Performance Analysis and Optimization for

Multicore Crimson

Liu, Chunmei chunmei.liu@intel.com

Cheng, Yingxin yingxin.cheng@intel.com

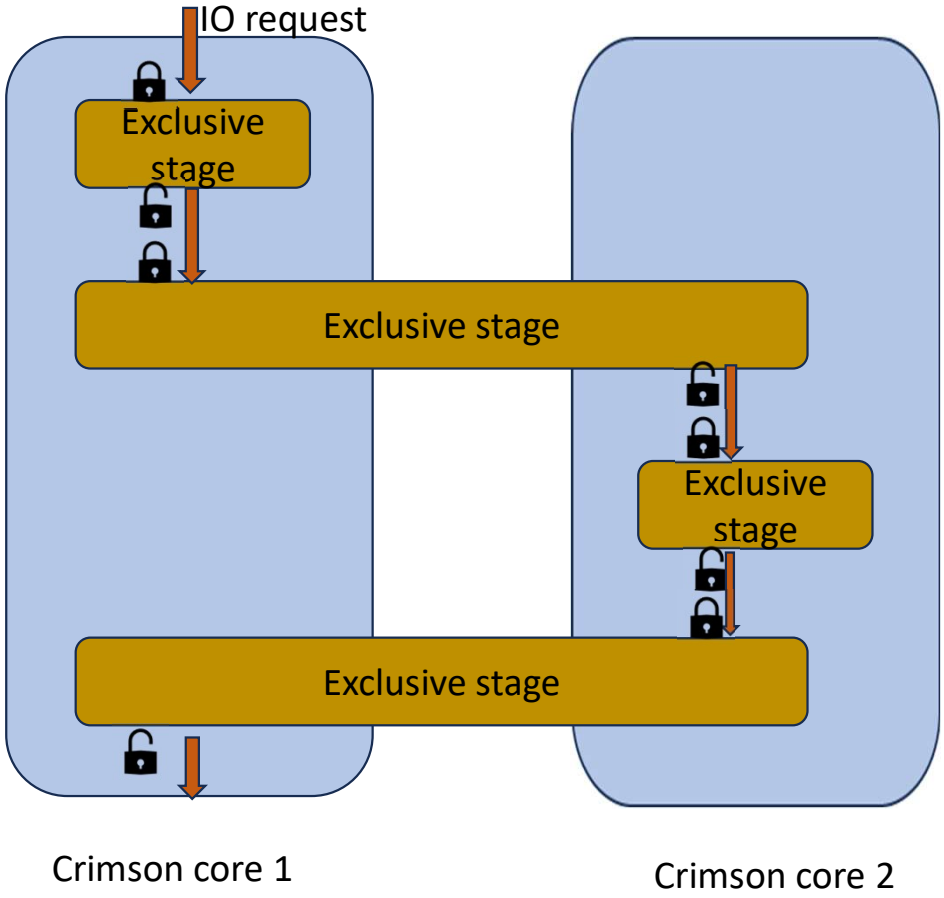
Contents

- Cross core stage mutex analysis and optimization
- Crimson and alien store thread policy
- Avoid sharing variables between threads
- Avoid extra copy for cross-core ptr
- Make some stage concurrent
- Read amplification caused by not coarse-grained cache
- Enlarge LRU cache
- Original vs optimized performance comparation

Cross core stage mutex analysis

- Exclusive stage to guarantee IO request order, only one request go into the stage.
- Using perf to see that seastar reactor polling time is about 50%, seems it has heavy starvation.
- Calculate enter and exit stage time and found IO request stay in some stages for a long time.
- The stage is cross core stage,when send request to another core, mutex will not be unlocked until request is send to another core to guarantee cross core sequence, so block the following request.
- Cross core send take more times.

