Reducing Crash Recoverability to Reachability

Eric Koskinen
Yale University

Junfeng Yang
Columbia University

Principles of Programming Languages
St. Petersburg, Florida
20 January 2016
We are pretty good at writing programs...

```csharp
using UnityEngine;
using System.Collections;

public class [code] : MonoBehaviour {
    public RaycastHit hit = new RaycastHit();
    public Ray ray;

    // Update is called once per frame
    void Update () {
        if(Input.GetMouseButtonDown(0)) {
            ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            hit = new RaycastHit();

            if(Physics.Raycast(ray, out hit)) {
                if(hit.transform.tag == "") {
                    StartCoroutine("StartLevel", hit.audio);
                } else {
                    StartCoroutine("QuitGame", hit.audio);
                }
            }
        }
    }
}
We are pretty good at writing programs...
We are pretty good at writing programs...
We are pretty good at writing programs...
We are pretty good at writing programs…
We are pretty good at writing programs...

```csharp
using UnityEngine;
using System.Collections;

public class [code] : MonoBehaviour {
    public RaycastHit hit = new RaycastHit();
    public Ray ray;

    // Update is called once per frame
    void Update () {
        if(Input.GetMouseButtonDown(0)){
            ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            hit = new RaycastHit();
            if(Physics.Raycast(ray, out hit)){
                if(hit.transform.tag == " instability"){
                    StartCoroutine("StartLevel");
                }
                if(hit.transform.tag == ""){
                    StartCoroutine("QuitGame");
                }
            }
        }
    }
}
```
1. *What do we mean by crash and recovery?*

2. *Can we prove (automatically) that a program recovers from a crash?*

3. *Does this actually work on real examples?*
What do we mean by crash and recovery?
**What do we mean by crash and recovery?**

1. Boot machine
2. Establish program env.
3. Execute program
4. Crash mid-execution
5. Re-boot computer
6. Execute Recovery Script
7. Establish program env.
8. Re-execute program
What do we mean by crash and recovery?

1. Boot machine
2. Establish program env.
3. Execute program
4. Crash mid-execution
5. Re-boot computer
6. Execute Recovery Script
7. Establish program env.
8. Re-execute program
What do we mean by crash and recovery?

Initial State
1. Boot machine
2. Establish program env.

Crash
3. Execute program
4. Crash mid-execution
5. Re-boot computer
6. Execute Recovery Script
7. Establish program env.
8. Re-execute program
What do we mean by crash and recovery?

1. Boot machine
2. Establish program env.
3. Execute program
4. Crash mid-execution
5. Re-boot computer
6. Execute Recovery Script
7. Establish program env.
8. Re-execute program
What do we mean by crash and recovery?

0 in = open(input)
1 read(in,buf);
2 CRASH
3 ...

CRASH
```python
in = open(input)
out = open(output, 0_CREAT | 0_WRONLY | 0_WRONLY)
write(out, "A")
```

CRASH

...
```python
0 in = open(input)
1 out = open(output, 0_CREAT | 0_WRONLY | 0_TRUNC)
2 write(out, "A")
3 **CRASH**
4 ...
```
Is this new trace “ok”?

```
0 in = open(input)
1 out = open(output, O_CREAT | O_WRONLY | O_TRUNC)
2 write(out, "A")
3 CRASH
4 ...
```
Program states
**Program states**

With possibility of crashes . . . possibility of new behaviors

- If the program crashes, when it is re-executed, should not have new behaviors that weren’t in the original program.
- Matches what the program does
- Program must handle new initial states
Program states

With possibility of crashes . . . possibility of new behaviors

Would like to prove that they are already included in the original program.

\[ (\hat{\mathcal{C}}) \subseteq (\mathcal{C}) \]

Therefore . . .

We can use the original program as the specification for how the program should behave in the presence of crashes.
Program states

With possibility of crashes . . . possibility of new behaviors

Would like to prove that they are already included in the original program.

Therefore . . .

We can use the original program as the specification for how the program should behave in the presence of crashes.
Non-determinism

```c
0    in = open(input)
1    out = open(output, O_CREAT
2        | O_WRONLY | O_TRUNC)
3    if(rand()) {
4        write(out,"A");
5            CRASH
6    } else {
7        write(out,"B");
8    }
9    ...
```
Non-determinism

