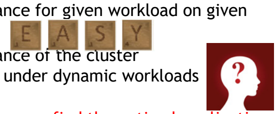
Hestia: Performance and Cost Optimization for NoSQL Distributed Databases in the Cloud

Ashraf Mahgoub[1], Paul Wood[2] , Subrata Mitra[3], Somali Chaterji[2] , Saurabh Bagchi[1,2]

[1] Department of Computer Science, Purdue University; [2]School of Electrical and Computer Engineering, Purdue University; [3]Adobe Research

Motivation

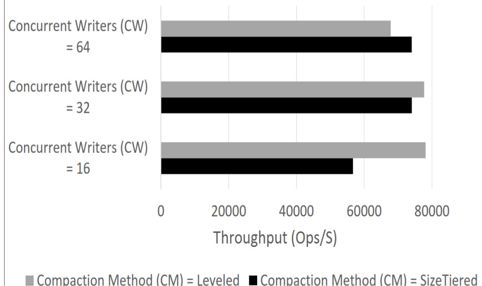
- NoSQL DBMS have numerous configuration parameters
  - Apache Cassandra has 50+ parameters
  - Redis has 40+ parameters
- Configuration parameters control the system's behavior
- Parameter tuning is time-consuming for DBAs
- Optimal configurations are workload dependent
- Dynamic workloads: **MG-RAST (Metagenomics portal), Tiramisu (Bus-Tracking mobile application).**
- Cloud services provide many configurations for the type, and size of the VMs, which control the compute capacity, RAM, and network bandwidth
- Estimate the performance for given workload on given cloud instance type
- Estimate the performance of the cluster (Throughput/Latency) under dynamic workloads



For a NoSQL database, can we find the optimal application and cloud configurations that achieves best performance under a cost bound?

Apache Cassandra

- Popular NoSQL DB
- Distributed (fault tolerant)
- Horizontally scalable (performance scales with # of instances)
- 50+ performance related configuration parameters
- Interdependent parameters (one-by-one tuning provides sub-optimal performance)



Redis

- Stores data in key, value pairs
- In-memory, no tables, schema, or collections
- 40+ performance related configuration parameters
- Data has to fit in memory, disk is only used for snapshots and fault-tolerance purposes
- If dataset size exceeds what can fit in RAM, Redis either stops accepting write request (Default) or starts evicting least-recently-used rows
- Selecting the appropriate VM type and size is very important to achieve data consistency and **durability**

Challenges

- Application and Cloud configurations space is huge
- Exhaustive searching at runtime is impractical
- Agile approach is needed to adapt to workload shifts
- Many systems can provide performance prediction for a single server (Such as Rafiki[1] or Ottertune[2])
- Predicting the overall cluster performance is challenging as it also depends on other parameters such as Replication Factor (RF) and Consistency Level (CL)
- Prediction models trained on a particular infrastructure (e.g. VM type and size) perform very poorly when the infrastructure changes
- Need to transfer knowledge across workloads and across infrastructures
- Can we predict the performance of a heterogeneous cluster? (i.e., nodes with different application/cloud configurations)
- Reconfigurations have a *cost*:
  - Changing application configurations may require application restart
  - Changing VMs incurs a downtime

Grid Search Limitations

Assume we want to tune the application configurations only:

CPU-Related Parameters

- *Concurrent_reads* (7 values)
- *Concurrent_writes* (7 values)
- *Concurrent_compactors* (7 values)
- *Memtable_flush_writers* (7 values)

Memory-Related

- *Memtable_space* (mb) (4 values)
- *Row_cache_size* (4 values)
- *Key_cache_size* (4 values)

Disk-Related

- *Compaction_throughput* (mb/sec) (5 values)
- *Memtable_cleanup_threshold* (4 values)
- *Compaction_Method* (2 values)

Amount of data needed

- 7*4 * 4^4 * 5 * 2 * 10 = 6,146,560 data points
- Takes 600 years to collect (for a 5 min benchmark run with each setting)

Solution Approach

- Automated Impactful Parameters Identification (D-Optimal Experimental Design)
- Surrogate models training (Offline)

