Phased Transactional Memory

Dan Nussbaum

Scalable Synchronization Research Group
Joint work with Yossi Lev and Mark Moir
Sun Microsystems Labs

August 16, 2007

Transactional Memory (TM)

- Replace locks with atomic sections.
- (Word-based.)
- Hardware or Software?
  - Software (STM) is cheaper and more flexible.
  - Hardware (HTM) is faster.
- Hybrid Transactional Memory (HyTM) gets best of both worlds.
  - Common case in hardware -- fast.
  - Uncommon case in software – correct.
  - Make effective use of *best-effort* hardware (Rock).

Hybrid Transactional Memory (HyTM)

- HyTM approach: compiler plus library.
- Two code paths:
  - *Hardware path* attempts to use transactional hardware.
    - Fast.
    - Transactions don't always succed:
      - Resource limitations.
      - "*Difficult*" instructions.
      - Contention.
  - *Software path* contains calls into an STM library.
    - Slower.
    - Take this path whenever the hardware comes up short.

HyTM: Instrument Hardware Path

- Problem: TM *hardware* unaware of software txns.
- Solution: make hardware *transactions* aware of software transactions, by augmenting hardware path:

```
Y = X + 5;      ==>     txn_begin;
                        if(!canHardwareRead(&X))
                          txn_abort;
                        tmp = X;
                        if(!canHardwareWrite(&Y))
                          txn_abort;
                        Y = tmp + 5;
                        txn_end;
```

Phased Transactional Memory

- Arrange to only be executing in a single *mode* at a time, system-wide.
  - **`HARDWARE`**, **`SOFTWARE`**, (others).
- Partition time into *phases*.
- When in **`HARDWARE`** mode, use the hardware path.
  - Don't have to allow for concurrent execution of software transactions: minimal overhead.
- When in **`SOFTWARE`** mode, use the software path.
  - Don't have to allow for concurrent execution of hardware transactions: fewer constraints on STM.

PhTM Prototype: Mode Transitions

- Start out in **`HARDWARE`** mode, until some transaction (T) has to run in software.
- Switch to **`SOFTWARE`** mode, making sure that no hardware transactions are still running.
- Run in **`SOFTWARE`** mode for a while.
  - Allow other transactions to start up in **`SOFTWARE`** mode.
- Switch back to **`HARDWARE`** mode.

PhTM Prototype: Schema

```
if (FirstTryInHardware()) {
HW:
  chkpt(F);
  HWPostCheckpoint(); <== if (mode != HARDWARE) fail;
  <body>
  commit();
} else {
  while (!<try body with STM>) {
F:  if (!RetryInSoftware()) goto HW;
  }
}
```

