Pacman: Tolerating Asymmetric Data Races with Unintrusive Hardware

Shanxiang Qi, Norimasa Otsuki*, Lois Orosa Nogueira**, Abdullah Muzahid and Josep Torrellas

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

*Renesas Electronics Corporation, Japan
**Universidade de Santiago de Compostela, Spain
Data Race

* Two threads access the same variable without intervening sync
* At least one is write
* Data races are very common; some are more harmful than others
* Must focus on removing the harmful ones --> those in bug reports

---

Shanxiang Qi

Pacman: Tolerating Asymmetric Races
Asymmetric Data Race

* Data race where at least one of the threads is inside a critical section

```c
Lock
if(pointer != NULL){
    pointer->x = X1;
    pointer->y = X2;
}
Unlock
```

Unsafe Thread

```c
pointer = NULL;
```

Safe Thread

Friday, March 9, 12
Importance

- Asymmetric data races are likely *harmful*
- The data being corrupted is data protected in a critical section
- Asymmetric data races are *common*
- 20% of reported harmful data races are asymmetric
- A Microsoft report confirmed this problem
  - Third party drivers
  - Legacy single-threaded libraries
Contributions

- Quantitative evidence to show asymmetric data races are *harmful* and *common*
- **Pacman**: a scheme to *prevent* asymmetric races in production runs
  - Unintrusive hardware support
  - Negligible execution overhead
  - No changes to the software
- Discovered *two* unreported asymmetric race bugs
Outline

- Motivation
- Study of Asymmetric Races
- Pacman Design
- Evaluation
- Conclusions
### Study of Harmful Races

- Collected 50 races from open source bug libraries and reports
  - servers
  - desktop apps
  - OS & libraries

<table>
<thead>
<tr>
<th>Apps.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Web Server</td>
</tr>
<tr>
<td>MySQL</td>
<td>Database sever</td>
</tr>
<tr>
<td>Mozilla</td>
<td>Browser</td>
</tr>
<tr>
<td>Pbzip2</td>
<td>Parallel bzip2</td>
</tr>
<tr>
<td>Redhat</td>
<td>glibc library</td>
</tr>
<tr>
<td>JAVA</td>
<td>SDK</td>
</tr>
<tr>
<td>Windows</td>
<td>Kernel</td>
</tr>
<tr>
<td>libcprops</td>
<td>C library</td>
</tr>
</tbody>
</table>
Asymmetric Data Races are Common

+ 10 out of 50 harmful data races are asymmetric races

<table>
<thead>
<tr>
<th>Application</th>
<th>Source</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Bug# 1507</td>
<td>Exception</td>
</tr>
<tr>
<td>MySQL</td>
<td>Bug# 48930</td>
<td>System hangs</td>
</tr>
<tr>
<td>Mozilla-JS</td>
<td>Bug# 622691</td>
<td>Incorrect result</td>
</tr>
<tr>
<td>Mozilla-XPConnect</td>
<td>Bug# 557586</td>
<td>Segmentation fault</td>
</tr>
<tr>
<td>Mozilla-Video/Audio</td>
<td>Bug# 639721</td>
<td>Incorrect result</td>
</tr>
<tr>
<td>Pbzip2</td>
<td>Report</td>
<td>Segmentation fault</td>
</tr>
<tr>
<td>Windows</td>
<td>Report</td>
<td>Incorrect result</td>
</tr>
<tr>
<td>Windows</td>
<td>Report</td>
<td>Incorrect result</td>
</tr>
<tr>
<td>Windows</td>
<td>Report</td>
<td>Incorrect result</td>
</tr>
<tr>
<td>Libcprops</td>
<td>Report</td>
<td>Incorrect result</td>
</tr>
</tbody>
</table>
Outline

• Motivation
• Study of Asymmetric Races
• Pacman Design
• Evaluation
• Conclusions
Design Philosophy

* Preventing versus detecting bugs
  * All users benefit, not just programmers
  * No need to wait for lengthy bug-fixing cycles
  * No training phase
  * Hard to define “correct” run

* Goal: Prevent asymmetric data races in production runs with no training.
Main Idea

* Leverage **hardware cache coherence** to temporarily **protect** the variables being accessed in a **critical section**
Allowed Interleaving Accesses

T1

Acquire(L)

rd x

rd x

Release(L)

T2

wr x

wr x

Friday, March 9, 12
Allowed Interleaving Accesses

T1

Acquire(L)

wr x

Release(L)

T2

rd x

rd x

wr x
Pacman: Tolerating Asymmetric Races

University of Illinois

http://iacoma.cs.uiuc.edu

Shanxiang Qi

Overall System

- Pacman module: watches the traffic
- Associated with the bus controller or directory controller
- Summarizes addresses in hardware signatures

