Embedded System Programming

Multicore ES
(Module 40)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014
The Era of Multi-core Processors

- Will the application run correctly
  - a benign data race may become a true race
  - scheduling anomaly
- How can we debug and monitor ES on multicore processors
Debugging Embedded Software

- In a 2002 NIST survey, an average bug found in post-product release takes 15.3 hours to fix.
  - Cost of software development and product liability
  - Testing process and software release time
- Finding bugs in multithreaded programs is difficult
  - The bug and symptom are widely separated in space and time
  - The system is nondeterministic
  - The occurrence of potential errors may only be triggered after a long period of execution.
- Why is it challenging?
  - Probe effect may alter program behavior
  - Logged data could be enormous
Reproducible Execution

- Execution information must be logged for re-execution
- Overhead – ordering information or data, probe effect
- Static or dynamic (instrumentation at source or object code level)
Approach to Reproducible Execution

- Execution sequence → Partial order of synchronous events
- Preserve the order and apply the same IO events → reproducible execution
Existence of Probe Effect

- Any instrumentation of multi-threaded program execution may
  - change the temporal behavior of program execution
  - result in different ordering of execution events

- To detect event order variations caused by instrumentation
  - simulate program execution based on execution time (w/o overhead), arrival events, synchronization and scheduling actions.
  - program events from instrumented execution with execution time
  - interrupts arrive at absolute time
Test Cases on Probe Effect (1)

- Total order is changed but with same partial order

Instrumented

Philosopher 2

Philosopher 4

Simulation

Philosopher 2

Philosopher 4

35 36 37 38 39 40

Keyboard Interrupt

Real-time Systems Lab, Computer Science and Engineering, ASU
Test Cases on Probe Effect (2)

- Different logical order leading to different execution path

Instrumented

<table>
<thead>
<tr>
<th>23.0M</th>
<th>23.5M</th>
<th>24.0M</th>
<th>24.5M</th>
<th>25.0M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barber</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulation

<table>
<thead>
<tr>
<th>23.0M</th>
<th>23.5M</th>
<th>24.0M</th>
<th>24.5M</th>
<th>25.0M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barber</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer 5</td>
<td>SW(4)</td>
<td>SP(2)</td>
<td>SP(4)</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Keyboard Interrupt

Serve the three customers

A chair available for him

Three available chairs taken

A chair NOT available for him
Data Race Detectors

- A shared location is accessed by two different threads that
  - are not ordered by happens-before relation
  - at least one of the accesses is a write
- Many detectors for Java programs
- Static detectors – false alarms
- Dynamic detectors – need to instrument data accesses
- LockSet algorithms (Eraser) -- imprecise
- Happens-before algorithms – based on Lamport’s vector clock
Race Detector with Dynamic Granularity

- **Vector clock based data race detector for C/C++ programs**
  - On top of FastTrack and using Intel PIN for dynamic instrumentation
  - No need for a full VC on variables
  - VC from $O(n)$ to $O(1)$

- **Share vector clock with neighboring memory locations**
  - Neighboring memory locations tend to be protected by the same lock (e.g. array, struct)
### Performance Benchmark (1)

#### Comparison with Valgrind DRD and Inspector XE

<table>
<thead>
<tr>
<th>Benchmark program</th>
<th>Base time (sec)</th>
<th>Base Mem. (MB)</th>
<th>Slowdown</th>
<th>Memory Overhead</th>
<th>Data race detected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valgrind DRD</td>
<td>Intel Inspector XE</td>
<td>Dynamic granularity</td>
</tr>
<tr>
<td>facesim</td>
<td>6.1</td>
<td>288</td>
<td>59</td>
<td>128</td>
<td>102</td>
</tr>
<tr>
<td>ferret</td>
<td>6.7</td>
<td>146</td>
<td>748</td>
<td>87</td>
<td>52</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>2.0</td>
<td>248</td>
<td>--</td>
<td>89</td>
<td>81</td>
</tr>
<tr>
<td>raytrace</td>
<td>9.5</td>
<td>170</td>
<td>42</td>
<td>17</td>
<td>27</td>
</tr>
<tr>
<td>dedup</td>
<td>7.7</td>
<td>2682</td>
<td>--</td>
<td>--</td>
<td>85</td>
</tr>
<tr>
<td>streamcluster</td>
<td>3.8</td>
<td>30</td>
<td>66</td>
<td>108</td>
<td>137</td>
</tr>
<tr>
<td>ffmpeg</td>
<td>3.0</td>
<td>95</td>
<td>120</td>
<td>--</td>
<td>109</td>
</tr>
<tr>
<td>pbzip2</td>
<td>5.7</td>
<td>67</td>
<td>64</td>
<td>99</td>
<td>39</td>
</tr>
<tr>
<td>hmmsearch</td>
<td>26.6</td>
<td>23</td>
<td>74</td>
<td>64</td>
<td>45</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>168</td>
<td>85</td>
<td>75</td>
</tr>
</tbody>
</table>

Real-time Systems Lab, Computer Science and Engineering, ASU
Conclusion

- **Continuous improvement for the replay mechanism**
  - Record network messages at a sniff server
  - Checkpointing for long running systems

- **Multicore**
  - To overcome potential problems caused by concurrency and scheduling

---

**Correctness**

**Performance**

**Real-time Multicore Application**

**Concurrency & Synchronization**