Hardware Support for On-Demand Software Analysis

Joseph L. Greathouse
Advanced Computer Architecture Laboratory
University of Michigan

December 8, 2011
Software Errors Abound

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

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

- NIST: Software errors cost U.S. ~$60 billion/year
A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c
Software Errors Abound

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

A problem has been detected and Windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SRCMOCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you’ve seen this stop error screen, restart your computer. If this screen appears again, follow these steps:

- Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.
- If problems continue, disable or remove any newly installed hardware or software, disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to disable or remove components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

```
*** STOP: 0x00000005 (0x803044c2,0x00000001,0xfbe7617,0x00000002)

*** SRCMOCON.SYS - Address FBE7617 base at FBFE5000, DateStamp 3d6dd7c
```
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
  - >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.

---

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable non-essential memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical Information:

*** STOP: 0X00000000 (0XFD30BAC2, 0X00000001, 0XFBFE7617, 0X00000000)

*** SPCMODCON.SYS - Address FBFE7617 base at FBFEE000, DateStamp 3d6d67c
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.

Adobe Warns of Critical Zero Day Vulnerability

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

Parallel Programming is Here – And it’s Hard!
Parallel Programming is Here – And it’s Hard!
Hardware Plays a Role

Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
X + Y &= 10 \\
Y + Z &= 20
\end{align*}
\]

\[
Y = \quad Z =
\]

\[
X = 5
\]

\[
X = 0
\]
Parallel Programming is Here – And it’s Hard!

\[ x + y = 10 \]
\[ y + z = 20 \]

\[ y = \]
\[ z = \]
Hardware Plays a Role

Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
X + Y &= 10 \\
Y + Z &= 20
\end{align*}
\]

\[
\begin{align*}
Y &= 5 \\
Z &=
\end{align*}
\]
Hardware Plays a Role

Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
x + y &= 10 \\
y + z &= 20
\end{align*}
\]

\[
\begin{align*}
y &= 5 \\
z &= \quad \quad \quad (X = 0)
\end{align*}
\]
Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
    x + y &= 10 \\
    y + z &= 20
\end{align*}
\]

\[
\begin{align*}
    y &= 10 \\
    z &=
\end{align*}
\]
Parallel Programming is Here – And it’s Hard!

\[ \begin{align*}
  x + y &= 10 \\
  y + z &= 20 \\
  \hline
  y &= 10 \\
  z &= \quad
\end{align*} \]
Hardware Plays a Role

Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
    x + y &= 10 \\
    y + z &= 20 \\
    \hline
    y &= 10 \\
    z &= \phantom{0}
\end{align*}
\]
Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
\text{X} + \text{Y} &= 10 \\
\text{Y} + \text{Z} &= 20 \\
\hline
\text{Y} &= 10 \\
\text{Z} &= 10
\end{align*}
\]
Parallel Programming is Here – And it’s Hard!

\[ x + y = 10 \]
\[ y + z = 20 \]

\[ \begin{align*}
\text{Y} & = 10 \\
\text{Z} & = 10
\end{align*} \]
Hardware Plays a Role

Parallel Programming is Here – And it’s Hard!

\[
\begin{align*}
  x + y &= 10 \\
  y + z &= 20 \\
  \hline
  y &= 10 \\
  z &= 10
\end{align*}
\]
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

ptr
∅
Example of a Modern Bug

Thread 1
mylen=small

if(ptr==NULL)

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

Thread 2
mylen=large

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

Thread 1
mylen=small

if(ptr==NULL)

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

Thread 2
mylen=large

if(ptr==NULL)

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

ptr

LEAKED
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)

LENKED
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
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
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)
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)
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)
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)

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

len2 = thread_local -> mylen;

ptr = malloc(len2);

memcpy(ptr, data2, len2)

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

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

Synchronized?
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)

Synchronized?
Data Race Detection is Slow

![Graph showing race detector slowdowns for Phoenix and PARSEC benchmarks.](graph.png)

- Phoenix benchmarks include histogram, kmeans, matrix_multiply, pca, and geo_mean, with varying race detector slowdowns.
- PARSEC benchmarks show similar trends with additional benchmarks like raytrace, fluidanimate, vips, x264, and streamcluster, also with GeoMean at the end.
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);
```
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

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

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

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

Thread-local data
NO SHARING

Shared data, but
NO DYNAMIC
SHARING
Inter-thread Sharing is What’s Important

- **Thread-local data**
  - NO SHARING

- **Shared data, but NO DYNAMIC SHARING**

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

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

<table>
<thead>
<tr>
<th>% Write-Sharing Events</th>
<th>Phoenix</th>
<th>PARSEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kmeans</td>
<td></td>
<td></td>
</tr>
<tr>
<td>linear_regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matrix_multiply</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pca</td>
<td></td>
<td></td>
</tr>
<tr>
<td>string_match</td>
<td></td>
<td></td>
</tr>
<tr>
<td>word_count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>blackscholes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bodytrack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>facesim</td>
<td></td>
<td></td>
</tr>
<tr>
<td>freqmine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>raytrace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>swaptions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vips</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td></td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dedup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Very Little Dynamic Sharing

<table>
<thead>
<tr>
<th>% Write-Sharing Events</th>
<th>Phoenix</th>
<th>PARSEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>kmeans</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>linear_regression</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>matrix_multiply</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>pca</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>string_match</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>word_count</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>blackscholes</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>bodytrack</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>facesim</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>freqmine</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>raytrace</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>swaptions</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>vips</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>x264</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>canneal</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>dedup</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>streamcluster</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

0 0.5 1 1.5 2 2.5 3

bar chart showing % Write-Sharing Events for Phoenix and PARSEC for various applications.
Little Sharing Means Wasted Work

Multi-threaded Application

Software Race Detector
Little Sharing Means Wasted Work

Inter-thread sharing

Software Race Detector
Little Sharing Means Wasted Work
Little Sharing Means Wasted Work
Little Sharing Means Wasted Work

Local Access

Software Race Detector
Little Sharing Means Wasted Work
Little Sharing Means Wasted Work

Multi-threaded Application
Use Demand-Driven Analysis!

Multi-threaded Application

Software Race Detector

Inter-thread Sharing Monitor
Use Demand-Driven Analysis!

Multi-threaded Application

Software Race Detector

Local Access

Inter-thread Sharing Monitor
Use Demand-Driven Analysis!

Inter-thread Sharing Monitor

Local Access

Software Race Detector

Multi-threaded Application Software Race Detector
Use Demand-Driven Analysis!

Inter-thread Sharing Monitor

Software Race Detector

Inter-thread sharing
Use Demand-Driven Analysis!

- Multi-threaded Application Software
- Race Detector
- Inter-thread sharing
- Inter-thread Sharing Monitor
Use Demand-Driven Analysis!

Inter-thread Sharing Monitor
Use Demand-Driven Analysis!

Inter-thread Sharing Monitor
Use Demand-Driven Analysis!

Inter-thread Sharing Monitor

Multi-threaded Application

Local Access

Software Race Detector
Use Demand-Driven Analysis!

Inter-thread Sharing Monitor

Local Access

Software Race Detector

Multi-threaded Application
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

