Erasure Coding & Replication for Edge Devices

4th May 2019
Presenter : Sheshadri K R
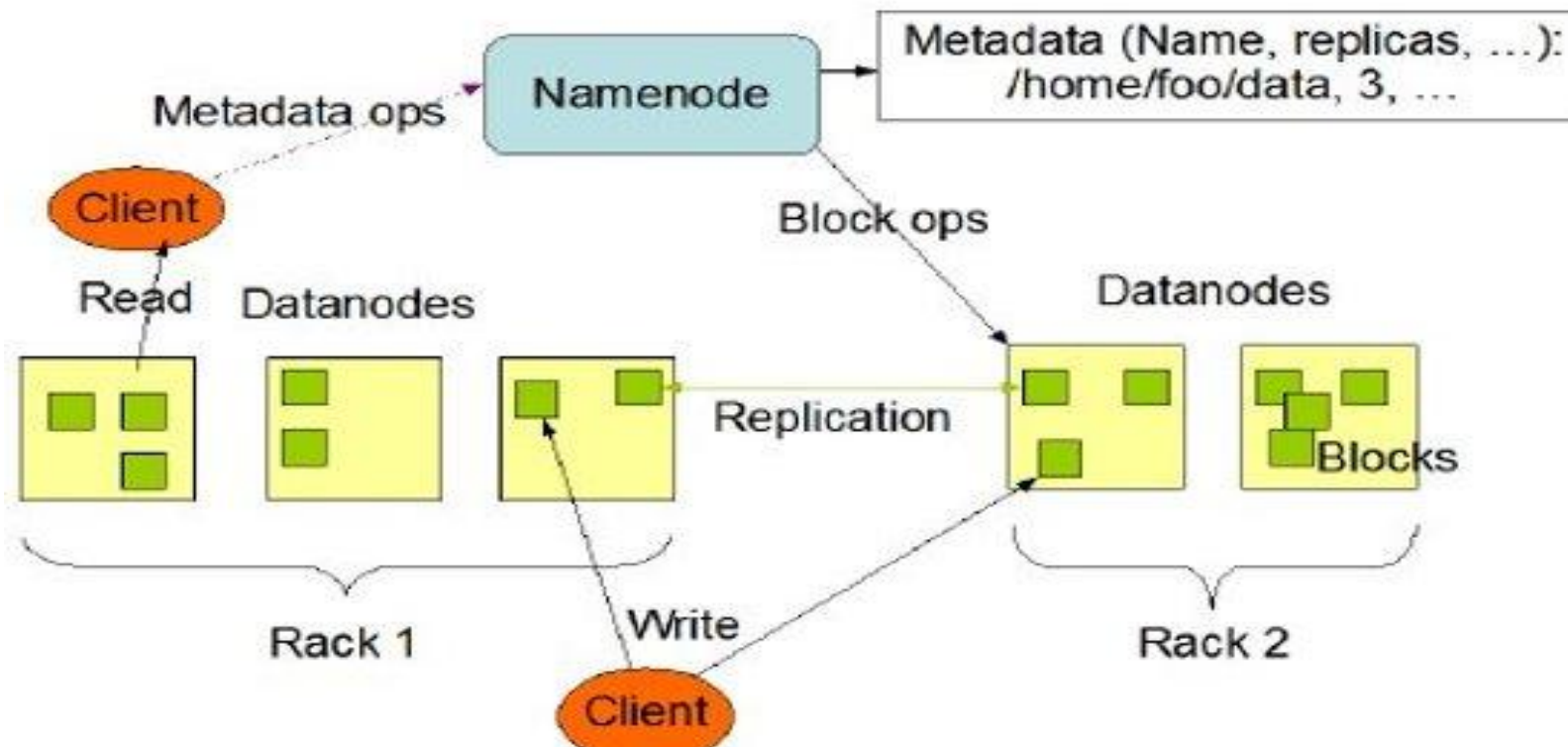## Ph.D student, DREAM lab
Course : DS256

