Rafiki: A Middleware for Parameter Tuning of NoSQL Data-stores for Dynamic Metagenomics Workloads

Ashraf Mahgoub, Paul Wood, Sachandhan Ganesh
Purdue University

Subrata Mitra
Adobe Research

Wolfgang Gerlach, Travis Harrison, Folker Meyer
Argonne National Laboratory

Ananth Grama, Saurabh Bagchi, Somali Chaterji
Purdue University

Dependable Computing Systems Lab (DCSL)
School of Electrical and Computer Engineering
Purdue University

Work supported by:
NIH
Adobe Research

Agenda

- Database Parameters Tuning.
- Cassandra Internal Read/Write path.
- Tuning challenges.
- Workflow and evaluation.
- Related work.
- Conclusion

Database Parameters Tuning

- Database systems have numerous configuration parameters which control the internal behavior of the system.
- DBAs often spend a significant portion of their time to tune these parameters to get the best performance.
  - A survey in 2013 show that **40%** of engagement requests for a large Postgres service company were for DBMS tuning and knob configuration issues.

Challenges

- **Search space size:** Cassandra YAML file has **50+** configuration parameters to be specified by the user. 25 of them are marked as **performance related.**
- **Dynamic Workloads:** Workloads can change very frequently (every few minutes).
- **Performance tradeoff:** Tuning for read-heavy workloads makes it bad for write-heavy ones and vice-versa.

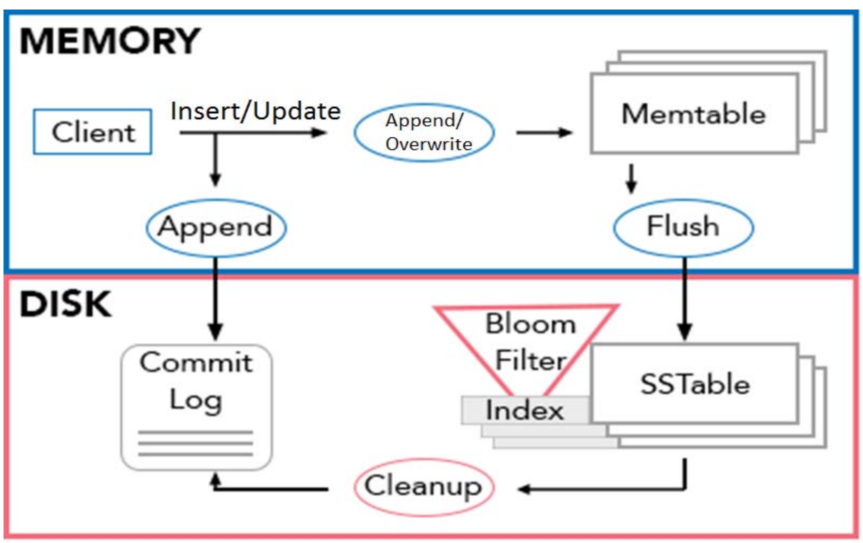
➢ **Is it possible to automatically find the best configurations for a given workload? Can we do it fast enough to adapt to highly dynamic workloads?**

NoSQL Data-stores

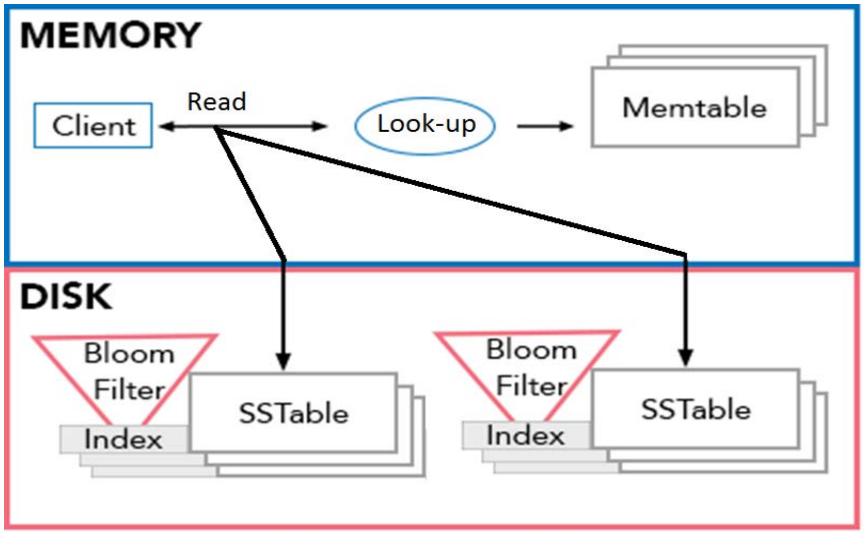
- Motivation: HPC applications need to store and analyze huge volumes of semi-structured data.

