An Integrated Hardware-Software Approach to Transactional Memory

Sean Lie

6.895 Theory of Parallel Systems

Monday December 8th, 2003

Transactional Memory

- Transactional memory provides atomicity without the problems associated with locks.

Locks

```
if (i<j) {
  a = i; b = j;
} else {
  a = j; b = i; }
Lock(L[a]); Lock(L[b]);
Flow[i] = Flow[i] – X;
Flow[j] = Flow[j] + X;
Unlock(L[b]); Unlock(L[a]);
```

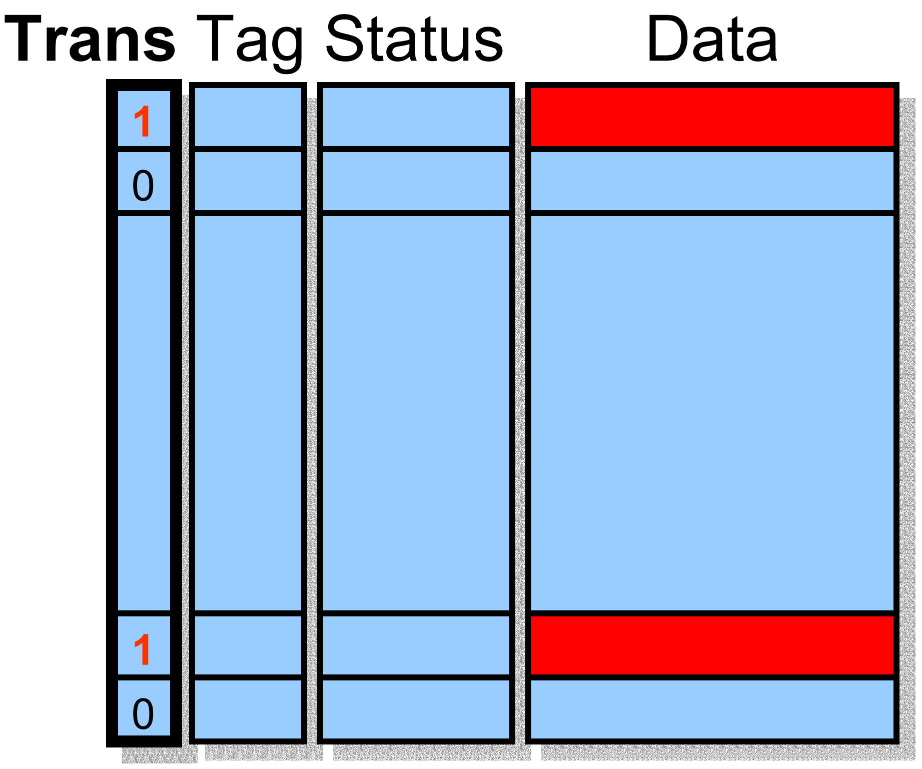
Transactional Memory

```
StartTransaction;
Flow[i] = Flow[i] – X;
Flow[j] = Flow[j] + X;
EndTransaction;
```

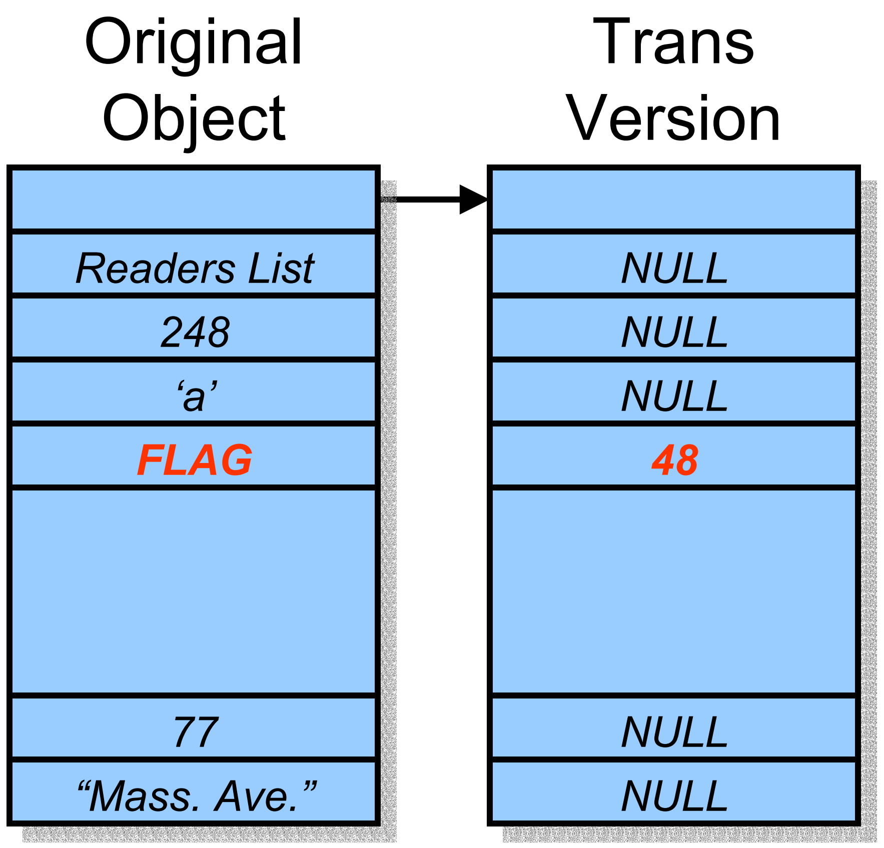
- I propose an integrated hardware-software approach to transactional memory.

