STM in the Small: Trading Generality for Performance in Software Transactional Memory

Aleksandar Dragojević

Microsoft Research

Microsoft

Our motivation

- Concurrent data structures
- Hash-tables, skip-lists
  - In-memory database indices[2]
  - Key-value stores

[2] *Larson et al.* "High-performance concurrency control mechanisms for main-memory databases" VLDB Dec. 2011

Microsoft Research

Software Transactional Memory

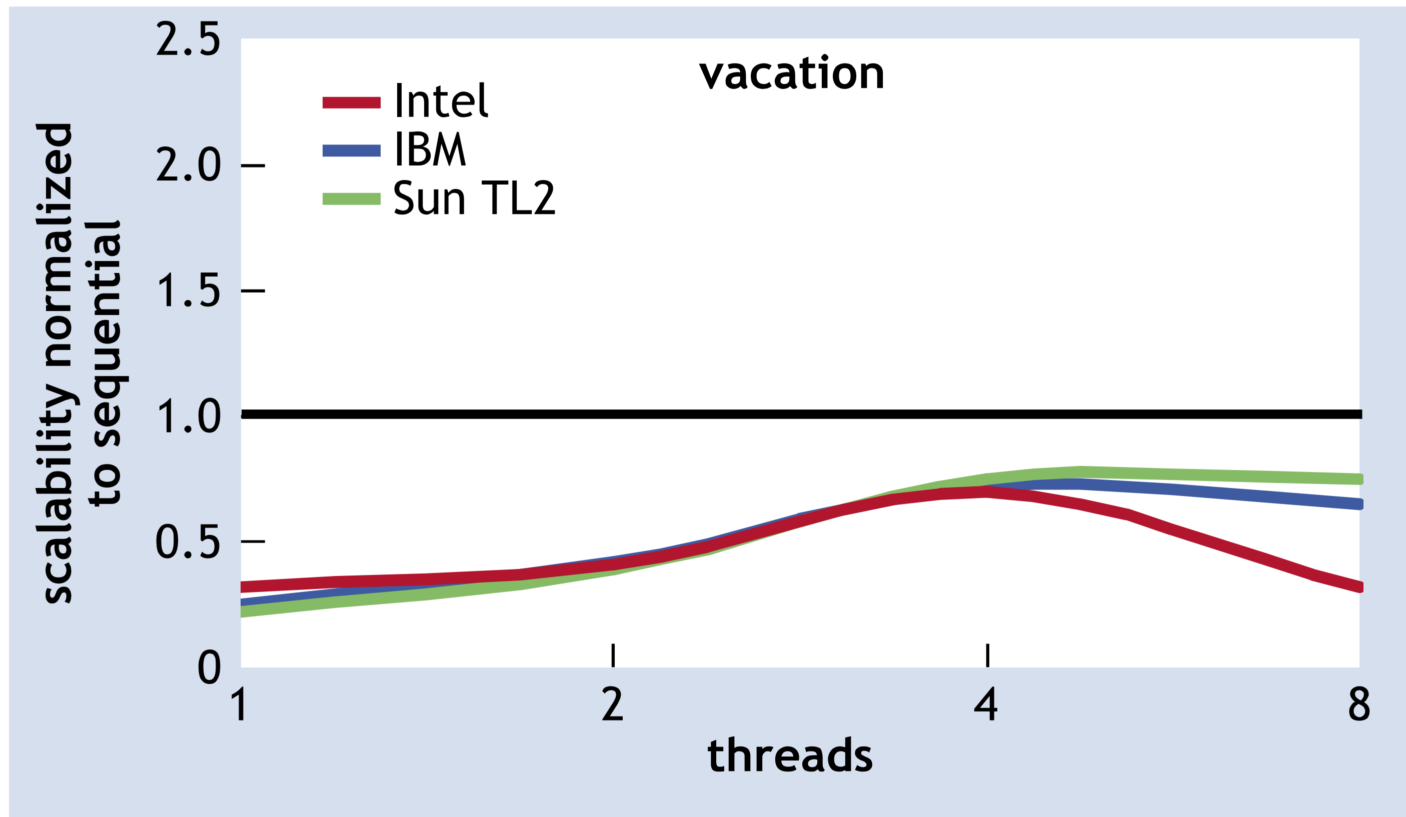
- Simplifies data structures
  – Ensures correct implementations
- Enables use of complex data structures
  – More complex than when using CAS directly

Microsoft Research

What is STM performance like?

Microsoft Research

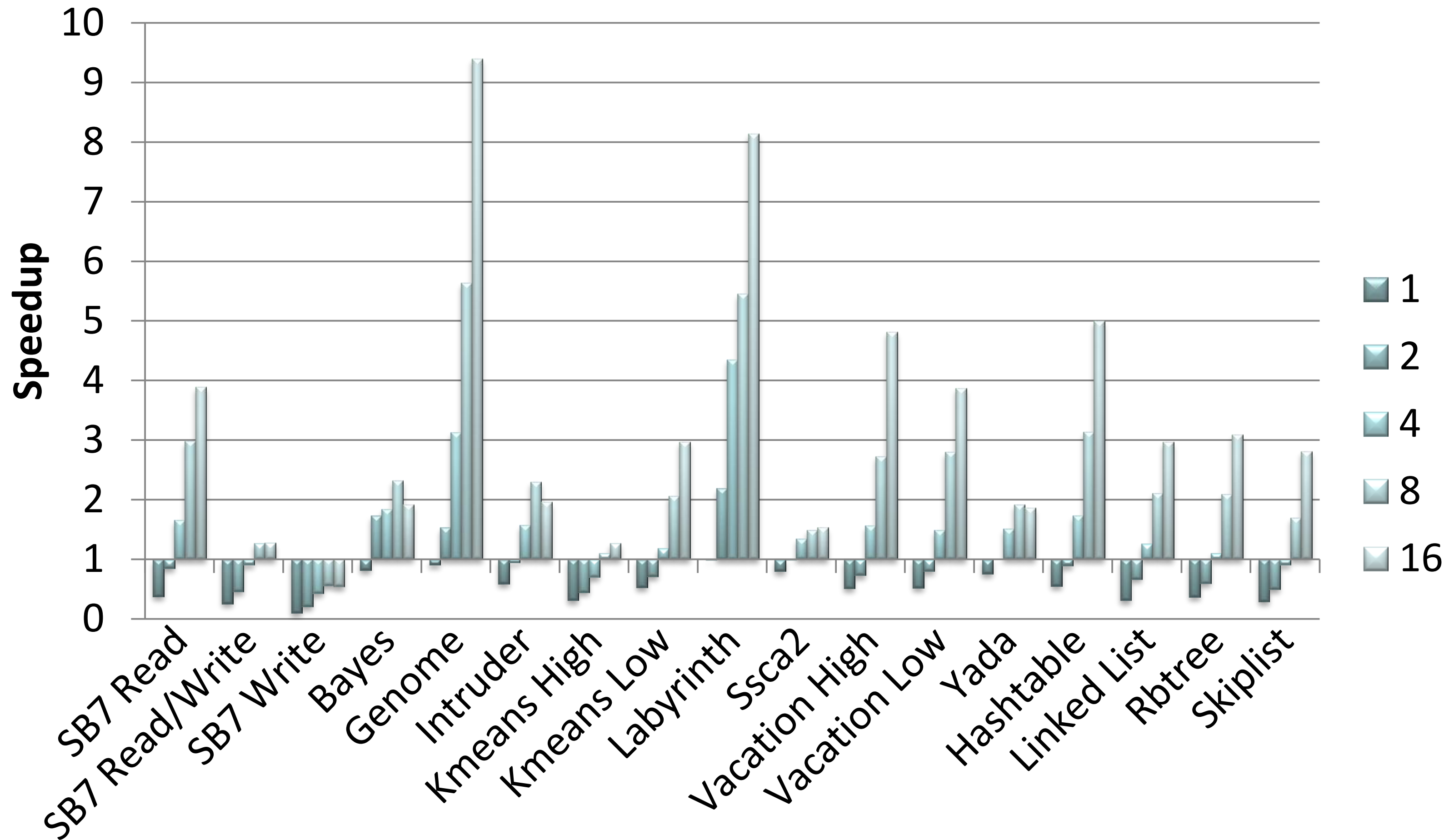
What is STM performance like?



