Prometheus: Scalable and Accurate Emulation of Task-Based Applications on Many-Core Systems

GOKCEN KESTOR, ROBERTO GIOIOSA, DANIEL CHAVARRÍA-MIRANDA

2015 Workshop on Modeling & Simulation of Systems and Applications
Exascale parallel programming models

- Exascale systems will achieve high performance through high level of parallelism
  - $O(1k-10k)$ cores per node
  - $O(\text{billion})$ concurrent threads
- Task-based programming models are a promising way to program exascale applications
  - Applications are divided into a myriad of small tasks
  - The system is oversubscribed with tasks ($N_{\text{tasks}} \gg N_{\text{cores}}$)
  - e.g., Cilk++, Intel TBB, Charm++, etc.

There are no tools to model these systems (hardware and software) at the level of scalability required.
Simulators/Emulators landscape

1-8 Cores
Error < 10%
Cycle-accurate simulators

1-100 Cores
Error < 25%
Cycle-approximate simulators

1-1k Cores
Error ?
Full-system emulators

1-1000 Cores
Error ?
Discrete-event emulators

Accurate Emulation
Prometheus
Model-Driven
Trace can be manipulated

Pros: All
Cons: Slow
Cores: FPGAs
App: Simulations
Ex: GE

Pros: Fast
Cons: Limited accuracy
Cores: GPUs
App: Simulations
Ex: Zsim

Pros: Flexible
Cons: Limited accuracy
Cores: CPUs
App: Simulations
Ex: KVM

Pros: Scalable
Cons: Reordering change the app behavior
Cores: 1-thousands
App: OMP, MPI
Ex: BigSim, DIMEMAS

August 14, 2015
Prometheus

Key Observation:
When modeling parallel applications on large systems, performance is largely dominated by runtime and synchronization effects

- Fast enough to emulate future exascale nodes
- Accurate and reliable results
- Modular, swap components in and out
- Model non-determinism of task-based applications
Prometheus Architecture

Cilk++ code

Prometheus Tracer

DAG

Timing information

Hardware contention model drivers

Real system measurements

Cycle-accurate simulator

Architecture topology

Scheduling algorithm

Configuration file

Performance model

\[ p^2(A) = \frac{1}{N} A + 2 \sum \left(1 - \frac{k}{N}\right) \omega_k \]

Representation of application

Representation of architecture

Task scheduler

Prometheus Input

Emulation Engine

Statistics

Emulated time
unsigned long fib(unsigned long n) {
  unsigned long x, y;
  if ( n < 2 )
    return n;
  x = spawn fib(n-1);
  y = spawn fib(n-2);
  sync;
  return x + y;
}
Hardware contention model

- The DAG is extracted from single thread execution
  - Difficult and not necessary to extract a DAG from parallel execution
  - DAGs do not depend on the level of execution parallelism

- Hardware contention in multi-core and multi-threaded processor is modeled by the contention module.
Validation – Experimental setup

Modeling a AMD Interlagos systems:
- 2 processors chips
- 4 modules, 4 cores/module, 2 threads/core
- 32 threads total
- 64 GB DRAM, 4 NUMA domains

Cilk++ task-based applications
- Random work-stealing, work-first scheduler

<table>
<thead>
<tr>
<th>Application</th>
<th>Configuration</th>
<th>Number of tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fib</td>
<td>n = 35</td>
<td>74,651,756</td>
</tr>
<tr>
<td>Heat</td>
<td>nt = 200, nx = 4096*4, ny = 1024</td>
<td>1,234,945</td>
</tr>
<tr>
<td>Integrate</td>
<td>xMax = 5000</td>
<td>193,385,368</td>
</tr>
<tr>
<td>Jacobi</td>
<td>n =1024, steps = 100</td>
<td>139,809,901</td>
</tr>
<tr>
<td>MatrixMul</td>
<td>n = 256</td>
<td>71,902,351</td>
</tr>
<tr>
<td>QuickSort</td>
<td>n = 75000000</td>
<td>152,452,586</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Same core</th>
<th>Local NUMA</th>
<th>Remote NUMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful steal</td>
<td>7555</td>
<td>9135</td>
<td>10635</td>
</tr>
<tr>
<td>Unsuccessful steal</td>
<td>1148</td>
<td>1199</td>
<td>1382</td>
</tr>
</tbody>
</table>