Hardware Transactional Memory

- HTM: Transactional memory can be implemented in hardware using the cache and cache coherency mechanism. [Herlihy & Moss]
- Uncommitted transactional data is stored in the cache.
- Transactional data is marked in the cache using an additional bit per cache line.
- HTM has very low overhead but has size and length limitations.



Software Transactional Memory



FLEX Software Transaction System

- **STM: Transactional memory can be implemented in software using compiler and library support.**
- **Uncommitted transactional data is stored in a copy of the object.**
- **Transactional data is marked by flagging the object field.**
- **STM does not have size or length limitations but has high overhead.**

Results

- An integrated approach gives the best of both worlds.
  - Common case:
    - HTM mode - Small/short transactions run fast.
  - Uncommon case:
    - STM mode - Large/long transactions are slower but possible.
- An integrated hardware-software transactional memory system was implemented and evaluated.
  - HTM was implemented in the UVSIM software simulator.
  - A subset of STM functionality was implemented for the benchmark applications.
  - HTM was modified to be software-compatible.

Hardware vs. Software

- **HTM has much lower overhead than STM.**
- **A network flow μbenchmark (node-push) was implemented for evaluating overhead.**

1 processor overheads:

Atomicity Mechanism	**"*Worst*"**	**"*More Realistic*"**
	Cycles (% of Base)	
Locks	**505%**	**136%**
HTM	**153%**	**104%**
STM	**1879%**	**206%**

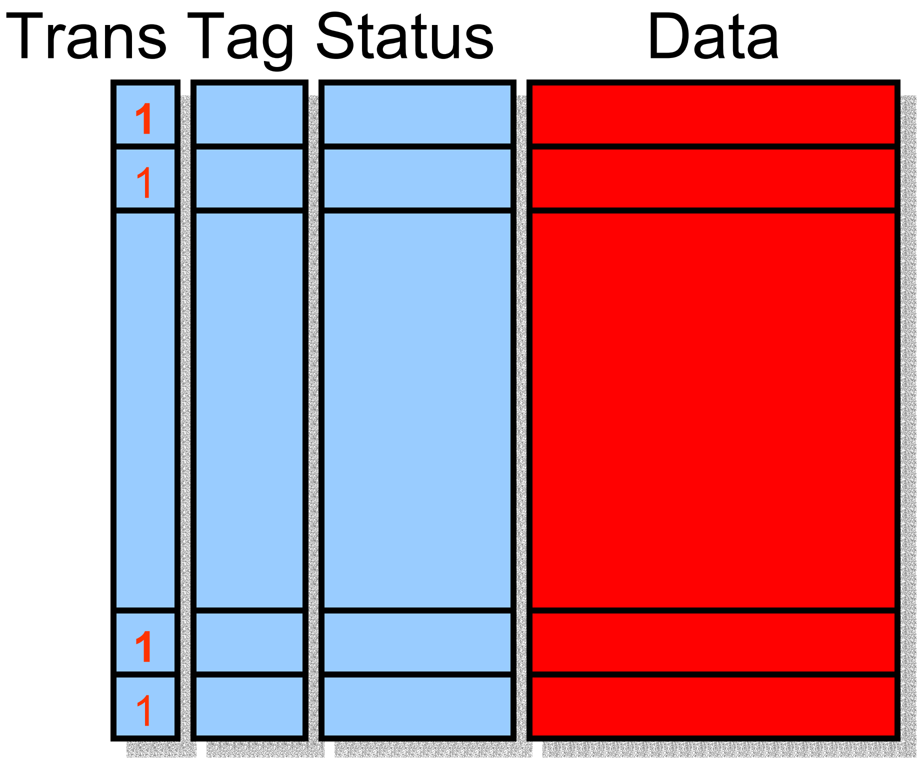
"worst" case:
Back-to-back small transactions