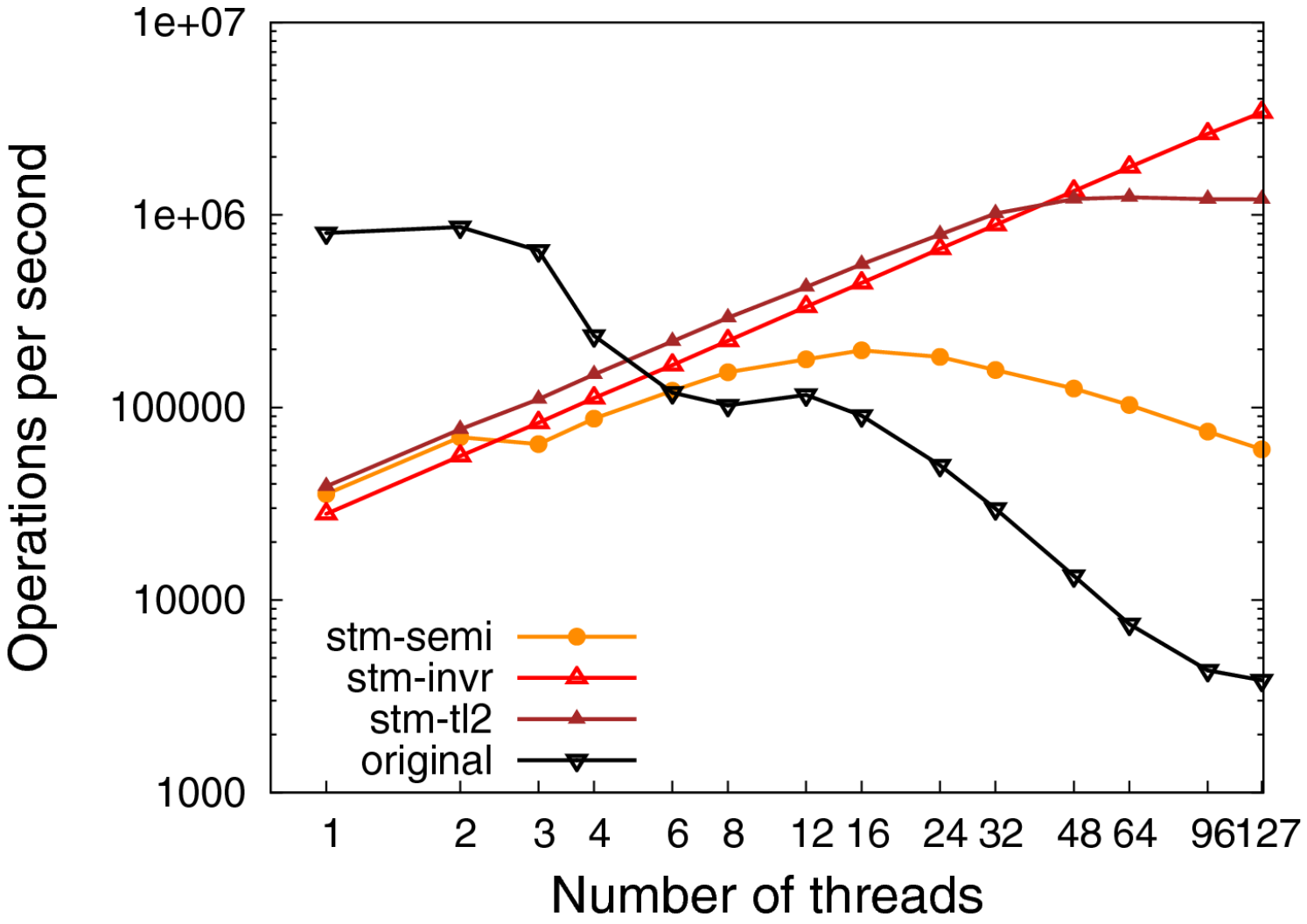
Experimental Set-Up

- STM-only experiments: E25K.
  - Big (144-core) SMP.
- Transactional hardware: simulator.
  - Wisconsin GEMS/ruby/LogTM (Simics-based).
  - Modified LogTM to better reflect best-effort constraints.

