Testing Concurrency Runtime via a Stochastic Stress Framework

Atilia Gunal
Technical Computing Group
Microsoft
Agenda

• The Product
  ◦ Concurrency Runtime

• The Problem
  ◦ Testing concurrent software

• The Model
  ◦ A stochastic stress framework

• The Results
  ◦ Bugs found
  ◦ Effectiveness of framework
  ◦ Comparison with similar tools
The Product: Concurrency Runtime
Concurrency Runtime

- Runtime for parallel execution
  - Parallel_for(1, 10, Foo )

- Dynamic
  - Composed of interacting components

- Performance centric

- Ocean of race conditions
The Problem: Testing Concurrent Software
**Problem: State Space Explosion**

<table>
<thead>
<tr>
<th>Number of Functionalities</th>
<th>Enumeration of Interactions</th>
<th>Number of Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>F1, F2, F1F2, F2F1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>F1, F2, F3, F1F2, F1F3, ...</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>F1, F2, ..., FN, F1F2, F1F3, ..., F1F2F3, F1F2F4, ..., F1F2F3... FN, F2F1F3... FN, ...</td>
<td>N + N x(N -1) + N x(N -1)x(N -2) + ... N !</td>
</tr>
</tbody>
</table>
Problem: Thread Interactions

- Interaction boundary for serial software
  
  Interaction boundary: A point

- Interaction boundary for concurrent software
  
  Interaction boundary: An interval

It’s even harder to test concurrent software!
The Model: Stochastic Stress Framework
Key Abilities of the Model

- Use random distributions
- Combine multiple tests
- Randomize thread executions
Ability 1: Use Random Distributions

Feature 1   Feature 2   ...   Feature N

F1(int)

Stress Test 1   Stress Test 2   ...   Stress Test N

F1(N1)   F1(N2)

Thread 1   Thread 2
Use Random Distributions

N1, N2 is fixed

N1, N2 comes from a Random distribution

Thread 1  F1  F1  F1  F1  F1  Thread 1
Thread 2  F1  F1  F1  F1  F1  Thread 2

Improves thread interactions!
Ability 2: Combine Multiple Tests

Feature 1  Feature 2  ...  Feature N
F1(int)    F2(int)    F3(int)    F4(int)

↓          ↓          ↓
Stress Test 1  Stress Test 2  ...  Stress Test N

State
F1(N1)  F2(N2)  
Thread 1  Thread 2

State
F3(N1)  F4(N2)  
Thread 1  Thread 2

Shared State
F1(N1)  F2(N2)  F3(N1)  F4(N2)  
Thread 1  Thread 2  Thread 3  Thread 4

Copes with state space explosion!
Ability 3: Randomize Thread Executions

- Reason for thread interruption
  - Thread blocks
  - Thread quantum (Q) expires

This part rarely gets interrupted

OS resumes Thread 1 later on

F1 completes
Randomize Thread Executions

- **Pseudo code of the tool**
  
  Pick a random thread
  Suspend that thread
  Sleep for some random time
  Resume thread

![Probability distribution of Q](image)

Improves thread interactions!
## Summary of Problems and Abilities

<table>
<thead>
<tr>
<th>Problems</th>
<th>Abilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>State space explosion</td>
<td>Combine multiple tests</td>
</tr>
<tr>
<td>Interactions between threads</td>
<td>Use random distributions</td>
</tr>
<tr>
<td></td>
<td>Apply thread randomization</td>
</tr>
</tbody>
</table>
The Results
Efficiency of The Stress Tests

Efficiency = \frac{Output}{Input} = \frac{Bugs found}{Lines of test code}

<table>
<thead>
<tr>
<th>Bug finding activity</th>
<th>Number of Bugs Found</th>
<th>1000 Lines of code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Testing</td>
<td>282</td>
<td>163</td>
</tr>
<tr>
<td>Stress</td>
<td>88</td>
<td>63</td>
</tr>
</tbody>
</table>
Effectiveness of stress tests

Bugs found

- Stress Test
Comparison

- Repetitive Functional Execution
  - Limited Randomization
  - Limited shared runtime state

- Cuzz
  - Smarter thread randomization tool
  - Running stress under Cuzz – no issues found
Conclusion

- Key takeaways to cope with concurrency
  - Combine multiple tests
  - Apply random distributions
  - Randomize thread executions
Questions?