A Framework for Benchmarking Consistency in Distributed key-Value Storage Systems

Summer intern: Muntasir R. Rahman (HP Labs / UIUC)
Presenter: Wojciech Golab (HP Labs / University of Waterloo)
Collaborators: Alvin AuYoung, Kimberly Keeton, Jay J. Wylie (HP Labs, HP Labs, HP Labs / LinkedIn)
Motivation
Eventual consistency

Write(x, 1)

writer

x = 0

x = 0

x = 0
Eventual consistency

Write(x, 1)

writer

x = 1
x = 0
x = 0
Eventual consistency

\[ \text{Read}(x) \rightarrow 0 \]
Eventual consistency

eventually…
Eventual consistency

“If no new updates are made to the object, eventually all accesses will return the last updated value”
– Werner Vogels, CACM 2008

1. How soon is eventual?
2. What happens if updates are made continuously?
Goal

Quantify the actual consistency observed by clients accessing an EC key-value store in arbitrary workloads.

(in)consistency = staleness
Related Work
Measuring consistency

How much consistency do EC key-value stores provide in practice?


Data-centric vs. client-centric consistency

$\text{client #1 writes 1, then}$

$\text{client #2 reads 0}$
Measurement technique [BT11, Pa11, Wa11]

Write(x, 1)
Read(x) → 0
Read(x) → 0
Read(x) → 1

client #1

client #2
If a tree falls…

\[ x = 1 \]

\[ x = 0 \]

\[ x = 0 \]

client reads \( x = 1 \)
Observer effect
Our Approach
Passive measurement

config. parameters
workload
failure pattern

Trace: start + end time of every operation (collected at clients)

staleness calculation

<table>
<thead>
<tr>
<th>Chi</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>143</td>
</tr>
<tr>
<td>23</td>
<td>60</td>
</tr>
<tr>
<td>678</td>
<td>1432</td>
</tr>
</tbody>
</table>
Definitions

**Fresh**: a value is considered fresh from the “moment” it’s written until it’s overwritten.

**Staleness**: how long ago was the value read last considered fresh? (e.g., 100ms ago or 0ms ago)

*Note*: not looking at causality
Calculating staleness


\[ \Delta = \max_{i,j} \chi(i, j) \]
Experimental Results
Experimental setup: Cassandra + YCSB + Δ

Key id: 0

10 servers
- dual-socket Xeons
- ample DRAM
- 1 GbE network
- SSD

each server runs:
- 1 Cassandra node
- 1 YCSB client
  (“hot spot” distribution, 80% read, 20% write)

Hot-set (first 20% keys)

Cold-set (remaining 80% keys)

1M

200k
Histogram of score function ($\chi$) values

- $\chi$ computed for all keys
- only positive $\chi$ shown
- 255 positive scores among approx. 324k writes and 1.3M reads
Time series plot of score function ($\chi$) values

- $\chi$ computed for all keys
- only positive $\chi$ shown
- 255 positive scores among approx. 324k writes and 1.3M reads
Summary

Methodology for passive benchmarking of consistency in EC key-value stores that captures faithfully the consistency actually observed by clients.
Ongoing and future work

Science:
• understanding the actual behavior of EC key-value storage systems in practice (workloads, configurations, failures)

Engineering:
• defining and enforcing consistency SLAs
• consistency “amplification”
Thank You!

Muntasir R. Rahman (mrahman2@illinois.edu)
Wojciech Golab (wgolab@uwaterloo.ca)
Alvin AuYoung (alvin.auyoung@hp.com)
Kimberly Keeton (kimberly.keeton@hp.com)
Jay J. Wylie (jay.j.wylie@gmail.com)