CDS
Department of Computational and Data Sciences

Intro & Motivation

- Replication is a very popular technique to achieve reliable storage!
- It is used a lot in a lot of companies such a Facebook, Yahoo! Etc.,
- Data is divided into chunks and is replicated to ensure it is stored reliably
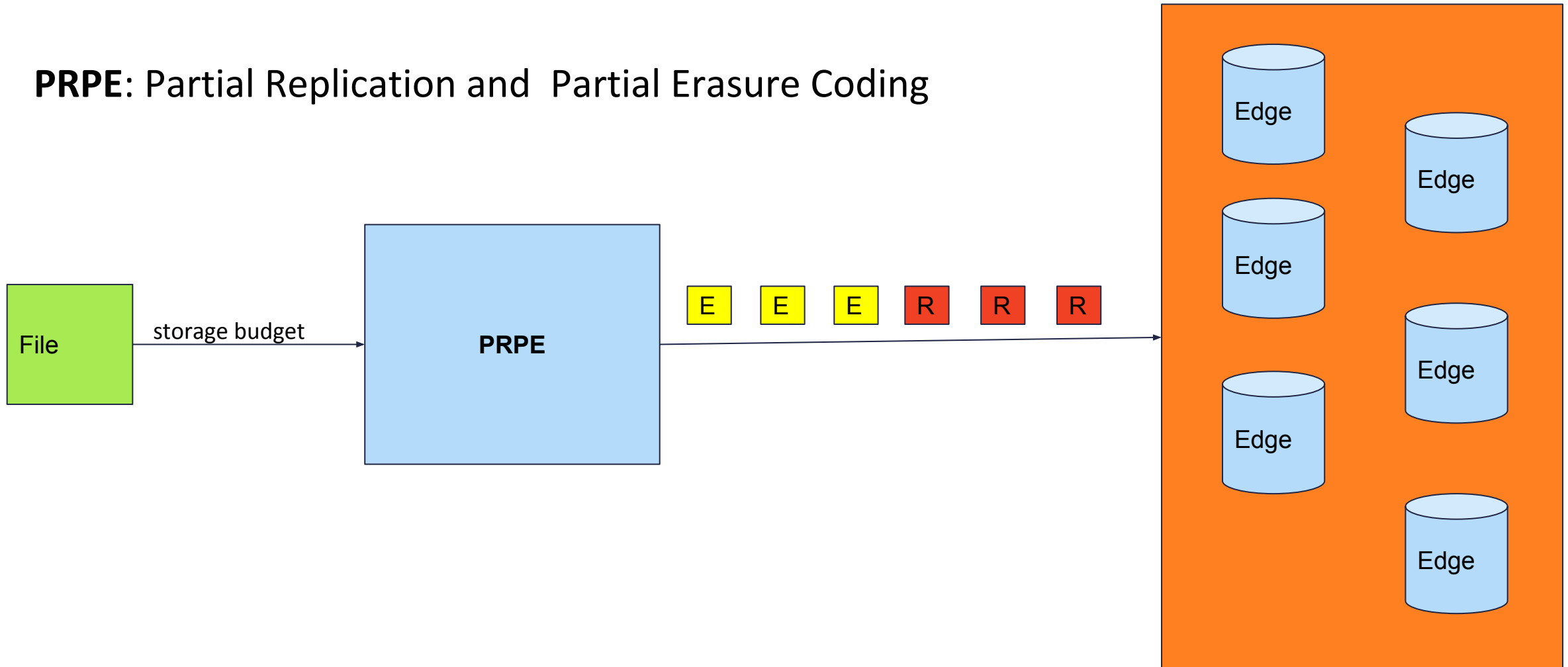
Erasure Coding

