Understanding and Generating High Quality Patches for Concurrency Bugs

Haopeng Liu, Yuxi Chen and Shan Lu
What are concurrency bugs

• Synchronization mistakes in multi-threaded programs
What are concurrency bugs

- Synchronization mistakes in multi-threaded programs
What are concurrency bugs

- Synchronization mistakes in multi-threaded programs

```c
//child thread
if(...){
    unlock(f->mut);
    return;
}

//parent thread
f->mut = NULL;
```
What are concurrency bugs

- Synchronization mistakes in multi-threaded programs

```
//child thread
if (...){
    unlock(f->mut);
    return;
}

//parent thread
f->mut = NULL;
```
Concurrency bugs need to be fixed

• Prevalence
  – Widely exist in multi-threaded programs

• Seriousness
  – Cause real world disasters

• Difficulty
  – Time-consuming to fix [LuASPLOS’08]
  – Patches are the most error-prone [YinFSE’11]

• 2X times more often to be buggy than memory/semantic bugs
Concurrency bug automated fixing

• Observation:
  – Program is correct in most timing interleavings.

• Solution:
  – Only need to remove some bad timing.
    • add lock-related synchronization [AFix PLDI’11] [CFix OSDI’12] [Grail FSE’14].
An automated fixing example

//child thread
if(...){
    unlock(f->mut);
+   lock(L);
+   signal(con);
+   unlock(L);
return;
}
+ lock(L);
+ signal(con);
+ unlock(L);

//parent thread
+ lock(L)
+ while(...){
+    wait(con, L);
+ }  
+ unlock (L)

f->mut = NULL;

• Overall:
  – 9 new synchronizations
  – 1 new while loop

Reality is even worse!
The manual patch

//child thread
if(...){
  unlock(f->mut);
  return;
}

//parent thread
+ thread_join(...);
  f->mut = NULL;

• Overall:
  – New sync.: 1 VS. 9
  – New while loop: 0 VS. 1

Simpler patch → Easier to read and maintain

Can we automatically generate such simple patches?
Contributions

• Manual patch study
  – 77 real-world concurrency bugs
  – The limitation of existing techniques
  – Guidance to design new tools

• HFix

[1] Guoliang Jin etc. AFix. In PLDI’11
Outline

• Motivation
• Manual patch study
• HFix
• Evaluation
• Conclusion
Research Questions

• Q1: Do manual patches mostly use lock-related synchronizations?
• Q2: Do manual patches mostly disable buggy timing by adding synchronization ops?
Manual patch study methodology\textsuperscript{[1]}

<table>
<thead>
<tr>
<th>App.</th>
<th>Source</th>
<th>Description</th>
<th>#. Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td></td>
<td>Web Server</td>
<td>13</td>
</tr>
<tr>
<td>Mozilla</td>
<td>Lu ASPLOS’08</td>
<td>Browser Suite</td>
<td>41</td>
</tr>
<tr>
<td>MySQL</td>
<td></td>
<td>Database Server</td>
<td>12</td>
</tr>
<tr>
<td>OpenOffice</td>
<td></td>
<td>Office Suite</td>
<td>5</td>
</tr>
<tr>
<td>Misc</td>
<td>Recent papers[2-4]</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>77</td>
</tr>
</tbody>
</table>

• Study developer patches & on-line discussion

\textsuperscript{[1]} Threads to validity are listed in paper.
\textsuperscript{[2]} Guoliang Jin etc. AFix. In PLDI’11
\textsuperscript{[3]} Guoliang Jin etc. CFix. In OSDI’12
\textsuperscript{[4]} Wei Zhang etc. ConSeq. In ASPLOS’11
Q1: How many patches use lock?

Use Sync. primitives?

- Yes
- None
Q1: How many patches use lock?

Use Sync. primitives?
- Yes
- None

Synchronization primitives
- Lock/Con. Var
- Create/Join
- Misc.

AV patches
- Lock/Con. Var
- Create/Join
- Misc.

OV patches
- Lock/Con. Var
- Create/Join
- Misc.
Q1: How many patches use lock?

Use Sync. primitives?
- Yes
- None

Synchronization primitives
- Lock/Con. Var
- Create/Join
- Misc.

AV patches:  
- Lock/Con. Var
- Create/Join
- Misc.

OV patches:  
- Lock/Con. Var
- Create/Join
- Misc.
Q1: How many patches use lock?

Use Sync. primitives?
- Yes
- None

Synchronization primitives
- Lock/Con. Var
- Create/Join
- Misc.

AV patches
- Lock/Con. Var
- Misc.

OV patches
- Create/Join
- Misc.
Q1: How many patches use lock?

Use Sync. primitives?
- Yes
- None

Synchronization primitives:
- Lock/Con. Var
- Create/Join
- Misc.

AV patches vs. OV patches.
Q1: How many patches use lock?