```
0 in = open(input)
1 out = open(output, O_CREAT | O_WRONLY | O_TRUNC)
2 if(rand()) {
    write(out,"A");
    CRASH
}
3 ... else {
4   write(out,"B");
5 }
```

Recovery Scripts

Described in the Paper
Checkpoints

Described in the Paper

```python
in = open(input)
out = creat(output)
write(out,"pre");
fsync_commit(out);
chkpt:
if(rand()) {
    CRASH; RECOVER()
...
}
else {
    ...
}
RECOVER() {
    if(committed) {
        in=open(input);
        out=open(output);
        goto chkpt;
    }
}
```
Hierarchy of Crash Recoverability

\[ R_0 \quad \text{0-recoverability} \]
\[ R_1 \quad \text{1-recoverability} \]
\[ R_N \quad \text{N-recoverability} \]
**Hierarchy of Crash Recoverability**

\[ R_0 \quad \text{0-recoverability} \]

\[ R_1 \quad \text{1-recoverability} \]

\[ R_N \quad \text{N-recoverability} \]

\[ Labeled \ transition \ system \ M = (\Sigma, \Lambda, \Gamma, \sigma_0) \]

- States \( \Sigma \)
- Labels \( \Lambda \)
- Transitions \( \Gamma \subseteq \Sigma \times \Lambda \times \Sigma \) denoted \( \sigma \xrightarrow{\lambda} \sigma' \), and
- Initial state \( \sigma_0 \)
Hierarchy of Crash Recoverability

\[ R_0 \] 0-recoverability
\[ R_1 \] 1-recoverability
\[ R_N \] N-recoverability

Labeled transition system \( M = (\Sigma, \Lambda, \Gamma, \sigma_0) \)

- States \( \Sigma \)
- Labels \( \Lambda \)
- Transitions \( \Gamma \subseteq \Sigma \times \Lambda \times \Sigma \) denoted \( \sigma \xrightarrow{\lambda} \sigma' \), and
- Initial state \( \sigma_0 \)

Trace \( \pi \) is a sequence \( \sigma_0 \xrightarrow{\lambda_0} \sigma_1 \xrightarrow{\lambda_1} \sigma_2 \cdots \) s.t. \( \forall i \geq 0. \sigma_i \xrightarrow{\lambda_i} \sigma_{i+1} \).

Notations \( \sigma_0 \xrightarrow{\lambda_0,\lambda_1,\ldots} \cdots \) and \( \sigma_0 \xrightarrow{\Lambda_a} \cdots \)

- \( \Pi(M,S) \) to mean the set of all traces of \( M \) from \( S \)
- Write \( \sigma \xrightarrow{\lambda} \pi \) when we want to talk about the prefix \( \sigma \xrightarrow{\lambda} \) of a trace \( \sigma \xrightarrow{\lambda} \pi \).
- Write \( \pi^0 \) and \( \pi_n \)
Hierarchy of Crash Recoverability

\( R_0 \) 0-recoverability

\( R_1 \) 1-recoverability

\( R_N \) N-recoverability

Simulation

\[
\forall \lambda_1. \sigma_1 \xrightarrow{\lambda_1} \sigma_1'. \exists \lambda_2. \sigma_2 \xrightarrow{\lambda_2} \sigma_2' \land \lambda_1 \equiv \lambda_2 \land \sigma_1' \preceq \sigma_2' \quad \sigma_1 \preceq \sigma_2
\]
Hierarchy of Crash Recoverability

\[ R_0 \quad 0\text{-recoverability} \]
\[ R_1 \quad 1\text{-recoverability} \]
\[ R_N \quad N\text{-recoverability} \]

Recoverability

\[
\begin{align*}
\sigma & \xrightarrow{\lambda} \pi^0 & R_N(\pi) \\
R_0(\pi) & \quad R_N(\sigma \xrightarrow{\lambda} \pi) \\
\exists k < n, j \geq 0. \pi_j^0 & \preceq \sigma_k & R_{N-1}(\sigma_0 \xrightarrow{\lambda_0\cdots\lambda_{k-1}} \pi_j) \\
R_N(\sigma_0 \xrightarrow{\lambda_0\cdots\lambda_{n-1}} \sigma_n \xrightarrow{\lambda_{\text{crash}}} \pi) & 
\end{align*}
\]
**Hierarchy of Crash Recoverability**

$\mathcal{R}_0 \quad 0$-recoverability

$\mathcal{R}_1 \quad 1$-recoverability

$\mathcal{R}_N \quad N$-recoverability

$\mathcal{R} \quad \infty$-recoverability

\[
\frac{\sigma \xrightarrow{\lambda} \pi^0 \quad \mathcal{R}(\pi)}{\mathcal{R}(\sigma \xrightarrow{\lambda} \pi)}
\]

\[
\mathcal{R}(\sigma_0 \xrightarrow{\lambda_0\ldots\lambda_{k-1}} \pi_j)
\]

\[
\exists k < n, j \geq 0. \pi_j^0 \preceq \sigma_k \land \mathcal{R}(\sigma_0 \xrightarrow{\lambda_0\ldots\lambda_{n-1}} \sigma_n \xrightarrow{\lambda_{\text{crash}}} \pi)
\]
1. *What do we mean by crash and recovery?*

2. *Can we prove (automatically) that a program recovers from a crash?*