"more realistic" case:
Some processing between small transactions

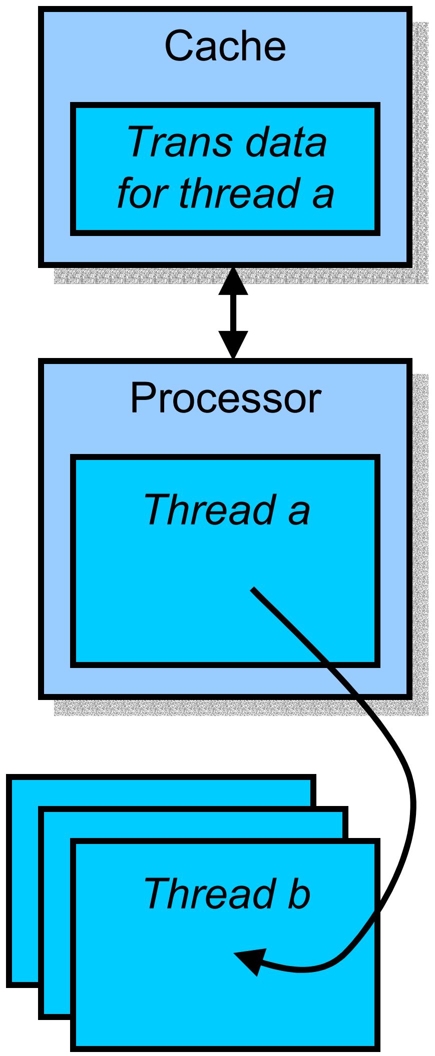
- **However, HTM has 2 serious limitations.**

Hardware Limitation: Cache Capacity

- **HTM uses the cache to hold all transactional data.**
- **Therefore, HTM aborts transactions larger than the cache.**
- **Restricting transaction size is awkward and not modular.**
  - **Size will depend on associativity, block size, etc. in addition to cache size.**
  - **Cache configuration change from processor to processor.**



