Repair Methods
- MLEC vs. Other EC Schemes
  - vs. SLEC
  - vs. LRC

# Chunk Placement Schemes

Example:
SLEC 2+1

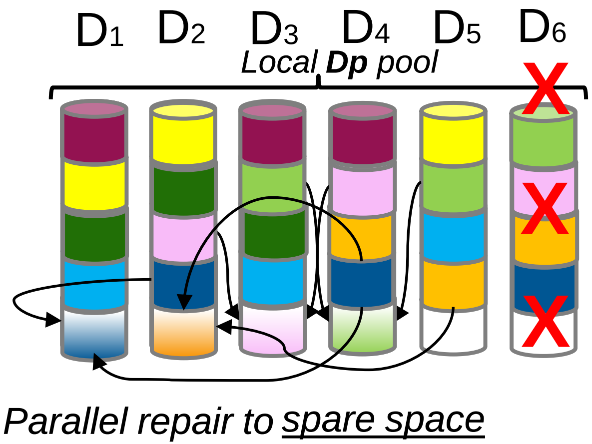
- SLEC chunk placement schemes

## **Clustered** Parity




- 3 disks participate in the repair
  - Repair speed bottlenecked by disk IO

## **Declustered** Parity

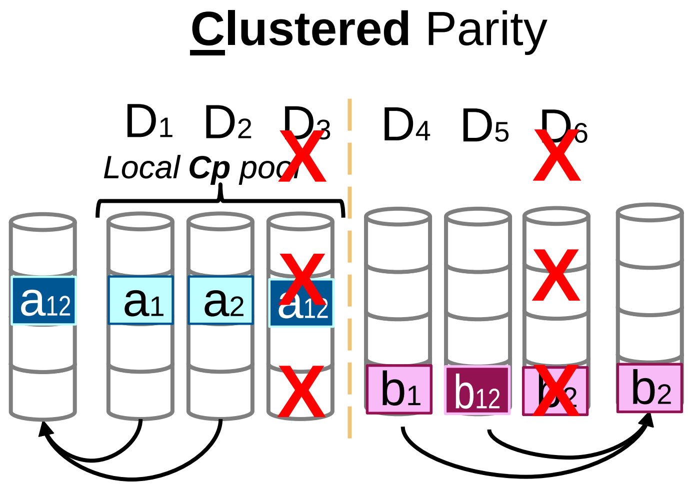



- 5 disks participate in the repair
  - **Faster repair!** ☺

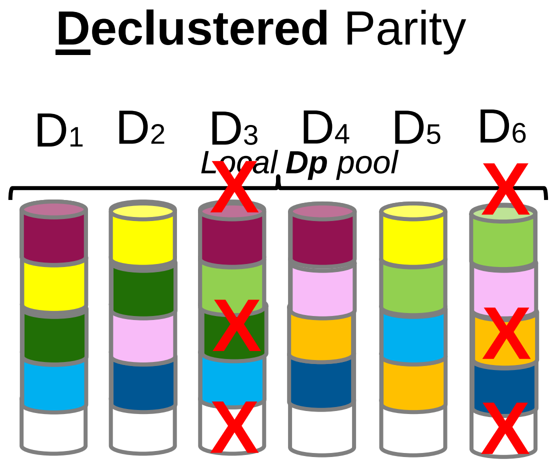
# Chunk Placement Schemes

Example:
SLEC 2+1

- SLEC chunk placement schemes




- Repair speed bottlenecked by disk IO
- If D3 and D6 fail…
  - Can survive

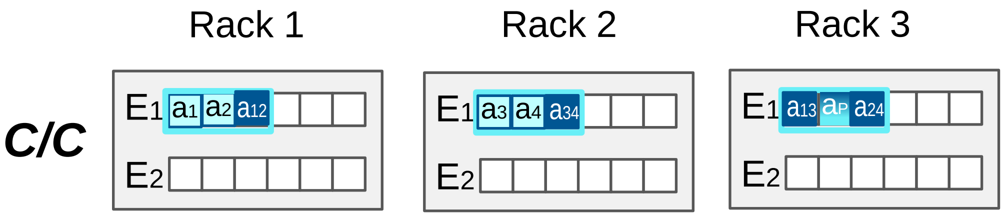



- Faster Repair
- If D3 and D6 fail…
  - **Data loss!**

# Chunk Placement Schemes

Example:
MLEC (2+1)/(2+1)

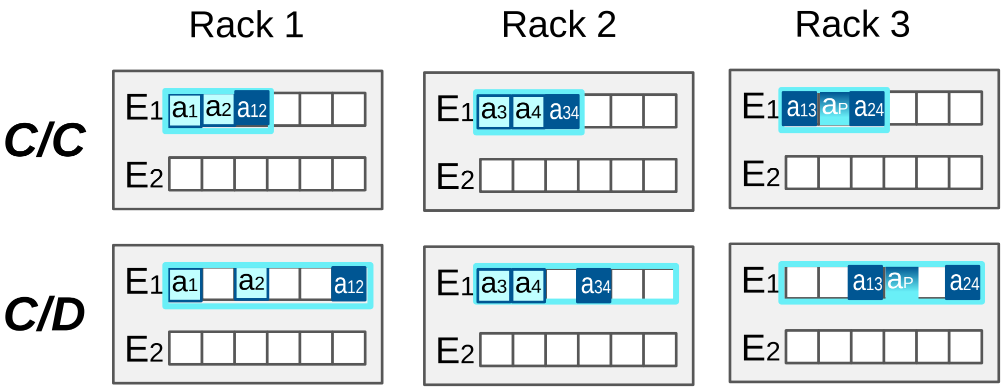
- MLEC chunk placement schemes
  - C/C
    - Clustered-Clustered



