Transactional Memory at BoostCon'10

BoostCon 2010

Vicente Botet, Justin E. Gottschlich, Maurice Herlihy, Mark Moir, Tatiana Shpeisman, Michael Wong

TM Lineup

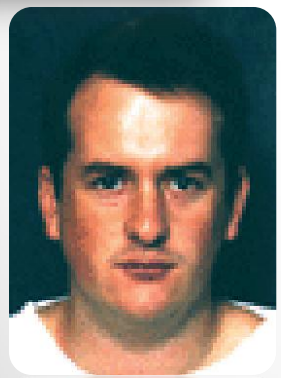
Vicente Botet

Maurice Herlihy

Justin Gottschlich

Tatiana Shpeisman

Michael Wong



Mark Moir

Tatiana (Part II)

- Library
  - TBoost.STM Engine
- Compiler
  - Proposed C++ language extension
  - IBM
  - Sun / Oracle
  - Intel

- Experts panel
  - Maurice Herlihy
  - Mark Moir
  - Tatiana Shpeisman
  - Michael Wong
- Submit questions today

Why Should You Care?

- May affect you
  - C++ language extension
  - Library / Compiler
- Insight
  - Understand parallel algorithms
  - Understand how TM applies
- Contacts

The TBoost.STM Engine

BoostCon 2010

Justin E. Gottschlich

University of Colorado-Boulder

Raytheon

Motivation

- Problem
  - TM is not fast enough! (Cascaval et al., 2008)
- Reason
  - Conflict Detection and Opacity
  - Most TMs use ***Validation***
- Our solution:
  - ***Full Invalidation***
  - ***InvalSTM***

TM Performance Bottleneck

- ***Conflict Detection***
  - Determine if transaction can commit
    - (Papadimitrou, "Theory of Database Concurrency Control," 1986)
- ***Opacity***
  - Keep in-flight transactions consistent
    - (Guerraoui & Kapalka, PPoPP'08)

Conflict Detection

Conflict: $W_{T1} \cap (W_{T2} \cup R_{T2}) \neq \emptyset$

- **Validation** (T2)
  - ***Analyze the Past***
    - Version # is same
- **Invalidation** (T1)
  - ***Analyze the Future***
    - *T2.valid = false*

Opacity

- **Validation**
  - Version # is same
- **Invalidation**
  - **Check *valid != false***

