Towards Automatically Checking Thousands of Failures with Micro-Specifications

Haryadi S. Gunawi, Thanh Do†, Pallavi Joshi, Joseph M. Hellerstein, Andrea C. Arpaci-Dusseau†, Remzi H. Arpaci-Dusseau†, Koushik Sen

University of California, Berkeley
†University of Wisconsin, Madison
Cloud Era

Solve bigger human problems
Use cluster of thousands of machines
Failures in The Cloud
Failures in The Cloud

“The future is a world of failures everywhere” – Garth Gibson
Failures in The Cloud

“The future is a world of failures everywhere” – Garth Gibson

“Recovery must be a first-class operation” – Raghu Ramakrishnan
Failures in The Cloud

“The future is a world of failures everywhere” – Garth Gibson

“Recovery must be a first-class operation” – Raghu Ramakrishnan

“Reliability has to come from the software” – Jeffrey Dean
When the Cloud Fails: T-Mobile, Microsoft Lose Sidekick Customer Data

By Om Malik  |  Oct. 10, 2009, 3:57pm PDT  |  32 Comments

If you’ve ever been curious about what would happen when a cloud service fails, then you don’t have to wonder any longer. Earlier today, customers of T-Mobile and Sidekick data services provider Danger, a subsidiary of Microsoft, lost access to all their data. Some believe that this data wipeout is because of a botched upgrade. Why it happened matters little to those who are unlikely to get their data back, according to a note posted on T-Mobile forums.
Facebook temporarily loses more than 10% of photos in hard drive failure

Facebook is working to restore access to the photos

By Lucas Mearian
March 9, 2009 12:00 PM ET

Computerworld - Popular social networking site Facebook.com admitted in a blog post today that over the weekend, a hard drive failure led to the temporary loss of 10% to 15% of the photographs stored by its users.

According to the company, several drives failed at once during a routine upgrade Friday night.

“You may have noticed in the past day that some photos aren’t appearing or are displaying a ‘question mark’ graphic when you go to view them. We’re trying to fully understand what happened, since simultaneous hardware failures like this are rare,” Evan Priestley, an engineer at Facebook
Why Failure Recovery Hard?

• Testing is not advanced enough against complex failures
  – Diverse, frequent, and multiple failures
  – FaceBook photo loss
• Recovery is under specified
  – Need to specify failure recovery behaviors
  – Customized well-grounded protocols
• Example: Paxos made live – An engineering perspective [PODC’ 07]
Our Solutions
Our Solutions

• **FTS ("FATE")** – Failure Testing Service
  – New abstraction for failure exploration
  – Systematically exercise **40,000** unique combinations of failures
Our Solutions

• **FTS (“FATE”)** – Failure Testing Service
  – New abstraction for failure exploration
  – Systematically exercise **40,000** unique combinations of failures

• **DTS (“DESTINI”)** – Declarative Testing Specification
  – Enable concise recovery specifications
  – We have written **74** checks (3 lines / check)
Our Solutions

- **FTS** ("FATE") – Failure Testing Service
  - New abstraction for failure exploration
  - Systematically exercise 40,000 unique combinations of failures
- **DTS** ("DESTINI") – Declarative Testing Specification
  - Enable concise recovery specifications
  - We have written 74 checks (3 lines / check)
- Note: Names have changed since the paper
Summary of Findings

• Applied FATE and DESTINI to **three cloud systems**: HDFS, ZooKeeper, Cassandra
• Found **16** new bugs
• Reproduced **74** bugs
• Problems found
  – Inconsistency
  – Data loss
  – Rack awareness broken
  – Unavailability
Outline

✓ Introduction
  • FATE
  • DESTINI
  • Evaluation
  • Summary
No failures
No failures
No failures
No failures
No failures

Setup Stage Recovery:
Recreate fresh pipeline
No failures

Setup Stage Recovery: Recreate fresh pipeline

Data transfer Stage Recovery: Continue on surviving nodes
No failures

Setup Stage Recovery:
Recreate fresh pipeline

Data transfer Stage Recovery:
Continue on surviving nodes

Bug in Data Transfer Stage Recovery
Failures at DIFFERENT STAGES lead to DIFFERENT FAILURE BEHAVIORS

