What consistency does your key-value store actually provide?

Eric Anderson, Xiaozhou (Steve) Li, Mehul Shah, Joseph Tucek, and Jay Wylie
HP Labs
HotDep 2010
October 3, 2010
Outline

• Key-value stores
• Consistencies
• Checking consistencies
• Algorithms
• Findings
KEY-VALUE STORES

Simple Storage Service (S3), Dynamo

Google Storage for Developers

Cassandra

Project Voldemort
A distributed database.
CONSISTENCIES

Eventually consistent
Read-your-writes

Quorum-based, multiple levels
Read-repair, vector clocks, hinted hand-off

Sequential writes
DO YOU BELIEVE THEM?

Why not?
WHY DO YOU WANT TO KNOW?
Verify SLAs that may contain consistency guarantees
WHY DO YOU WANT TO KNOW?

Choose the one that meets your consistency requirements
WHY DO YOU WANT TO KNOW?

Choose a proper service level for own workload

- What you pay is what you get
- What you get depends on your workload
- **Tough** workloads & failures: Worse than expected / promised
- **Benign** workload & good operating conditions: Better than minimal guarantee
WHAT CAN A USER DO?

If we know the internal protocols …
WHAT CAN A USER DO?
If we don’t know the internal protocols …

User machine
get(key)
put(key, value)
User machine
User machine
Key-value store
User machine
User machine
User machine
CLIENT TRACES

Machine 1
- get/put
- request sent
- reply received

Machine 2
- request sent
- reply received

Machine 3
- request sent
- reply received
REGISTER-BASED CONSISTENCY
[Lamport, Distributed Computing, 1986]

– Atomic
– Safe
– Regular
ATOMIC/REGULAR/SAFE

Atomic

Regular

Safe

Unsafe
OVERALL APPROACH
For all three: safe, regular, atomic

1. Construct a digraph
   - Vertices = operations
   - Edges = precedence

2. Add edges
   - Time
   - Data
   - Hybrid

3. Check if graph is DAG
ASSUMPTIONS

– Client timestamps are reasonably synchronized
– Or they are calibrated during merge
  – Chirp [Anderson et al., MASCOTS, 2009]
– All writes write a distinct value
– There is a default value for each key
ADDING TIME EDGES

W(0)  W(1)

R(1)  R(0)

W(0)  W(1)

R(1)  R(0)
ADDING DATA EDGES
ADDING HYBRID EDGES
DETECTING CYCLES

DFS

W(0) → W(1) → R(1) → R(0)

W(0) ← W(1)
COUNTING VIOLATIONS

Number of cycles found in DFS

cycles=1

cycles=2

Feedback arc set
Feedback vertex set
## CHECKING REGULARITY AND SAFETY

<table>
<thead>
<tr>
<th>Atomicity</th>
<th>Regularity</th>
<th>Safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Keep all reads and writes</td>
<td>Remove reads that read a concurrent write’s value</td>
</tr>
<tr>
<td>2</td>
<td>Add time edges</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Add data edges</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Add hybrid edges</td>
<td></td>
</tr>
</tbody>
</table>
Edges $m=n^2$ even in typical cases; all-pair reachability takes $mn=n^3$ time. Reduced to $mn=n^2$ time in typical cases.
PAHOEHOE
[Anderson et al., DSN, 2010]

- A key-value store prototype
- Erasure-coded
- Multi-datacenter
EXPERIMENT SETUP

Proxy

get(key)
put(key, value)

Emulated wide-area link between datacenters
EXPERIMENT SETUP

- Proxy is in data center and shares NTP w/ servers
- 1000 operations
- Similar to YCSB microbenchmark
  - Larger object size: 128KB
  - 40% gets + 60% updates = 70% gets + 30% puts
- Varying
  - Number of keys
  - Number of processes
  - Distribution (uniform, Zipfian)
VIOLATIONS VS. KEYS

Concurrency = 128
VIOLATIONS VS. CONCURRENCY

Keys = 8

The graph shows the relationship between the number of violations and the number of concurrent processes for different key distribution patterns. The x-axis represents the number of concurrent processes, while the y-axis represents the number of violations.

- uni-atomic
- uni-reg
- uni-safe
- zipf-atomic
- zipf-reg
- zipf-safe

As the number of concurrent processes increases, the number of violations also increases for all key distribution patterns, but the rate of increase varies. The uni-atomic pattern shows the highest number of violations, followed by the zipf-patterns, and finally the uni-safe pattern.
RELATED WORK

[Misra, TOPLAS, 1986]

– Misra’s algorithm
– Reasons about values
– Only for atomicity
– Probably can be extended for safety and regularity
– Harder to quantify violation severity
ONLINE CONSISTENCY CHECKING

failures

time
CONCLUSIONS

– Independent checking useful
– Algorithms for checking three semantics
– Eventually consistent may perform atomically
– Future work
  • Other semantics
  • Implement online checking
  • Monitor key-value stores