Hardware Mechanisms for Distributed Dynamic Software Analysis

Joseph L. Greathouse

Advisor: Prof. Todd Austin

May 10, 2012
Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >\(\frac{1}{3}\) from viruses, network intrusion, etc.
Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >1/3 from viruses, network intrusion, etc.
Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >\frac{1}{3} from viruses, network intrusion, etc.

Adobe Warns of Critical Zero Day Vulnerability

Posted by Soulskill on Tuesday December 06, @08:18PM
from the might-want-to-just-trademark-that-term dept.
Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >⅓ from viruses, network intrusion, etc.
Software Errors Abound

- NIST: Software errors cost U.S. \(~$60\) billion/year
- FBI: Security Issues cost U.S. \($67\) billion/year
  - >\(\frac{1}{3}\) from viruses, network intrusion, etc.

Adobe Warns of Critical Zero Day Vulnerability

Global Spam Drops by a Third After Rustock Botnet Gets Crushed, Symantec Says

Stuxnet attackers used 4 Windows zero-day exploits
Example of a Modern Bug

Nov. 2010 OpenSSL Security Flaw
Example of a Modern Bug

Nov. 2010 OpenSSL Security Flaw
Example of a Modern Bug

```c
if(ptr == NULL) {
    len=thread_local->mylen;
    ptr=malloc(len);
    memcpy(ptr, data, len);
}
```
Example of a Modern Bug

Thread 1
mylen=small

Thread 2
mylen=large

\[ \text{ptr} \quad \emptyset \]
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);

memcpy(ptr, data2, len2)

ptr
∅
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)

len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)

len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)

ptr ∅
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)

len2=thread_local->mylen;
ptr=malloc(len2);

memcpy(ptr, data2, len2)
Example of a Modern Bug

Thread 1
mylen=small
if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large
if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)

LEAKED

TIME
ptr
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

len2=thread_local->mylen;
ptr=malloc(len2);

if(ptr==NULL)

memcpy(ptr, data2, len2)

LEAKED

ptr
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)

len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)

LEAKED
Hardware Plays a Role in this Problem
Hardware Plays a Role in this Problem
In spite of proposed hardware solutions
In spite of proposed hardware solutions
In spite of proposed hardware solutions
Dynamic Software Analyses

- Analyze the program as it runs
  - Find errors on any executed path
Dynamic Software Analyses

- Analyze the program as it runs
  - Find errors on any executed path

- Data Race Detection
  (e.g. Inspector XE)

- Taint Analysis

- Memory Checking
  (e.g. MemCheck)

- Dynamic Bounds Checking
## Dynamic Software Analyses

- Analyze the program as it runs
  + Find errors on any executed path
    - LARGE overheads, only test one path at a time

- Data Race Detection
  (e.g. Inspector XE)

- Taint Analysis

- Memory Checking
  (e.g. MemCheck)

- Dynamic Bounds Checking
Dynamic Software Analyses

- Analyze the program as it runs
  - Find errors on any executed path
    - LARGE overheads, only test one path at a time

- Data Race Detection (e.g. Inspector XE)
  - 2-300x

- Memory Checking (e.g. MemCheck)
  - 5-50x

- Taint Analysis
  - 2-200x

- Dynamic Bounds Checking
  - 2-80x
Goals of this Thesis

- Allow high quality dynamic software analyses
  - Find **difficult bugs** that weaker analyses miss

- **Distribute the tests** to large populations
  - Must be low overhead or users will get angry

- **Sampling + Hardware** to accomplished this
  - Each user only tests a small part of the program
  - Each test should be helped by hardware
Meeting These Goals - Thesis Overview
Meeting These Goals - Thesis Overview

Allow high quality dynamic software analyses
Meeting These Goals - Thesis Overview

- Dataflow Analysis
- Allow high quality dynamic software analyses
- Data Race Detection
Meeting These Goals - Thesis Overview

Allow high quality dynamic software analyses
Meeting These Goals - Thesis Overview

<table>
<thead>
<tr>
<th>Dataflow Analysis</th>
<th>Software Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Race Detection</td>
<td>Hardware Support</td>
</tr>
</tbody>
</table>
# Meeting These Goals - Thesis Overview

<table>
<thead>
<tr>
<th>Data Race Detection</th>
<th>Dataflow Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Support</td>
<td>Hardware Support</td>
</tr>
</tbody>
</table>
Meeting These Goals - Thesis Overview

Dataflow Analysis

Data Race Detection

Software Support

Hardware Support

Distribute the tests + Sample the analyses
Meeting These Goals - Thesis Overview

Dataflow Analysis

Data Race Detection

Software Support

Hardware Support

Dataflow Analysis Sampling (CGO’11)

Distribute the tests + Sample the analyses
Meeting These Goals - Thesis Overview

Dataflow Analysis

Software Support

Dataflow Analysis Sampling (CGO’11)

Hardware Support

Dataflow Analysis Sampling (MICRO’08)

Distribute the tests + Sample the analyses
Meeting These Goals - Thesis Overview

<table>
<thead>
<tr>
<th>Dataflow Analysis</th>
<th>Software Support</th>
<th>Hardware Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Race Detection</td>
<td>Dataflow Analysis Sampling (CGO’11)</td>
<td>Dataflow Analysis Sampling (MICRO’08)</td>
</tr>
</tbody>
</table>
Meeting These Goals - Thesis Overview

<table>
<thead>
<tr>
<th>Dataflow Analysis</th>
<th>Software Support</th>
<th>Hardware Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataflow Analysis Sampling (CGO'11)</td>
<td>Tests must be low overhead</td>
<td>Dataflow Analysis Sampling (MICRO’08)</td>
</tr>
<tr>
<td>Data Race Detection</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Meeting These Goals - Thesis Overview

<table>
<thead>
<tr>
<th>Software Support</th>
<th>Hardware Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataflow Analysis Sampling (CGO’11)</td>
<td>Dataflow Analysis Sampling (MICRO’08)</td>
</tr>
<tr>
<td>Data Race Detection</td>
<td>Hardware-Assisted Demand-Driven Race Detection (ISCA’11)</td>
</tr>
</tbody>
</table>
Meeting These Goals - Thesis Overview

