Benchmarking Key-Value Stores via Trace Replay
(APPENDIX)

Edwin F. Boza, Cesar San-Lucas, Cristina L. Abad, Jose A. Viteri
Escuela Superior Politecnica del Litoral, ESPOL
{*eboza,csanluca,cabad,javiteri*}*@fiec.espol.edu.ec*

I. ADDITIONAL EXPERIMENTS AND GRAPHS

A. Performance

Our goal was to see if KV-replay could issue events as fast as YCSB. For KV-replay, we used the closed model with multiple threads, which may violate conservative ordering [1]. This experiment assumes that the events are independent but replayed using the closed model because this model throttles the throughput by using blocking calls. We did not use other models because they are not supported by YCSB and the results would not be comparable.

The motivation was that others may want to compare the results **at full speed** of a synthetic benchmark (YCSB) vs. the replay of a real trace (KV-replay); these comparisons are meaningless if both tools yield a different throughput.

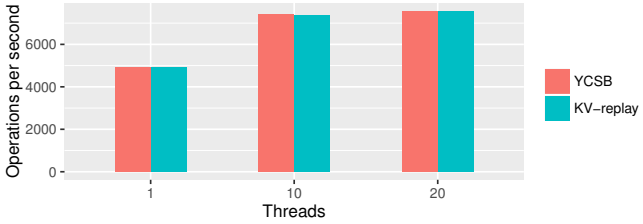

Figure 1. Operations per second issued by YCSB and KV-replay, with 1, 10 and 20 threads. At full speed, KV-replay can issue operations as fast as YCSB.

Figure 1 shows the average count of operations per second in both scenarios. Five million operations were executed on each test. We observe no significant difference in throughput; **KV-replay can stress the system as effectively as YCSB**.

B. Temporal locality

Synthetic workloads, like those generated with a Zipfian distribution, lacks temporal locality and performance results have low accuracy, leading to cache size provisioning errors.

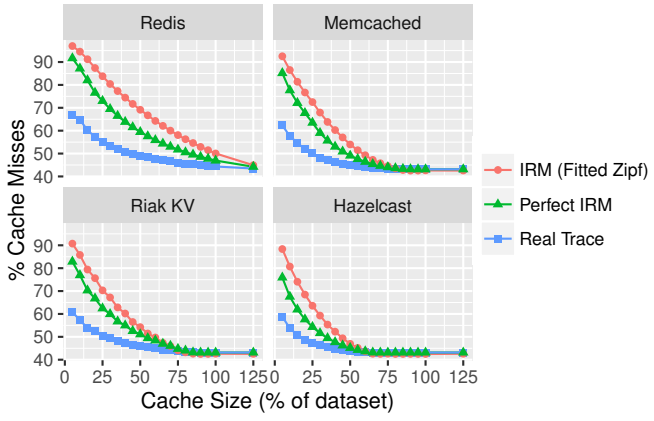

Figure 2. MRCs for in-memory DBs acting as caches with LRU eviction, for three workloads: Zipf (YCSB), a real trace (YouTube, KV-replay), and the real trace shuffled to eliminate the temporal locality and simulate perfect IRM.

Figure 2 shows the miss rate curves (MRCs) for different in-memory DBs acting as caches. Synthetic workload performance is more closer to that of perfect IRM than to the real trace.

C. Burstiness

The aim of this set of experiments was to observe the variability of the arrivals for each workload, by counting the number of stream references for each time window.

For these experiments, we used the real trace, and compare it with a Zipf trace modeled with the popularity distribution observed in the real trace. We also include the reproduction of a suffled version of the real trace to remove temporal locality.

1) YouTube traces: For the YouTube traces, we ran a distributed experiment with a Redis-cluster with 8 nodes. To run the evaluation, KV-replay was executed with 10 threads.

Table I and Figure3, shows the burstiness statistics for the YouTube traces and its corresponding modeled workloads. Arrivals for the real trace are more dispersed than arrivals for the Zipfian modeled workload. Also, the *Perfect IRM* workload, which is obtained after removing the temporal locality from the real trace, is more similar to the Zipfian modeled workload, supporting the claims that synthetic workloads do not reproduce temporal locality.