- Replication has a 33% storage efficiency. **(upto 2 failures)**
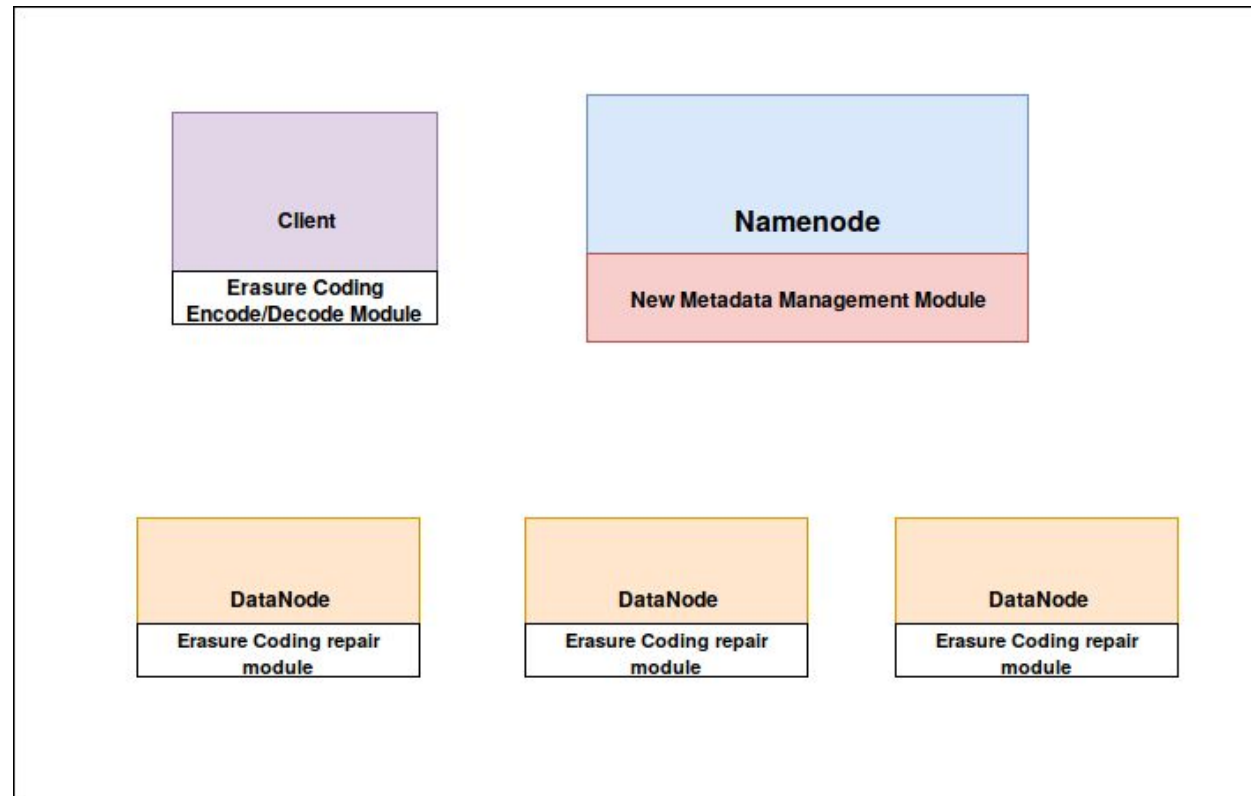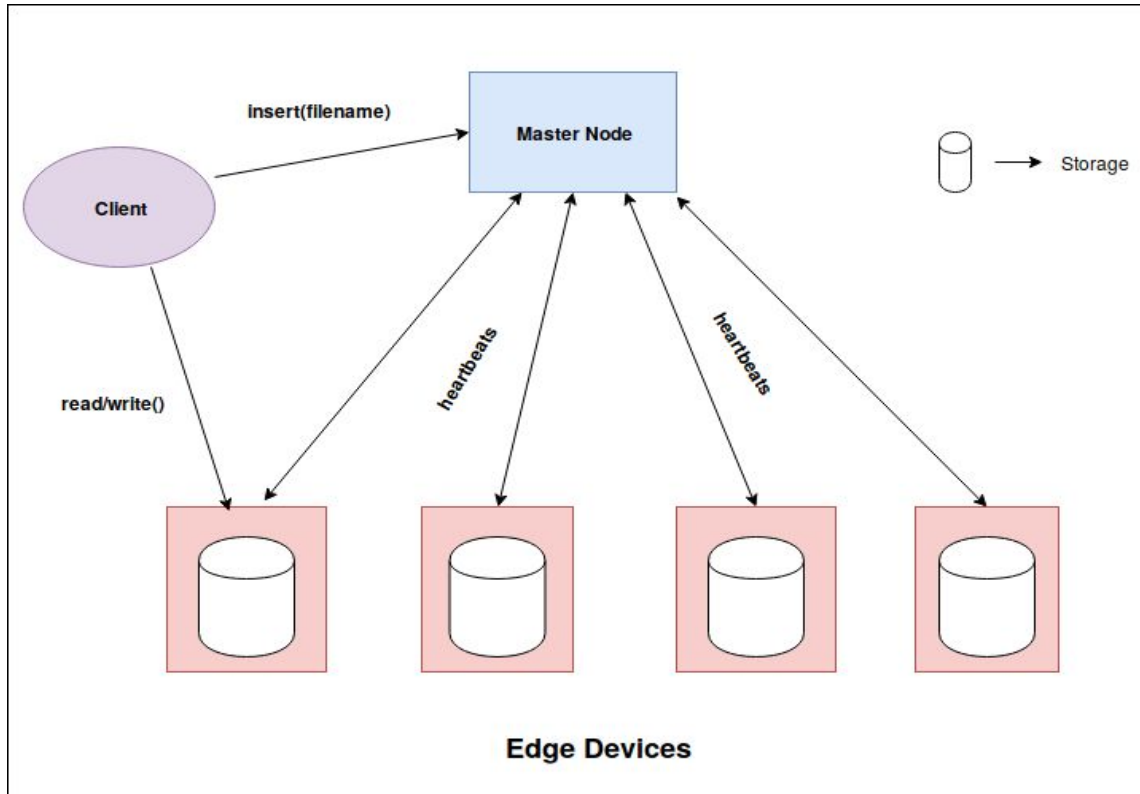- Erasure coding has 50% storage efficiency. **(upto 2 failures)** RS(6,4)
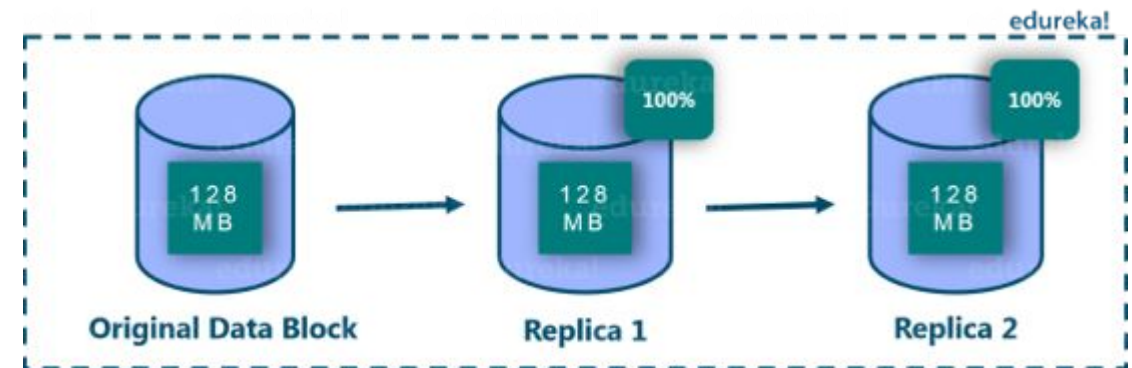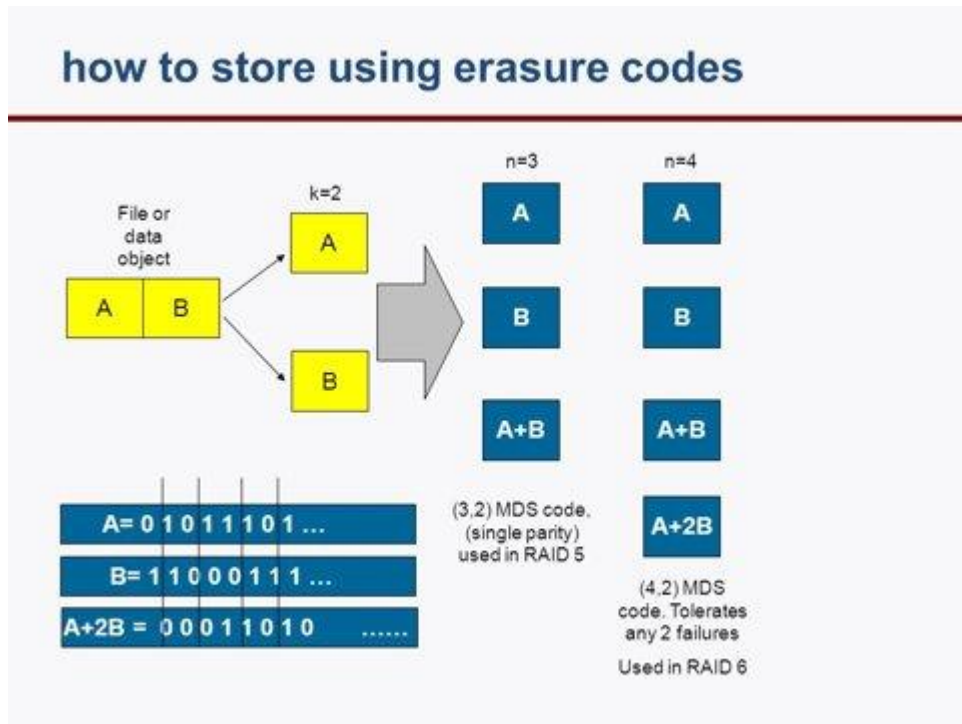
Proposed Technique

