YCSB-D: Benchmarking Adaptive Frameworks

Subhajit Sidhanta*, Supratik Mukhopadhyay*, and Wojciech Golab†
*Louisiana State University, Baton Rouge, Louisiana, USA, Email: {ssidha1, supratik}@csc.lsu.edu
†University of Waterloo, Waterloo, Ontario, Canada, Email: wgolab@uwaterloo.ca

***Abstract*—We demonstrate YCSB-D, a tool that builds upon YCSB (Yahoo Cloud Serving Benchmark suite) to assist users in simulating dynamic variations in workloads, YCSB-D automatically varies the parameters in YCSB workloads over time according to user-specified time series functions, without requiring users to manually change the workload configuration in individual nodes each time the workload parameters needs to be modified. The dynamic workload variations simulated with YCSB-D can be used to evaluate the adaptability of such frameworks to changing workload characteristics. We demonstrate the ability of YCSB-D to evaluate the adaptability of OptCon, an automated framework, that tunes the consistency settings of Cassandra with respect to the latency and staleness thresholds in an SLA.**

I. Introduction

YCSB [1] is widely used to simulate standard benchmark workloads for cloud-based systems, like Cassandra, MongoDB, Hbase, etc. The YCSB suite comprises a workload generator, that can simulate a set of core workload types, characterized by parameters, like target throughput, thread count, etc., emulating different types of real world workload scenarios. To simulate scenarios where concurrent applications access the datastore from multiple nodes, YCSB can be set up to run as a collection of parallel client processes, where each node in a cluster runs a separate YCSB process. For executing YCSB in parallel mode, a user must manually start the YCSB workload executor from the command line at each individual node of the cluster. Workloads in real world web-based applications (like Netflix, Youtube, Facebook, Amazon, BitTorrent, etc.), that use distributed datastores, widely vary over time [4]. Workload variation for such systems typically exhibit a specific pattern; the workload parameters (like target throughput and thread count) vary in a characteristic fashion over time. Such workload variation patterns can be represented as time series functions. For example, Netflix [4] observes that the network traffic for its applications reaches almost 37% of Internet traffic during peak workload hours. For benchmarking such systems, a user must execute benchmark workloads for the target system in an uninterrupted back-to-back *sequence*, following the characteristic workload variation pattern specific to the use case. We present YCSB-D, which extends YCSB, allowing users to specify intended variations in YCSB workload parameters, in the form of time series functions (refer to Figures 1(a), 1(b), and 1(c)). YCSB-D automatically varies the YCSB parameters and simulates real world workload variation patterns on the target datastore. YCSB-D can benchmark adaptive frameworks, like OptCon [3], and verify their adaptability to dynamic workload variations.

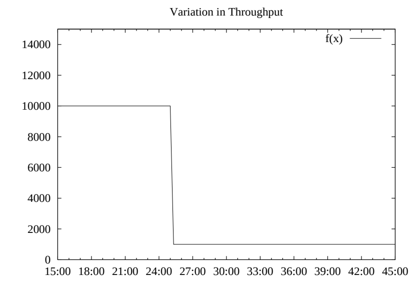
II. Implementation:

YCSB-D implementation comprises two artifacts. 1) A Java-based web application that comprises a web console that collects the input parameters from the users, and calls the YCSB-D client. 2) A modified version of the YCSB framework that alters the source code of the YCSB client to accept a workload variation pattern. 3) A Workload Generator, that can execute core YCSB workloads, varying the target throughput according to a variation pattern, specified by the user in form of a time series function $T = f(t)$, where T is the target throughput in op/s, and t is time (in ms) (refer to Figures 1(a), 1(b), and 1(c)). From the web console, the user can specify the time series function for varying the target throughput. The Result Visualizer module comprises a web page that displays the observed latency and throughput for the benchmark operations in the form of a time series. The Result Visualizer also displays the observed consistency or staleness (i.e., how stale (old) is the version of the data item (observed by a client application) with respect to the most recent version), computed in terms of the $\mathbf{\Gamma}$ metric of Golab et. al. [2]. As demonstrated in [2], $\mathbf{\Gamma}$ is preferred over other client-centric staleness measures for its proven sensitivity to workload parameters.