Software Support

Dataflow Analysis Sampling (CGO’11)

Dataflow Analysis Sampling (MICRO’08)

Hardware Support

Unlimited Watchpoint System (ASPLOS’12)

Hardware-Assisted Demand-Driven Race Detection (ISCA’11)
Meeting These Goals - Thesis Overview

Dataflow Analysis

- Dataflow Analysis Sampling (CGO’11)
- Unlimited Watchpoint System (ASPLOS’12)
- Hardware-Assisted Demand-Driven Race Detection (ISCA’11)

Hardware Support

- Dataflow Analysis Sampling (MICRO’08)

Software Support
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints
Distributed Dynamic Dataflow Analysis

- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test
Distributed Dynamic Dataflow Analysis

- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test
Distributed Dynamic Dataflow Analysis

- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test
Distributed Dynamic Dataflow Analysis

- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test
Distributed Dynamic Dataflow Analysis

- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test
The Problem: OVERHEADS

- Analyze the program as it runs
  + System state, find errors on any executed path
    - LARGE runtime overheads, only test one path

- Data Race Detection (e.g. Thread Analyzer)
  - 2-300x

- Memory Checking (e.g. MemCheck)
  - 5-50x

- Taint Analysis (e.g. TaintCheck)
  - 2-200x

- Dynamic Bounds Checking
  - 2-80x
Solution: Sampling

- Lower overheads by skipping some analyses
Sampling Allows Distribution

- Lower overheads mean more users

![Graph showing the relationship between Error Detection Rate and Overhead for Few Users and Many Users. The graph illustrates that as overhead increases, error detection rate decreases for Few Users, while it increases for Many Users.]

- No Analysis
- Complete Analysis
Sampling Allows Distribution

- Lower overheads mean more users

- Error Detection Rate vs. Overhead

- No Analysis vs. Complete Analysis

- Many Users vs. Easy Users

- Developers

- Few Users

- Lower overheads mean more users
Sampling Allows Distribution

- Lower overheads mean more users

![Graph showing the relationship between overhead and error detection rate, with beta testers and developers depicted.]

No Analysis

Complete Analysis
Sampling Allows Distribution

- Lower overhead means more users

<table>
<thead>
<tr>
<th>Error Detection Rate</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>End Users</td>
</tr>
<tr>
<td>75</td>
<td>Beta Testers</td>
</tr>
<tr>
<td>50</td>
<td>Developers</td>
</tr>
<tr>
<td>25</td>
<td>Complete Analysis</td>
</tr>
<tr>
<td>0</td>
<td>No Analysis</td>
</tr>
</tbody>
</table>

Many Users

Few Users
Sampling Allows Distribution

- Lower overhead mean more users

Many users testing at little overhead see more errors than one user at high overhead.
Example Dynamic Dataflow Analysis
Example Dynamic Dataflow Analysis

\[ x = \text{read\_input()} \]
Example Dynamic Dataflow Analysis

```
x = read_input()
```

Diagram:
- Data
- Meta-data
- Input
- Associate

$x = \text{read\_input()}$
Example Dynamic Dataflow Analysis

\[ y = x \times 1024 \]

Example Dynamic Dataflow Analysis

\[ x = \text{read}\_\text{input}() \]

\[ y = x \times 1024 \]

Propagate

Data

Meta-data
Example Dynamic Dataflow Analysis

```
# Input
x = read_input()

# Data
y = x * 1024
a += y
z = y * 75
```
Example Dynamic Dataflow Analysis

```
Data
Meta-data

Input

x = read_input()

validate(x)

Clear

y = x * 1024

a += y

z = y * 75
```
Example Dynamic Dataflow Analysis

\begin{align*}
x &= \text{read\_input}() \\
y &= x \times 1024 \\
a &= a + y \\
z &= y \times 75 \\
w &= x + 42 \\
\end{align*}
Example Dynamic Dataflow Analysis

\[
\begin{align*}
  a & += y \\
  z & = y \times 75 \\
  y & = x \times 1024 \\
  w & = x + 42 \\
  \text{Check } w
\end{align*}
\]

\( x = \text{read\_input()} \)

\( \text{validate}(x) \)
Example Dynamic Dataflow Analysis

```
a += y
z = y * 75
y = x * 1024
w = x + 42
```

Check w
Check a
Check z
Check a
Check z

Input

x = read_input()

Data

validate(x)

Meta-data

w = x + 42

y = x * 1024
Sampling Dataflows

- Sampling must be aware of meta-data

- Remove meta-data from skipped dataflows
Sampling Dataflows

- Sampling must be aware of meta-data

- Remove meta-data from skipped dataflows
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

- Application
- Analysis Tool

OH Threshold

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Clear meta-data

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application
Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Dataflow Sampling

Sampling Tool

Application

Analysis Tool

Meta-Data Detection
Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data
Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data
  - Solution: Virtual Memory Watchpoints
Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

\[ \text{Solution: Virtual Memory Watchpoints} \]
Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

Solution: Virtual Memory Watchpoints
Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

Solution: Virtual Memory Watchpoints
Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data
  - Solution: Virtual Memory Watchpoints

\[ V \rightarrow P \]
Prototype Setup

- Xen+QEMU Taint analysis sampling system
  - Network packets untrusted

- Performance Tests – Network Throughput
  - Example: *ssh_receive*

- Sampling Accuracy Tests
  - Real-world Security Exploits
Performance of Dataflow Sampling

Throughput (MB/s) vs. Maximum % Time in Analysis for ssh_receive

Throughput with no analysis
Accuracy with Background Tasks

`ssh_receive` running in background

% Chance of Detecting Exploit

- Apache
- Eggdrop
- Lynx
- ProFTPD
- Squid

Maximum % Time in Analysis

10% 25% 50% 75% 90%

- 0.1
- 0.7
- 2
- 0
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints

Demand-Driven Race Detection
Dynamic Data Race Detection

- Add checks around every memory access
- Find inter-thread sharing
- Synchronization between write-shared accesses?
  - No? Data race.
SW Race Detection is Slow

![Race Detector Slowdown Graph]