Goal: Exercise different failure recovery path

Data transfer Stage Recovery: Continue on surviving nodes

Bug in Data Transfer Stage Recovery
FATE

• A failure injection framework
  – target IO points
  – Systematically exploring failure
  – Multiple failures

• New abstraction of failure scenario
  – Remember injected failures
  – Increase failure coverage
FATE

- A failure injection framework
  - target IO points
    - Systematically exploring failure
    - Multiple failures

- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage
FATE

• A failure injection framework
  – target IO points
  – Systematically exploring failure
  – Multiple failures

• New abstraction of failure scenario
  – Remember injected failures
  – Increase failure coverage
FATE

• A failure injection framework
  – target IO points
  – Systematically exploring failure
  – Multiple failures

• New abstraction of failure scenario
  – Remember injected failures
  – Increase failure coverage
FATE

• A failure injection framework
  – target IO points
  – Systematically exploring failure
  – Multiple failures

• New abstraction of failure scenario
  – Remember injected failures
  – Increase failure coverage
FATE

• A failure injection framework
  – target IO points
    – Systematically exploring failure
    – Multiple failures

• New abstraction of failure scenario
  – Remember injected failures
  – Increase failure coverage
<table>
<thead>
<tr>
<th>Fields</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static</strong></td>
<td></td>
</tr>
<tr>
<td>Func. Call</td>
<td>OutputStream.read()</td>
</tr>
<tr>
<td>Source File</td>
<td>BlockReceiver.java</td>
</tr>
<tr>
<td><strong>Dynamic</strong></td>
<td></td>
</tr>
<tr>
<td>Stack Track</td>
<td>...</td>
</tr>
<tr>
<td><strong>Domain specific</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Node 2</td>
</tr>
<tr>
<td>Destination</td>
<td>Node 3</td>
</tr>
<tr>
<td>Net. Message</td>
<td>Data Packet</td>
</tr>
<tr>
<td><strong>Failure</strong></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Crash</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>12348729</td>
</tr>
</tbody>
</table>
How Developers Build Failure ID?

- FATE intercepts all I/Os
- Use aspectJ to collect information at every I/O point
  - I/O buffers (e.g. file buffer, network buffer)
  - Target I/O (e.g. file name, IP address)
- Reverse engineer for domain specific information
## Failure ID

### Fields & Values

<table>
<thead>
<tr>
<th>Fields</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static</strong></td>
<td></td>
</tr>
<tr>
<td>Func. Call</td>
<td>OutputStream.read()</td>
</tr>
<tr>
<td>Source File</td>
<td>BlockReceiver.java</td>
</tr>
<tr>
<td><strong>Dynamic</strong></td>
<td></td>
</tr>
<tr>
<td>Stack Track</td>
<td>...</td>
</tr>
<tr>
<td><strong>Domain specific</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Node 2</td>
</tr>
<tr>
<td>Destination</td>
<td>Node 3</td>
</tr>
<tr>
<td>Net. Message</td>
<td>Data Packet</td>
</tr>
<tr>
<td><strong>Failure</strong></td>
<td>Type</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>12348729</td>
</tr>
<tr>
<td></td>
<td>Crash After</td>
</tr>
</tbody>
</table>
### Failure ID

<table>
<thead>
<tr>
<th>Fields</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static</strong></td>
<td></td>
</tr>
<tr>
<td>Func. Call</td>
<td>OutputStream.read()</td>
</tr>
<tr>
<td>Source File</td>
<td>BlockReceiver.java</td>
</tr>
<tr>
<td><strong>Dynamic</strong></td>
<td></td>
</tr>
<tr>
<td>Stack Track</td>
<td>...</td>
</tr>
<tr>
<td><strong>Domain specific</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Node 2</td>
</tr>
<tr>
<td>Destination</td>
<td>Node 3</td>
</tr>
<tr>
<td>Net. Message</td>
<td>Data Packet</td>
</tr>
<tr>
<td><strong>Failure</strong></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Crash After</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>12348729</td>
</tr>
</tbody>
</table>
Exploring Failure Space
Exploring Failure Space