Opacity: Validation

```
atomic { return (x == 0 ? 0 : y / z); }
```

read x → store x's ver / ref

read y → store y's ver / ref,
opacity: validate x

read z → store z's ver / ref,
opacity: validate x, y

commit → *conflict*: validate x, y, z

Opacity: Invalidation

```
atomic { return (x == 0 ? 0 : y / z); }
```

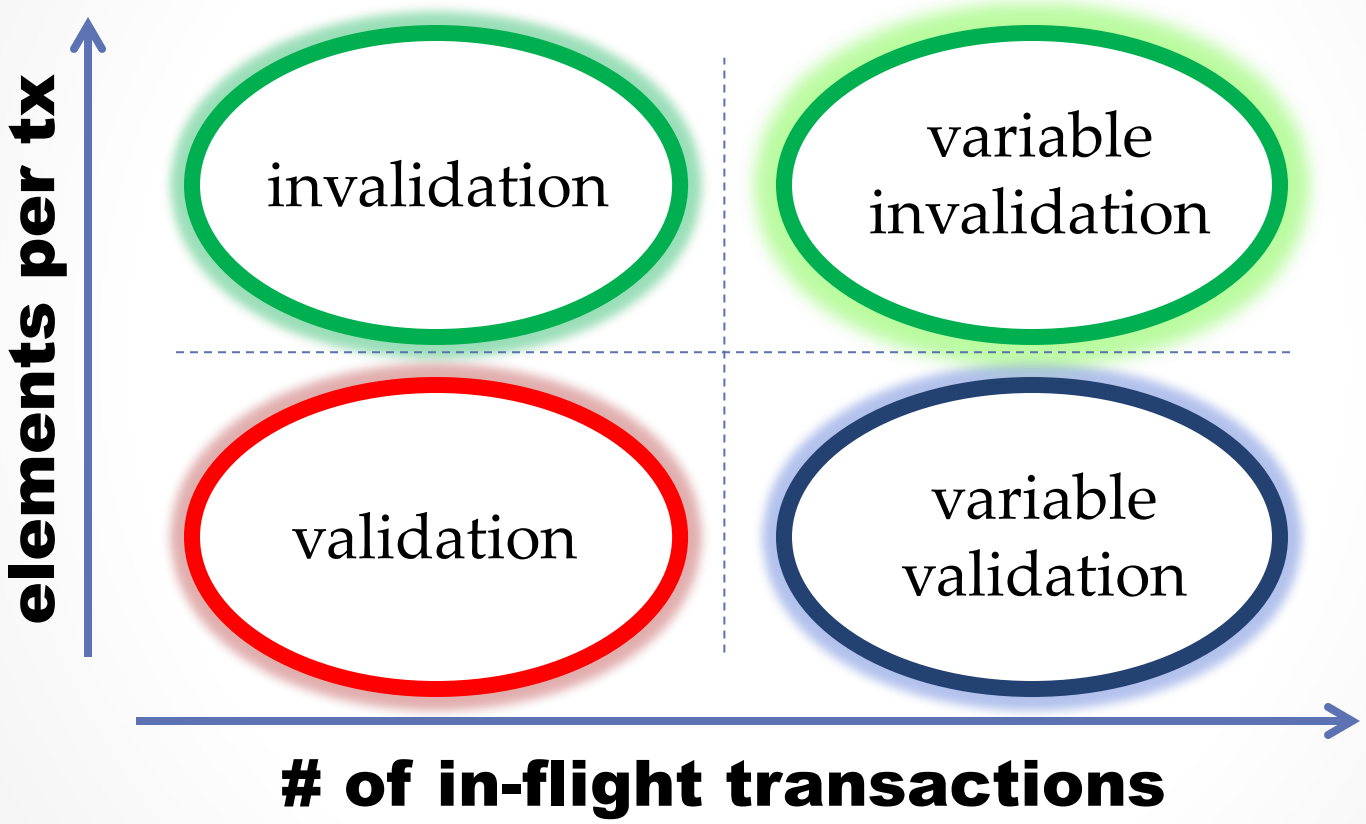
read x → store ref to x

read y → store ref to y
opacity: if (!*valid)* retry

read z → store ref to z
opacity: if (!*valid)* retry

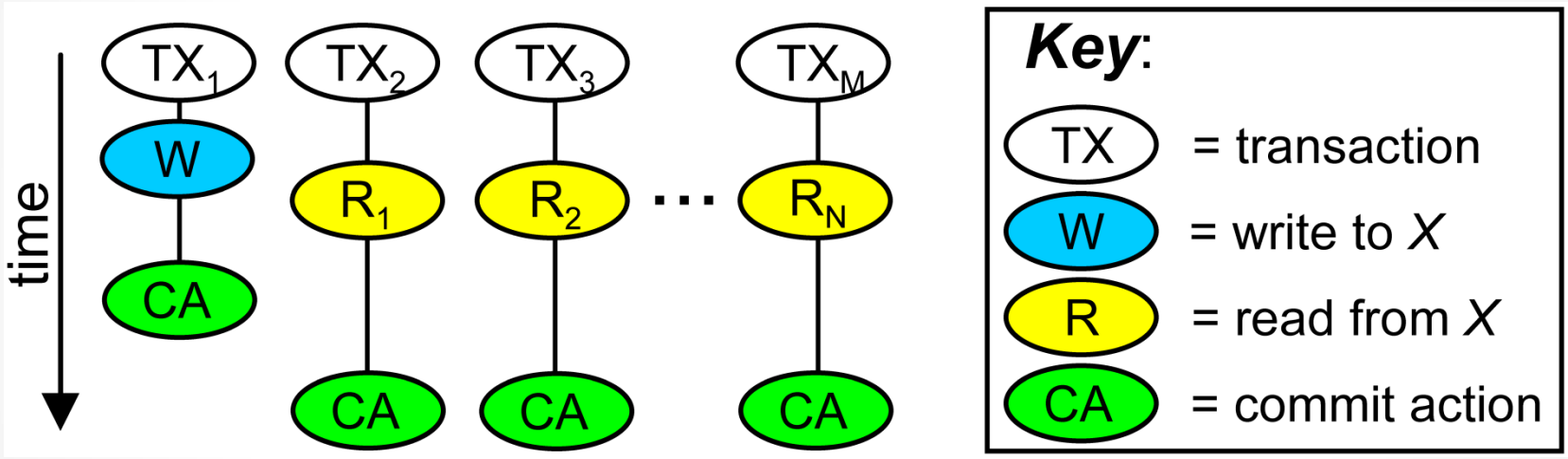
commit → *conflict*: none