- **Phoenix**
- **PARSEC**

<table>
<thead>
<tr>
<th>Race Detector Slowdown (x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
</tr>
<tr>
<td>kmeans</td>
</tr>
<tr>
<td>linear_regression</td>
</tr>
<tr>
<td>matrix_multiply</td>
</tr>
<tr>
<td>string_match</td>
</tr>
<tr>
<td>pca</td>
</tr>
<tr>
<td>word_count</td>
</tr>
<tr>
<td>GeoMean</td>
</tr>
<tr>
<td>blackscholes</td>
</tr>
<tr>
<td>bodytrack</td>
</tr>
<tr>
<td>facesim</td>
</tr>
<tr>
<td>ferret</td>
</tr>
<tr>
<td>freqmine</td>
</tr>
<tr>
<td>raytrace</td>
</tr>
<tr>
<td>swaptions</td>
</tr>
<tr>
<td>fluidanimate</td>
</tr>
<tr>
<td>vips</td>
</tr>
<tr>
<td>x264</td>
</tr>
<tr>
<td>canneal</td>
</tr>
<tr>
<td>dedup</td>
</tr>
<tr>
<td>streamcluster</td>
</tr>
<tr>
<td>GeoMean</td>
</tr>
</tbody>
</table>
Inter-thread Sharing is What’s Important

TIME

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Inter-thread Sharing is What’s Important

\[
\text{if}(\text{ptr}==\text{NULL})
\]
\[
\text{len1}=\text{thread\_local}\rightarrow\text{mylen};
\]
\[
\text{ptr}=\text{malloc}(\text{len1});
\]
\[
\text{memcpy}(\text{ptr}, \text{data1}, \text{len1})
\]

\[
\text{if}(\text{ptr}==\text{NULL})
\]
\[
\text{len2}=\text{thread\_local}\rightarrow\text{mylen};
\]
\[
\text{ptr}=\text{malloc}(\text{len2});
\]
\[
\text{memcpy}(\text{ptr}, \text{data2}, \text{len2})
\]
Inter-thread Sharing is What’s Important

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)

Thread-local data
NO SHARING
Inter-thread Sharing is What’s Important

```c
if(ptr == NULL)
    len1 = thread_local->mylen;
    ptr = malloc(len1);
    memcpy(ptr, data1, len1);

if(ptr == NULL)
    len2 = thread_local->mylen;
    ptr = malloc(len2);
    memcpy(ptr, data2, len2);
```

Thread-local data
NO SHARING
Inter-thread Sharing is What’s Important

```c
if(ptr==NULL) {
    len1=thread_local->mylen;
    ptr=malloc(len1);
    memcpy(ptr, data1, len1);
}

if(ptr==NULL) {
    len2=thread_local->mylen;
    ptr=malloc(len2);
    memcpy(ptr, data2, len2);
}
```

Thread-local data

- NO SHARING

Shared data

- NO INTER-THREAD SHARING EVENTS
Inter-thread Sharing is What’s Important

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Inter-thread Sharing is What’s Important

```c
if(ptr==NULL)
    len1=thread_local->mylen;
    ptr=malloc(len1);
    memcpy(ptr, data1, len1)

if(ptr==NULL)
    len2=thread_local->mylen;
    ptr=malloc(len2);
    memcpy(ptr, data2, len2)
```
Very Little Dynamic Sharing

% Write-Sharing Events

<table>
<thead>
<tr>
<th>Phoenix</th>
<th>PARSEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td>0.1</td>
</tr>
<tr>
<td>kmeans</td>
<td>0.2</td>
</tr>
<tr>
<td>linear_regression</td>
<td>0.3</td>
</tr>
<tr>
<td>matrix_multiply</td>
<td>0.4</td>
</tr>
<tr>
<td>pca</td>
<td>0.5</td>
</tr>
<tr>
<td>string_match</td>
<td>0.6</td>
</tr>
<tr>
<td>word_count</td>
<td>0.7</td>
</tr>
<tr>
<td>blackscholes</td>
<td>0.8</td>
</tr>
<tr>
<td>bodytrack</td>
<td>0.9</td>
</tr>
<tr>
<td>facesim</td>
<td>1.0</td>
</tr>
<tr>
<td>ferret</td>
<td>1.1</td>
</tr>
<tr>
<td>freqmine</td>
<td>1.2</td>
</tr>
<tr>
<td>raytrace</td>
<td>1.3</td>
</tr>
<tr>
<td>swaptions</td>
<td>1.4</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>1.5</td>
</tr>
<tr>
<td>vips</td>
<td>1.6</td>
</tr>
<tr>
<td>x264</td>
<td>1.7</td>
</tr>
<tr>
<td>canneal</td>
<td>1.8</td>
</tr>
<tr>
<td>dedup</td>
<td>1.9</td>
</tr>
<tr>
<td>streamcluster</td>
<td>2.0</td>
</tr>
</tbody>
</table>

0.0 0.5 1.0 1.5 2.0 2.5 3.0

% Write-Sharing Events
Run the Analysis On Demand

Multi-threaded Application

Software Race Detector

Inter-thread Sharing Monitor
Run the Analysis On Demand

Multi-threaded Application

Software Race Detector

Local Access

Inter-thread Sharing Monitor
Run the Analysis On Demand

Multi-threaded Application Inter-thread Sharing Monitor

Software Race Detector

Inter-thread Sharing Monitor
Run the Analysis On Demand

- Multi-threaded Application
- Inter-thread sharing
- Inter-thread Sharing Monitor
- Software Race Detector
Run the Analysis On Demand

Software Race Detector

Inter-thread Sharing Monitor
Run the Analysis On Demand

Inter-thread Sharing Monitor
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?