Graph from: Cascaval et al. "Software transactional memory: why is it only a research toy" ACMQ September 2008

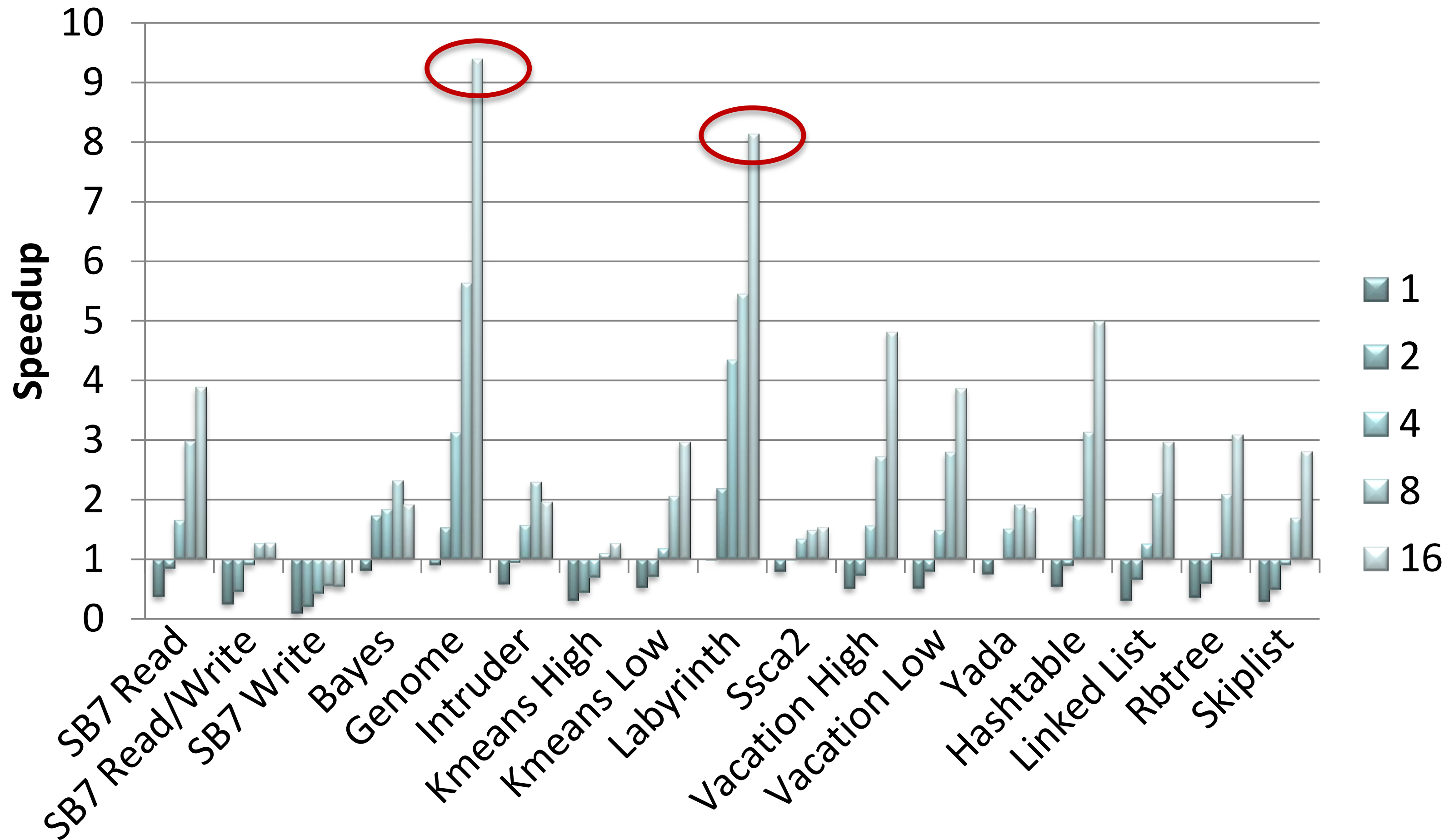
Microsoft Research

What is STM performance like?