- **PRPE**: Partial Replication and Partial Erasure Coding

Design of the system

PRPE Equation

$$Nr = (K/N) * SB * totalBlocks \qquad (1)$$

$$Ne = totalBlocks - Nr \qquad (2)$$

Where,

- $Nr \rightarrow$ number of blocks that can be replicated
- $Ne \rightarrow$ number of blocks that needs to be replicated
- $SB \rightarrow$ storage budget given by the user
- $totalblocks \rightarrow$ total number of blocks for a given file.
- $K \rightarrow$ data blocks in the given file.

Experimental Setup

- Google Cloud VM
- 8-cores Intel
- 8GB Main Memory
- Ubuntu 16.04 LTS

- Ran a total of 11 containers
- 1 Fog Container : 2 cores , 8GB of RAM
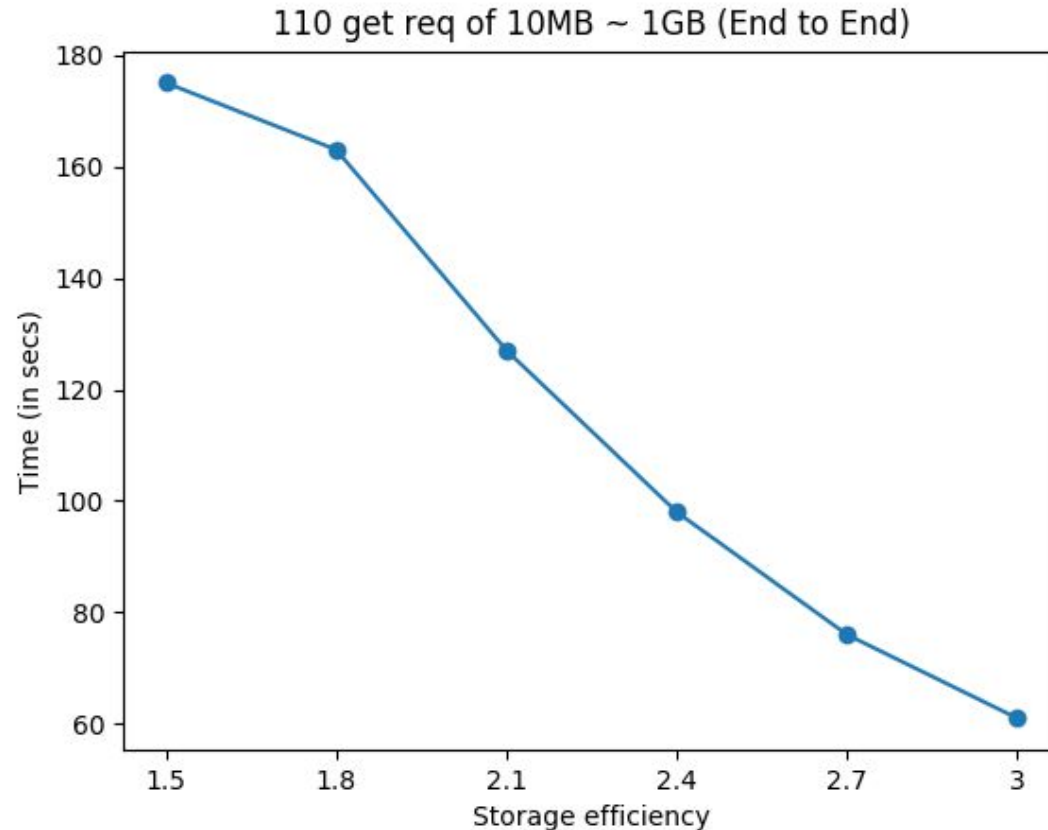- 10 Edge Container : 0.5 Core, 2GB of RAM

