```
[<c219ec5f>] security_sk_free+0xf/0x2
[<c2451efb>] __sk_free+0x9b/0x120
[<c25ae7c1>] ? _raw_spin_unlock_irqre
[<c2451ffd>] sk_free+0x1d/0x30
[<c24f1024>] unix_release_sock+0x174/
```

Managing Resource Limitation of Best-Effort HTM
Part-HTM

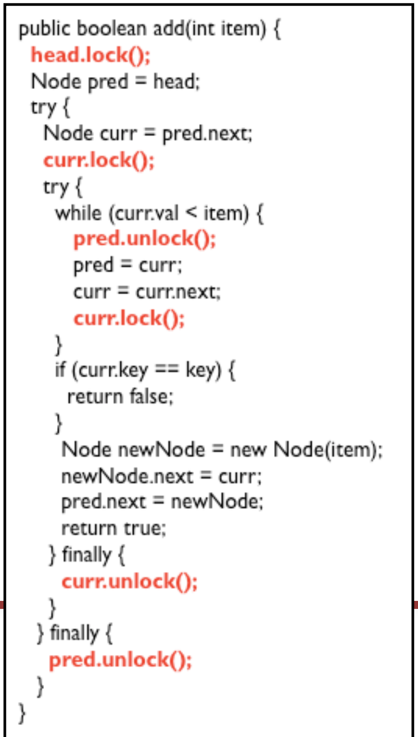
Mohamed Mohamedin, Roberto Palmieri,
Ahmed Hassan and Binoy Ravindran

Systems Software Research Group
Virginia Tech

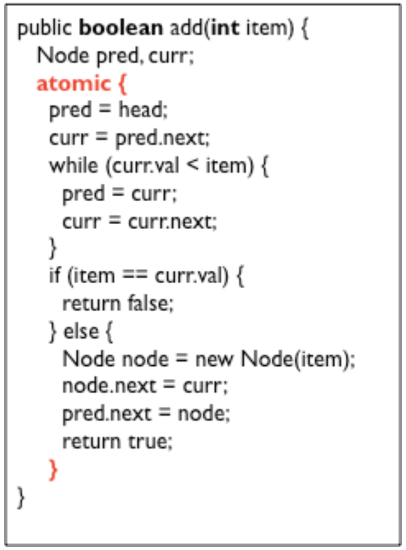
SPAA 2015

Transactional Memory

- Synchronization made easy
- No fine-tuned locking programming difficulties
- Programmer just marks parts of the code as atomic



```
public boolean add(int item) {
  head.lock();
  Node pred = head;
  try {
    Node curr = pred.next;
    curr.lock();
    try {
      while (curr.val < item) {
        pred.unlock();
        pred = curr;
        curr = curr.next;
        curr.lock();
      }
      if (curr.key == key) {
        return false;
      }
       Node newNode = new Node(item);
       newNode.next = curr;
       pred.next = newNode;
       return true;
     } finally {
       curr.unlock();
     }
   } finally {
     pred.unlock();
   }
}
```