Network

Pacman Module

Memory
Hardware Signature

- Bloom filter that accumulates addresses
- Summarizes the footprint of a piece of code (critical section)
- Inexpensive operations on groups of addresses
Example

T1

Acquire(L)

wr x

T2

rd x/ wrx

Release(L)

* Thread enters a critical section to be protected
* HW informs the Pacman module
* Pacman module allocates a signature + inserts L

Pacman Module

<table>
<thead>
<tr>
<th>PID</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Sig(L)</td>
</tr>
</tbody>
</table>
Example

- For every cache miss/coherence transaction from the safe thread:
  - HW adds address to signature

<table>
<thead>
<tr>
<th>PID</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Sig(L, x)</td>
</tr>
</tbody>
</table>

T1

Acquire(L)

wr x

T2

Release(L)

rd x/ wrx

Pacman Module

Shanxiang Qi

Pacman: Tolerating Asymmetric Races
Example

Every cache miss/coherence transaction from unsafe thread
- HW checks the signature
- If match, bounce it (the processor will retry)
Example

T1

Acquire(L)

wr x

T2

Release(L)

rd x/ wrx

* Release the lock
  * HW deallocates the entry
Issues

- Cache effects
  - Pacman Module does not see the accesses intercepted by the caches
- Deadlocks
  - Two or more processors Nack each other
Cache State Before Entering the Critical Section

- Data accessed in the critical section may already be cached in the safe processor in a “silent” state.

T1

- Acquire(L)
- wr x
- Release(L)
- x: D in cache

T2

- rd x

Write-back, insert in signature and Nack requester

T1

- Acquire(L)
- rd x
- rd x
- wr x

T2

- rd x
- Release(L)
- Invalidate, insert in signature and Nack requester

Shanxiang Qi

Pacman: Tolerating Asymmetric Races
Cache Displacements During the Critical Section (CS)

- Data displaced from the cache while executing a monitored CS
  - May have been silently accessed
  - Must be conservatively placed in the signature
  - If dirty line --> line naturally written back
  - If clean line --> issue a notification signal
  - Inside a monitored CS, put the cache in *Notification mode*

```
Acquire(L) ...
```

```
Release(L)
```

```
Cache
```

```
Network
```

Shanxiang Qi  Pacman: Tolerating Asymmetric Races
Deadlocks

* Occurs in race bugs where all the threads synchronize

\[
\begin{align*}
T0 & \quad \text{Acquire } L0 \quad g0= \\
T1 & \quad \text{Acquire } L1 \quad g1= \\
\quad & \quad g1= \quad g0= \\
\quad & \quad \text{Nacked}
\end{align*}
\]

* Cross-thread stall cycle: no thread makes progress
Break the Deadlock

- Add one extra field in each entry of the Pacman Module

<table>
<thead>
<tr>
<th>PID</th>
<th>Stall_index</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Detect a cycle in hardware immediately

```
  i   j
  j   i
```

- Break the cycle right away
  - Allow one of the threads to perform one access without Nack
  - May break the atomicity of one of the critical sections
Advantages of Pacman

- Since goal is to prevent races:
  - Ok to have false positives (as long as little slowdown)
  - Simpler hardware is enough (no epoch IDs, etc)

- No software changes:
  - Synch primitives changed but hidden in library calls
  - Usable as primitive for security and debugging
Limitations

- Requires identifying critical section entry and exit points
- Not designed for certain unusual types of critical sections
  - Very large critical sections
  - Spinning on flag inside a critical section
Outline

- Motivation
- Study of Asymmetric Races
- Pacman Design
- Evaluation
- Conclusions
Evaluation Setup

- Use Pin instrumentation + SESC simulator
- Simulate CMP with 4 or 8 processors, snoopy-based MESI
- Evaluate on SPLASH-2, PARSEC, Sphinx3 and Apache
- Find two real bugs
Characterizing Critical Sections

- Critical sections are small
- Critical sections account for only 0.24% of total dynamic instructions
Execution Time Overhead of Pacman

- Sources of execution overhead in Pacman
  - Processes are Nacked and have to retry
  - Extra traffic: notifications and retries

- Pacman performance overhead is negligible
  - Very few false positives

Shanxiang Qi

Pacman: Tolerating Asymmetric Races

Friday, March 9, 12
Unreported Bug from FMM

T1

pb = b->parent

Lock

pb->subtree_cost +=
   b->subtree_cost;

pb->interaction_sync +=1;

Unlock

T2

pb->subtree_cost +=
   b->subtree_cost;

b->interaction_sync =0;
Also in the Paper

- Implementation of Pacman
- Virtualization
- Extensions for multithreaded processors
- Pacman Module for directory protocol
Software Schemes to Tolerate Asymmetric Data Race