Validation Vs. Invalidation



Contending + Concurrent Workload

1-Writer, N-Reader

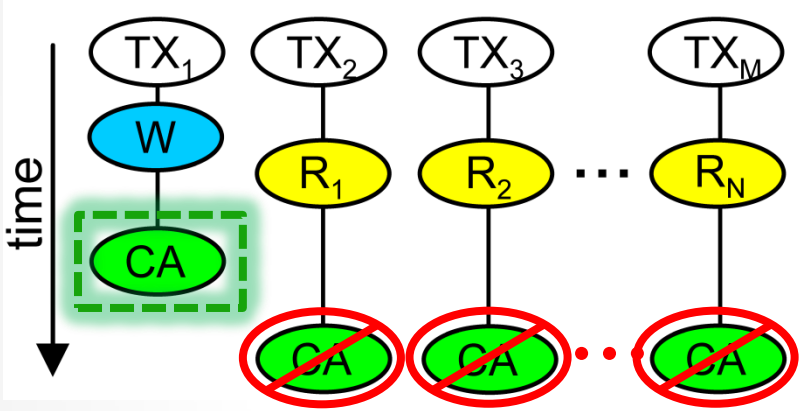


Commit to Executed Ratio: ***Commits / Executed***

Max = 1, Min = 0

Side-By-Side Analysis

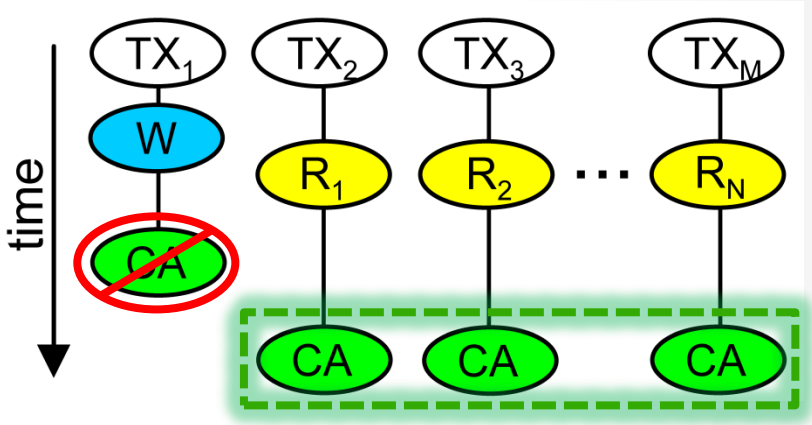
Validation



Commit / Executed: **1 / M**

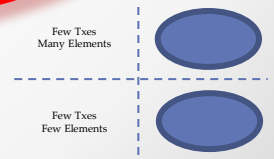
$$\lim_{M\to\infty}\left(\frac{1}{M}\right) = 0$$

Invalidation



Commit / Executed: **(M-1) / M**

$$\lim_{M\to\infty}\left(\frac{(M-1)}{M}\right) = 1$$

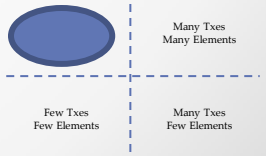


Algorithmic Growth