Put Experiments

- File size 10.8 MB

- Number of puts : 100

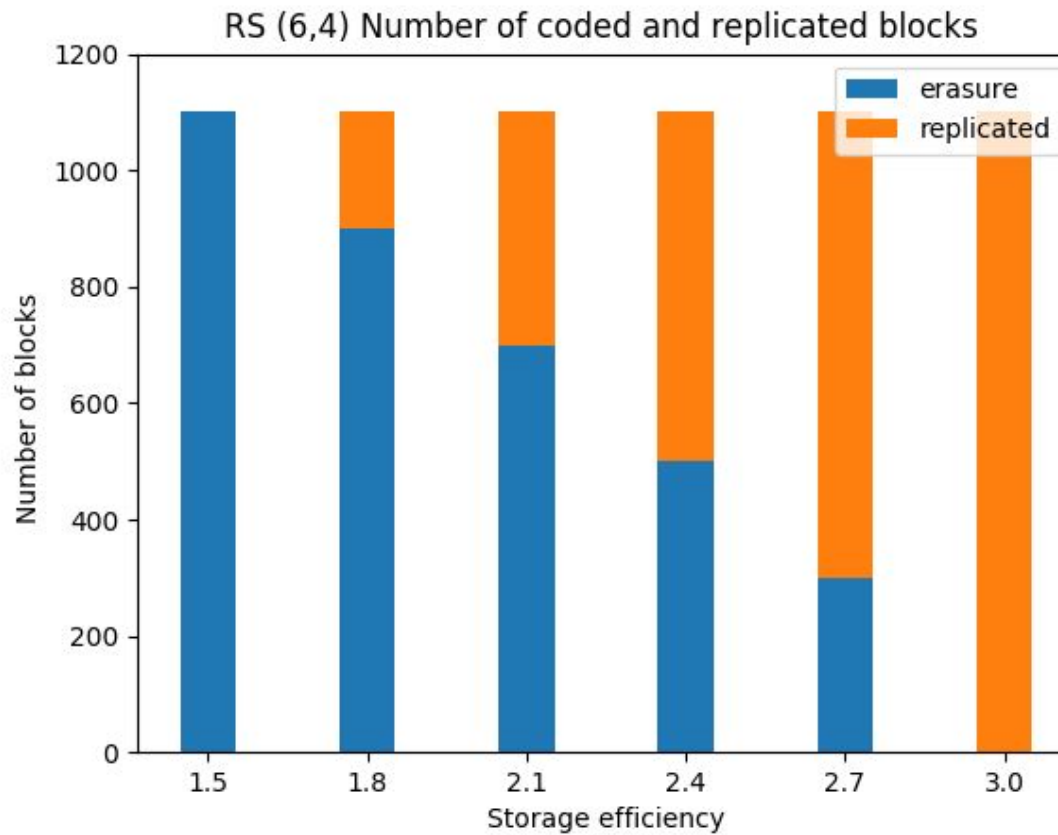- Storage Efficiency:
    - 1.5x to 3.0x

- RS(6,4)