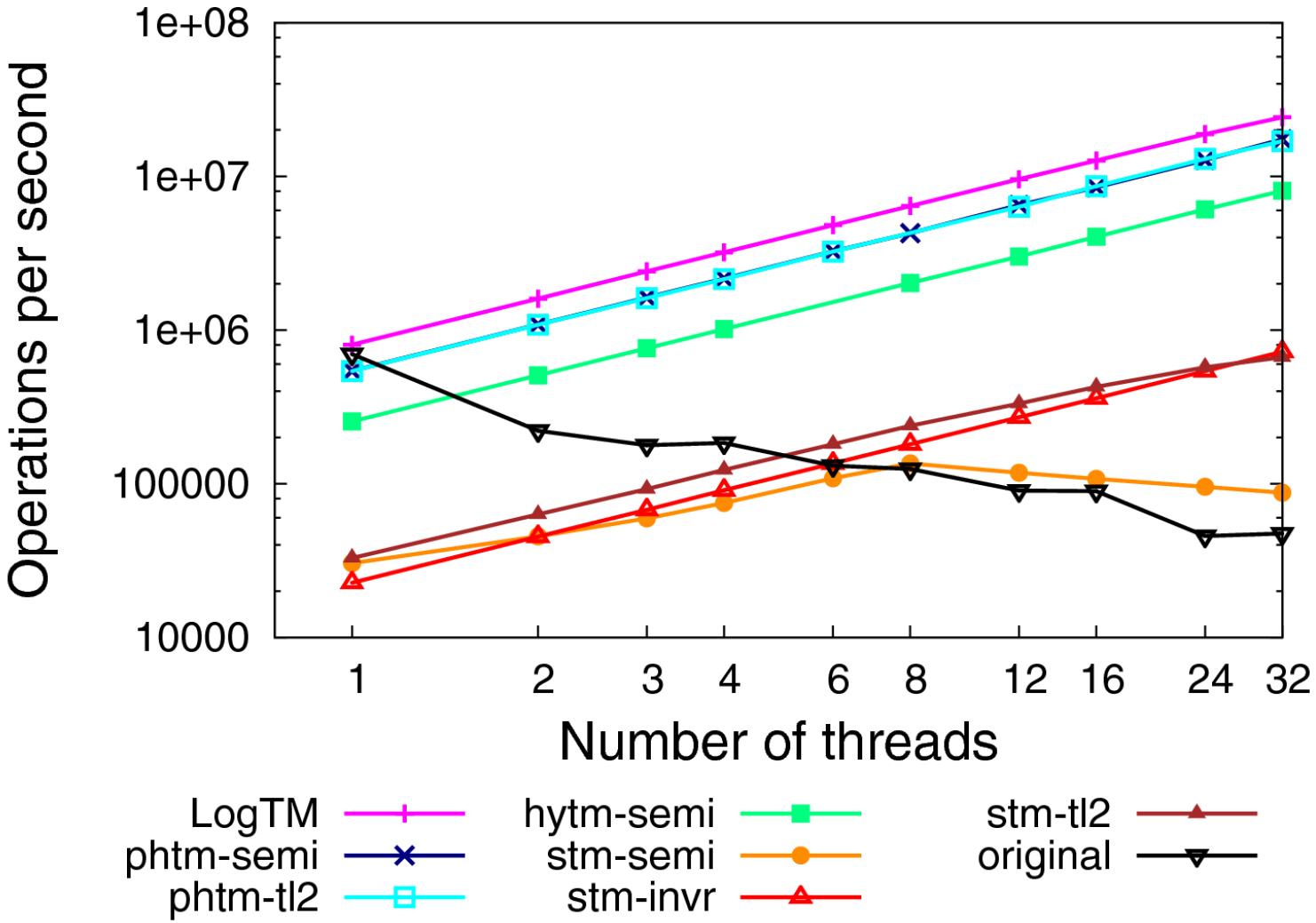
Benchmarks

- Transactified Berkeley DB Lock subsystem.
  - Every thread repeatedly locks and unlocks its own object.
  - Ought to scale perfectly.
- Red-Black Tree.
  - Tree is half full.
  - 20% inserts, 20% deletes, 60% lookups.
  - On larger machines, contention is significant.