# Chunk Placement Schemes

Example:
MLEC (2+1)/(2+1)

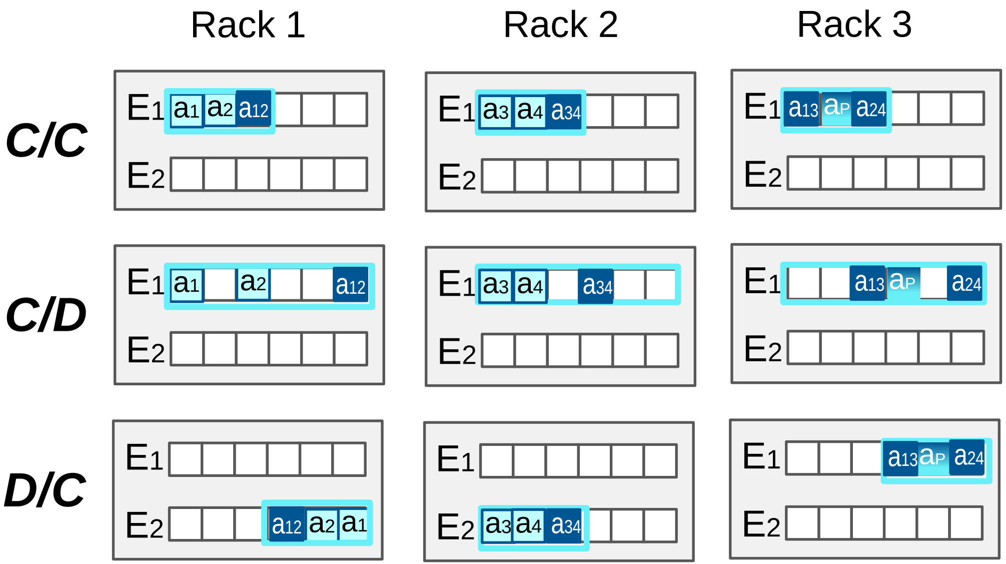
- MLEC chunk placement schemes
  - C/C
    - Clustered-Clustered
  - C/D
    - Clustered-Declustered



# Chunk Placement Schemes

Example:
MLEC (2+1)/(2+1)

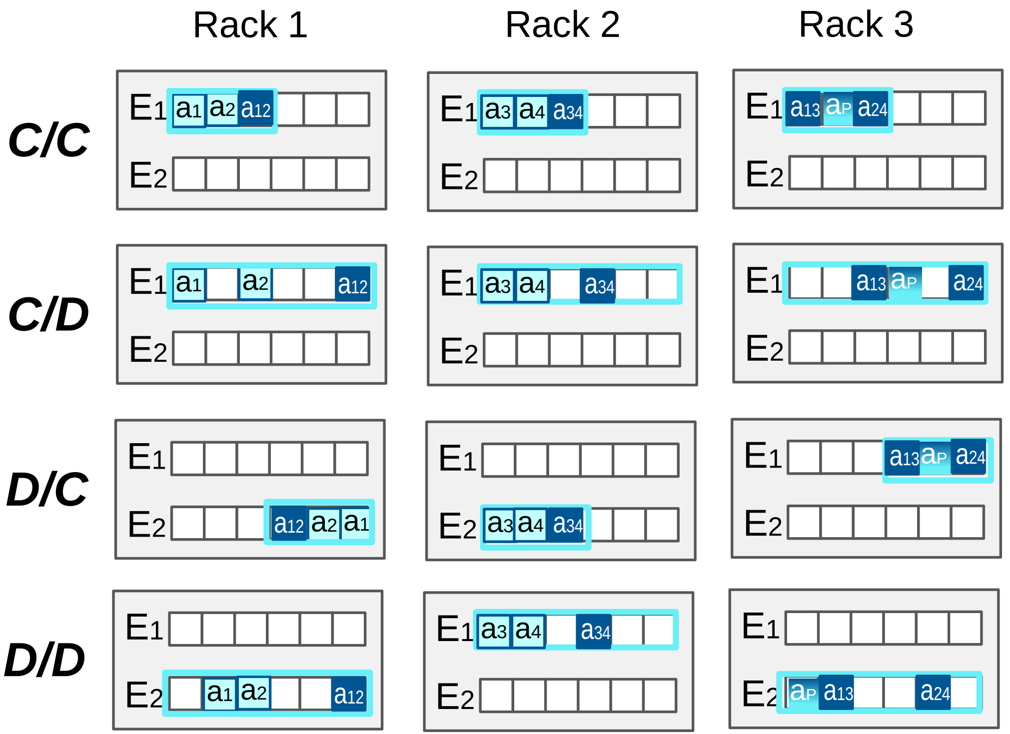
- MLEC chunk placement schemes
  - C/C
    - Clustered-Clustered
  - C/D
    - Clustered-Declustered
  - D/C
    - Declustered-Clustered



