Crimson Seastore: performance comparison cache pin algorithms LRU vs. 2Q – progress report

José Juan Palacios Pérez
Ceph IBM,
Manchester UK

August 28, 2025

Abstract

In this brief report we show the performance comparisonm between the two Seastore cache pinboard algorithms, namely LRU and 2Q. We used the following configurations:

- we used a single RBD volume of size 2 GB (which is the default size of the cache per reactor).
- We ranged the number of Seastar reactors as 1, 4, and 8 (to show scalability).
- We used the same ceph dev build from main branch (hash ab72fd0620f) for all test runs.
- All configurations used balanced OSD algorithm on a single NUMA socket (this is to be consistent with previous tests).
- We used three random distributions via FIO options, namely: normal (Gaussian), zoned, and Zipfian.

Our preliminary conclusions:

- The performance of the two pinboard cache algorithms do not show a significant difference, either in throughput or latency, nor in CPU resource utilisation.
- As expected from previous tests, Crimson SeaStore on random workloads scales well with the number of reactors.

Contents

1. Summary performance scaling Seastore LRU vs 2Q

In this Chapter we show the performance comparison between the two pinboard cache algorithms, LRU and 2Q. As in previous reports, we ran long duration tests to produce response latency curves.

1. We used a single OSD, running within a single NUMA socket, we only increased the number of reactors as indicated. We used the balanced OSD algorithm for the reactors affinity. We used the default cache size of 2 GB per reactor for Seastore.
2. We used a single RBD image of 2GB size. All FIO processes were pinned to the other NUMA socket, so that the OSD processes were not affected by the FIO processes.
3. We disabled RBD coalescing (eventhough we did not run sequential workloads for this test).

We first compare the two algorithms on each workloads, summarising the random distribution used. To make the test valid, it is important to ensure that the highest IO intensity of the distribution fits within the cache.

For each workload comparison, we show the CPU utilisation (from the Linux top command) for the two main threads in the OSD process, the reactor and the crimson-osd thread.[1] For multiple reactors, we coalesce the data from all reactors, showing the average along the time duration of the test.

[1] We do not show the memory utilisation since its mostly remains constant, and does not add much to the analysis.

1.1 randwrite_zoned

This random distribution exercises IO over the storage defined zones (in LBA) according to the parameters chosen, in our case as follows:

- 60% of accesses should be to the first 10% (of the total storage space, in LBA)
- 30% of accesses should be to the next 20%
- 8% of accesses should be to the next 30%
- 2% of accesses should be to the next 40%.

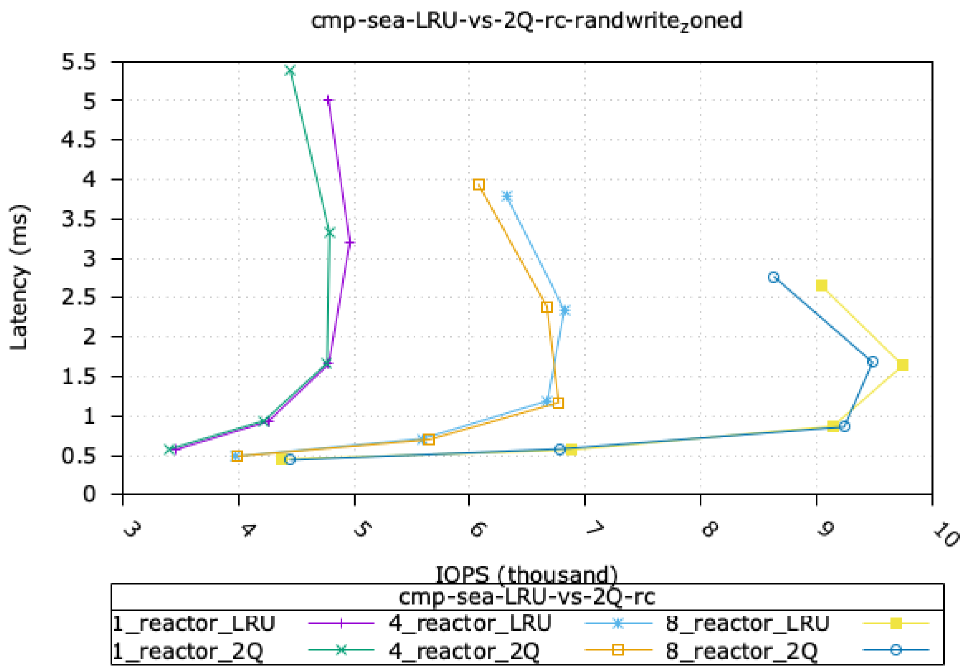
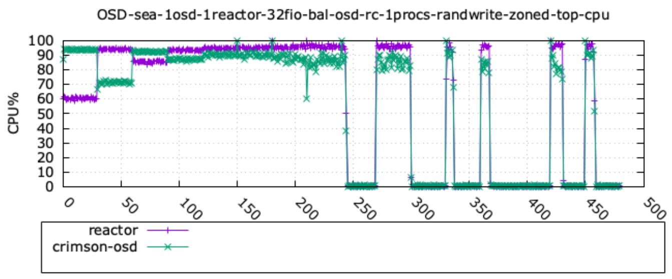