Exp #1: A
Exploring Failure Space

Exp #1: A

Exp #2: B
Exploring Failure Space

Exp #1: A

Exp #2: B

Exp #3: C
Exploring Failure Space

Exp #1: A

Exp #2: B

Exp #3: C

AB
Exploring Failure Space

Exp #1: A

Exp #2: B

Exp #3: C

AB

AC
Exploring Failure Space

Exp #1: A

Exp #2: B

Exp #3: C

AB

AC

BC
Outline

✓ Introduction
✓ FATE
  • DESTINI
• Evaluation
• Summary
DESTINI

• Enable concise recovery specifications
• Check if expected behaviors match with actual behaviors
• Important elements:
  – Expectations
  – Facts
  – Failure Events
  – Check Timing
• Interpose network and disk protocols
Writing specifications
Writing specifications

“Violation if expectation is different from actual facts”
Writing specifications

“Violation if expectation is different from actual facts”
Writing specifications

“Violation if expectation is different from actual facts”

violationTable() :- expectationTable(), NOT-IN actualTable()
Writing specifications

“Violation if expectation is different from actual facts”

violationTable() :- expectationTable(), NOT-IN actualTable()
Writing specifications

“Violation if expectation is different from actual facts”

violationTable() :- expectationTable(), NOT-IN actualTable()
Writing specifications

“Violation if expectation is different from actual facts”

violationTable() :- expectationTable(), NOT-IN actualTable()

DataLog syntax:
    :- derivation
Writing specifications

“Violation if expectation is different from actual facts”

violationTable() :- expectationTable(), NOT-IN actualTable()

DataLog syntax:
  :- derivation
  ,  AND
Correct recovery

Incorrect Recovery
\text{incorrectNodes}(B, N) \leftarrow \text{expectedNodes}(B, N), \text{NOT-IN} \text{ actualNodes}(B, N);
\[ \text{incorrectNodes}(B, N) :- \text{expectedNodes}(B, N), \text{NOT-IN} \; \text{actualNodes}(B, N); \]

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\text{Expected Nodes} & \text{(Block, Node)} \\
\hline
B & Node 1 \\
B & Node 2 \\
\hline
\end{tabular}
\end{table}
**Incorrect Recovery**

- Expected Nodes
  - (Block, Node)
  - B Node 1
  - B Node 2

**incorrectNodes**(B, N) :- **expectedNodes**(B, N), NOT-IN **actualNodes**(B, N);
\textbf{incorrectNodes}(B, N) :- \textbf{expectedNodes}(B, N), NOT-IN \textbf{actualNodes}(B, N);
\textbf{incorrectNodes}(B, N) :- \textbf{expectedNodes}(B, N), \text{NOT-IN} \ \textbf{actualNodes}(B, N);
incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);
incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);
incorrectNodes(\text{B, N}) \leftarrow \text{expectedNodes(\text{B, N})}, \text{NOT-IN actualNodes(\text{B, N})};
incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);
BUILD EXPECTATIONS

\[
\text{incorrectNodes}(B, N) :\text{ expectedNodes}(B, N), \text{ NOT-IN } \text{actualNodes}(B, N);
\]
IncorrectNodes(Block, Node) :- expectedNodes(Block, Node), NOT-IN actualNodes(Block, Node);

BUILD EXPECTATIONS

CAPTURE FACTS
Building Expectations
Building Expectations

Master

Give me list of nodes for B

[Node 1, Node 2, Node 3]

Client
Building Expectations

expectedNodes(B, N) :- getBlockPipe(B, N);
Building Expectations

\[ \text{expectedNodes}(B, N) :- \ \text{getBlockPipe}(B, N); \]

**Expected Nodes(Block, Node)**

<table>
<thead>
<tr>
<th>Block</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Node 1</td>
</tr>
<tr>
<td>B</td>
<td>Node 2</td>
</tr>
<tr>
<td>B</td>
<td>Node 3</td>
</tr>
</tbody>
</table>
Updating Expectation

<table>
<thead>
<tr>
<th>Expected Nodes (Block, Node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>
DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage), Stage = "Data Transfer", expectedNode(B, N)
DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)
### Updating Expectation

DEL \texttt{expectedNodes}(B, N) :- \texttt{fateCrashNode}(N), \texttt{writeStage}(B, \texttt{Stage}), 
\texttt{Stage} = \text{"Data Transfer"}, \texttt{expectedNode}(B, N)