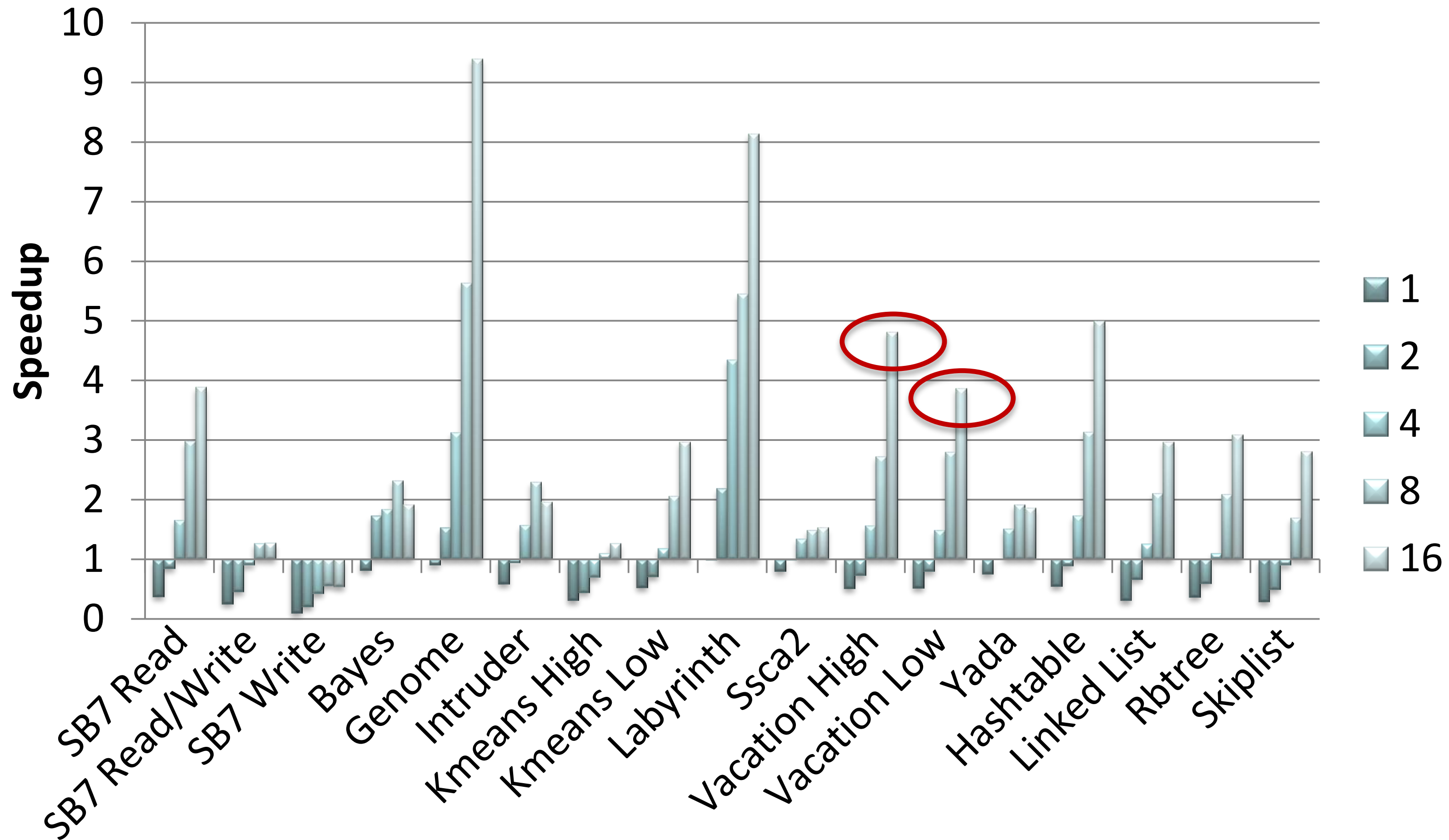
Microsoft Research

What is STM performance like?



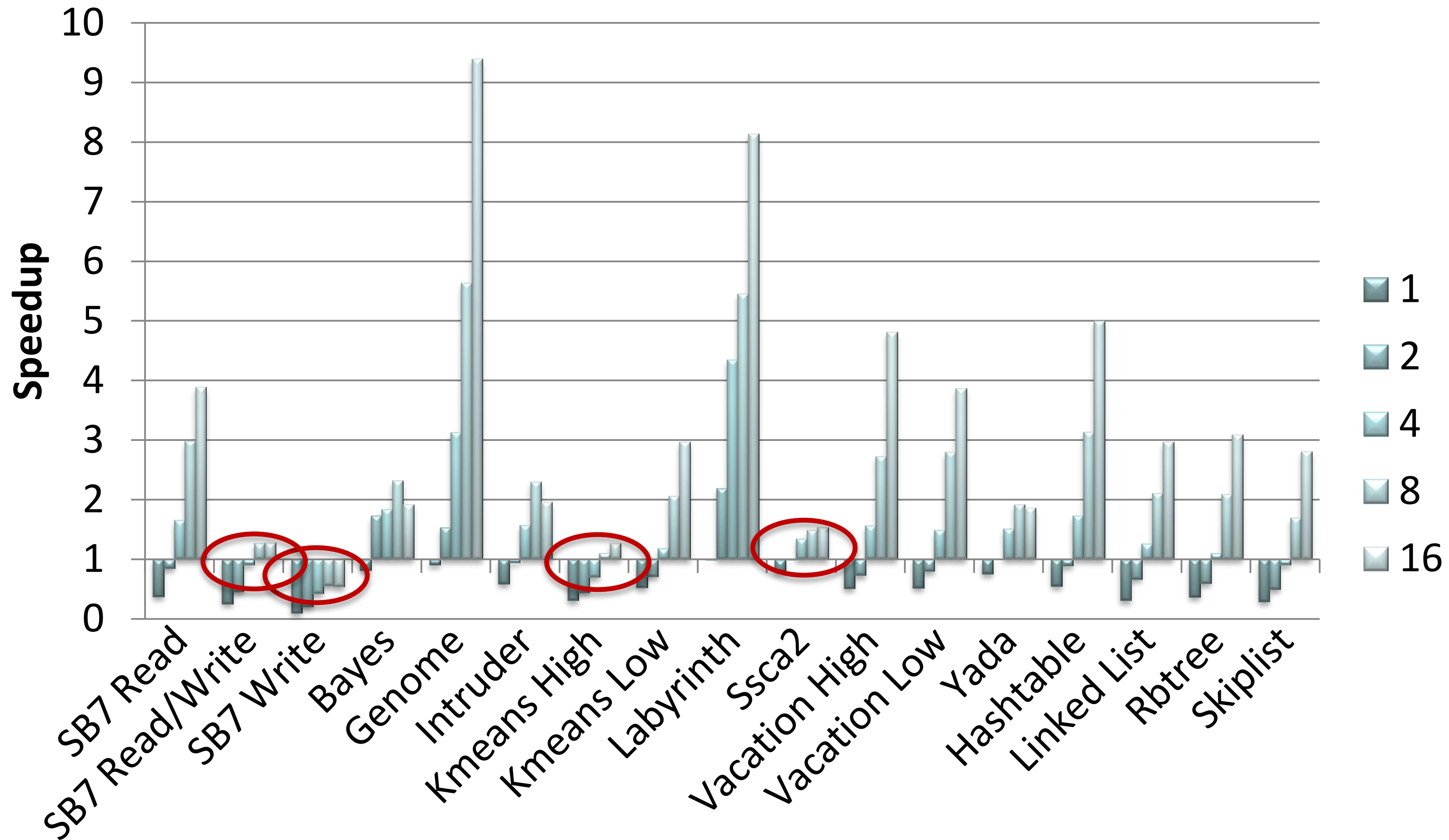
Microsoft Research

What is STM performance like?



