

# Have a Seat on the ErasureBench: Easy Evaluation of Erasure Coding Libraries for Distributed Storage Systems

2016 Workshop on Planetary-Scale Distributed Systems

**Sébastien Vaucher**, Hugues Mercier, Valerio Schiavoni

Institute of Computer Science  
Université de Neuchâtel, Switzerland

sebastien.vaucher@unine.ch

Budapest, Hungary, 26 September 2016



# Outline

- Motivation
- Recap about erasure coding
- ERASUREBENCH
- Evaluation
- Conclusion

# Motivation

More and more data needs to be stored reliably on online servers.  
Reliability can be provided through:

- Replication
- Erasure coding

# Motivation

More and more data needs to be stored reliably on online servers.  
Reliability can be provided through:

- Replication
- Erasure coding

# Motivation

- The characteristics of erasure coding algorithms are difficult to evaluate (encoding, decoding, complexity, latency, ...)
- Evaluation is often done theoretically or by simulation

# Erasure coding

Goal: add redundancy to cope with data loss/corruption

Example using a  $(5, 2)$  Reed-Solomon code:

File contents

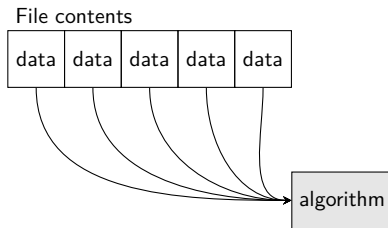
data	data	data	data	data
------	------	------	------	------

algorithm

# Erasure coding

Goal: add redundancy to cope with data loss/corruption

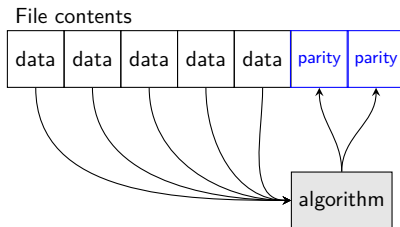
Example using a  $(5, 2)$  Reed-Solomon code:



# Erasure coding

Goal: add redundancy to cope with data loss/corruption

Example using a  $(5, 2)$  Reed-Solomon code:





# Erasure coding

Goal: add redundancy to cope with data loss/corruption

Example using a  $(5, 2)$  Reed-Solomon code:

File contents

data	data	data	data	data	parity	parity
------	------	------	------	------	--------	--------

algorithm

# Erasure coding

Goal: add redundancy to cope with data loss/corruption

Example using a  $(5, 2)$  Reed-Solomon code:

File contents

data	data	data	parity	parity
------	------	------	--------	--------

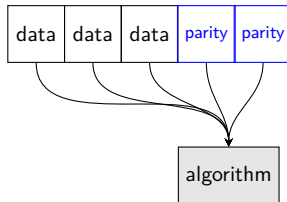
algorithm

# Erasure coding

Goal: add redundancy to cope with data loss/corruption

Example using a  $(5, 2)$  Reed-Solomon code:

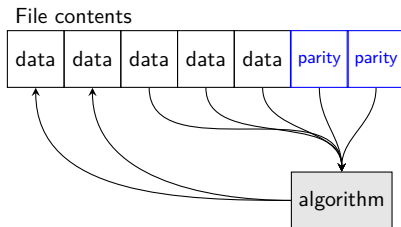
File contents



# Erasure coding

Goal: add redundancy to cope with data loss/corruption

Example using a  $(5, 2)$  Reed-Solomon code:



## ERASUREBENCH key features

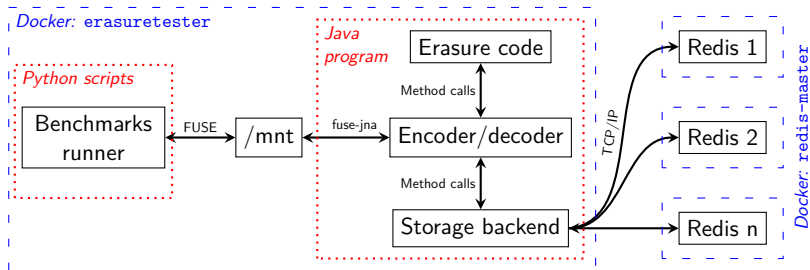
- Compatible with existing benchmark programs
- Automated benchmarks execution
- Containerized storage nodes ( $> 1$  per physical node)
- Can replay fault traces

# Evaluation example

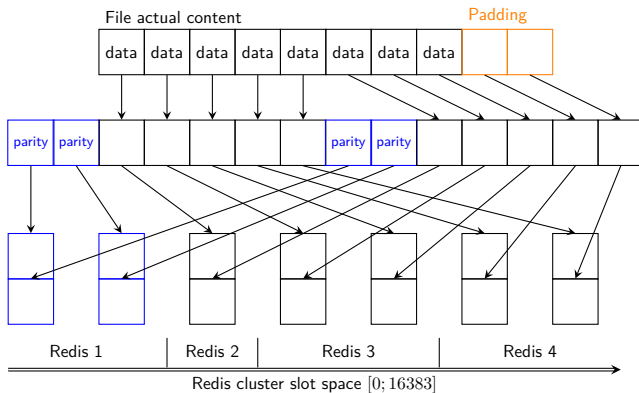
How to evaluate a new erasure coding algorithm

1. Program the algorithm as a Java class
2. Write benchmarks as Python functions
  - Debian-compatible programs can be launched as sub-processes
3. Configure the evaluation
  - e.g. algorithm parameters, fault trace, ...
4. Easily deploy the solution to a Docker cluster
5. Collect results