FAULT
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
  - ~100% of accesses cause page faults
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
  - ~100% of accesses cause page faults

- Granularity Gap
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?
  - ~100% of accesses cause page faults

- Granularity Gap
- Per-process not per-thread
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>
# Hardware Sharing Detector

- HITM in Cache Memory: \( W \rightarrow R \) Data Sharing

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>

**Write Y=5**
**Hardware Sharing Detector**

- HITM in Cache Memory: $W \rightarrow R$ Data Sharing

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Y=5</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>I</td>
</tr>
</tbody>
</table>
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

```
Core 1
Y=5  M

Core 2
S     Read Y
I
```
Hardware Sharing Detector

- HITM in Cache Memory: \( W \rightarrow R \) Data Sharing

![Diagram showing Core 1 with Y=5, S, and M, and Core 2 with S and I, with an arrow indicating Read Y connecting the two cores.]
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

![Diagram showing HITM in Cache Memory](image)
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

- Hardware Performance Counters
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

- Hardware Performance Counters
Hardware Sharing Detector

- HITM in Cache Memory: $W \rightarrow R$ Data Sharing

- Hardware Performance Counters
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

- Hardware Performance Counters
Hardware Sharing Detector

- HITM in Cache Memory: $W \rightarrow R$ Data Sharing

- Hardware Performance Counters
Hardware Sharing Detector

- HITM in Cache Memory: $W \rightarrow R$ Data Sharing

- Hardware Performance Counters

![Diagram showing hardware sharing and performance counters.](image-url)
Potential Accuracy & Perf. Problems

- Limitations of Performance Counters
  - Intel HITM only finds $W \rightarrow R$ Data Sharing

- Limitations of Cache Events
  - SMT sharing can’t be counted
  - Cache eviction causes missed events

- Events go through the kernel
On-Demand Analysis on Real HW
On-Demand Analysis on Real HW

Execute Instruction
On-Demand Analysis on Real HW

Execute Instruction

Analysis Enabled?
On-Demand Analysis on Real HW

Execute Instruction

Analysis Enabled?

YES

SW Race Detection
On-Demand Analysis on Real HW

Execute Instruction

Analysis Enabled?

YES

SW Race Detection

Sharing Recently?
On-Demand Analysis on Real HW

- Execute Instruction
- Analysis Enabled?
- SW Race Detection
- Sharing Recently?

YES

YES
On-Demand Analysis on Real HW

- Execute Instruction
- Analysis Enabled?
  - YES: SW Race Detection
  - NO: Disable Analysis
  - YES: Sharing Recently?
  - YES: Analysis Enabled?
On-Demand Analysis on Real HW

HITM Interrupt?

Execute Instruction

Analysis Enabled?

SW Race Detection

Disable Analysis

Sharing Recently?

NO

YES

NO

YES
On-Demand Analysis on Real HW

1. HITM Interrupt?
   - YES: Enable Analysis
   - NO: Analysis Enabled?

2. Analysis Enabled?
   - NO: Execute Instruction
   - YES: SW Race Detection

3. SW Race Detection
   - NO: Sharing Recently?
   - YES: Disable Analysis

4. Sharing Recently?
   - NO: Execute Instruction
   - YES: YES

5. Execute Instruction

6. Disable Analysis

7. YES
On-Demand Analysis on Real HW

1. Execute Instruction
2. Analysis Enabled?
   - NO: HITM Interrupt?
     - NO: Enable Analysis
     - YES: SW Race Detection
   - YES: Disable Analysis
3. Sharing Recently?
   - NO: Analysis Enabled?
   - YES: NO

NO

YES

NO

YES
On-Demand Analysis on Real HW

![Flowchart Diagram]

- **HITM Interrupt?**
  - NO: > 97%
  - YES: Execute Instruction
- **Analysis Enabled?**
  - NO: Enable Analysis
  - YES: SW Race Detection
- **Sharing Recently?**
  - NO: Disable Analysis
  - YES: > 97%
On-Demand Analysis on Real HW

- Execute Instruction

- No
  - HITM Interrupt?
    - No
      - Analysis Enabled?
        - No
          - Enable Analysis
        - Yes
          - SW Race Detection
    - Yes
      - Sharing Recently?
        - No
          - Disable Analysis
        - Yes
          - > 97%

- Yes
  - > 97%

- < 3%
Performance Increases

![Graph showing performance increases for Phoenix and PARSEC with a speedup of 51x.](image)

- **Phoenix**
- **PARSEC**

The graph compares the performance of different benchmarks between Phoenix and PARSEC, showing significant speedup for various applications.
Performance Increases

![Graph showing demand-driven analysis speedup for Phoenix and PARSEC.]

- **Accuracy vs. Continuous Analysis:** 97%

---

**Phoenix vs. PARSEC**

- Speedup (x51)

---

**Bar chart**

- Comparison between Phoenix and PARSEC for various tasks:
  - histogram
  - kmeans
  - linear_regression
  - matrix_multiply
  - pca
  - string_match
  - GeoMean
  - blackscholes
  - bodytrack
  - facesim
  - ferret
  - freqmine
  - raytrace
  - swaptions
  - fluidanimate
  - vips
  - x264
  - canneal
  - dedup
  - streamcluster
  - GeoMean
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints
Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints
Watchpoints Work for Many Analyses

- Bounds Checking
- Data Race Detection
- Taint Analysis
- Deterministic Execution
- Transactional Memory
- Speculative Parallelization
Watchpoints Work for Many Analyses

- Bounds Checking
- Taint Analysis
- Data Race Detection
- Deterministic Execution
- Transactional Memory
- Speculative Parallelization
Desired Watchpoint Capabilities

- Large Number

V W X Y
Desired Watchpoint Capabilities

- Large Number

V W X Y
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread

![Diagram of watchpoints and fault types](image)
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread
- Ranges
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread
- Ranges

![Diagram showing watchpoint fault locations and false faults]
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread
- Ranges
  - Range Cache
## Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Set Addresses**

0x5 – 0x2000

R-Watched
### Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Set Addresses**

0x5 – 0x2000

R-Watched
Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

Set Addresses
0x5 – 0x2000
R-Watched
### Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

**Set Addresses**

0x5 – 0x2000  
R-Watched
## Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>
# Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

Load Address

0x400
**Range Cache**

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

Load Address
0x400
## Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

$\leq 0x400?$  $\geq 0x400$?

### Load Address

0x400
# Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

≤ 0x400?      ≥ 0x400?