Table I
BURSTINES STATISTICS FOR YOUTUBE TRACE

	Coefficient of Variation	**Range**	**IQR**	**Standard Deviation**	**0.99 Quantil**	**0.999 Quantil**
Trace real	5.899302	793, 1184	74.25	58.20356	1132.670	1174.096
Trace Permutado	4.084764	814, 1116	52	39.90103	1079.000	1098.668
Trace Zipf modeled	3.671868	860, 1070	45	35.84311	1055.620	1069.381

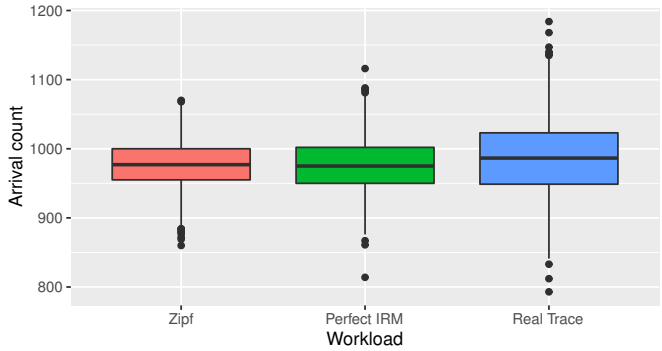

Figure 3. Summary of arrival counts per each 1-second slots for YouTube traces, and its corresponding modeled workloads

In Figure 4, we can see the per node arrival counts for each 1-second slots for YouTube traces, and its corresponding modeled workloads. No hot-spots are observed between nodes for Zipfian modeled and Perfer IRM workloads.

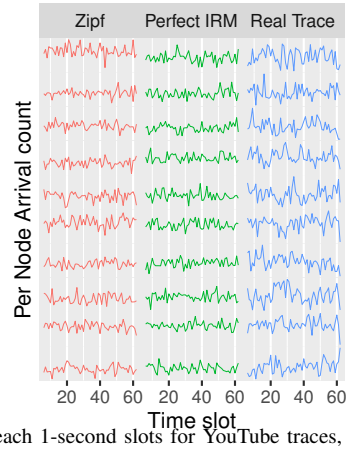

Figure 4. Per node arrival counts for each 1-second slots for YouTube traces, and its corresponding modeled workloads

2) Yahoo traces: For the Yahoo traces, we ran a distributed experiment with a Redis-cluster with 10 nodes. To run the evaluation, KV-replay was executed with 10 threads.

Table II and Figure5, shows the burstiness statistics for the Yahoo traces and its corresponding modeled workloads. Results are similar to those obtained from YouTube traces; Yahoo traces, which holds a higher temporal locality, show greater differences to their corresponded modeled workloads.

Table II
BURSTINES STATISTICS FOR YAHOO TRACE

	Coefficient of Variation	Range	IQR	Standard Deviation	0.99 Quantil	0.999 Quantil
Trace real	5.023625	825, 1223	66	50.02215	1123.000	1183.082
Trace Permutado	3.380317	845, 1115	45	33.67624	1075	1103
Trace Zipf modeled	3.364452	872, 1109	44	33.36714	1069.410	1096.041

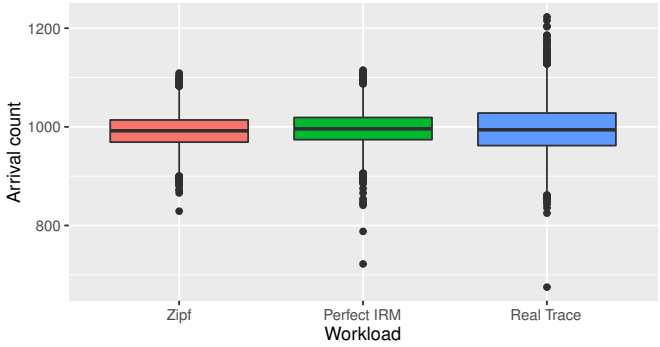

Figure 5. Summary of arrival counts per each 1-second slots for Yahoo traces, and its corresponding modeled workloads