3. *Does this actually work on real examples?*
Key Idea: Transformation...
Key Idea: Transformation...

\[ A \xrightarrow{\beta^I} A_{\mathcal{E}} \]

... reduce to reachability:

\[ A_{\mathcal{E}} \] cannot reach \( q_{err} \)

\[ \Rightarrow \]

\[ A \] is crash-recoverable
Key Idea: Transformation...

\[ A \xrightarrow{\bullet} A_{\mathcal{E}} \]

... reduce to reachability:

\[ \exists \mathcal{D}, A_{\mathcal{E}}^{\mathcal{D}} \text{ cannot reach } q_{err} \]

\[ \Rightarrow \]

\[ A \text{ is crash-recoverable} \]
∃D. \mathcal{A}_{\mathcal{E}} \text{ cannot reach } q_{err} \Rightarrow \mathcal{A} \text{ is crash-recoverable}
Main Theorem

$$\exists D. A^D \text{ cannot reach } q_{err} \Rightarrow A \text{ is crash-recoverable}$$
Main Theorem

\[ \exists D. A_\mathcal{E} \text{ cannot reach } q_{err} \Rightarrow A \text{ is crash-recoverable} \]
Main Theorem

∃D. ADanielle cannot reach qerr ⇒ A is crash-recoverable
Main Theorem

\[ \exists \mathcal{D}, \mathcal{A}^\mathcal{D}_\mathcal{E} \text{ cannot reach } q_{err} \implies \mathcal{A} \text{ is crash-recoverable} \]
Main Theorem

\[ \exists D. \mathcal{A}^D \text{ cannot reach } q_{err} \Rightarrow \mathcal{A} \text{ is crash-recoverable} \]
∃ cannot reach $q_{err}$ is crash-recoverable

Execute uncrashed snapshot
`s := n(`b);`

And recovered state
`s := n(b);`

$\sigma$
\[ \exists D. A^D \] cannot reach \( q_{err} \) \( \Rightarrow \) \( A \) is crash-recoverable
Main Theorem

\[ \exists D. A_{D}^{P} \text{ cannot reach } q_{err} \Rightarrow A \text{ is crash-recoverable} \]
For $\mathcal{A} = \langle Q, q_0, 0p, X, 0p, \rightarrow \rangle$ let $\mathcal{A}_\mathcal{E}^\mathcal{D} = \langle Q_\mathcal{E}, q_0^\mathcal{E}, X_\mathcal{E}, 0p_\mathcal{E}, \rightarrow_\mathcal{E} \rangle$,

$Q_\mathcal{E} = Q \cup \{q_{\text{err}}\} \cup (q_{i\rightarrow q_j})_\epsilon \rightarrow \{q_{ij}\}$

$q_0^\mathcal{E} = q_0$

$X_\mathcal{E} = \{\text{\textunderscore CR}\} \cup 'X \cup X \cup \bigcup_i 'X_i \bigcup \tilde{X}$

$\rightarrow_\mathcal{E} = \{q_{\text{err}} \xrightarrow{\epsilon} q_{\text{err}}\} \cup$

\[
\begin{align*}
\bigcup (q_{i\rightarrow q_j})_\epsilon \
\{\text{\textunderscore CR}\} Y := m(X); 'X_i := X & \rightarrow q_j \\
q_i \quad \{\text{\textunderscore CR}\} Y := m(X); 'Y := m('X) & \rightarrow q_{ij} \\
q_i \quad \{Y = 'Y\} & \rightarrow q_{ij} \\
q_i \quad \{Y \neq 'Y\} & \rightarrow q_{\text{err}} \\
\bigcup q_i \in Q, q_k \in Q_{\text{rcv}} \cdot q_i \quad \text{crash}(X); \text{\textunderscore CR} := \text{true}; 'X := 'X_k; \tilde{X} := X & \rightarrow \hat{q}_k \\
\bigcup \hat{q}_k, \hat{q}_{k'} \in Q_{\text{rcv}} \cdot \hat{q}_k \quad \{(\hat{X}, X) \in D\} & \rightarrow q_{\text{err}} \\
\hat{q}_k \quad \hat{X} := X; Y := m(X) & \rightarrow \hat{q}_{k'} \\
\hat{q}_k \quad \hat{Y} := m(X) & \rightarrow \hat{q}_k
\end{align*}
\]

**Main Theorem**

$\exists \mathcal{D}. \mathcal{A}_\mathcal{E}^\mathcal{D}$ cannot reach $q_{\text{err}} \Rightarrow \mathcal{A}$ is crash-recoverable
1. **What do we mean by crash and recovery?**

2. **Can we prove (automatically) that a program recovers from a crash?**

3. **Does this actually work on real examples?**
```
int cur = open("passwd");
char *buf = read(cur, MAX_BUF);
close(cur);
if(-strstr(buf,newUser,length(newUser)) {
    // add newUser
    int new = creat("passwd.new");
    ...
    copy(cur,new);
    write(new,newUser);
    write(new,newPassword);
    close(new);
    fsync(new);
    unlink("passwd");
    rename("passwd.new","passwd");
}
cur = open ("passwd");
buf = read(cur, MAX_BUF);
```
Notes

- Built on CPAchecker
- Compiler Macros
- Model of the filesystem with arrays and integers
- Copying with arrays

