



Crimson Performance: 56 weeks later

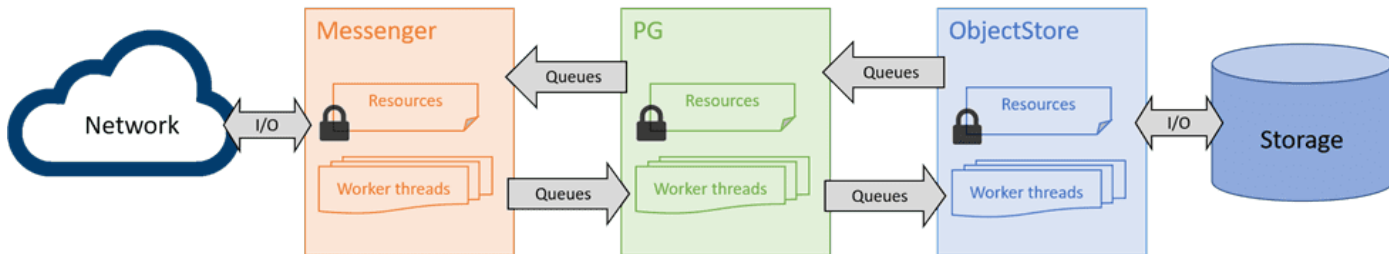
On becoming a Jack of all trades

Jose Juan Palacios-Perez
IBM UK

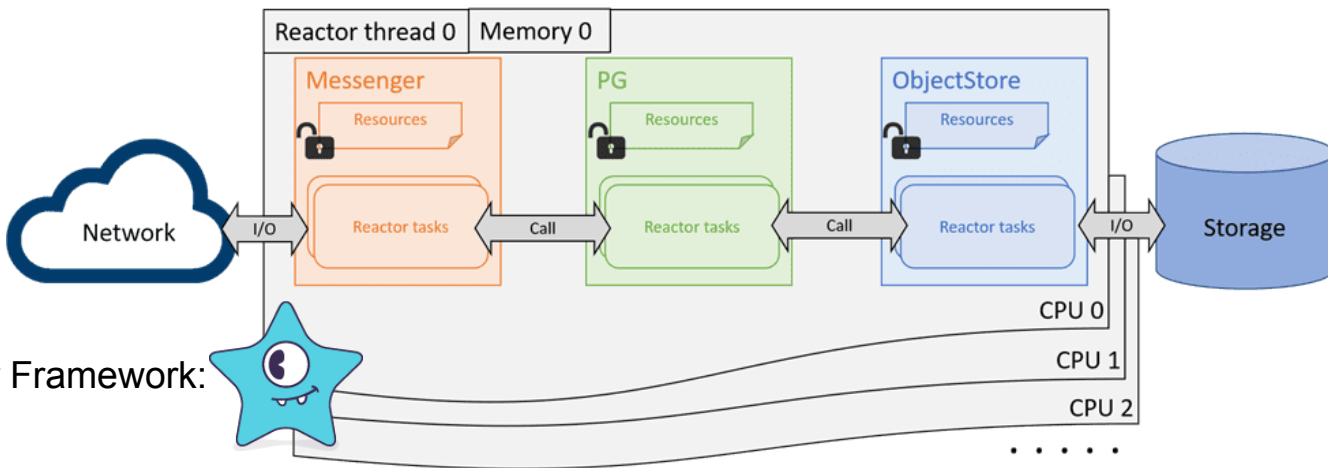
Crimson: reactor (share nothing) OSD architecture



Classic:



Crimson:



Seastar Framework:

Crimson: reactor (share nothing) OSD architecture



Crimson is engineered to be a high-performance Object Storage Daemon (OSD) optimized for fast storage devices such as NVMe