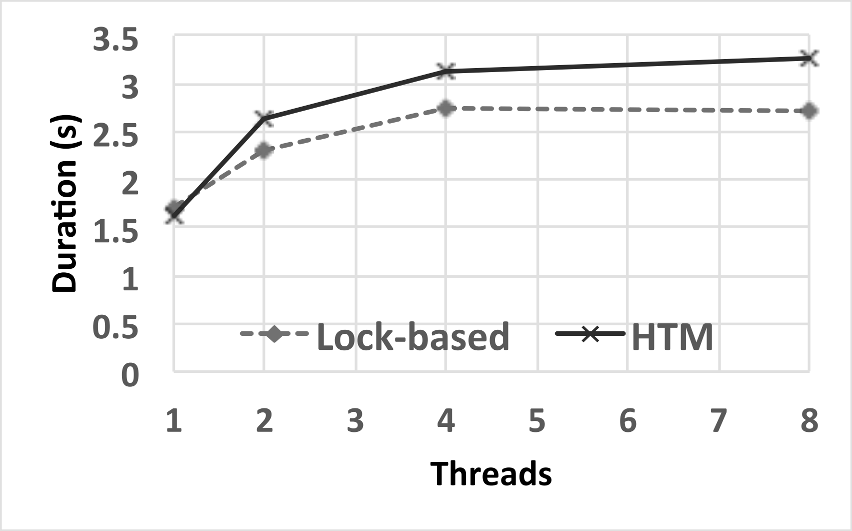
```
public boolean add(int item) {
  Node pred, curr;
  atomic {
    pred = head;
    curr = pred.next;
    while (curr.val < item) {
      pred = curr;
      curr = curr.next;
    }
    if (item == curr.val) {
      return false;
    } else {
      Node node = new Node(item);
      node.next = curr;
      pred.next = node;
      return true;
    }
  }
}
```

HTM of Commodity Processor

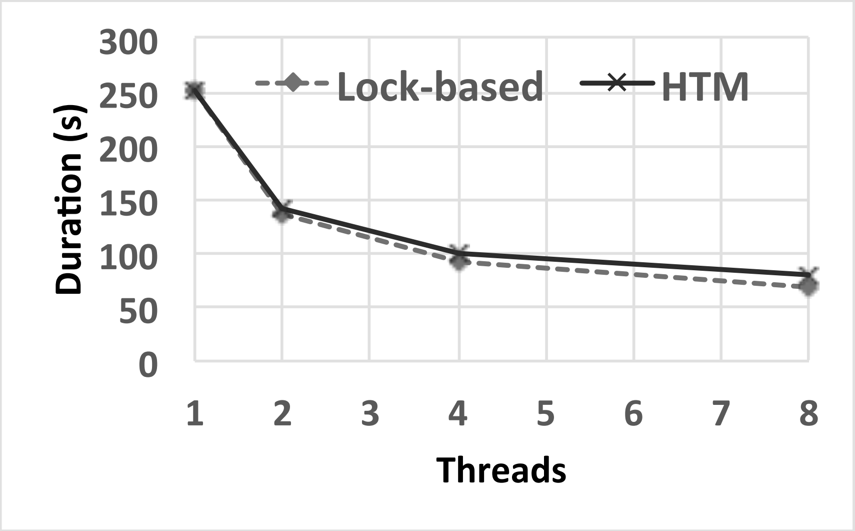
- Intel Haswell (TSX)
  - It is in your laptop!
- Integrated into the hardware cache-coherence protocol

HTM Performance

- As good as fine-grained locking and sometimes better
  - Ease of programming



Memcached



Fluidanimate (Parsec)

Is HTM Perfect?...no

- Best-effort
  - – Limited in size & time
    - Resource limitations
  - – Must define a software fallback path

Previous Work

- Mainly focused on tuning fallback path
  - Tuning the number of retries in HTM
  - How to use STM efficiently as a fallback
- What about transactions that cannot fit in HTM?
  - Can we still leverage HTM capabilities to execute them?