Figure 1.1: cmp-sea-LRU-vs-2Q-rc - randwrite_zoned



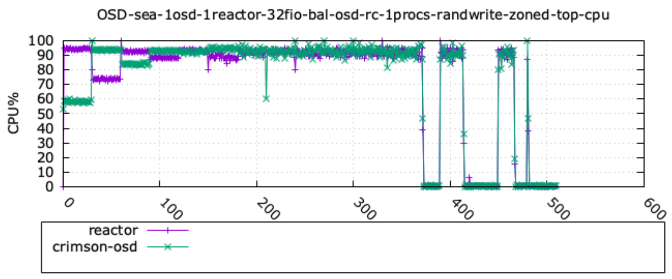
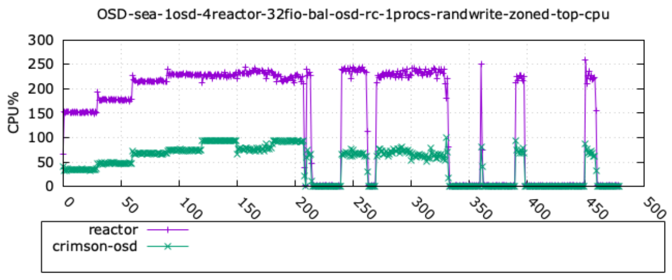


Figure 1.2: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_zoned, 1 reactor



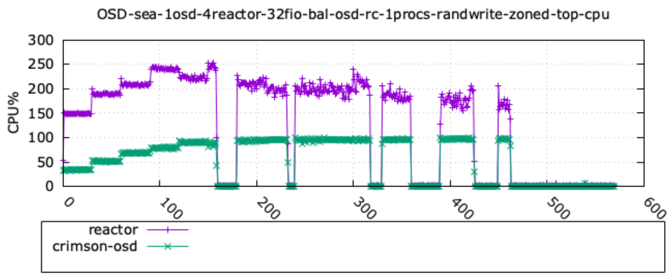


Figure 1.3: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_zoned, 4 reactor

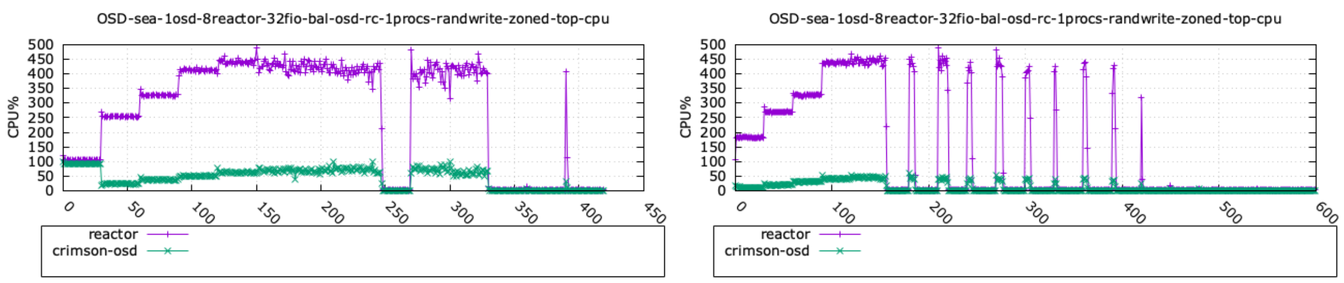


Figure 1.4: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_zoned, 8 reactor

1.2 randwrite_zipf

This random distribution was chosen to exercise IO as follows:

```
Generating Zipf distribution with 0.768780 input and 2 GiB size and 4096 block_size.

   Rows           Hits %         Sum %           # Hits          Size
-----------------------------------------------------------------------
Top   16.67%       56.93%         56.93%          298480          1.14G
|->   33.33%       13.88%         70.81%           72775        284.28M
|->   50.00%        8.38%         79.19%           43946        171.66M
|->   66.67%        6.94%         86.13%           36364        142.05M
|->   83.33%        6.94%         93.07%           36364        142.05M
|-> 100.00%         6.93%        100.00%           36359        142.03M
-----------------------------------------------------------------------
Total                                             524288
```