$$\textit{Validation} = \sum_{i=1}^{M} \sum_{j=1}^{r_i} j$$

$$\textit{Invalidation} = \sum_{i=1}^{M} \left(r_i + \sum_{j=1}^{F_i} w_i \left(s_{rj}(r_j) + s_{wj}(w_j) \right) \right)$$

$$\textit{Bloom Inval} = \sum_{i=1}^{M} (r_i + (2kw * Fi))$$

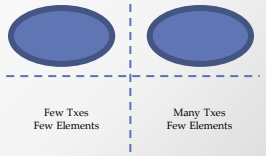


Efficient Read-Only Transactions

$$\textit{Validation Read-Only} = \sum_{i=1}^{M} \sum_{j=1}^{r_i} j$$

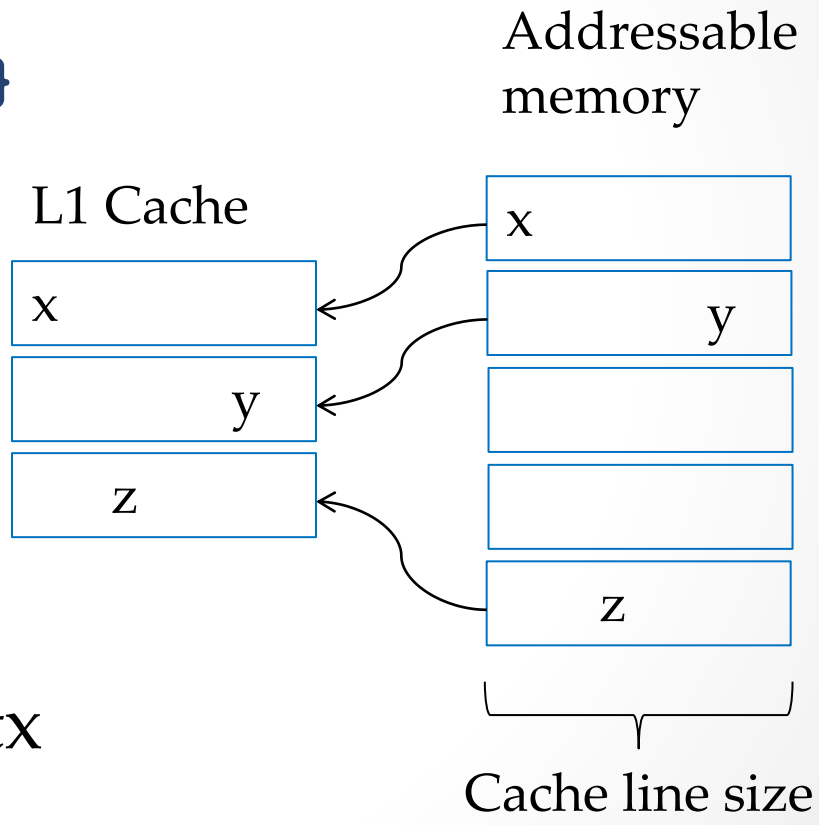
$$\textit{Invalidation} = \sum_{i=1}^{M} \left(r_i + \sum_{j=1}^{F_i} w_i\left(s_{rj}(r_j) + s_{wj}(w_j)\right) \right)$$

