SAMC: Semantic-Aware Model Checking for Fast Discovery of Deep Bugs in Cloud Systems

Tanakorn Leesatapornwongsa, Mingzhe Hao, Pallavi Joshi*, Jeffrey F. Lukman†, and Haryadi S. Gunawi
Internet Services
Reliability

Complex failures → “Deep bugs”
**Deep Bug Example**

**ZooKeeper** (synchronization service)

*Issue #335.*

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes
8. A and B are back online after C crashes
9. A becomes leader
10. A's commits new txid-value pair (11, Y)
11. C is back online after A's new tx commit
12. C announce to B (11, X)
13. B replies diff starting with tx 12
14. Inconsistency: A has (11, Y), C has (11, X)

**PERMANENT INCONSISTENT REPLICA**
Deep Bug Characteristics

**ZooKeeper** (synchronization service)

**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes
8. A and B are back online after C crashes
9. A becomes leader
10. A's commits new txid-value pair (11,Y)
11. C is back online after A's new tx commit
12. C announce to B (11, X)
13. B replies diff starting with tx 12
14. Inconsistency: A has (11,Y), C has (11, X)

1. Out-of-order messages
2. Multiple crashes
3. Multiple reboots

1 2 3 HAPPEN IN ANY ORDER
Study of Deep Bugs

CAUTION

SEVERE IMPLICATIONS

INCONSISTENT REPLICAS, DATA LOSS, DOWNTIMES, ETC.
How do we catch deep bugs in distributed systems?
How to Catch Deep Bugs

• Distributed system model checker
  – Re-ordering all non-deterministic events
  – Find which specific orderings lead to bugs

<table>
<thead>
<tr>
<th>ZooKeeper (synchronization service) Issue #335. Permanent inconsistent data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Nodes A, B, C start (w/ txid: 10)</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>2. B becomes leader</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>3. B crashes</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>4. C becomes leader</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>5. C commits new txid-value pair (11, X)</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>6. A crashes, before committing the new txid 11</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>7. C loses quorum and C crashes</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>8. A and B are back online after C crashes</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>9. A becomes leader</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>10. A’s commits new txid-value pair (11, Y)</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>11. C is back online after A’s new tx commit</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>12. C announce to B (11, X)</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>13. B replies diff starting with tx 12</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>14. Inconsistency: A has (11, Y), C has (11, X)</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>
What’s Wrong with Existing Model Checkers?

• Last 7 years
  • MaceMC [NSDI’07], Modist [NSDI’09], dBug [SSV’10], Demeter [SOSP’13], etc.

• BUT
  – Too many events
  – Multiple crashes and reboots
    • Create more messages
  – No model checker incorporate multiple crashes and reboots
  – Cannot find deep bugs!
How do we catch deep bugs \textbf{REALLY FAST}?
Black-Box Approach

- Existing model checkers are so slow
- They treat target systems as black boxes
  - A large number of event orderings are generated

\[
\begin{array}{cccc}
A & B & C & D \\
\end{array}
\]

Black Box Model Checker

- ABCD
- ABDC
- ACBD
- ACDB
- ADBC
- ...
- (24 total)
Semantic Knowledge

• How can we make model checker fast?
  – Exploit semantic knowledge
• Semantic-aware model checker (SAMC)
Black Box vs. SAMC

Black Box Model Checker

<table>
<thead>
<tr>
<th>ABCD</th>
<th>ABDC</th>
<th>ACBD</th>
<th>ACDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADBC</td>
<td>ADCB</td>
<td>ADBC</td>
<td>ADDB</td>
</tr>
<tr>
<td>ADBC</td>
<td>ACDB</td>
<td>ADBD</td>
<td>ADDB</td>
</tr>
<tr>
<td>BACD</td>
<td>BACD</td>
<td>BACD</td>
<td>BACD</td>
</tr>
<tr>
<td>BADC</td>
<td>BADC</td>
<td>BADC</td>
<td>BADC</td>
</tr>
<tr>
<td>BCAD</td>
<td>BCAD</td>
<td>BCAD</td>
<td>BCAD</td>
</tr>
<tr>
<td>BCDA</td>
<td>BCDA</td>
<td>BCDA</td>
<td>BCDA</td>
</tr>
<tr>
<td>BDAC</td>
<td>BDAC</td>
<td>BDAC</td>
<td>BDAC</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