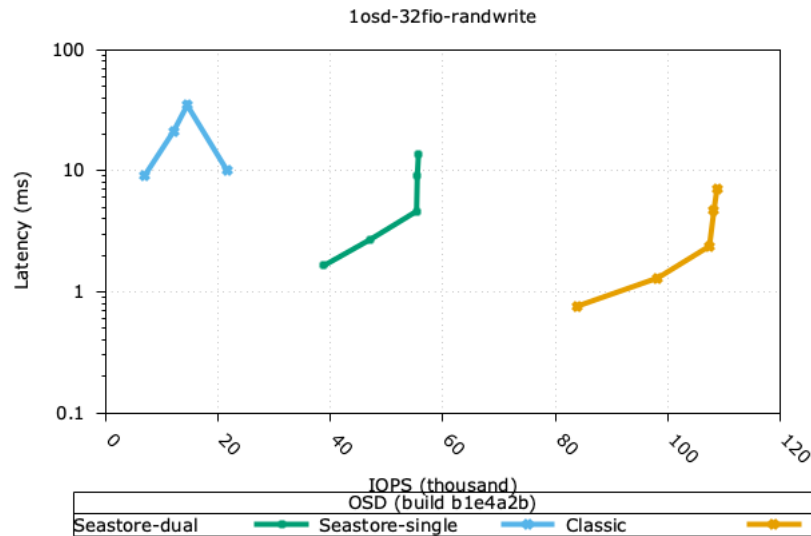
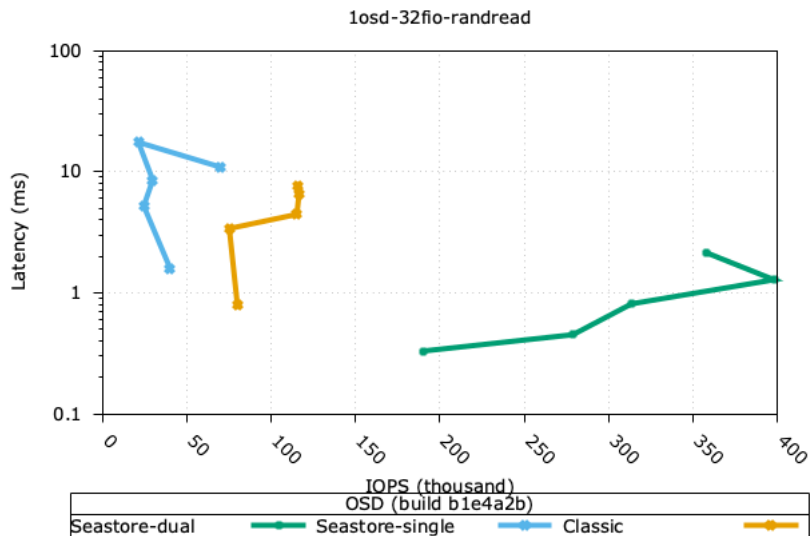
- **Kernel Bypass:** Direct communication with networking and storage devices that support polling, eliminating the overhead of kernel involvement.
- **Shared-Nothing Architecture:** Minimizes lock contention, ensuring smoother operation and higher performance.
- **Computational Efficiency:** Balances load across multi-core systems, enabling each core to handle the same volume of I/O with reduced CPU usage.
- CPU core pinning to reactor strategy

Crimson progress after 56 weeks



- **Robustness:** Over 100 pull requests have been merged
- **Coroutines:** (support in C++20) any new code added to Crimson is highly prioritised to be implemented with coroutines.
- **Seastar Reactor Options:** exposed some of these options as Ceph configurables
- **SeaStore optimisations:** conduct extensive testing, gain a deeper understanding of system behaviors, and prioritize addressing the most impactful problems
 - Partial Reads PR
 - Cache MD during cleanup
- **Periodic status reports:** for development and optimization purposes and are continuously evolving.

Crimson Performance



- Crimson performs better than Classic OSD for random read 4k
- Opportunity to improve for random write 4k



CPU core reactor pinning

CPU core reactor pinning



- Delivered PR [60822](#) to select the CPU reactor pinning strategy:
 - OSD Balanced
 - NUMA separated
- Caveat: only for Intel HT architectures, not tested on AMD SMT
- Detailed Ceph Performance Blog <https://ceph.io/en/news/blog/2025/crimson-balance-cpu-part1/>
- Simplified and improved by Sam Just in PR [63137](#)

CPU core reactor pinning



```
hboard --cyanstore --redirect-output --crimson --crimson-smp 3 --no-restart --crimson-balance-cpu osd
```