Load Address 0x400
## Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \leq 0x400? \quad \geq 0x400? \]

Load Address 0x400
Range Cache

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Watchpoint?</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x4</td>
<td>Not Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x5</td>
<td>0x2000</td>
<td>R Watched</td>
<td>1</td>
</tr>
<tr>
<td>0x2001</td>
<td>0xffff_ffff</td>
<td>Not Watched</td>
<td>1</td>
</tr>
</tbody>
</table>

≤ 0x400?  ≥ 0x400?

Load Address 0x400

WP Interrupt
Watchpoint System Design

- Store Ranges in Main Memory
Watchpoint System Design

- Store Ranges in Main Memory
Watchpoint System Design

- Store Ranges in Main Memory
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow

![Diagram showing T1 Memory, T2 Memory, and Core 1, Core 2 with overlapping and separate ranges]
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter

![Diagram of Watchpoint System Design](image)
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter

T1 Memory  T2 Memory

Core 1  Core 2

WP Changes
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter
- Precise, user-level watchpoint faults

Diagram:
- T1 Memory
- T2 Memory
- Core 1
- Core 2
Experimental Evaluation Setup

- Trace-based timing simulator using Pin
- Taint analysis on SPEC INT2000
- Race Detection on Phoenix and PARSEC
- Comparing only shadow value checks
Watchpoint-Based Taint Analysis

- 128 entry RC –or– 64 entry RC + 2KB Bitmap

Slowdown (x)

10x 30x 206x 423x 23x 28x 1429x 19x

Umbra
VM
MT
RC
RC+ Bitmap
Watchpoint-Based Taint Analysis

- 128 entry RC or 64 entry RC + 2KB Bitmap

<table>
<thead>
<tr>
<th>Slowdown (x)</th>
<th>Umbra</th>
<th>VM</th>
<th>MT</th>
<th>RC</th>
<th>RC+Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>206x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>423x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1429x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20% Slowdown
Watchpoint-Based Data Race Detection

![Graph showing speedup of sharing check for different methods. The methods include VM, MT, RC, and RC+Bitmap. The x-axis represents the benchmarks, and the y-axis represents the speedup.](image-url)
Watchpoint-Based Data Race Detection

Speedup of Sharing Check (x)

-VM
-MT
-RC
-RC+ Bitmap

+10% +20%

histogram kmeans matrix_multiply string_match word_count GeoMean blackscholes bodytrack facesim ferret raytrace swaptions fluidanimate vips x264 canneal dedup streamcluster GeoMean
Future Directions

- Dataflow Tests find bugs on executed code
  - What about code that is never executed?

- Sampling + Demand-Driven Race Detection
  - Good synergy between the two, like taint analysis

- Further watchpoint hardware studies:
  - Clear microarchitectural analysis
  - More software systems, different algorithms
Conclusions
Conclusions

- **Sampling** allows distributed dataflow analysis
Conclusions

- **Sampling** allows distributed dataflow analysis
- **Existing hardware** can speed up race detection
Conclusions

- **Sampling** allows distributed dataflow analysis
- **Existing hardware** can speed up race detection
- **Watchpoint hardware** useful everywhere
Conclusions

- **Sampling** allows distributed dataflow analysis
- **Existing hardware** can speed up race detection
- **Watchpoint hardware** useful everywhere

Distributed Dynamic Software Analysis
Thank You
BACKUP SLIDES
Finding Errors

- Brute Force
  - Code review, fuzz testing, whitehat/grayhat hackers
    - Time-consuming, difficult
Finding Errors

- Brute Force
  - Code review, fuzz testing, whitehat/grayhat hackers
    - Time-consuming, difficult

- Static Analysis
  - Automatically analyze source, formal reasoning, compiler checks
    - Intractable, requires expert input, no system state
Dynamic Dataflow Analysis

- **Associate** meta-data with program values
- **Propagate/Clear** meta-data while executing
- **Check** meta-data for safety & correctness
- **Forms** dataflows of meta/shadow information
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Meta-Data Detection

Non-Shadowed Data
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

![Diagram showing Native Application and Instrumented Application with Meta-Data Detection]

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Shadowed Data

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Meta-Data Detection

Instrumented Application

Native Application
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Meta-Data Detection
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data

Native Application

Instrumented Application

Meta-Data Detection
Results by Ho et al.

- **Imbench Best Case Results:**

<table>
<thead>
<tr>
<th>System</th>
<th>Slowdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taint Analysis</td>
<td>101.7x</td>
</tr>
<tr>
<td>On-Demand Taint Analysis</td>
<td>1.98x</td>
</tr>
</tbody>
</table>

- **Results when everything is tainted:**

  ![Graph showing slowdown for different systems](image)

  - netcat_transmit: 150x slowdown
  - netcat_receive: 10x slowdown
  - ssh_transmit: 110x slowdown
  - ssh_receive: 100x slowdown
Sampling Allows Distribution

- No Analysis
- Complete Analysis

Error Detection Rate vs. Overhead

- Developer
Sampling Allows Distribution

- Error Detection Rate vs. Overhead
- Beta Testers
- No Analysis: Lower Error Detection Rate, Lower Overhead
- Complete Analysis: Higher Error Detection Rate, Higher Overhead

Developer
Sampling Allows Distribution

Error Detection Rate vs Overhead

- No Analysis
- Complete Analysis
- End Users
- Beta Testers
- Developer
Sampling Allows Distribution

Many users testing at little overhead see more errors than one user at high overhead.
Cannot Naïvely Sample Code
Cannot Naively Sample Code

\[ a += y \]
\[ y = x \times 1024 \]
\[ x = \text{read\_input}() \]
Cannot Naïvely Sample Code

a += y

y = x * 1024

x = read_input()
Cannot Naïvely Sample Code

\[ a += y \]

\[ z = y \times 75 \]

\[ y = x \times 1024 \]

\[ x = \text{read}_\text{input}() \]

Cannot Naïvely Sample Code
Cannot Naïvely Sample Code

\[ a += y \]

\[ z = y \times 75 \]

\[ y = x \times 1024 \]

\[ x = \text{read\_input()} \]

\[ \text{validate}(x) \]