- **Apache Cassandra** has the following features:
  - **Non-relational** (simpler design)
  - **Distributed** (Fault tolerant)
  - **Horizontally scalable** (performance increases linearly with number of instances)
  - **Popular** (2nd most popular NoSQL datastore, DB-Engines Nov 2017)
  - **Prominent users:** Facebook, Apple, Twitter, Netflix

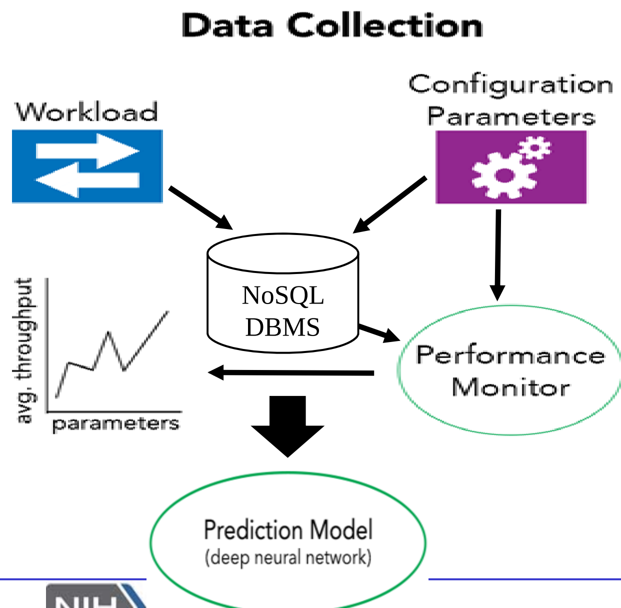
Cassandra Write Path



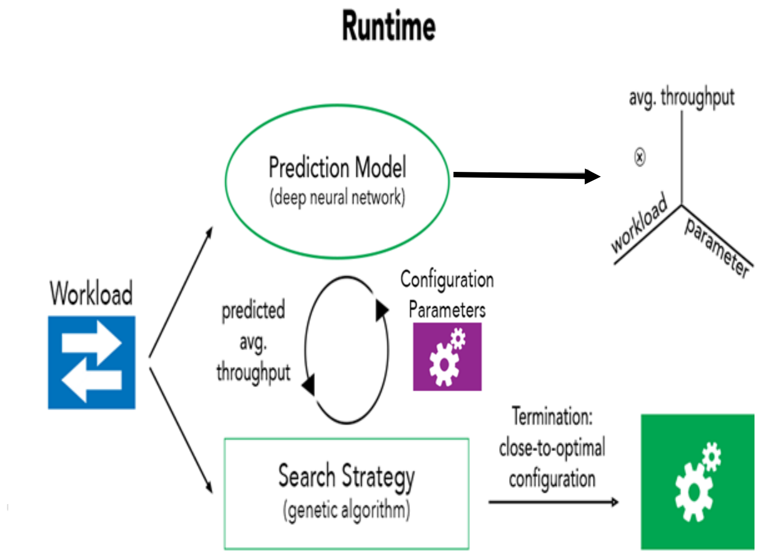
Cassandra Read Path



Rafiki: Offline Phase



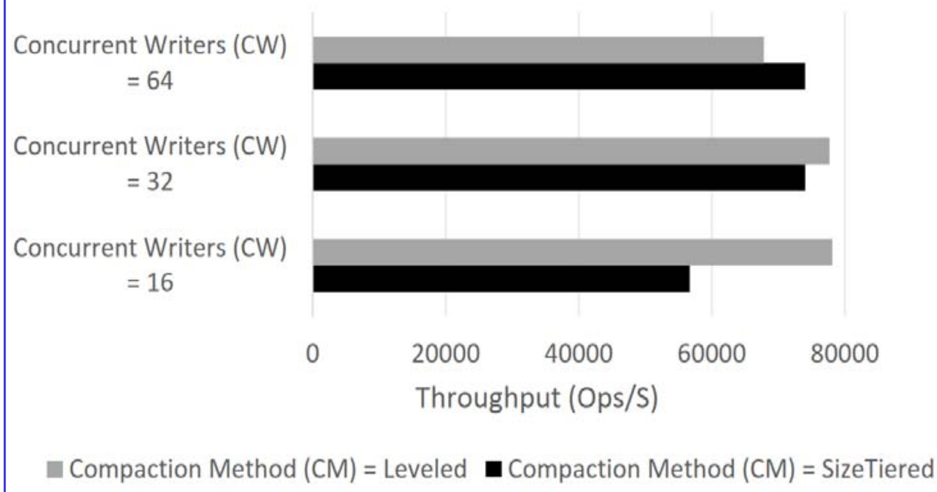
Rafiki: Online Phase



Challenges: Search Space Size

- **CPU Related**
  - **Concurrent_reads (7)**
  - **Concurrent_writes (7)**
  - **Concurrent_compactors (7)**
  - **Memtable_flush_writers (7)**
- **Memory Related**
  - **Memtable_space (mb) (4)**
  - **Row cache size (4)**