<table>
<thead>
<tr>
<th>Expected Nodes(Block, Node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>

![Diagram showing the expected nodes and fate crash nodes]
Updating Expectation

DEL `expectedNodes`(B, N) :- `fateCrashNode`(N), `writeStage`(B, Stage),
Stage = "Data Transfer", `expectedNode`(B, N)

- "Client receives all acks from setup stage `writeStage" \rightarrow enter Data Transfer stage"
Updating Expectation

setupAcks (B, Pos, Ack) :- cdpSetupAck (B, Pos, Ack);
goodAcksCnt (B, COUNT<Ack>) :- setupAcks (B, Pos, Ack), Ack == 'OK';
nodesCnt (B, COUNT<Node>) :- pipeNodes (B, , N, );
writeStage (B, Stg) :- nodesCnt (NCnt), goodAcksCnt (ACnt), NCnt == Acnt, Stg := "Data Transfer";

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

• "Client receives all acks from setup stage writeStage" → enter Data Transfer stage
Updating Expectation

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

- “Client receives all acks from setup stage writeStage” → enter Data Transfer stage
Updating Expectation

DEL \texttt{expectedNodes}(B, N) :\texttt{fateCrashNode}(N), \texttt{writeStage}(B, \texttt{Stage}),
\texttt{Stage} = \textquote{Data Transfer}, \texttt{expectedNode}(B, N)

- "Client receives all acks from setup stage writeStage" $\rightarrow$ enter Data Transfer stage
- Precise failure events
Updating Expectation

\[
\text{DEL } \text{expectedNodes}(B, N) :\!\!:\!\!\Rightarrow \text{fateCrashNode}(N), \text{writeStage}(B, \text{Stage}), \\
\text{Stage} = \text{"Data Transfer"}, \text{expectedNode}(B, N)
\]

- “Client receives all acks from setup stage writeStage” \(\Rightarrow\) enter Data Transfer stage
- Precise failure events
  - Different stages \(\Rightarrow\) different recovery behaviors \(\Rightarrow\) different specifications

<table>
<thead>
<tr>
<th>Expected Nodes(Block, Node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>
Updating Expectation

DEL \text{expectedNodes}(B, N) :\rafteCrashNode(N), \text{writeStage}(B, \text{Stage}), \text{Stage} = \text{“Data Transfer”}, \text{expectedNode}(B, N)

- “Client receives all acks from setup stage writeStage” \rafte enter Data Transfer stage
- Precise failure events
  - Different stages \rafte different recovery behaviors \rafte different specifications
  - FATE and DESTINI must work hand in hand

<table>
<thead>
<tr>
<th>Expected Nodes(Block, Node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>
Capture Facts

Correct recovery

Incorrect recovery

M C 1 2 3

M C 1 2 3

B_{gs2} B_{gs1} B_{gs1}
Capture Facts

**Actual Nodes** (B, N) := blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

**Correct recovery**

**Incorrect recovery**

```
M C 1 2 3
---

M C 1 2 3
---

B_gs2 B_gs1 B_gs1
```
Capture Facts

actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

<table>
<thead>
<tr>
<th>blocksLocations(B, N, Gs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>

Correct recovery

Incorrect recovery
Capture Facts

actualNodes(B, N) :-
blocksLocation(B, N, Gs),
latestGenStamp(B, Gs)

blocksLocations(B, N, Gs)

<table>
<thead>
<tr>
<th>B</th>
<th>Node 1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Node 2</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>Node 3</td>
<td>1</td>
</tr>
</tbody>
</table>

latestGenStamp(B, Gs)

| B   | 2 |

Correct recovery

Incorrect recovery
Capture Facts

\[ \text{actualNodes}(B, N) \quad :- \quad \text{blocksLocation}(B, N, Gs), \quad \text{latestGenStamp}(B, Gs) \]

<table>
<thead>
<tr>
<th>blocksLocations($B$, $N$, $Gs$)</th>
<th>latestGenStamp($B$, $Gs$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>Node 1</td>
</tr>
<tr>
<td>$B$</td>
<td>Node 2</td>
</tr>
<tr>
<td>$B$</td>
<td>Node 3</td>
</tr>
</tbody>
</table>