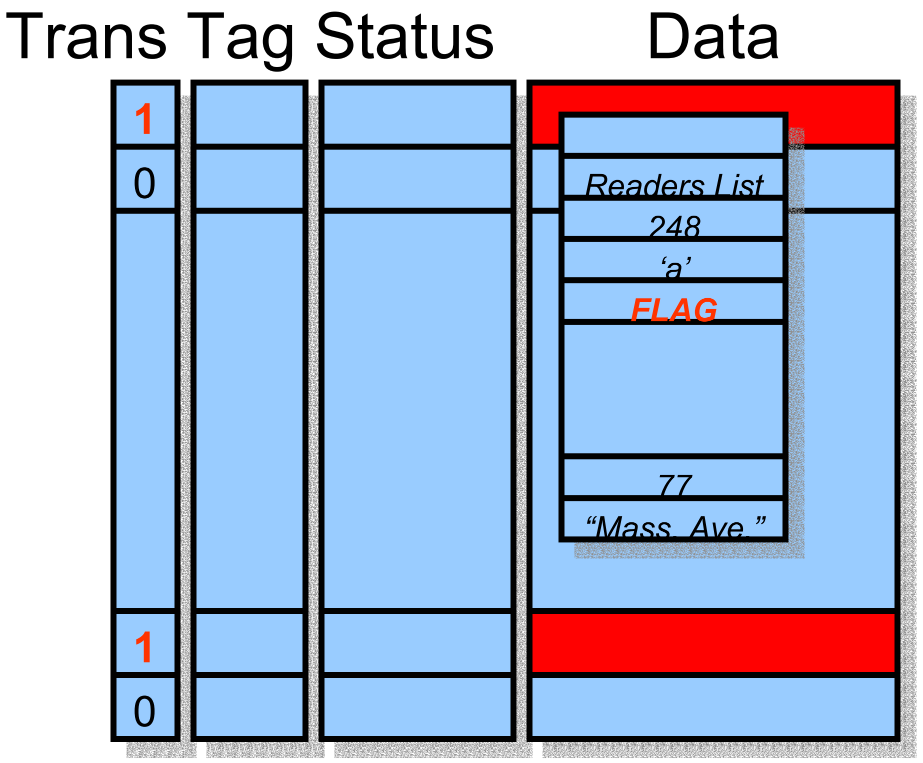
Hardware Limitation: Context Switches



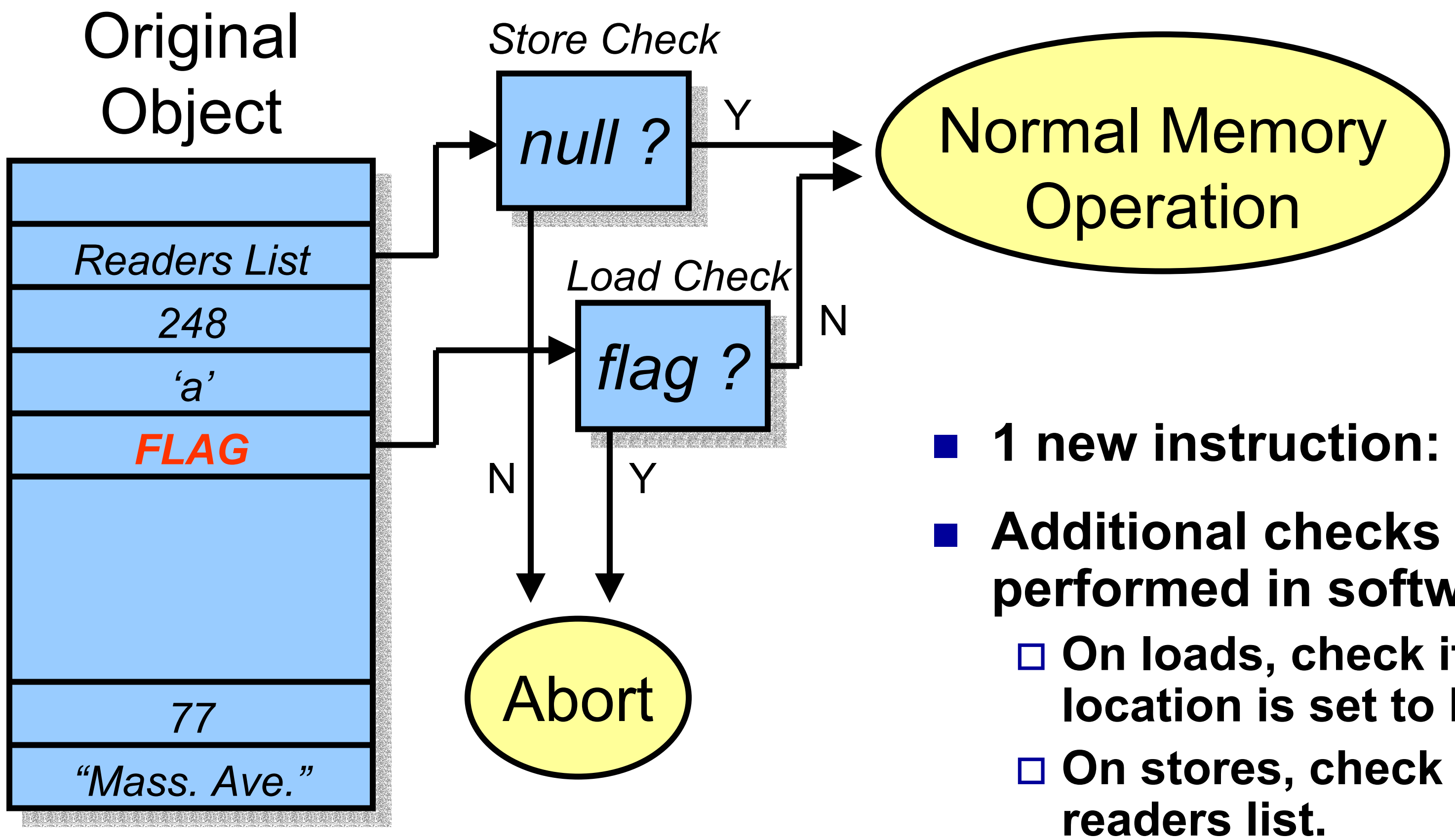
- The cache is the only transactional buffer for all threads.
- Therefore, HTM aborts transactions on context switches.
- Restricting context switches is awkward and not modular.
  - Context switches occur regularly in modern systems (e.g.. TLB exceptions).

