BARRIERPONT: SAMPLED SIMULATION OF MULTI-THREADED APPLICATIONS

TREVOR E. CARLSON, WIM HEIRMAN
KENZO VAN CRAEYNEST AND LIEVEN ECKHOUT

MONDAY, MARCH 24TH, 2014
ISPASS 2014 – MONTEREY, CA, USA
**DEMANDS ON SIMULATION ARE INCREASING**

- Simulation targets are evolving
  - Increasing core counts per processor
  - More complex memory hierarchies

- Traditional cycle-level simulation is single-threaded
  - Single-threaded performance is not improving significantly

- Results in a large simulation gap

- New solutions are needed
Many reduction techniques exist today

- Application reduction
  - Smaller input sizes
  - Reduced numbers of iterations

- Sampling: same workload, but
  - Only part of the workload is simulated in detail
  - Whole-program performance is extrapolated

- Examples:
  - SimPoint
  - SMARTS/Flex Points
  - Time-based MT-Sampling
MT-Sampling Wish List

• Multi-Threaded SimPoints-like solution
  – Simulation Time = $O(\# \text{SimPoints})$ instead of $O(\# \text{insns})$
  – Easy to use, fast to run (in parallel)

• Multi-threaded SimPoints is not a valid solution
  – Operates on average CPI, not application runtime
    • Does not allow for runtime (non-idle + idle) reconstruction
  – What is the starting point of a SimPoint region?
    • Must constitute a valid thread ordering for all architectures
CURRENT SAMPLING SOLUTION SPACE

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native Application Simulation</td>
<td>1000 s</td>
</tr>
<tr>
<td>Time-based Multi-Threaded Sampling</td>
<td>100 s</td>
</tr>
<tr>
<td>Workload-specific methodologies</td>
<td>&lt; 1 s</td>
</tr>
</tbody>
</table>

- **Throughput (server) workloads:**
  - client/server, etc.

- **Unstructured synchronization mechanisms:**
  - Work-stealing, mutexes

- **Barrier-synchronizing workloads:**
  - OpenMP, auto-parallelized

- **Thread Synchronization Amount**

- **SMARTS/Flex Points**
- Averages User-IPC

- **BarrierPoint**
- Slowest thread matters
• Key Contributions
  – Micro-architecture independent selection of representative multi-threaded regions
  – Extrapolate and estimate total application runtime
  – Evaluation with realized speedups and errors
  – Propose a straight-forward multi-threaded warmup technique
**BarriernoPoint**

- Application Trends
  - Scientific applications use barriers (OpenMP)
  - Auto-parallelization of applications uses fork-join parallelism

- Main Idea
  - Simulate just the representative regions between barriers (potentially in parallel)
  - We call these barrierpoints
**BarrierPoint Methodology**

- Compare workloads between barriers for similarity
- Select and simulate the representative barrier points
- Reconstruct the runtime from the barrier points’ results
What is an inter-barrier region?

- The execution of all threads after a barrier, up to and including the completion of the following barrier.
MArch-Independent Region Selection

• Basic-block vectors (BBVs)
  – Application execution fingerprint
  – Captures basic-block execution

• LRU-stack distance vectors (LDVs)
  – Application data access fingerprint
  – Counts the number of *unique* address accesses that occur between two accesses to the same address (at cache line granularity)

• BBVs + LDVs
  – Combine instruction and data fingerprint into a single inter-barrier signature
**Unique Address Warmup**

- Multi-threaded warmup technique for Barrierpoint
  - Avoid long execution-driven simulation before ROI
  - Warmup data part of checkpoint, relatively μarchitecture independent
  - Ensure cache coherency

- Unique Address Warmup
  - Similar to MTR\(^1\), but avoids cache-specific reconstruction
  - Collect, from program start up to barrierpoint start
    - Each core records the most recent read, write and instruction cache accesses (by cache line)
    - We collect \((M \times (\text{last } N \text{ cache lines})),\) where \(N\) is the number required to fill up the entire cache hierarchy, and \(M\) is the number of threads
  - Replay:
    - Issue per-core list of unique addresses in parallel
    - Feed into real cache models, which remain coherent during warmup

---

\(^1\) K. C. Barr, et al., “Accelerating Multiprocessor Simulation with a Memory Timestamp Record,” in ISPASS 2005
RECONSTRUCTING PROGRAM METRICS

• Each barrierpoint is given a weight
  – The number of times that it occurs in the run

• With the list and weights, we can reconstruct the runtime
  – $\text{Runtime}_{cg/A/8} = \text{Runtime}_{cg/A/8} (bp_0) \times 1.0$
    + $\text{Runtime}_{cg/A/8} (bp_{15}) \times 12.0$
    + $\text{Runtime}_{cg/A/8} (bp_{21}) \times 2.0 + ...$

• Similar to SimPoint reconstruction, but now with time (including idle/sync.) rather than CPI

• Also works for other application metrics: MPKI, etc.

<table>
<thead>
<tr>
<th>BM/input</th>
<th>cores</th>
<th>barriers</th>
<th>barrierpoints</th>
<th>barrierpoint # and multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>npb-cg/A</td>
<td>8</td>
<td>46</td>
<td>5</td>
<td>0 (1.0), 15 (12.0), 21 (2.0), ...</td>
</tr>
<tr>
<td>npb-mg/A</td>
<td>8</td>
<td>245</td>
<td>8</td>
<td>2 (2.0), 52 (4.6), 57 (9.0), ...</td>
</tr>
</tbody>
</table>
ARCH-INDEPENDENT REGION SELECTION

NPB, A input, 32-cores; Aggregate IPCs shown
**Experimental Setup**

- We model a Xeon/Nehalem-like machine
  - 8-core and 32-core architecture
  - 8-cores share an LLC
- Sniper Multi-Core Simulator
- Benchmarks
  - Most NAS Parallel Benchmarks (NPB)
    - A inputs
  - Parsec
    - Bodytrack Large
- Implemented for OpenMP applications
  - Fork/join parallelism, one barrier per #omp parallel for
  - Can be extended to other types of global synchronization, e.g.
    - pthread_barrier()
    - MPI_(All)Reduce(MPI_COMM_WORLD)
RESULTS

• BarrierPoint shows accurate absolute results

Application runtime

Average error of 0.9% with a maximum of 2.9% error
RESULTS

- BarrierPoint shows accurate absolute results and relative scaling results
• Realized simulation speedups are good
  – Resource utilization reduction (improved throughput) by 78x
  – Speedup of 25x on average, 867x maximum
Additional Results

• Barrierpoints are a common unit of work across architecture configurations
  – 8-core vs. 32-core
  – Allows for a single characterization run

• Fingerprinting across both instruction and data profiles provide the best results
  – Equal combination of BBVs and LDVs
Key Contributions

- Micro-architecture independent selection of representative multi-threaded regions
  - Explore alternatives to BBVs, such as LRU-stack distances
  - Extrapolate and estimate total application runtime
- Evaluation
  - Average reduction of machine resources of 78x
  - Realized an average speedup of 25x and maximum of 867x
  - Average error of 0.9%, maximum of 2.9%
- Propose a straight-forward multi-threaded warmup technique
- Technology Preview to be released soon
  - http://snipersim.org
BARRIERPOINT: SAMPLED SIMULATION OF MULTI-THREADED APPLICATIONS

TREVOR E. CARLSON, WIM HEIRMAN
KENZO VAN CRAEYNEST AND LIEVEN ECKHOUT

MONDAY, MARCH 24TH, 2014
ISPASS 2014 – MONTEREY, CA, USA