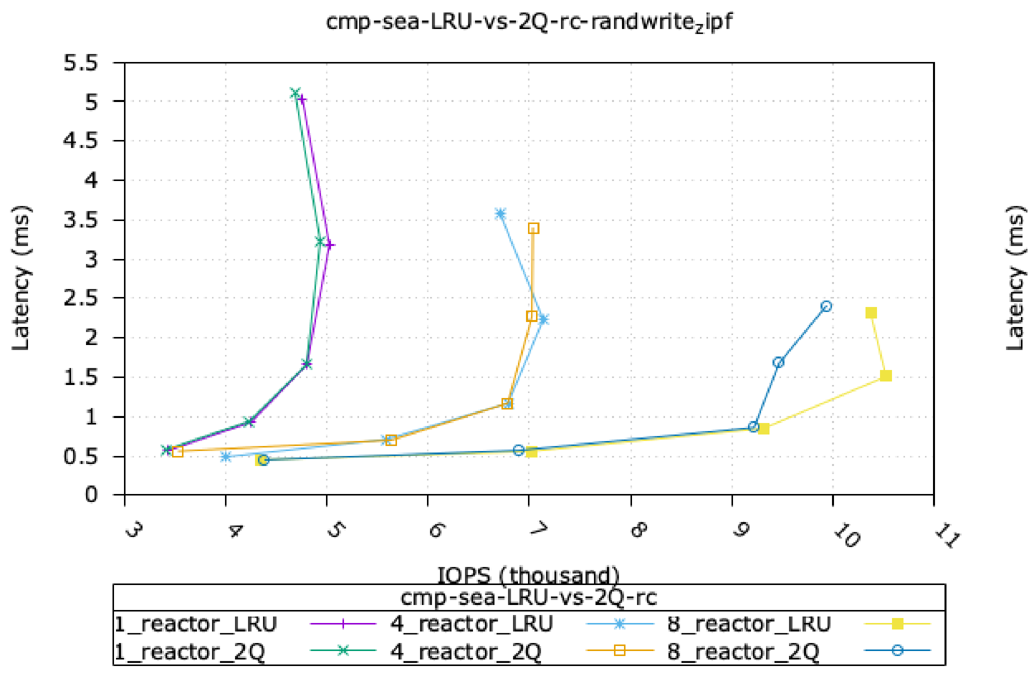
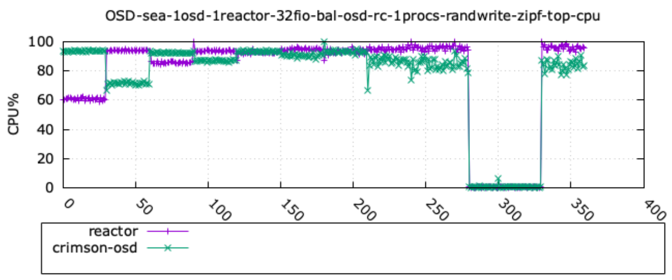


Figure 1.5: cmp-sea-LRU-vs-2Q-rc - randwrite_zipf



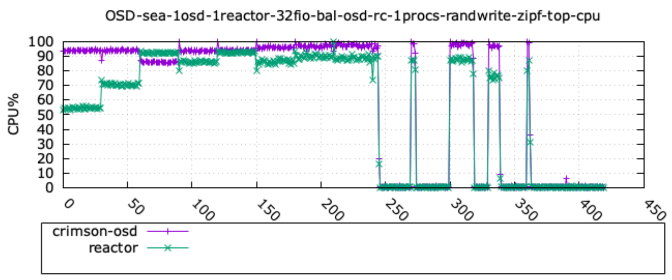
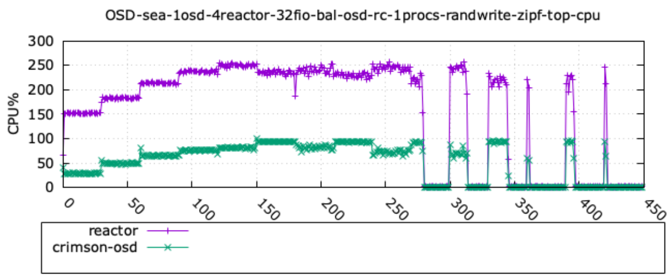


Figure 1.6: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_zipf, 1 reactor



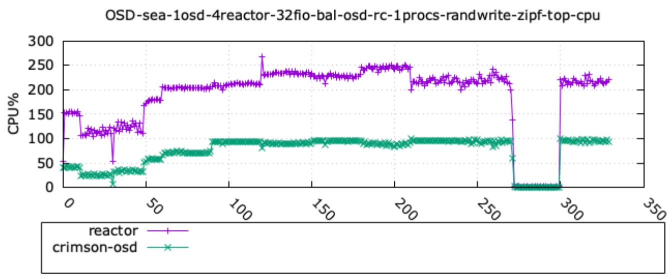
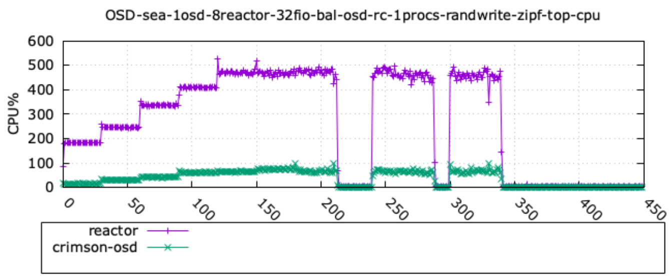


Figure 1.7: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_zipf, 4 reactor



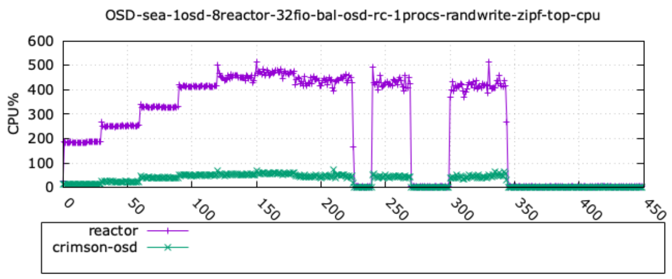


Figure 1.8: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_zipf, 8 reactor

1.3 randwrite_norm

This is the traditional Gaussian distribution, using a generated mean and standard deviation 0.6.