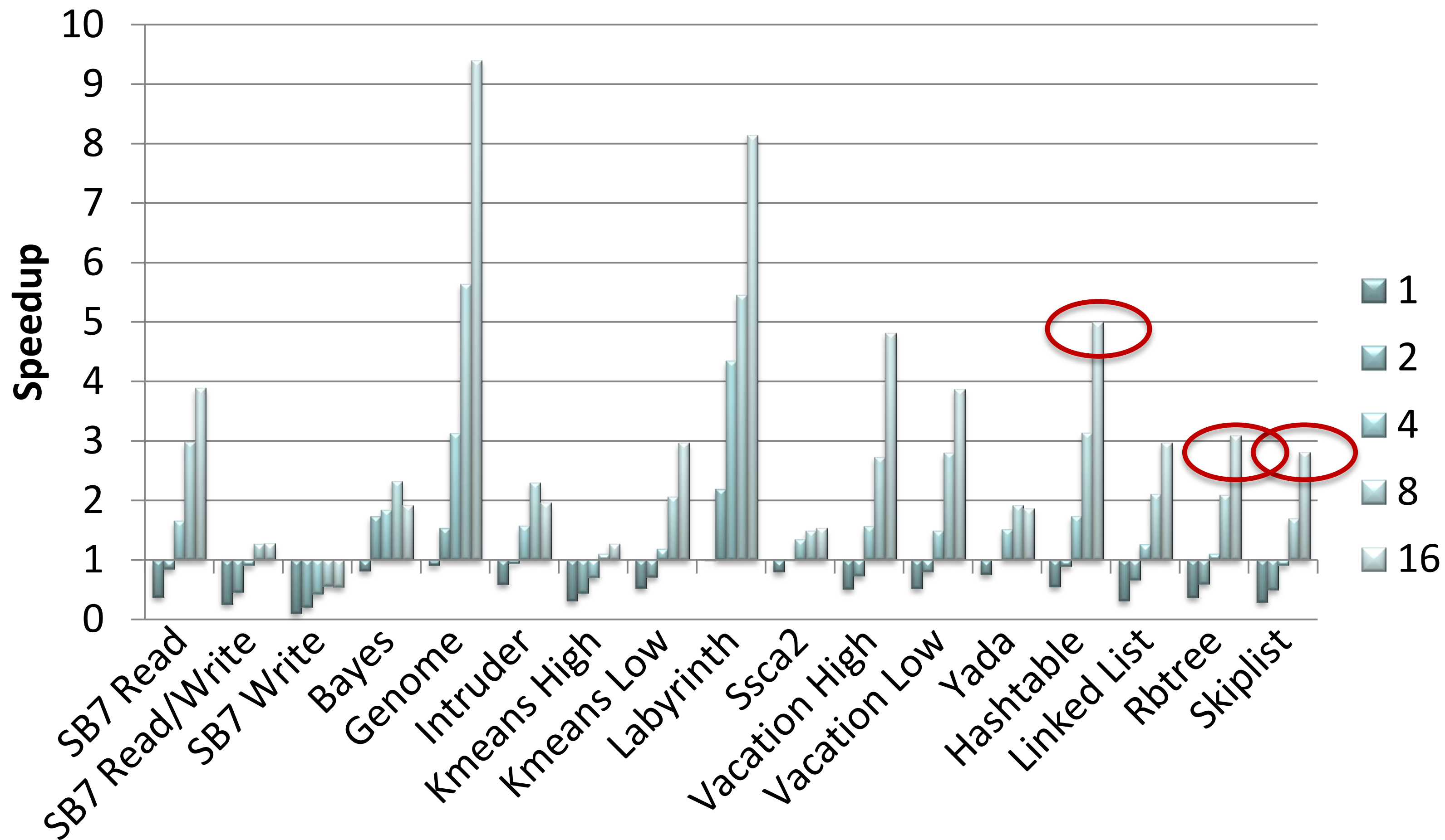
Microsoft Research

What is STM performance like?



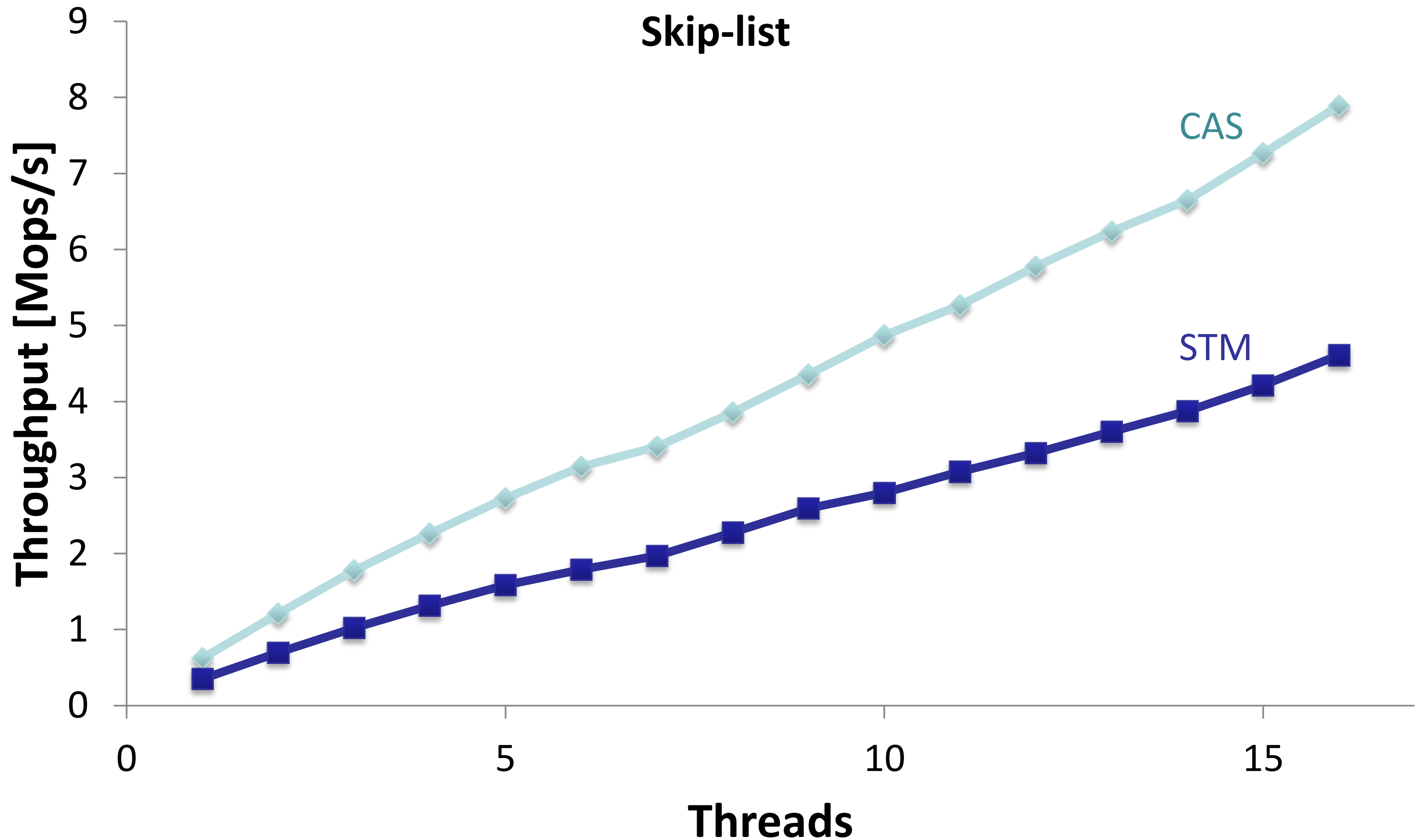
Microsoft Research

What is STM performance like?



