Optimising Erasure Coding: The testing that makes us confident

Jonathan Bailey

How did we test optimised erasure coding?

Unit Tests

Teuthology Testing

Performance Testing

Ceph IO Sequence Exerciser

How did we test optimised erasure coding?

Unit Tests

Teuthology Testing

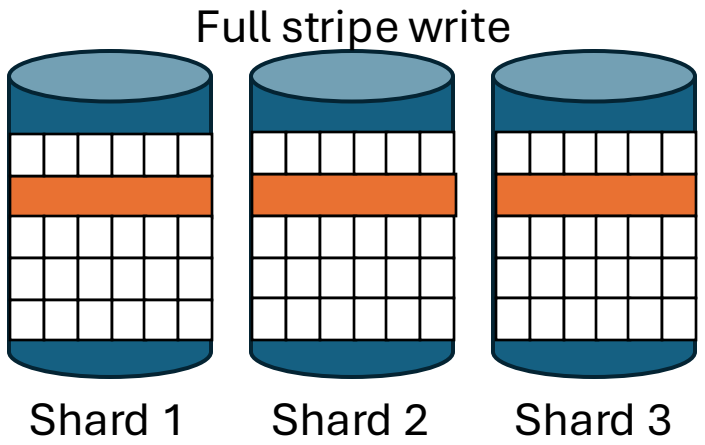
Performance Testing

Ceph IO Sequence Exerciser

Why create a new tool?

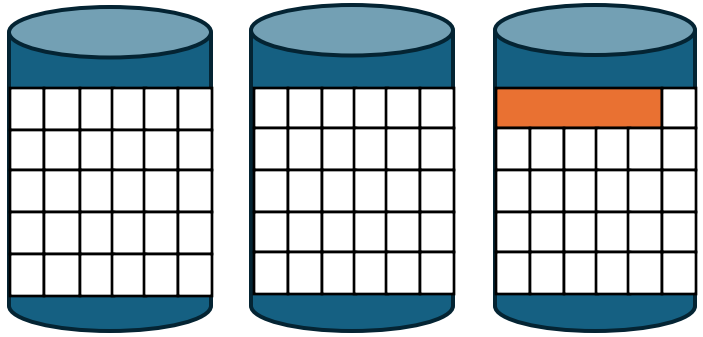
Why create a new tool?

- Explicitly test and stress Erasure Coding Boundary Cases
- Adapt tests for specific Erasure Coding properties
  - Plugin and technique
  - k+m
  - Chunk size
  - Shards/missing shards