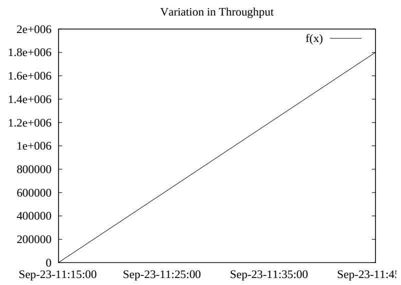
III. Demonstration

To automate the process of consistency tuning in distributed datastores, a number of prototype adaptive tuning frameworks have been developed, like TACT, Pileus, Tuba, Quela, OptCon [3], etc. Such frameworks can adapt the underlying datastore to variations in the workload, thus ensuring that the system always satisfies the SLA. However, testing such capabilities require the framework to be subjected to an exhaustive set of possible variations in the workload. We propose that the adaptability of such adaptive frameworks can be evaluated with dynamic variations in YCSB benchmark workload, simulated using the YCSB-D tool. YCSB-D can be used to simulate real world scenarios, characterized by specific patterns of workload variations (Figures 1(a), 1(b), and 1(c)). Thus, YCSB-D can be used to test if an adaptive framework can adjust the underlying datastore to satisfy the SLA under varying workloads; hence, YCSB-D can be used to benchmark the above mentioned adaptive frameworks. We quantify the adaptability of adaptive frameworks by the metric M, which computes the percentage of operations which did not violate the SLA.

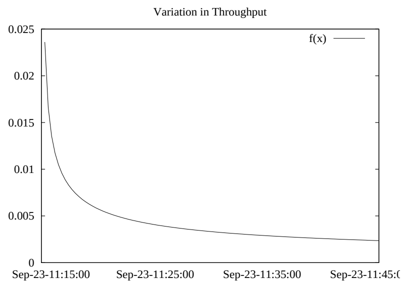
Demonstration Setup: OptCon [3] is an open-source example of an automated adaptive framework, that tunes Cassandra with respect to the latency and staleness thresholds in the SLA. We use YCSB-D to evaluate the adaptability of OptCon to dynamically varying YCSB workloads. The results obtained by running dynamically varying workloads (simulated with YCSB-D) on OptCon can be quantified in terms of the M-metric, that measures the adaptability of the framework (i.e., OptCon). YCSB-D is built as an extension of YCSB, which is a widely accepted benchmark for cloud-based systems



(a) With a Step Function

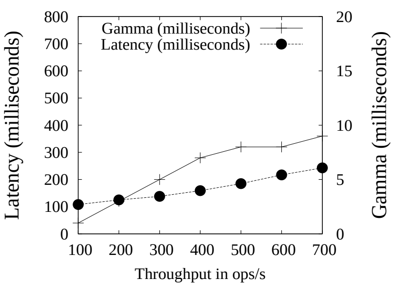


(b) With a Linear Function

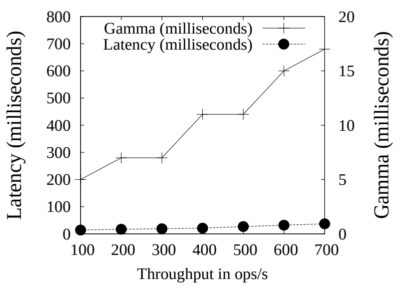


(c) With a Spline Function

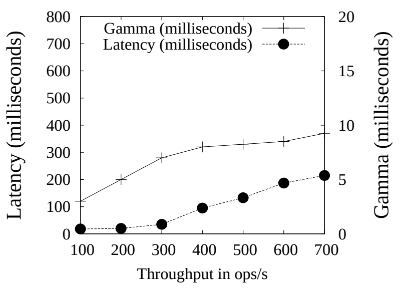
Fig. 1: Variations in the throughput with different user defined function f(x) supplied to YCSBD



