An Efficient Software Transactional Memory Using Commit-Time Invalidation

CGO

Justin E. Gottschlich, Manish Vachharajani, and Jeremy G. Siek

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

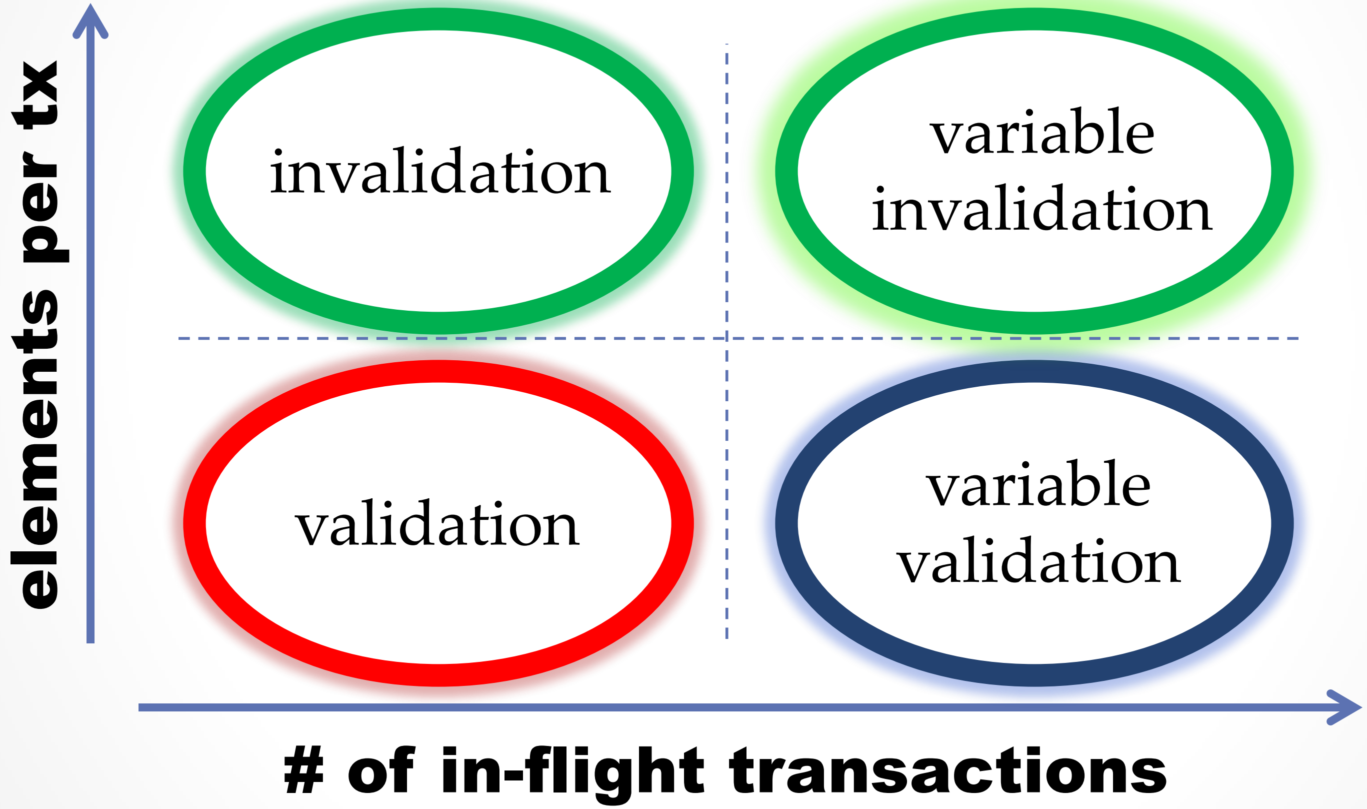
- **Validation** (T2)
  - ***Analyze the Past***
    - Version # is same

