A Trace-driven Approach for Fast and Accurate Simulation of Manycore Architectures

Anastasiia Butko, Luciano Ost, Abdoulaye Gamatié, Vianney Lapôtre, Rafael Garibotti and Gilles Sassatelli
LIRMM (CNRS and University of Montpellier), Montpellier, France

Chris Adeniyi-Jones, ARM Ltd., Cambridge, United Kingdom
Introduction

Design a new type of computer architecture capable of setting future global High Performance Computing (HPC) standards that will deliver Exascale performance while using 15 to 30 times less energy.

An efficient manycore system exploration framework is required:

- Architectural parameters exploration
- Architectural scaling (increasing core count)
## Simulation Frameworks

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Accuracy</th>
<th>Supported processor architectures</th>
<th>License</th>
<th>Development/Support activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simics</td>
<td>Functionally-accurate</td>
<td>Alpha, ARM, MIPS, PowerPC, SPARC and x86</td>
<td>Private</td>
<td>Yes</td>
</tr>
<tr>
<td>PTLsim</td>
<td>Cycle-accurate</td>
<td>x86</td>
<td>Open</td>
<td>Yes</td>
</tr>
<tr>
<td>SimpleScalar</td>
<td>Cycle-accurate</td>
<td>Alpha, ARM, PowerPC and x86</td>
<td>Open</td>
<td>No</td>
</tr>
<tr>
<td>OVPsim</td>
<td>Instruction-accurate</td>
<td>Open Cores Open RISC, ARM, Synopsys ARC, MIPS, PowerPC and others</td>
<td>Open and Private</td>
<td>Yes</td>
</tr>
<tr>
<td>Gem5</td>
<td>Quasi-cycle-accurate</td>
<td>Alpha, ARM, x86, SPARC, PowerPC and MIPS</td>
<td>Open</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Gem5 Simulation Framework

- **Flexibility**
  - Multiple ISAs and CPU models
  - Memory systems
  - Debugging capabilities

- **Accuracy**
  - Reported between 2% and 18% *

- **Speed**
  - Between 1 KIPS and 1MIPS

* Butko A. et al., *Accuracy evaluation of Gem5 simulator system*
Efficient system simulation techniques

Motivation:

- For regular applications:
  - CPU Computation phase is deterministic
  - Memory Communication phase is traffic dependent

Trace-Driven simulation principle:

- Collect execution traces in a reference simulation
- Replay those traces on the trace-driven simulation in which only interconnect and memory system are simulated
Outline

• Introduction
• Trace-driven approach
  • General concept
  • Integration in Gem5
• Evaluation
• Conclusion and Future Work
Trace-Driven Approach: General Concept

(a) Trace Collection

- Reference Memory System
- Processing Element (PE)
Trace-Driven Approach: General Concept

(a) Trace Collection

- PE #1
- ... PE #N
- Trace Collection Interface
- I$ D$ I$ D$
- Communication Infrastructure
- Memory Infrastructure

(b) Trace Processing

- Trace Files #1 .. #M
- Replication
- Trace Files #1 .. #N

Reference Memory System
Processing Element (PE)
Trace-Driven Approach: General Concept

(a) Trace Collection
- PE #1
- ... PE #N
- Trace Collection Interface
- IS$ D$ IS$ D$
- Communication Infrastructure
- Memory Infrastructure

(b) Trace Processing
- Trace Files #1 .. #M
- Replication
- Trace Files #1 .. #N

(c) Trace Simulation
- Trace File #1
- ... Trace File #M
- TI #1
- ... TI #M
- Communication Infrastructure
- Memory Infrastructure
- vIS$ vD$
- Simulated Memory System
- Trace Injector (TI)
- Reference Memory System
- Processing Element (PE)
Case Studies
General Concept: Simulation Gain
General Concept
General Concept

Traffic dependent phase
General Concept

Event-Driven

Memory Request $T_1$ Memory Response $T_2$

Memory Interconnect IS/DS Core

computation communication computation

Time
General Concept

Event-Driven

Memory Request \[ T_1 \] Memory Response \[ T_2 \]

Memory Interconnect IS/DS Core

computation communication computation

Time

Trace-Driven

Memory Interconnect vIS/VD$ TI

computation communication

Time
Integration in gem5

Time tick | Event source | Type | Flags | Address | Size | Value
--- | --- | --- | --- | --- | --- | ---
2902097628000: system.cpu0.icache: cmd: ReadReq f: 1 l: a: fff018 s: 4 v: 4043833472
2902097674000: system.cpu0.icache: resp

Memory Interconnect vL$/$vD$

T1

shifting delay

$ t_1 \quad t_2 \quad t_2' \quad t_3 \quad t_3'$

Time
Synchronization

Trace Collection

Macros are inserted into the application source code

```c
#define PTHREAD_CREATE(...) pthread_create(&threads[i], 0, func, (void *) &params);
    printf("Thread id %i create tick: %lu\n", ID, rpns());

#define PTHREAD_BARRIER(...) pthread_barrier_wait(barrier);
    printf("Thread id %i barrier tick: %lu\n", ID, rpns());

#define PTHREAD_JOIN(...) pthread_join(threads[i], 0);
    printf("Thread id %i join tick: %lu\n", ID, rpns());
```

Trace file

```
Thread id 0 create tick: 1627449500
Thread id 1 create tick: 1627690378

Thread id 1 barrier tick: 1628594414
Thread id 0 barrier tick: 1628632384

Thread id 0 join tick: 1727194499
Thread id 1 join tick: 1727322559
```
Synchronization