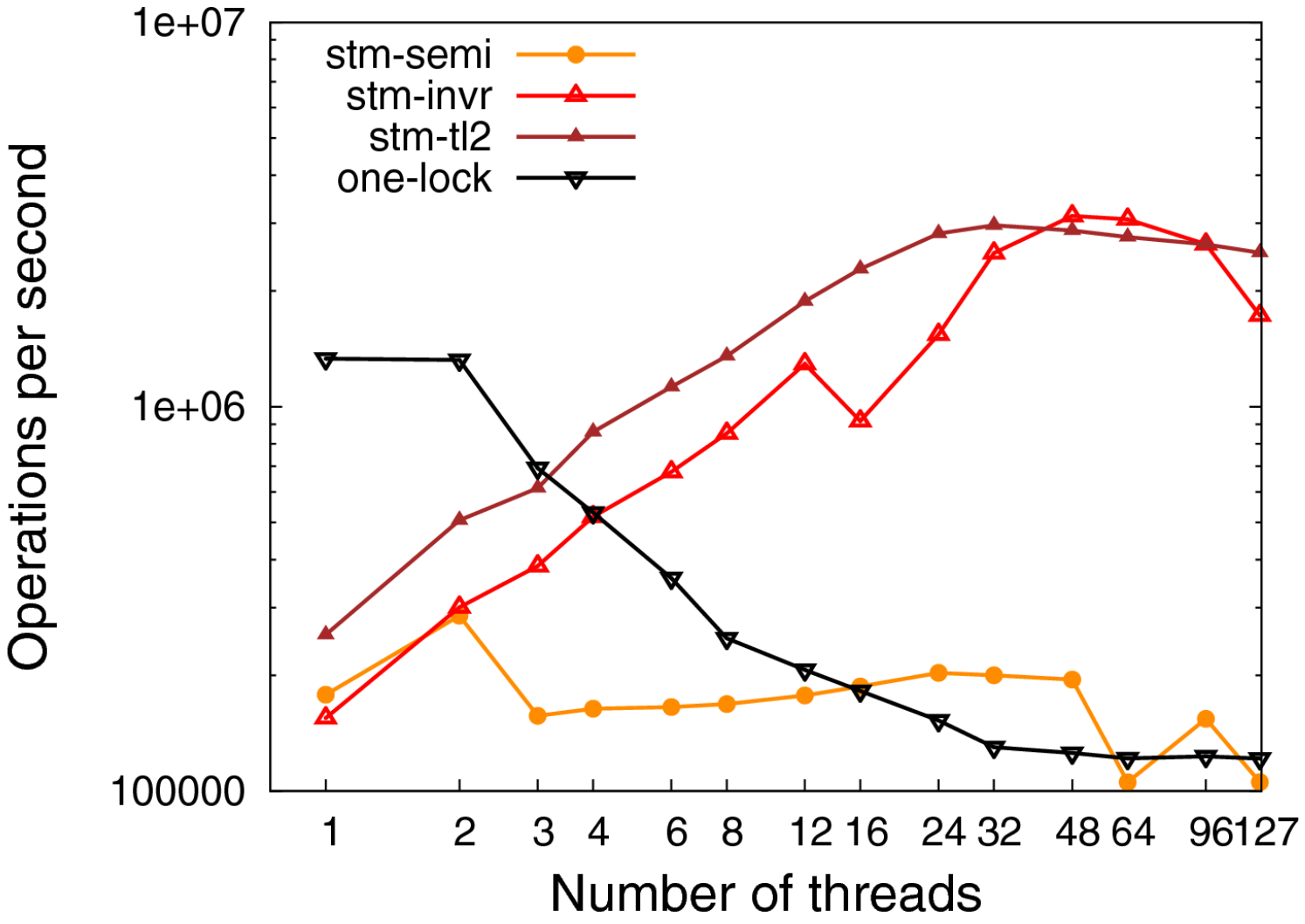
Berkeley DB: STM-only



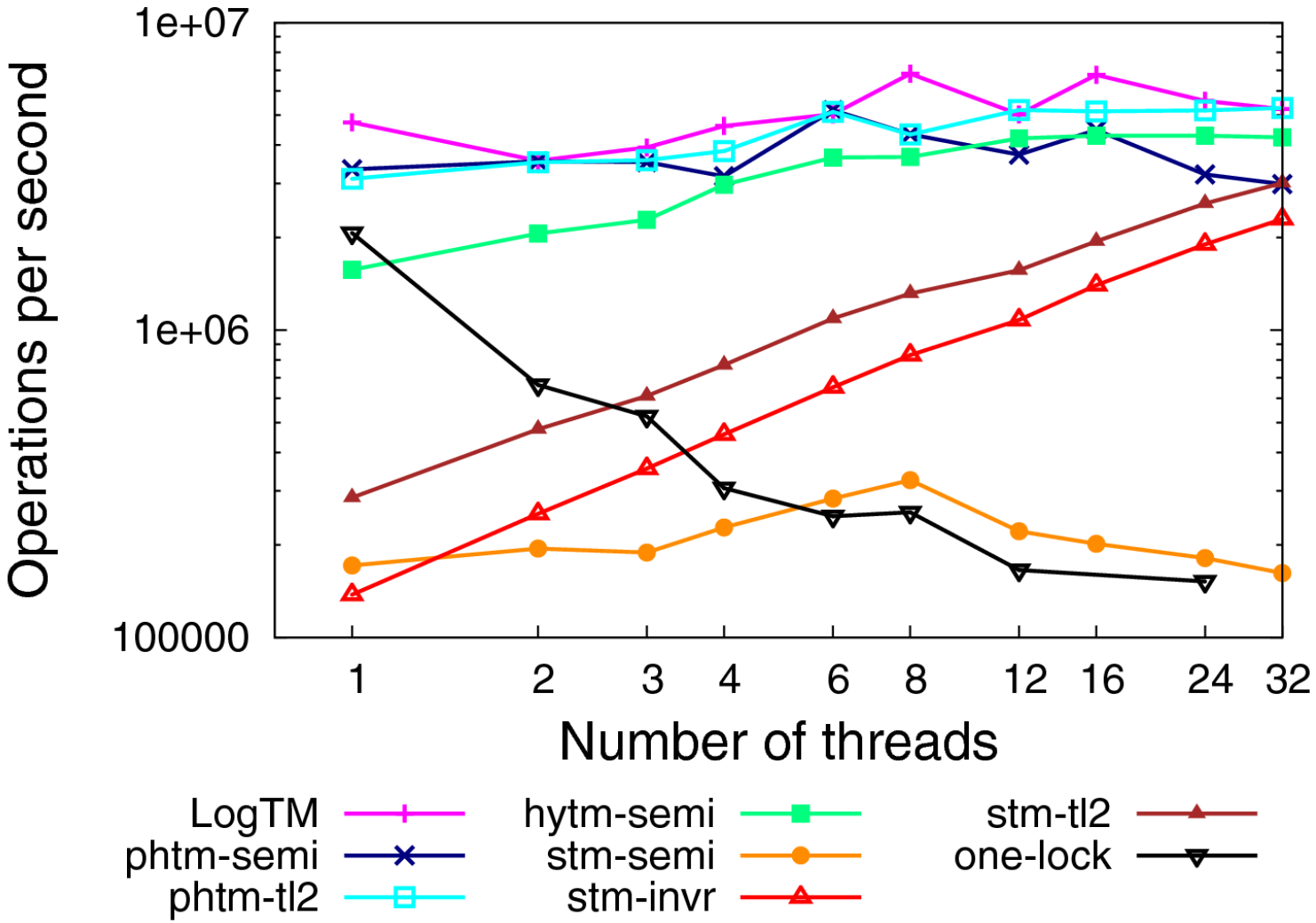
Berkeley DB: Simulations



RedBlackTree: STM-only



RedBlackTree: Simulations



Conclusions

- First-cut PhTM implementation already performs pretty well.
  - TL2 looks like best choice for PhTM's software phase.
- With a bit more work, we hope to close the gap between PhTM and pure hardware even further.

Future Work

- Performance Improvements.
  - Improve *phase management* strategy.
  - Improve *contention control* strategy.
  - Inline more of the fast path.
  - Compiler optimizations.
- More than just two (`HARDWARE, SOFTWARE`) modes.
  - Requires a more generalized approach (see paper).
  - `HYBRID` mode.
  - `SEQUENTIAL` mode.

References

- HyTM Paper (ASPLOS 2006)
  - http://research.sun.com/scalable/pubs/ASPLOS2006.pdf
- TL2 Paper (DISC 2006)
  - http://research.sun.com/scalable/pubs/DISC2006.pdf

Acknowledgements

- Dave Dice and Nir Shavit (TL2).
- Brian Whitney (E25K).
- Peter Damron (Compiler).
- Sasha Fedorova and Victor Luchangco (Discussions).
- Kevin Moore (Simulator).

Yossi Lev, Mark Moir and
Dan Nussbaum

yosef.lev@sun.com
mark.moir@sun.com
daniel.nussbaum@sun.com

PhTM Prototype: Mode Transitions (2)

