SigRace: Signature-Based Data Race Detection

Abdullah Muzahid, Dario Suarez*, Shanxiang Qi & Josep Torrellas

Computer Science Department
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

*Universidad de Zaragoza, Spain
“Debugging a multithreaded program has a lot in common with medieval torture methods”

-- Random quote found via Google search
Data Race

- Two threads access the same variable without intervening synchronization and at least one is a write

- Common bug

- Hard to detect and reproduce

```
T1
lock L
x++
unlock L
```

```
T2
x++
```
• Mainly two approaches
Dynamic Data Race Detection

- Mainly two approaches
  - Lockset: Finds violation of locking discipline
Dynamic Data Race Detection

- Mainly two approaches
  - *Lockset*: Finds violation of locking discipline
  - *Happened-Before*: Finds concurrent conflicting accesses
Happened-Before Approach
Happened-Before Approach

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0]</td>
<td>[0, 0]</td>
</tr>
</tbody>
</table>
Happened-Before Approach

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0]</td>
<td>[0, 0]</td>
</tr>
<tr>
<td>Lock L</td>
<td></td>
</tr>
<tr>
<td>[1, 0]</td>
<td></td>
</tr>
</tbody>
</table>
Happened-Before Approach

Thread 0   Thread 1

[0, 0]   [0, 0]

Lock L

[1, 0]

Unlock L

[2, 0]
Happened-Before Approach

Thread 0

[0, 0]

Lock L

[1, 0]

Unlock L

[2, 0]

Thread 1

[0, 0]
Happened-Before Approach

Thread 0

[0, 0]
Lock L
[1, 0]
Unlock L
[2, 0]

Thread 1

[0, 0]
Lock L
Happened-Before Approach

Thread 0

[0, 0]

Lock L

[1, 0]

Unlock L

[2, 0]

Thread 1

[0, 0]

[0, 1]
Happened-Before Approach

Thread 0
- [0, 0]
  *Lock L*
- [1, 0]
  *Unlock L*
- [2, 0]

Thread 1
- [0, 0]
- [0, 1]
- [2, 1]

[Diagram: A circular diagram showing the sequence of lock and unlock operations for two threads, Thread 0 and Thread 1.]
Happened-Before Approach

- Epoch: sync to sync

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0]</td>
<td>[0, 0]</td>
</tr>
<tr>
<td>Lock L</td>
<td></td>
</tr>
<tr>
<td>[1, 0]</td>
<td></td>
</tr>
<tr>
<td>Unlock L</td>
<td></td>
</tr>
<tr>
<td>[2, 0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2, 1]</td>
</tr>
</tbody>
</table>
Happened-Before Approach

- Epoch : sync to sync

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0]</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>d</td>
</tr>
<tr>
<td>[1, 0]</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>[2, 0]</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2, 1]</td>
</tr>
</tbody>
</table>

- Lock L
- Unlock L
Happened-Before Approach

- Epoch: sync to sync
Happened-Before Approach

Thread 0
- [0, 0]
  - Lock L
- [1, 0]
  - Unlock L
- [2, 0]

Thread 1
- [0, 0]
- [2, 1]

- Epoch: sync to sync
Happened-Before Approach

Thread 0
[0, 0]  a  [1, 0]  b
Lock L
[1, 0]  b  [2, 0]  c
Unlock L
[2, 0]  c  [2, 1]  e
Lock L

Thread 1
[0, 0]  d

• Epoch : sync to sync
Happened-Before Approach

- Epoch: sync to sync
- a, b happened before e
Happened-Before Approach

- Epoch: sync to sync
- a, b happened before e
Happened-Before Approach

- Epoch: sync to sync
- $a, b$ happened before $e$
- $c, d$ unordered
Happened-Before Approach

Thread 0
[0, 0]  a
[1, 0]  b
[2, 0]  c
x =

Thread 1
[0, 0]  d
[2, 1]  e

= x

Data Race

- Epoch: sync to sync
- a, b happened before e
- c, d unordered
Software Implementation

