AMD'S ADVANCED SYNCHRONIZATION FACILITY

Stephan Diestelhorst
Advanced Micro Devices, Inc.

Feb 08, 2011
Public
**BASICS**

*Experimental* HTM support; public specification at [http://developer.amd.com/CPU/ASF/Pages/default.aspx](http://developer.amd.com/CPU/ASF/Pages/default.aspx)

Would ensure forward progress under certain conditions

- Lower bound on worst-case capacity
- No recurring abort in absence of contention

Instructions:

- `SPECULATE`, `COMMIT`
- `LOCK MOV`
- `WATCHR, WATCHW`
- `RELEASE`
- `ABORT`
INTRODUCTION
**ASF 2.0 – OVERVIEW**

- Atomic read-modify-write construct
  - Flexible
  - Multiple locations
  - Normal instruction set

- General Idea
  - Speculative execution
  - Monitor for inconsistent probes
  - Undo modifications & redirect control flow
**ASF 2.0 – PRIMITIVES**

- **SPECULATE**
  - Start critical section
  - Anchor for later roll-back due to inconsistent probe
    - Keep: rIP, rSP
    - Don’t keep: GPR, XMM, MMX, FPU,…
  - Roll-back
    - Jump behind SPECULATE
    - Non-zero “return” value

- **COMMIT**
  - Stop monitoring
  - Commit updates
ASF – PRIMITIVES

- LOCK load / store
  - Prefixed loads (mov…, prefetch, prefetchw)
    - Monitor these cache lines for probes
    - “Declarators”
  - Prefixed stores (mov…)
    - Modify monitored (aka. protected) lines
    - Stores subject to undo / roll-back
ASF – NESTING

- ASF supports flat nesting
  - Outer and inner lines must fit into capacity
  - Roll-back to the outermost SPECULATE
- Maintain internal nest count
  - SPECULATE increments
  - COMMIT decrements
- Inner SPECULATE & COMMIT become no-ops (except for inc / dec)

![Diagram showing ASF nesting with SPECULATE and COMMIT blocks.](image-url)
CAPACITY

- Minimum **worst-case** capacity of **four** cache lines
  - Dependent on address layout, actual capacity often much higher
  - Useful for software?
  - Providing anything > 0 is non-trivial in HW

- Plan B (STM, global lock) for generic use case

- What does it mean?
  - In absence of spurious conditions, speculative region will eventually succeed
    - Success on first try the vast majority of time
Usage
LOCK-FREE PROGRAMMING — DCAS

- DCAS:
  
  SPECULATE
  JNZ fail
  MOV RCX, 1
  LOCK MOV RAX, [mem1]
  LOCK MOV RBX, [mem2]
  CMP R8, RAX
  JNZ out
  CMP R9, RBX
  JNZ out
  LOCK MOV [mem1], RDI
  LOCK MOV [mem2], RSI
  XOR RCX, RCX
  
  out:
  COMMIT
  ...  
  fail: ...

; Double Compare and Swap
; Critical section begins
; Signal abort as failure
; Specification begins
; Update protected memory
; End of critical section
R&D: SOFTWARE TRANSACTIONAL MEMORY (STM)

- EU project VELOX: Transactional Memory
  - Whole-system approach
    - Applications, run times, STM systems, operating systems, CPU architecture
  - Partners: Barcelona Supercomputing Center, Chalmers University, École Polytechnique Fédérale de Lausanne, Tel Aviv University, University of Neuchâtel

- OSRC activities:
  - Evaluate and scrutinize hardware proposals
  - Simulation / evaluation of STM accelerators

- Side activities:
  - Deriving requirements / proposing architecture extensions for AMD
  - Further ASF evaluation
ASF IN THE VELOX STACK

C/C++ PLATFORM

JAVA PLATFORM

Applications
QuakeTM (C)
Globulation2 (C++)

Benchmarks
RMSTM, TUnit (C)
STMbench7 (C++)

gcc-tm
DTMC

TM Compiler

TM Compiler

Applications
TMBeans
Benchmarks
STMbench7

.java
.java
.java

.java

API

API

.class
.class
.class

Transactifier

STM Runtime

JVM/JDK
Virtual Machine

ABI

ABI

Deuce-agt
Deuce-rt

TMdietlibc
Gibraltar

TinySTM

(TinySTM)