HSTM: An Integrated Approach

- Transactions are switched from HTM to STM when necessary.
- When a transaction aborts in HTM, it is restart in STM.
- HTM is modified to be software-compatible.



Software-Compatible HTM

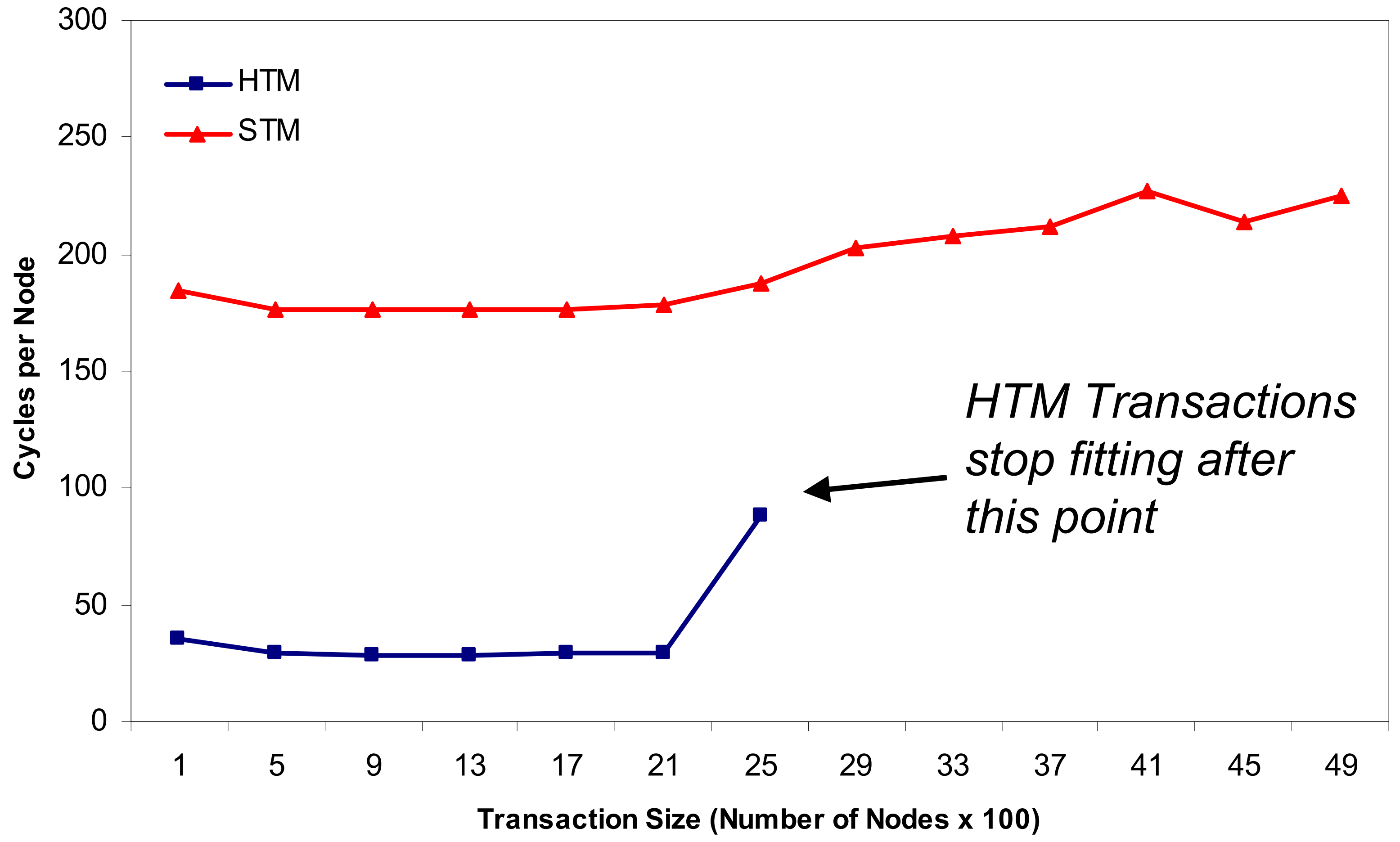


- 1 new instruction: xAbort
- Additional checks are performed in software:
  - On loads, check if the memory location is set to FLAG.
  - On stores, check if there is a readers list.
- Software checks are slow.

- Performing checks adds a 2.2x performance overhead over pure HTM in "worst" case (1.1x in "more realistic" case).