- **Invalidation** (T1)
  - ***Analyze the Future***
    - *T2.valid = false*

Opacity

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

Validation Vs. Invalidation



elements per tx

invalidation

variable invalidation

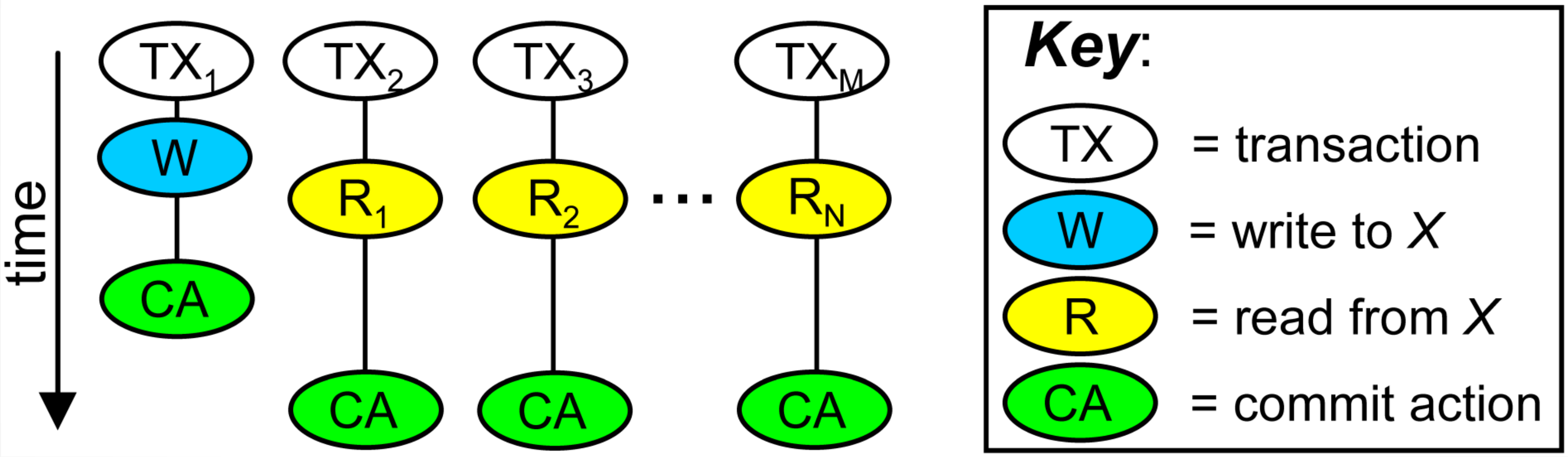
validation

variable validation

# of in-flight transactions

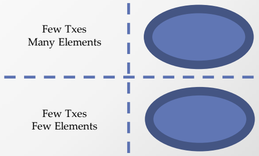
Contending + Concurrent Workload

1-Writer, N-Reader



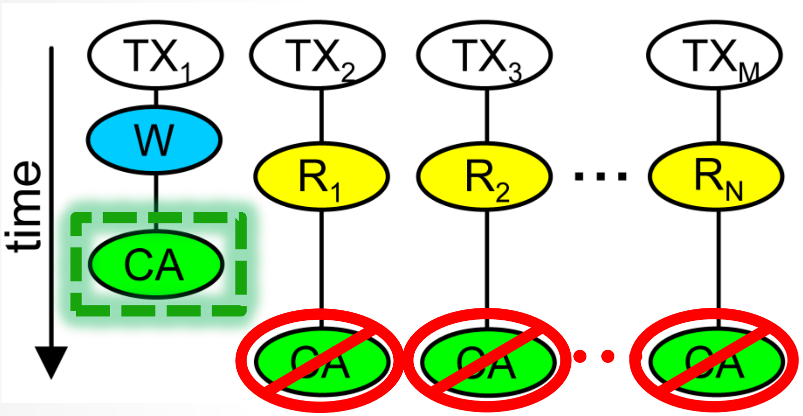
Commit to Executed Ratio: *Commits / Executed*