# Chunk Placement Schemes

Example:
MLEC (2+1)/(2+1)

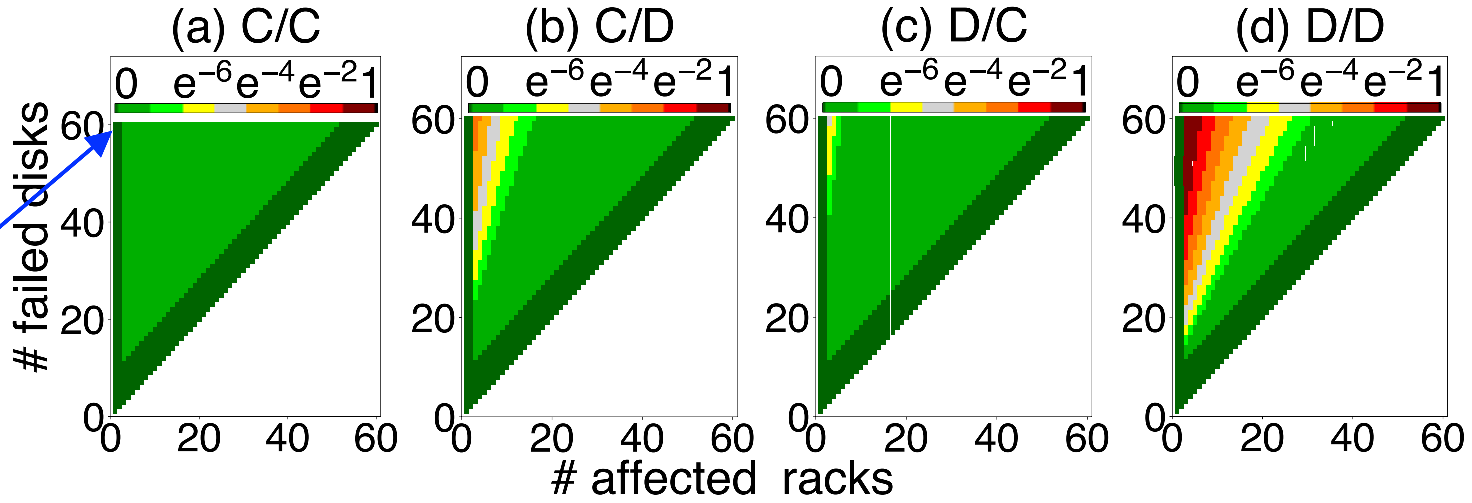
- MLEC chunk placement schemes
  - C/C
    - Clustered-Clustered
  - C/D
    - Clustered-Declustered
  - D/C
    - Declustered-Clustered
  - D/D
    - Declustered-Declustered



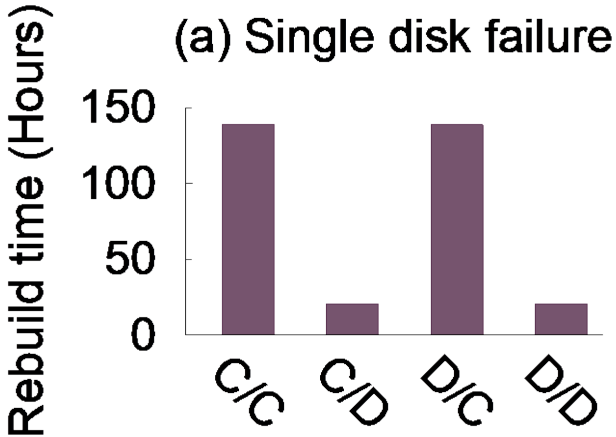
# Schemes: PDL under Failure Bursts

- Probability of data loss (PDL) under correlated failure bursts
  - 57,600 disks across 60 racks, MLEC (10+2)/(17+3)
  - Failure burst: Failures that happen concurrently in a small time window
  - C/C has the best failure burst tolerance, while D/D worst

