Eggstrain

Vectorized Push-Based inpired Execution Engine

Asynchronous Buffer Pool Manager

Authors: Connor, Sarvesh, Kyle

Original Proposed Goals

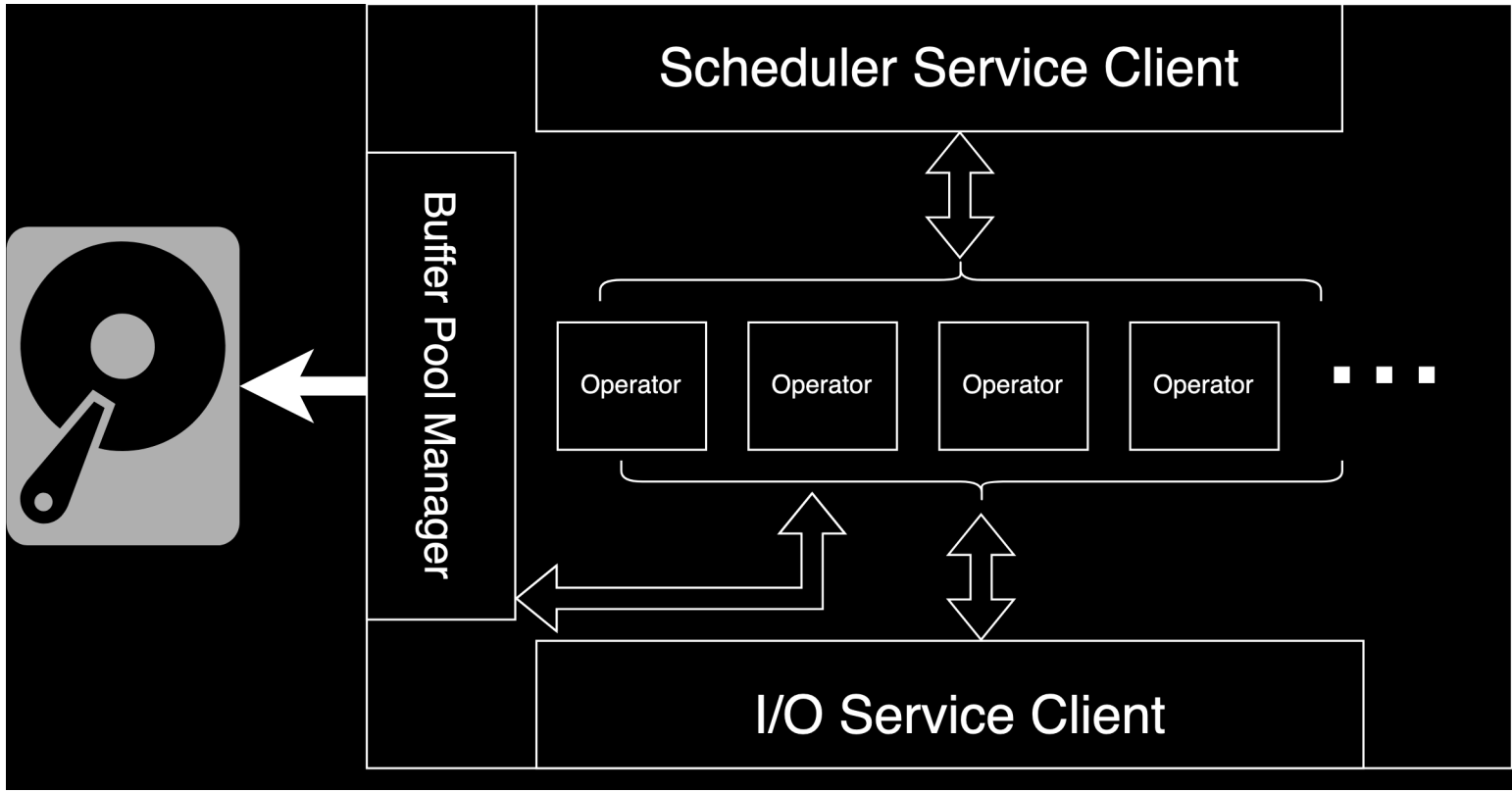
- 75%: First 7 operators working + integration with other components
- 100%: All operators listed above working
- 125%: TPC-H benchmark working

Design Goals

- Robustness
- Modularity
- Extensibility
- Forward Compatibility

We made heavy use of `tokio` and `rayon` in our implementation.