  - **Key cache size (4)**
- **Disk Related**
  - **Compaction_throughput (mb/sec) (5)**
  - **Memtable_cleanup_threshold (4)**
  - **Compaction_Method (2)**
- **Amount of data needed**
  - 7^4 * 4^4 *10 * 10 = 6,146,560 data points to be collected, takes around 600 years to collect (in a 5 min setup).

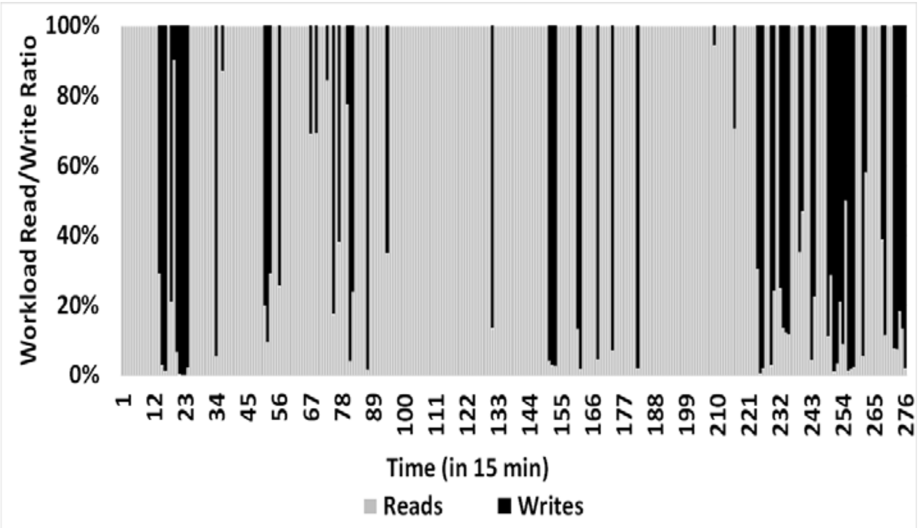
Challenges: Interdependence between parameters



Challenges: Dynamic Workloads

- Use case: MG-RAST
  - The most popular metagenomics portal and analysis pipeline
  - Allows users to upload metagenomes for automated analysis
  - 35 GB of data being uploaded to its database on a daily basis
  - We analyzed 60 days of traces from MG-RAST to capture its workload characterization.
- Workload Characterization:
  - Read/Write ratio (changes sharply over time).
  - Key-reuse distance
  - Record size.