$$\textit{Invalidation Read-Only} = \sum_{i=1}^{M} r_i$$



Validation + Memory

```
atomic { x = y / z; }
```

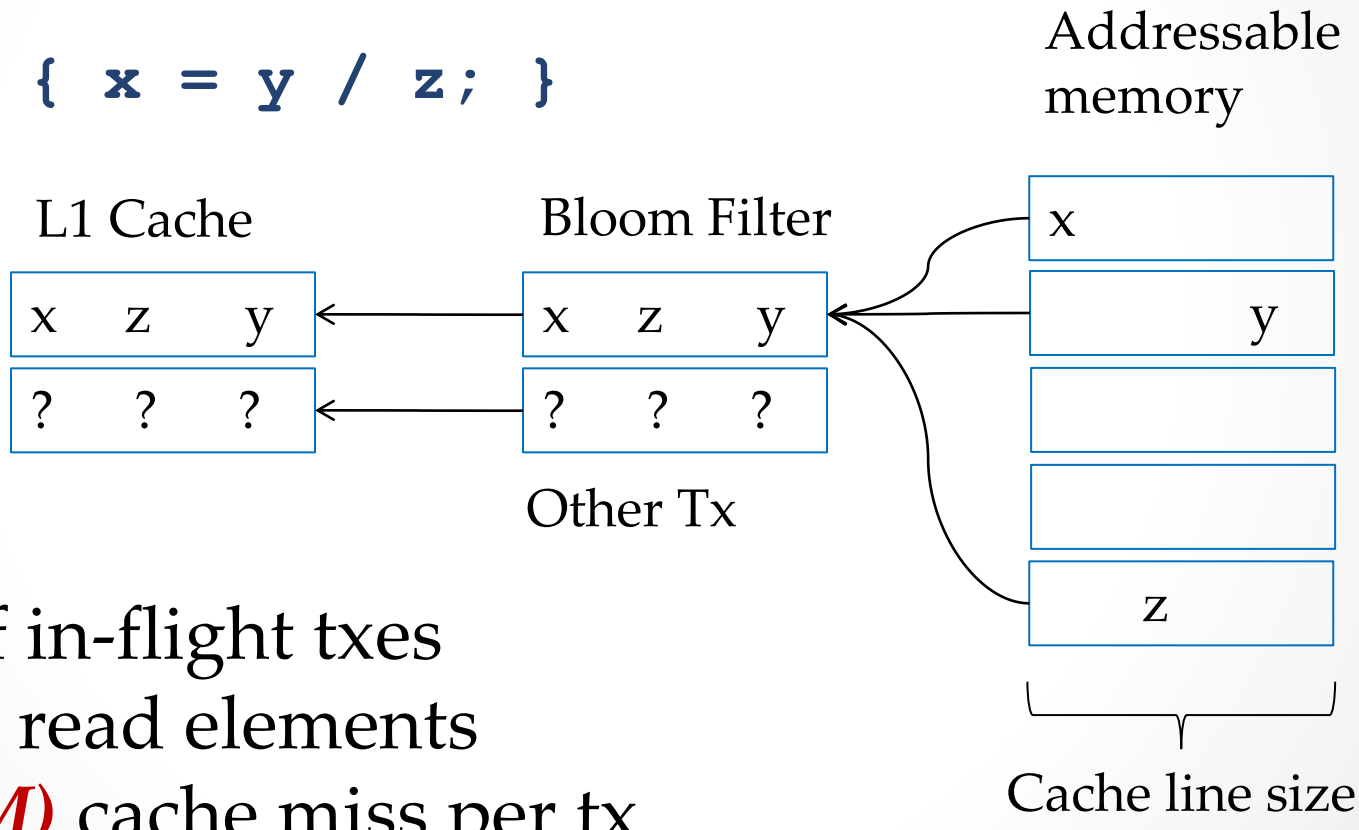


N = elements per tx

$O(N^2)$ cache misses per tx

Invalidation + Memory