Skip Instr.
Cannot Naïvely Sample Code

\[
\begin{align*}
a &\ += y \\
y &\ = x \ast 1024 \\
z &\ = y \ast 75 \\
x &\ = \text{read_input}() \\
\text{validate}(x)
\end{align*}
\]
Cannot Naïvely Sample Code

\[
a += y
\]
\[
z = y \times 75
\]
\[
y = x \times 1024
\]
\[
w = x + 42
\]

\[
x = \text{read}_\text{input}()
\]
\[
\text{validate}(x)
\]
\[
a += y
\]
\[
z = y \times 75
\]
Cannot Naïvely Sample Code

```
a += y

z = y * 75

y = x * 1024

w = x + 42

validate(x)

Input

x = read_input()
```

Check w

Check z

Check a
Cannot Naïvely Sample Code

```
a += y
z = y * 75
y = x * 1024
w = x + 42
```

```
Cannot Naïvely Sample Code
```
Dataflow Sampling Example

![Image of a switch with 'Input' button]
Dataflow Sampling Example

\[ \text{Input} \]
\[ x = \text{read_input()} \]
\[ y = x \times 1024 \]
\[ a += y \]
Dataflow Sampling Example

\[ a += y \]
\[ y = x \times 1024 \]
\[ x = \text{read\_input()} \]

Input

\[ a += y \]

\[ y = x \times 1024 \]

\[ x = \text{read\_input()} \]
Dataflow Sampling Example

\[ a += y \]
\[ z = y * 75 \]
\[ y = x * 1024 \]
\[ x = \text{read}_\text{input}() \]
Dataflow Sampling Example

```
Input

x = read_input()

y = x * 1024

a += y

z = y * 75

Skip Dataflow
```
Dataflow Sampling Example

\[
a += y
\]

\[
z = y \times 75
\]

\[
y = x \times 1024
\]

\[
x = \text{read\_input()}\]

\[
\text{validate}(x)
\]
Dataflow Sampling Example

\[ a += y \]
\[ z = y \times 75 \]
\[ y = x \times 1024 \]
\[ w = x + 42 \]

\[ x = \text{read\_input}() \]

\[ \text{validate}(x) \]
Dataflow Sampling Example

$\text{Input}$

$x = \text{read\_input}()$

$y = x \times 1024$

$z = y \times 75$

$w = x + 42$

$\text{validate}(x)$

$a += y$

Check $w$

Check $z$

Check $a$

False Negative
Benchmarks

- Performance – Network Throughput
  - *Example:* `ssh_receive`

- Accuracy of Sampling Analysis
  - Real-world Security Exploits

<table>
<thead>
<tr>
<th>Name</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Stack overflow in Apache Tomcat JK Connector</td>
</tr>
<tr>
<td>Eggdrop</td>
<td>Stack overflow in Eggdrop IRC bot</td>
</tr>
<tr>
<td>Lynx</td>
<td>Stack overflow in Lynx web browser</td>
</tr>
<tr>
<td>ProFTPD</td>
<td>Heap smashing attack on ProFTPD Server</td>
</tr>
<tr>
<td>Squid</td>
<td>Heap smashing attack on Squid proxy server</td>
</tr>
</tbody>
</table>
Performance of Dataflow Sampling (2)

Throughput with no analysis

Throughput (MB/s) vs. Maximum % Time in Analysis for netcat_receive
Performance of Dataflow Sampling (3)

Throughput (MB/s) vs. Maximum % Time in Analysis for ssh_transmit

Throughput with no analysis
Accuracy at Very Low Overhead

- Max time in analysis: 1% every 10 seconds
- Always stop analysis after threshold
  - Lowest probability of detecting exploits

<table>
<thead>
<tr>
<th>Name</th>
<th>Chance of Detecting Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>100%</td>
</tr>
<tr>
<td>Eggdrop</td>
<td>100%</td>
</tr>
<tr>
<td>Lynx</td>
<td>100%</td>
</tr>
<tr>
<td>ProFTPD</td>
<td>100%</td>
</tr>
<tr>
<td>Squid</td>
<td>100%</td>
</tr>
</tbody>
</table>
Accuracy with Background Tasks

netcat_receive running with benchmark

% Chance of Detecting Exploit

- Apache
- Eggdrop
- Lynx
- ProFTPD
- Squid

Maximum Allowed Overhead %

10% 25% 50% 75% 90%
Outline

- Problem Statement

- Proposed Solutions
  - Distributed Dynamic Dataflow Analysis
  - Testudo: Hardware-Based Dataflow Sampling
  - Demand-Driven Data Race Detection

- Future Work

- Timeline
Outline

- Problem Statement
- Proposed Solutions
  - Distributed Dynamic Dataflow Analysis
  - Testudo: Hardware-Based Dataflow Sampling
  - Demand-Driven Data Race Detection
- Future Work
- Timeline
Virtual Memory Not Ideal
Virtual Memory Not Ideal

FAULT
Virtual Memory Not Ideal

FAULT
Virtual Memory Not Ideal

Throughput (MB/s) vs. Maximum % Time in Analysis

FAULT

netcat_receive
What happens when the cache overflows?
- Increase the size of main memory?
- Store into virtual memory?

Use Sampling to Throw Away Data
On-Chip Sampling Mechanism

512-entry cache

Avg. # of executions

5000
4000
3000
2000
1000
0

pdf
sql_injct

17601

1024-entry cache

5000
4000
3000
2000
1000
0

pdf
sql_injct

17601
Useful for Scaling to Complex Analyses

If each shadow operation uses 1000 instructions:

![Bar Chart]

Average % Overhead

- pdf
- sql_injct

1024-entry Sample Cache
Useful for Scaling to Complex Analyses

If each shadow operation uses 1000 instructions:

- pdf: 0.23%
- sql_injct: 0.20%

Average % Overhead

1024 entry Sample Cache

- pdf: 0.23%
- sql_injct: 0.20%

Average % Overhead

- telnet server benchmark: 17.3%
- 512 entry: 0.3%
Useful for Scaling to Complex Analyses

If each shadow operation uses 1000 instructions:

Average % Overhead

<table>
<thead>
<tr>
<th>pdf</th>
<th>sql_injct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>0.19</td>
</tr>
</tbody>
</table>

1024-entry Sample Cache

Average % Overhead

<table>
<thead>
<tr>
<th>1024 entry telnet server benchmark</th>
<th>512 entry benchmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.3%</td>
<td>0.3%</td>
</tr>
</tbody>
</table>

3500 executions

169,000 executions
Example of Data Race Detection

Thread 1
mylen=small

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Example of Data Race Detection

Thread 1
mylen=small

if(ptr==NULL)

len1=thread_local->mylen;

ptr=malloc(len1);

memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)

len2=thread_local->mylen;

ptr=malloc(len2);

memcpy(ptr, data2, len2)
Example of Data Race Detection

Thread 1
mylen=small

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Example of Data Race Detection

Thread 1
mylen=small

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

tp write-shared?