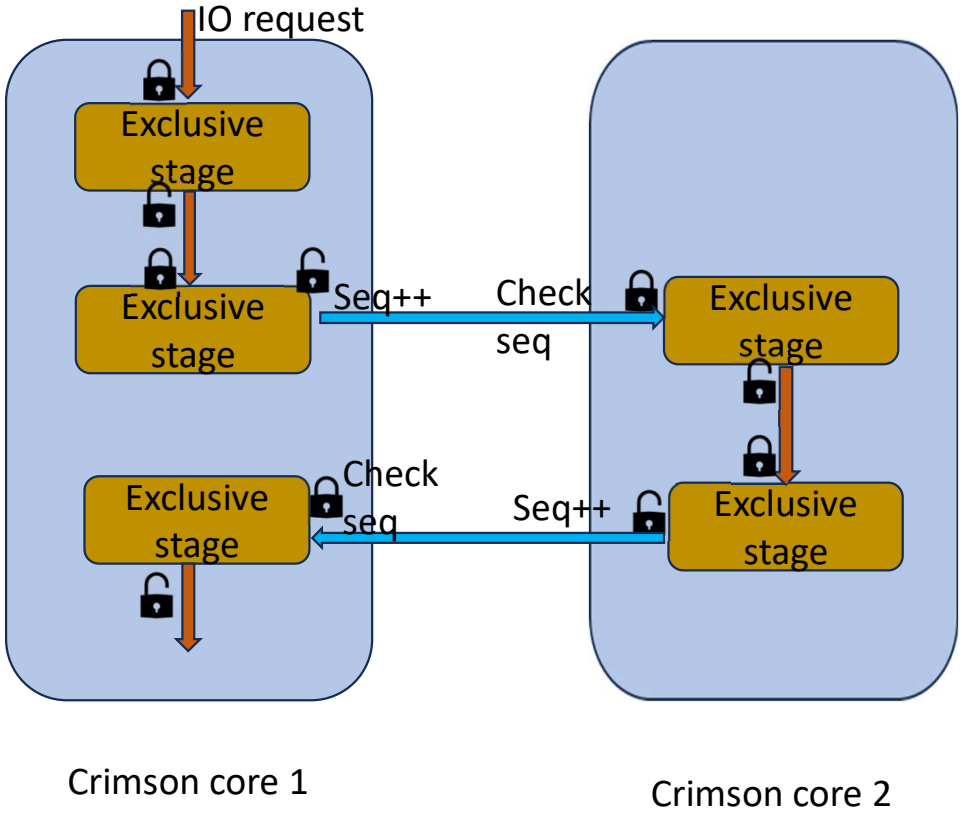
Original exclusive stage workflow



Cross core stage mutex optimization

- Optimize the stage phase, free the lock before send request to another core.
- And use seq number to guarantee the cross core sending sequence.
- The target core will check the sequence number order for a certain original core.
- Then Using perf to see that seastar reactor polling time is below 10%

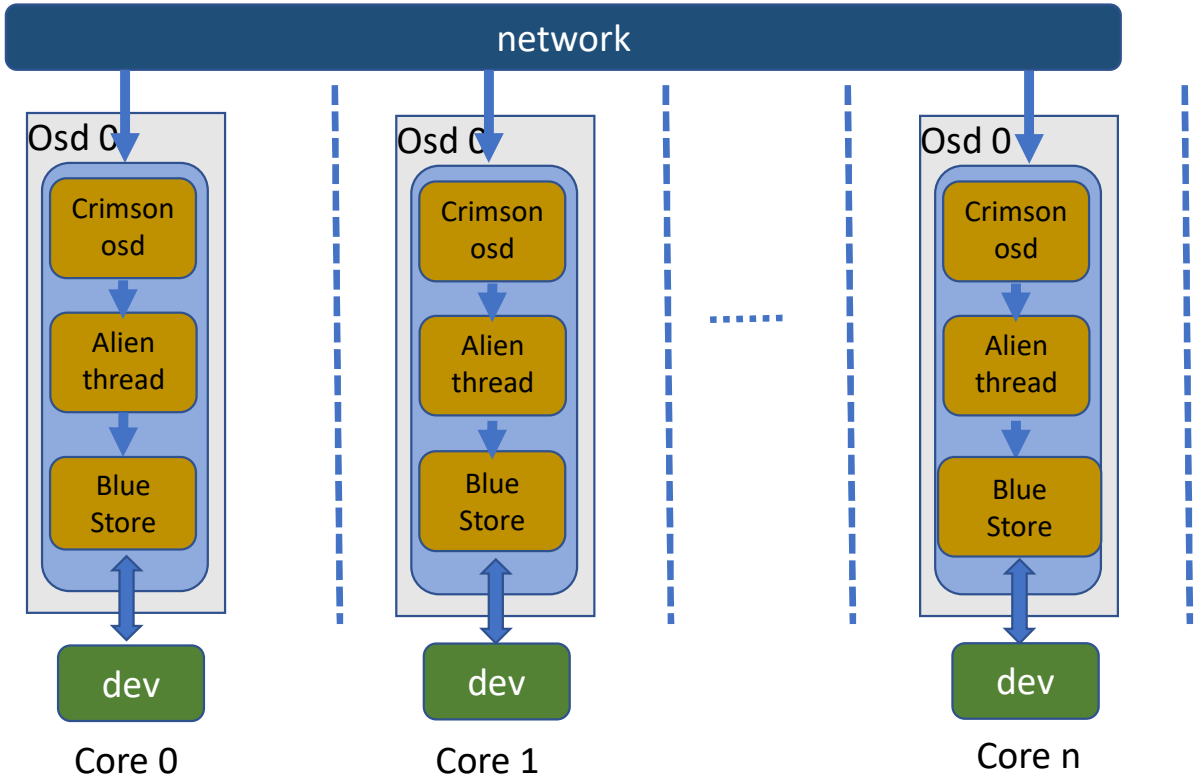
Optimize cross core exclusive stage



Crimson+Alienstore threads analysis

- Crimson osd is seastar thread, and bluestore is POSIX thread. Alienstore is a wrapper of bluestore to provide interface to crimson osd.
- At the testing beginning, crimson threads, alien threads, and bluestore threads mixed and share the same cores. This thread policy can't provide the best throughput since crimson osd is run to complete design which use almost the whole cpu.