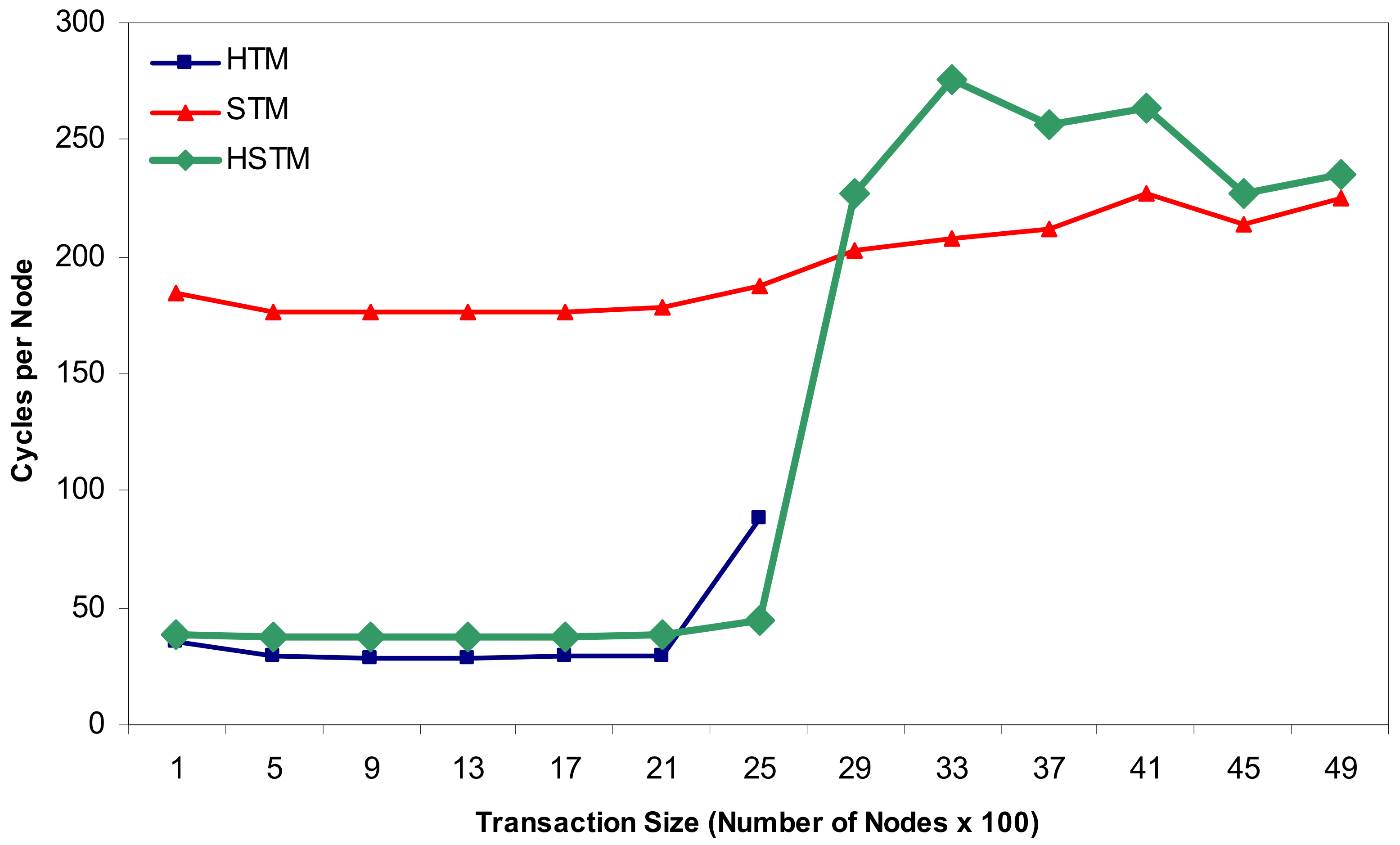
Overcoming Size Limitations

- The node-push benchmark was modified to touch more nodes to evaluate size limitations.
- HSTM uses HTM when possible and STM when necessary.