When 60 disks in the same rack fail, the PDL is 0



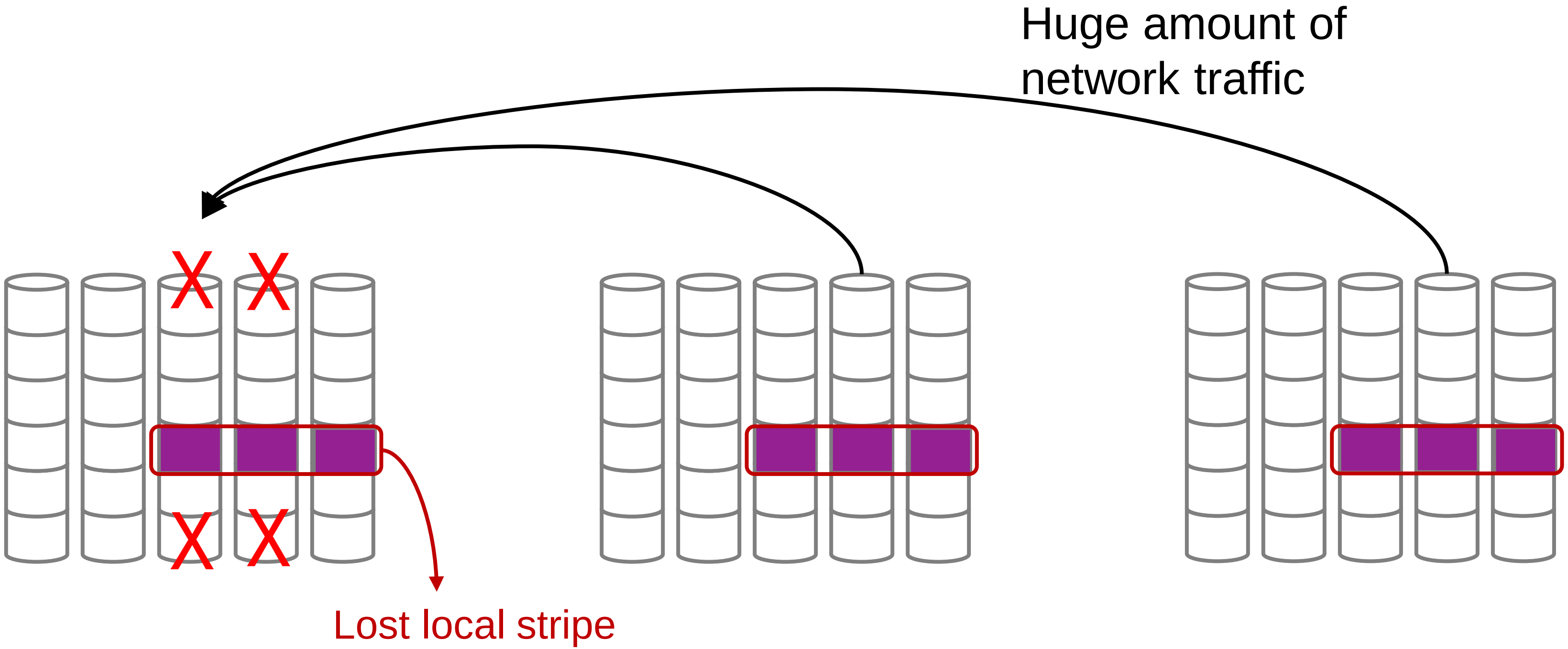
# Schemes: Repair Speed




In repairing a **single disk failure**, local declustered placement in $C/D$ and $D/D$ makes rebuilding faster

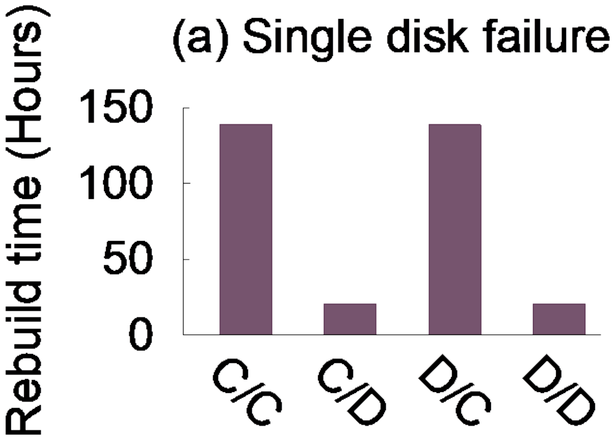
# Catastrophic Local Failure

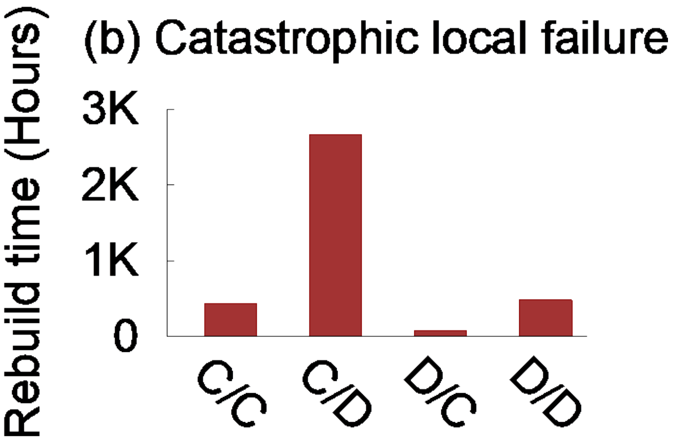
Example:
MLEC (2+1)/(2+1)