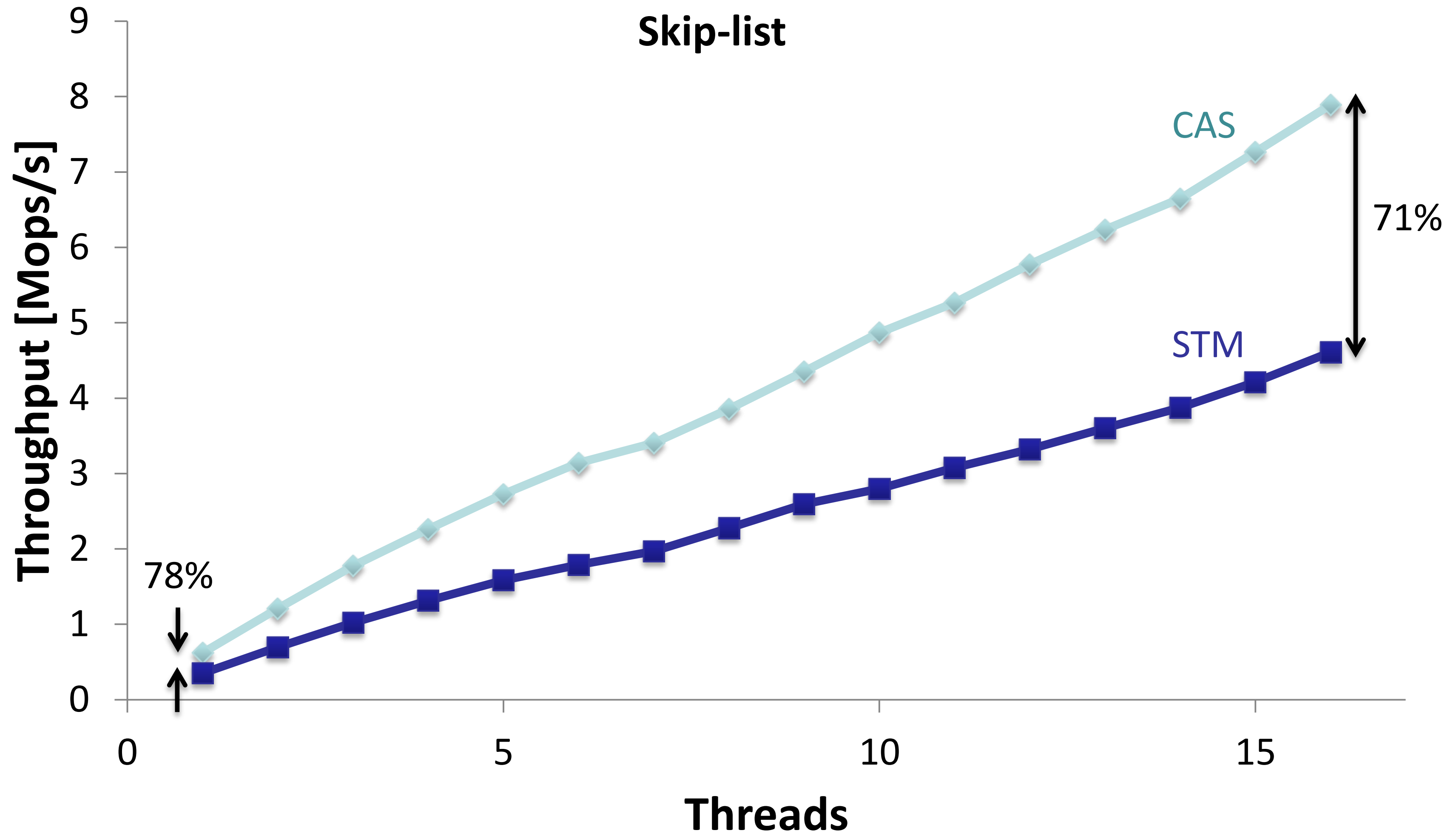
Microsoft Research

What is STM performance like?



