Preemption Sealing for Efficient Concurrency Testing

Thomas Bell  MSR
Sebastian Burckhardt  MSR
Katherine E Coons  UT Austin
Madanlal Musuvathi  MSR
Shaz Qadeer  MSR

to appear, TACAS 2010
Overview

- Concurrency Unit Tests and CHESS
- Multiple Distinct Errors
- Compositional Testing

Preemption Sealing
Concurrent Unit Testing with CHESS

Every run takes a different schedule
Every run is repeatable

Detect
- Assertion violations
- Deadlocks
- Dataraces
- Livelocks
int x=0;

Thread 1
- x++;
- x++;

Thread 2
- x*=2;
- x*=2;

Enumerating thread schedules
State space explosion

- \( O(n^k) \) schedules
- Exponential in both \( n \) and \( k \)
  - Typically: \( n < 10 \quad k > 100 \)
- Limits scalability to large programs

Goal: Scale CHESS to large programs (large \( k \))
Preemption Bounding

- First, explore schedules with small number of preemptive context switches (*preemptions*)

- Preemptions cause bugs

```c
x = 1;
if (p != 0)
{
    x = p->f;
}
p = 0;
```

Thread 1

Thread 2

preemption

non-preemption
Polynomial state space

- Program with
  - $n$ threads, $k$ steps each
  - $c$ preemptions

- $O((n^2k)^c n!)$ schedules

- Exponential in $n$ and $c$, but not in $k$
Preemption Demo

Thread 1
x++;
++;

Thread 2
x*2;
x*2;
Find lots of bugs with 2 preemptions

<table>
<thead>
<tr>
<th>Program</th>
<th>Lines of code</th>
<th>Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Stealing Q</td>
<td>4K</td>
<td>4</td>
</tr>
<tr>
<td>CDS</td>
<td>6K</td>
<td>1</td>
</tr>
<tr>
<td>CCR</td>
<td>9K</td>
<td>3</td>
</tr>
<tr>
<td>ConcRT</td>
<td>16K</td>
<td>4</td>
</tr>
<tr>
<td>Dryad</td>
<td>18K</td>
<td>7</td>
</tr>
<tr>
<td>APE</td>
<td>19K</td>
<td>4</td>
</tr>
<tr>
<td>STM</td>
<td>20K</td>
<td>2</td>
</tr>
<tr>
<td>TPL</td>
<td>24K</td>
<td>9</td>
</tr>
<tr>
<td>PLINQ</td>
<td>24K</td>
<td>1</td>
</tr>
<tr>
<td>Singularity</td>
<td>175K</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37 (total)</td>
</tr>
</tbody>
</table>

Acknowledgement: testers from PCP team
CHESS

• Download/papers
  – http://research.microsoft.com/chess/

• Stateless model checking
  – scales to large programs

• Preemption bounding
  – effectively explores schedule space
Multiple Distinct Errors
Hi Tom, today one of our CCR bvt's failed (and they have not failed in a long time) which means there is some very rare race in either the MultipleItemReceive primitive or something more fundamental.

* Starting Unit test: IteratorWithMultipleItemReceiveManyPorts.Suite: Iterator Suite.Iterations:(100).

The test above was the one that did not terminate. This is something you can throw CHESS at to see if it catches anything.

Thanx

g
Preemption-bounded Search

- Explore all schedules with $c$ preemptions before those with $c+1$ preemptions

- Quantified Coverage guarantee
(Non) Atomic Methods

```java
public class Account {
    volatile int balance;
    public Account(int n) { balance = n; }

    public void Withdraw(int n) {
        int tmp = Read();
        lock (this) { balance = tmp - n; }
    }

    public int Read() {
        return balance;
    }

    public void Deposit(int n) {
        lock (this) {
            var tmp = balance;
            balance = 0;
            balance = tmp + n;
        }
    }
}
```
Preemption Sealing

Don't preempt threads in particular scopes

A scheduling directive

i.e.