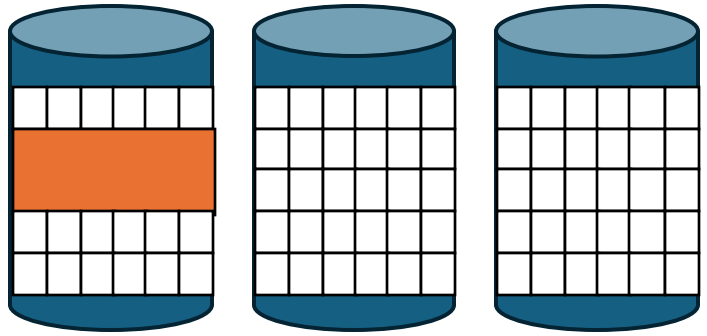


Difficult IOs for Erasure Coding

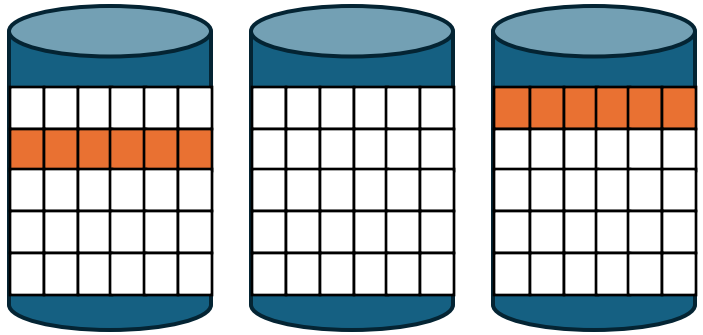
Non-whole shard lengths



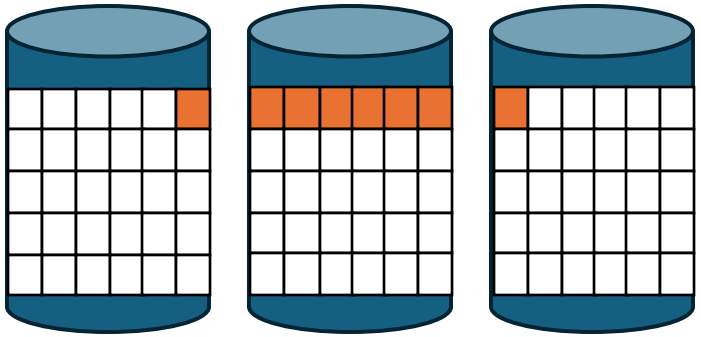
Writes that do not include all shards



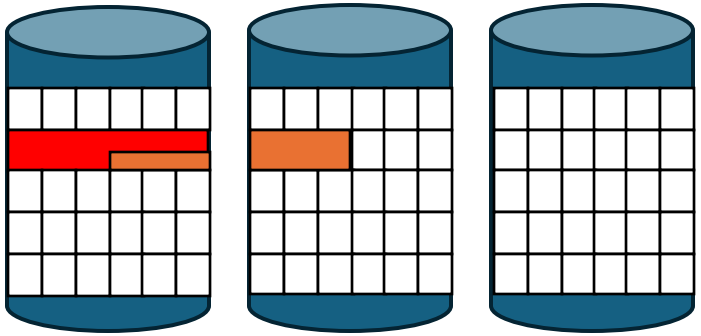
Crossing stripe boundaries



Crossing shard boundaries



Multiple overlapping writes



Ceph IO Sequence Exerciser Aims

- Aims:
  - Reproducability of tests while also varied
  - Runs IO Sequences specifically designed to stress erasure coding

An addition to the suite of applications for Ceph test, allowing targeted testing of erasure coding specifically.

How does the tool work?

How does this tool work?

- Data Generation
- Error injects
- IO Sequences

Data Generation

- Random data from seed
- Generated in “blocks”
  - User specified OR
  - From a pre-determined list of interesting values
- Data chunks have headers
  - Application ID
  - Seed
  - Time
- Recalculates data from seed on each read

Header: <Application ID><Seed><Time>

Random Data:

```
6AEB931B58CFFA67B8A289DDF5F76
210E51FCE66D2D23DE31CAF8A1101
E78A2A1F4929122022065146 2DAE9
9D7FA0D74B47CE5536ED239DA29ED
1489F11A97100620256BF2A13FC87
017E327AB2CA13C960FEC72998C1C
089250C5462A9593A9A64848A127D
CEBE111A2B874576B7B3245981D8A
D2F230A9E4D08641045981D8A02F2
```

Error Injects

- Simulator errors
  - Read Errors
    - EIO returned during read from shard
    - OSD missing during read
  - Write Errors
    - Write fails during write so write is rolled back and failed
    - OSD assert during write