Correct recovery

Incorrect recovery
Capture Facts

\[\text{actualNodes}(B, N) :\text{-}\text{blocksLocation}(B, N, Gs), \text{latestGenStamp}(B, Gs)\]

<table>
<thead>
<tr>
<th>blocksLocations(B, N, Gs)</th>
<th>latestGenStamp(B, Gs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Node 1</td>
</tr>
<tr>
<td>B</td>
<td>Node 2</td>
</tr>
<tr>
<td>B</td>
<td>Node 3</td>
</tr>
</tbody>
</table>

Correct recovery

Incorrect recovery
Capture Facts

\[ \text{actualNodes}(B, N) \quad :\quad \text{blocksLocation}(B, N, Gs), \quad \text{latestGenStamp}(B, Gs) \]

---

**Correct recovery**

- M: Node 1
- C: Node 2
- 1: Node 3

**Incorrect recovery**

- M: Node 3
- C: Node 2
- 1: Node 1

---

**actualNodes(Block, Node)**

<table>
<thead>
<tr>
<th>Block</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Node 1</td>
</tr>
</tbody>
</table>

**blocksLocations(B, N, Gs)**

<table>
<thead>
<tr>
<th>B</th>
<th>Node 1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Node 2</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>Node 3</td>
<td>1</td>
</tr>
</tbody>
</table>

**latestGenStamp(B, Gs)**

| B     | 2 |

---

**M**

- C
- 1
- 2
- 3

**C**

- M
- 1
- 2
- 3

**1**

- M
- C
- 2
- 3

**2**

- M
- C
- 1
- 3

**3**

- M
- C
- 1
- 2
Violation and Check–Timing

\[
\text{incorrectNodes}(B, N) :\text{ expectedNodes}(B, N), \text{ NOT-IN actualNodes}(B, N), \text{ cnpComplete}(B) ;
\]
Violation and Check–Timing

\[
\text{incorrectNodes}(B, N) :- \text{expectedNodes}(B, N), \text{NOT-IN actualNodes}(B, N), cnpComplete(B);
\]

- There is a point in time where recovery is ongoing, thus specifications are violated
Violation and Check–Timing

\[
\text{incorrectNodes}(B, N) :\text{-} \text{expectedNodes}(B, N), \text{NOT-IN actualNodes}(B, N), \text{cnpComplete}(B);
\]

- There is a point in time where recovery is ongoing, thus specifications are violated
- Need precise events to decide when the check should be done
Violation and Check-Timing

\[\text{incorrectNodes}(B, N) :\text{ expectedNodes}(B, N), \text{ NOT-IN actualNodes}(B, N), \text{ cnpComplete}(B);\]

<table>
<thead>
<tr>
<th>IncorrectNodes \hspace{1cm} \text{(Block, Node)}</th>
<th>ExpectedNodes \hspace{1cm} \text{(Block, Node)}</th>
<th>actualNodes \hspace{1cm} \text{(Block, Node)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{B}                     \hspace{1cm} \text{Node 2}</td>
<td>\text{B}                     \hspace{1cm} \text{Node 1}</td>
<td>\text{B}                     \hspace{1cm} \text{Node 1}</td>
</tr>
</tbody>
</table>

- There is a point in time where recovery is ongoing, thus specifications are violated.
- Need precise events to decide when the check should be done.
  - In this example, upon block completion.
Rules