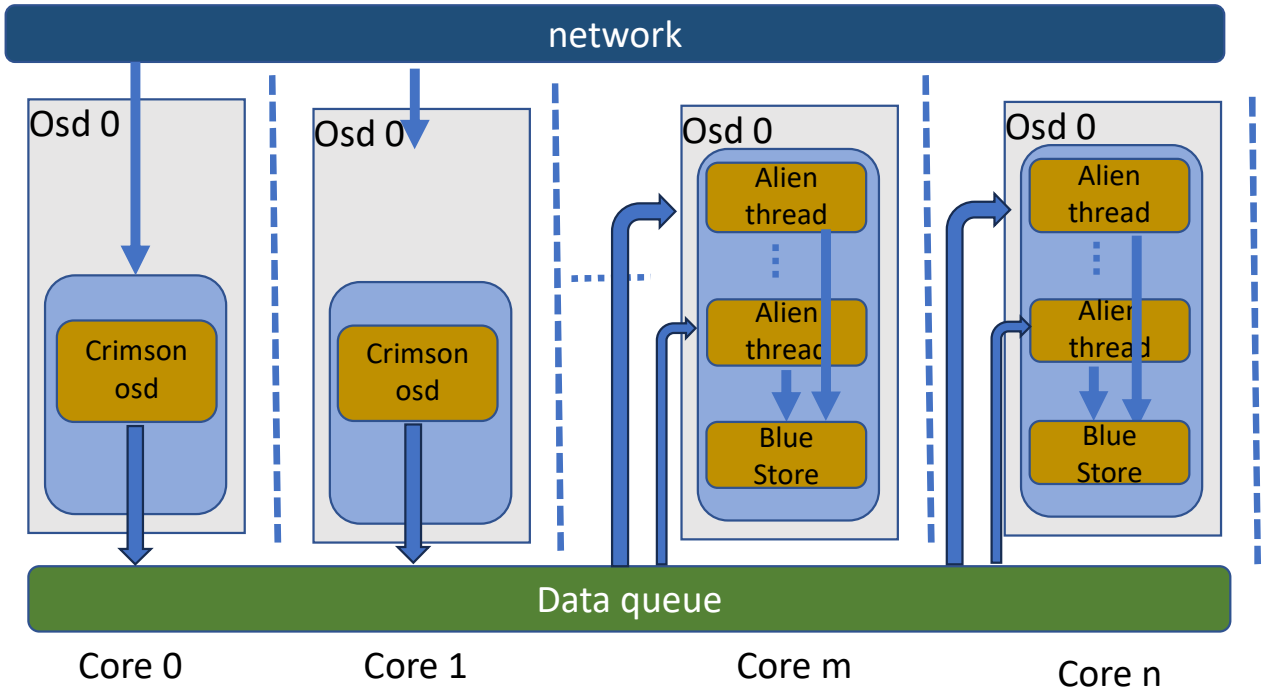
Original crimson and Alien Store threads policy



Crimson+Alienstore threads optimization

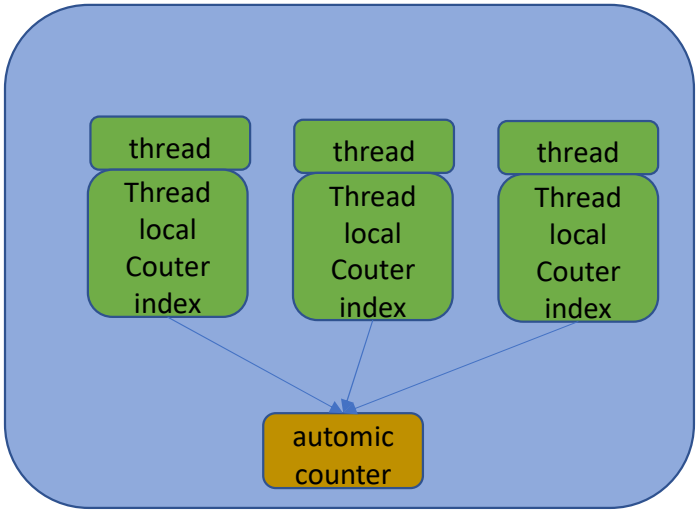
- Optimize the thread policy, let each crimson osd thread occupy a logic core, and POSIX threads share other logical cores.
- After adjust the ratio of crimson osd thread cores and alien thread cores, also alien thread numbers, we can get the best performance for a certain total cores test case.