(a) Operations With READ/WRITE QUORUM



(b) Operations With READ ONE/WRITE ONE



(c) Operations With OptCon

Fig. 2: Adaptability of OptCon to Varying Target Throughput: under a subSLA (Latency: 300ms Staleness: 10ms)

and distributed datastores. Hence, YCSB-D can be used to evaluate adaptive frameworks that tune distributed datastores. However, OptCon is the only open-source available adaptive framework that we could find. The source code for OptCon can be found at: https://github.com/ssidhanta/HectorCient/. We have run YCSB-D on a testbed of 20 Amazon ec2 small instances, located in the same Ec2 region, running Ubuntu 13.10, loaded with Cassandra 2.1.2 with a replication factor of 3 (the number of replicas per data item).

Demonstrating Adaptability of OptCon: YCSB workloads are invoked from the YCSB-D web console, running on top of a Cassandra cluster. The staleness and latency thresholds for a given SLA are supplied to OptCon in a configuration file. YCSB-D tunes the target throughput (operations per second) in the YCSB client as per the function f specified in the web console (Figures 1(a), 1(b), and 1(c)), and observe the variations in the severity of staleness (i.e., 95 percentile $\mathbf{\Gamma}$ score) and latency against changing throughput, under a subSLA comprising a 300 ms threshold for latency (following the SLA for Amazon's shopping cart application and a 10 ms threshold for staleness. Following [2], we run YCSB workloads for spans of 60 seconds with the following configuration: 50% ratio of reads and writes, 8 client threads, "latest" distribution, and keyspace size of 100,000. Figures 2(c), 2(a), and 2(b) plot the 95 percentile values of observed $\mathbf{\Gamma}$ scores and observed latency in milliseconds, against varying target throughput, with manually chosen consistency levels and with OptCon. Since frequency of operations on each key increases with increase in target throughput applied from the YCSB client, the chances of observing stale results increases as well. Hence, the plots show that, with increasing target throughput, the 95 percentile $\mathbf{\Gamma}$ score increases. Stronger consistency levels QUORUM, in Figure 2(a)) satisfy the staleness threshold against all values of applied target throughput. Weak consistency level (i.e., ONE, in Figure 2(b)) satisfies both the latency and staleness threshold for lower values of target throughput, but violates the 10 ms staleness threshold where target throughput exceeds 400 op/s. Hence, for throughput ≤ 400 op /s, ONE or QUORUM is the matching choice, while QUORUM is matching for throughput > 400 op/s. OptCon (Figure 2(c)) applies weak consistency level ONE for target throughput ≤ 400 op/s, to achieve the latency threshold and to maximize the observed throughput. It applies strong consistency level QUORUM for target throughput > 400 op/s. QUORUM (Figure 2(a)) satisfies the subSLA for all values of target throughput. However, choosing ONE for throughput ≤ 400 op/s, OptCon (Figure 2(c)) is able is able to produce 61 ms less latency, on an average, than with QUORUM (Figure 2(a)).

References

[1] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB. In *SoCC '10*.

[2] W. M. Golab, M. R. Rahman, A. AuYoung, K. Keeton, J. J. Wylie, and I. Gupta. Client-centric benchmarking of eventual consistency for cloud storage systems. In *ICDCS*, page 28, 2014.

[3] S. Sidhanta, W. M. Golab, S. Mukhopadhyay, and S. Basu. Optcon: An adaptable sla-aware consistency tuning framework for quorum-based stores. In *CCGrid 2016*.

[4] T. Sprangler. Netflix bandwidth usage climbs to nearly 37% of internet traffic at peak hours. http://variety.com/2015/digital/news/netflix-bandwidth-usage-internet-traffic-1201507187/, 2015.