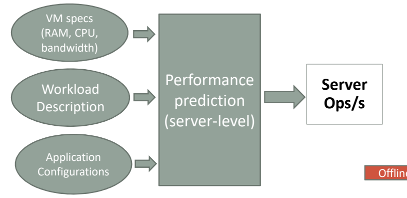
$$Perf_{single} = f_{pred}(WL, AppConf, CloudConf)$$

$$Perf_{cluster} = f_{pred}(WL, RF, CL, Perf_{single1}, \dots, Perf_{singleN})$$

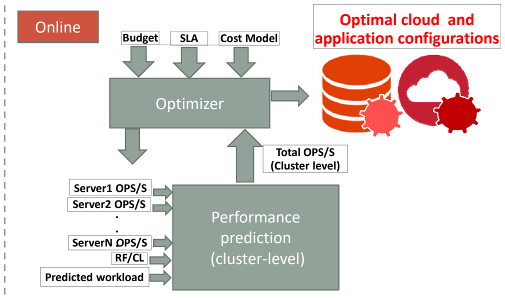
- At runtime, search configurations space for optimal configurations for the current workload
- Generate a reconfiguration plan that maximizes predicted benefit and minimizes reconfiguration cost

Design Overview

A workload predictor is trained with historical traces from the application under tuning. It provides Hestia with the future pattern of the workload, which is essential to apply reconfigurations in a timely manner

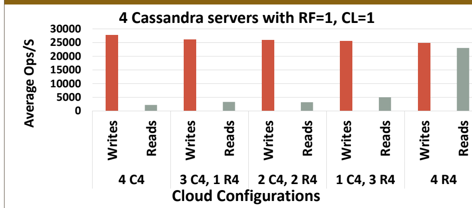


Also in the offline phase, a single server performance predictor is trained to map workload description, VM specs and app configurations to Ops/s

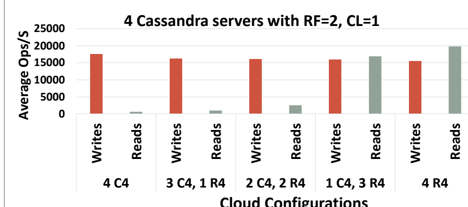


A cluster-level performance predictor (trained offline) is used to estimate the throughput of the whole cluster. In the online phase, our optimizer uses this predictor to evaluate the fitness of different cloud/application configurations and provides the best performance within budgeted cost

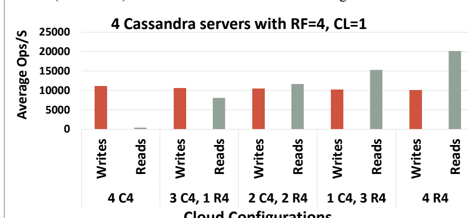
Motivating Results



- We vary the number of C4 (Compute Optimized) to R4 (Memory Optimized) VMs in a 4-servers Cassandra cluster. We start by 4 C4 VMs and notice that with RF=1 and CL=1, the performance of the reads is very poor and stays poor till all VMs are reconfigured to R4.



- Now with RF=2, we notice that the cluster can tolerate having one instance to be of type C4 and still achieves good performance for reads. This is because now (with RF=2) all data records are accessible through the 3 R4 servers



- Similarly, if RF=4, every server has a copy of all records in the cluster and now reconfiguration of only one server's type to R4 improves the performance for reads by 21X

Related Work

- Majority of existing tuning tools were created by vendors to only support their particular company's DBMS (Dias *et al.* 2005 & S. Kumar 2003)
- Other systems require more intervention of DBAs to identify important parameters ior guide the searching process (Tran, Dinh Nguyen, *et al.* 2008)
- Ottertune (Aken-SIGMOD17) and iTuned (Duan-VLDB09): Uses nearest-neighbor interpolation between previously collected data points. Ottertune [2] takes 30-45 min to start suggesting a better configuration, whereas iTuned [3] takes 60–120 min
- Rafiki (Mahgoub-Middleware17): Reduces the searching time significantly by training the surrogate model offline, which it then queries to find the best configurations for a new workload. However, it lacks two fundamental features: cluster-level prediction, and the capability of knowledge transfer across different architectures
- Cherrypick (Alipourfard-NSDI17): Tunes cloud configurations (VMs types and cluster size) for big data analytics. However, only homogenous clusters are supported and no cost of reconfiguration is considered

Acknowledgements

This work is supported in part by NSF grant 1527262, NIH Grant 1R01AI123037, and a gift from Adobe Research. Argonne National Laboratory's work was supported 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.

References

[1] Mahgoub, Ashraf, et al. "Rafiki: a middleware for parameter tuning of NoSQL datastores for dynamic metagenomics workloads." Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference. ACM, 2017.

[2] Van Aken, Dana, et al. "Automatic database management system tuning through large-scale machine learning." *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017.

[3] Duan, Songyun, Vamsidhar Thummala, and Shivnath Babu. "Tuning database configuration parameters with iTuned." *Proceedings of the VLDB Endowment* 2.1 (2009): 1246-1257.

[4] Alipourfard, Omid, et al. "CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics." *NSDI*. Vol. 2. 2017.