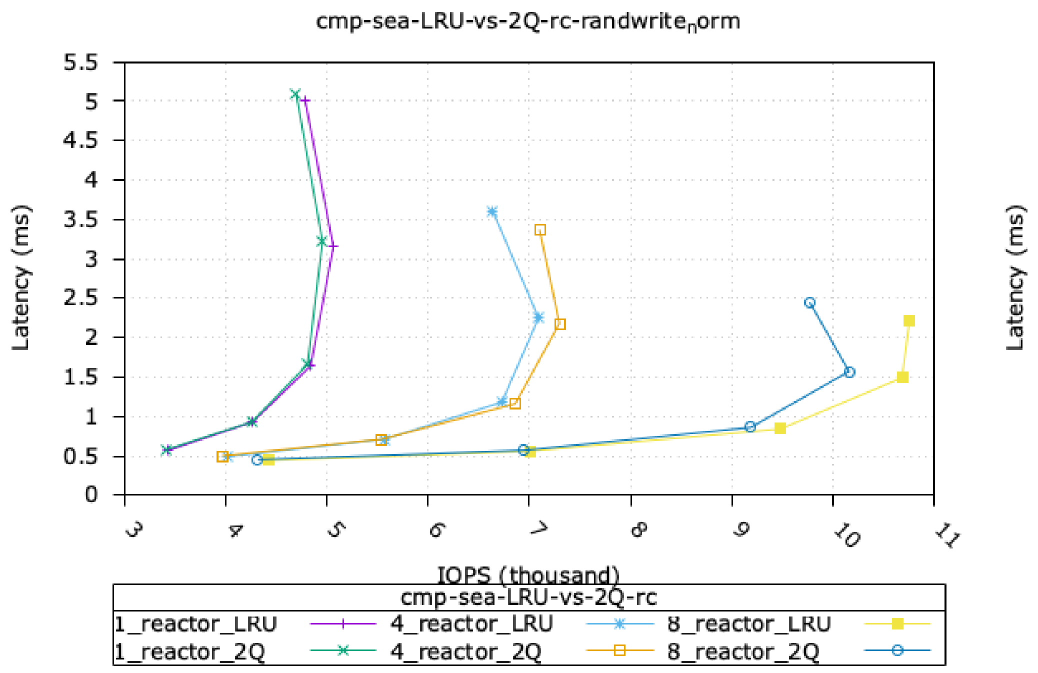


Figure 1.9: cmp-sea-LRU-vs-2Q-rc - randwrite_norm

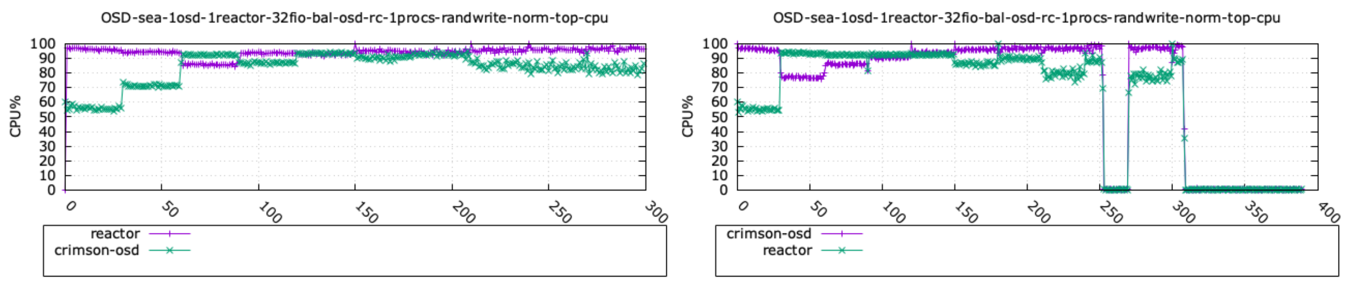


Figure 1.10: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_norm, 1 reactor

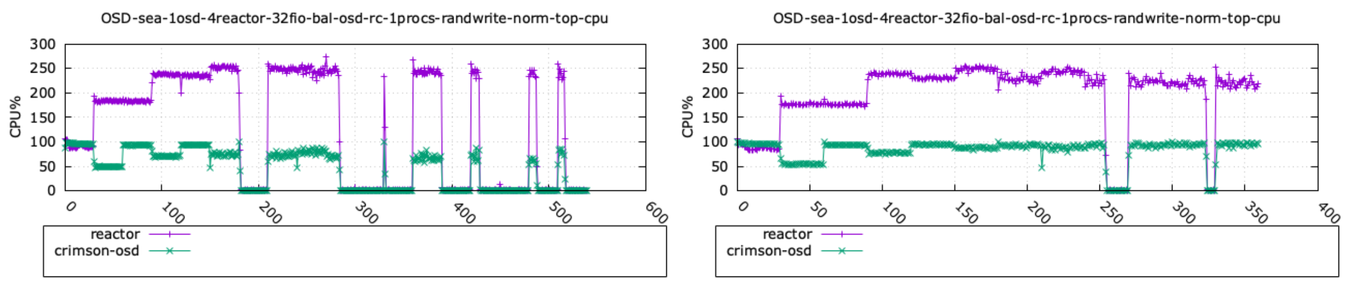


Figure 1.11: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_norm, 4 reactor

