# The Adaptive Transactional Memory Test Platform: A Tool for Experimenting with Transactional Code for Rock

Dan Nussbaum, Sun Microsystems Labs

Joint work with Mark Moir and Kevin Moore

# The Adaptive Transactional Memory Test Platform (ATMTP)

- Simulator for Rock HTM feature.
- GPLv2 license.
- Release scheduled for **today**.

# ATMTP

- Provides a first-order approximation of the success and failure characteristics of transactions on Rock processor.
  - *Best-effort* HTM.
- *Not* an accurate model of Rock's implementation or performance.
- Based on Simics/GEMS/Ruby/LogTM[-SE]

Two Target Audiences:

- Programmers who want to write code for Rock, and who want to know how useful Rock's HTM feature is to them.
- Computer Architecture researchers, who are interested in looking at how useful best-effort HTM is, and at the cost imposed by various best-effort features.

# Outline

- Introduction.
- History and Motivation (GEMS 1.4).
- Current Functionality (GEMS 2.1).
- Tidbits and War Stories.
- Conclusion.

# History and Motivation (GEMS 1.4)

- Based on LogTM.
  - SMP-like machine.
  - "EE" (Wisconsin Taxonomy):
    - Eager conflict detection.
    - Eager (undo log) version management – roll back on abort.
- Modifications to LogTM:
  - "Generic HTM" API.
    - (We've been talking about our work since long before Rock's TM support was announced – target machine had to be “vanilla”.)
  - Generic Best-effort features.

# GEMS 1.4: “Generic HTM”

- Instruction handlers:
  - `chkpt <fail-addr>`
  - `commit`
  - `fail` ('`ta %xcc, %g0+15`')
- Disabled LogTM's contention management – instead do it in software, at `<fail-addr>`.
- *Neutering* (aborts caused by resource limitations):
  - Store buffer size.
  - Cache overflow.
- Modified protocol:
  - *Requestor wins:* upon conflict, “local” transaction aborts.

# GEMS 1.4: *Mortal Wound* Strategy

- Used this strategy to realize best-effort aborts.
- Difficult to abort out of the middle of an instruction; instead, notice intent to abort on the next transactional memory access (*).

  (*) Actually, next transactional ***read,*** because of *store-abort* bug, the fixing of which was one of the first things Kevin did after joining us.

- Recall that this is *history* – in fact, GEMS 2.0 now has a facility that provides similar functionality for us, and does a better job of it than we did.

# GEMS 1.4: Add Other *Rock-like* Qualities

- Rock TM announcement made it possible for us to start talking about more Rock-like behavior.
- Feedback from failed transactions:
  - `rd cps, <rd>`
- Other best-effort behavior:
  - Abort on “difficult” instructions.
  - Abort on synchronous traps.
    - (No longer needed handler for `fail` instruction.)
- TBD:
  - Asynchronous traps (interrupts).

# Outline

- Introduction.
- History and Motivation (GEMS 1.4).
- Current Functionality (GEMS 2.1).
- Tidbits and War Stories.
- Conclusion.