crimson and Alien Store threads policy optimization

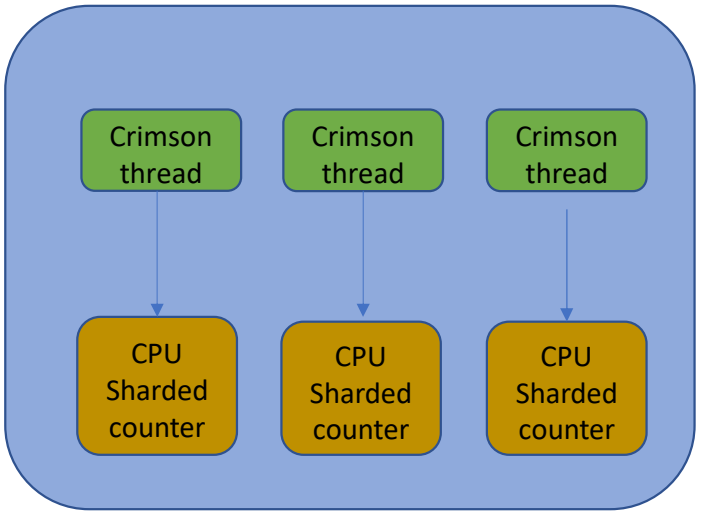


Avoid sharing variables

- Original mempool counters

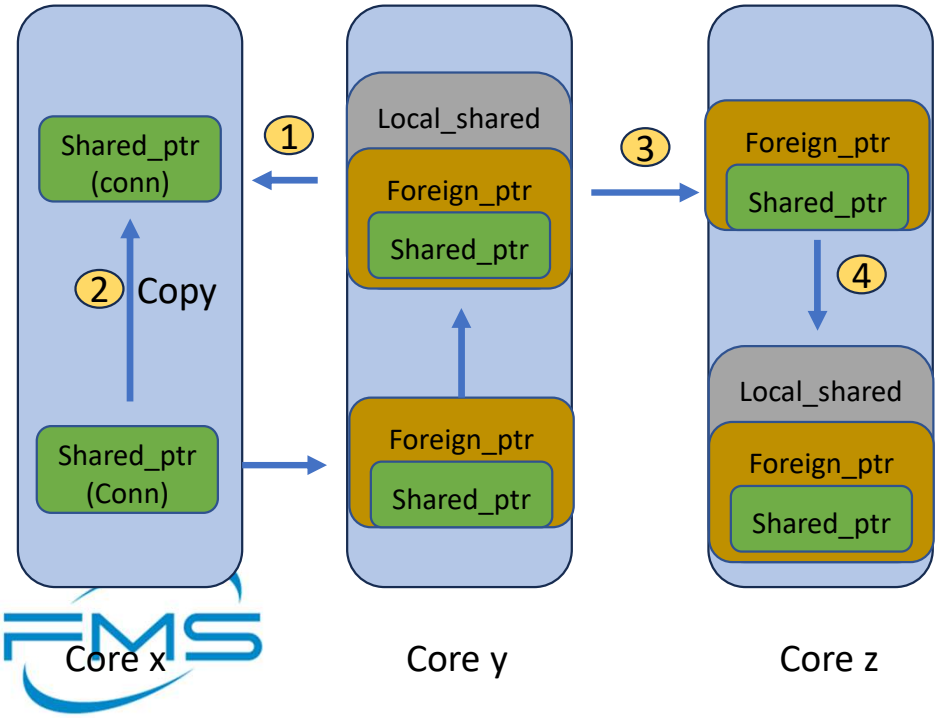


- Optimized for crimson

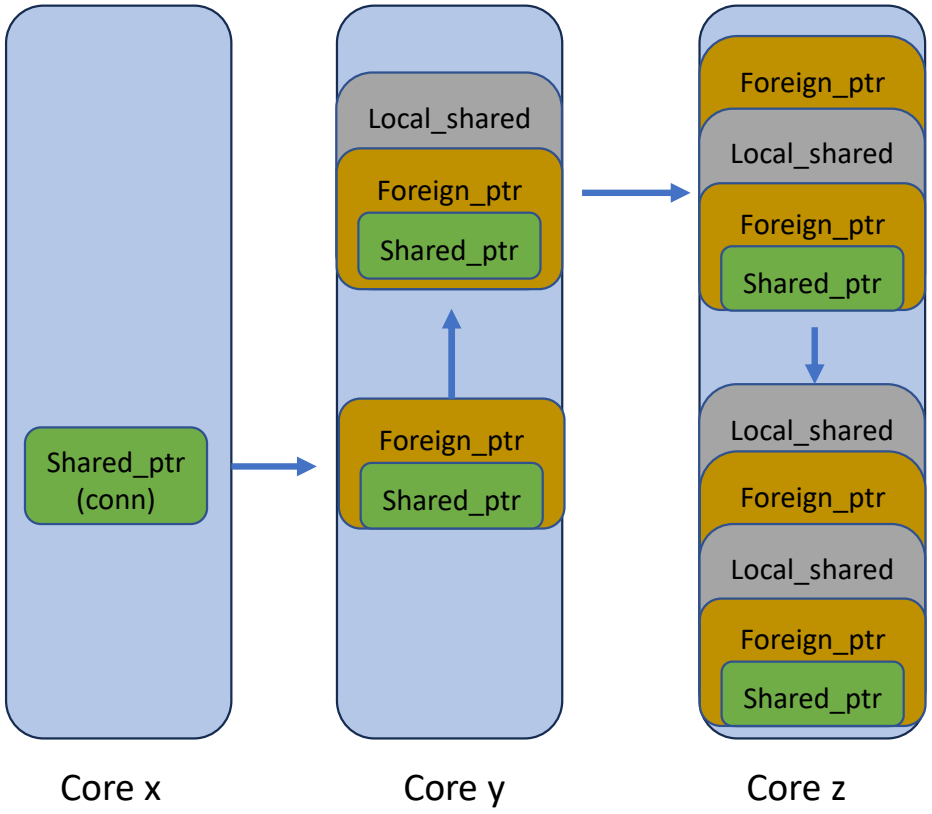


Avoid extra copy

- Original foreign_ptr usage

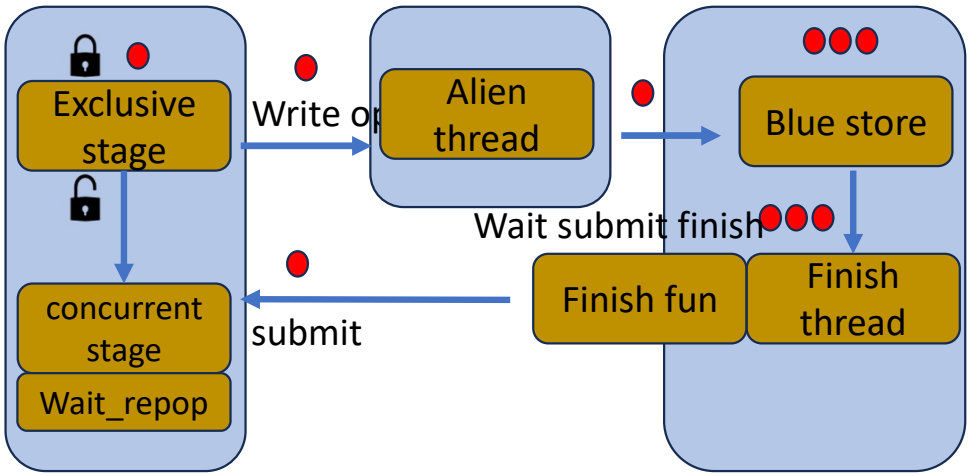


- Optimized foreign_ptr usage avoid cross-core copy