Max = 1, Min = 0

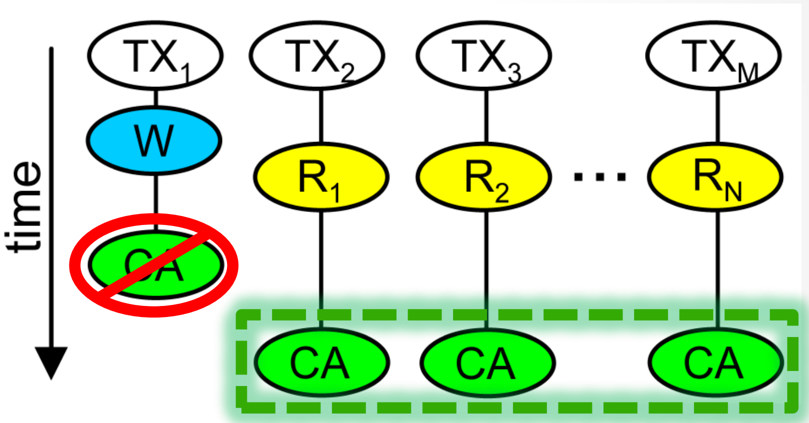


Side-By-Side Analysis

Validation



Invalidation

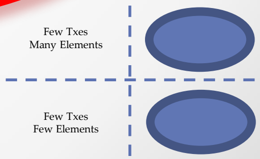


Commit / Executed: **1 / M**

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

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

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

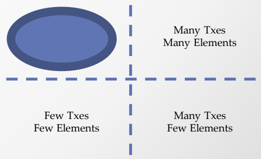


Algorithmic Growth

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

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

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

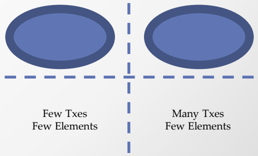


Efficient Read-Only Transactions

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

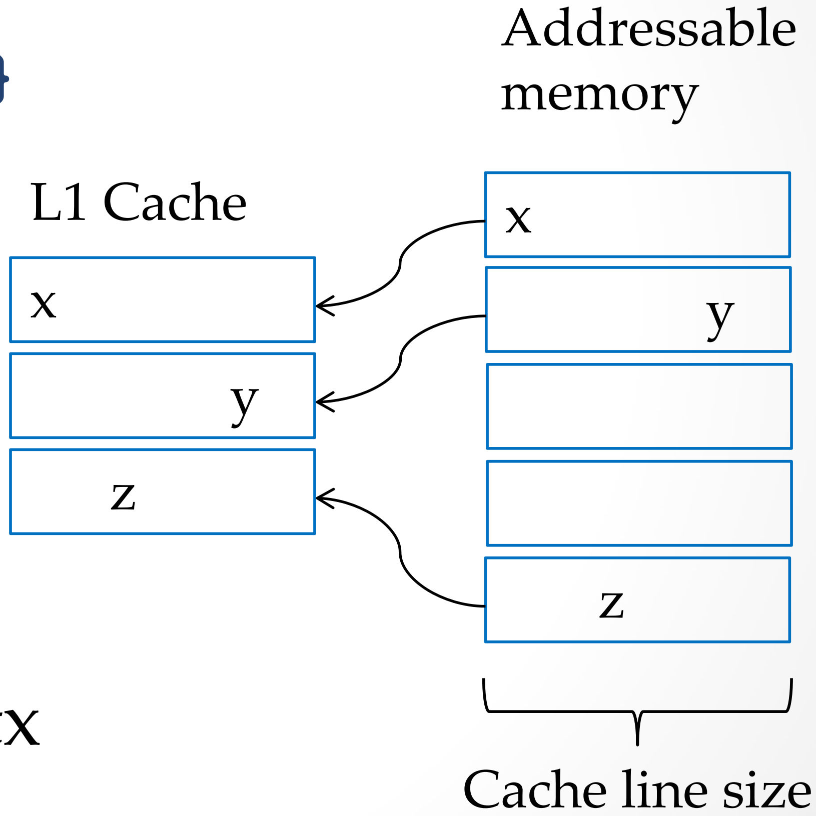
$$\textit{Invalidation} = \sum_{i=1}^{M} \left(r_i + \sum_{j=1}^{F_i} w_i \left(s_{rj}(r_j) + s_{wj}(w_j) \right) \right)$$