ASFTM

TMsched

Scheduler

Simulator

Operating System

Hardware

System Libraries

Libraries

.exe

.exe

.exe

STM

HyTM-SW

HyTM-HW

HTM

HW

WP 2

WP 5/4/3

SYSTEM SW

APPS

WP 6
PROGRAMMER'S VIEW

Language integration:

atomic {
  a = b;
  c = 1;
}

Library interface:

stm_start();

x = stm_load(&b);
stm_store(&a, x);
stm_store(&c, 1);
stm_commit();
SPECULATIVE LOCK ELISION

- Hash table example
  - ASF for insert
  - Lock for expensive resize

Insert:
```assembly
SPECULATE
JNZ <Insert>
LOCK MOV RAX, [table_lock]
CMP RAX, 0
JE <ActualInsert>
ABORT

ActualInsert:

; insert an element
COMMIT
...
```

Resize:
```
LOCK BTS [table_lock], 0
JC <Out>

; resize the table
MOV [table_lock], 0
```

Out:
SPECULATIVE LOCK ELISION (CONTD.)

- Critical sections → transactions
- Looking at
  - Memcached
  - Python
- Performance improvement
- Enable parallelism
EVALUATION PLATFORM AND RESULTS
 Publications with more detail


- **WTTM'10**: Sane Semantics of Best-effort Hardware Transactional Memory - Stephan Diestelhorst, Michael Hohmuth, Martin Pohlack

- **EuroSys'10**: Evaluation of AMD's Advanced Synchronization Facility within a Complete Transactional Memory Stack - Dave Christie, Jae-Woong Chung, Stephan Diestelhorst, Michael Hohmuth, Martin Pohlack, Christof Fetzer, Martin Nowack, Torvald Riegel, Pascal Felber, Patrick Marlier, Etienne Riviere

- **Transact'10**: Compilation of Thoughts about AMD Advanced Synchronization Facility and First-Generation Hardware Transactional Memory Support - Jaewoong Chung, David Christie, Martin Pohlack, Stephan Diestelhorst, Michael Hohmuth, Luke Yen

- **Transact'10**: Implementing AMD's Advanced Synchronization Facility in an Out-of-order x86 Core - Stephan Diestelhorst, Martin Pohlack, Michael Hohmuth, Dave Christie, Jae-Woong Chung, Luke Yen
IMPLEMENTATION IN PTLSIM

Motivation

- Accurately reflects real systems
- OoO system, gain experience for real implementation
- Previous experience with PTLsim / Xen
- Good open-source solution, allows collaboration with external groups
- PTLsim has fast turn-around (native execution mode based on Xen hypervisor)
IMPLEMENTATION IN PTLSIM (CONT.)

- Design and implementation variants
  - Locked-line buffer (LLB)
    - Fully associative
    - With different sizes (8, 256 lines)
    - Stores old data
    - Next to L1 cache, but not integrated
  - Cache-based read-set
  - Support for cache-based write set underway

Influence of ASF capacity on throughput for different ASF variants (red-black tree and linked list with 20% update rate with 8 threads). (from EuroSys paper)
Scalability of applications with four ASF implementations and varying thread count (execution time; lower is better). The arrows indicate STM values that did not fit into the diagram. The horizontal bars show the execution time for execution of sequential code (without a TM).
Scalability of applications with four ASF implementations and varying thread count (execution time; lower is better). The arrows indicate STM values that did not fit into the diagram. The horizontal bars show the execution time for execution of sequential code (without a TM).
CONCLUSION

 ASF accelerates / simplifies multi-threaded programs

 Full stack available
  – Simulator
  – Accelerated STM
  – Compiler

 Try it out
  – http://www.amd64.org
  – http://tm.inf.tu-dresden.de/
  – ASF_Feedback@amd.com
This research is conducted within the FP7 Integrated Approach to Transactional Memory on Multi-and Many-core Computers (VELOX) project (ICT-216852) supported by the European Commission.

DISCLAIMER
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION
© 2011 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Microsoft Windows and Vista are registered trademarks, of Microsoft Corporation in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.