`ModeIndicator = <mode, deferredCount, undeferredCount>`

- Transactions waiting to run in **`SOFTWARE`** mode increment **`deferredCount`**. Eventually one of them sets **`mode=SOFTWARE`**.
- Transactions that come into being when **`mode==SOFTWARE`** and **`deferredCount>0`** increment **`undeferredCount`** and then run in software.
- Transactions that come into being when **`mode==SOFTWARE`** and **`deferredCount==0`** wait for **`mode`** to change to **`HARDWARE`**.
- All software transactions decrement appropriate count after they commit. The last of these sees **`deferredCount==0 && undeferredCount==0`**, and sets **`mode=HARDWARE`**.

PhTM: Generalized Approach

- Many possible modes.
  - `HARDWARE, SOFTWARE, HYBRID, SEQUENTIAL, ...`
- Managing transitions between modes.
  - No interference between one phase and the next.
  - When to switch; which mode to switch to?

PhTM: Generalized Approach (cont.)

- **`ModeIndicator=<mode, mustFinishThis, otherTxns, nextMode, mustFinishNext, version>`**
- *Collect* candidates for next mode.

  `NextMode=<next Mode>; mustFinishNext++`
- Mode Transition
  - Only OK when **`mustFinishThis==0 && otherTxns==0`**

```
mode=nextMode;
nextMode=NONE;
mustFinishThis=mustFinishNext;
mustFinishNext=0;
version++;
```