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



Validation + Memory

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

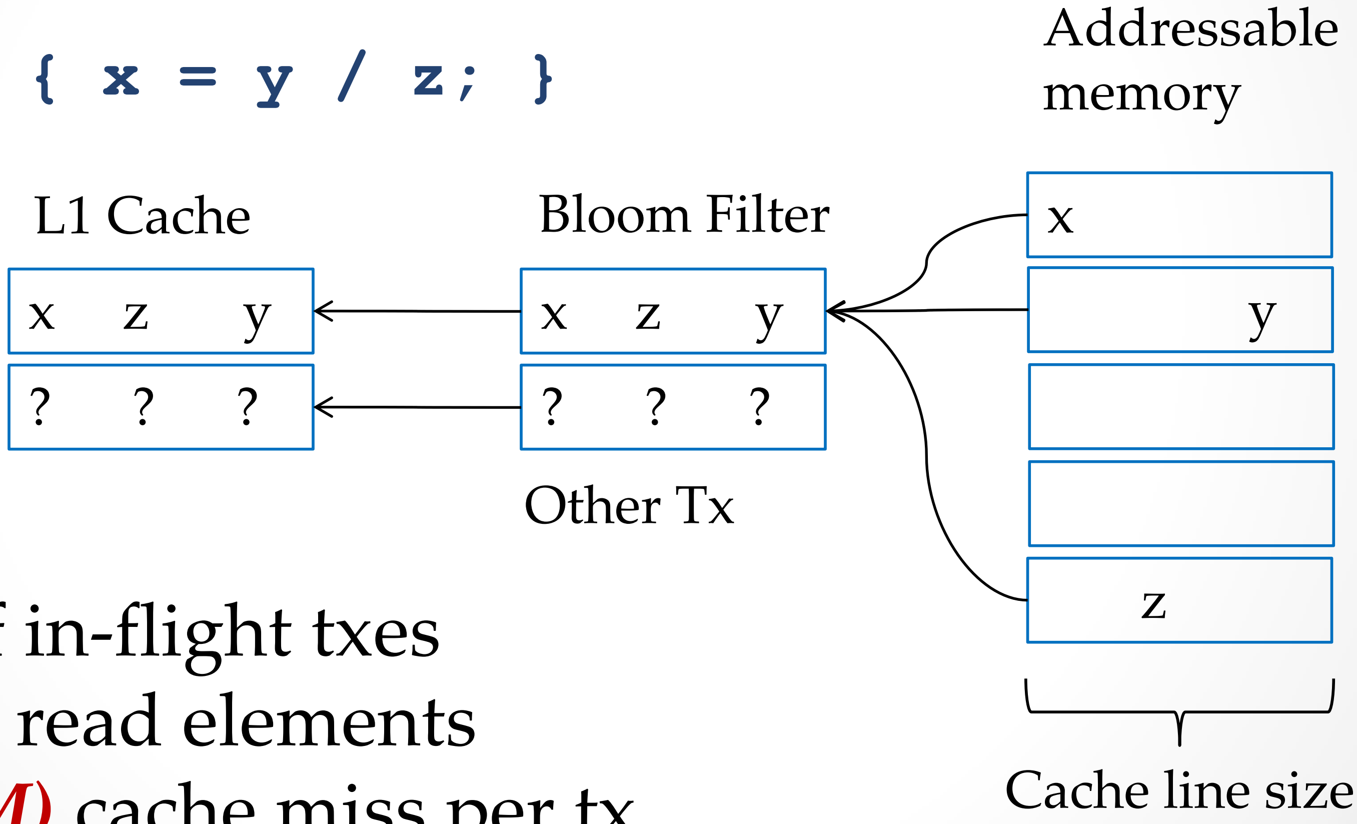


N = Elements per tx