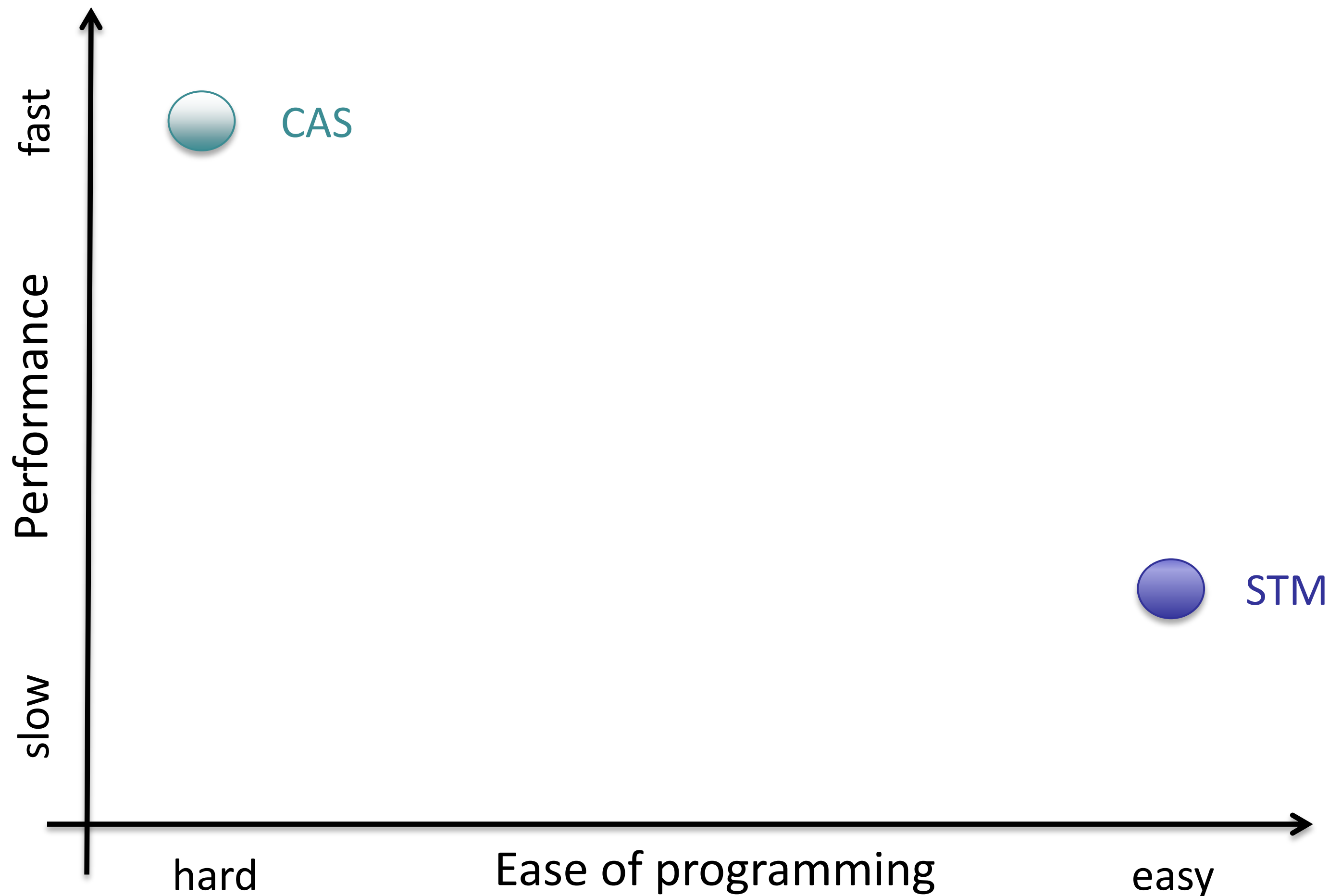
Microsoft Research

What is STM performance like?