Part-HTM

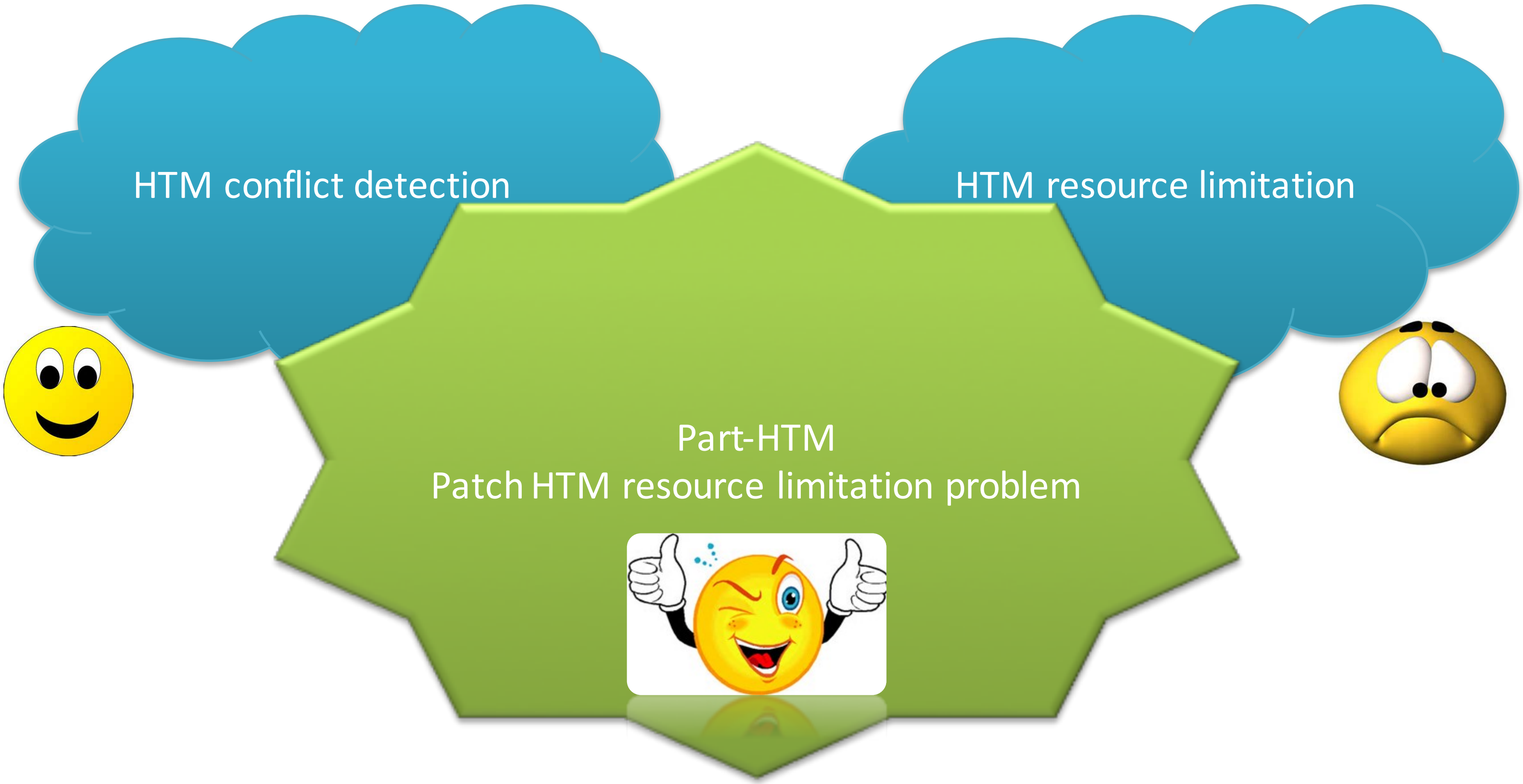
HTM conflict detection

Part-HTM

HTM conflict detection

HTM resource limitation

Part-HTM



HTM conflict detection

HTM resource limitation

Part-HTM

Patch HTM resource limitation problem

Part-HTM

- Handles the resource-limitations problem
  - If a transaction does not fit → split it into parts
  - Exploit HTM advantages to execute those parts
- Core Idea
  - Try first in HTM
  - If it fails due to resource limitations
    - Divide into sub-HTM
- Problems
  - Isolation
    - Sub-HTM commits directly to the memory

Isolation?