The following is the corresponding CPU distribution:

```
loading 3 .out files ...
```

```
== osd_0_cyan_3osd_3react_newbal_threads.out ==
```

```
== osd_1_cyan_3osd_3react_newbal_threads.out ==
```

```
== osd_2_cyan_3osd_3react_newbal_threads.out ==
```

```
----- Socket 0 ----- +
      0      1      2      3      4      5      6      +
+-----+-----+-----+-----+-----+-----+-----+
0 |  R.0 + R.0 + R.0 + R.2 + R.2 + R.2 + . | +
7 |  . + . + . + . + . + . + . | +
14 | . + . + . + . + . + . + . | +
21 | . + . + . + . + . + . + . | +
+-----+-----+-----+-----+-----+-----+-----+
56 | . + . + . + . + . + . + . | +
63 | . + . + . + . + . + . + . | +
70 | . + . + . + . + . + . + . | +
77 | . + . + . + . + . + . + . | +
+-----+-----+-----+-----+-----+-----+-----+

----- Socket 1 -----
      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+
28 |  R.1 + R.1 + R.1 + . + . + . + . |
35 | . + . + . + . + . + . + . |
42 | . + . + . + . + . + . + . |
49 | . + . + . + . + . + . + . |
+-----+-----+-----+-----+-----+-----+-----+
84 | . + . + . + . + . + . + . |
91 | . + . + . + . + . + . + . |
98 | . + . + . + . + . + . + . |
105 | . + . + . + . + . + . + . |
+-----+-----+-----+-----+-----+-----+-----+
```

```
(3 12) jinarez@lasec-Air: /tmp
```




IOP Cost estimation

IOP cost estimation for Crimson



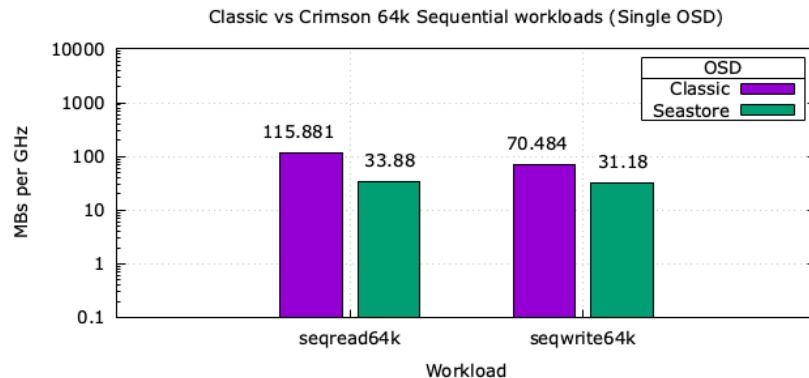
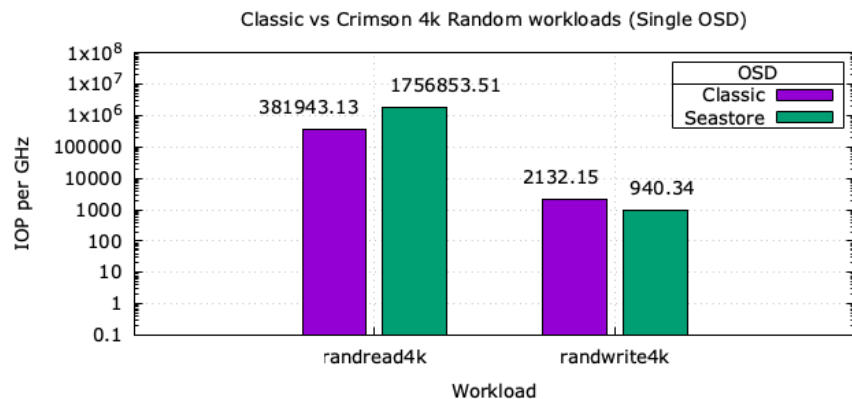
Query from Kyle Bader

- Crimson performance metrics revisited (equation showing the estimations)

$$\text{IOP per GHz} = \frac{IOPkHz}{(n \times 2.2GHz \times x_i)}$$

- Code (mostly Python) to collect, filter, transform and present (as Pandas data frames) the OSD reactor performance measurements
- Public project repo <https://github.com/perezjosibm/ceph-aprg/tree/main>
- Production of custom performance reports, discussions on the weekly Crimson call (charts and sections of the report)

IOP cost estimation for Crimson



These estimations are based on a dual-socket Intel(R) Xeon(R) Platinum 8276M CPU @ 2.20GHz (56 cores each socket)