RS(6,4) 110 put req of 10MB ~ 1GB (End to End)
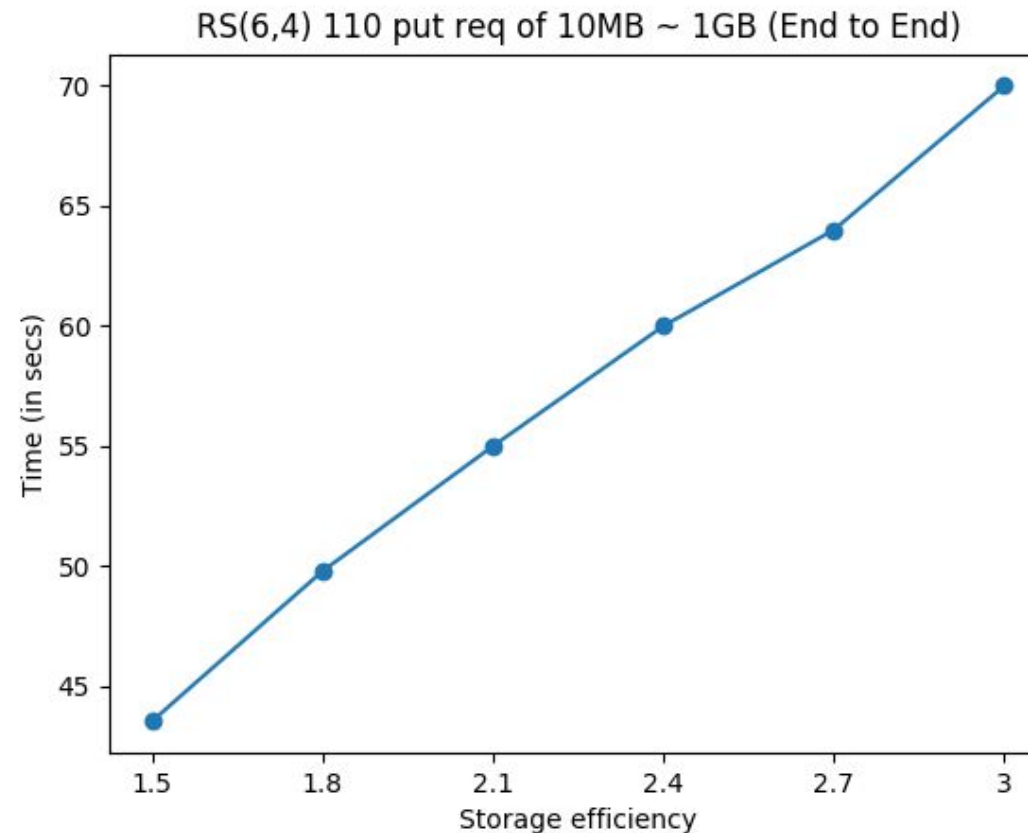
Get Experiments

- File size 10.8 MB
- Number of gets : 100
- Storage Efficiency:
  - 1.5x to 3.0x

- RS(6,4)

110 get req of 10MB ~ 1GB (End to End)