Catastrophic local pool

# Schemes: Repair Speed



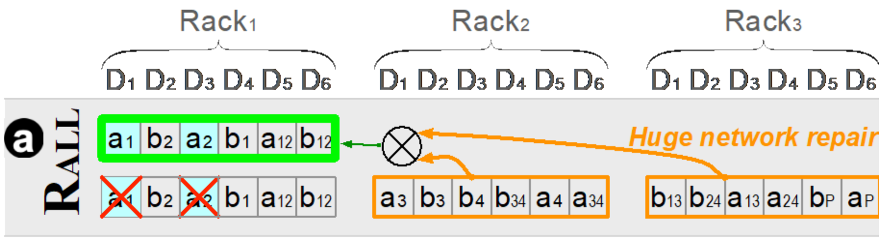




In repairing a catastrophic local failure, *D/C* is the fastest scheme.
But the time is very long for all other schemes

# Repair Methods

Example:
MLEC (2+1)/(2+1)

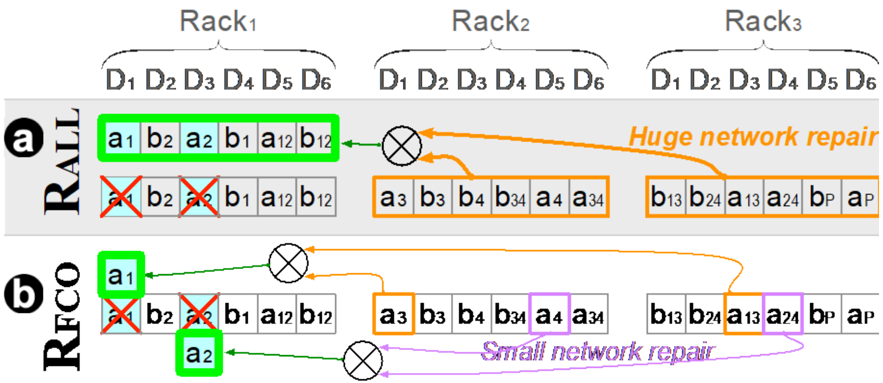
- Repair a catastrophic pool
  - Repair All (RALL)
    - Reconstruct **entire pool**
    - Easy to implement and it works
    - Used in practice
    - High network traffic



# Repair Methods

Example:
MLEC (2+1)/(2+1)

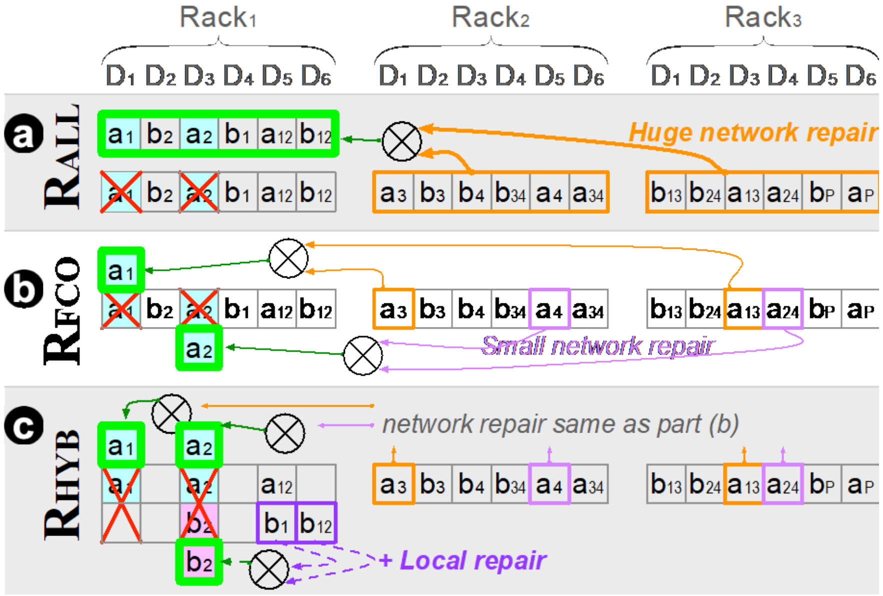
- Repair a catastrophic pool
  - Repair All (RALL)
  - Repair Failed Chunks Only (RFCO)
    - Only reconstruct **a1a2**
    - Less network traffic
    - *Requires proper API and metadata management*



# Repair Methods

- Repair a catastrophic pool
  - Repair All (RALL)
  - Repair Failed Chunks Only (RFCO)
  - Repair Hybrid (RHYB)
    - Repair stripe a from network
    - Repair stripe b locally
    - **Even less** network traffic