r1 incorrectNodes (B, N) :- cnpComplete (B), expectedNodes (B, N), NOT-IN actualNodes (B, N);
r2 pipeNodes (B, Pos, N) :- getBlkPipe (UFile, B, Gs, Pos, N);
r3 expectedNodes (B, N) :- getBlkPipe (UFile, B, Gs, Pos, N);
r4 DEL expectedNodes (B, N) :- fateCrashNode (N), pipeStage (B, Stg), Stg == 2, expectedNodes (B, N);
r5 setupAcks (B, Pos, Ack) :- cdpSetupAck (B, Pos, Ack);
r6 goodAcksCnt (B, CUUNT<Ack>) :- setupAcks (B, Pos, Ack), Ack == 'OK';
r7 nodesCnt (B, COUNT<Node>) :- pipeNodes (B, , N, );
r8 pipeStage (B, Stg) :- nodesCnt (NCnt), goodAcksCnt (ACnt), NCnt == Acnt, Stg := 2;
r9 blkGenStamp (B, Gs) :- dnpNextGenStamp (B, Gs);
r10 blkGenStamp (B, Gs) :- cnpGetBlkPipe (UFile, B, Gs, , );
r11 diskFiles (N, File) :- fsCreate (N, File);
r12 diskFiles (N, Dst) :- fsRename (N, Src, Dst), diskFiles (N, Src, Type);
r13 DEL diskFiles (N, Src) :- fsRename (N, Src, Dst), diskFiles (N, Src, Type);
r14 fileTypes (N, File, Type) :- diskFiles(N, File), Type := Util.getType(File);
r15 blkMetas (N, B, Gs) :- fileTypes (N, File, Type), Type == metafile, Gs := Util.getGs(File);
r16 actualNodes (B, N) :- blkMetas (N, B, Gs), blkGenStamp (B, Gs);
## Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>\textbf{incorrectNodes} $(B, N)$ :– cnpComplete $(B)$, expectedNodes $(B, N)$, NOT–IN actualNodes $(B, N)$;</td>
</tr>
<tr>
<td>r2</td>
<td>pipeNodes $(B, Pos, N)$ :– getBlkPipe $(UFile, B, Gs, Pos, N)$;</td>
</tr>
<tr>
<td>r3</td>
<td>\textbf{expectedNodes} $(B, N)$ :– getBlkPipe $(UFile, B, Gs, Pos, N)$;</td>
</tr>
<tr>
<td>r4</td>
<td>DEL expectedNodes $(B, N)$ :– fateCrashNode $(N)$, pipeStage $(B, Stg)$, Stg == 2, expectedNodes $(B, N)$;</td>
</tr>
<tr>
<td>r5</td>
<td>setupAcks $(B, Pos, Ack)$ :– cdpSetupAck $(B, Pos, Ack)$;</td>
</tr>
<tr>
<td>r6</td>
<td>goodAcksCnt $(B, \text{COUNT}&lt;Ack&gt;)$ :– setupAcks $(B, Pos, Ack)$, Ack == 'OK';</td>
</tr>
<tr>
<td>r7</td>
<td>nodesCnt $(B, \text{COUNT}&lt;Node&gt;)$ :– pipeNodes $(B, N, N)$;</td>
</tr>
<tr>
<td>r8</td>
<td>pipeStage $(B, Stg)$ :– nodesCnt $(NCnt)$, goodAcksCnt $(ACnt)$, NCnt == Acnt, Stg := 2;</td>
</tr>
<tr>
<td>r9</td>
<td>blkGenStamp $(B, Gs)$ :– dnpNextGenStamp $(B, Gs)$;</td>
</tr>
<tr>
<td>r10</td>
<td>blkGenStamp $(B, Gs)$ :– cnpGetBlkPipe $(UFile, B, Gs, N)$;</td>
</tr>
<tr>
<td>r11</td>
<td>diskFiles $(N, File)$ :– fsCreate $(N, File)$;</td>
</tr>
<tr>
<td>r12</td>
<td>diskFiles $(N, Dst)$ :– fsRename $(N, Src, Dst)$, diskFiles $(N, Src, Type)$;</td>
</tr>
<tr>
<td>r13</td>
<td>DEL diskFiles $(N, Src)$ :– fsRename $(N, Src, Dst)$, diskFiles $(N, Src, Type)$;</td>
</tr>
<tr>
<td>r14</td>
<td>fileTypes $(N, File, Type)$ :– diskFiles $(N, File)$, Type := Util.getType$(File)$;</td>
</tr>
<tr>
<td>r15</td>
<td>blkMetas $(N, B, Gs)$ :– fileTypes $(N, File, Type)$, Type == metafile, Gs := Util.getGs$(File)$;</td>
</tr>
<tr>
<td>r16</td>
<td>\textbf{actualNodes} $(B, N)$ :– blkMetas $(N, B, Gs)$, blkGenStamp $(B, Gs)$;</td>
</tr>
</tbody>
</table>
### Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td><code>incorrectNodes (B, N) :- cnpComplete (B), expectedNodes (B, N), NOT-IN actualNodes (B, N);</code></td>
</tr>
<tr>
<td>r2</td>
<td><code>pipeNodes (B, Pos, N) :- getBlkPipe (UFile, B, Gs, Pos, N);</code></td>
</tr>
<tr>
<td>r3</td>
<td><code>expectedNodes (B, N) :- getBlkPipe (UFile, B, Gs, Pos, N);</code></td>
</tr>
<tr>
<td>r4</td>
<td><code>DEL expectedNodes (B, N) :- fateCrashNode (N), pipeStage (B, Stg), Stg == 2, expectedNodes (B, N);</code></td>
</tr>
<tr>
<td>r5</td>
<td><code>setupAcks (B, Pos, Ack) :- cdpSetupAck (B, Pos, Ack);</code></td>
</tr>
<tr>
<td>r6</td>
<td><code>goodAcksCnt (B, CUUNT&lt;Ack&gt;) :- setupAcks (B, Pos, Ack), Ack == 'OK';</code></td>
</tr>
<tr>
<td>r7</td>
<td><code>nodesCnt (B, COUNT&lt;Node&gt;) :- pipeNodes (B, , N, );</code></td>
</tr>
<tr>
<td>r8</td>
<td><code>pipeStage (B, Stg) :- nodesCnt (NCnt), goodAcksCnt (ACnt), NCnt == Acnt, Stg := 2;</code></td>
</tr>
<tr>
<td>r9</td>
<td><code>blkGenStamp (B, Gs) :- dnpNextGenStamp (B, Gs);</code></td>
</tr>
<tr>
<td>r10</td>
<td><code>blkGenStamp (B, Gs) :- cnpGetBlkPipe (UFile, B, Gs, , );</code></td>
</tr>
<tr>
<td>r11</td>
<td><code>diskFiles (N, File) :- fsCreate (N, File);</code></td>
</tr>
<tr>
<td>r12</td>
<td><code>diskFiles (N, Dst) :- fsRename (N, Src, Dst), diskFiles (N, Src, Type);</code></td>
</tr>
<tr>
<td>r13</td>
<td><code>DEL diskFiles (N, Src) :- fsRename (N, Src, Dst), diskFiles (N, Src, Type);</code></td>
</tr>
<tr>
<td>r14</td>
<td><code>fileTypes (N, File, Type) :- diskFiles(N, File), Type := Util.getType(File);</code></td>
</tr>
<tr>
<td>r15</td>
<td><code>blkMetas (N, B, Gs) :- fileTypes (N, File, Type), Type == metafile, Gs := Util.getGs(File);</code></td>
</tr>
<tr>
<td>r16</td>
<td><code>actualNodes (B, N) :- blkMetas (N, B, Gs), blkGenStamp (B, Gs);</code></td>
</tr>
</tbody>
</table>

