CoRD: Collaborative Data Race Detection

Baris Kasikci, Cristian Zamfir, and George Candea

School of Computer & Communication Sciences
Data Races
Data Races

- Accesses to shared memory location
Data Races

- Accesses to shared memory location
  - By *multiple threads*
Data Races

- Accesses to shared memory location
  - By multiple threads
  - At least one of the accesses is a write
Data Races

- Accesses to shared memory location
  - By multiple threads
  - At least one of the accesses is a write
  - The accesses can happen simultaneously
Data Races
Data Races

Races are numerous in modern software
Data Races

1000 Races

Races are numerous in modern software
Data races
C/C++

POSIX
Compilers can arbitrarily break racy programs
How to Ensure Software is Race-free?
How to Ensure Software is Race-free?

- Static race detectors
  - Full path analysis ✔️
  - Fast ✔️
  - Few false negatives ✔️
  - Many false positives ✗
How to Ensure Software is Race-free?

- **Static race detectors**
  - *Full path analysis* ✓
  - *Fast* ✓
  - *Few false negatives* ✓
  - *Many false positives* ✗

- **Dynamic race detectors**
  - *Per-run analysis* ✗
  - *Slow* ✗
  - *Many false negatives* ✗
  - *Few false positives* ✓
How to Ensure Software is Race-free?

- **Static race detectors**
  - Full path analysis ✔
  - Fast ✔
  - Few false negatives ✔
  - Many false positives ✗

- **Dynamic race detectors**
  - Per-run analysis ✗
  - Slow ✗
  - Many false negatives ✗
  - Few false positives ✔

Existing detectors have important limitations
How to Ensure Software is Race-free?

- Static race detectors
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔

- Dynamic race detectors
  - *Few false positives* ✔
CoRD

- Collaborative race detection
  - Full path analysis ✔
  - Fast ✔
  - Few false negatives ✔
  - Few false positives ✔
CoRD

- Collaborative race detection
  - Full path analysis ✔
  - Fast ✔
  - Few false negatives ✔
  - Few false positives ✔

Statically detect potential races
CoRD

- Collaborative race detection
  - Full path analysis ✔
  - Fast ✔
  - Few false negatives ✔
  - Few false positives ✔

Statically detect potential races
Dynamically validate detected races
CoRD

- Collaborative race detection
  - Full path analysis ✔
  - Fast ✔
  - Few false negatives ✔
  - Few false positives ✔

Effectively detected 8 real races in two real programs with 1% overhead

Statically detect potential races
Dynamically validate detected races
CoRD Architecture
CoRD Architecture

instances of program P

Hive
CoRD Architecture

instances of program P

Statically detect potential races in P
CoRD Architecture

instances of program P

Statically detect potential races in P

Hive

pods
CoRD Architecture

- Dynamically validate races
- Statically detect potential races in P
- Instances of program P
- Pods

Hive
1 Google chrome blog. http://chrome.blogspot.ch/2012_06_01_archive.html
4 http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/
computers running Chrome

300 Million

1 Google chrome blog. http://chrome.blogspot.ch/2012_06_01_archive.html
4 http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/
computers running Chrome: 300 Million

servers owned by Google: 1.8 Million

---

1. Google chrome blog. [http://chrome.blogspot.ch/2012_06_01_archive.html](http://chrome.blogspot.ch/2012_06_01_archive.html)
2. [https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY, 2012](https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY)
computers running Chrome 1

servers owned by Google 2

mobile devices bought in 2011 3

---

1 Google chrome blog. [http://chrome.blogspot.ch/2012_06_01_archive.html](http://chrome.blogspot.ch/2012_06_01_archive.html)
computers running Chrome 1

300 Million

servers owned by Google 2

1.8 Million

Windows 7 beta testers 4

8 Million

mobile devices bought in 2011 3

1.8 Billion

Static Race Detection
Static Race Detection
Static Race Detection
Static Race Detection
Static Race Detection
Static Race Detection

CFG entry

main()

f()  g()
Static Race Detection

```c
main()
g()
f()
```
Static Race Detection
Static Race Detection

\[ x = 0 \]
Static Race Detection

\[ x = 0 \]

Path 1

lock(l)
\[ x = 1 \]
unlock(l)
Static Race Detection

\[ x = 0 \]

Path 1

lock(1)
\[ x = 1 \]
unlock(1)

Path 2

lock(k)
\[ x = 2 \]
unlock(k)
Static Race Detection

\[x = 0\]

Path 1

\[\text{lock}(l)\]
\[x = 1 \quad \text{LS}_1 = \{l\}\]
\[\text{unlock}(l)\]