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

Invalidation + Memory

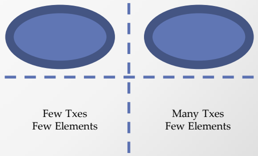
```
atomic { x = y / z; }
```



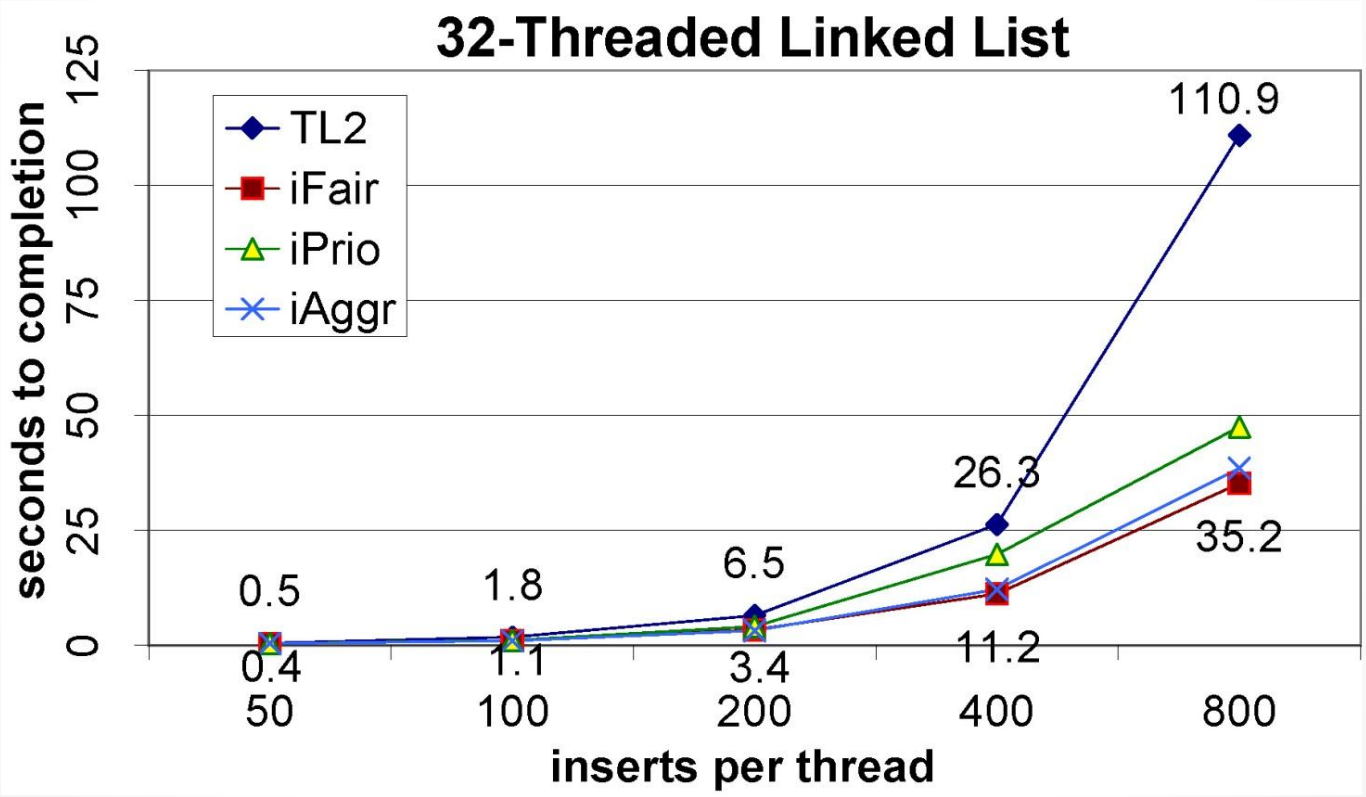
M = # of in-flight txes

