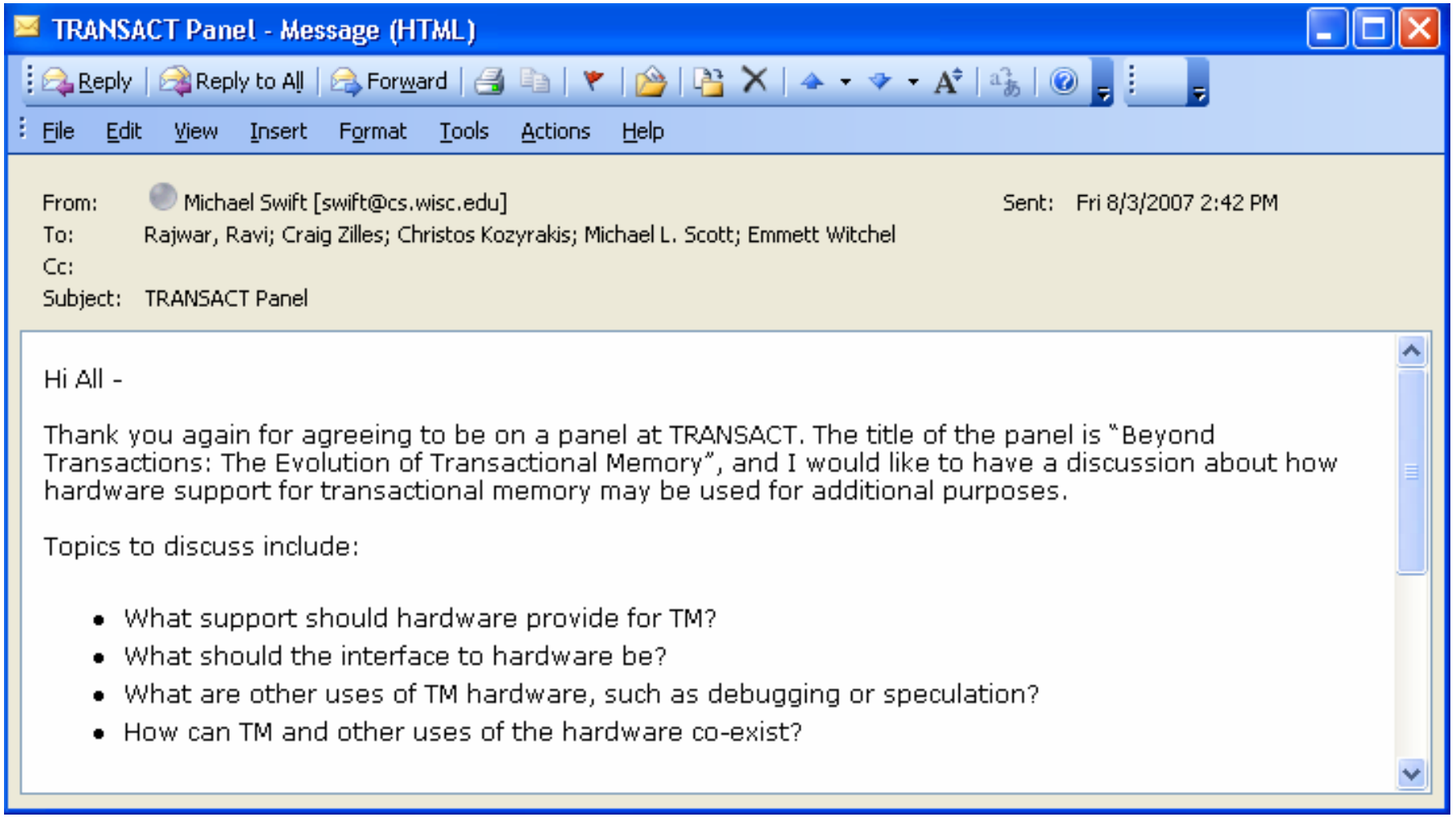
TRANSACT 2007 Panel

“Beyond Transactions: The Evolution of Transactional Memory”

Ravi Rajwar
8/16/2007

Disclaimer: These are strictly my personal opinions and not necessarily the opinions of my employer. They are based on my observing/working in this field for the past decade.


