Enhancing an HTM System with Monitoring, Visualization and Analysis Capabilities

Philipp Kirchhofer$^1$, Martin Schindewolf$^1$, Nehir Sonmez$^2$, Oriol Arcas$^2$, Osman S. Unsal$^2$, Adrián Cristal$^2$, Wolfgang Karl$^1$

$^1$Karlsruhe Institute of Technology (KIT) $^2$Barcelona Supercomputing Center (BSC)
Motivation

Transactional Memory simplifies parallel programming

But how to achieve good performance and scalability?

Need for a monitoring infrastructure to

- identify runtime characteristics of an application running on an HTM system
- detect application bottlenecks
- get insight into interaction between application and HTM system
Project Goals

Develop a monitoring infrastructure for the TMbox HTM system using event logs:

- Generate events at run time and save for later processing
- Recreate HTM state offline
- Visualize and analyse saved information

Additional goals:
No probe effect, small hardware overhead, high extensibility, ease of use
Design
The TMbox system

- Developed at Barcelona Supercomputing Center
- MIPS compatible multi-core system (FPGA based, 16 cores on 1 FPGA)

- Supports
  - STM (TinySTM)
  - HTM (BeeTM)
  - HybridTM (Modified TinySTM)
Design
The TMbox system - Block Diagram

-- Bus 1: Memory Requests / Responses
-- Bus 2: Invalidation / Events
Design
Event Generation, Log Unit, Bus Controller
Design
Event Generation, Log Unit, Bus Controller

Added components (in every core):
- Event Generation: Monitor HTM state
- Log Unit: Timestamp event transfer via bus 2

Diagram with components:
- Host PC
- PCI-E Channel
- Bus Controller
- DDR Controller
- RAM
- Core 6
- Core 7
- Core 0
- Core 1
- Core 5
- Core 4
- Core 3
- Core 2
Design
Event Generation, Log Unit, Bus Controller

Added components (in every core):
- Event Generation: Monitor HTM state
- Log Unit: Timestamp event transfer via bus 2

Transfer events via PCI-E

Host PC

PCI-E Channel

Core 6
Core 7
Bus Controller
DDR Controller
RAM
Core 0
Core 1
Core 5
Core 4
Core 3
Core 2
Design
Event Generation, Log Unit, Bus Controller

Added components (in every core):
- **Event Generation**: Monitor HTM state
- **Log Unit**: Timestamp event transfer via bus 2

- **Save events to file**
- **Transfer events via PCI-E**

- Host PC
- PCI-E Channel

- Core 6 → Core 7
- Core 7 → Bus Controller
- Bus Controller → DDR Controller
- DDR Controller → RAM
- RAM → Core 0
- Core 0 → Core 1
- Core 1 → Core 2
- Core 2 → Core 3
- Core 3 → Core 4
- Core 4 → Core 5
- Core 5

10.04.2012 Kirchhofer et. al. - Enhancing an HTM System with Monitoring, Visualization and Analysis Capabilities
Design
Post Processing, Visualization, Analysis

Host PC
Post Processing | Visualization | Analysis

Hard drive
Design

Post Processing, Visualization, Analysis

- Rebuild HTM state
- Host PC
  - Post Processing
  - Visualization
  - Analysis
- Event stream
- Hard drive
Design
Post Processing, Visualization, Analysis

- Rebuild HTM state
- Use Paraver tool

Host PC
- Post Processing
- Visualization
- Analysis

Event stream

Hard drive
Results

Paraver workflow
Results

Example: 4 conflicting threads

- Thread 1: 80% Compute, 15% Compute Wasted, 5% Idle
- Thread 2: 75% Compute, 20% Compute Wasted, 5% Try Lock
- Thread 3: 60% Compute, 30% Compute Wasted, 10% Try Lock
- Thread 4: 50% Compute, 40% Compute Wasted, 10% Try Lock

Legend:
- Idle
- Compute
- Compute Wasted
- Try Lock
- Commit
- Abort

Graph shows the time distribution across threads and activities.
Results

Example: 4 conflicting threads

<table>
<thead>
<tr>
<th>Thread</th>
<th>Idle</th>
<th>Compute</th>
<th>Compute Wasted</th>
<th>Try Lock</th>
<th>Commit</th>
<th>Abort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Invalidation
Summary

✓ Application / HTM runtime behavior is trackable with no probe effects

✓ Visualization capabilities lead to in-depth understanding of application / HTM runtime behavior

- Metrics provided via offline analysis, e.g.
  - Time spent in Committed & Aborted Transactions
  - Contention / Commit & Abort Rate
  - Contention between specific threads
  - HTM System Overhead
Summary at a glance

The TMbox system now supports:

✓ Identification of detailed runtime characteristics of an application

✓ Easy detection of application bottlenecks

✓ Getting hints to optimize application concerning both performance and scalability
Ongoing work

Combine monitoring with STM runtime environment: Allows analysis of HybridTM systems

See further work in full paper “A low-overhead profiling and visualization framework for Hybrid Transactional Memory” (to appear in FCCM 2012)

Acknowledgment

This work was supported by a Short Term Scientific Mission (STSM) grant from the European Cooperation in Science and Technology (COST) Action IC1001 (EuroTM) during June and July 2011.
Enhancing an HTM System with Monitoring, Visualization and Analysis Capabilities

Philipp Kirchhofer¹, Martin Schindewolf¹, Nehir Sonmez², Oriol Arcas², Osman S. Unsal², Adrián Cristal², Wolfgang Karl¹

¹Karlsruhe Institute of Technology (KIT) ²Barcelona Supercomputing Center (BSC)
Additional slides
Workflow

Synthesize system and run on FPGA

Simulate system with Xilinx ISIM/Mentor Graphics ModelSim

Event Stream

Post-Processing Tool BusEventConverter

Paraver File

Visualize events, states and conflicts with Paraver

Legend
- Modified component
- New component
- File on hard drive
## Event Diagram

### Secondary Ring Bus: Event Format

<table>
<thead>
<tr>
<th>Message Header</th>
<th>Message Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 Bit 2 Bit 4 Bit</td>
<td>28 Bit 20 Bit 4 Bit 4 Bit</td>
</tr>
</tbody>
</table>

- **Message Type 3**
- **Sender ID**
- **Timestamp (delta-encoded)**
- **Event Type**
- **Event Data**

Event diagram
FPGA Usage

TMbox FPGA Usage (with monitoring infrastructure) - Increase per Core