Exploiting Multicores to Optimize Business Process Execution

Daniele Bonetta, Achille Peternier, Cesare Pautasso, Walter Binder
Faculty of Informatics
University of Lugano - USI
Switzerland
http://sosoa.inf.usi.ch
daniele.bonetta@usi.ch
BP Execution Engine

Focus: Business Process Runtime Execution Environment

Client

Composite

Web Service

Web Service

Web Service
How to scale?
How to scale?
More clients == More BP Instances
How to scale?

More clients \(\leq\) More BP Instances
Multicores

IBM Power7
Outline

1. Multicore Issues

2. JOpera Business Process Execution Engine
   1. Thread Level Parallelism
   2. CPU/Core Level Parallelism

3. Experimental Results

4. Conclusion
Multicore Issues

- Number of cores
- Type of cores (e.g. SMT)
- On Chip Caching Layout (e.g. L2, L3...)
- On Board Memory Layout (e.g. NUMA, NUMA-CC, ...)

Tuesday, December 14, 2010
Multicore Issues

- Cores Num
- Cores Type
- Cache Layout
- Memory Layout

Th Migrations, Ctx Switches
Multicore Issues

- Cores Num
- Cores Type
- Cache Layout
- Memory Layout

- Th Migrations, Ctx Switches
- Data Locality, Contention
BP Execution Engine

Java Business Process Execution Engine
3 Layers Approach

Concurrent Business Process Instances

![jopera](jopera.png)  
*Process Support for Web Services*

OS Threads

Hardware Cores
Abstraction Layers

Concurrent Business Process Instances

OS Threads

Hardware Cores
Engine Architecture

- Request Handler
- Kernel
- Invoker

Request Queue
Execution Queue
Abstraction Layers

Concurrent Business Process Instances

OS Threads

Hardware Cores
Engine Architecture

- Request Handler
  - Request Queue
- Kernel
- Invoker
  - Execution Queue

Tuesday, December 14, 2010
Abstraction Layers

Concurrent Business Process Instances

OS Threads

Hardware Cores
Deployment on Multicores

// threads

---

Tuesday, December 14, 2010
Deployment on Multicores

// threads

How?

core core core
core core core
core core core
core core core

core core core
core core core
core core core
core core core
OverHPC Library

- Jopera Engine (Java)
- OverHPC (JNI, C, Java)
- libpfm
- Linux Kernel
- Multicore Hardware
1. Control and *Change* per-thread scheduling

2. *Measure* low level thread performance data
1) Control and *Change* per-thread scheduling

**Thread-Core Dynamic Affinity Binding**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getThreadPID()</code></td>
</tr>
<tr>
<td><code>getThreadAffinity()</code></td>
</tr>
<tr>
<td><code>setThreadAffinity()</code></td>
</tr>
<tr>
<td><code>getAffinityInfo()</code></td>
</tr>
</tbody>
</table>
2) Measure low level thread performance:

**Hardware Performance Counters**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>getEventsFromCache()</td>
</tr>
<tr>
<td>getEventsFromThread()</td>
</tr>
<tr>
<td>bindEventsToCore()</td>
</tr>
<tr>
<td>bindEventsToThread()</td>
</tr>
</tbody>
</table>
Evaluation

DAG

Parallel

Sequential

Loop
6 cores, 3 cache levels, 1 last level cache

2 x

L3 Cache

L2 L2 L2 L2 L2 L2
L1 L1 L1 L1 L1 L1
L1 C1 C2 C3 C4 C5 C6
Experimental Setup

Concurrent Business Process Instances

Up to 30'000

OS Threads

k

Hardware Cores

12
Thread-level Parallelism

How many *threads*?

Just *increase* the number of parallel concurrent threads in the pools for an increasing number of instances?
Thread-level Parallelism

Just increasing the number of threads...
# Experimental Setup

<table>
<thead>
<tr>
<th>Concurrent Business Process Instances</th>
<th>Up to 30’000</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Opera Logo" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OS Threads</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="OS Threads" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware Cores</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Hardware Cores" /></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Setup

6 cores, 3 cache levels, 1 last level cache

2 Thread pools:

- Kernel
- Invoker
CPU Affinity Binding

Policy 1: Default

Unconstrained scheduling of threads by the OS
CPU Affinity Binding

Policy 2: per CPU

Constrain each thread pool within a CPU
CPU Affinity Binding

Policy 3: per Core

Policy 2 + Constrain each thread on a specific core
CPU Affinity Binding

Policy 4: Interleaved

Mix thread pools across CPUs
## Experimental Setup

<table>
<thead>
<tr>
<th>Concurrent Business Process Instances</th>
<th>Up to 30’000</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="opera.png" alt="opera logo" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OS Threads</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="threads.png" alt="OS Threads icon" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware Cores</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="cores.png" alt="Hardware Cores icon" /></td>
<td></td>
</tr>
</tbody>
</table>
Performance Layers

Concurrent Business Process Instances

Throughput, Walltime, ...

OS Threads

Hardware Performance Counters:
Cache miss, Thread Migrations, Context sw, ...

24

12

5’000 - 30’000

Tuesday, December 14, 2010
Experimental Results

Relative Speedup with 30k instances

2 x AMD Barcelona 6 cores processors with 2 LLC

Tuesday, December 14, 2010
Ineffective sw prefetches
A prefetch request for a memory address already in the cache

L3 cache evictions
Data that needs to be stored in the cache is bigger than free available space
Experimental Results

Ineffective SW prefetches

2 x AMD Barcelona 6 cores processors with 2 LLC
Experimental Results

L3 Cache evictions

2 x AMD Barcelona 6 cores processors with 2 LLC

Tuesday, December 14, 2010
Experimental Results

Relative Speedup with 5k instances

2 x AMD Barcelona 6 cores processors with 2 LLC
Experimental Results

Ineffective SW prefetches

2 x AMD Barcelona 6 cores processors with 2 LLC
Experimental Results

L3 Cache evictions

2 x AMD Barcelona 6 cores processors with 2 LLC
# Experimental Results

## Relative Speedup with 10k instances

<table>
<thead>
<tr>
<th></th>
<th>DAG</th>
<th>Parallel</th>
<th>Sequential</th>
<th>Loop</th>
<th>Geomean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Per CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default Per core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default Interleaved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interleaved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 x AMD Barcelona 6 cores processors with 2 LLC
Experimental Results

Ineffective SW prefetches

2 x AMD Barcelona 6 cores processors with 2 LLC
Experimental Results

L3 Cache evictions

2 x AMD Barcelona 6 cores processors with 2 LLC
Experimental Results

Correlation Coefficients
(Hardware events - JOpera throughput)

<table>
<thead>
<tr>
<th>Workload Size (Number of Instances)</th>
<th>Ineffective SW Pref</th>
<th>L3 Cache Evictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>0.9842</td>
<td>0.9456</td>
</tr>
<tr>
<td>10000</td>
<td>0.9125</td>
<td>0.9883</td>
</tr>
<tr>
<td>30000</td>
<td>0.9661</td>
<td>0.9946</td>
</tr>
</tbody>
</table>
Conclusion

• Multicore machines offer powerful hardware parallelism, but what matters is not just the number of PEs

• The performance depends on how a limited amount of threads are mapped to the HW

• Multicore Aware Thread Scheduling significantly impacts the performance (up to 10% speedup)
Thank you!

OverHPC Library: 
http://sosoa.inf.usi.ch

JOpera business process execution engine: 
http://www.jopera.org

Twitter: 
@jopera_org

me: 
daniele.bonetta@usi.ch