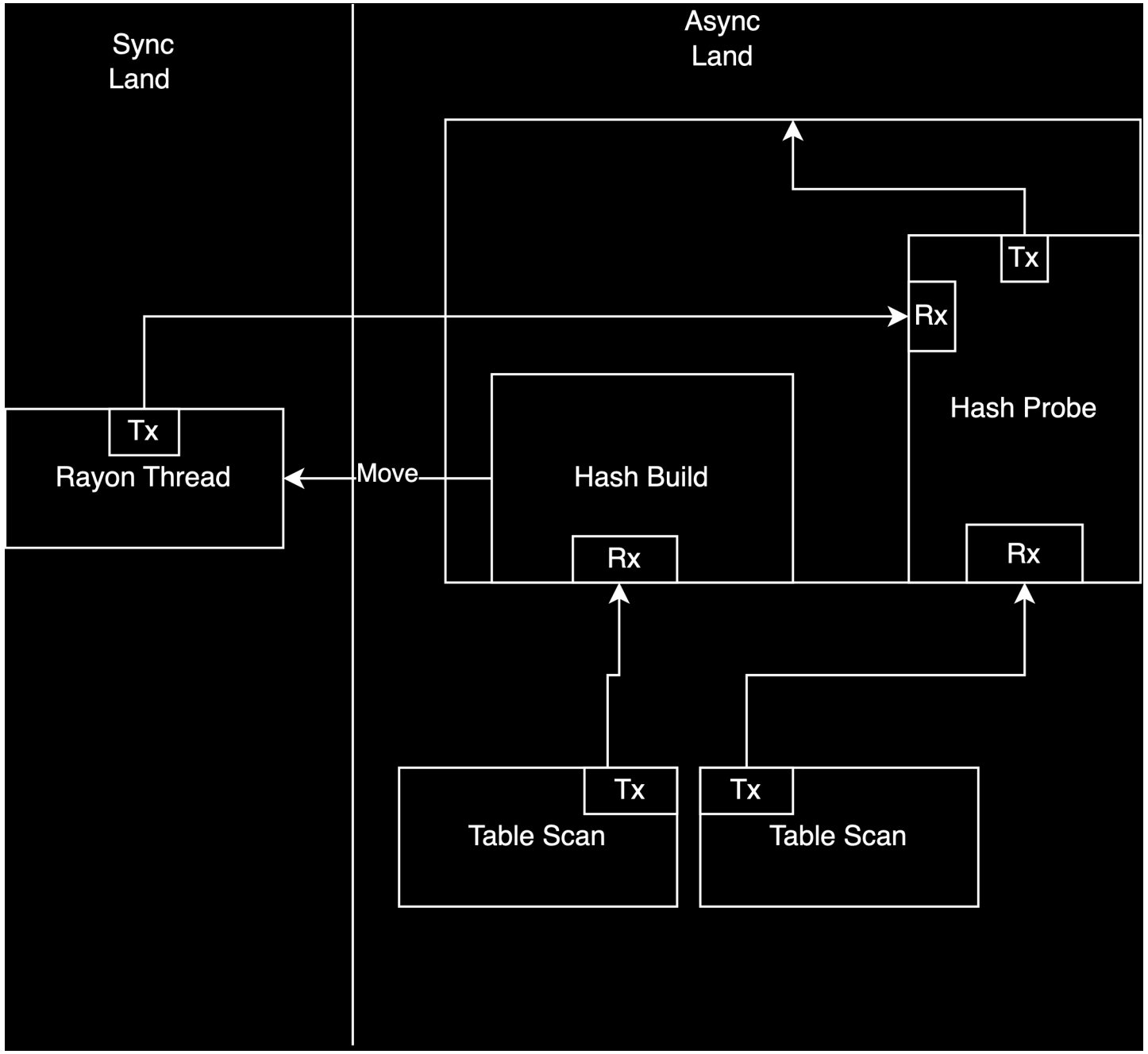
Refresher on Architecture



Refresher on operators

- `TableScan`
- `Filter`
- `Projection`
- `HashAggregation`
- `HashJoin` (`HashProbe` + `HashBuild`)
- `OrderBy`
- `TopN`

Example Operator Workflow



Progress Towards Goals

- 100%: All operators implemented, excluding `HashJoin`
- 125%: TPC-H benchmark working for Q1

Execution Engine Benchmarks

Hardware:

- Cray/Appro GB512X - 32 Threads Xeon E5-2670 @ 2.60GHz, 64 GiB DDR3 RAM, 1x 240GB SSD, Gigabit Ethernet, QLogic QDR Infiniband

TODO

Problem: In Memory? We need a buffer pool!