- Software framework
  - Lightweight instrumentation
    - Signatures and bitwise operations
  - Write locks
    - Needed because other transactions cannot
      - Overwrite objects
      - Read intermediate state (e.g., a committed sub-HTM)
  - Undo-log
  - Software validation between sub-HTMs

Expected Performance

- Close to HTM when HTM is the best
- Better than STM when HTM cannot commit most transactions
  - Exploiting HTM fast execution for sub-HTM

Part-HTM vs. Part-HTM-O

Non-opaque

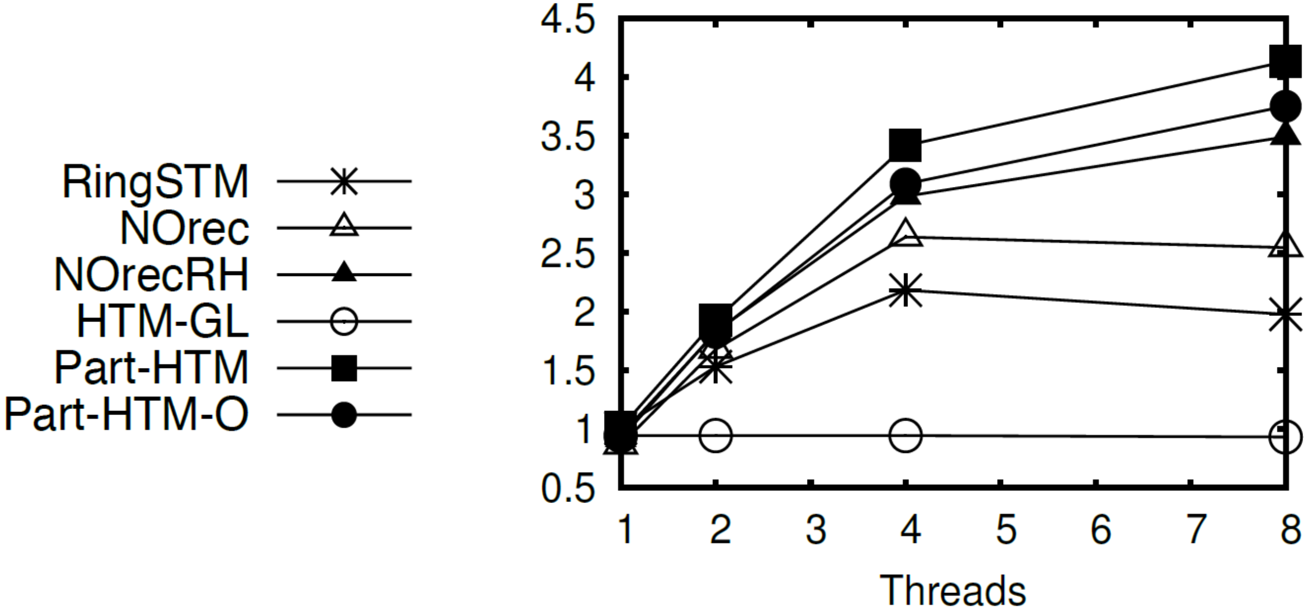
- Global write-locks
- Lazy validation
- Leveraging HTM sandboxing

Opaque

- Encounter time locks
- Encounter time checking
- Validate after each sub-HTM and when the global timestamp changes