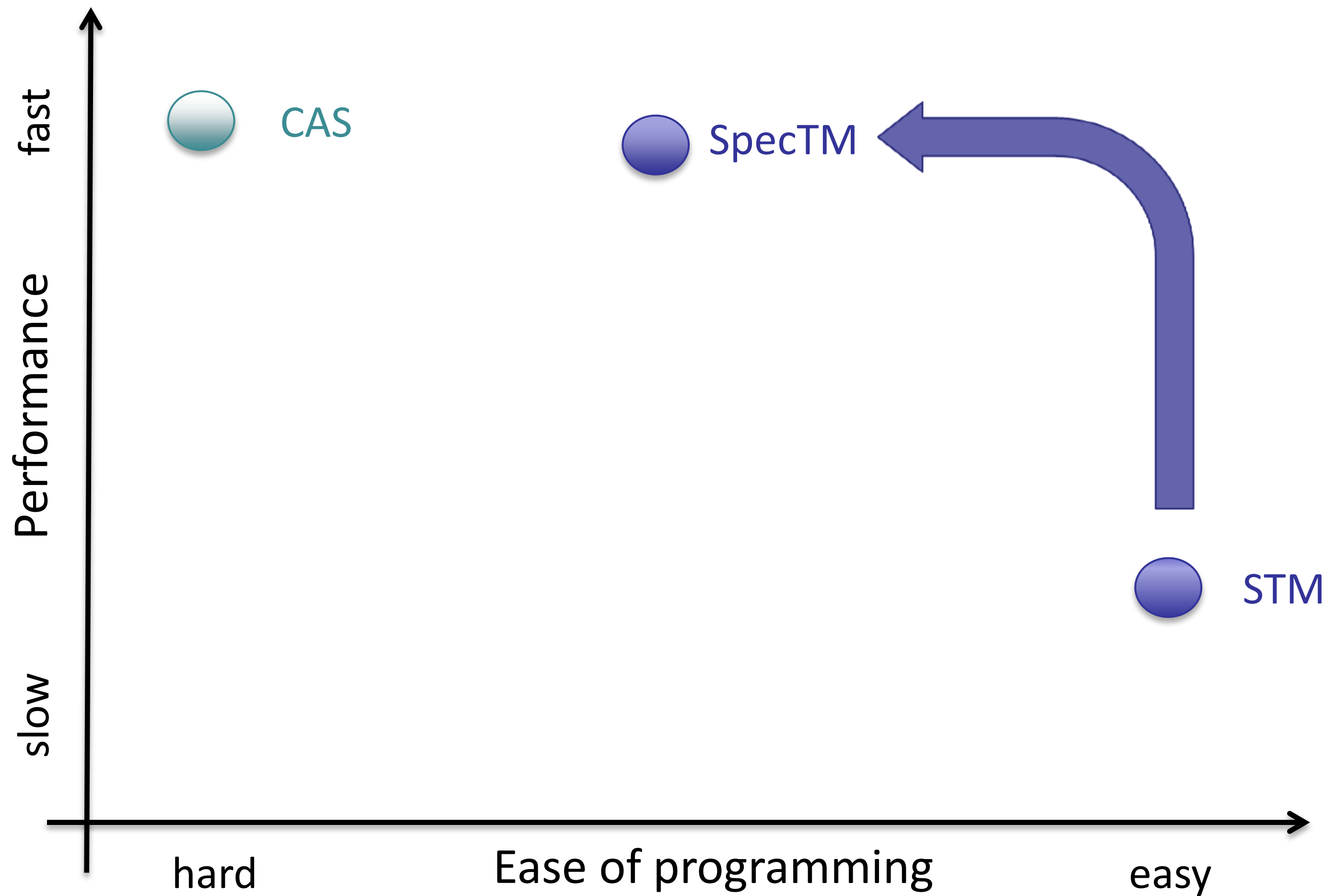
Microsoft Research

SpecTM: Specialized STM for Concurrent Data Structures



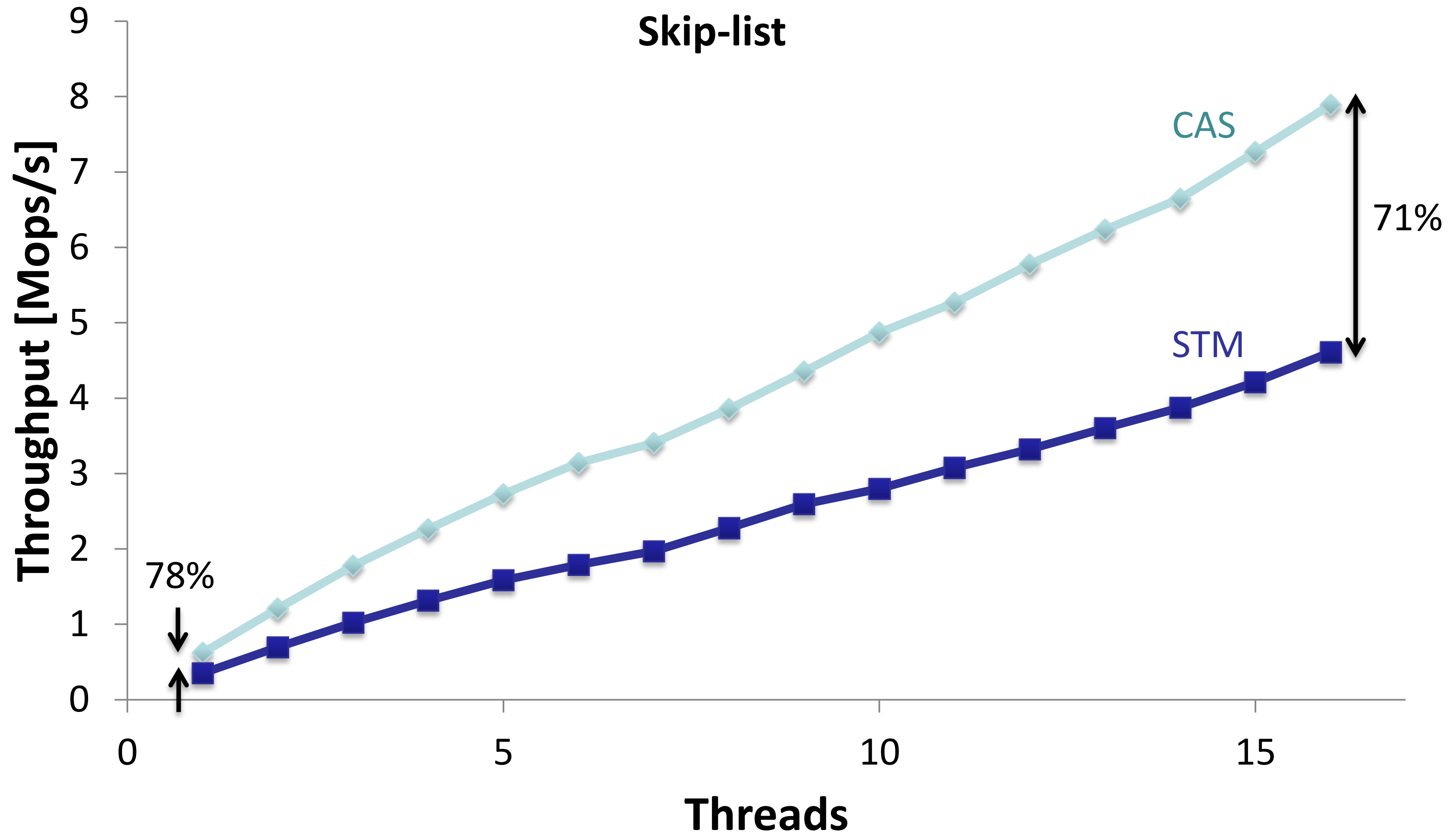
Microsoft Research

SpecTM: Specialized STM for Concurrent Data Structures



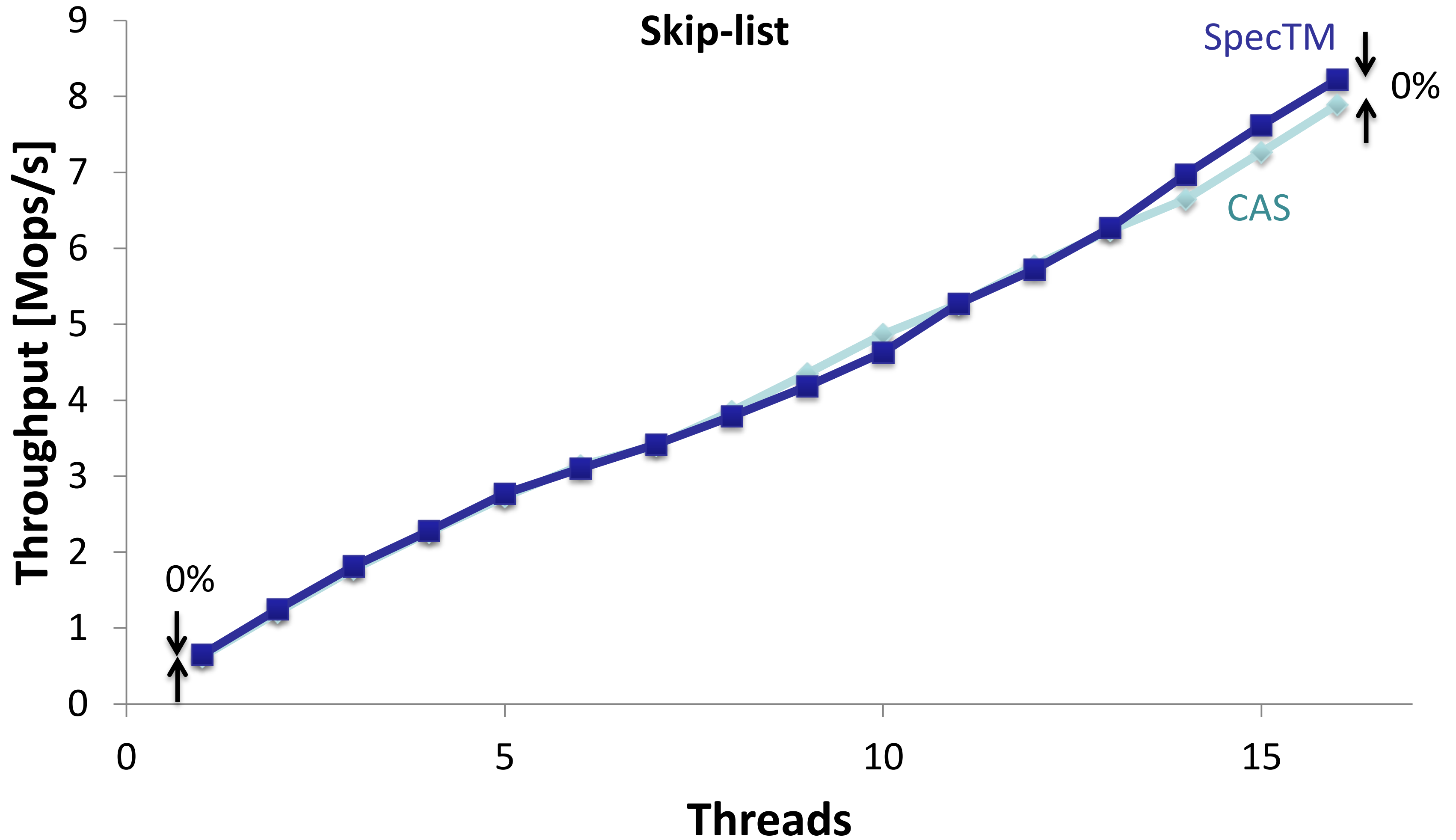
Microsoft Research

This talk



Microsoft Research

This talk



Microsoft Research

Overview

- STM overheads
- SpecTM
  - Short-transaction API
  - Collocating data and meta-data
  - In-word meta-data
- Evaluation