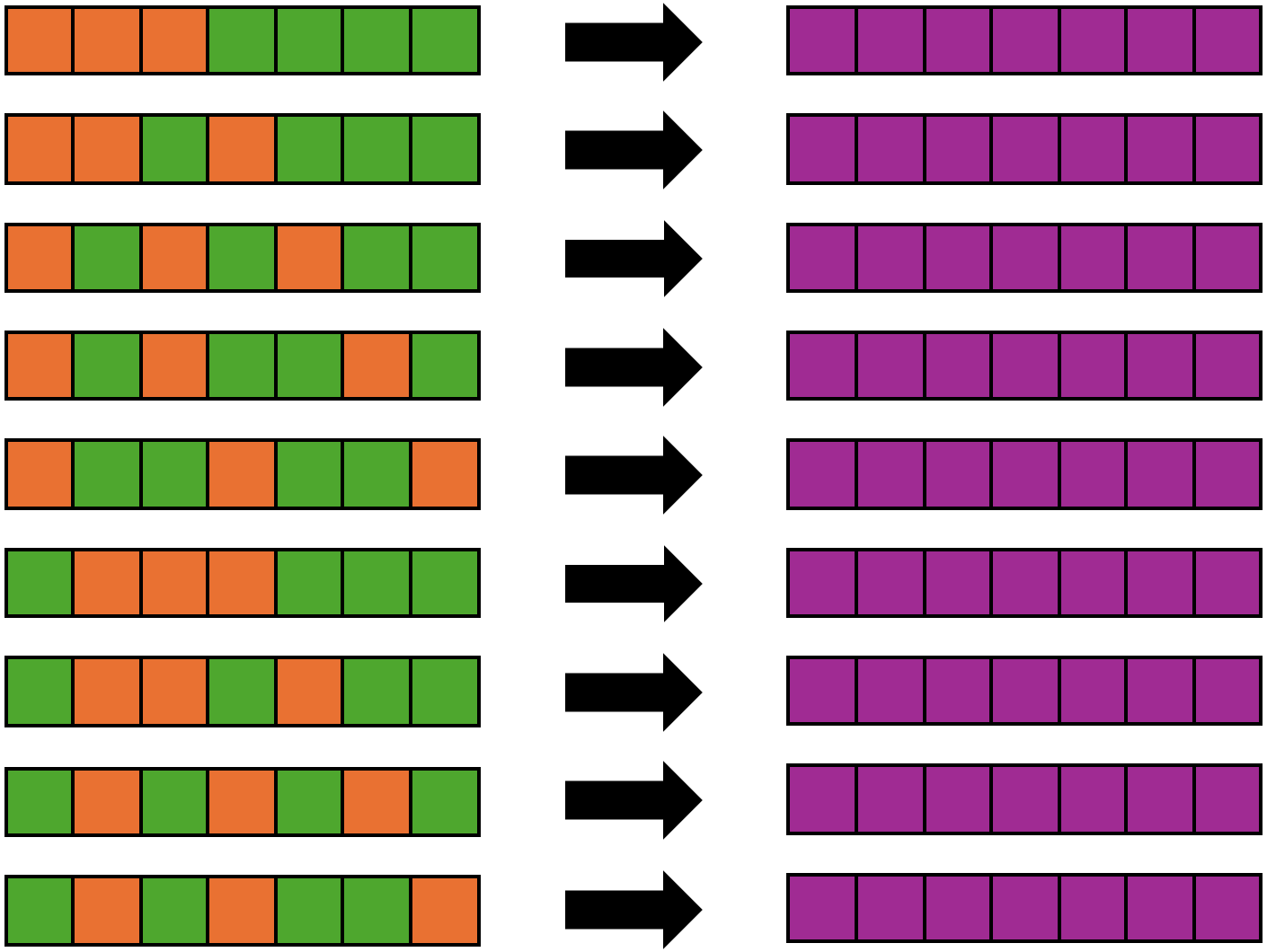
Sequence

- Sequence Generators create different patterns of IOs
  - Operation Type (Create/Remove/Read/Write/Append/Truncate)
  - Offset/Length
  - Number of operations per IO
  - Order of IOs to send
  - blocksize IOs, which often is a divisor of the EC chunksize

Sequences: Sequence 8

Select operations from 7th iteration of sequence 8.