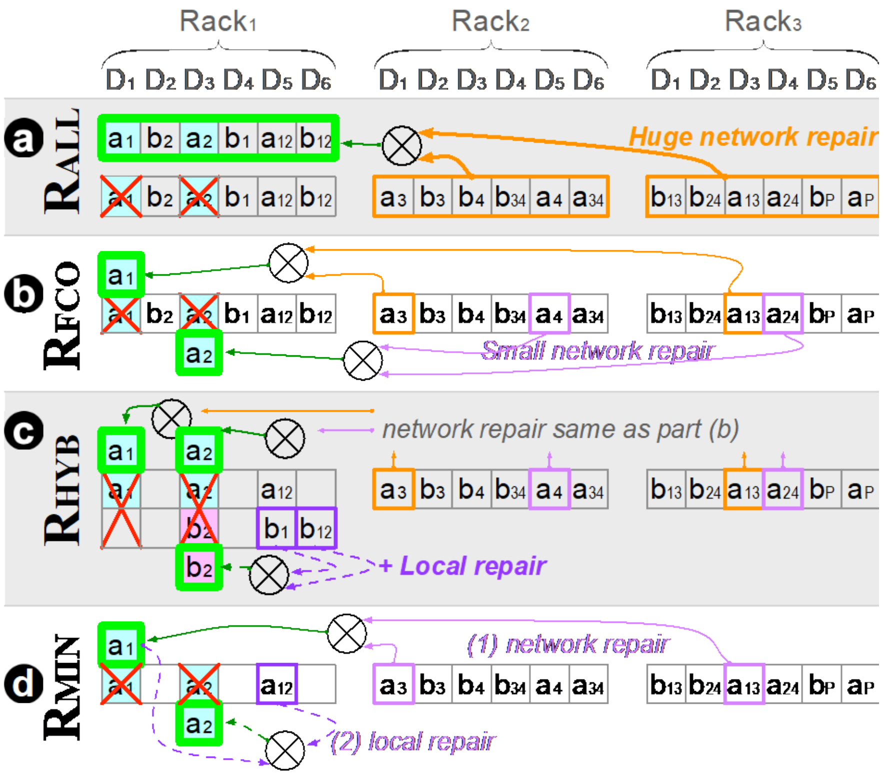
Example:
MLEC (2+1)/(2+1)



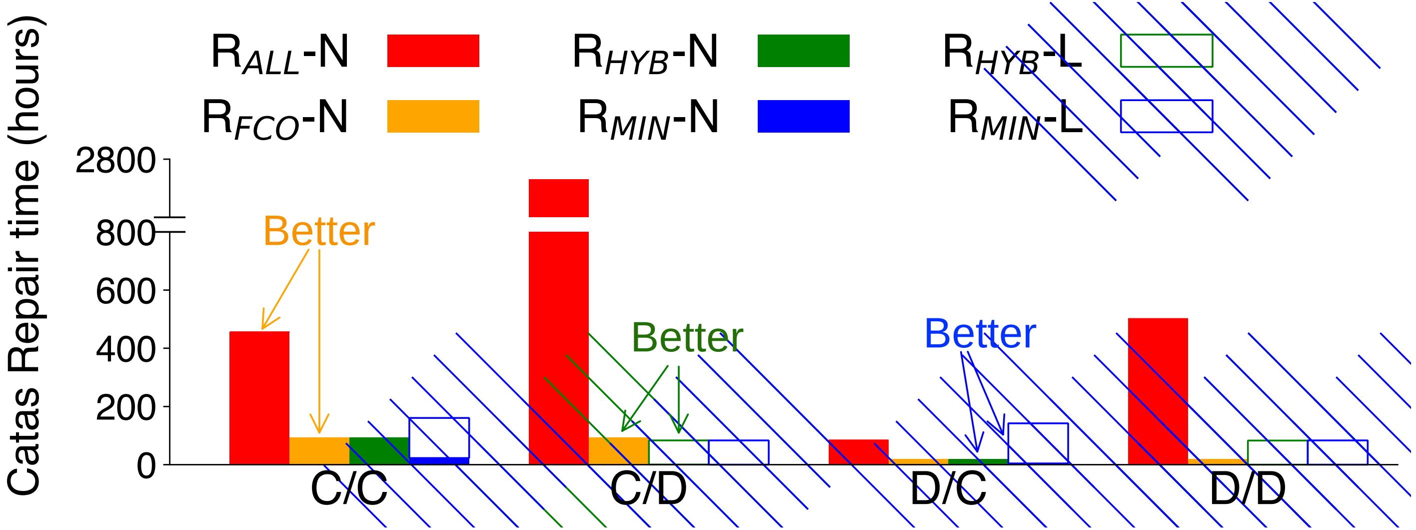
# Repair Methods

Example:
MLEC (2+1)/(2+1)

- Repair a catastrophic pool
  - Repair All (RALL)
  - Repair Failed Chunks Only (RFCO)
  - Repair Hybrid (RHYB)
  - Repair Minimum (RMIN)
    - First repair chunk a1 from network
    - Then repair a2 locally
    - Minimum network traffic