**Use Sync. primitives?**
- Yes
- None

**Synchronization primitives**
- Lock/Con. Var
- Create/Join
- Misc.

**Graphs**
- AV patches
- OV patches

**Diagram**
- Thread\_pc
- Thread\_r
Q1: How many patches use lock?

Use Sync. primitives?
- Yes
- None

Synchronization primitives
- Lock/Con. Var
- Create/Join
- Misc.

AV patches
- Misc.
- Create/Join

OV patches
- Misc.
- Create/Join
Q1: How many patches use lock?

Use Sync. primitives?

- Yes
- None

Synchronization primitives

- Lock/Con. Var
- Create/Join
- Misc.

AV patches

- None

OV patches

- Lock/Con. Var
- Create/Join
- Misc.

Thread\textsubscript{pc} → Thread\textsubscript{r}

\text{Thread}_A \rightarrow \text{Thread}_B
Q1 implication

- Use non-lock synchronization to fix OV.

![Synchronization primitives chart]

- AV patches
- OV patches
Q1 implication

• Use non-lock synchronization to fix OV.
Q2: What are developers’ fix strategy?
Q2: What are developers’ fix strategy?

[1] Guoliang Jin etc. AFix. In PLDI’11
Q2: What are developers’ fix strategy?

[1] Jeff Huang etc. Execution Privatization of Scheduler-Oblivious Concurrent Programs. In OOPSLA’11
Q2: What are developers’ fix strategy?
Q2: What are developers’ fix strategy?
Q2 implication

• Add-Sync widely applies, but often not best

• Need to automate:
  – Move synchronization
  – Bypass
  – Tolerate

• Different bugs are suitable different strategies
Outline

• Motivation
• Manual patch studying
• HFix
  – HFixJoin
  – HFixmove
• Evaluation
• Conclusion
HFix overview

- **Goal:**
  - Fix bugs with simple patches
  - Do not introduce new bugs
HFixjoin Intuition

• Which kind of bugs can be fixed by HFixjoin?
HFixjoin Intuition

• Which kind of bugs can be fixed by HFixjoin?
HFixjoin

• Step 1: Suitability checking
  – Goal: do not introduce new bugs.

• Step 2: Insert join synchronization
  – Goal: fix the bug
Suitability checking (static analysis)

• Is ThreadA a joinable child thread of ThreadB?
• Only one ThreadB?
• Is ThreadA already joined?
• Will there be deadlock/severe performance slow-down?
Suitability checking (static analysis)

- Is ThreadA a joinable child thread of ThreadB?
- Only one ThreadB?
- Is ThreadA already joined?
- Will there be deadlock/severe performance slow-down?

```c
//parent thread
+ thread_join(…);
fp = NULL;
signal(condV);

//child thread
fputs(fp,…);
wait(condV);
```
Insert join synchronization

• Insert join before operation B
• Patch Example

```
//child thread
if(...) {
    unlock(f->mut);
    return;
}

//parent thread
pthread_create(&tid,...)
...;

f->mut = NULL;
```
Insert join synchronization

- Insert join before operation B
- Patch Example

```c
//child thread
if (...) {
    unlock(f->mut);
    return;
}

//parent thread
pthread_create(&tid,...)
    tmp = tid;
    ...
    pthread_join(tmp);
    f->mut = NULL;
```
HFixmove Intuition

• What kind of bugs can be fixed by HFixmove?

(a) \text{Move}_{\text{create}} \text{ fixes OV bug move-up A or move-down S}
HFixmove Intuition

• What kind of bugs can be fixed by HFixmove?

(a) $\text{Move}_{\text{create}}$ fixes OV bug move-up A or move-down S
(b) $\text{Move}_{\text{join}}$ fixes OV bug move-up S or move-down B
HFixmove Intuition

• What kind of bugs can be fixed by HFixmove?

(a) Move_{create} fixes OV bug move-up A or move-down S
(b) Move_{join} fixes OV bug move-up S or move-down B
(c) Move_{unlock} fixes AV bug move-up c or move-down S
(d) Move_{lock} fixes AV bug move-up S or move-down p
Step 1: identify two operations to move

(a) **Move**\textsubscript{create} fixes OV bug move-up A or move-down S

(b) **Move**\textsubscript{join} fixes OV bug move-up S or move-down B

(c) **Move**\textsubscript{unlock} fixes AV bug move-up c or move-down S

(d) **Move**\textsubscript{lock} fixes AV bug move-up S or move-down p
Step 1: identify two operations to move

(a) $\text{Move}_{\text{create}}$ fixes OV bug move-up $A$ or move-down $S$

(b) $\text{Move}_{\text{join}}$ fixes OV bug move-up $S$ or move-down $B$

(c) $\text{Move}_{\text{unlock}}$ fixes AV bug move-up $c$ or move-down $S$

(d) $\text{Move}_{\text{lock}}$ fixes AV bug move-up $S$ or move-down $p$
Step 2: decide the moving destination $X^*$

- Goal
  
  $\# \text{ of executions of } X = \# \text{ of executions of } X^*$
Strawman solution doesn’t work

# of executions of X ≠ # of executions of X*

1 ≠ 0

Originally, X executes once. After moving, X* does not execute.
Strawman solution doesn’t work

\# of executions of X ≠ \# of executions of X*

1 ≠ N

Originally, X executes once. After moving, X* executes more than once.
HFixmove demonstration

- Guarantee: X* will execute exactly once before Y.