Path 2

\[\text{lock}(k)\]
\[x = 2 \quad \text{LS}_1 = \{k\}\]
\[\text{unlock}(k)\]
Static Race Detection

\[ x = 1 \quad \text{LS}_1 = \{1\} \quad x = 2 \quad \text{LS}_1 = \{k\} \]
Static Race Detection

\[ x = 1 \]
\[ \text{LS}_1 = \{1\} \]
\[ \bigcap \]
\[ x = 2 \]
\[ \text{LS}_1 = \{k\} \]
\[ = \{\} \]
Static Race Detection

\[ x = 1 \quad x = 2 \]

\[ \text{LS}_1 = \{1\} \quad \cap \quad \text{LS}_1 = \{k\} \quad = \quad \{\} \]

\[ \Rightarrow x = 1 \text{ and } x = 2 \text{ are RACING!} \]
Static Race Detection

\[ x = 1 \quad \cap \quad x = 2 \]

\[ \text{LS}_1 = \{1\} \quad \cap \quad \text{LS}_1 = \{k\} = {} \]

\[ \Rightarrow x = 1 \quad \text{and} \quad x = 2 \quad \text{are RACING!} \]

Top-down, flow sensitive, interprocedural, lockset-based
Dynamic Race Validation

Hive

x = 1
x = 2
Dynamic Race Validation

Hive

Pod 1

Pod 2

x = 1

x = 2
Dynamic Race Validation

Pod 1

Pod 2

Hive

\[ x = 1 \]
\[ x = 2 \]

\[ x = 2 \]
\[ x = 1 \]
Dynamic Race Validation

Pod 1 → Hive (executed)

Pod 2 ← Hive (executed)

Time:

\[
x = 1
\]
\[
x = 2
\]

\[
x = 2
\]
\[
x = 1
\]
Dynamic Race Validation

Pod 1

Pod 2

Hive

RACE

RACE CAUSES DEADLOCK

x = 1

x = 2

x = 1

x = 2

Time
Dynamic Race Validation

Time

Pod 1

Pod 2

Hive

RACE

RACE CAUSES DEADLOCK

x = 1

x = 2

x = 1

executed

deadlock

executed

RACE CAUSES CRASH

crash

executed
Dynamic Race Validation

Pod 1: 
- x = 1: executed (deadlock)
- x = 2: executed

Pod 2: 
- x = 2: executed
- x = 1: not executed

Hive:
- RACE
- RACE CAUSES DEADLOCK
- RACE CAUSES CRASH
- LIKELY FP
Dynamic Race Validation

P
Pod 1

executed

deadlock

executed

executed

not executed

RACE

RACE CAUSES
DEADLOCK

executed

executed

RACE CAUSES
CRASH

executed

P
Pod 2

RACE CAUSES
DEADLOCK

x = 1

x = 2

x = 2

x = 1

executed

not executed

LIKELY FP

NOT EXECUTED

not executed

not executed

not executed

not executed
## Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RACE CAUSES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEADLOCK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RACE CAUSES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRASH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIKELY FP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOT EXECUTED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUNTIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OVERHEAD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td>37</td>
<td>63</td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td>37</td>
<td>63</td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td>37</td>
<td>63</td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td>0.99%</td>
<td>0.91%</td>
</tr>
</tbody>
</table>
### Detection Results and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL</strong></td>
<td>88</td>
<td>122</td>
</tr>
<tr>
<td><strong>RACE</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>RACE CAUSES DEADLOCK</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>RACE CAUSES CRASH</strong></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>LIKELY FP</strong></td>
<td>37</td>
<td>63</td>
</tr>
<tr>
<td><strong>NOT EXECUTED</strong></td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td><strong>RUNTIME OVERHEAD</strong></td>
<td>0.99%</td>
<td>0.91%</td>
</tr>
</tbody>
</table>

Effective and low overhead
Comparison to Other Detectors

- Dynamic detectors have high overhead

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoRD</td>
<td>0.99%</td>
<td>0.91%</td>
</tr>
<tr>
<td>ThreadSanitizer</td>
<td>972%</td>
<td>3,001%</td>
</tr>
</tbody>
</table>
Comparison to Other Detectors

- Dynamic detectors have high overhead

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>Pbzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoRD</td>
<td>0.99%</td>
<td>0.91%</td>
</tr>
<tr>
<td>ThreadSanitizer</td>
<td>972%</td>
<td>3,001%</td>
</tr>
</tbody>
</table>

- Static detectors have false positives and don’t provide any classification
Summary

• Collaborative race detection
  • *Statically detect races*
  • *Dynamically validate them*

• Effective
  • *Detected 8 real races in 2 real programs*

• Efficient
  • *Has < 1% overhead*
Roadmap

- Synthesizing fixes
- Privacy implications
- Extension to other types of bugs