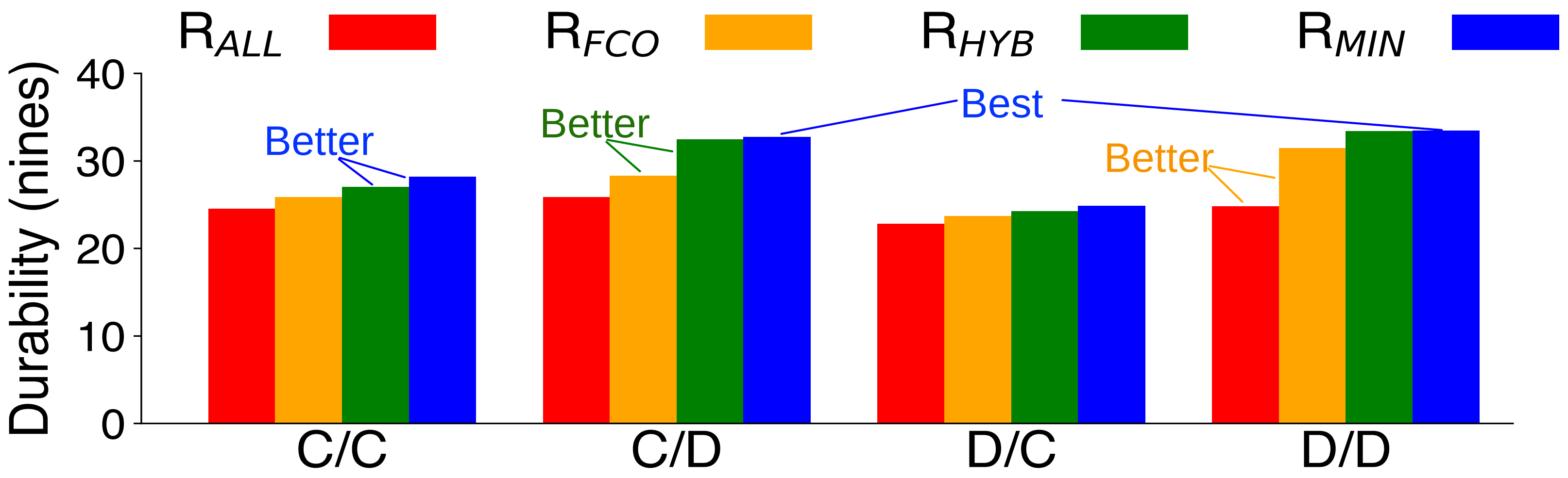
# Repair Methods: Repair Time




Our optimizations **greatly reduces** network repair time!

RMIN takes time to repair locally, but is fine as local IO is much cheaper than network traffic.

# Repair Methods: Durability

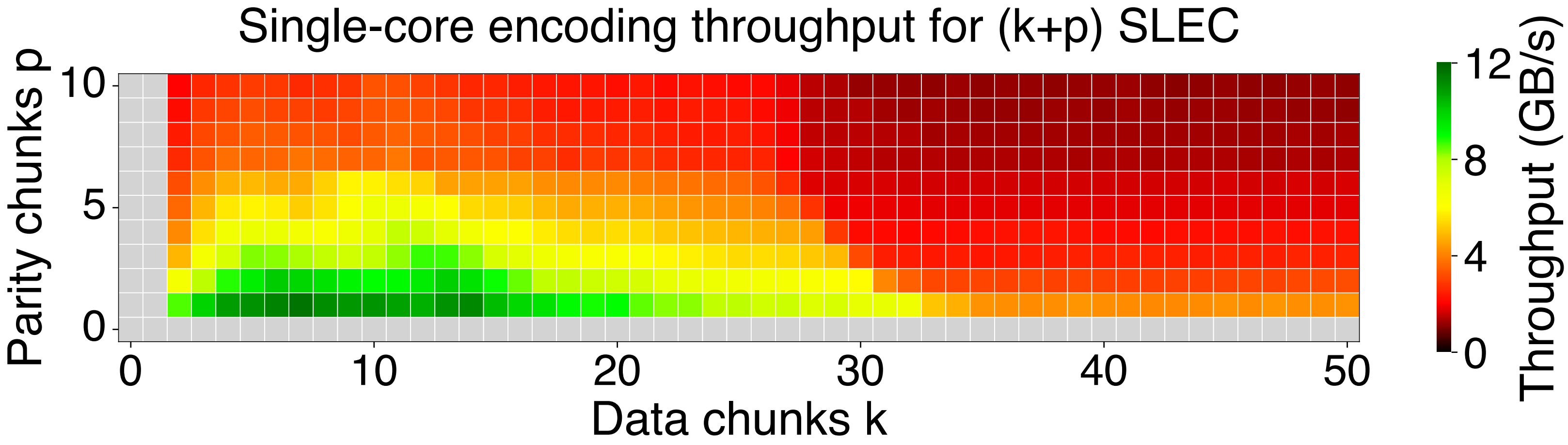



Our optimizations **increase** the durability a lot.

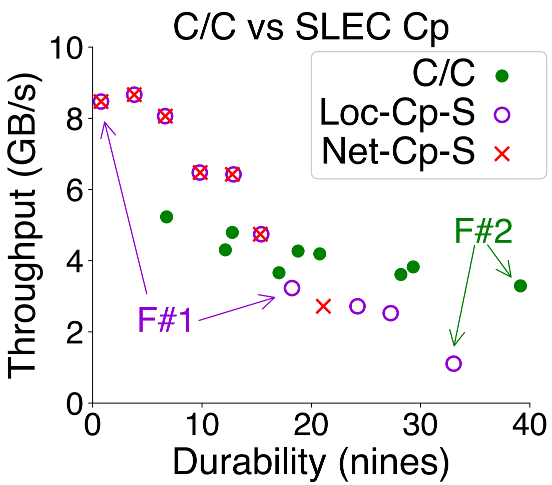
After all the optimizations, *C/D* and *D/D* provide the **best** durability.