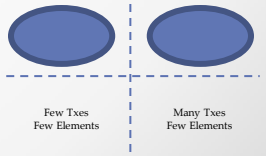
```
atomic { x = y / z; }
```



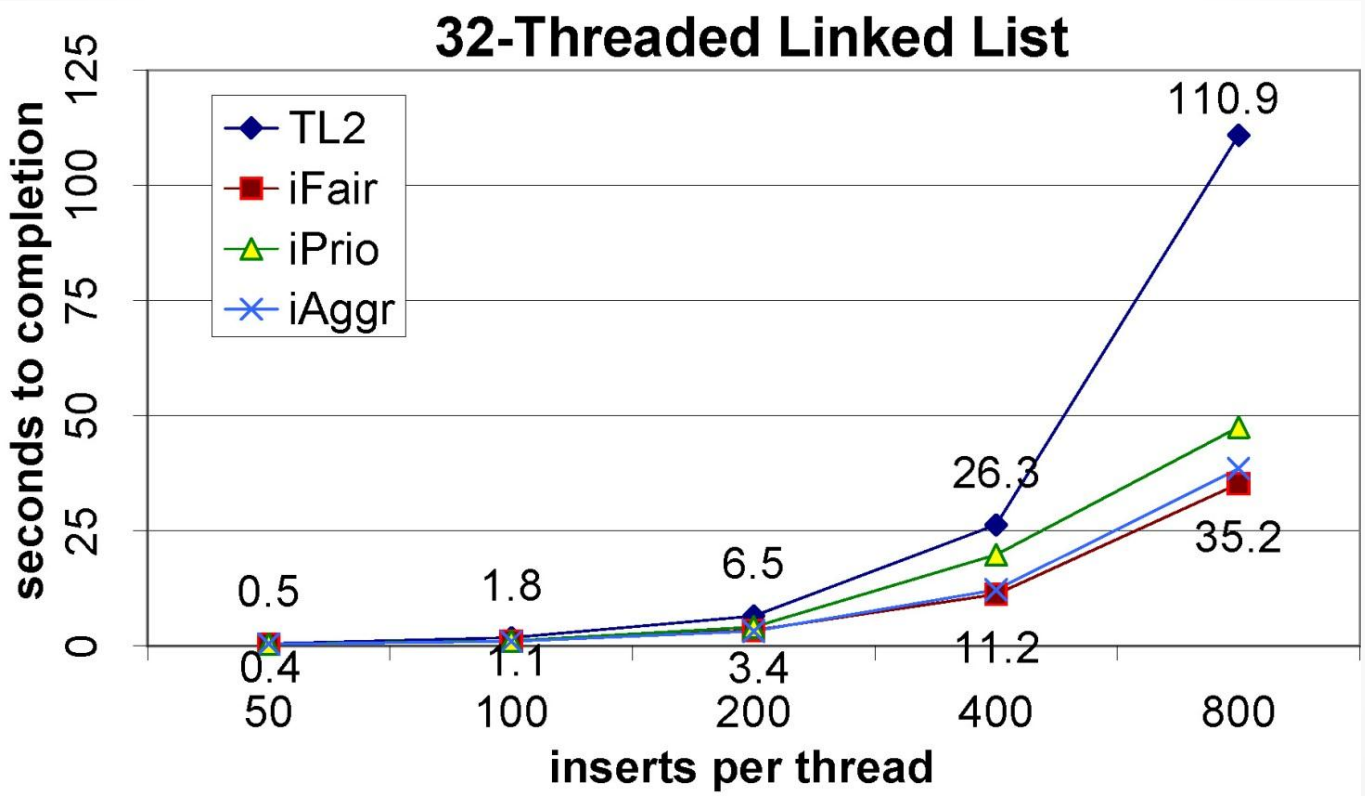
M = # of in-flight txes

N = # of read elements

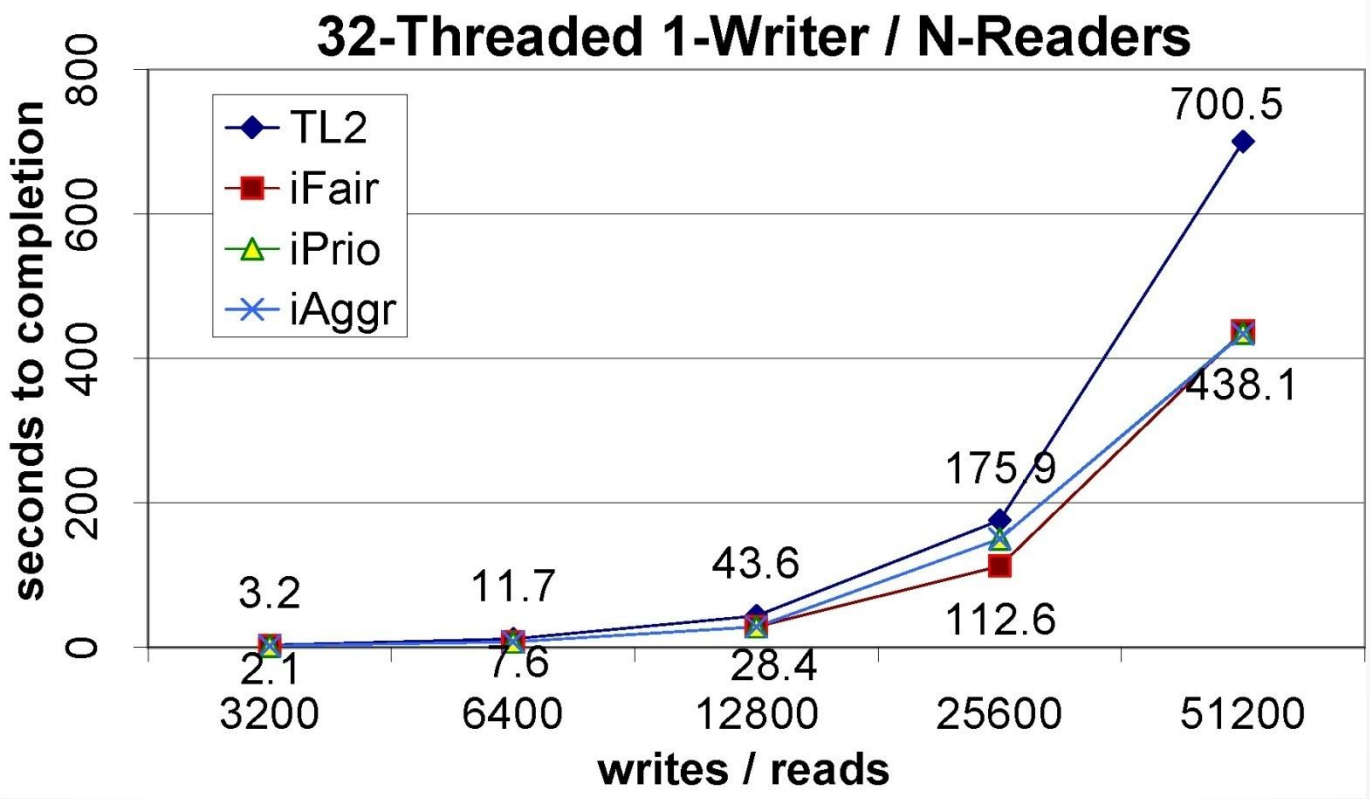
O(N + M) cache miss per tx



