The Influence of Environmental Parameters on Concurrency Fault Exposures
An Exploratory Study

Sahitya Kakarla
AdVanced Empirical Software Testing and Analysis (AVESTA)
Department of Computer Science
Texas Tech University, USA
sahitya.kakarla@ttu.edu

Akbar Siami Namin
AdVanced Empirical Software Testing and Analysis (AVESTA)
Department of Computer Science
Texas Tech University, USA
akbar.namin@ttu.edu

International Symposium on Empirical Software Engineering and Measurements (ESEM 2010)
Bozano-Bozen, Italy, September 2010
Outline

- Motivation
- Environmental parameters
- Multi-core systems: A case study
  - Experimental procedure
  - Data analysis
- Discussion
- Conclusion
Motivation
The Importance of Concurrency

- The advent of multicore systems
- The future trends in exploiting the power of multiple cores
- The software industry needs programmers capable of developing multi-threaded applications
- Developing parallel applications is harder than programming sequential code
- Testing concurrent programs is much harder than testing sequential applications
  - Parallel programs specifically those using threading can be non-deterministic
Motivation

Research Question

- Interleaving faults occur when there exists threads contentions that produce faulty behaviours

- Reproducing and debugging such systems might be very challenging
  - How to reproduce the interleaving defects
  - How to increase the frequency of interleaving faults occurring?
Motivation

Existing Techniques and Our Approach

- Existing programming solutions
  - Reproduce the faulty interleaving using programming commands (e.g. `yield` and `sleep`) (Eytani et al., 2007)
  - Model checking techniques (Stoller, 2002)
  - Statistical probabilistic techniques (Burckhardt et al., 2010)

- Our approach
  - An alternative view seeking influential environmental parameters that influence the frequency of interleaving faults occurring
Environmental Parameters

Classification of Possible Parameters

- Hardware parameters
- Software algorithms
- Concurrency defect types
- Concurrency levels
Environmental Parameters

Hardware Parameters

- Hardware architecture
- #cores
- Cache and buffer size
- CPU, memory, and bus interrupt speeds

- Examples
  - Threads context switch when clock ticks
  - The time allocated for executing threads reaches its limit (memory speed)
Core management technology
  - Dataflow-based
    - The task assignments are based on data-dependencies
  - Master-slave
    - A single core manages task assignments
  - CoolThreads, Hyperthreads, and virtualizations
Environmental Parameters

Software Parameters

- Scheduling algorithms implemented by VM and OS
  - First-Come First-Served
  - Round Robin
  - Shortest-Job-First
  - Shortest Remaining Time
- Examples
  - Solaris OS – 60 threads priorities
  - Windows XP – 32 threads priorities
  - Linux 2.5 – 140 threads priorities
Environmental Parameters

Concurrency Defects Types

- Interleavings
- Deadlock
- Livelock
- Starvation
- Race condition
- Orphaned thread
Environmental Parameters

Concurrency Levels

- Number of threads
  - Direct relationship with complexity of execution of concurrent programs
- Needs for a model to determine the relationship between number of threads created and number of faults exhibited
Multicore Systems: A Case Study

Goal and Approach

- **Goal**
  - Study the effect of multicore systems on frequency of interleaving faults exhibitions

- **Approach**
  - A number of experiment on various computer systems offering multiple cores and with different threads implementations on a number of programs with known interleaving defects
  - Controlling the cores assigned to an application using Solaris containers
Multicore Systems: A Case Study

Computer Systems Used

- **Sun Fire T1000**
  - UltraSPARC T1 processor 1.2 GHz, 32 GB memory
  - Supporting 32 concurrent hardware threads
  - Suitable for:
    - Tightly coupled multi-threaded applications
    - Computational less expensive threads: serving more threads

- **Sun SPARC Enterprise M3000**
  - SPARC64 VII processor 2.75 GHz, 64 GB memory
  - Supporting eight concurrent hardware threads
  - Suitable for:
    - Single threaded workloads
Multicore Systems: A Case Study