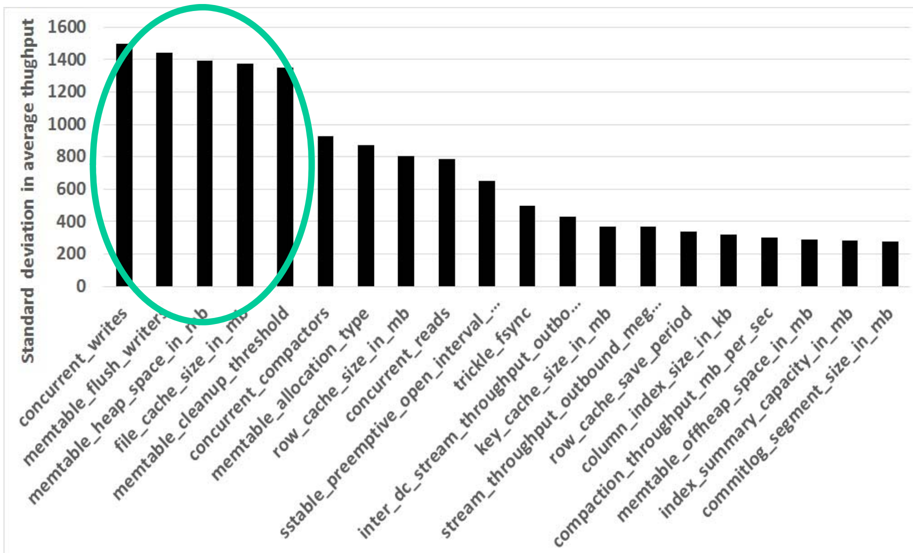
Read/Write ratios (averaged every 15 min)



Rafiki Solution: Impactful Parameter Identification

1. Search space reduction by identifying impactful parameters:
    1. Intuition: Sort parameters by their effectiveness to performance so that least-effective parameters can be pruned.
    2. Method: Change values of each parameter (one-by-one), and use ANOVA to numerically evaluate the parameter to performance.
    3. Highly effective parameters are selected for the next step (Data collection).

Ranking Impactful Parameters



Prediction Model Training

2. Collect data points enough to train a sufficiently accurate prediction model.

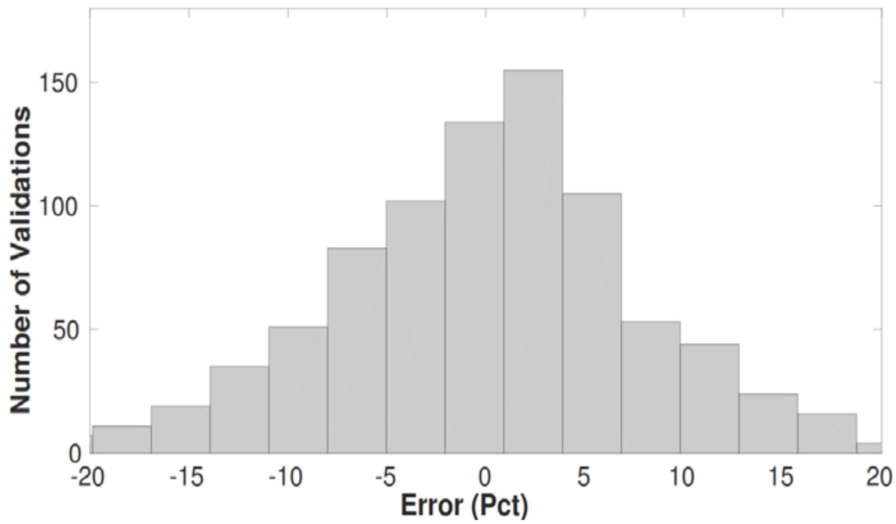
1. The target is to predict the performance given the workload (W) and values for selected impactful parameters (C).