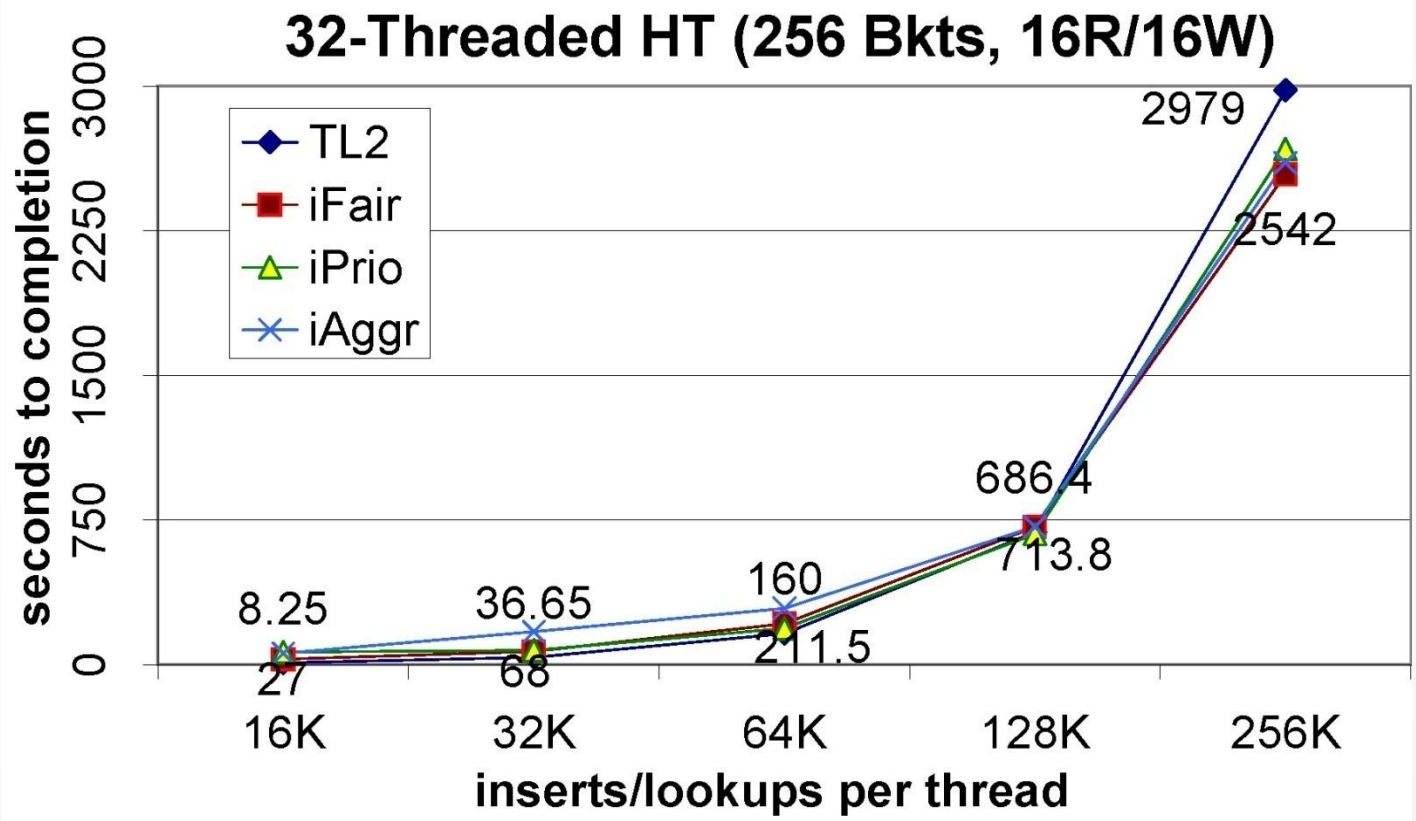
Linked List


32-Threaded Linked List
seconds to completion
0
25
50
75
100
125
TL2
iFair
iPrio
iAggr
0.5
1.8
6.5
26.3
110.9
0.4
1.1
3.4
11.2
35.2
50
100
200
400
800
inserts per thread