Trace File 0

Thread id 0 create tick: 2859888754000
2859888764000: cpu0.dcache: WriteReq f: 0 0 a: f89e240 s: 4 v:0
2859888795000: cpu0.dcache: resp
2859888831000: cpu0.dcache: WriteReq f: 0 0 a: f89e280 s: 4 v:0
2859888862000: cpu0.dcache: resp
Thread id 0 barrier tick: 2859888890000
2859888924000: cpu0.dcache: WriteReq f: 0 0 a: f89e2c0 s: 4 v:0
2859888955000: cpu0.dcache: resp
28598889904000: cpu0.dcache: ReadReq f: 0 0 a: f89e240 s: 4 v:0
2859888935000: cpu0.dcache: resp
Thread id 0 join tick: 28598889120000

Trace File 1

Thread id 1 create tick: 2859888754000
2859888774000: cpu1.dcache: WriteReq f: 0 0 a: f89e240 s: 4 v:0
2859888805000: cpu1.dcache: resp
2859888841000: cpu1.dcache: WriteReq f: 0 0 a: f89e280 s: 4 v:0
2859888872000: cpu1.dcache: resp
Thread id 1 barrier tick: 2859888900000
2859888934000: cpu1.dcache: WriteReq f: 0 0 a: f89e2c0 s: 4 v:0
2859888965000: cpu1.dcache: resp
2859888941000: cpu1.dcache: ReadReq f: 0 0 a: f89e240 s: 4 v:0
28598889045000: cpu1.dcache: resp
Thread id 1 join tick: 28598889100000
Integration in gem5

- Trace Replication
  - timing behavior, synchronization, **address distribution**
Speedup and Accuracy

- Scientific application benchmark suite

- Error percentage on execution time varies from 0.02% to 6%
- Speedup ranges from 6x to 800x
Exploration of Architectural Parameters

• Memory latency variation

![Bar chart showing execution time variation with memory latency variation. The chart compares Full System and Trace Driven approaches. Example values include 2.78% for 5ns, 4.51% for 15ns, 5.79% for 30ns, 5.17% for 45ns, and 6.07% for 55ns.]

• L2 cache

![Bar chart showing execution time variation with L2 cache size. The chart compares Full System, ET, ET + init, and ET + boot approaches. Example values include 0.04% for reference, 14.86% for 256kB, 8.09% for 1MB, 7.51% for 1MB (ET + init), 1.17% for 1MB (ET + boot), 18.49% for 8MB, 13.89% for 8MB (ET + init), 10.48% for 8MB (ET + boot), and 15.18% for 16MB, 8.38% for 16MB (ET + init), 7.39% for 16MB (ET + boot).]
Consistency of Simulations in Presence of Dependencies

- Similarity between cache miss behaviors
- Negligible fluctuations are compensated
Trace Replication

- Correlation of traces from 1-, 2-, 4-, 8-cores RP collection
Simulation Cost

- Simulation profiling on the work station

Intel Core i5 CPU @ 2.60 GHz x 4
Conclusion and Future Work

• **Results (for regular not data-dependent workloads)**
  • Simulation speedup of up to 800x
  • Error percentage varies from 0.02% to 6%
  • Implementation is freely available online
    http://www.lirmm.fr/ADAC

• **Future Work**
  • optimization of trace file
  • out-of-order processor support
  • different memory mapping algorithms
Thank you for your attention! Questions?
Efficient system simulation techniques

- Trace-driven simulations:
  - TaskSim: exploring multithreaded application behaviors (BSC, Spain)
  - ManySim: memory models evaluation for 32-cores architecture (Systems Technology Lab, Intel)
  - Netrace: network exploration only (The University of Texas, NVIDIA)
Integration in gem5

1. Trace Collection/Processing Phases
   • Memory access traces

```c
DPRINTF(CacheTraceMiss, "cmd: %s f: %d %d a: %x s: %d v: %d \n",
        pkt->cmdString(), // request type
        pkt->req->isInstFetch(), // is instruction fetch
        pkt->isRead(), // is read
        pkt->getAddr(), // destination address
        pkt->getSize(), // packet size
        data); // packet value

DPRINTF(CacheTraceMiss, "resp\n");
```

<table>
<thead>
<tr>
<th>Time tick</th>
<th>Event source</th>
<th>Type</th>
<th>Flags</th>
<th>Address</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2902097628000: system.cpu0.icache: cmd: ReadReq</td>
<td>11</td>
<td>a: ffff018</td>
<td>s: 4</td>
<td>v: 4043833472</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2902097674000: system.cpu0.icache: resp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Traces Collection
Evaluation

- Reference platform (RP)
  - Gem5 Full System experimental setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># of cores</td>
<td>from 1 to 8</td>
</tr>
<tr>
<td>CPU frequency</td>
<td>500 MHz</td>
</tr>
<tr>
<td>L1 cache size</td>
<td>4 kB</td>
</tr>
<tr>
<td>Channel width</td>
<td>64 bits</td>
</tr>
<tr>
<td>DDR latency</td>
<td>30 ns</td>
</tr>
<tr>
<td>OS</td>
<td>Linux Kernel 2.6.38</td>
</tr>
<tr>
<td>Programming</td>
<td>POSIX Threads</td>
</tr>
</tbody>
</table>