SAMC with message processing semantic

<table>
<thead>
<tr>
<th>ABCD</th>
<th>ABDC</th>
<th>ACBD</th>
<th>ACDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADBC</td>
<td>ADCB</td>
<td>ADBC</td>
<td>ADDB</td>
</tr>
<tr>
<td>ADBC</td>
<td>ACDB</td>
<td>ADBD</td>
<td>ADDB</td>
</tr>
<tr>
<td>BACD</td>
<td>BACD</td>
<td>BACD</td>
<td>BACD</td>
</tr>
<tr>
<td>BADC</td>
<td>BADC</td>
<td>BADC</td>
<td>BADC</td>
</tr>
<tr>
<td>BCAD</td>
<td>BCAD</td>
<td>BCAD</td>
<td>BCAD</td>
</tr>
<tr>
<td>BCDA</td>
<td>BCDA</td>
<td>BCDA</td>
<td>BCDA</td>
</tr>
<tr>
<td>BDAC</td>
<td>BDAC</td>
<td>BDAC</td>
<td>BDAC</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Unnecessary Re-orderings

Lead to the same state
SAMC with Crashes

<table>
<thead>
<tr>
<th>Black Box Model checker</th>
<th>SAMC with crash recovery semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCDX</td>
<td>ABCDX</td>
</tr>
<tr>
<td>ABCXD</td>
<td>ABCXD</td>
</tr>
<tr>
<td>ABXCD</td>
<td>ABXCD</td>
</tr>
<tr>
<td>AXBCD</td>
<td>AXBCD</td>
</tr>
<tr>
<td>XABCD</td>
<td>XABCD</td>
</tr>
<tr>
<td>ABDCX</td>
<td>ABDCX</td>
</tr>
<tr>
<td>ABDXC</td>
<td>ABDXC</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Unnecessary Re-orderings

Crash Recovery Semantic

N1

N2

N3

N4
SAMC Implementation and Integration

• SAMC implementation
  – 10,000 LOC from scratch
• Apply SAMC to 3 cloud systems
  – 7 protocols
  – 10 versions

<table>
<thead>
<tr>
<th>Cloud systems</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra</td>
<td>Gossiper</td>
</tr>
<tr>
<td></td>
<td>Hinted handoff</td>
</tr>
<tr>
<td></td>
<td>Read/write</td>
</tr>
<tr>
<td>Hadoop 2.0</td>
<td>Cluster management</td>
</tr>
<tr>
<td></td>
<td>Speculative execution</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>Atomic broadcast</td>
</tr>
<tr>
<td></td>
<td>Leader election</td>
</tr>
</tbody>
</table>
Result

• Reproduced 12 old bugs
  – Compare to state-of-the-art techniques
    • Dynamic Partial Order Reduction (DPOR)
    • Random-DPOR
  – Find bugs 2x to 340x faster
    • 49x on average

• Found 2 new bugs
  – Submit them to developers
Outline

• Intuition

• SAMC
  – Local-Message Independence
  – Crash-Message Independence
  – Crash Recovery Symmetry
  – Reboot Synchronization Symmetry

• Evaluation
Dependency vs. Independency

A, B = Dependent

A, B = Independent

INDEPENDENT = NO REORDERING
2X SPEED-UP
Black Box vs. SAMC

Black Box Model Checker

ABC D
ABDC
ACBD
ACDB
A DBC
ADCB
BACD
BADC
BCAD
BCDA
...

4!=24 orderings

SAMC

ABCD
ABDC
BACD
BADC

6X SPEED-UP
How to Declare Message Independency?

Q: Which concurrent messages are independent?

A: Use message processing semantic
Message Processing Semantic in Simplified Leader Election

Belief = n3

Vote = 1
Vote = 2
Vote = 4
Removing Re-ordering via Message Processing Semantic

\[
\text{if } \left( \text{vote} \leq \text{belief} \right) \\
\quad \text{// do nothing} \\
\text{else} \\
\quad \text{belief} = \text{vote};
\]
Formalizing the Intuition

MESSAGE PROCESSING SEMANTIC

```java
if (vote <= belief)
    // do nothing
else
    belief = vote;
```

DISCARD PATTERN

```java
if (isDiscard(msg, ls)) {
    // do nothing;
}
```

DISCARD PREDICATE

```java
boolean discardPredicate(msg, ls) {
    if (msg.vote <= ls.belief)
        return true;
    else
        return false;
}
```
Formalizing the Intuition