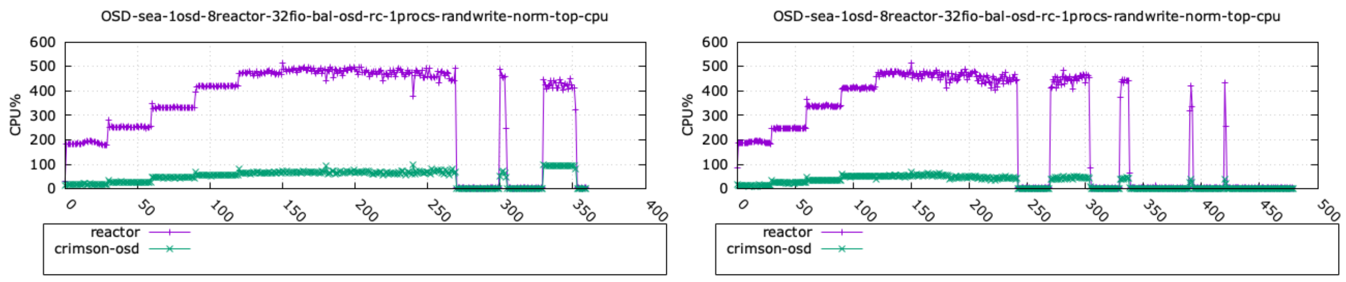


Figure 1.12: Top CPU utilization - LRU (left) vs 2Q (right) - randwrite_norm, 8 reactor

1.4 randread_zoned

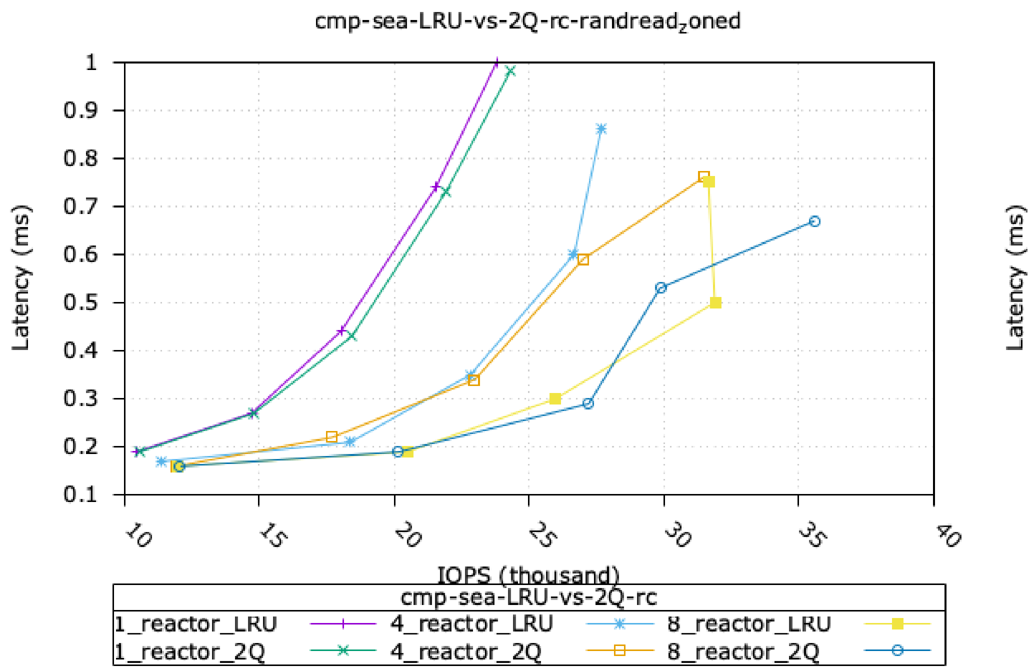


Figure 1.13: cmp-sea-LRU-vs-2Q-rc - randread_zoned

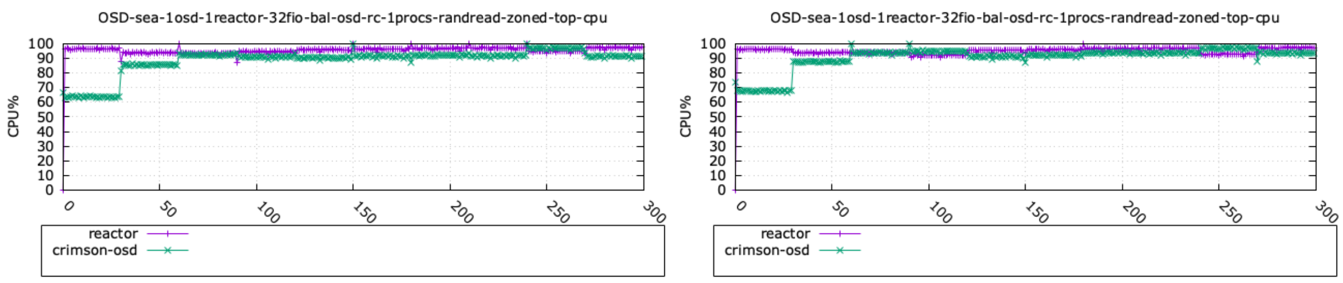


Figure 1.14: Top CPU utilization - LRU (left) vs 2Q (right) - randread_zoned, 1 reactor