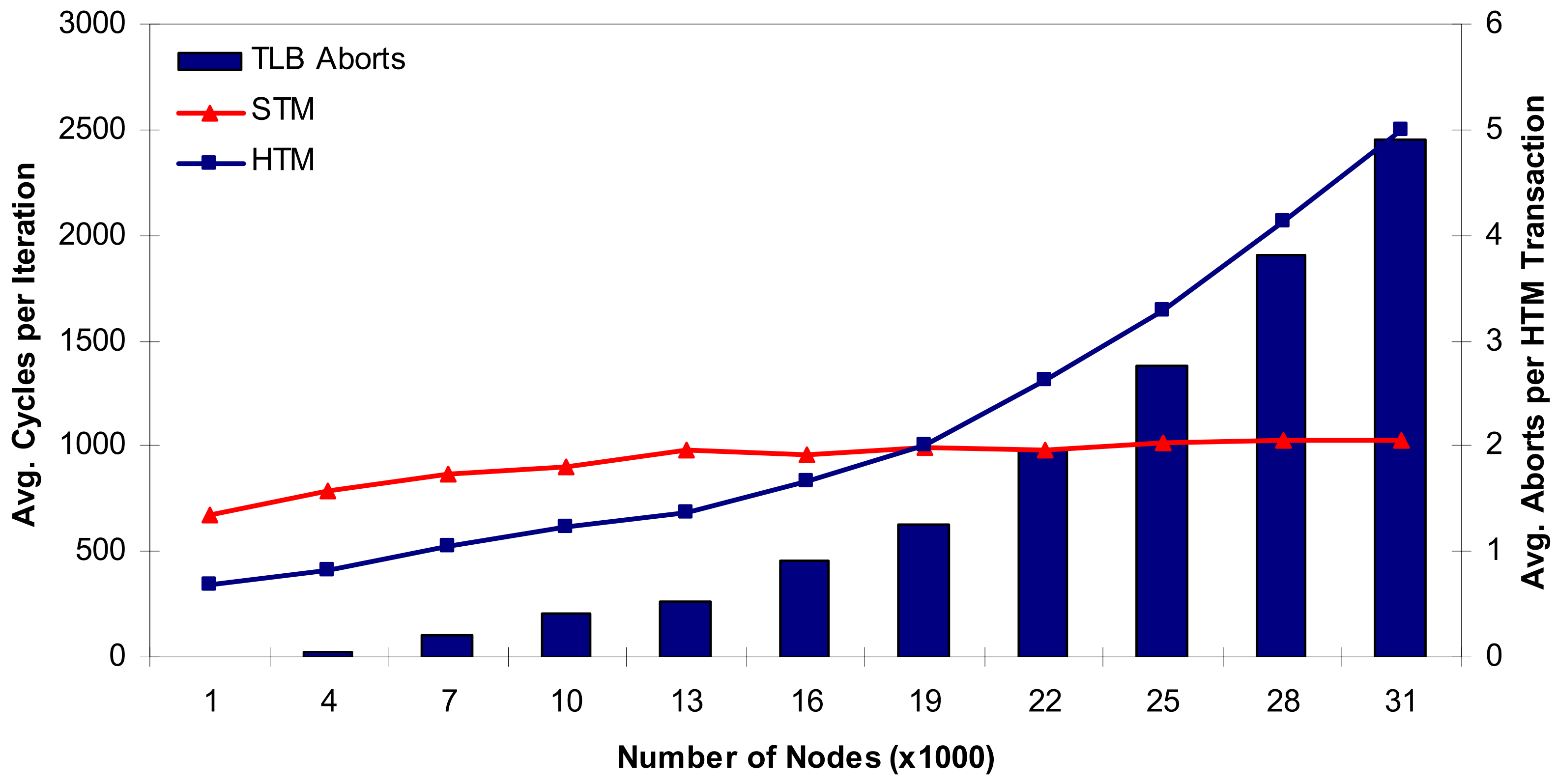
Overcoming Size Limitations

- **The node-push benchmark was modified to touch more nodes to evaluate size limitations.**
- **HSTM uses HTM when possible and STM when necessary.**