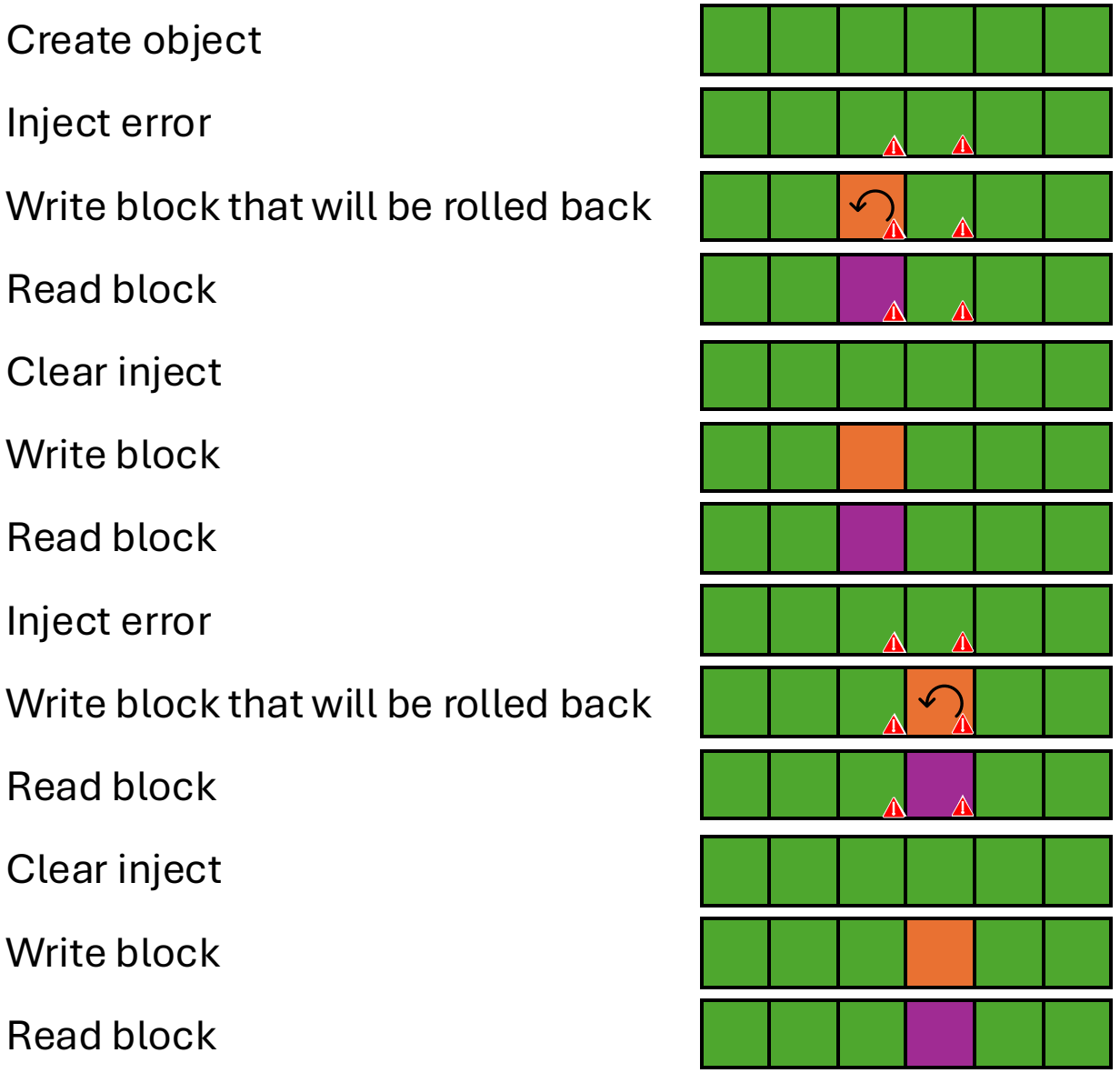
1) Write 3 ranges -> Read all
2) Write 3 ranges -> Read all
3) Write 3 ranges -> Read all
4) Write 3 ranges -> Read all
5) Write 3 ranges -> Read all
6) Write 3 ranges -> Read all
7) Write 3 ranges -> Read all
8) Write 3 ranges -> Read all
9) Write 3 ranges -> Read all



Object
Read IO
Write IO

Sequence desc: Permutations of offset 3-region single read/write I/O operations.