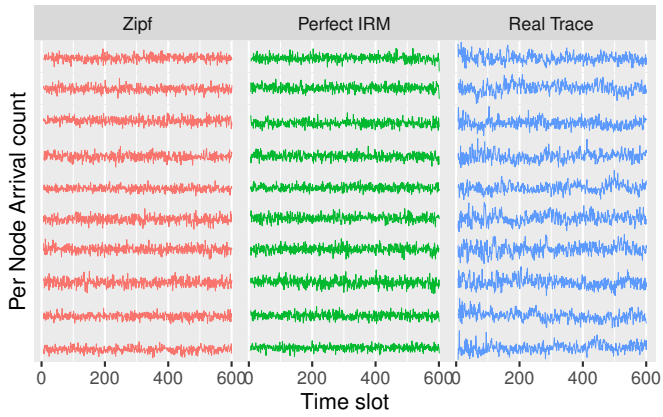

Figure 6. Per node arrival counts for each 1-second slots for Yahoo traces, and its corresponding modeled workloads

We can also assess the load (un)balance, using the coefficient of variation (c_v), which is a measure of the variability of a series [2] (Figure 7). A c_v of 0% means that the requests are perfectly balanced across the nodes in the cluster and across the duration of the experiment. The c_v is 33.9% and 18.4% for the Zipf and YouTube workloads, respectively.

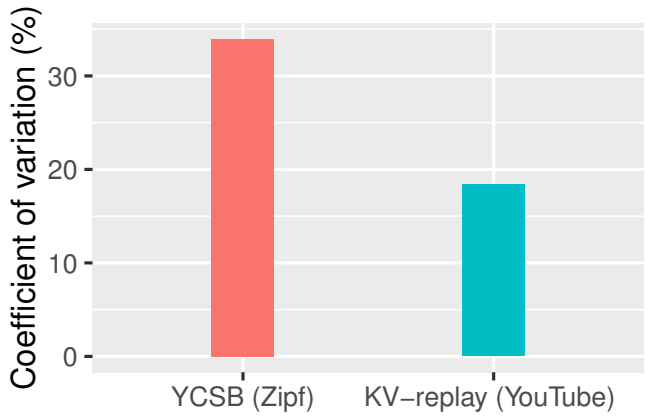

Figure 7. Coefficient of variation (c_v) of the number of requests that each node in a 10-node Redis Cluster received in each 1-second slot of the experiment. A c_v of 0% indicates a perfectly balanced load.

D. Workload Scaling

1) Temporal Scaling: For testing the impact of **temporal scaling**, we ran a set of experiments replaying the YouTube traces, against a Redis database configured as caching system, varying the scaling factor with the values: 0.01, 0.10, 0.25, 0.50, 0.75, 1.0, 1.25, 1.50, 1.75 and 2.0.

Figure 8, shows the variation of the observed *throughtput*, for each scaling factor value. Throughput will increase when the scaling factor is less than 1, because the magnitude of the interarrival times will be reduced; this feature allows us to increase the pressure on the storage system being evaluated. Conversely, when the scaling factor is greater than 1, the interarrival times will be augmented, and the storage system will receive a slower stream of requests.

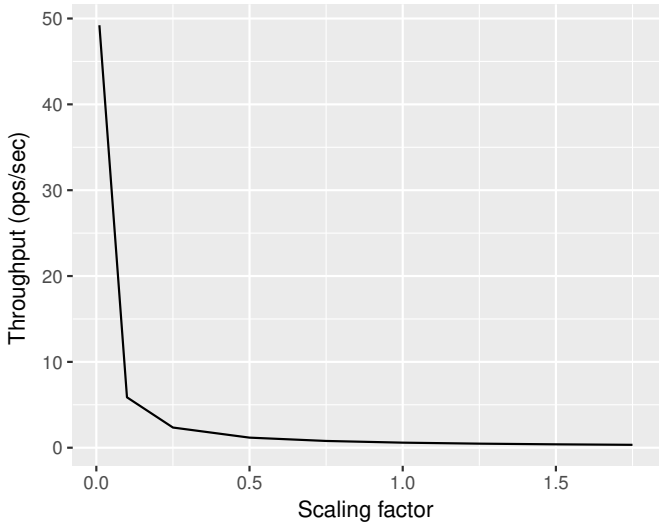

Figure 8. Temporal Scaling - YouTube 30min

REFERENCES

[1] N. Zhu, J. Chen, and T. Chiueh, "TBBT: Scalable and accurate trace replay for file server evaluation," in *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, 2005.

[2] H. Abdi, "Coefficient of variation," *Encyclopedia of research design*, pp. 169–171, 2010.