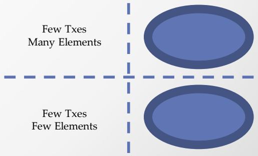
N = # of read elements

O(N + M) cache miss per tx



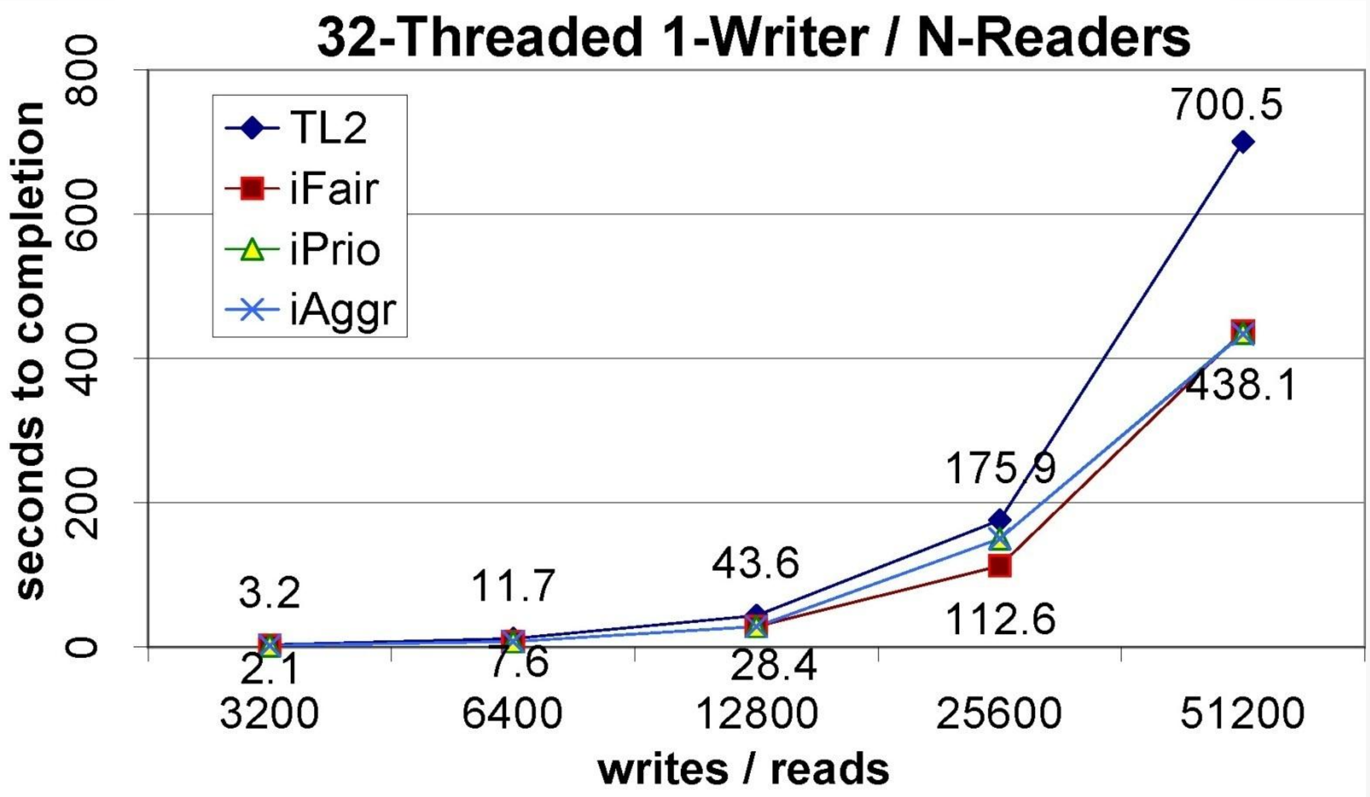
Linked List

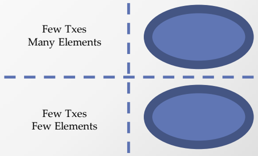



Few Txes
Many Elements