# ERASUREBENCH technical components



# Blocks distribution





# ERASUREBENCH metadata management

- Each block is identified by a 32-bit key. Using it, we derive:
  1. Key of the blocks aggregation stored in Redis
  2. Offset within that aggregation
- The list of all block keys is kept in memory

# ERASUREBENCH automated deployment and scaling

As part of ERASUREBENCH, we provide scripts that automate the deployment of the solution to a Docker Swarm cluster, up to the collection of results

# Evaluation

We evaluated algorithms from the following paper:

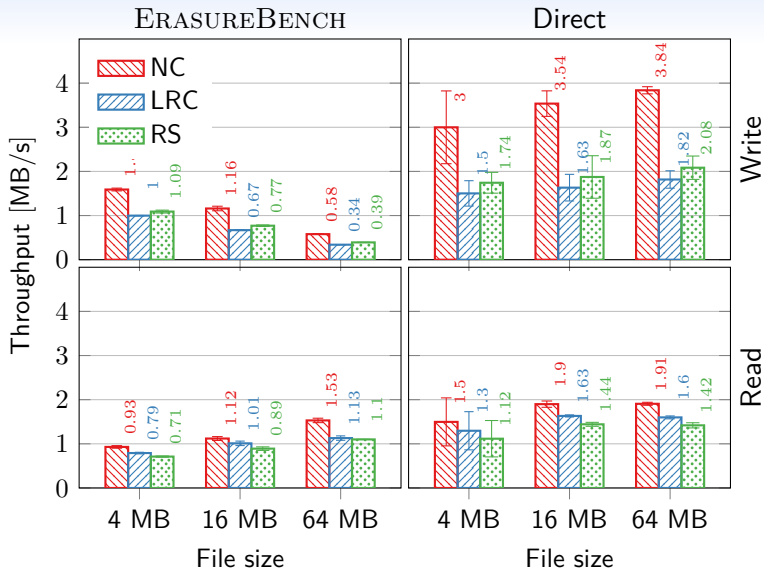
M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, “**XORing elephants: Novel erasure codes for big data**,” in Proceedings of the VLDB Endowment, vol. 6, 2013, pp. 325–336.

NC No erasure coding

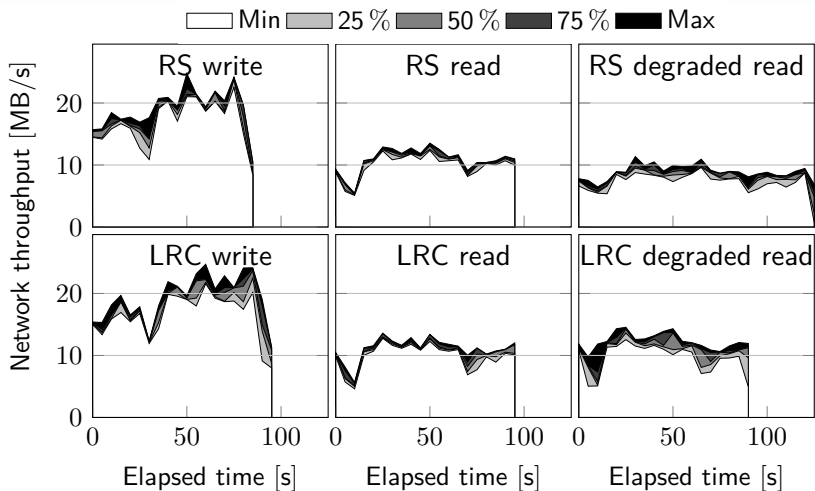
RS (10, 4) Reed-Solomon code

LRC (10, 6, 5) Locally Repairable Code

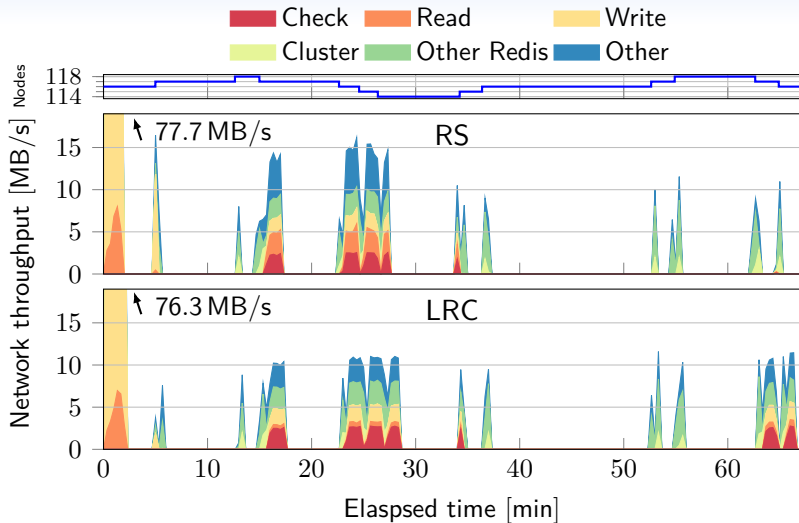
# Throughput



# Traffic



# Trace



# Limitations

- Algorithms need to be written in Java
- High memory consumption
- Strong dependency on Docker and Redis

# Conclusion

Using ERASUREBENCH, evaluating an erasure coding algorithm under real conditions is easier and cheaper

Available open-source at  
<https://github.com/safecloud-project/erasurebench>