- ToleRace [PPOPP’09], ISOLATOR [ASPLOS’09]
- When safe thread enters critical section
  - Software makes copy of the data and redirects safe thread to access the copy
  - May also change the page protections
- Limitations
  - Significant slowdown
  - Requires significant software changes
  - May induce inconsistent execution or new deadlocks
Conclusion

- Quantitative study of asymmetric data races
- Pacman: hardware scheme to prevent asymmetric data races in production runs
  - Unintrusive hardware support
  - Negligible execution overhead
  - No changes to the software
- Discovered two unreported asymmetric race bugs
Thank You!
Pacman:
Tolerating Asymmetric Data Races with Unintrusive Hardware

Shanxiang Qi, Norimasa Otsuki*, Lois Orosa Nogueira**, Abdullah Muzahid and Josep Torrellas

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

*Renesas Electronics Corporation, Japan
**Universidade de Santiago de Compostela, Spain
Backup Slide
# Evaluation

<table>
<thead>
<tr>
<th>Category</th>
<th>Application</th>
<th>Number of Nacks (L1 only, 4 threads)</th>
<th>Number of Nacks (L1 only, 8 threads)</th>
<th>Increase in traffic with L1 only (%)</th>
<th>Increase in traffic with L1+L2 (%)</th>
<th>Sync hits per dyn inst (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPLASH-2 Kernels</td>
<td>cholesky</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>fft</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>lu/contiguous</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>lu/non_cont.</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>radix</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>SPLASH-2 Apps</td>
<td>barnes</td>
<td>0 2</td>
<td>4 2</td>
<td>4.0</td>
<td>3.0</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>fnm</td>
<td>1 1</td>
<td>1 1</td>
<td>1.0</td>
<td>1.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>ocean/contiguous</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>ocean/non_cont.</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>0 13</td>
<td>15 28</td>
<td>32.0</td>
<td>14.0</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>raytrace</td>
<td>0 0</td>
<td>4 0</td>
<td>6.0</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>water-nquared</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>water-spatial</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>PARSEC Kernels</td>
<td>canneal</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>dedup</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>streamcluster</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>PARSEC Apps</td>
<td>blackscholes</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>bodytrack</td>
<td>1 1</td>
<td>2 1</td>
<td>2.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>facesim</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>ferret</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>raytrace</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>vips</td>
<td>0 0</td>
<td>2 0</td>
<td>3.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>x264</td>
<td>0 0</td>
<td>0 0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>Other Apps</td>
<td>Apache</td>
<td>- -</td>
<td>- 3</td>
<td>8.0</td>
<td>0.3</td>
<td>0.02</td>
</tr>
</tbody>
</table>
|                  | Sphinx3       | 0 4                                  | 6 10                                 | 14.0                                | 0.8                              | 1.1                       | 0.02
## Detecting vs. Tolerating

<table>
<thead>
<tr>
<th></th>
<th>Detecting data race</th>
<th>Tolerating data race</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program users</strong></td>
<td>🙁</td>
<td>😊</td>
</tr>
<tr>
<td><strong>HW Support</strong></td>
<td>🙁</td>
<td>😊</td>
</tr>
<tr>
<td><strong>Correctness (program)</strong></td>
<td>🙁</td>
<td>😊</td>
</tr>
<tr>
<td><strong>False positive</strong></td>
<td>🙁</td>
<td>😊</td>
</tr>
</tbody>
</table>
Basic Protocol

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful lock acquire</td>
<td>if (No entry for thread){ Allocate entry; NestedLevel=1;</td>
</tr>
<tr>
<td></td>
<td>else NestedLevel++; Signature += Hash(Lock_address);</td>
</tr>
<tr>
<td>Load/store by owner thread</td>
<td>Signature += Hash(address);</td>
</tr>
<tr>
<td>Load/store by other thread</td>
<td>if (address ∈ Signature) Nack address;</td>
</tr>
<tr>
<td>Lock release</td>
<td>NestingLevel--; if (NestingLevel == 0) Deallocate entry;</td>
</tr>
</tbody>
</table>

SigTable

Pid | Signature | NestingLevel
--- |-----------|---------------
|   |           |               
|   |           |               
|   |           |               
|   |           |               

Pacman: Tolerating Asymmetric Races
Breaking the Deadlock

\[ \begin{array}{cccc}
  \text{Index} & \text{PID} & \text{Signature} & \text{Stall\_index} & \text{Lock\_acquire?} \\
  0 & T0 & \text{Sig}(L_0, g_0) & 1 & 1 \\
  1 & T1 & \text{Sig}(L_1) & 0 & 0 \\
\end{array} \]