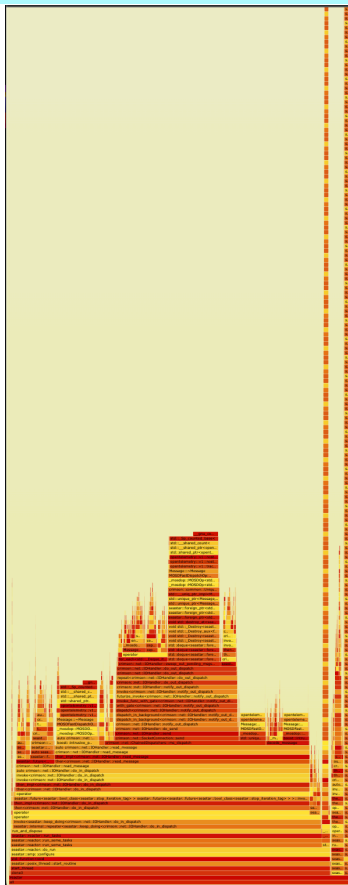


Crimson vs. Asynchronous messenger

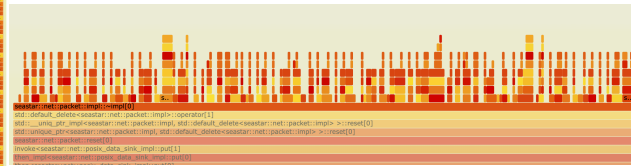
Crimson vs Classic (Asynchronous) Messenger



- Performance testing of the Messenger component of the OSD
- Implemented a custom test driver trigger script to execute the messenger benchmark (written by Yinxing Cheng as a standalone) to traverse over:
 - Number of clients, number of CPU cores
 - NUMA Separated vs Balanced CPU pinning
 - Monitor and collect measurements from Seastar
- Analysis based on flame graphs
 - Implemented a filter to deal with huge towers in the flame graphs
- Identifying bottlenecks and recommendations for optimisations on going



Tall towers occur in code path/stacks that involve lambda resolution of the asynchronous computation in Seastar (future/promises)

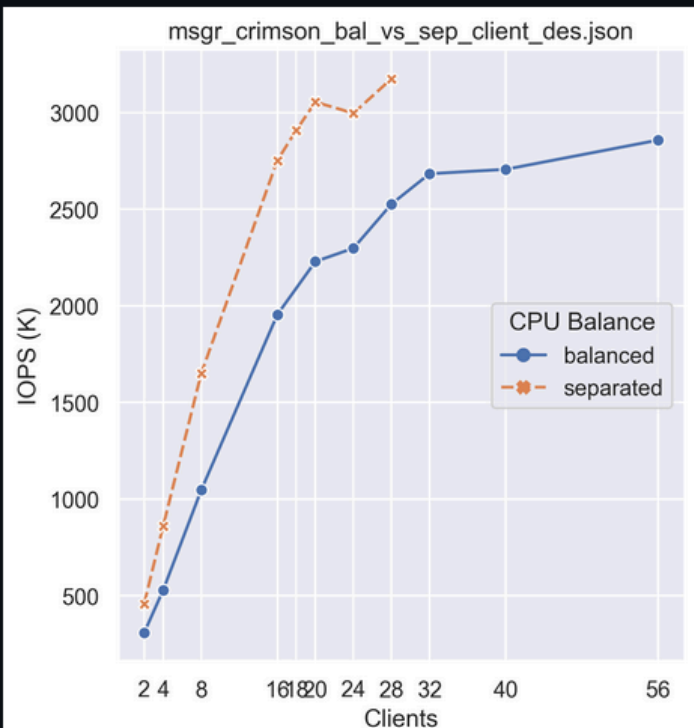


Essentially, each code stack is encoded as a *multiset* (item with number of occurrences)