Belief=4

- Vote=1
- Vote=2
- Vote=3

<table>
<thead>
<tr>
<th>vote</th>
<th>belief</th>
<th>discardPredicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m_x</th>
<th>m_y</th>
<th>discard(m_x)</th>
<th>discard(m_y)</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>true</td>
<td>✔</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>true</td>
<td>true</td>
<td>✔</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>true</td>
<td>true</td>
<td>✔</td>
</tr>
</tbody>
</table>
Outline

• Intuition

• **SAMC**
  – Local-Message Independence
  – Crash-Message Independence
  – Crash Recovery Symmetry
  – Reboot Synchronization Symmetry

• Evaluation
Local state: $ls_1$

interceptor

Local state: $ls_2$

interceptor

release(c)

SAMC Architecture

| Protocol Specific Rules | Leader Election | Atomic Broadcast | ... | ...
|-------------------------|-----------------|-------------------|-----|-----
| Generic Reduction Policies | LMI | CMI | CRS | RSS |
| Basic Reduction Techniques | Symmetry | Dynamic Partial Order Reduction (DPOR) |
Local-Message Independence

SAMC

Generic Reduction Policies

Local-Message Independence (LMI)

Crash-Message Independence (CMI)

Crash Recovery Symmetry (CRS)

Reboot Synchronization Symmetry (RSS)
Local-Message Independence

- **Discard pattern**
- **Increment pattern**

```java
if (msg.type == ack) {
    node.ackCount++;
}
```

```java
boolean incrementPredicate(msg, ls) {
    if (msg.type == ack)
        return true;
    else
        return false;
}
```

- **Constant pattern**
Crash-Message Independence

**SAMC**

Generic Reduction Policies

- Local-Message Independence (LMI)
- Crash-Recovery Symmetry (CRS)
- Crash-Message Independence (CMI)
- Reboot Synchronization Symmetry (RSS)
Crash-MESSAGE INDEPENDENCE

void handleCrash() {
    if (X == follower && isQuorum())
        followerCount--;
    // No new messages
}

boolean localImpact(X, ls) {
    if (X == follower && isQuorum())
        return true;
    else
        return false;
}
void handleCrash() {
    if (X == leader || !isQuorum())
        electLeader();
        // New messages created
}

boolean globalImpact(X, ls) {
    if (X == leader || !isQuorum())
        return true;
    else
        return false;
}
Crash Recovery Symmetry
Reboot Synchronization Symmetry

SAMC

Generic Reduction Policies

- Local-Message Independence (LMI)
- Crash-Message Independence (CMI)
- Crash Recovery Symmetry (CRS)
- Reboot Synchronization Symmetry (RSS)
Outline

• Intuition
• SAMC
  – Local-Message Independence
  – Crash-Message Independence
  – Crash Recovery Symmetry
  – Reboot Synchronization Symmetry
• Evaluation
Evaluation

- SAMC implementation
  - 10,000 LOC from scratch
- Apply SAMC to 3 cloud systems
  - 7 protocols
  - 10 versions

<table>
<thead>
<tr>
<th>Cloud systems</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra</td>
<td>Gossiper</td>
</tr>
<tr>
<td></td>
<td>Hinted handoff</td>
</tr>
<tr>
<td></td>
<td>Read/write</td>
</tr>
<tr>
<td>Hadoop 2.0</td>
<td>Cluster management</td>
</tr>
<tr>
<td></td>
<td>Speculative execution</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>Atomic broadcast</td>
</tr>
<tr>
<td></td>
<td>Leader election</td>
</tr>
</tbody>
</table>
Protocol-Specific Rules
(e.g. ZooKeeper Leader Election)