- **Capture Facts, Build Expectation from IO events**
  - No need to interpose internal functions
- **Specification Reuse**
  - For the first check, # rules : #check is 16:1
  - Overall, #rules: # check ratio is 3:1
Outline

✓ Introduction
✓ FATE
✓ DESTINI
  • Evaluation
  • Summary
Evaluation

- **FATE**: 3900 lines, **DESTINI**: 1200 lines
- Applied FATE and DESTINI to **three cloud systems**
  - HDFS, ZooKeeper, Cassandra
- **40,000** unique combination of failures
- Found **16** new bugs, reproduced **74 bugs**
- **74** recovery specifications
  - **3** lines / check
Bugs found

- Reduced availability and performance
- Data loss due to multiple failures
- Data loss in log recovery protocol
- Data loss in append protocol
- Rack awareness property is broken
Conclusion

• FATE explores multiple failure systematically
• DESTINI enables concise recovery specifications
• FATE and DESTINI: a unified framework
  – Testing recovery specifications requires a failure service
  – Failure service needs recovery specifications to catch recovery bugs
Thank you!

QUESTIONS?

Berkeley Orders of Magnitude
http://boom.cs.berkeley.edu

The Advanced Systems Laboratory
http://www.cs.wisc.edu/adsl

Downloads our full TR paper from these websites