Don't preempt thread during execution of method m
Preemption bounded search +
Preemption sealing of methods

\[ \begin{align*}
  & b := 0; \quad M := \emptyset \\
  & \text{loop} \\
  & \quad \text{match \ chess}(T, b, M) \text{ \ with} \\
  & \quad \quad \text{Pass} \Rightarrow b := b + 1 \\
  & \quad \quad \text{Fail}(p_m) \Rightarrow \\
  & \quad \quad \quad \quad \text{print}(p_m); \quad M := M \cup p_m \\
  & \text{forever}
\end{align*} \]
### Preemption Sealing and CCR

<table>
<thead>
<tr>
<th>Sealed Methods</th>
<th>Asserts</th>
<th>Timeouts</th>
<th>Livelocks</th>
<th>Deadlocks</th>
<th>Leaks</th>
<th>Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>3</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>+TryDequeue</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>+WaitForTask</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>+Reg.Recv. +PostInternal</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>43</td>
</tr>
</tbody>
</table>

53 unit tests
Compositional Testing
.NET 4.0 Parallel Extensions

- PLINQ
- Parallel.For
- TaskScheduler
- Task
- BlockingCollection
- ConcurrentBag
- ConcurrentDictionary
- Barrier
- SemaphoreSlim
- ManualResetEventSlim

Diagram showing the relationships between the classes.
Composition / Abstraction
If B's functions are atomic
Then it is safe to seal B
From preemptions
Evaluation