Block split-up put and get

RS (6,4) Number of coded and replicated blocks
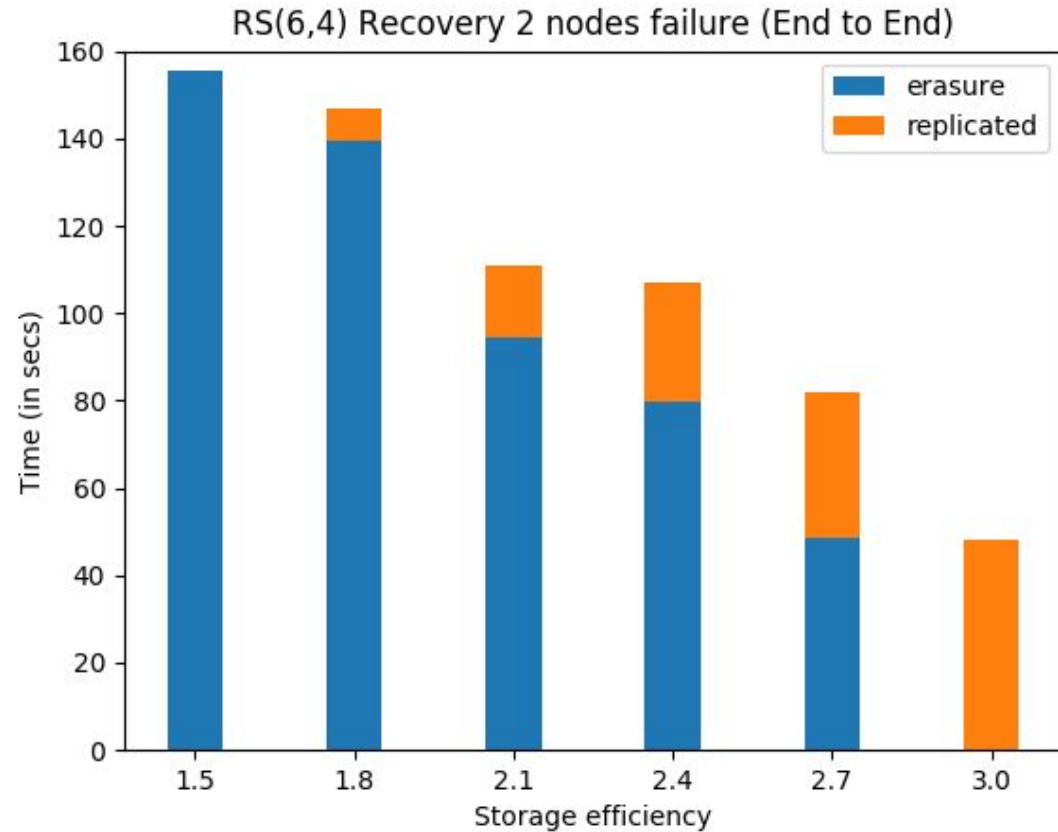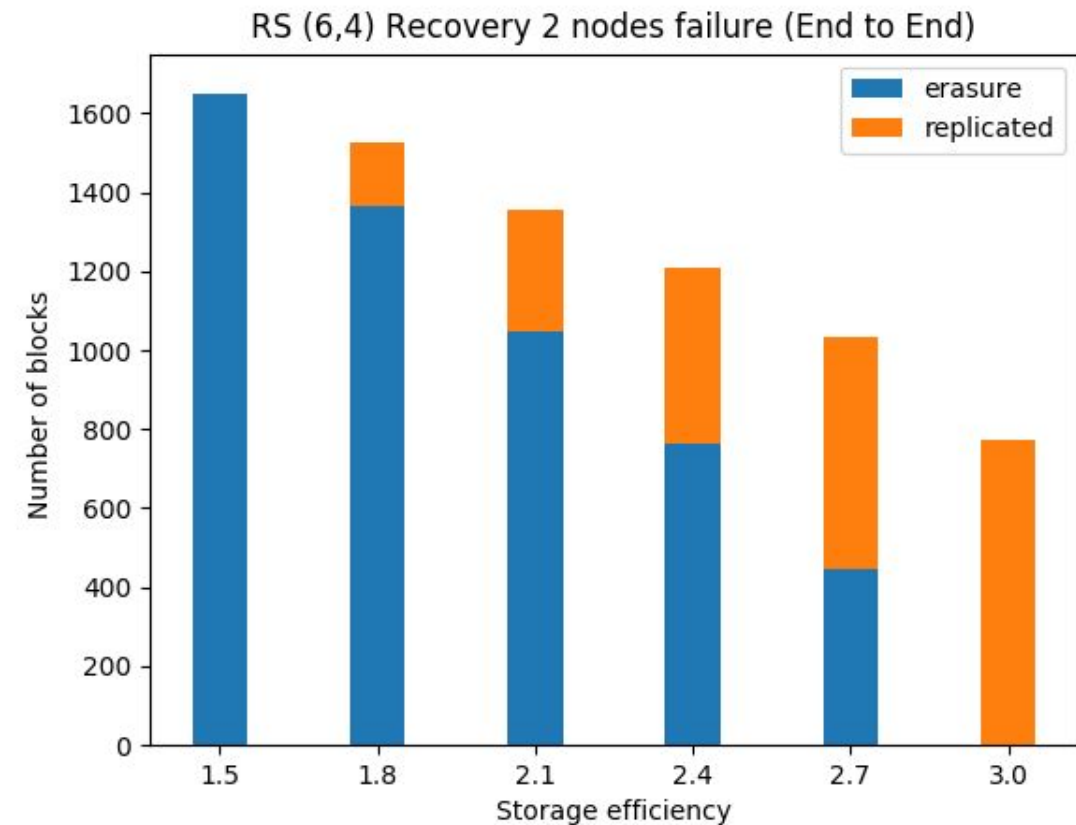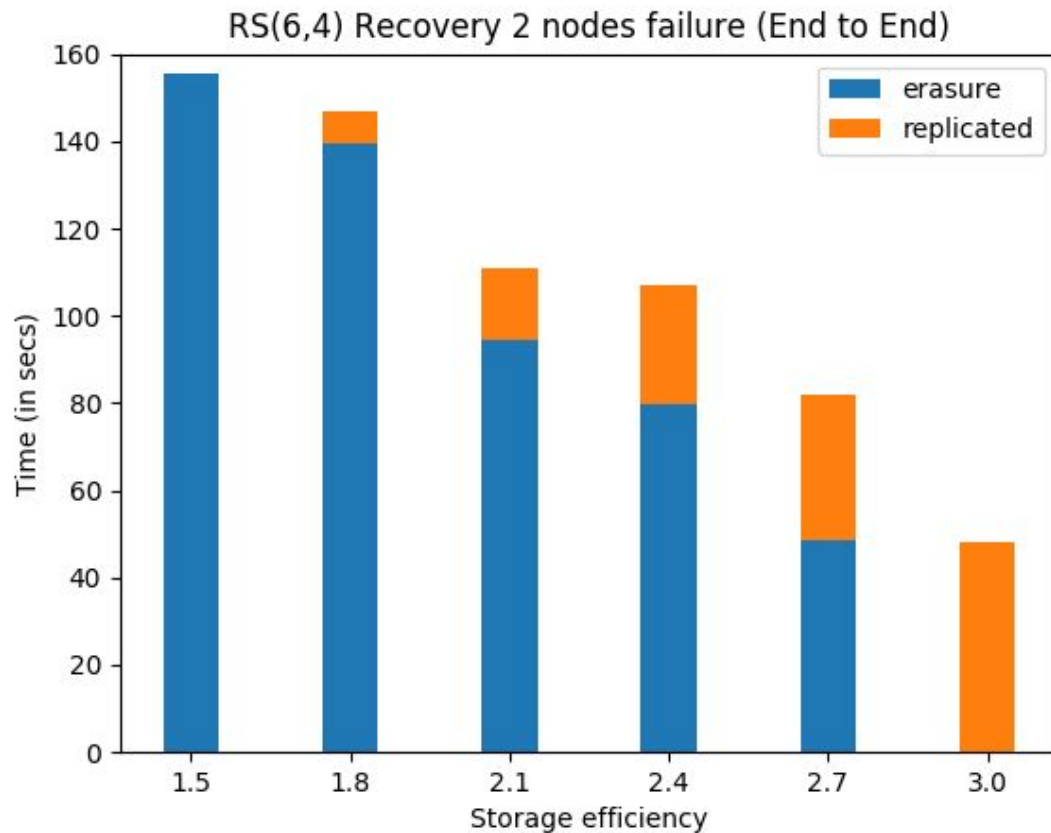
Recovery