• Need to instrument every memory access
  – 10x – 50x slowdown
  – Not suitable for production runs
Hardware Implementation
Hardware Implementation
Hardware Implementation

P1

TS

C1

P2

TS

C2
Hardware Implementation

P1

\[ \text{ts1} \quad x \quad \ldots \quad C1 \]

P2

\[ \text{TS} \quad \ldots \quad C2 \]
Hardware Implementation

P1

<table>
<thead>
<tr>
<th>ts1</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

C1

TS

P2

<table>
<thead>
<tr>
<th>ts2</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

C2

TS

ts2

WR
Hardware Implementation

check

P1

P2

C1

C2

ts2

ts2

TS

TS

x

x

…

…

WR

SigRace: Signature Based Data Race Detection
Limitations of HW Approaches
Limitations of HW Approaches

• Modify cache and coherence protocol
Limitations of HW Approaches

- Modify cache and coherence protocol
- Perform checking at least on every coherence transaction
Limitations of HW Approaches

- Modify cache and coherence protocol
- Perform checking at least on **every coherence transaction**
- Lose detection ability when cache line is displaced or invalidated
Our Contributions

• SigRace: Novel HW mechanism for race detection based on signatures
  – Simple HW
    • Cache and coherence protocol are unchanged
  – Higher coverage than existing HW schemes
    • Detect races even if the line is displaced/invalidated
  – Usable on-the-fly in production runs

• SigRace finds 150% more injected races than a state-of-the-art HW proposal
Outline

• Motivation
• Main Idea
• Implementation
• Results
• Conclusions
Main Idea

Address Signature + Happened-before
Hardware Address Signatures

[Ceze et al, ISCA06]
Hardware Address Signatures
Hardware Address Signatures

- Logical AND for intersection
- Has false positives but not false negatives

\[ S_1 \cap S_2 \]
Using Signatures for Race Detection

Epoch

\[ \text{sync} \]

\[ \text{sync} \]
Using Signatures for Race Detection

Epoch

sync

sync

TS  Sig
• Block is a fixed number of dynamic instructions (not a cache block or basic block or atomic block)
Using Signatures for Race Detection

Epoch

sync

Sync

Block

TS1  Sig1

Race Detection Module

SigRace: Signature Based Data Race Detection
Using Signatures for Race Detection

Race Detection Module

Epoch

sync

Sync

Block

TS1 Ø
Using Signatures for Race Detection

- Epoch
- Block
- Sync
- TS1
- Sig2
- Race Detection Module

Abdullah Muzahid
Using Signatures for Race Detection

Epoch

sync

Block

sync

TS1  Sig2

Race Detection Module
Using Signatures for Race Detection
Using Signatures for Race Detection

Epoch

sync

Block

TS2  Sig3

Race Detection Module
Using Signatures for Race Detection

Sync

Epoch

Sync

Block

TS2

Sig3

Race Detection Module

SigRace: Signature Based Data Race Detection
Using Signatures for Race Detection

Sync

Epoch

Sync

Block

TS2  Sig3

Race Detection Module

SigRace: Signature Based Data Race Detection
Using Signatures for Race Detection

Epoch

sync

Block
TS2 Sig3

Sig \cap Sig

Race Detection Module

Abdullah Muzahid

SigRace: Signature Based Data Race Detection
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)

![Diagram of On Chip Race Detection Module (RDM)]
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)
On Chip Race Detection Module (RDM)

If T2 & TJ unordered
R2 ∪ WJ
W2 ∪ WJ
W2 ∪ RJ
Else stop
On Chip Race Detection Module (RDM)

If T2 & T J' unordered

\[ R2 \cap W J' \]
\[ W2 \cap W J' \]
\[ W2 \cap R J' \]

Else stop
On Chip Race Detection Module (RDM)