```c
write(new,newUser);
write(new,newPassword);
close(new);
fsync(new);
unlink("passwd");
rename("passwd.new","passwd");
}
cur = open ("passwd");
buf = read(cur, MAX_BUF);
```
### Benchmarks

- Simple examples from earlier in this talk
- Examples of crash recovery protocols of real-world examples [Pillai et al. OSDI’14]

- Google’s LevelDB
- PostgreSQL - Used by 30% of tech companies
- SQLite - Used by probably every Android app (1B users)
- VMware
- ZooKeeper - Distributed applications, used by Yahoo

<table>
<thead>
<tr>
<th>#</th>
<th>System</th>
<th>Var.</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fig 1, Ex 2</td>
<td>-</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>Fig 1, Ex 2</td>
<td>A</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>Fig 1, Ex 3</td>
<td>-</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>Fig 1, Ex 3</td>
<td>A</td>
<td>61</td>
</tr>
<tr>
<td>5</td>
<td>Fig 1, Ex 4</td>
<td>-</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Fig 1, Ex 4</td>
<td>A</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>Fig 1, Ex 5</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>8</td>
<td>Fig 1, Ex 5</td>
<td>A</td>
<td>88</td>
</tr>
<tr>
<td>9</td>
<td>GDBM create</td>
<td>-</td>
<td>87</td>
</tr>
<tr>
<td>10</td>
<td>GDBM create</td>
<td>A</td>
<td>83</td>
</tr>
<tr>
<td>11</td>
<td>GDBM create</td>
<td>B</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>GDBM create</td>
<td>C</td>
<td>85</td>
</tr>
<tr>
<td>13</td>
<td>GDBM create</td>
<td>D</td>
<td>84</td>
</tr>
<tr>
<td>14</td>
<td>Level DB</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>Level DB</td>
<td>A</td>
<td>58</td>
</tr>
<tr>
<td>16</td>
<td>Level DB</td>
<td>B</td>
<td>52</td>
</tr>
<tr>
<td>17</td>
<td>Level DB</td>
<td>C</td>
<td>58</td>
</tr>
<tr>
<td>18</td>
<td>LMDB</td>
<td>-</td>
<td>51</td>
</tr>
<tr>
<td>19</td>
<td>LMDB</td>
<td>A</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>LMDB</td>
<td>B</td>
<td>51</td>
</tr>
<tr>
<td>21</td>
<td>LMDB</td>
<td>C</td>
<td>50</td>
</tr>
<tr>
<td>22</td>
<td>PostgreSQL</td>
<td>-</td>
<td>68</td>
</tr>
<tr>
<td>23</td>
<td>PostgreSQL</td>
<td>A</td>
<td>68</td>
</tr>
<tr>
<td>24</td>
<td>PostgreSQL</td>
<td>B</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>PostgreSQL</td>
<td>C</td>
<td>59</td>
</tr>
<tr>
<td>26</td>
<td>SQLite</td>
<td>-</td>
<td>65</td>
</tr>
<tr>
<td>27</td>
<td>SQLite</td>
<td>A</td>
<td>69</td>
</tr>
<tr>
<td>28</td>
<td>VMware</td>
<td>-</td>
<td>68</td>
</tr>
<tr>
<td>29</td>
<td>VMware</td>
<td>A</td>
<td>68</td>
</tr>
<tr>
<td>30</td>
<td>VMware</td>
<td>B</td>
<td>65</td>
</tr>
<tr>
<td>31</td>
<td>ZooKeeper</td>
<td>-</td>
<td>55</td>
</tr>
<tr>
<td>32</td>
<td>ZooKeeper</td>
<td>A</td>
<td>56</td>
</tr>
<tr>
<td>#</td>
<td>System</td>
<td>Var.</td>