* X and Y are inside the same function; neither is in loop
HFixmove demonstration

• Guarantee: X* will execute exactly once before Y.

* X and Y are inside the same function; neither is in loop
HFixmove demonstration

- Guarantee: X* will execute exactly once before Y.

* X and Y are inside the same function; neither is in loop
HFixmove demonstration

• Guarantee: X* will execute exactly once before Y.

* X and Y are inside the same function; neither is in loop
HFixmove demonstration

• Guarantee: X* will execute exactly once before Y.

* X and Y are inside the same function; neither is in loop
HFixmove demonstration

- Guarantee: X* will execute exactly once before Y.

* X and Y are inside the same function; neither is in loop
Step 3: suitability checking

• Does move change local def-use relationship?

```plaintext
+ gX = m;
  s_create(...);
  m = 0;
  gX = m;
```
Step 3: suitability checking

- Does move change local def-use relationship?

```plaintext
+ gX = m;
  s_create(...);
  m = 0;
  gX = m;
```
Step 3: suitability checking

- Does *move* change local def-use relationship?

```plaintext
+ gX = m;
s_create(...);
m = 0;
gX = m;
```
Step 3: suitability checking

• Does move change local def-use relationship?

```
+ gX = m;
s_create(...);
m = 0;
```

• Might move introduce deadlocks?
HFixmove patch example

//child thread
assert(h->band);

//parent thread
+ h->band = bdNew(h);
 pthread_create(…)
  ...
- h->band = bdNew(h);
Outline

• Motivation
• Manual patch studying
• HFix
• Evaluation
  – Methodology
  – Results
• Conclusion
Methodology

• Benchmark Suite
  – 13 bugs set up by Cfix \cite{OSDI’12}
  – All the AV bugs in ASPLOS’08 suite that are fixed through moving synchronization by developers.
## Results for OV bugs

<table>
<thead>
<tr>
<th>App.</th>
<th>HFix</th>
<th>VS. Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>+ join</td>
<td>✓</td>
</tr>
<tr>
<td>HTTrack</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>Mozilla</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>PBZIP2</td>
<td>+ join</td>
<td>✓</td>
</tr>
<tr>
<td>Transmission</td>
<td>↓ create</td>
<td>✓</td>
</tr>
<tr>
<td>X264</td>
<td>↓ join</td>
<td>✓</td>
</tr>
<tr>
<td>ZSNES</td>
<td>↓ create</td>
<td>✓</td>
</tr>
</tbody>
</table>
Results for OV bugs

<table>
<thead>
<tr>
<th>App.</th>
<th>HFix</th>
<th></th>
<th>CFix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strategy</td>
<td>#.Sync</td>
<td>Strategy</td>
</tr>
<tr>
<td>FFT</td>
<td>+ join</td>
<td>1</td>
<td>+ signal/wait</td>
</tr>
<tr>
<td>HTTrack</td>
<td>-</td>
<td>-</td>
<td>+ signal/wait</td>
</tr>
<tr>
<td>Mozilla</td>
<td>-</td>
<td>-</td>
<td>+ signal/wait</td>
</tr>
<tr>
<td>PBZIP2</td>
<td>+ join</td>
<td>1</td>
<td>+ signal/wait</td>
</tr>
<tr>
<td>Transmission</td>
<td>↓ create</td>
<td>0</td>
<td>+ signal/wait</td>
</tr>
<tr>
<td>X264</td>
<td>↓ join</td>
<td>0</td>
<td>+ signal/wait</td>
</tr>
<tr>
<td>ZSNES</td>
<td>↓ create</td>
<td>0</td>
<td>+ signal/wait</td>
</tr>
</tbody>
</table>
## Results for AV bugs

<table>
<thead>
<tr>
<th>App.</th>
<th>HFix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>✗</td>
</tr>
<tr>
<td>Mozilla1</td>
<td>lock</td>
</tr>
<tr>
<td>Mozilla2</td>
<td>lock</td>
</tr>
<tr>
<td>MySQL1</td>
<td>unlock</td>
</tr>
<tr>
<td>MySQL2</td>
<td>unlock</td>
</tr>
<tr>
<td>MySQL3</td>
<td>unlock</td>
</tr>
</tbody>
</table>
Outline

• Motivation
• Manual patch studying
• HFix
• Evaluation
• Conclusion
Conclusion

• Manual patch study:
  – Add join for OV (HFix)
  – Move existing sync. (HFix)
  – Bypass & tolerate (Future work)

• HFix
  – A useful complement.
  – Generate much simple patches.
Thank you!

Q&A