If T2 & TJ” unordered
R2 ∩ WJ”
W2 ∩ WJ”
W2 ∩ RJ”
Else stop

P1

P2

T2 R2 W2
T1 R1 W1

Q1

Q2

RDM

Chip

Abdullah Muzahid

SigRace: Signature Based Data Race Detection
On Chip Race Detection Module (RDM)

If $T_2$ & $T J''$ unordered

$R_2 \cap W J''$

$W_2 \cap W J''$

$W_2 \cap R J''$

Else stop
On Chip Race Detection Module (RDM)

If $T_2 \& T_{J''} unordered$

$R_2 \cap W_{J''}$

$W_2 \cap W_{J''}$

$W_2 \cap R_{J''}$

Else stop

Done in Background
On Chip Race Detection Module (RDM)

If $T_2$ & $T_{J''''}$ unordered
- $R_2 \cap W_{J''''}$
- $W_2 \cap W_{J''''}$
- $W_2 \cap R_{J''''}$

Else stop
Re-execution

• Needed for
  – Identify the accesses involved
  – Discard if a false positive
Support for Re-execution

- Take periodic checkpoints: ReVive [Prvulovic et al, ISCA02]
- Log inputs (interrupts, sys calls, etc)
- Save synchronization history in TS Log
  - Timestamp at sync points
Modes of Operation

- **Normal Execution**
- **Re-Execution**: Bring the program to just before the race
- **Race Analysis**: Pinpoint the racy accesses or discarding the false positive
SigRace Re-execution Mode

- Can be done in another machine
SigRace Re-execution Mode

- Can be done in another machine
- Periodic checkpoint of memory state
SigRace Re-execution Mode

- Can be done in another machine
- Periodic checkpoint of memory state
SigRace Re-execution Mode

- Can be done in another machine
- Periodic checkpoint of memory state

```
T0  T1  T2
sync sync sync
sync
sync
```

Conflict Sig Conflict Sig

Data Race
SigRace Re-execution Mode

- Can be done in another machine
- Periodic checkpoint of memory state
SigRace Re-execution Mode

- Can be done in another machine
- Periodic checkpoint of memory state

Use the TS Log
SigRace Analysis Mode
SigRace Analysis Mode

- **T0**
  - sync
  - sync

- **T1**
  - sync
  - sync

- **T2**
  - sync

Checkpoint
SigRace Analysis Mode

T0 -> sync -> T1 -> sync -> T2

checkpoint

Conflict Sig

ld Ω

Conflict Sig

SigRace: Signature Based Data Race Detection
SigRace Analysis Mode

- T0
- T1
- T2

Sync connections:
- T0 → T1
- T1 → T2
- T2 → log
- Conflicts:
  - Conflicts between T0 and T1
  - Conflicts between T1 and T2
  - Conflicts between T2 and log
SigRace Analysis Mode

Abdullah Muzahid

SigRace: Signature Based Data Race Detection
SigRace Analysis Mode

Abdullah Muzahid

SigRace: Signature Based Data Race Detection
SigRace Analysis Mode

- Pinpoints racy addresses or,
- Identifies and discards false positives
Outline

• Motivation
• Main Idea
• Implementation
• Results
• Conclusions
New Instructions

- **collect_on**
  - Enable R and W address collection in current thread
New Instructions

- **collect_on**
  - Enable R and W address collection in current thread

- **collect_off**
  - Disable R and W address collection in current thread
New Instructions

- `sync_reached`
New Instructions

- **sync_reached**
  - Dump TS, R and W
New Instructions

- `sync_reached`
  - Dump TS, R and W
  - Clear signatures

```
+---+  P  +---+  TS  Ø  Ø  +---+  Network  +---+  RDM  +---+
|   |     |   |     |     |     |     |     |
|   |     |   |     |     |     |     |     |
|   |     |   |     |     |     |     |     |
```

Abdullah Muzahid  SigRace: Signature Based Data Race Detection
New Instructions

• `sync_reached`
  – Dump TS, R and W
  – Clear signatures
  – Update TS
Modifications in Sync Libraries
Modifications in Sync Libraries