<td>LOC</td>
</tr>
<tr>
<td>----</td>
<td>------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>1</td>
<td>Fig 1, Ex 2</td>
<td>-</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>Fig 1, Ex 2</td>
<td>A</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>Fig 1, Ex 3</td>
<td>-</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>Fig 1, Ex 3</td>
<td>A</td>
<td>61</td>
</tr>
<tr>
<td>5</td>
<td>Fig 1, Ex 4</td>
<td>-</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Fig 1, Ex 4</td>
<td>A</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>Fig 1, Ex 5</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>8</td>
<td>Fig 1, Ex 5</td>
<td>A</td>
<td>88</td>
</tr>
<tr>
<td>9</td>
<td>GDBM create</td>
<td>-</td>
<td>87</td>
</tr>
<tr>
<td>10</td>
<td>GDBM create</td>
<td>A</td>
<td>83</td>
</tr>
<tr>
<td>11</td>
<td>GDBM create</td>
<td>B</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>GDBM create</td>
<td>C</td>
<td>85</td>
</tr>
<tr>
<td>13</td>
<td>GDBM create</td>
<td>D</td>
<td>84</td>
</tr>
<tr>
<td>14</td>
<td>Level DB</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>Level DB</td>
<td>A</td>
<td>58</td>
</tr>
<tr>
<td>16</td>
<td>Level DB</td>
<td>B</td>
<td>52</td>
</tr>
<tr>
<td>17</td>
<td>Level DB</td>
<td>C</td>
<td>58</td>
</tr>
<tr>
<td>18</td>
<td>LMDB</td>
<td>-</td>
<td>51</td>
</tr>
<tr>
<td>19</td>
<td>LMDB</td>
<td>A</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>LMDB</td>
<td>B</td>
<td>51</td>
</tr>
<tr>
<td>21</td>
<td>LMDB</td>
<td>C</td>
<td>50</td>
</tr>
<tr>
<td>22</td>
<td>PostgreSQL</td>
<td>-</td>
<td>68</td>
</tr>
<tr>
<td>23</td>
<td>PostgreSQL</td>
<td>A</td>
<td>68</td>
</tr>
<tr>
<td>24</td>
<td>PostgreSQL</td>
<td>B</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>PostgreSQL</td>
<td>C</td>
<td>59</td>
</tr>
<tr>
<td>26</td>
<td>SQLite</td>
<td>-</td>
<td>65</td>
</tr>
<tr>
<td>27</td>
<td>SQLite</td>
<td>A</td>
<td>69</td>
</tr>
<tr>
<td>28</td>
<td>VMware</td>
<td>-</td>
<td>68</td>
</tr>
<tr>
<td>29</td>
<td>VMware</td>
<td>A</td>
<td>68</td>
</tr>
<tr>
<td>30</td>
<td>VMware</td>
<td>B</td>
<td>65</td>
</tr>
<tr>
<td>31</td>
<td>ZooKeeper</td>
<td>-</td>
<td>55</td>
</tr>
<tr>
<td>32</td>
<td>ZooKeeper</td>
<td>A</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>System</td>
<td>Var.</td>
<td>LOC</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>1</td>
<td>Fig 1, Ex 2</td>
<td>-</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>Fig 1, Ex 2</td>
<td>A</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>Fig 1, Ex 3</td>
<td>-</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>Fig 1, Ex 3</td>
<td>A</td>
<td>61</td>
</tr>
<tr>
<td>5</td>
<td>Fig 1, Ex 4</td>
<td>-</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Fig 1, Ex 4</td>