TRANSACT Panel - Message (HTML)
Reply
Reply to All
Forward
File Edit View Insert Format Tools Actions Help
From: Michael Swift [swift@cs.wisc.edu]
Sent: Fri 8/3/2007 2:42 PM
To: Rajwar, Ravi; Craig Zilles; Christos Kozyrakis; Michael L. Scott; Emmett Witchel
Cc:
Subject: TRANSACT Panel
Hi All -
Thank you again for agreeing to be on a panel at TRANSACT. The title of the panel is "Beyond Transactions: The Evolution of Transactional Memory", and I would like to have a discussion about how hardware support for transactional memory may be used for additional purposes.
Topics to discuss include:
What support should hardware provide for TM?
What should the interface to hardware be?
What are other uses of TM hardware, such as debugging or speculation?
How can TM and other uses of the hardware co-exist?

What is TM?

1. TM as a scalable synchronization model
   - Jensen et al., Herlihy&Moss 1993, Stone et al. 1993, …
   - Primary focus of TM (including many STMs…)
2. TM as a new programming model
   - Started ~2005… (composability)
   - Primary focus of most recent papers on STMs and HW support for STMs/Unbounded TMs.

- I ignore
  - Speculative parallelization
  - Speculative optimization
  - Coherence, Consistency
  - These are irrelevant to the overall question

TM for scalable synchronization

- Goal very domain specific
  - Performance: scalable synch (measurable)
  - Lock-free data structures (goodness?)
  - Non-blocking, Deadlock-free (how to measure?)
- Variants
  - Lock elision: gets most of what you need perf. wise
- Metric
  - Compare with locks – performance and usage
  - Easier? Harder? Hard to say
- I think this is a fine idea
  - Can clearly see domains where useful…
  - Is the domain large enough?
  - Does it make parallel programming easier?

TM as a new programming model (1)

- Taken research world by storm last 2 years
  - Lots of great research by many smart folks
  - Not just synctactic sugar…
- Fundamental benefit here is “composability”
  - Scalability/performance comes from earlier slide…
    - Insufficient for driving a “programming model”
- What is composability?
  - Misunderstood by many?
  - Can I call arbitrary stuff? Connect arbitrary modules? No
  - Talks only about abstraction
    - By itself does not provide correctness or deadlock-freedom
    - Property is actually very synchronization construct focused…

TM as a new programming model (2)

- Most of the work so far…
  - STM implementations
  - Allow to answer the questions:
    - Is this even going to ever work?
    - Do the claims and promise even have any merit?
- Similar to how research student writes a simulator
  - Test out ideas
  - Just writing a simulator does not guarantee a thesis
    - You have to prove/demonstrate your idea
- Assumed goodness w/o demonstration “in the field”
  - We are still in “is this even a good idea/worth it phase”…

What was the goal of the programming model?

- Are we…
  - The grad. student who sees simulator writing as end goal?
- TM was meant to be an easy programming model
  - We need to prove/disprove this
  - Looking at the papers on semantics, nesting, subtleties, etc., I argue that the discussion should have been put to rest right there…
- Solution looking for a problem?
  - Are we afraid to entertain the thought that TM as a programming model may not be the right thing?

What I want to see…

- Let's evaluate TM as a programming model
  - Experimental languages, experimental systems in the field
- Show me
  - Composability—how exactly, what exactly? Domain?
  - Ease of programming, metrics, etc.
  - Integration into existing world (many many many issues…)
  - Value…
    - If integration needs large scale rewrite/recompile, must have corresponding ROI
    - If I am doing large scale re-write/re-compile, is TM what I really want to do?
- Don't show me
  - Performance, Scalability
  - I know how to do that without a new programming model…

Final thoughts on TM as a programming model

- Panel focus
  - What HW support for this TM programming model

"Let us not put the cart before the horse.
Lest the horse may already be dead.
Let us first find that out..."