- Synchronization object
  | variable | timestamp |
Modifications in Sync Libraries

- Synchronization object
  - variable
  - timestamp
- Unlock macro

  UNLOCK (‘{

      unlock($1.lock)

  }’)}
Modifications in Sync Libraries

- Synchronization object
  - variable
  - timestamp

- Unlock macro

  UNLOCK (‘{
  sync_reached
  unlock($1.lock)
  }’)

Network

P

RDM
Modifications in Sync Libraries

- Synchronization object
  
  | variable | timestamp |
  
- Unlock macro

  UNLOCK (‘{
    sync_reached
    unlock($1.lock)
  }’)
Modifications in Sync Libraries

- Synchronization object
  - variable
  - timestamp

- Unlock macro

```c
UNLOCK ('{
    sync_reached
    unlock($1.lock)
}
')
```
Modifications in Sync Libraries

- Synchronization object
  - variable timestamp

- Unlock macro

  UNLOCK (‘{
    sync_reached
    $1.timestamp = TS
    unlock($1.lock)
  }’)
Modifications in Sync Libraries

- Synchronization object
  
  | variable | timestamp |
  
- Unlock macro

```c
UNLOCK (‘{
    sync_reached
    $1.timestamp = TS
    unlock($1.lock)
}
)```

Abdullah Muzahid
Modifications in Sync Libraries

• Synchronization object

| variable | timestamp |

• Unlock macro

UNLOCK (‘{
    sync_reached
    $1.timestamp = TS
    unlock($1.lock)
    AppendtoTSLog(TS)
}’)

TS Log

TS
Modifications in Sync Libraries

- Synchronization object
  - variable
  - timestamp

- Lock macro

  LOCK (‘{
      ...
      lock($1.lock)
      ...
  }’)
Modifications in Sync Libraries

- Synchronization object

<table>
<thead>
<tr>
<th>variable</th>
<th>timestamp</th>
</tr>
</thead>
</table>

- Lock macro

  LOCK ("{
    ...
    lock($1.lock)
    TS = GenerateTS (TS, $1.timestamp)
    ...
  }")
Modifications in Sync Libraries

- Synchronization object
  - variable timestamp

- Lock macro