Subject Programs Used

<table>
<thead>
<tr>
<th>Program</th>
<th>NLOC</th>
<th>Fault Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bubble sort</td>
<td>236</td>
<td>Data Race</td>
</tr>
<tr>
<td>airline</td>
<td>61</td>
<td>Data Race</td>
</tr>
<tr>
<td>account</td>
<td>119</td>
<td>Deadlock, Data Race</td>
</tr>
<tr>
<td>deadlock</td>
<td>95</td>
<td>Deadlock</td>
</tr>
<tr>
<td>allocation vector</td>
<td>163</td>
<td>No Lock</td>
</tr>
</tbody>
</table>

Developed and maintained by IBM Haifa
Multicore Systems: A Case Study

Generation of Solaris Containers

- Introduced by Solaris 10

- Resource management for applications using *projects*
  - Workload control
  - Security control by restricting access

- Generation
  1. $k =$ number of CPUs
  2. For $k$ in 1, 2, 4, 6, 8, 16
  3. $\text{create } (\text{pset.max} = k, \text{pset.min}=\text{pset.max})$

- Monitor using *mpstat* command
Multicore Systems: A Case Study

Setup

- For T1000 machine:
  - Created 5 containers (projects)
    - One-CPU, Two-CPU, Four-CPU, Eight-CPU, Sixteen-CPU

- For M3000 machine:
  - Created 3 containers (projects)
    - One-CPU, Two-CPU, Four-CPU

- Commands used:
  - `poolcfg`: To create pools and processor sets
  - `projadd`: To create projects
  - `mpstate`: To monitor the assignment and utilization
Multicore Systems: A Case Study

Setup (con’t)

- Ran each benchmark for 100 times for each pair of:
  - <concurrency level, container>
- Count the number of times the interleaving fault exhibited
- Statistically compared the counted values for their significance
- Only two concurrency levels (little and lot) were considered
# Multicore Systems: A Case Study

## Data Analysis – The Mean Values of Defect Exposures

<table>
<thead>
<tr>
<th>Program</th>
<th>M1000</th>
<th>T1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default</td>
<td>1-CPU</td>
</tr>
<tr>
<td>a) Little Concurrency Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bubble sort</td>
<td>99.1</td>
<td>98.9</td>
</tr>
<tr>
<td>airline</td>
<td>98.7</td>
<td>98.9</td>
</tr>
<tr>
<td>account</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>deadLock</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>all. vector</td>
<td>5.5</td>
<td>3.5</td>
</tr>
<tr>
<td>b) Lot Concurrency Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bubble sort</td>
<td>3.4</td>
<td>3.9</td>
</tr>
<tr>
<td>airline</td>
<td>76.3</td>
<td>74.3</td>
</tr>
<tr>
<td>account</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>deadLock</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>all. vector</td>
<td>99</td>
<td>99.8</td>
</tr>
</tbody>
</table>
Multicore Systems: A Case Study

(a) M3000 - Concurrency Level: Little
(b) M3000 - Concurrency Level: Lot
(c) T1000 - Concurrency Level: Little
(d) T1000 - Concurrency Level: Lot
## Data Analysis

Table 3: The $p$-values of the t-test performed on the number of faults exposed for the load concurrency level on the T1000 computer system. D: Default, 1:1-CPU, 2:2-CPU, 4:4-CPU, 8:8-CPU, 16:16-CPU containers.

<table>
<thead>
<tr>
<th>Containers</th>
<th>D</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1</td>
<td>0.936</td>
<td>0.981</td>
<td>0.93</td>
<td>0.934</td>
<td>0.965</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
<td>0.955</td>
<td>0.993</td>
<td>0.872</td>
<td>0.971</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0.949</td>
<td>0.917</td>
<td>0.983</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0.866</td>
<td>0.965</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>
Discussion

Some Observations

- There is no evidence to believe that there is a dependency between number of cores and interleaving faults
- The number of threads influences the variance of fault exposures
- The concurrency level influences the variance of fault exposures
- The two computer systems had some effects on the frequencies of faults exhibited
  - Recall: two different threading mechanisms and architectures
Conclusion & Research Directions

- Identify environmental factors influencing the frequency of concurrency faults exhibitions
- A case study investigating the effect of multicore environment on concurrency faults
- The research is still in its early stages
Thank You