Evaluation

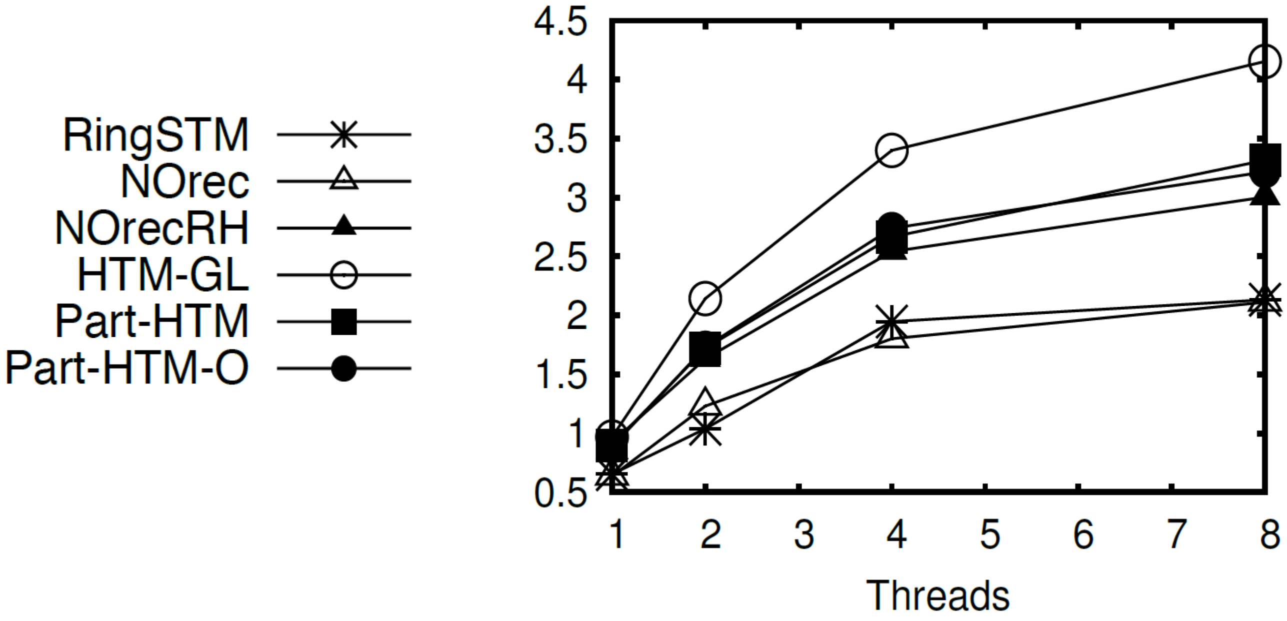
- Labyrinth from STAMP



	Conflict	Capacity	Explicit	Other
HTM-GL	10.11%	70.76%	0.04%	19.09%
PART-HTM	93.95%	1.09%	1.14%	3.82%

Evaluation (2)

- Kmeans from STAMP



Thanks!

Questions?

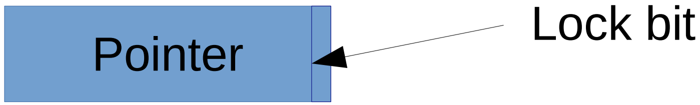
Hyflow

Systems Software Research Group

Research project's web-site: www.hyflow.org

Address-Embedded Locks

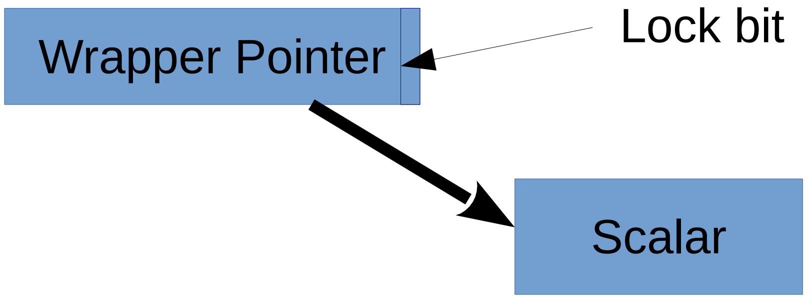
- All memory addresses are aligned
- Steel a single bit from the address
- Shared location is a pointer



Pointer

Lock bit

- Shared location is a scalar
  - Add a wrapper pointer



Wrapper Pointer

Lock bit

Scalar

Transactions has no direct access to it

Fine-grained synchronization with fallback path

- No-meta data shared.
- No global lock (in principle, but at the end is needed for irrevocable calls, but it is rarely used).
- No lock-table

- Maybe not cache friendly
- Reengineering effort to wrap addresses

HTM

Upon read/write

- HTM checks if object is locked by reading the last bits of the address into the wrapper

STM

Upon read/write

- STM writes the lock on the object into the address accessed through the wrapper

Upon commit/abort

- Release locks iterating over accessed objects