```
Core 1  Core 2
S       S
I       I
```
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>

Write Y=5
```
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>Y=5</td>
<td>Read Y</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
</tr>
</tbody>
</table>

M = 5
Hardware Sharing Detector

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

![Diagram](image_url)

- Core 1:
  - S
  - M
  - \( Y = 5 \)

- Core 2:
  - S
  - I
  - Read Y
Hardware Sharing Detector

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

![Diagram showing data sharing between two cores](image)
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

- Hardware Performance Counters

Core 1

- S
- Y=5
- M

Core 2

- S

Pipeline

- Perf. Ctrs
  - 0
  - 0
  - 0
  - 0

Cache
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

Core 1

| S  | Y=5 | M |
Core 2

| S  |
Read Y

HITM

- Hardware Performance Counters

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Perf. Ctrs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>Cache</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

- Hardware Performance Counters
On-Demand Analysis on Real HW

Execute Instruction
On-Demand Analysis on Real HW

Execute Instruction

Analysis Enabled?
On-Demand Analysis on Real HW

1. Execute Instruction
2. Analysis Enabled?
   - YES: SW Race Detection
On-Demand Analysis on Real HW

1. Execute Instruction
2. Analysis Enabled? (YES)
3. SW Race Detection
4. Sharing Recently?
On-Demand Analysis on Real HW

1. Execute Instruction
2. Analysis Enabled?
   - Yes: SW Race Detection
   - No: Execute Instruction
3. SW Race Detection
4. Sharing Recently?
   - Yes: Execute Instruction
   - No: Analysis Enabled?
On-Demand Analysis on Real HW

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

- HITM Interrupt?
  - NO
  - Analysis Enabled?
    - NO
    - SW Race Detection
    - YES
    - Disable Analysis
      - NO
      - Sharing Recently?
        - YES
      - YES
    - YES
- Execute Instruction
  - YES
  - NO

On-Demand Analysis on Real HW

1. Execute Instruction
2. Analysis Enabled?
3. SW Race Detection
4. Sharing Recently?
5. Disable Analysis
6. HITM Interrupt?
7. Enable Analysis
On-Demand Analysis on Real HW

- Execute Instruction
- SW Race Detection
- Analysis Enabled?
  - YES
  - NO
- Disable Analysis
- Sharing Recently?
  - NO
  - YES

- Enable Analysis
- HITM Interrupt?
  - YES
  - NO

89
On-Demand Analysis on Real HW

- Execute Instruction
- SW Race Detection
- Enable Analysis
- Disable Analysis
- HITM Interrupt?
- Analysis Enabled?
- Sharing Recently?

> 97%
On-Demand Analysis on Real HW

- Execute Instruction
- Analysis Enabled?
  - Yes
    - SW Race Detection
  - No
    - Disable Analysis

- HITM Interrupt?
  - Yes
    - Enable Analysis
  - No
    - > 97%
      - NO
        - < 3%
          - YES
          - NO
Performance Increases

Demand-driven Analysis Speedup (x)

Phoenix   PARSEC

<table>
<thead>
<tr>
<th></th>
<th>Phoenix</th>
<th>PARSEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>kmeans</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>matrix_multiply</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>string</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>word_count</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>GeoMean</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>blackscholes</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>bodytrack</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>facesim</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>ferret</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>fregnine</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>raytrace</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>swaptions</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>vips</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>canmeal</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>dedup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GeoMean</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Performance Increases

Demand-driven Analysis

<table>
<thead>
<tr>
<th>Speedup (x)</th>
<th>Phoenix</th>
<th>PARSEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>kmeans</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>linear_regression</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>matrix_multiply</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>string_match</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>word_count</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>GeoMean</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>blackscholes</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>bodytrack</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>facesim</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>ferret</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>fregmine</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>raytrace</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>swaptions</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>vips</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>dedup</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>GeoMean</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

51x
Performance Increases

Accuracy vs. Continuous Analysis: 97%
In Summary

Hardware makes constructing software difficult.

Tools make software better.

Hardware can (and should!) help these tools.
BACKUP SLIDES
Width Test