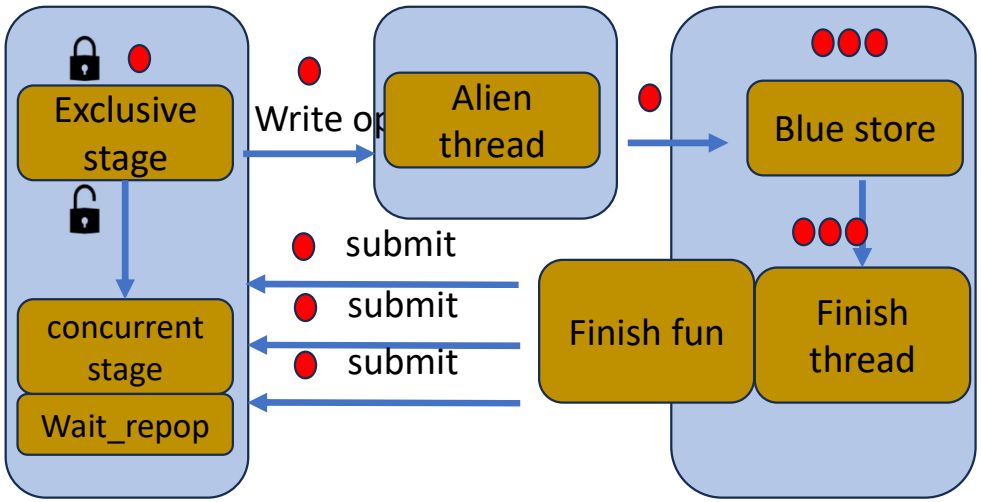


Submit concurrently

- Original alien submit to crimson osd

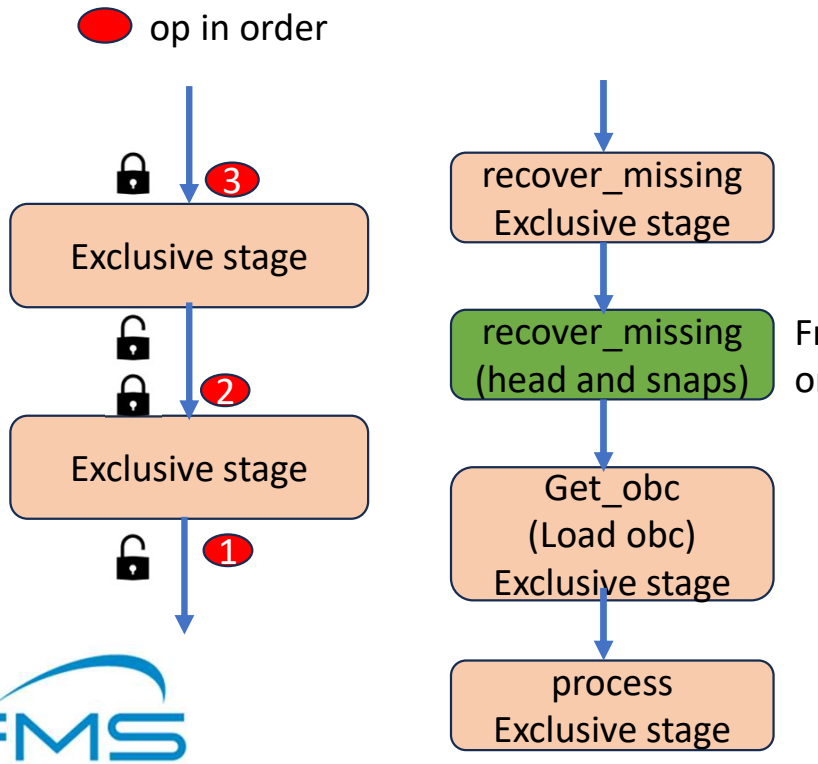


- Optimized alien submit to crimson osd

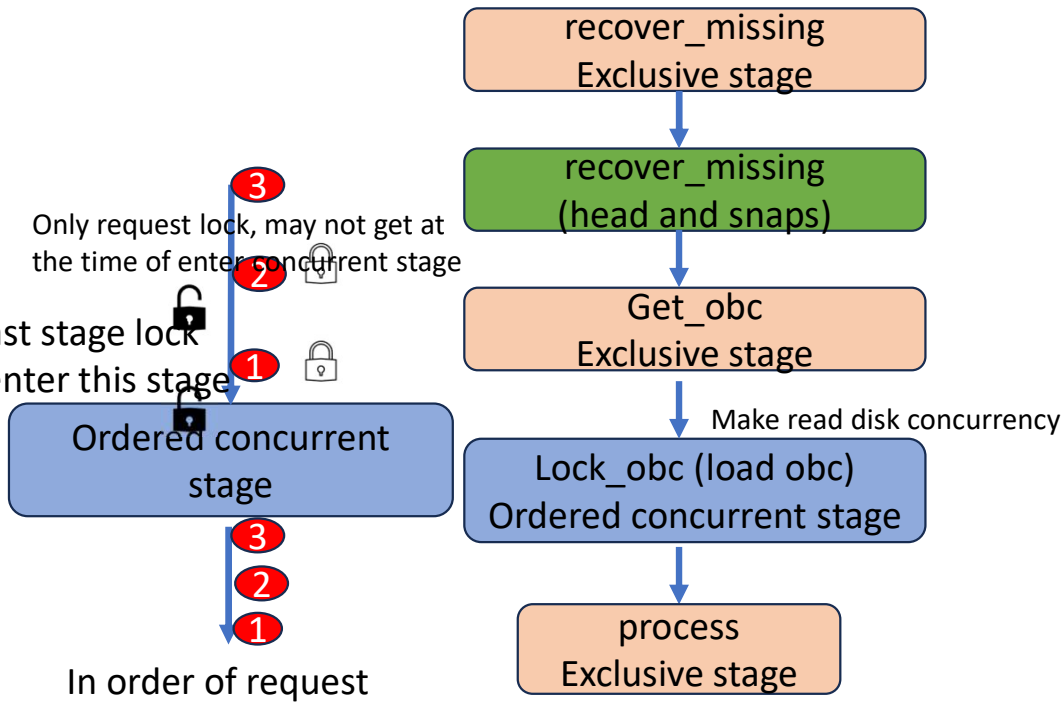


Change exclusive stage to ordered concurrent stage

- Exclusive stage

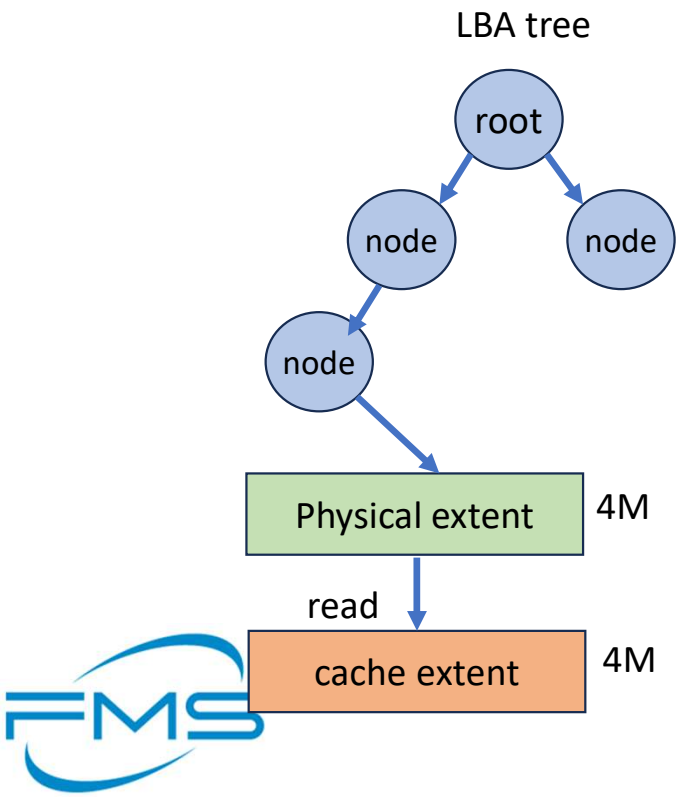


- Ordered concurrent stage