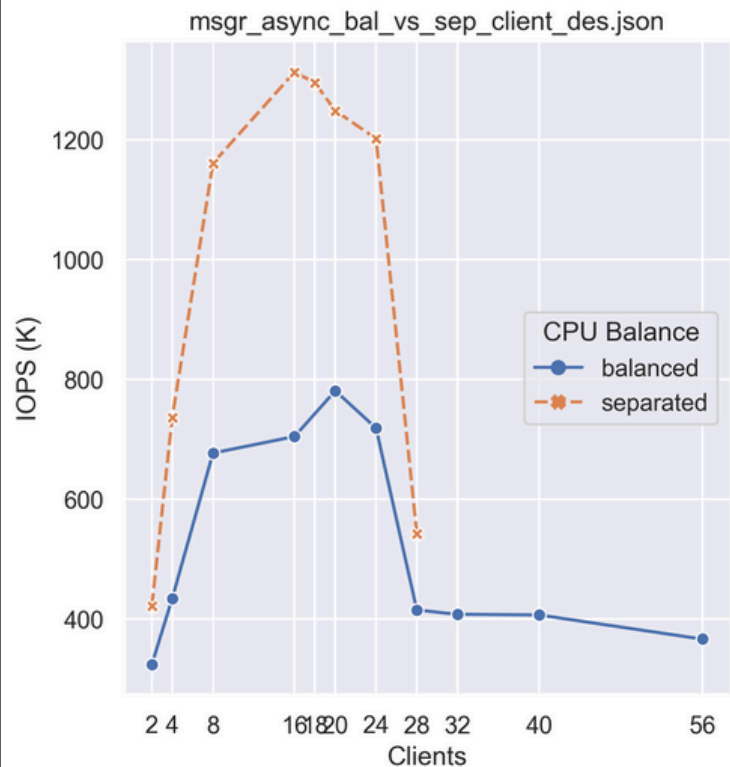
Crimson vs Classic (Asynchronous) Messenger



Crimson



Asynchronous





Code Contributions to the Ceph Benchmark Toolkit



- Defined a ‘workload’ section in the test plan .yaml schema for modular test execution (PR [306](#))
- Implemented an automated unit test generator to support regression testing when refactoring the exiting code (PR [320](#))
 - Based on the notion of object serialisation
- Extending the documentation (PR [316](#))
- Extending the tools in the Crimson folder
 - Collect, transform, filter data from measurements (monitoring) and present as Pandas data frames

	ceph/ceph.io Add blog entry Crimson balance CPU ✓
#797 by perezjosibm was merged on Feb 10 • Approved	
	ceph/cbt First version of the report generator. Add further tools. enhancement
#320 by perezjosibm was merged on Jan 28 • Approved	
	ceph/ceph [vstart]: add --crimson-balance-cpu option to set CPU distribution policy ✓
crimson crimson-perf documentation performance ready-to-merge	
#60822 by perezjosibm was merged on Feb 10 • Approved 6 of 14 tasks	
	ceph/cbt Update documentation. Minor reformatting on the fio-poarse-jsons.py
#316 by perezjosibm was merged on Oct 1, 2024 • Approved	
	ceph/cbt Integrate CPU utilisation in the response curve charts generated by f...
#312 by perezjosibm was merged on Aug 15, 2024 • Approved	
	ceph/cbt Add fio-parse-jsons.py tool to parse a set of json FIO files.
#307 by perezjosibm was merged on Jun 10, 2024 • Approved	
	ceph/ceph script/: add cpu-map.sh to aid manual selection of CPU cores for threads for profiling ✓ crimson script
#57562 by perezjosibm was merged on Aug 7, 2024 • Approved 4 of 14 tasks	
	ceph/ceph vstart.sh: add options to set number of alien threads, and number of cpu cores for alien threads ✗ crimson ready-to-merge
#57359 by perezjosibm was merged on Jun 17, 2024 • Approved 4 of 14 tasks	
	ceph/cbt Extend librbdfio.py with a new workloads block. Add minor fixes.
#306 by perezjosibm was merged on Jul 23, 2024 • Approved	



Future Ideas



- Performance analysis of coroutines in Seastar
- Port to Crimson the <https://github.com/taodd/cephtrace> project on dynamic latency monitoring (Dongdong Tao, Cephalocon 2023)
- Memory efficient atomic shared pointers: reduce contention, e.g. avoid shared mutable state via the "blue/green deployment" pattern (Arthur O'Dwyer - CppCon 2020)

Feedback from the Community



8 replies



Samuel Just 4 Mar



That is an **incredibly** useful analysis! Is bluestore restricted to the same cores as the reactor threads in these configurations?



Kyle Bader 10 Mar at 16:56

excellent, that was really fast 😊

THANK
YOU

1