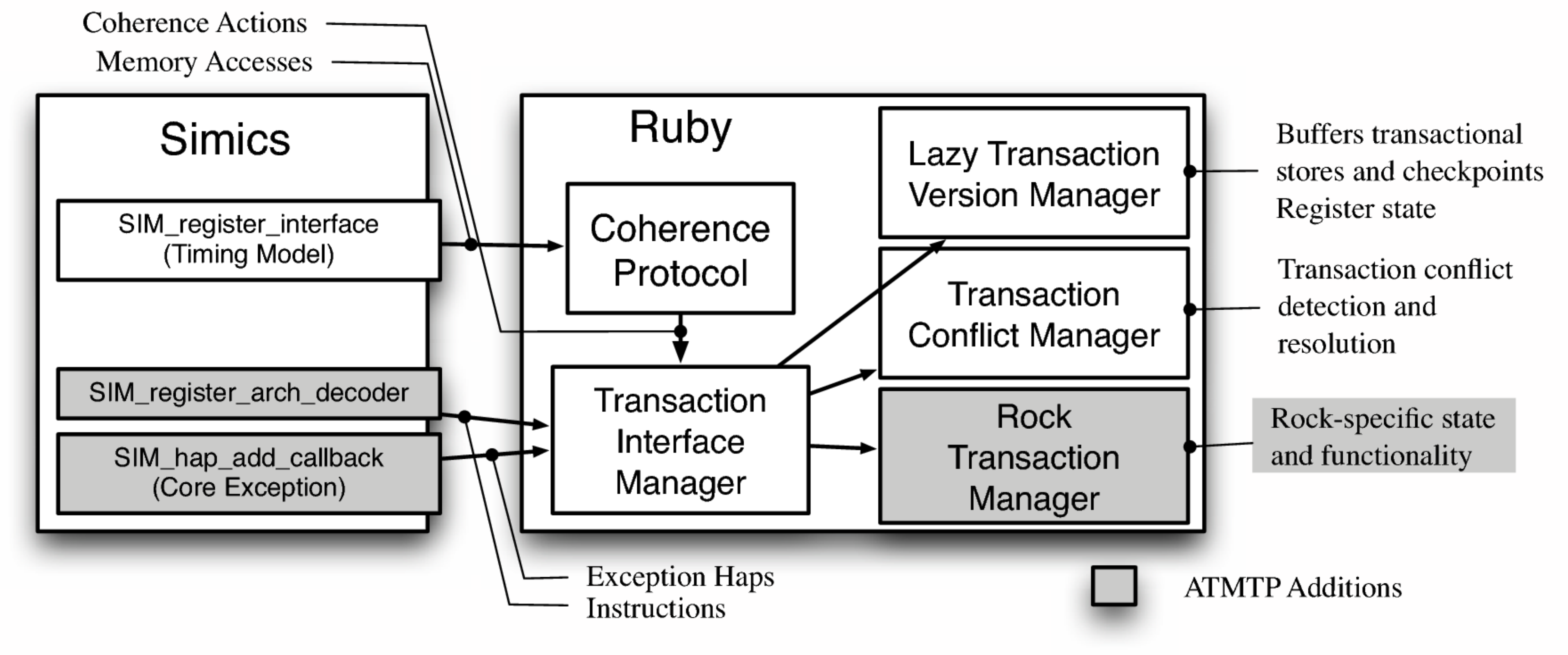
# GEMS 2.1

- Hot off the presses.
- Based on LogTM-SE:
  - Range of different options for TM implementation.
- "Base" machine is far more CMP-like.
  - All cores share a single L2$.
    - Inter-core communication far faster than for old SMP-like machine.
  - Renders our performance numbers slightly less useless.

# GEMS 2.1: Simplifications

- Choose “EL” (Wisconsin Taxonomy)
  - Eager conflict detection.
  - Lazy version management: hold speculative writes in store buffer.
- No longer need *requestor wins* mods:
  - Base machine already has that functionality.
- No longer need to cook up "mortal wound" strategy:
  - **`setAbortFlag()`**.

# GEMS 2.1 Implementation


Coherence Actions
Memory Accesses
Simics
SIM_register_interface (Timing Model)
SIM_register_arch_decoder
SIM_hap_add_callback (Core Exception)
Ruby
Coherence Protocol
Transaction Interface Manager
Lazy Transaction Version Manager
Transaction Conflict Manager
Rock Transaction Manager
Buffers transactional stores and checkpoints Register state
Transaction conflict detection and resolution
Rock-specific state and functionality
Exception Haps
Instructions
ATMTP Additions

- < 2000 lines of diffs to existing files.
- ~1000 lines of code in new files.
  - Not including "example usage" code.

# Outline

- Introduction.
- History and Motivation (GEMS 1.4).
- Current Functionality (GEMS 2.1).
- Tidbits and War Stories.
- Conclusion.

# Trap Hack (Synchronous Traps)

- Problem similar to that solved by mortal wound scheme.
- General idea: make use of functionality that already exists in processor simulator to accomplish abort-on-synchronous-trap behavior.
- "Hap" handlers: **`exception`**, **`exception-done`**.
- Exception occurs; **`exception`** fires.
  - > Too late to abort: too much state to unwind.
  - > Instead, arrange to execute **`retry`** instruction next.
- **`retry`** instruction runs; **`exception-done`** fires.
  - > *Now* it's safe to abort.
    - Restore registers, jump to **`<fail-pc>`**.

# “LBOLT” Problem

- Symptom: threads were mysteriously disappearing into the scheduler.
  - May very well have been responsible for an inordinate amount of pain -- Wisc, Auckland, Rochester (?), us.
- Time-based JVM tests were never terminating.
- LBOLT (tick counter) stopped incrementing.
- Incorrect to save/restore certain registers on transaction begin/abort: {**`tick,stick`**}**`_cmpr`**, **`softint`**.
- Side effect of writing back even the same same value to any of these registers causes tick clock to stop incrementing ==> scheduler no longer functions correctly.

# Conclusion

- ATMTP is a Simics/Ruby/GEMS/LogTM-SE–based simulator for Rock HTM feature.
- Provides a first-order approximation of the success and failure characteristics of transactions on Rock processor.
- ATMTP is available for public consumption!  Check SSRG and GEMS websites for details (see references).

# References

**`http://research.sun.com/scalable`**

**`http://www.cs.wisc.edu/gems`**

Companion Paper:

- **`http://research.sun.com/scalable/pubs/TRANSACT2008-ATMTP-Apps.pdf`**

## More information, papers, etc:
## http://research.sun.com/scalable

Dan Nussbaum

dan.nussbaum@sun.com