Debugging Deployed Distributed Systems

Xuezheng Liu, Zhenyu Guo, Xi Wang, Feibo Chen, Xiaochen Lian, Jian Tang, Ming Wu, M. Frans Kaashoek, Zheng Zhang

Donghan Miao
Overview

- Introduction
- Design of $D^3S$
- Global snapshot
- Case study
- Performance evaluation
Introduction

• Distributed systems are difficult to test, and debug - they have bugs that are hard to track down.
  • Large scale
  • Network/Node failure
Introduction

• Log analysis
• Consistent global view of the whole system
Introduction

• Collect the local states
• Periodically send the buffer to a central machine
• Order the state of each machine in a global snapshot, and check for incorrect behavior.
Introduction

• Developers must record the state of each machine and order the states into a globally-consistent snapshot.

• Developers must decide what state to record: monitoring too much may slow down the system, while monitoring too little may miss abnormal behaviors.

• Developers may need to distribute checkers across several machines, because a central checker may not be able to handle a system deployed on many machines.

• Developers should approximate a globally-consistent snapshot when some processes being checked fail.
Introduction - D³S Overview

- A simple model for writing distributed predicates*
- Run-time checking
- Failure-tolerant
- decentralized checker, scalable to large systems

*Predicate: a distributed invariant that must hold for a system as a whole (e.g. no two machines can share an exclusive lock).
Design

Predicates (States + Logic)

Symbol Info

State Exposer (SE)

Checking Logic (CL)

Dynamic Injection

Violation reports, Seq of states
**D³S Workflow**

Predicate: No conflicts

Conflict!

Violation!
Distributed lock service in Boxwood

• Distributed reader-writer locks allows clients to acquire multiple reader locks or single writer lock
• Lock mode: exclusive, shared
• Invariant: only one client can hold a lock in the exclusive mode
Writing Predicates

//Computation graph
V0: exposer \rightarrow \{ (client: ClientID, lock: LockID, mode: LockMode) \}
V1: V0 \rightarrow \{ (conflict: LockID) \}

after (ClientNode::OnLockAcquired) addtuple ($0->m_NodeID, $1, $2)
after (ClientNode::OnLockReleased) deltuple ($0->m_NodeID, $1, $2)

//source code from example app
class ClientNode {
    ClientID m_NodeID;
    void OnLockAcquired(LockID, LockMode);
    void OnLockReleased(LockID, LockMode);
};

- Reuse of application code
- Binary Instrumentation

Tuples of (C, L, M)
Writing Predicates

//Computation graph
V0: exposer \rightarrow \{ (\text{client: ClientID, lock: LockID, mode: LockMode}) \} 
V1: V0 \rightarrow \{ (\text{conflict: LockID}) \}

after (ClientNode::OnLockAcquired) \textbf{addtuple} ($0\rightarrow m\_\text{NodeID}, 1, 2$)
after (ClientNode::OnLockReleased) \textbf{deltuple} ($0\rightarrow m\_\text{NodeID}, 1, 2$)

• Wait for snapshot to complete
• Mapping()
• More complex computation graphs

// C++ code for Predicate
class LockVerifier : public vertex<V1> {
    virtual void \textbf{Execute} (const V0::Collection & snapshot);
    // verify predicate in the required snapshots, output conflicts
    static Key \textbf{Mapping} (const V0::tuple & t);
    // map states to key space
};
• BOX: Icing the APIs

• Zhenyu GUO, Xi WANG*, Xuezheng LIU, Wei LIN, and Zheng ZHANG
• Microsoft Research Asia       Tsinghua University*
• {zhenyu.guo, xueliu, weilin, zzhang}@microsoft.com   wangxi01@mails.tsinghua.edu.cn
Partitioning and Parallelism in D³S

- Dynamic assignment of key spaces to verifiers by a central master.
Global Snapshots

• Predicates are defined over a finite number of consecutive snapshots.
• Each process has a logical clock, initialized to 0.
• The logical clock is increased by 1 every time the process reads it.
• The receiving process set its logical clock to the max(local, received timestamp) \( \rightarrow \) preserves happens-before relationship
Consistent Snapshots

App node should report its timestamp periodically even if it has nothing to report.

The receiving process set its logical clock to the max(local, received timestamp)
Case Study: PacificA

- PacificA: Replication in Log-Based Distributed Storage Systems
  - Wei Lin, Mao Yang
  - Microsoft Research Asia
  - fweilin, maoyangg@microsoft.com

- Lintao Zhang, Lidong Zhou
  - Microsoft Research Silicon Valley
  - flintaoz, lidongzg@microsoft.com
Case Study: PacificA

- Data slice: 100MB chunk, replicated in 3 nodes.
- Predicate
  - There is at most one primary replica in each group of replica nodes
- Deployment
  - 8 machines
  - Test scenario query with random I/O
  - Randomly crash & restart processes
  - $D^3S : \langle\text{Slice}_\text{identifier}, \text{MachineID}, \text{Primary/Secondary}\rangle$
- Debugging
  - 3 checkers, partitioned by replica groups
Pacific A: Architecture

Meta Server

Slice server
Sid=2,S; Sid=1,S

Slice server
Sid=2,S; Sid=1,P

Verifier

Report: timestamp, node, event seq
Performance Evaluation

- **PacificA**
- **Overhead is less than 8%**
- **I/O consumption < 0.5%**
## Practice in real-world applications

<table>
<thead>
<tr>
<th>Application</th>
<th>LoC</th>
<th>Predicates</th>
<th>LoP</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>PacificA</td>
<td>67,263</td>
<td>membership consistency; leader election; consistency among replicas</td>
<td>118</td>
<td>3 correctness bugs</td>
</tr>
<tr>
<td>(Structured data storage)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paxos implementation</td>
<td>6,993</td>
<td>consistency in consensus outputs; leader election</td>
<td>50</td>
<td>2 correctness bugs</td>
</tr>
<tr>
<td>Web search engine</td>
<td>26,036</td>
<td>unbalanced response time of indexing servers</td>
<td>81</td>
<td>1 performance problem</td>
</tr>
<tr>
<td>Chord (DHT)</td>
<td>7,640</td>
<td>aggregate key range coverage; conflict key holders</td>
<td>72</td>
<td>tradeoff bw/availability &amp; consistency</td>
</tr>
<tr>
<td>BitTorrent client</td>
<td>36,117</td>
<td>Health in neighbor set; distribution of downloaded pieces; peer contribution rank</td>
<td>210</td>
<td>2 performance bugs; free riders</td>
</tr>
</tbody>
</table>
# Performance Evaluation

<table>
<thead>
<tr>
<th></th>
<th>I/O</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chord</td>
<td>negligible</td>
<td>negligible</td>
</tr>
<tr>
<td>Paxos</td>
<td>negligible</td>
<td>negligible</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>-</td>
<td>&lt;2%</td>
</tr>
<tr>
<td>Web Search Engine</td>
<td>-</td>
<td>&lt;2%</td>
</tr>
<tr>
<td>PacificA</td>
<td>&lt;0.5%</td>
<td>&lt;8%</td>
</tr>
</tbody>
</table>
Thank you.