$$AOPS_{Cassandra} = f_{\mathrm{net}}(\mathrm{W}, \mathrm{C})$$

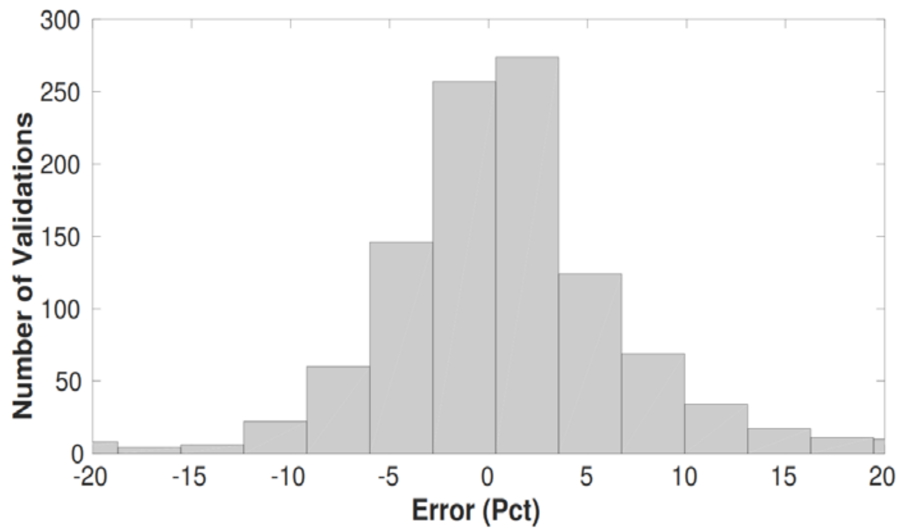
2. Evaluation: prediction accuracy of the model for two tasks:

- Predicting performance for unseen workloads (Read ratios not included in training data).
- Predicting performance for unseen configurations (Values for selected parameter not included in training data).

Prediction Evaluation: Unseen Configurations



Prediction Evaluation: Unseen Workloads



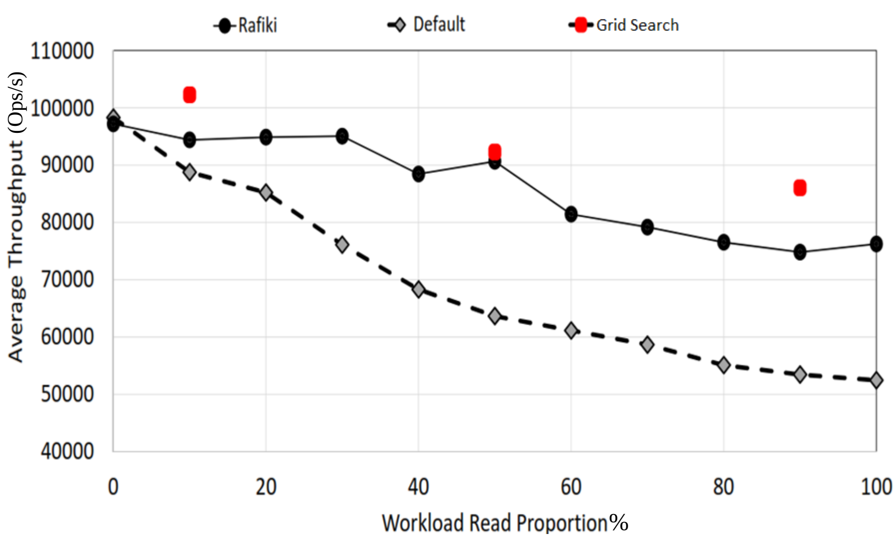
Finding Optimal Configurations (Runtime)

- At this point we have a prediction model that serves as a surrogate model for Cassandra.
- Now we can apply a search technique (GA) with the surrogate model to quickly find best configurations.

$$C_{opt} = \arg\max_{C} f_{net}(W, C)$$

- GA chromosome → Values of the selected parameters.
- GA fitness function → Target performance metric (Throughput for us).