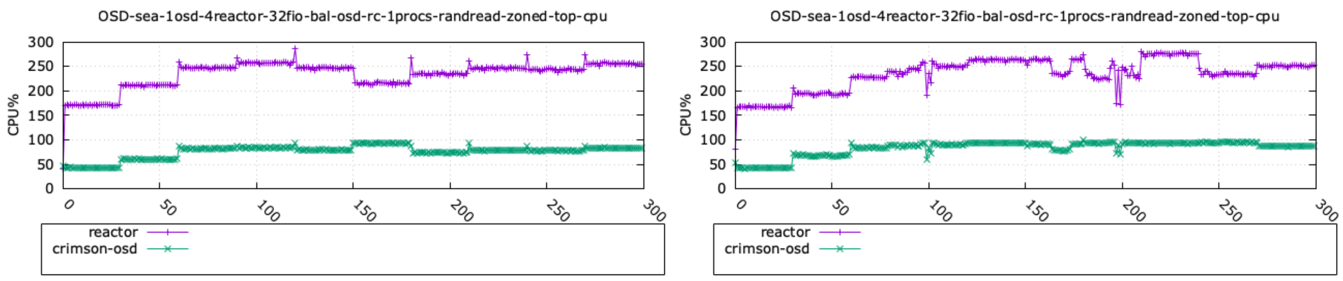


Figure 1.15: Top CPU utilization - LRU (left) vs 2Q (right) - randread_zoned, 4 reactor

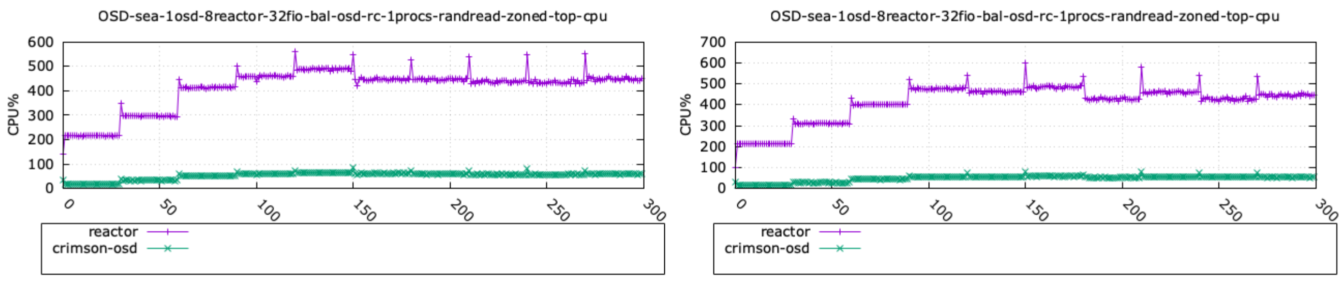


Figure 1.16: Top CPU utilization - LRU (left) vs 2Q (right) - randread_zoned, 8 reactor