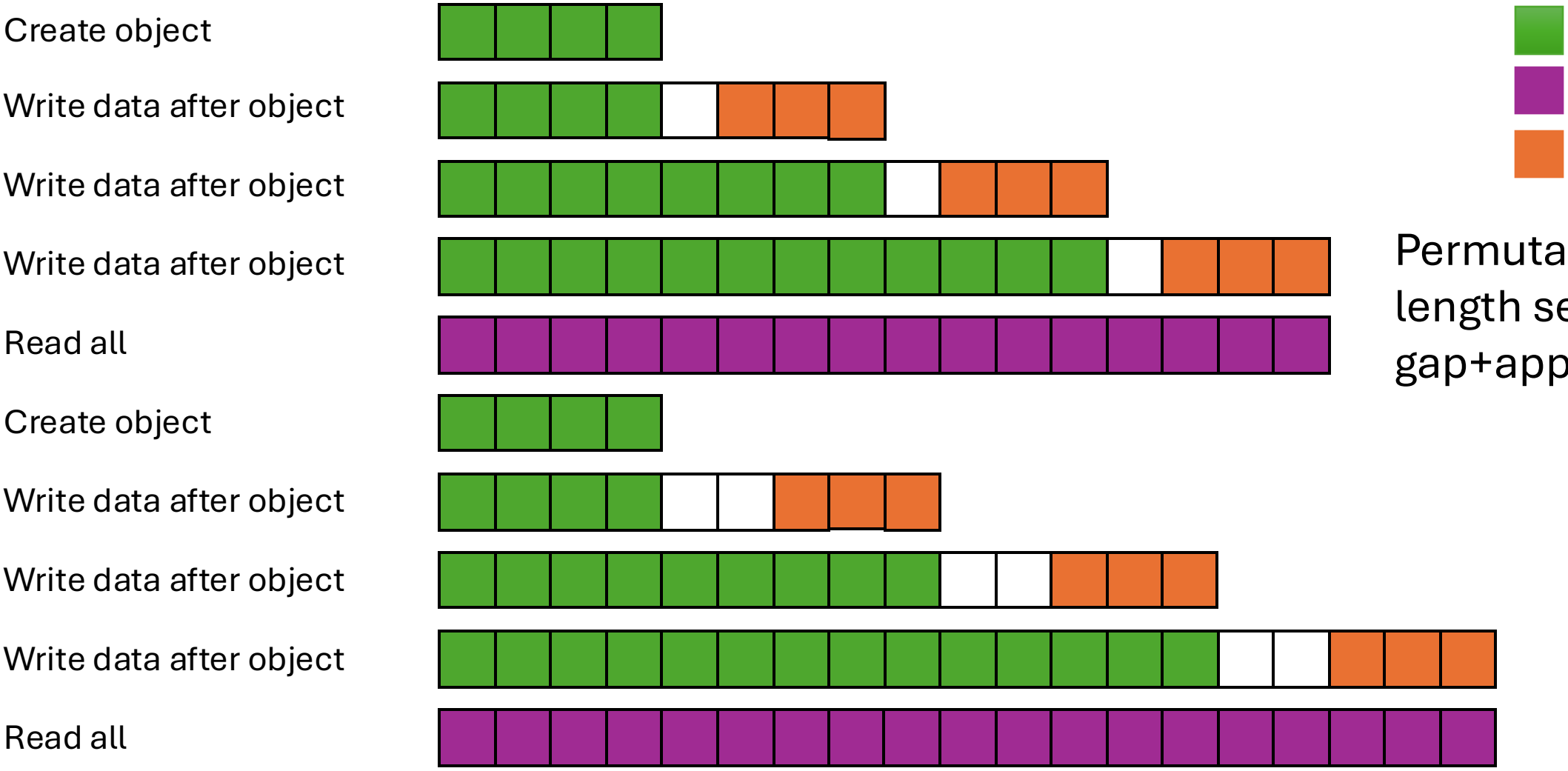
Sequences: Sequence 10



Object
Read IO
Write IO

Sequential writes of fixed length, first doing a roll back and check, then doing a write.

Sequences: Sequence 13



Permutations of length sequential gap+append I/O

Additional useful features

- Parameters
  - --threads
  - --parallel
  - --pool
  - --verbose
  - --interactive

Example invocations:

- `./ceph_test_rados_io_sequence`
- `./ceph_test_rados_io_sequence --km 2,2 --plugin isa`
- `./ceph_test_rados_io_sequence --blocksize 2048 --chunksize 4096 --km 4,2 --plugin isa --technique cauchy --objectsize 2,32 --threads 2 --parallel 4 --verbose`

Additional useful features

Interactive mode

Operations

- create <len>
- read|write|failedwrite <off> <len>
- read2|write2|failedwrite2 <off> <len> <off> <len>
- read3|write3|failedwrite3 <off> <len> <off> <len> <off> <len>
- Injecterror <inject_type> <type> <shard> <good_count> <fail_count>
- clearinject <inject_type> <type> <shard>
- append <length>
- truncate <size>
- remove
- sleep <duration>

Allows manual recreation of scenarios

```
build]# ./bin/ceph_test_rados_io_sequence --interactive
2025-05-19T11:26:12.344+0100 7f6cadaeaac0  0 Test using seed 1747650372
create 20
2025-05-19T11:26:17.409+0100 7f6cadaeaac0  0 Create (size=40K)
write 0 4
2025-05-19T11:26:20.435+0100 7f6cadaeaac0  0 Write1 (offset1=0,length1=8K)
write 4 4
2025-05-19T11:26:22.542+0100 7f6cadaeaac0  0 Write1 (offset1=8K,length1=8K)
read 0 8
2025-05-19T11:26:26.187+0100 7f6cadaeaac0  0 Read1 (offset1=0,length1=16K)
append 12
2025-05-19T11:26:28.538+0100 7f6cadaeaac0  0 Append1 (length1=24K)
write 8 24
2025-05-19T11:26:39.796+0100 7f6cadaeaac0  0 Write1 (offset1=16K,length1=48K)
read 0 32
2025-05-19T11:26:49.575+0100 7f6cadaeaac0  0 Read1 (offset1=0,length1=64K)
```

Future work

- Integrate the tool to run as a part of the Teuthology suite
- Add epoch change listener for
  - Crash detection
  - Verification of epoch changes in recovery tests
- Plan and add testing verifying snapshots
- User control of object deletion after test execution

Code can be found in the main Ceph repository:
https://github.com/ceph/ceph/tree/main/src/test/osd/ceph_test_rados_io_sequence
https://github.com/ceph/ceph/tree/main/src/common/io_exerciser

For contact/questions, you can reach me at:
Jonathan.bailey1@ibm.com


Thank you
Jonathan Bailey