- 2 nodes were killed randomly
- Every block in the dead node was recovered.
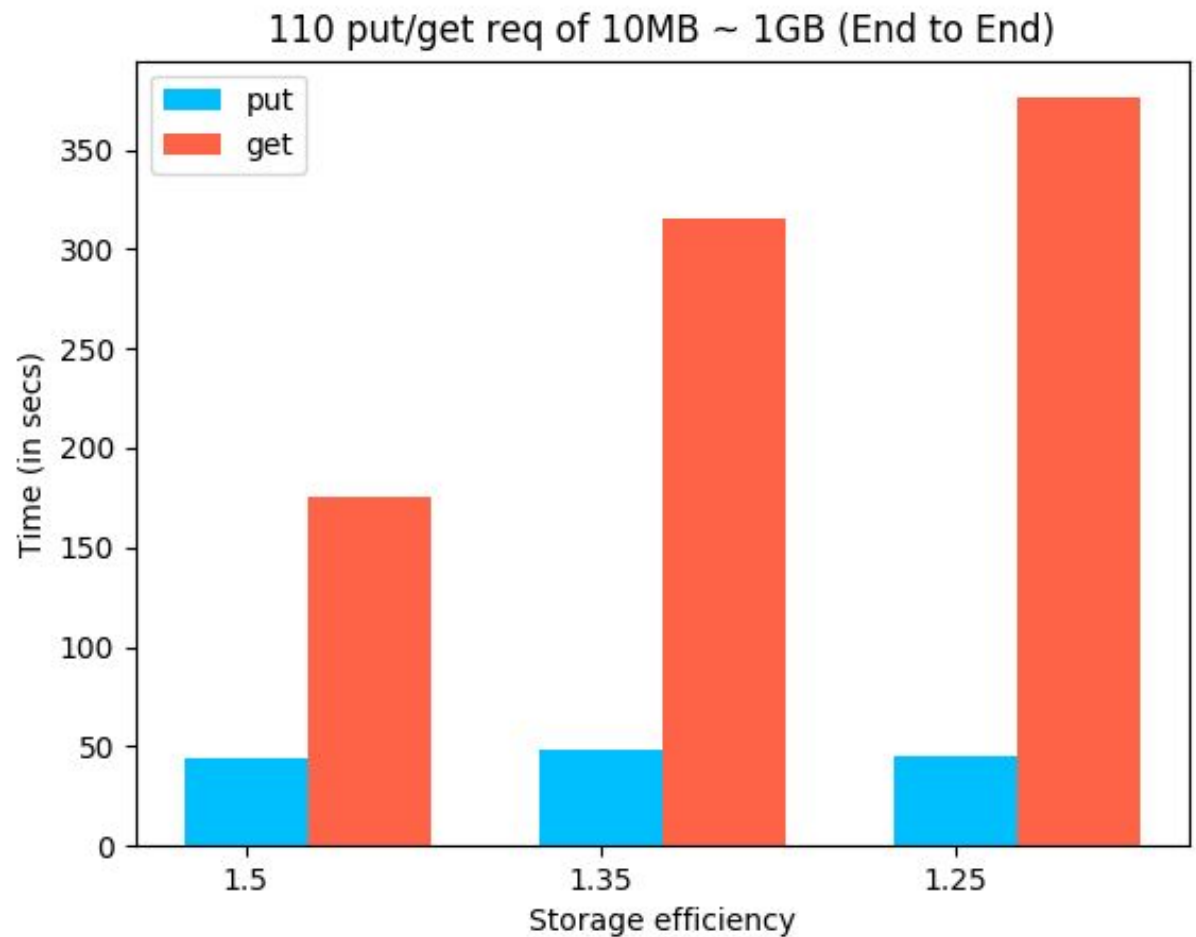- A recovery thread checks failed nodes every 180 seconds

Recovery time and block split-up

Erasure Coding Families

- Did 100 put() and get() for different Erasure coding families. File size 10.8MB

- RS(6,4) RS(8,6) RS(10,8)

- All are fault tolerant upto 2 node failures

110 put/get req of 10MB ~ 1GB (End to End)

13

Conclusions

- put() is faster is EC

- get() is faster in Replication

- Recovery is faster for replication.
- Tighter storage efficiency will cause slower read and writes.

Future Work

- Improve put()/get() by parallelising EC

- Improve single node recovery.

- Compare it against Product Codes and LRC (Locally Recoverable Codes)

Thank you