1.5 randread_zipf

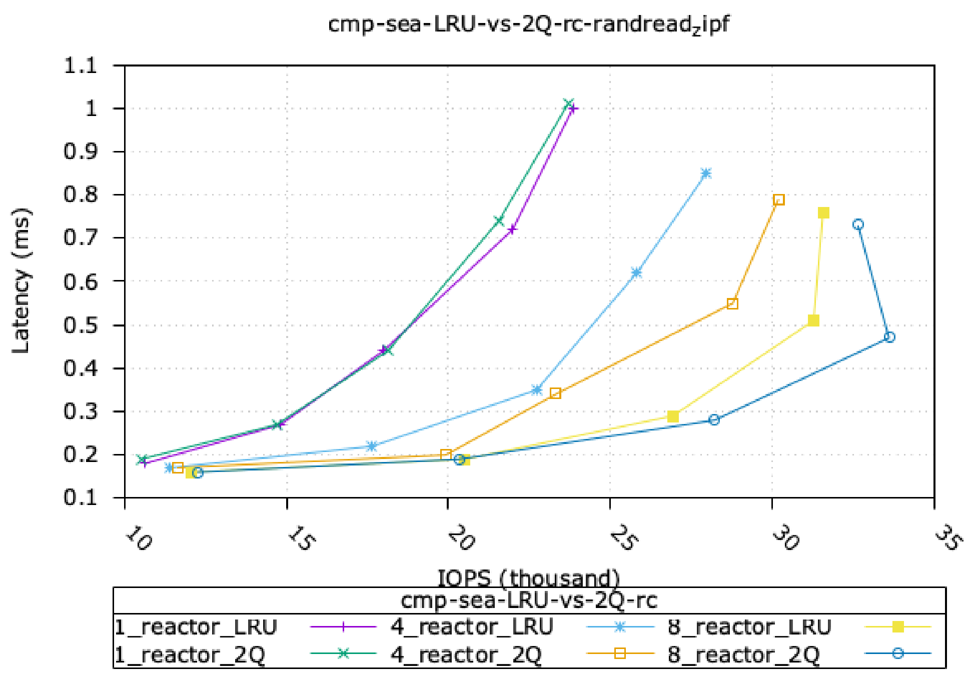


Figure 1.17: cmp-sea-LRU-vs-2Q-rc - randread_zipf

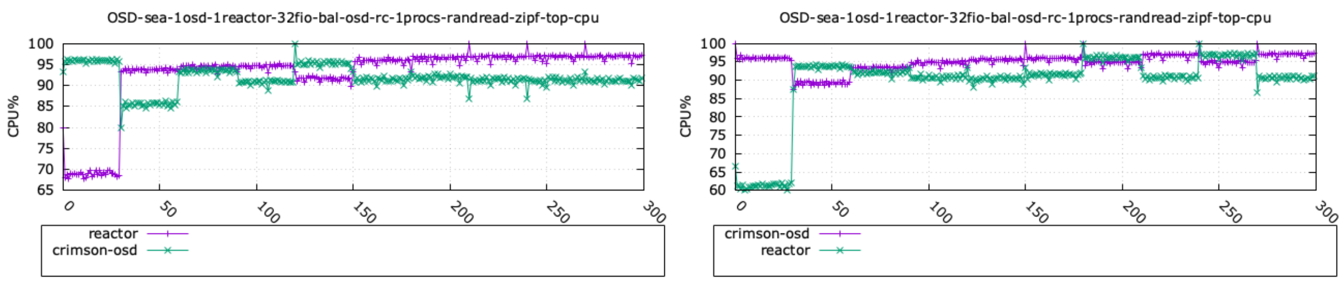


Figure 1.18: Top CPU utilization - LRU (left) vs 2Q (right) - randread_zipf, 1 reactor

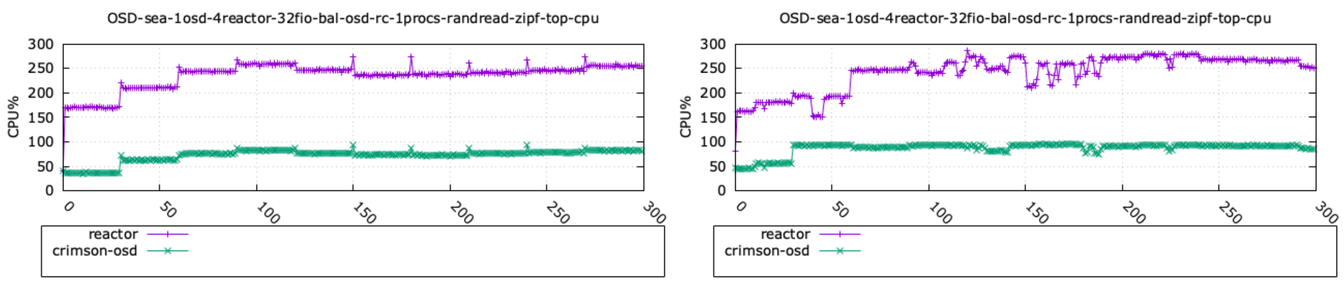


Figure 1.19: Top CPU utilization - LRU (left) vs 2Q (right) - randread_zipf, 4 reactor

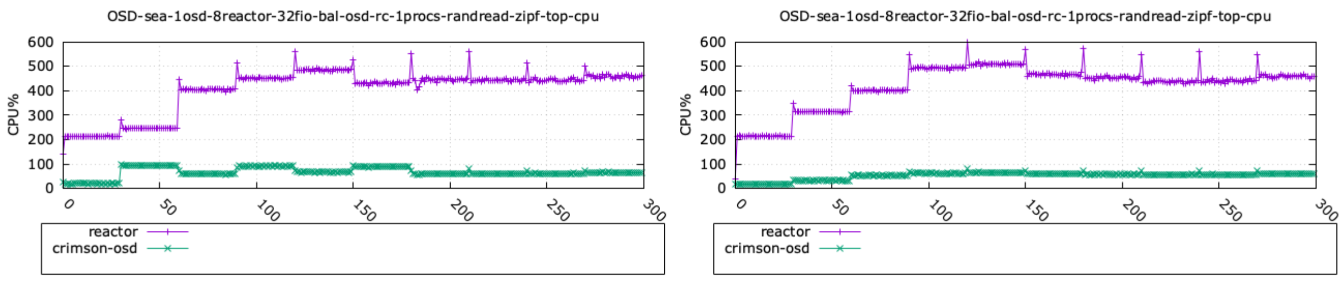


Figure 1.20: Top CPU utilization - LRU (left) vs 2Q (right) - randread_zipf, 8 reactor