- Guide SAMC to remove re-orderings
- 35 LOC on average per protocol

<table>
<thead>
<tr>
<th>Local-Message Independence (LMI)</th>
<th>Crash-Message Independence (CMI)</th>
<th>Crash Recovery Symmetry (CRS)</th>
</tr>
</thead>
</table>
| \texttt{bool pd : \neg newVote(n, s);} | \texttt{bool pg (s, X):}  
\hspace{1em} \texttt{if (s.r1 == F \&\& X.r1 == L)}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{if (s.r1 == L \&\& X.r1 == F  
\hspace{2em} \&\& \neg \text{quorumAfterX}(s))}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{if (s.r1 == S \&\& X.r1 == S)}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{bool pl (s, X):}  
\hspace{1em} \hspace{1em} \texttt{if (s.r1 == L \&\& X.r1 == F  
\hspace{2em} \&\& \neg \text{quorumAfterX}(s))}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{bool quorumAfterX(s):}  
\hspace{1em} \hspace{1em} \texttt{ret ((a.fol-1) > s.all/2);}  
\hspace{1em} \texttt{ret 0;} | \texttt{bool pr1(s,C):}  
\hspace{1em} \texttt{if (s.r1 == L \&\& C.r1 == F  
\hspace{2em} \&\& \neg \text{quorumAfterX}(s))}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{rals1:\{r1,fol,all\};}  
\texttt{bool pr2(s,C):}  
\hspace{1em} \texttt{if (s.r1 == L \&\& C.r1 == F  
\hspace{2em} \&\& \neg \text{quorumAfterX}(s))}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{rals2: \{r1,fol,lid,ep,tx,clk\}}  
\texttt{bool pr3(s,C):}  
\hspace{1em} \texttt{if (s.r1 == F \&\& c.r1 == L)}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{rals3: \{r1,fol,lid,ep,tx,clk\}}  
\texttt{bool pr4:}  
\hspace{1em} \texttt{if (s.r1 == S)}  
\hspace{1em} \hspace{1em} \texttt{ret 1;}  
\hspace{1em} \texttt{rals4: \{r1,lid,ep,tx,clk\}} |
Catching Old Bugs

A table shows number of executions to reach the bugs and speedup

<table>
<thead>
<tr>
<th>Issue#</th>
<th>SAMC</th>
<th>Black-Box DPOR</th>
<th>Random</th>
<th>Random DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZooKeeper-335</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper-790</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper-975</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper-1075</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper-1419</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper-1492</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZooKeeper-1653</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapReduce-4748</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapReduce-5489</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapReduce-5505</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra-3395</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra-3626</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Catching Old Bugs

A table shows number of executions to reach the bugs and speedup

<table>
<thead>
<tr>
<th>Issue#</th>
<th>SAMC</th>
<th>Black-Box DPOR</th>
<th>Random</th>
<th>Random DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#exe</td>
<td>#exe</td>
<td>speedup</td>
<td>#exe</td>
</tr>
<tr>
<td>ZooKeeper-335</td>
<td>117</td>
<td>5000+</td>
<td>43+</td>
<td>1057</td>
</tr>
<tr>
<td>ZooKeeper-790</td>
<td>7</td>
<td>14</td>
<td>2</td>
<td>225</td>
</tr>
<tr>
<td>ZooKeeper-975</td>
<td>53</td>
<td>967</td>
<td>18</td>
<td>71</td>
</tr>
<tr>
<td>ZooKeeper-1075</td>
<td>16</td>
<td>1081</td>
<td>68</td>
<td>86</td>
</tr>
<tr>
<td>ZooKeeper-1419</td>
<td>100</td>
<td>924</td>
<td>9</td>
<td>2514</td>
</tr>
<tr>
<td>ZooKeeper-1492</td>
<td>576</td>
<td>5000+</td>
<td>9+</td>
<td>5000+</td>
</tr>
<tr>
<td>ZooKeeper-1653</td>
<td>11</td>
<td>945</td>
<td>86</td>
<td>3756</td>
</tr>
<tr>
<td>MapReduce-4748</td>
<td>4</td>
<td>22</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>MapReduce-5489</td>
<td>53</td>
<td>5000+</td>
<td>94+</td>
<td>5000+</td>
</tr>
<tr>
<td>MapReduce-5505</td>
<td>40</td>
<td>1212</td>
<td>30</td>
<td>5000+</td>
</tr>
<tr>
<td>Cassandra-3395</td>
<td>104</td>
<td>2552</td>
<td>25</td>
<td>191</td>
</tr>
<tr>
<td>Cassandra-3626</td>
<td>96</td>
<td>5000+</td>
<td>52+</td>
<td>5000+</td>
</tr>
</tbody>
</table>
Reduction Ratio

- ZooKeeper leader election protocol
- Run black-box DPOR
- After each execution, find that this execution is executed by SAMC or not
- Count DPOR’s executions that are executed by SAMC too

<table>
<thead>
<tr>
<th>#Crash</th>
<th>#Reboot</th>
<th>Reduction Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ALL</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>37X</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>63X</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>103X</td>
</tr>
</tbody>
</table>
Conclusion

• Deep bugs live in the cloud
• Model checker needs to incorporate complex failure to reach deep bugs
  – State space explosion
• Semantic-aware model checking
  – LMI, CMI, CRS, RSS
• Bring future research questions
  – What other semantic knowledge is useful?
  – How to extract them from the code automatically?
Thank You

Questions?

http://ucare.cs.uchicago.edu