Throughput: Rafiki vs. Default vs. Grid search



Searching Time Reduction

- Without the surrogate model, testing a single configuration file takes 5-7 min.
  - Initial data loading (2 min).
  - Replay MG-RAST traces (3-5 min).

- Rafiki combines GA & trained surrogate model to test over 17K combinations/sec in the configuration search space, reaching convergence in 20-22 seconds.

- **Rafiki** can suggest configurations that are within 15% of best configurations found by grid searching, using only 1/10000-th of the searching time.

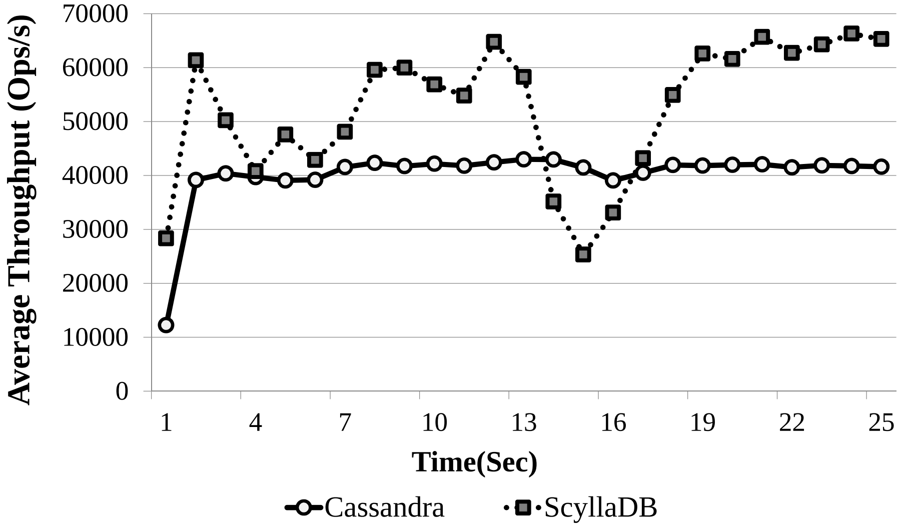
Tuning ScyllaDB's Performance

- ScyllaDB:
  - Based on Cassandra architecture.
  - Provides a user-transparent auto-tuning system internal to its operation.
  - High variance in throughput even for constant workloads.

	WL1(R=70%)		WL2(R=100%)	
Opt. Technique	Rafiki	Grid	Rafiki	Grid
Avg Throughput	69,411	75,351	66,503	63,595
Gain over Default	12.29%	21.8%	9%	4.57%

  - For other workloads, the improvement is not significant (some times less than default performance by 2-3%)

ScyllaDB vs. Cassandra Stability



Related Work

- Tuning tools created by vendors, only support particular company's DBMS (Dias *et al.*, CIdR'05 & Narayanan *et al.*, MASCOTS'03).
- Other tools require intervention of DBAs to identify important parameters or guide the searching process (Sullivan, *et al.*, SIGMETRICS'08).
- Ottertune (Aken *et al.*, Sigmod'17) and iTuned (Duan *et al.*, VLDB'09)
  - Rely on nearest-neighbor mapping with previously collected data points.
  - Ottertune takes 30-45 min to start suggesting a better configuration, whereas iTuned takes 60–120 min.

Conclusion

- We proposed Rafiki: a system for automatic tuning of NoSQL data-stores (Casandra, ScyllaDB) under dynamic workloads (MG-RAST)
- NoSQL data-stores' configuration space is huge, Rafiki selects impactful parameters.
- Rafiki trains a prediction model that serves as a surrogate model for the actual data-store allowing for efficient searching of large space.
- Rafiki provides close-to-optimal configuration parameters for highly dynamic DB workloads.

Questions ?


Thank you