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Example of Data Race Detection

Thread 1
mylen=small

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)

Thread 2
mylen=large

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)

ptr write-shared?
Example of Data Race Detection

Thread 1
mylen=small

if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1);

Thread 2
mylen=large

Interleaved Synchronization?

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
Example of Data Race Detection

Thread 1  
mylen=small

len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1);

Thread 2  
mylen=large

if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)

Interleaved Synchronization?

if(ptr==NULL)
Demand-Driven Analysis Algorithm

- **Thread Executes Instruction**
- **Synchro Operation?**
  - **YES** → **Mark All Data Unshared** → **Update Lamport Clocks**
  - **NO**
- **Memory Operation?**
  - **YES** → **Shared with another thread?**
    - **YES** → **Write shared?**
      - **YES** → **Run Software Race Detector**
      - **NO** → **Update Lamport Clocks**
    - **NO** → **Update Lamport Clocks**
  - **NO**
Demand-Driven Analysis on Real HW
Performance Difference

<table>
<thead>
<tr>
<th>Phoenix</th>
<th>PARSEC</th>
</tr>
</thead>
</table>

Race Detector Slowdown (x)
Demand-Driven Analysis Accuracy

The graph illustrates the speedup (x) for various benchmarks, with error bars indicating the variability. The benchmarks include:

- histogram
- kmeans
- linear_regression
- matrix_multiply
- pca
- string_match
- word_count
- GeoMean
- blackscholes
- bodytrack
- facesim
- ferret
- freqmine
- raytrace
- swaptions
- fluidanimate
- vips
- x264
- canneal
- dedup
- streamcluster
- GeoMean

The y-axis represents the demand-driven analysis speedup, while the x-axis lists the benchmarks.
Demand-Driven Analysis Accuracy

Speedup (x)

- histogram
- kmeans
- linear_regression
- matrix_multiply
- string_match
- word_count
- GeoMean
- blackscholes
- bodytrack
- facesim
- ferret
- freqmine
- raytrace
- swaptions
- fluidanimate
- vips
- x264
- canneal
- dedup
- streamcluster
- GeoMean

Demand-driven Analysis
Demand-Driven Analysis Accuracy

[Graph showing various benchmarks and their corresponding demand-driven analysis speedup values. The graph includes benchmarks such as histogram, kmeans, matrix_multiply, pca, word_count, GeoMean, blackscholes, bodytrack, facesim, ferret, fannet, raytrace, spawtions, fluidanimate, vips, x264, canneal, dedup, streamcluster, and GeoMean.]
Demand-Driven Analysis Accuracy

Accuracy vs. Continuous Analysis: 97%
## Accuracy on Real Hardware

<table>
<thead>
<tr>
<th></th>
<th>kmeans</th>
<th>facesim</th>
<th>ferret</th>
<th>freqmine</th>
<th>vips</th>
<th>x264</th>
<th>streamcluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>W→W</td>
<td>1/1 (100%)</td>
<td>0/1 (0%)</td>
<td>-</td>
<td>-</td>
<td>1/1 (100%)</td>
<td>-</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>R→W</td>
<td>-</td>
<td>0/1 (0%)</td>
<td>2/2 (100%)</td>
<td>2/2 (100%)</td>
<td>1/1 (100%)</td>
<td>3/3 (100%)</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>W→R</td>
<td>-</td>
<td>2/2 (100%)</td>
<td>1/1 (100%)</td>
<td>2/2 (100%)</td>
<td>1/1 (100%)</td>
<td>3/3 (100%)</td>
<td>1/1 (100%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spider Monkey-0</th>
<th>Spider Monkey-1</th>
<th>Spider Monkey-2</th>
<th>NSPR-1</th>
<th>Memcached-1</th>
<th>Apache-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>W→W</td>
<td>9/9 (100%)</td>
<td>1/1 (100%)</td>
<td>1/1 (100%)</td>
<td>3/3 (100%)</td>
<td>-</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>R→W</td>
<td>3/3 (100%)</td>
<td>-</td>
<td>1/1 (100%)</td>
<td>1/1 (100%)</td>
<td>1/1 (100%)</td>
<td>7/7 (100%)</td>
</tr>
<tr>
<td>W→R</td>
<td>8/8 (100%)</td>
<td>1/1 (100%)</td>
<td>2/2 (100%)</td>
<td>4/4 (100%)</td>
<td>-</td>
<td>2/2 (100%)</td>
</tr>
</tbody>
</table>
## Accuracy on Real Hardware