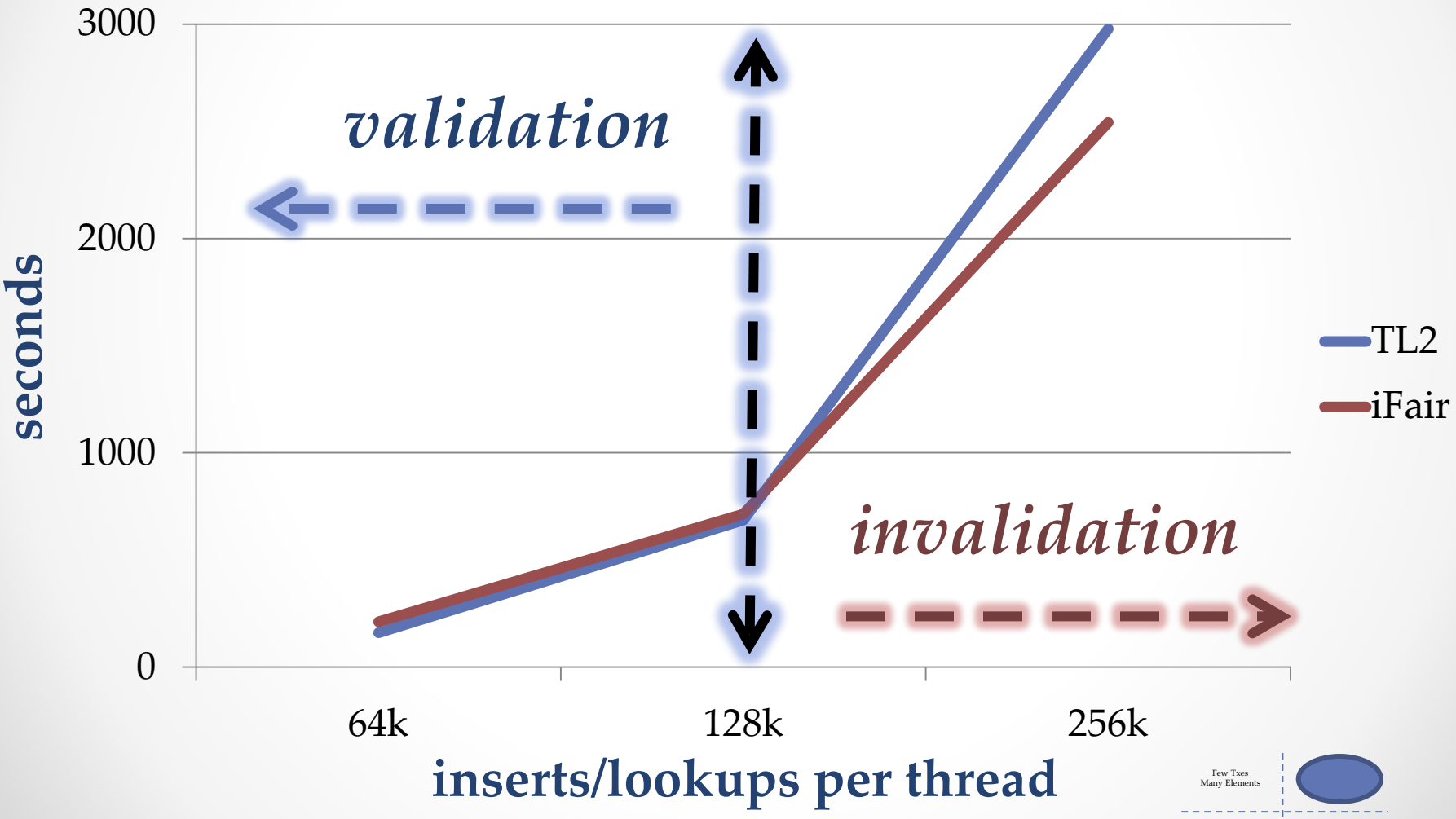
1-Writer / N-Readers



Hash Table

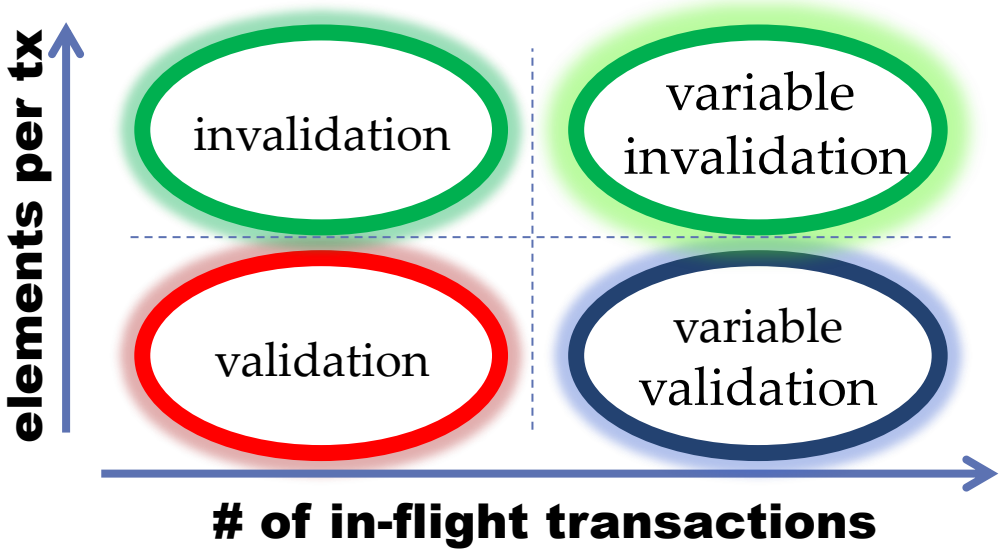

32-Threaded HT (256 Bkts, 16R/16W)
seconds to completion
0
750
1500
2250
3000
TL2
iFair
iPrio
iAggr
2979
2542
686.4
713.8
160
211.5
36.65
68
8.25
27
16K
32K
64K
128K
256K
inserts/lookups per thread

Zoomed Hash Table



Conclusion

- InvalSTM (TBoost.STM) competitive with TL2
- Invalidation (InvalSTM) can be efficient



- Next up
  - Proof of correctness for Full Invalidation
  - InvalSTM + STAMP
- Special thanks to Spear and Herlihy

Questions?

Justin E. Gottschlich
gottschl@colorado.edu
http://eces.colorado.edu/~gottschl/

Panel Questions

What is the most important TM problem to solve in the next 1-2 years?

Panel Questions

What is the most important TM problem to solve in the next 5-10 years?

Panel Questions

Is TM on track to becoming a real parallel programming solution?

If so, when will it be realized?

If not, what are we doing wrong and what do we need to do to fix it?

Panel Questions

In what types of environments (systems or programs) do you see TM being most useful?

Why?

Panel Questions

Where is TM least useful?

Why?

Panel Questions

What will parallel programs look like in the future?

Panel Questions

What should a transaction do when an exception occurs within it?

Abort? Commit?