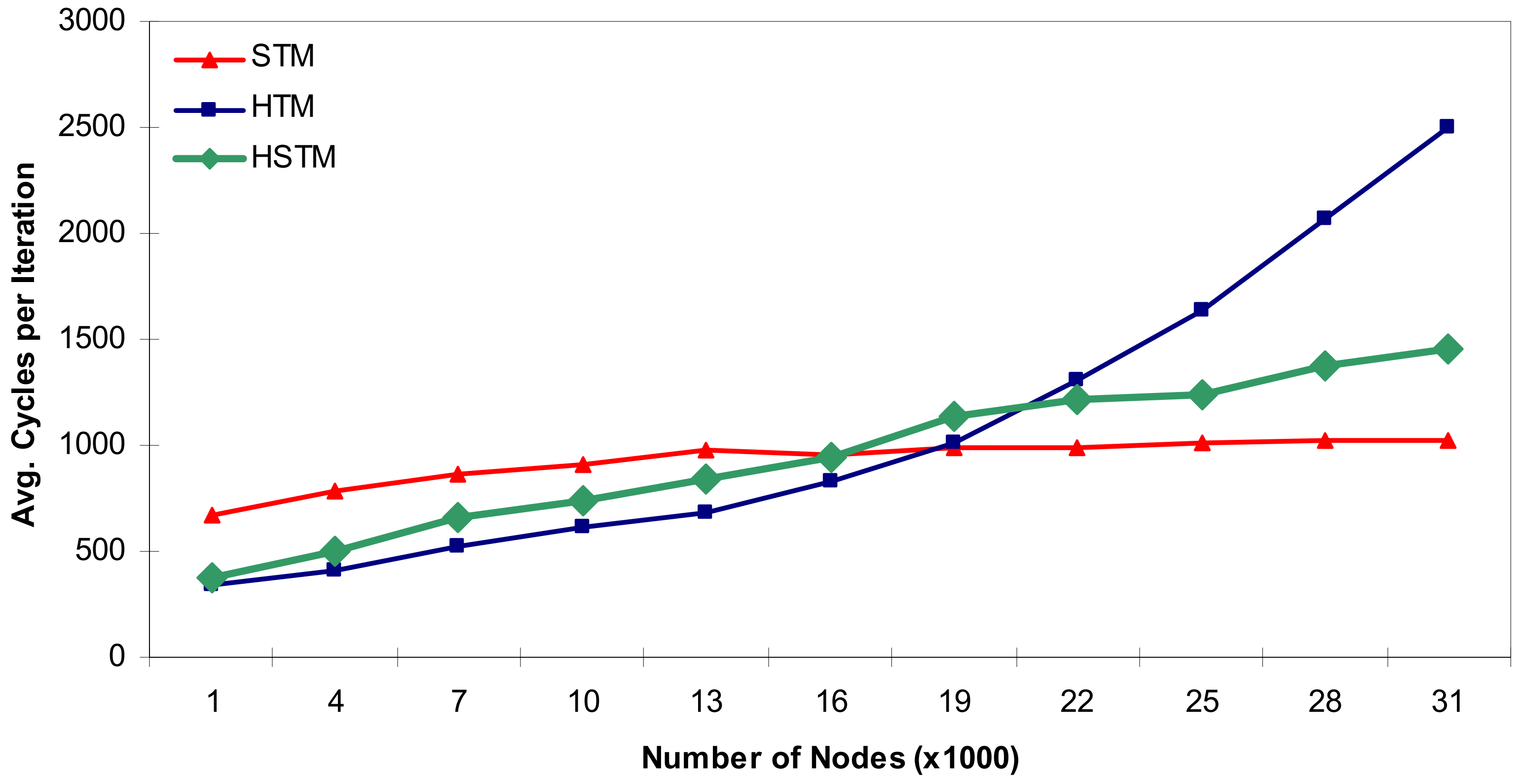
Overcoming Context Switching Limitations

- Context switches occur on TLB exceptions.
- The node-push benchmark was modified to choose from a larger set of nodes.
  - More nodes → higher probability of TLB miss (P_{abort}).
- HSTM behaves like HTM when P_{abort} is low and like STM when P_{abort} is high.



Overcoming Context Switching Limitations

- Context switches occur on TLB exceptions.
- The node-push benchmark was modified to choose from a larger set of nodes.
  - More nodes → higher probability of TLB miss (P_{abort}).
- HSTM behaves like HTM when P_{abort} is low and like STM when P_{abort} is high.



Conclusions

- An integrated approach gives the best of both worlds.
  - Common case:
    - HTM mode - Small/short transactions run fast.
  - Uncommon case:
    - STM mode - Large/long transactions are slower but possible.
- Trade-offs:
  - “STM mode” is not has fast as pure STM.
    - This is acceptable since it is uncommon.
  - “HTM mode” is not has fast as pure HTM.
    - Is this acceptable?

Future Work

- Full implementation of STM in UVSIM
- Integration of software-compatible HTM into the FLEX compiler
- Evaluate how software-compatible HTM performs for parallel applications
- Should software-compatible modifications be moved into hardware?
- Can a transaction be transferred from hardware to software during execution?