1.6 randread_norm

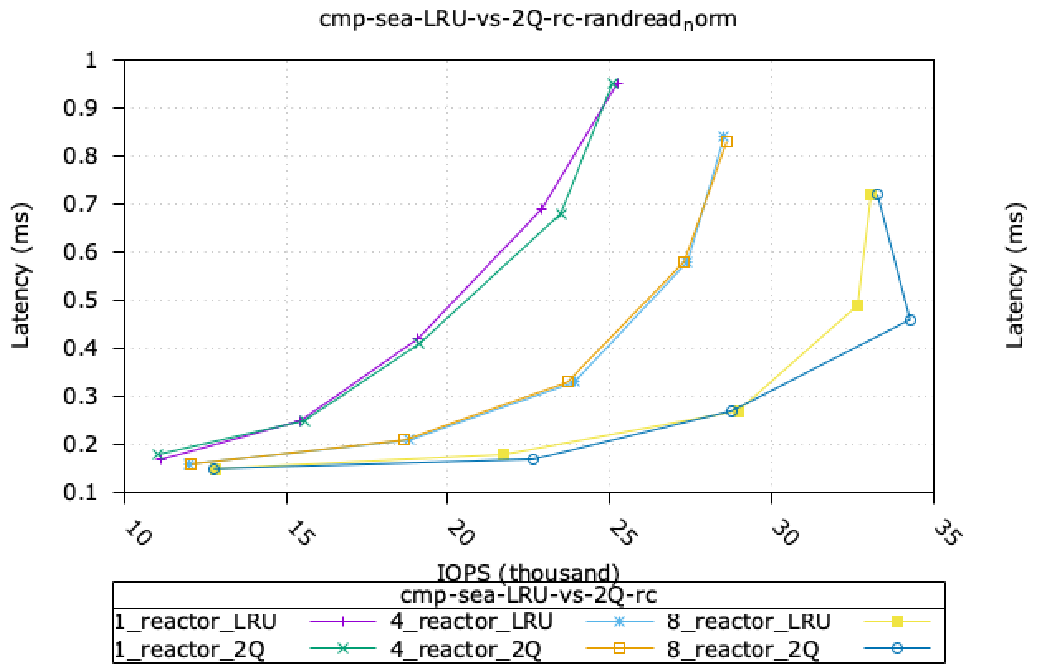


Figure 1.21: cmp-sea-LRU-vs-2Q-rc - randread_norm

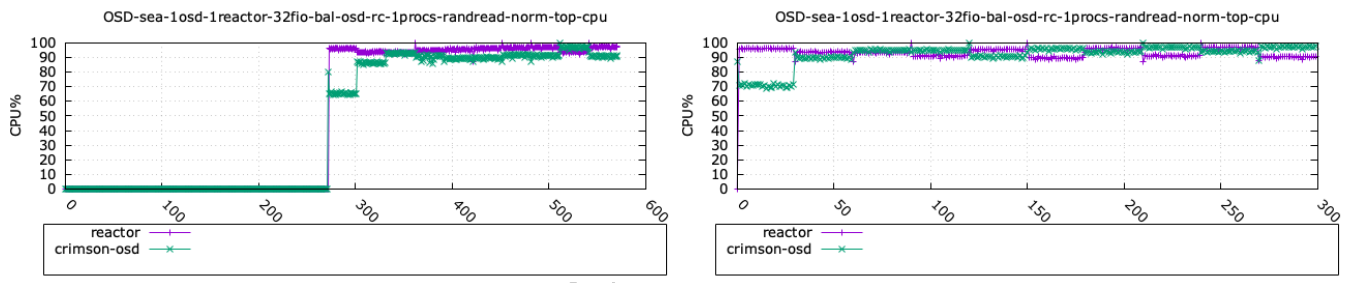


Figure 1.22: Top CPU utilization - LRU (left) vs 2Q (right) - randread_norm, 1 reactor

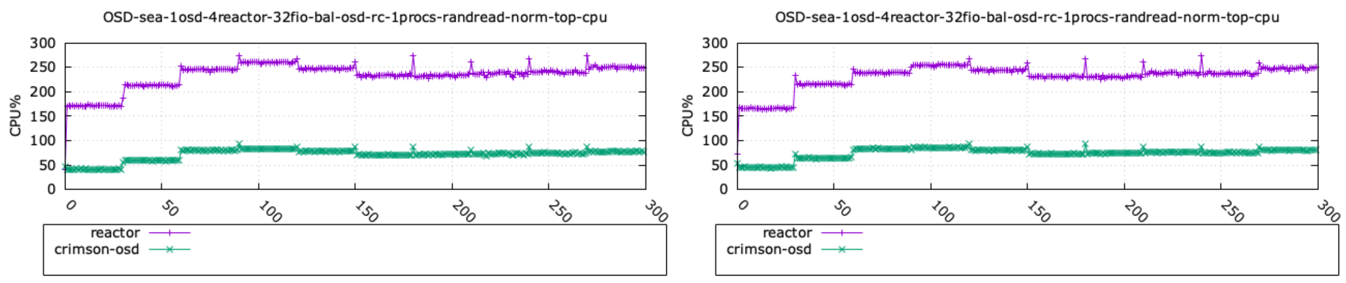


Figure 1.23: Top CPU utilization - LRU (left) vs 2Q (right) - randread_norm, 4 reactor

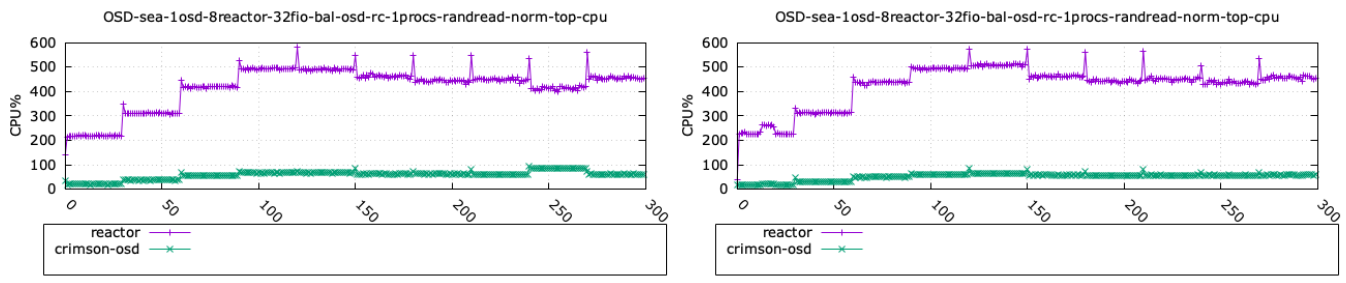


Figure 1.24: Top CPU utilization - LRU (left) vs 2Q (right) - randread_norm, 8 reactor