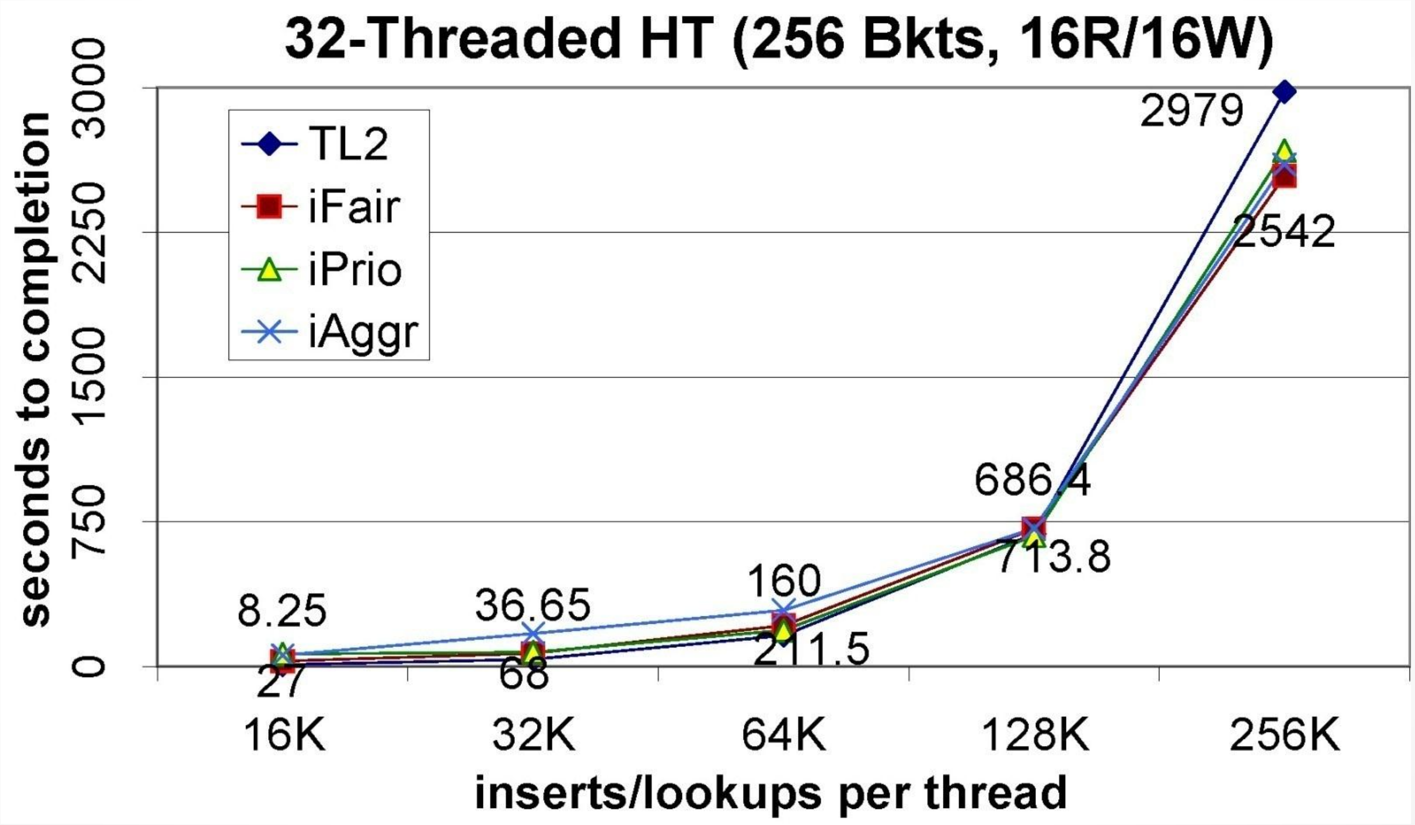
Few Txes
Few Elements

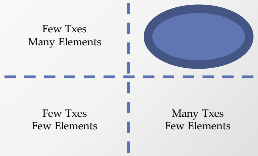
1-Writer / N-Readers





Hash Table

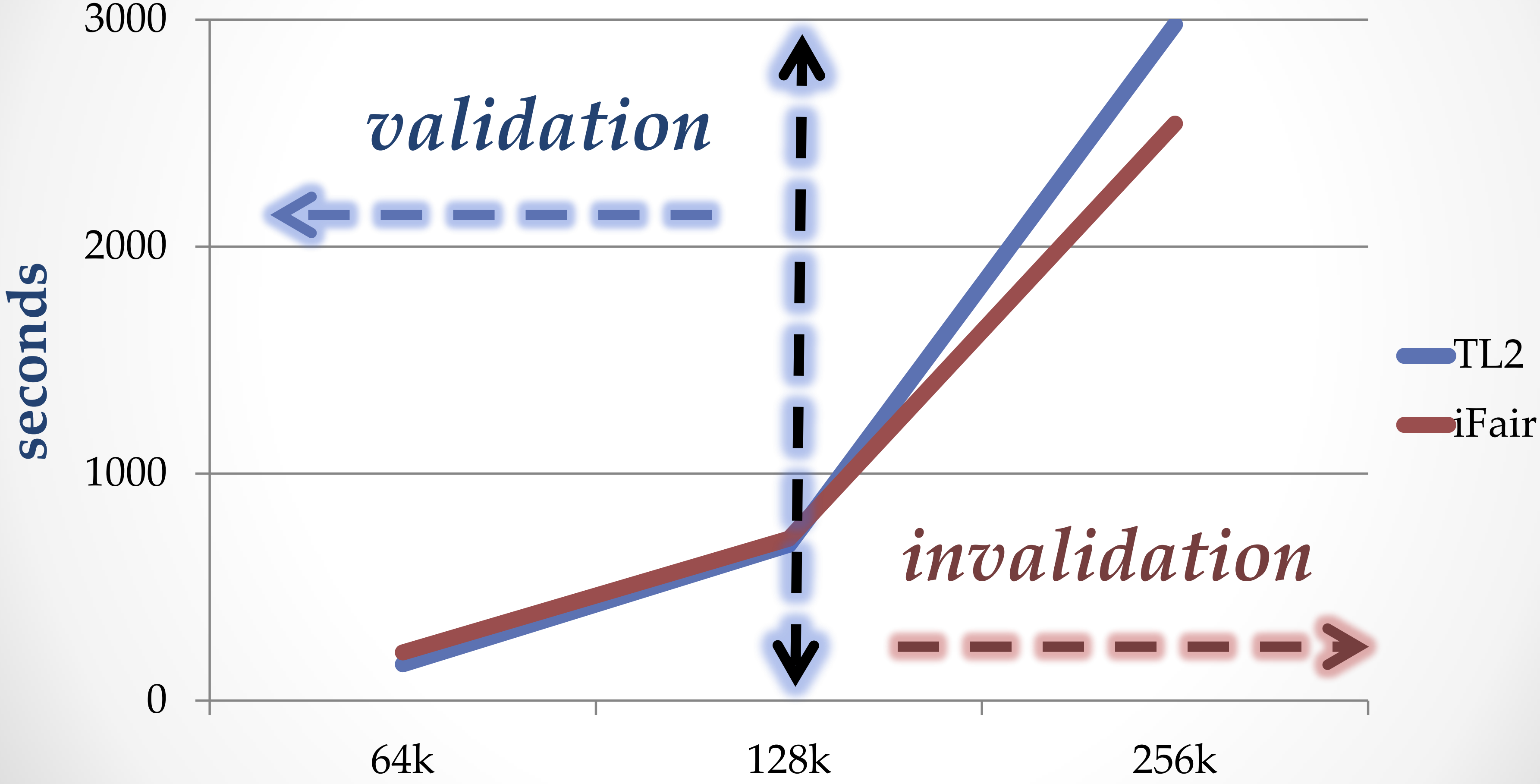


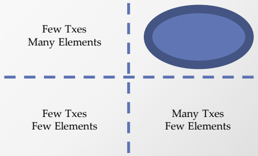