We found that we needed to spill data to disk to handle large queries. However, to take advantage of our asynchronous architecture, we needed to implement an asynchronous buffer pool manager.

Correctness Testing and Code Quality Assessment

We tested correctness by comparing our results to the results of the same queries run in DataFusion.

Our code quality is high with respect to documentation, integration tests, and code review.

However, we lack unit tests for each operator. We instead tested operators integrated inside of queries.

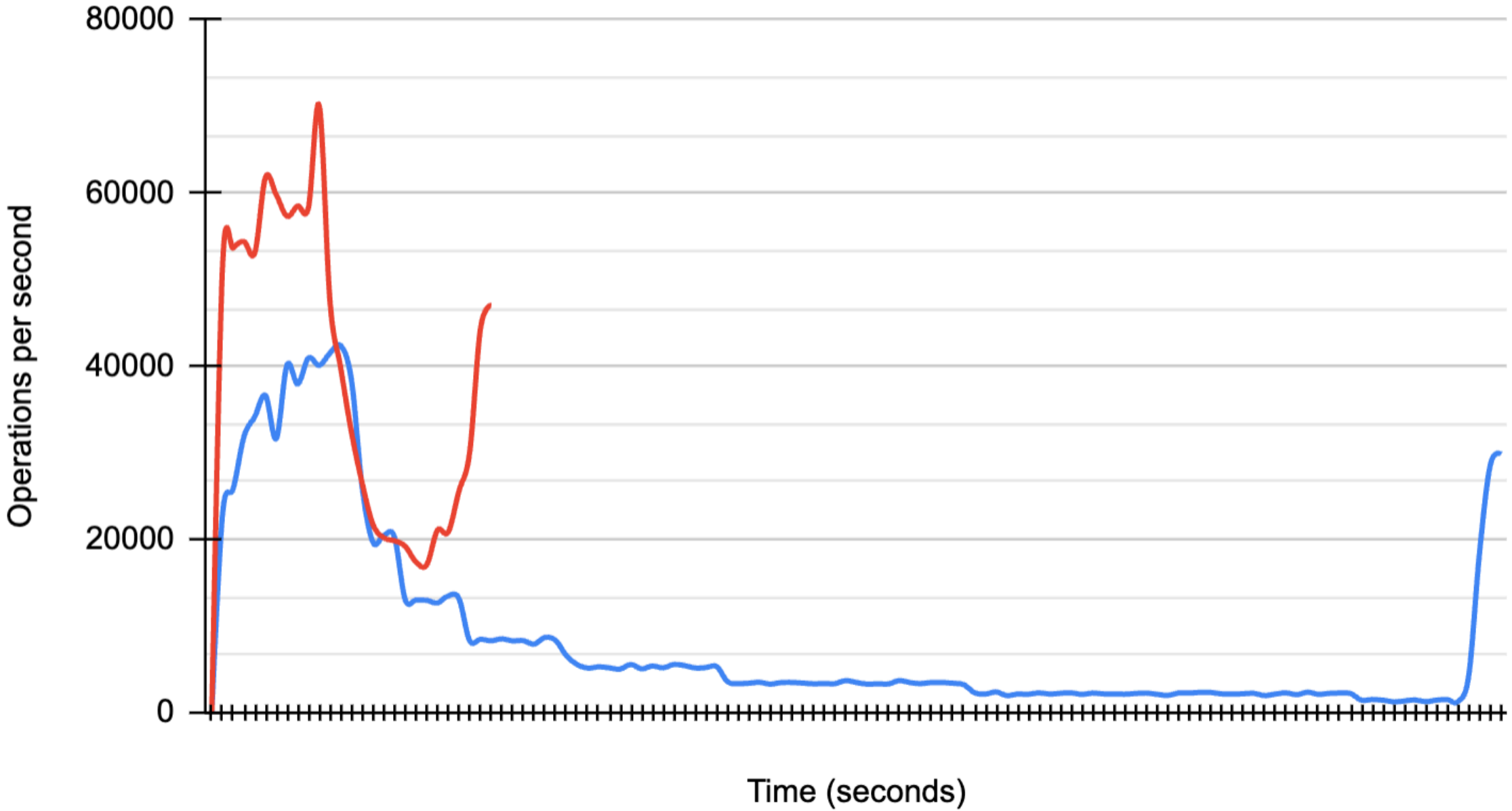
BPM Benchmarks

Hardware:

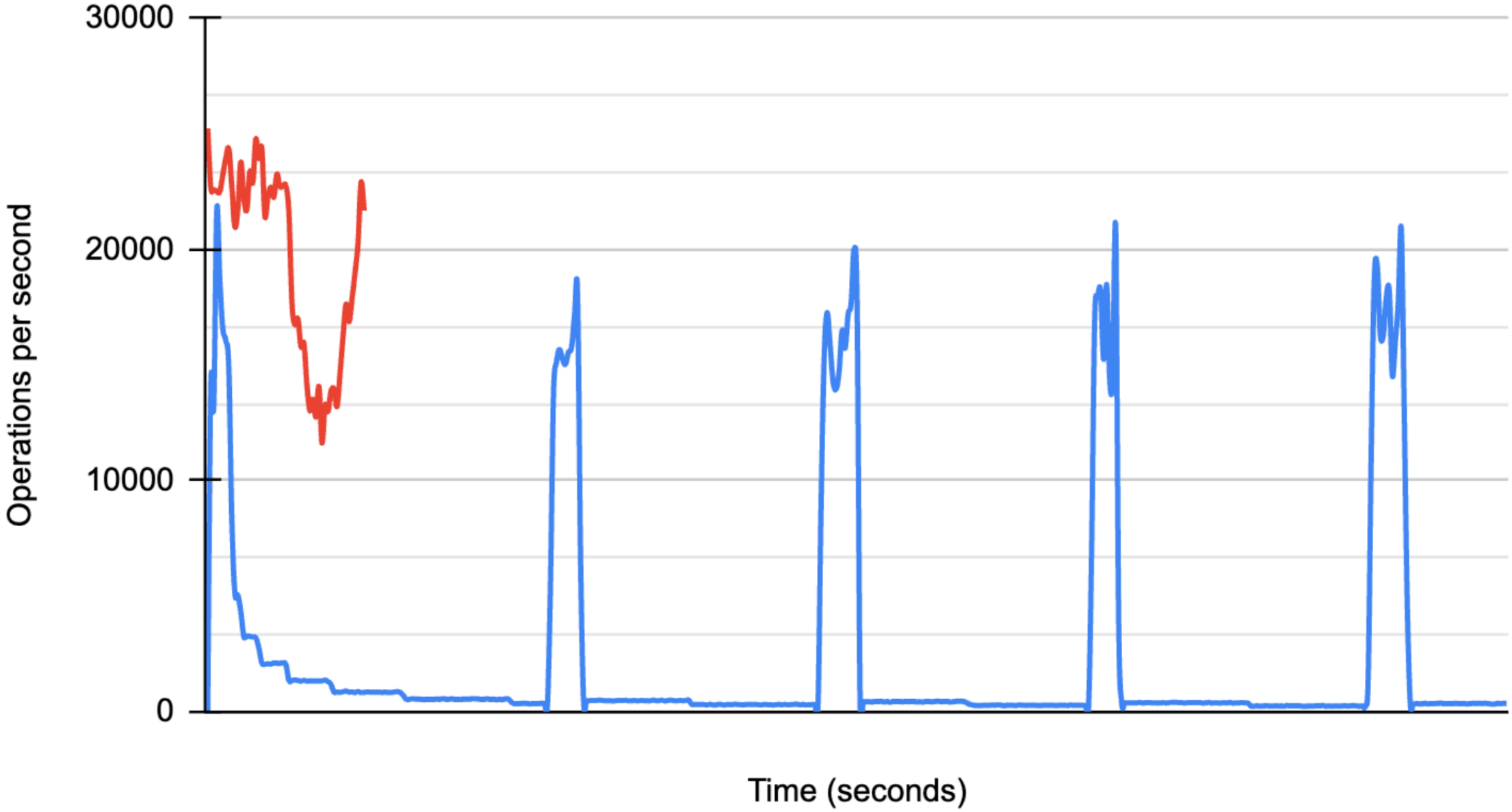
- Cray/Appro GB512X - 32 Threads Xeon E5-2670 @ 2.60GHz, 64 GiB DDR3 RAM, 1x 240GB SSD, Gigabit Ethernet, QLogic QDR Infiniband

We will benchmark against RocksDB as a buffer pool manager.

EggstrainBPM vs RocksDB on 20% write / 80% read



EggstrainBPM vs RocksDB on 80% write / 20% read



Future Work

TODO