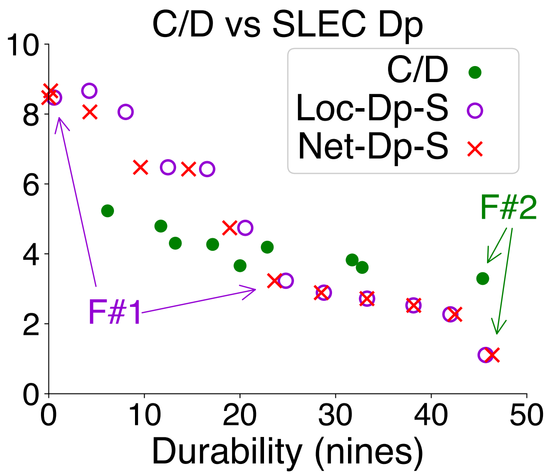
# SLEC Encoding Throughput

- Generally, EC with **larger** values of $k$ and $p$ has **lower** encoding throughput.
  - More parities → More computations
  - Wider stripe → Harder to fit into CPU cache



# MLEC vs. SLEC



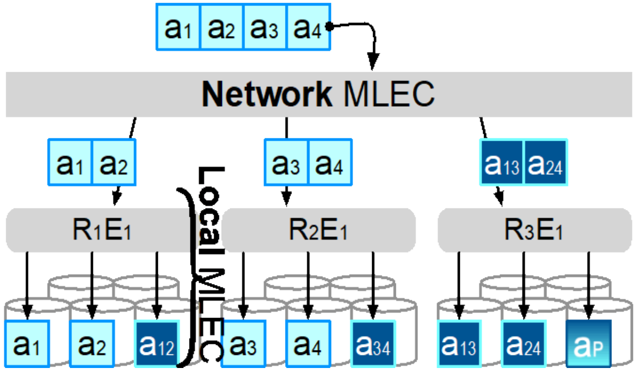



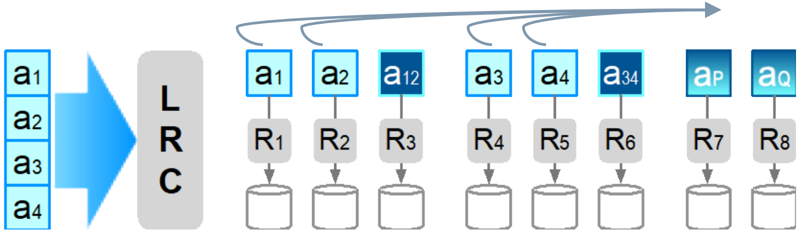

**Finding #1**: For both MLEC and SLEC, higher durability leads to lower encoding throughput.

**Finding #2**: MLEC can provide high durability while maintaining higher encoding throughput.

# MLEC vs. LRC

| MLEC | LRC |
|---|---|
| $a_{13}$ is computed from $a_1$ and $a_3$ | $a_P$ is computed from $a_1$, $a_2$, $a_3$, $a_4$ |
| A local stripe can have multiple parities | A local group has exactly one parity |
| One local stripe per rack | One chunk per rack |

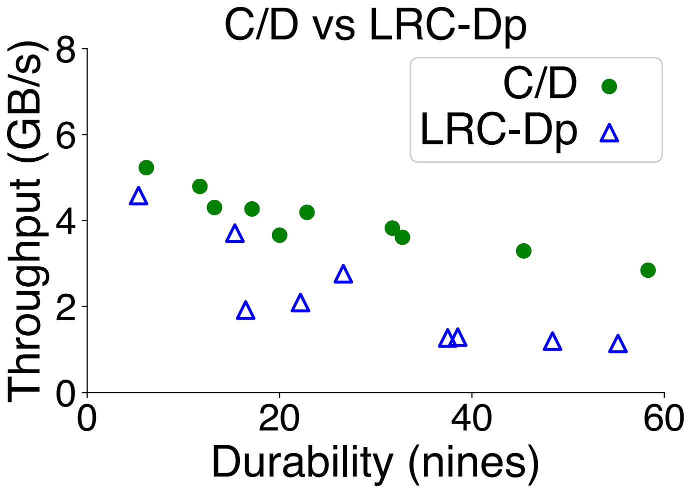


# MLEC vs. LRC

Both MLEC and LRC have their own benefits

In some scenarios, MLEC can provide a better tradeoff.
- e.g. when the network bandwidth is very limited



# Conclusion

- *Comprehensive design considerations and analysis of MLEC at scale*

| | |
|---|---|
| **Chunk placement schemes** | C/C, C/D, D/C, D/D |
| **Failure modes** | Single disk failure, Catastrophic local failure |
| **Repair methods** | RALL, RFCO, RHYB, RMIN |
| **Analysis** | Performance, durability |
| **Comparison** | Vs. SLEC, LRC, … |

# Thank you!
# Questions?