Kernel Parameter tuning for Memcached

Anand Bonde

Introduction

Tuning low-level systems can have substantial impact on the key metrics for the workloads hosted on these systems. In this document, we can see a case study of the Memcached benchmark for P99 latency in microseconds. It clearly shows that adjusting the tuning knobs in the Linux Kernel scheduler can have huge impacts on the latency metrics.

Environment

- Host (GCR machine): Cores: 28, LPs: 56, Memory: 256 GB, Hyperthreading: Y, Sockets: 2, Minroot configuration (16 LPs).
- Guest (VM):, LPs: 8, Memory: 8 GB, Hyperthreading: Y, Numa Nodes: 1, Sockets: 1.
- Other considerations: No other workload running in parallel, Reuses existing Memcached scenario in ABS.

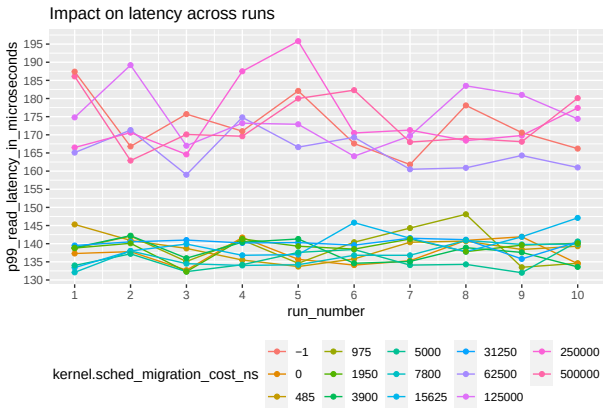
Benchmark Information

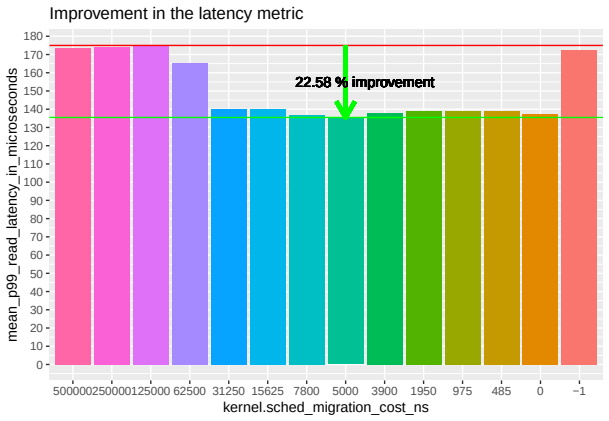
Memcached using the existing ABS scenario. It uses the fb_key dataset. Scenario run-time set to 1 second, so it allows for 1 complete run of the scenario. This scenarios runs 10 times, each time bringing up the VM from scratch to account cold/warm start differences. The average value of the benchmark metric for each of these runs is considered as the metric value. This allows us to factor in the variability in consequent benchmark runs.

Key Metric

Each ABS scenario/benchmark tracks a key metric for the workload under consideration. In case of Memcached, it is the P99 read latency in microseconds. Note that there variability in the existing benchmark metric results across several runs with the exact same configuration and query workloads.

Benchmarks and tuning results





Observations:

- Massive improvement (22.58%) in the metric just by tuning the kernel.sched_migration_cost_ns parameter.
- Variability reduces as latency gets better.
- Improvement is stable across multiple runs with the same configuration value.

- Improvement plateaus after reaching a lower limit.

Conclusions

The results of these benchmark experiments clearly show that tuning the low-level system parameters can have a big impact on the key workload metrics.

Next Steps

- Use the MLOS framework to find the globally optimal value for the tuning parameter.
- Associate telemetry data with impact on metrics and thus find correlations between parameters.
- Find optimize-able zones in the parameter space.
- Create performance profiles tuned for categories of workloads.
- Use code as a hint to find correlations and dependencies.