1. Bug yield
2. Efficiency
<table>
<thead>
<tr>
<th>Test</th>
<th>Sealed scope</th>
<th>Result N</th>
<th>S</th>
<th>Executions N</th>
<th>S</th>
<th>Seconds N</th>
<th>S</th>
<th>Execs/sec N</th>
<th>S</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlkCol1</td>
<td>SemSlim</td>
<td>D</td>
<td>D</td>
<td>59167</td>
<td>18733</td>
<td>527.0</td>
<td>206.0</td>
<td>112.3</td>
<td>91.0</td>
<td>2.6</td>
</tr>
<tr>
<td>BlkCol2</td>
<td></td>
<td>P</td>
<td>P</td>
<td>258447</td>
<td>106608</td>
<td>2128.0</td>
<td>1181.0</td>
<td>121.4</td>
<td>90.3</td>
<td>1.8</td>
</tr>
<tr>
<td>BlkCol3</td>
<td></td>
<td>D</td>
<td>D</td>
<td>265</td>
<td>265</td>
<td>1.8</td>
<td>2.2</td>
<td>147.2</td>
<td>120.5</td>
<td>0.8</td>
</tr>
<tr>
<td>BlkColRC1</td>
<td></td>
<td>P</td>
<td>P</td>
<td>1114</td>
<td>364</td>
<td>8.8</td>
<td>4.5</td>
<td>126.6</td>
<td>80.8</td>
<td>2.0</td>
</tr>
<tr>
<td>BlkColRC2</td>
<td></td>
<td>P</td>
<td>P</td>
<td>2406</td>
<td>510</td>
<td>22.8</td>
<td>7.2</td>
<td>105.5</td>
<td>70.8</td>
<td>3.2</td>
</tr>
<tr>
<td>BlkColRC3</td>
<td></td>
<td>P</td>
<td>P</td>
<td>6084</td>
<td>2391</td>
<td>49.0</td>
<td>33.9</td>
<td>124.1</td>
<td>70.5</td>
<td>1.4</td>
</tr>
<tr>
<td>BlkColRC4</td>
<td></td>
<td>P</td>
<td>P</td>
<td>5003</td>
<td>1012</td>
<td>36.9</td>
<td>13.7</td>
<td>135.6</td>
<td>73.9</td>
<td>2.7</td>
</tr>
<tr>
<td>BlkColRC5</td>
<td></td>
<td>D</td>
<td>D</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
<td>0.3</td>
<td>5.0</td>
<td>3.3</td>
<td>0.7</td>
</tr>
<tr>
<td>BarrierRC1</td>
<td>MRES</td>
<td>P</td>
<td>P</td>
<td>776</td>
<td>109</td>
<td>5.5</td>
<td>1.5</td>
<td>141.1</td>
<td>72.7</td>
<td>3.7</td>
</tr>
<tr>
<td>BarrierRC2</td>
<td></td>
<td>P</td>
<td>P</td>
<td>166</td>
<td>92</td>
<td>1.7</td>
<td>1.0</td>
<td>97.6</td>
<td>92.0</td>
<td>1.7</td>
</tr>
<tr>
<td>BarrierRaw</td>
<td></td>
<td>A</td>
<td>A</td>
<td>96</td>
<td>33</td>
<td>0.7</td>
<td>0.7</td>
<td>137.1</td>
<td>47.1</td>
<td>1.0</td>
</tr>
<tr>
<td>CBagRC1</td>
<td>CDict</td>
<td>P</td>
<td>P</td>
<td>559</td>
<td>375</td>
<td>3.7</td>
<td>3.7</td>
<td>151.1</td>
<td>101.4</td>
<td>1.0</td>
</tr>
<tr>
<td>CBagRC2</td>
<td></td>
<td>P</td>
<td>P</td>
<td>559</td>
<td>375</td>
<td>3.8</td>
<td>3.9</td>
<td>147.1</td>
<td>96.2</td>
<td>1.0</td>
</tr>
<tr>
<td>CBagAPC</td>
<td></td>
<td>P</td>
<td>P</td>
<td>6639</td>
<td>4168</td>
<td>46.0</td>
<td>39.0</td>
<td>144.3</td>
<td>106.9</td>
<td>1.2</td>
</tr>
<tr>
<td>CBagACTA</td>
<td></td>
<td>P</td>
<td>P</td>
<td>8230</td>
<td>5727</td>
<td>58.0</td>
<td>57.0</td>
<td>142.9</td>
<td>100.5</td>
<td>1.0</td>
</tr>
<tr>
<td>CBagTTE</td>
<td></td>
<td>P</td>
<td>P</td>
<td>1212</td>
<td>793</td>
<td>7.6</td>
<td>7.5</td>
<td>159.5</td>
<td>105.7</td>
<td>1.0</td>
</tr>
<tr>
<td>CBagTTP</td>
<td></td>
<td>P</td>
<td>P</td>
<td>1529</td>
<td>775</td>
<td>12.9</td>
<td>5.6</td>
<td>118.5</td>
<td>138.4</td>
<td>1.0</td>
</tr>
<tr>
<td>NQueens1</td>
<td>TSchd</td>
<td>L</td>
<td>L</td>
<td>1145</td>
<td>1318</td>
<td>19.4</td>
<td>61.8</td>
<td>59.0</td>
<td>21.3</td>
<td>0.3</td>
</tr>
<tr>
<td>NQueens2</td>
<td></td>
<td>T</td>
<td>T</td>
<td>10146</td>
<td>1027</td>
<td>182.1</td>
<td>55.8</td>
<td>55.7</td>
<td>18.4</td>
<td>3.2</td>
</tr>
<tr>
<td>NQueens3</td>
<td></td>
<td>T</td>
<td>T</td>
<td>9887</td>
<td>1027</td>
<td>181.8</td>
<td>56.4</td>
<td>54.4</td>
<td>18.2</td>
<td>3.2</td>
</tr>
<tr>
<td>PLINQ</td>
<td></td>
<td>P</td>
<td>P</td>
<td>3668</td>
<td>1031</td>
<td>33.1</td>
<td>12.9</td>
<td>110.8</td>
<td>79.9</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Table 2. Evaluation of preemption sealing for compositional testing. Columns labeled 'S' use preemption sealing and columns labeled 'N' do not. **Abbreviations**: BlkCol (BlockingCollection), CBag (ConcurrentBag), SemSlim (SemaphoreSlim), MRES (ManualResetEventSlim), CDict (ConcurrentDictionary), TSchd (TaskScheduler), P (Pass), D (Deadlock), A (Assert), L (Livelock), T (Thread leak).
Related Work

- Atomicity
- Root cause analysis
- Tolerating errors
- Partial order reduction
Test Effectiveness

Don't look for bugs

1. Where you know they are

2. Where you believe they aren't (or don't care if they are)
Preemption Sealing

Don't preempt threads in particular scopes

Effective for
- Multiple distinct errors
- Compositional testing