<table>
<thead>
<tr>
<th></th>
<th>kmeans</th>
<th>facesim</th>
<th>ferret</th>
<th>freqmine</th>
<th>vips</th>
<th>x264</th>
<th>streamcluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>W→W</td>
<td>1/1 (100%)</td>
<td>0/1 (0%)</td>
<td>-</td>
<td>-</td>
<td>1/1 (100%)</td>
<td>-</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>R→W</td>
<td>-</td>
<td>0/1 (0%)</td>
<td>2/2 (100%)</td>
<td>2/2 (100%)</td>
<td>1/1 (100%)</td>
<td>3/3 (100%)</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>W→R</td>
<td>-</td>
<td>2/2 (100%)</td>
<td>1/1 (100%)</td>
<td>2/2 (100%)</td>
<td>1/1 (100%)</td>
<td>3/3 (100%)</td>
<td>1/1 (100%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spider Monkey-0</th>
<th>Spider Monkey-1</th>
<th>Spider Monkey-2</th>
<th>NSPR-1</th>
<th>Memcached-1</th>
<th>Apache-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>W→W</td>
<td>9/9 (100%)</td>
<td>1/1 (100%)</td>
<td>1/1 (100%)</td>
<td>3/3 (100%)</td>
<td>-</td>
<td>1/1 (100%)</td>
</tr>
<tr>
<td>R→W</td>
<td>3/3 (100%)</td>
<td>-</td>
<td>1/1 (100%)</td>
<td>1/1 (100%)</td>
<td>1/1 (100%)</td>
<td>7/7 (100%)</td>
</tr>
<tr>
<td>W→R</td>
<td>8/8 (100%)</td>
<td>1/1 (100%)</td>
<td>2/2 (100%)</td>
<td>4/4 (100%)</td>
<td>-</td>
<td>2/2 (100%)</td>
</tr>
</tbody>
</table>
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

0 1 2 3 4 5 6 7
A B C D E F G H

LD 2
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
0 1 2 3 4 5 6 7
A B C D E F G H
```

LD 2
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
0 1 2 3 4 5 6 7
A B C D E F G H
WR X→7
```
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
0 1 2 3 4 5 6 7
A B C D E F G H
```

WR X→7
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
0 1 2 3 4 5 6 7
A B C D E F G X
```

**WR X→7**
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
 0  1  2  3  4  5  6  7
A  B  C  D  E  F  G  X
```

R-Watch 2-4
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```plaintext
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>X</td>
</tr>
</tbody>
</table>
```

R-Watch 2-4
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

W-Watch 6-7
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

W-Watch 6-7
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>X</td>
</tr>
</tbody>
</table>
```

LD 2
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>X</td>
</tr>
</tbody>
</table>
```

LD 2

LD 2
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
    0  1  2  3  4  5  6  7
   A   B   C   D   E   F   G   X
```

WR X→7
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

```
0  1  2  3  4  5  6  7
A  B  C  D  E  F  G  X
```

WR X → 7
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

- SW knows it’s touching important data

```plaintext
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>X</td>
</tr>
</tbody>
</table>
```
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data
- SW knows it’s touching important data
  - AT NO OVERHEAD
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data
  - SW knows it’s touching important data
    - AT NO OVERHEAD
- Normally used for debugging
Existing Watchpoint Solutions

- Watchpoint Registers
  - Limited number (4-16), small reach (4-8 bytes)
Existing Watchpoint Solutions

- **Watchpoint Registers**
  - Limited number (4-16), small reach (4-8 bytes)

- **Virtual Memory**
  - Coarse-grained, per-process, *only* aligned ranges
Existing Watchpoint Solutions

- **Watchpoint Registers**
  - Limited number (4-16), small reach (4-8 bytes)

- **Virtual Memory**
  - Coarse-grained, per-process, *only* aligned ranges

- **ECC Mangling**
  - Per physical address, all cores, no ranges
Meeting These Requirements

- Unlimited Number of Watchpoints
  - Store in memory, cache on chip

- Fine-Grained
  - Watch full virtual addresses

- Per-Thread
  - Watchpoints cached per core/thread
  - TID Registers

- Ranges
  - Range Cache
The Need for Many Small Ranges

- Some watchpoints better suited for ranges

  - 32b Addresses: 2 ranges x 64b each = 16B
The Need for Many Small Ranges

- Some watchpoints better suited for ranges
  - 32b Addresses: 2 ranges x 64b each = 16B

- Some need large # of small watchpoints
The Need for Many Small Ranges

- Some watchpoints better suited for ranges
  - 32b Addresses: 2 ranges x 64b each = 16B

- Some need large # of small watchpoints
  - 51 ranges x 64b each = 408B
  - Better stored as bitmap? 51 bits!
The Need for Many Small Ranges

- Some watchpoints better suited for ranges
  - 32b Addresses: 2 ranges x 64b each = 16B

- Some need large # of small watchpoints
  - 51 ranges x 64b each = 408B
  - Better stored as bitmap? 51 bits!

- Taint analysis has good ranges
- Byte-accurate race detection does not..
Watchpoint System Design II

- Make some RC entries point to bitmaps
Watchpoint System Design II

- Make some RC entries point to bitmaps

<table>
<thead>
<tr>
<th>Start Addr</th>
<th>End Addr</th>
<th>R</th>
<th>W</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>
Watchpoint System Design II

- Make some RC entries point to bitmaps

```
Start Addr  End Addr  R  W  V
---------------  ---  ---  --
```
Watchpoint System Design II

- Make some RC entries point to bitmaps

Start Addr | End Addr | R | W | V | B
--- | --- | --- | --- | --- | ---
Watchpoint System Design II

- Make some RC entries point to bitmaps

Start Addr | End Addr | R | W | V | B | Pointer to WP Bitmap

- - - 1 1
Watchpoint System Design II

- Make some RC entries point to bitmaps

<table>
<thead>
<tr>
<th>Start Addr</th>
<th>End Addr</th>
<th>Pointer to WP Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>V</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Memory

Ranges

Core

Range Cache
Watchpoint System Design II

- Make some RC entries point to bitmaps
Watchpoint System Design II

- Make some RC entries point to bitmaps

<table>
<thead>
<tr>
<th>Start Addr</th>
<th>End Addr</th>
<th>Pointer to WP Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>W</td>
<td>V</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory Ranges

Core Range Cache

Bitmap Cache
Watchpoint System Design II

- Make some RC entries point to bitmaps

<table>
<thead>
<tr>
<th>Start Addr</th>
<th>End Addr</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>V</th>
<th>B</th>
<th>Pointer to WP Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Memory
Ranges
Bitmaps

Core
Range Cache
Bitmap Cache

Accessed in Parallel
Watchpoint-Based Taint Analysis

- 128 entry Range Cache
Watchpoint-Based Taint Analysis

128 entry Range Cache

- 20% Slowdown

- 10x slowdown
- 30x slowdown
- 206x slowdown
- 423x slowdown
- 1429x slowdown

- Umbra
- VM
- MT
- RC

Slowdown (x)
Width Test