* Results for Intel MIC (244 hardware threads) in the paper.
## Validation – Results

<table>
<thead>
<tr>
<th>N</th>
<th>Fib</th>
<th>Heat</th>
<th>Integ.</th>
<th>Jacobi</th>
<th>MMul</th>
<th>qSort</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>RMin</td>
<td>53.75</td>
<td>29.20</td>
<td>53.86</td>
<td>24.46</td>
<td>104.94</td>
</tr>
<tr>
<td></td>
<td>Emu</td>
<td>54.10</td>
<td>29.03</td>
<td>52.10</td>
<td>24.53</td>
<td>105.50</td>
</tr>
<tr>
<td></td>
<td>RMax</td>
<td>54.88</td>
<td>30.01</td>
<td>54.40</td>
<td>25.21</td>
<td>106.24</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>0.00</td>
<td>-0.58</td>
<td>-3.28</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>16</td>
<td>RMin</td>
<td>26.39</td>
<td>18.46</td>
<td>52.43</td>
<td>12.86</td>
<td>52.28</td>
</tr>
<tr>
<td></td>
<td>Emu</td>
<td>26.52</td>
<td>18.36</td>
<td>50.61</td>
<td>12.93</td>
<td>52.54</td>
</tr>
<tr>
<td></td>
<td>RMax</td>
<td>26.53</td>
<td>19.03</td>
<td>52.88</td>
<td>12.96</td>
<td>52.62</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>0.00</td>
<td>-0.54</td>
<td>-3.46</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>32</td>
<td>RMin</td>
<td>17.93</td>
<td>17.95</td>
<td>37.14</td>
<td>8.19</td>
<td>29.44</td>
</tr>
<tr>
<td></td>
<td>Emu</td>
<td>17.99</td>
<td>18.08</td>
<td>35.87</td>
<td>8.13</td>
<td>29.44</td>
</tr>
<tr>
<td></td>
<td>RMax</td>
<td>18.15</td>
<td>18.89</td>
<td>37.40</td>
<td>8.36</td>
<td>29.54</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>0.00</td>
<td>0.00</td>
<td>-3.43</td>
<td>-0.82</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Generally <4%, max 5.2%

Two exceptions on MIC (6.97 and 10.16%)
Performance

- Minimum slowdown up to 1,024 cores
- 6.4x slowdown at 512k cores
- Emulation completed in 11.5 hours
- Idle cores slow down emulation!
Case study 1: Power-constrained systems

- Emulate the behavior of heterogeneous, power-constrained exascale systems
  - 1,024 total cores, 16 voltage islands, 64 cores/island
  - Varying the number of voltage islands in low-power mode
  - Low-power mode cores run at $\frac{1}{2}$ max frequency

- Automatic task balancing contains performance degradation
Case study 2: Reduced per-core cache

Assume exascale architectures with many-cores on a chip but reduced per-core cache

Use Gem5 SE as hardware contention driver to model multi-core ARM processor

- 16 processor chips
- 4 MB last-level cache per processor chip
- 4 to 64 cores per processor chip

Note: we could not run Cilk++ apps on Gem5 SE
Conclusions

Modeling exascale systems will require new scalable tools

Task-based programming models are non-deterministic

Prometheus: a fast, scalable, modular emulator for task-based applications

Prometheus scales up to 512k cores in 11.5h (6.4x slowdown)

Future work: add network, power, resilience...
Acknowledgements & more information

- Work sponsored by DOE ASCR under the Beyond the Standard Model (BSM) project


- Akhmetova D, G Kestor, R Gioiosa, S Markidis, and E Laure. 2015. "On the application task granularity and the interplay with the scheduling overhead in many-core shared memory systems.” To Appear in IEEE Cluster 2015