Rafiki: A Middleware for Parameter Tuning of NoSQL Datastores for Dynamic Metagenomics Workloads

Ashraf Mahgoub\textsuperscript{1,3}, Paul Wood\textsuperscript{2,3}, Sachandhan Ganesh\textsuperscript{1}, Subrata Mitra\textsuperscript{4}, Wolfgang Gerlach\textsuperscript{5}

Travis Harrison\textsuperscript{5}, Folker Meyer\textsuperscript{5,6}, Ananth Grama\textsuperscript{4}, Saurabh Bagchi\textsuperscript{2,3}, and Somali Chaterji\textsuperscript{1,2}

\textsuperscript{1}Department of Computer Science, Purdue University; \textsuperscript{2}Department of Electrical and Computer Engineering, Purdue University; \textsuperscript{3}Center for Resilient Infrastructures, Systems, and Processes, Purdue University; \textsuperscript{4}Adobe Research Labs; \textsuperscript{5}Argonne National Labs; \textsuperscript{6}Department of Medicine, University of Chicago.

Motivation

- DBMS have numerous configuration parameters
  - NoSQL Cassandra has 50+
  - NoSQL ScyllaDB has 25
- Configuration parameters control the system’s behavior
  - Parameter tuning is time-consuming for DBAs
  - Optimal configurations are workload dependent
  - Dynamic workloads
    - MG-RAST: Metagenomic Rapid Annotations using Subsystems Technology
      - automatic phylogenetic and functional analysis of metagenomes
      - is one of the biggest repositories for metagenomic data.
  - Wikipedia workloads

NoSQL Datastores (Cassandra)

- Non-relational (flexible design)
- Distributed (fault tolerant)
- Horizontally scalable (performance scales with [# of instances])
- 25+ configuration parameters
- Interdependent parameters (one-by-one tuning provides sub-optimal performance)

Challenges

- Configurations space is huge.
- Searching in runtime is non-practical
- A fast and efficient approach is needed to adapt with sudden workload shifts

Rafiki reaches within 15% of grid search performance using only 1/10000-th of the search time

Phase 1: Varying workloads and configurations are applied to the NoSQL datastore to identify the key configuration parameters and to generate training data for a surrogate model.

Phase 2: A prediction model is used to identify the relation of configurations and workloads to performance.

Phase 3: A search strategy is applied for a given workload to find close-to-optimal configurations.

A high-level overview of Rafiki searching through the configuration parameters’ space of NoSQL datastores to achieve close-to-optimal performance. Rafiki is agile enough to quickly adapt to changing workloads, such as in the MG-RAST system.

Triumphs of Rafiki

- We design and develop Rafiki for automatically configuring NoSQL datastore parameters in a workload-centric manner, using traces from MG-RAST.
- We apply ANOVA-based analysis to identify the key parameters that are the most impactful toward improving datastore throughput.
- We create a DNN framework (surrogate model) to predict the performance for unseen configurations and workloads. It achieves a performance prediction with an error in the range of 5-7% for Cassandra and 7-8% for ScyllaDB.
- We then create a Genetic Algorithm-based search process through the configuration parameter space, which improves the throughput for Cassandra by 41.4% for read-heavy workloads (more relevant to MG-RAST), and 30% on average.
- To get an estimate of the upper bound of improvement, we compare Rafiki to an exhaustive search process and see that Rafiki, using 4 orders of magnitude lower search time than exhaustive grid search, reaches within 15% and 9.5% of the theoretically best achievable performances for Cassandra and ScyllaDB, respectively.

References


Acknowledgments

This work is supported in part by the U.S. Department of Energy, Office of Science under contract DE-AC02-06CH11357. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

https://engineering.purdue.edu/dcsl/