```
LOC
...
lock($1.lock)
TS = GenerateTS (TS, $1.timestamp)
...
}’
```

Transparent to Application Code
Other Topics in Paper

- Easy to virtualize
- Queue Overflow
- Detailed HW structures
Outline

• Motivation
• Main Idea
• Implementation
• Results
• Conclusions
Experimental Setup

- PIN – Binary Instrumention Tool
- Default parameters
  - # of proc: 8
  - Signature size: 2 Kbits
  - Block size: 2,000 ins
  - Queue size: 16 entries
  - Checkpoint interval: 1 Million ins
- Benchmarks: SPLASH2, PARSEC
Race Detection Ability

- Three configurations
  - SigRace Default
  - SigRace Ideal: Stores every signature between 2 checkpoints
  - ReEnact [Prvulovic et al, ISCA03]: Cache based approach with timestamp per word
## Race Detection Ability

<table>
<thead>
<tr>
<th>App</th>
<th>Ideal SigRace</th>
<th>Default SigRce</th>
<th>ReEnact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Barnes</td>
<td>11</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Volrend</td>
<td>27</td>
<td>27</td>
<td>18</td>
</tr>
<tr>
<td>Ocean</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Radiosity</td>
<td>15</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>Raytrace</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Water-sp</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Streamclus</td>
<td>13</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>95</strong></td>
<td><strong>90</strong></td>
<td><strong>70</strong></td>
</tr>
</tbody>
</table>
# Race Detection Ability

<table>
<thead>
<tr>
<th>App</th>
<th>Ideal SigRace</th>
<th>Default SigRce</th>
<th>ReEnact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Barnes</td>
<td>11</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Volrend</td>
<td>27</td>
<td>27</td>
<td>18</td>
</tr>
<tr>
<td>Ocean</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Radiosity</td>
<td>15</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>Raytrace</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Water-sp</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>13</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

**Total** | **95** | **90** | **70**

- More coverage than ReEnact
## Race Detection Ability

- More coverage than ReEnact
- Coverage comparable to ideal configuration

<table>
<thead>
<tr>
<th>App</th>
<th>Ideal SigRace</th>
<th>Default SigRce</th>
<th>ReEnact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Barnes</td>
<td>11</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Volrend</td>
<td>27</td>
<td>27</td>
<td>18</td>
</tr>
<tr>
<td>Ocean</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Radiosity</td>
<td>15</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>Raytrace</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Water-sp</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>13</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>95</strong></td>
<td><strong>90</strong></td>
<td><strong>70</strong></td>
</tr>
</tbody>
</table>
Injected Races

- Removed one dynamic sync per run
- Each application runs 25 times with diff sync elimination
## Injected Races

<table>
<thead>
<tr>
<th>Application</th>
<th>Static Races Found</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SigRace</td>
</tr>
<tr>
<td>FFT</td>
<td>600</td>
</tr>
<tr>
<td>Cholesky</td>
<td>2</td>
</tr>
<tr>
<td>LU</td>
<td>28</td>
</tr>
<tr>
<td>Barnes</td>
<td>3</td>
</tr>
<tr>
<td>Volrend</td>
<td>345</td>
</tr>
<tr>
<td>Ocean</td>
<td>8</td>
</tr>
<tr>
<td>Radiosity</td>
<td>29</td>
</tr>
<tr>
<td>Raytrace</td>
<td>66</td>
</tr>
<tr>
<td>Water-ns</td>
<td>2</td>
</tr>
<tr>
<td>Water-spatial</td>
<td>6</td>
</tr>
<tr>
<td>Dedup</td>
<td>0</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>7</td>
</tr>
<tr>
<td>Blascholes</td>
<td>0</td>
</tr>
<tr>
<td>Fluidanimate</td>
<td>95</td>
</tr>
<tr>
<td>Swaptions</td>
<td>–</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1191</strong></td>
</tr>
</tbody>
</table>

**SigRace**: Signature Based Data Race Detection

Abdullah Muzahid
### Injected Races

#### More overall coverage than ReEnact

<table>
<thead>
<tr>
<th>Application</th>
<th>Static Races Found</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SigRace</td>
</tr>
<tr>
<td>FFT</td>
<td>600</td>
</tr>
<tr>
<td>Cholesky</td>
<td>2</td>
</tr>
<tr>
<td>LU</td>
<td>28</td>
</tr>
<tr>
<td>Barnes</td>
<td>3</td>
</tr>
<tr>
<td>Volrend</td>
<td>345</td>
</tr>
<tr>
<td>Ocean</td>
<td>8</td>
</tr>
<tr>
<td>Radiosity</td>
<td>29</td>
</tr>
<tr>
<td>Raytrace</td>
<td>66</td>
</tr>
<tr>
<td>Water-ns</td>
<td>2</td>
</tr>
<tr>
<td>Water-spatial</td>
<td>6</td>
</tr>
<tr>
<td>Dedup</td>
<td>0</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>7</td>
</tr>
<tr>
<td>Blackscholes</td>
<td>0</td>
</tr>
<tr>
<td>Fluidanimate</td>