<td>A</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>Fig 1, Ex 5</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>8</td>
<td>Fig 1, Ex 5</td>
<td>A</td>
<td>88</td>
</tr>
<tr>
<td>9</td>
<td>GDBM create</td>
<td>-</td>
<td>87</td>
</tr>
<tr>
<td>10</td>
<td>GDBM create</td>
<td>A</td>
<td>83</td>
</tr>
<tr>
<td>11</td>
<td>GDBM create</td>
<td>B</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>GDBM create</td>
<td>C</td>
<td>85</td>
</tr>
<tr>
<td>13</td>
<td>GDBM create</td>
<td>D</td>
<td>84</td>
</tr>
<tr>
<td>14</td>
<td>Level DB</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>Level DB</td>
<td>A</td>
<td>58</td>
</tr>
<tr>
<td>16</td>
<td>Level DB</td>
<td>B</td>
<td>52</td>
</tr>
<tr>
<td>17</td>
<td>Level DB</td>
<td>C</td>
<td>58</td>
</tr>
<tr>
<td>18</td>
<td>LMDB</td>
<td>-</td>
<td>51</td>
</tr>
<tr>
<td>19</td>
<td>LMDB</td>
<td>A</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>LMDB</td>
<td>B</td>
<td>51</td>
</tr>
<tr>
<td>21</td>
<td>LMDB</td>
<td>C</td>
<td>50</td>
</tr>
<tr>
<td>22</td>
<td>PostgreSQL</td>
<td>-</td>
<td>68</td>
</tr>
<tr>
<td>23</td>
<td>PostgreSQL</td>
<td>A</td>
<td>68</td>
</tr>
<tr>
<td>24</td>
<td>PostgreSQL</td>
<td>B</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>PostgreSQL</td>
<td>C</td>
<td>59</td>
</tr>
<tr>
<td>26</td>
<td>SQLite</td>
<td>-</td>
<td>65</td>
</tr>
<tr>
<td>27</td>
<td>SQLite</td>
<td>A</td>
<td>69</td>
</tr>
<tr>
<td>28</td>
<td>VMware</td>
<td>-</td>
<td>68</td>
</tr>
<tr>
<td>29</td>
<td>VMware</td>
<td>A</td>
<td>68</td>
</tr>
<tr>
<td>30</td>
<td>VMware</td>
<td>B</td>
<td>65</td>
</tr>
<tr>
<td>31</td>
<td>ZooKeeper</td>
<td>-</td>
<td>55</td>
</tr>
<tr>
<td>32</td>
<td>ZooKeeper</td>
<td>A</td>
<td>56</td>
</tr>
</tbody>
</table>
Related Work

• Chen et al. Using Crash Hoare logic for certifying the FSCQ file system. SOSP 2015
  • Broadly complementary: verified FS versus verifying user-level programs
  • Specifically different: we focus on automation while they focus on proof modularity/reusability (require user-provided CHL specifications and user help in proof obligations)

  • Novel logic explicitly tracking volatile/persistent
  • Support concurrency, Not automated

• Gardner et al. Local Reasoning for the POSIX filesystem. ESOP 2014.

• Ridge et al. SibylFS: formal specification and oracle-based testing for POSIX and real-world file systems. SOSP 2015
Contributions

- **Specification** - Definitions on what it means for a crash to recover
- **Automatic Verification** - Reduction to automaton reachability
- **Evaluation** - Proved recoverability of commit protocols from real systems (SQLite, LevelDB, ZooKeeper, etc.)
- **Open Challenges** -
  - Code scope, O/S layers
  - N-recoverability, infinite-recoverability
  - Timing - Does recovery happen promptly?
  - Concurrency
Thank you!