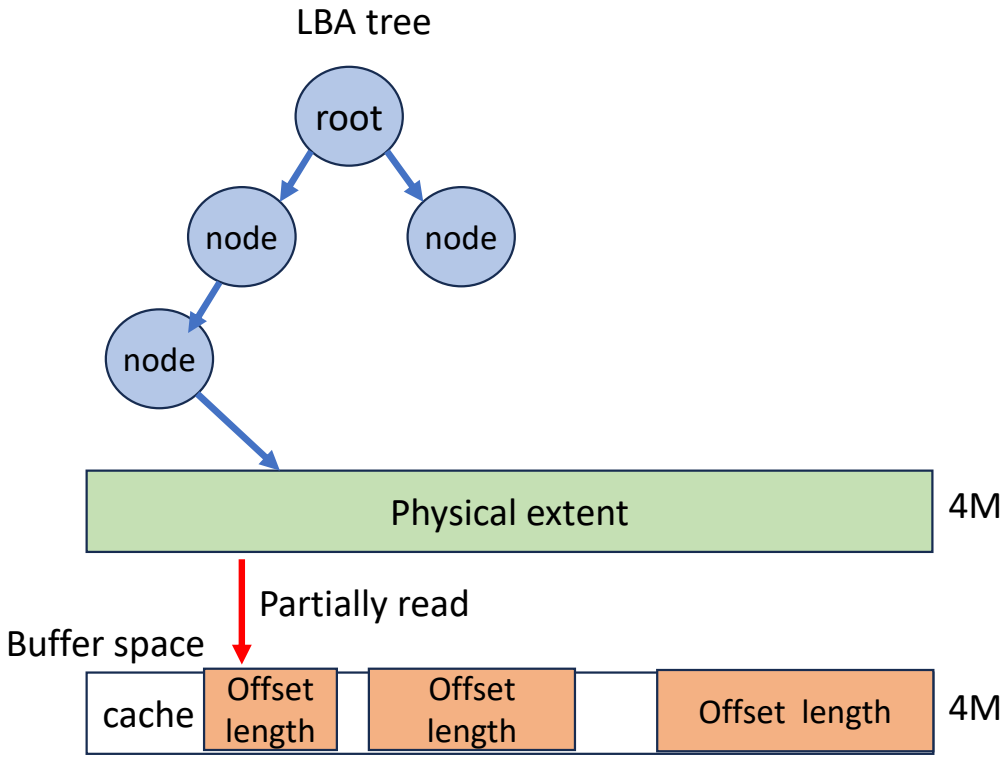


Reduce read amplification by partially read extent

- Original Read whole extent

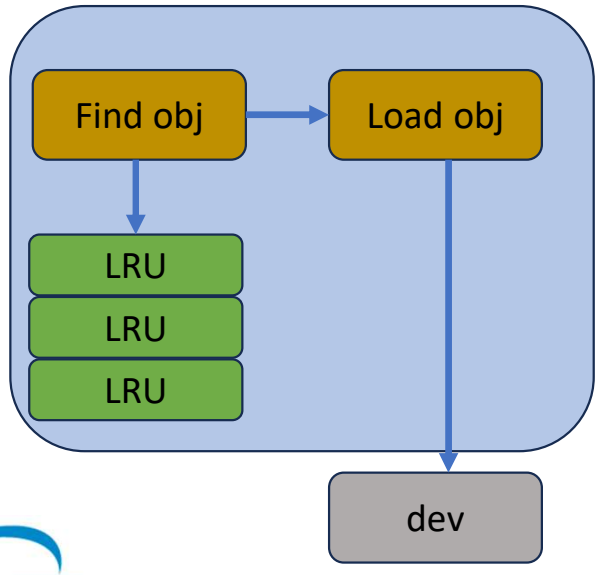


- Optimized to read partially

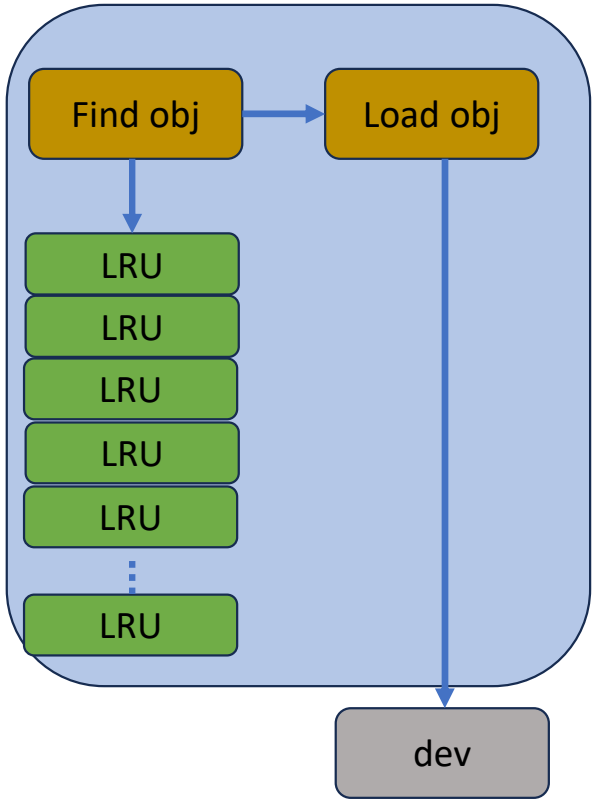


Increase obc LRU cache size

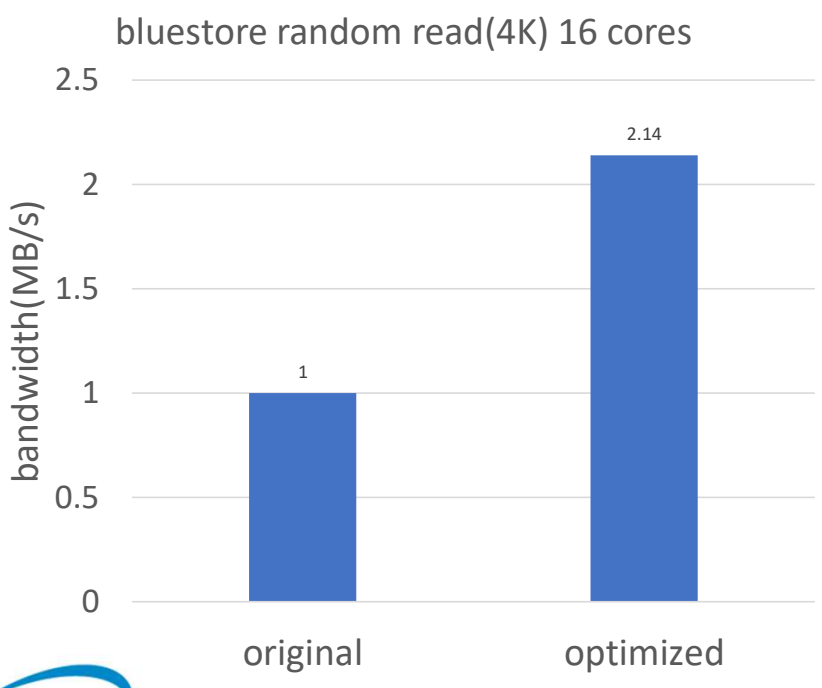
- Small cache size causing read data from device frequently

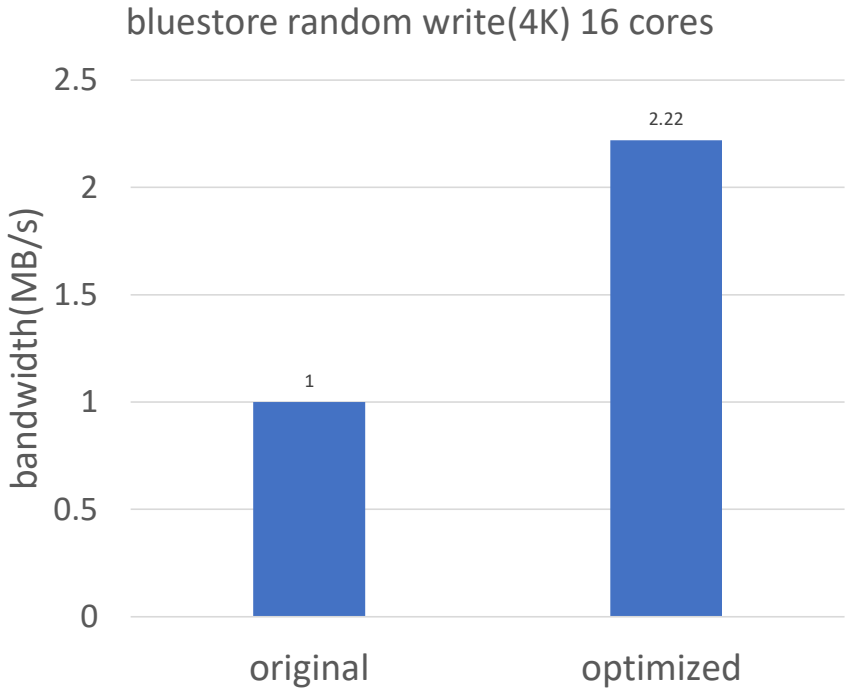


- Enlarge cache size to increase cache hitting ratio



Original vs optimized performance





Reference

1. Cross core stage: https://github.com/ceph/ceph/pull/53537 and https://github.com/ceph/ceph/pull/53934
2. Mempool shared counter: https://github.com/ceph/ceph/pull/53130
3. Make loading-obc concurrent: https://github.com/ceph/ceph/pull/55488
4. Alienstore submission: https://github.com/ceph/ceph/pull/55039
5. Optimize foreign copy: https://github.com/ceph/ceph/pull/54896
6. Reduce read amplification: https://github.com/ceph/ceph/pull/57787
7. Enlarge cache size: https://github.com/ceph/ceph/pull/55188
8. Crimson and alien threads policy: https://github.com/ceph/ceph/pull/55767
9. multi-core crimson messenger: https://github.com/ceph/ceph/pull/51916

Thanks!

Q & A