<td>95</td>
</tr>
<tr>
<td>Swaptions</td>
<td>–</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1191</strong></td>
</tr>
</tbody>
</table>
## Injected Races

- More overall coverage than ReEnact
  - 150% more coverage

<table>
<thead>
<tr>
<th>Application</th>
<th>SigRace</th>
<th>ReEnact</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>600</td>
<td>150</td>
</tr>
<tr>
<td>Cholesky</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>LU</td>
<td>28</td>
<td>75</td>
</tr>
<tr>
<td>Barnes</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Volrend</td>
<td>345</td>
<td>74</td>
</tr>
<tr>
<td>Ocean</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Radiosity</td>
<td>29</td>
<td>11</td>
</tr>
<tr>
<td>Raytrace</td>
<td>66</td>
<td>53</td>
</tr>
<tr>
<td>Water-ns</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Water-spatial</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Dedup</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Blackscholes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fluidanimate</td>
<td>95</td>
<td>90</td>
</tr>
<tr>
<td>Swaptions</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1191</strong></td>
<td><strong>476</strong></td>
</tr>
</tbody>
</table>
Injected Races

- More overall coverage than ReEnact
  - 150% more coverage

<table>
<thead>
<tr>
<th>Application</th>
<th>Static Races Found</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SigRace</td>
<td>ReEnact</td>
</tr>
<tr>
<td>FFT</td>
<td>600</td>
<td>150</td>
</tr>
<tr>
<td>Cholesky</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>LU</td>
<td>28</td>
<td>75</td>
</tr>
<tr>
<td>Barnes</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Volrend</td>
<td>345</td>
<td>74</td>
</tr>
<tr>
<td>Ocean</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Radiosity</td>
<td>29</td>
<td>11</td>
</tr>
<tr>
<td>Raytrace</td>
<td>66</td>
<td>53</td>
</tr>
<tr>
<td>Water-ns</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Water-spatial</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Dedup</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Blackscholes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fluidanimate</td>
<td>95</td>
<td>90</td>
</tr>
<tr>
<td>Swaptions</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1191</strong></td>
<td><strong>476</strong></td>
</tr>
</tbody>
</table>
Conclusions

• Proposed SigRace:
  – Simple HW
    • Cache and coherence protocol are unchanged
  – Higher coverage than existing HW schemes
    • Detect races even if the line is displaced/invalidated
  – Usable on-the-fly in production runs

• SigRace finds 150% more injected races than word-based ReEnact
SigRace: Signature-Based Data Race Detection

Abdullah Muzahid, Dario Suarez*, Shanxiang Qi & Josep Torrellas

Computer Science Department
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

*Universidad de Zaragoza, Spain
Back Up Slides
Execution Overhead

- No overhead in generating signatures (HW)
- Additional instructions are negligible
- Main overheads
  - Checkpointing (ReVive – 6.3%)
  - Network traffic (63 bytes per 1000 ins - compressed)
  - Re-execution (depends on false positives & race position)
    - Can be done offline
Network Traffic Overhead

Network Overhead (bytes/thousand ins)

- Splash2 kernels: ~60
- Splash2 apps: ~50
- Parsec kernels: ~40
- Parsec apps: 63
- Average: 63

~ 1 cache line
Re-execution Overhead

- Instructions re-executed until the first true data race is analyzed are shown as overhead.
- In this process, it may also encounter many false positive races.
- Instructions re-executed to analyze only the true race are shown as true overhead.
- Instructions re-executed to filter out the false positives are shown as false overhead.
Re-execution Overhead

Modest overhead

- **Splash2 kernels**
- **Splash2 apps**
- **Parsec kernels**
- **Parsec apps**
- **Average**

Instruction Overhead (%)
False Positives

- Parallel bloom filters with $H_3$ hash function

![Bar chart showing False Positives](chart)

**Signature Configuration**

Low False Positive

1.57%
Virtualization

(a) Vector Clock

(b) Race Detection Module (RDM)
Virtualization

- RDM uses as many queues as the number of threads
- Timestamp is accessed by thread id
- Thread id remains same even after migration
- Timestamps, flags, conflict signature are saved and restored at context switch
- RDM intersects incoming signatures against all other threads’ (even inactive ones) signatures
- Threads can be re-executed without any scheduling constraints
Scalability

• For small # of proc., scalability is not a problem
• The operation of RDM can be pipelined
  – Simple repetitive operation
• Network traffic (compressed message) around 63Bytes/thousand ins
• Checkpoint is an issue.