Few Txes
Many Elements

Few Txes
Few Elements

Many Txes
Few Elements

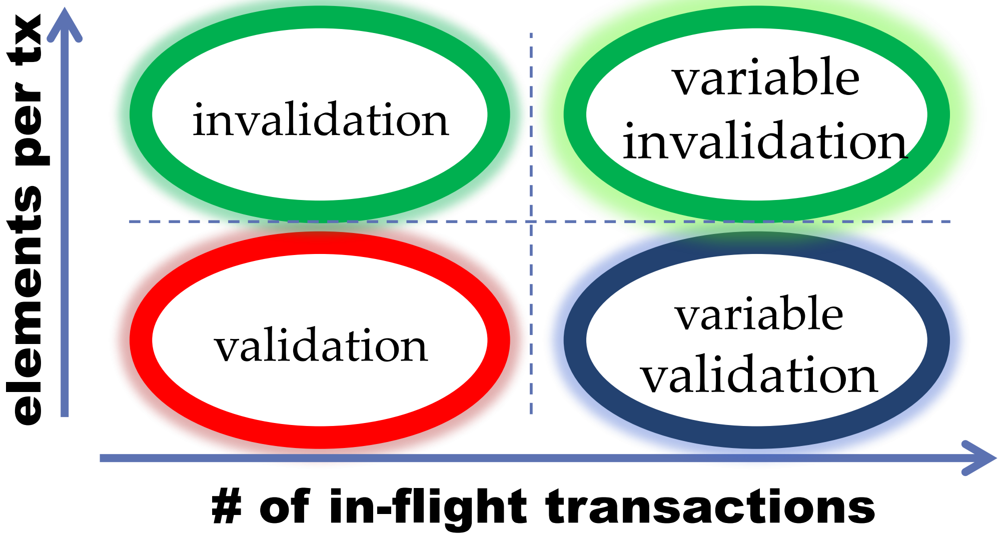
Zoomed Hash Table





Conclusion

- 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/