Microsoft Research

Overview

- STM overheads
- SpecTM
  - Short-transaction API
  - Collocating data and meta-data
  - In-word meta-data
- Evaluation

Microsoft Research

STM overheads

- Book-keeping done in software
- Memory accesses become STM calls
- Significant overheads[3]
  - With a single thread 50% overheads
  - Requires 4 threads to outperform sequential code in 75% of cases

[3] *Dragojević et al.* “Why STM Can Be more than a Research Toy” CACM Apr. 2011

Microsoft Research

STM API

```
void StartTx();

void CommitTx();

word_t ReadWord(word_t *addr);

void WriteWord(word_t *addr, word_t val);
```

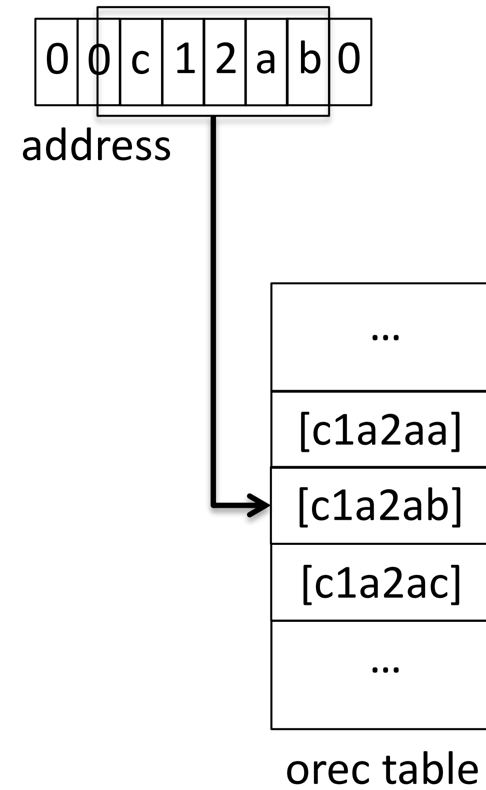
STM read

map(addr)

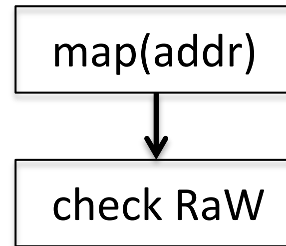
Microsoft Research

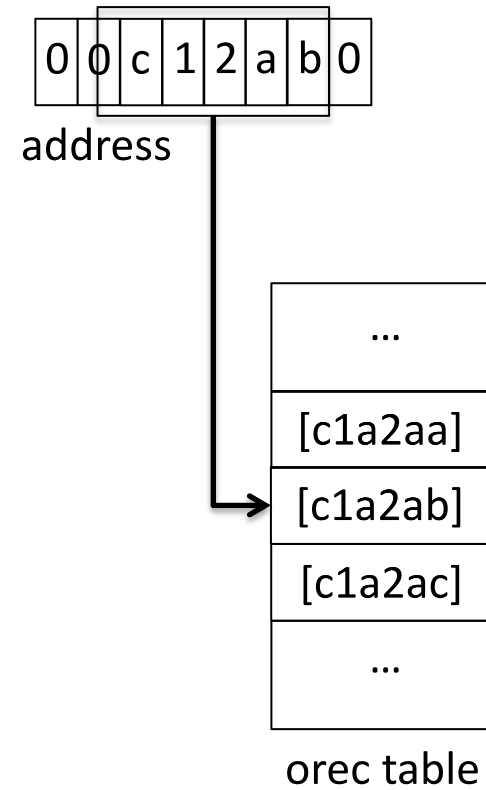
STM read

map(addr)



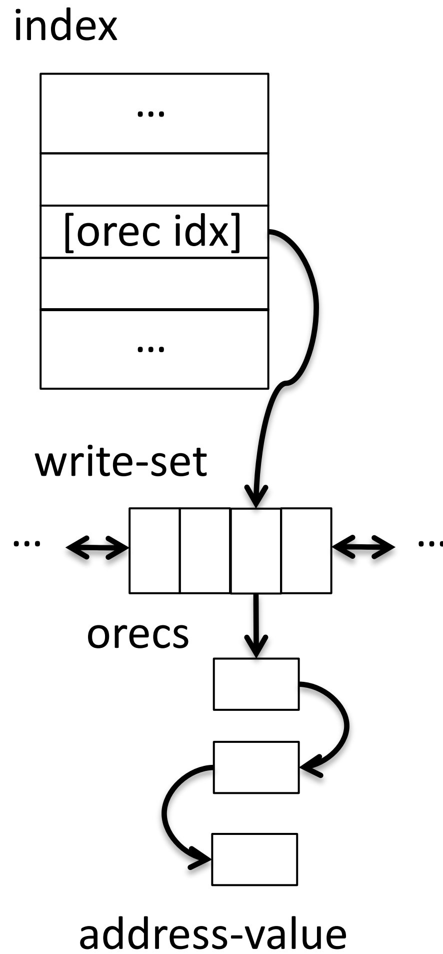
STM read

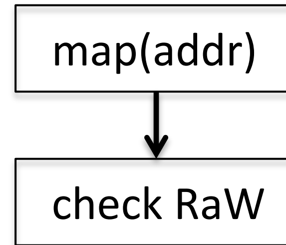


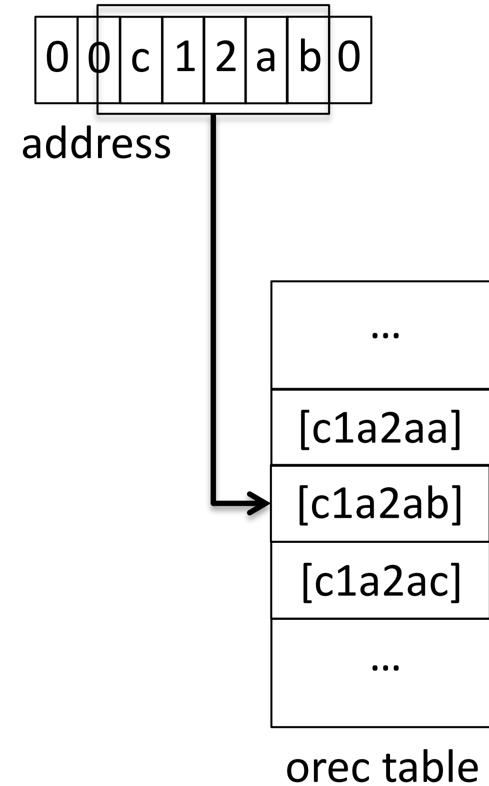


Microsoft Research

STM read

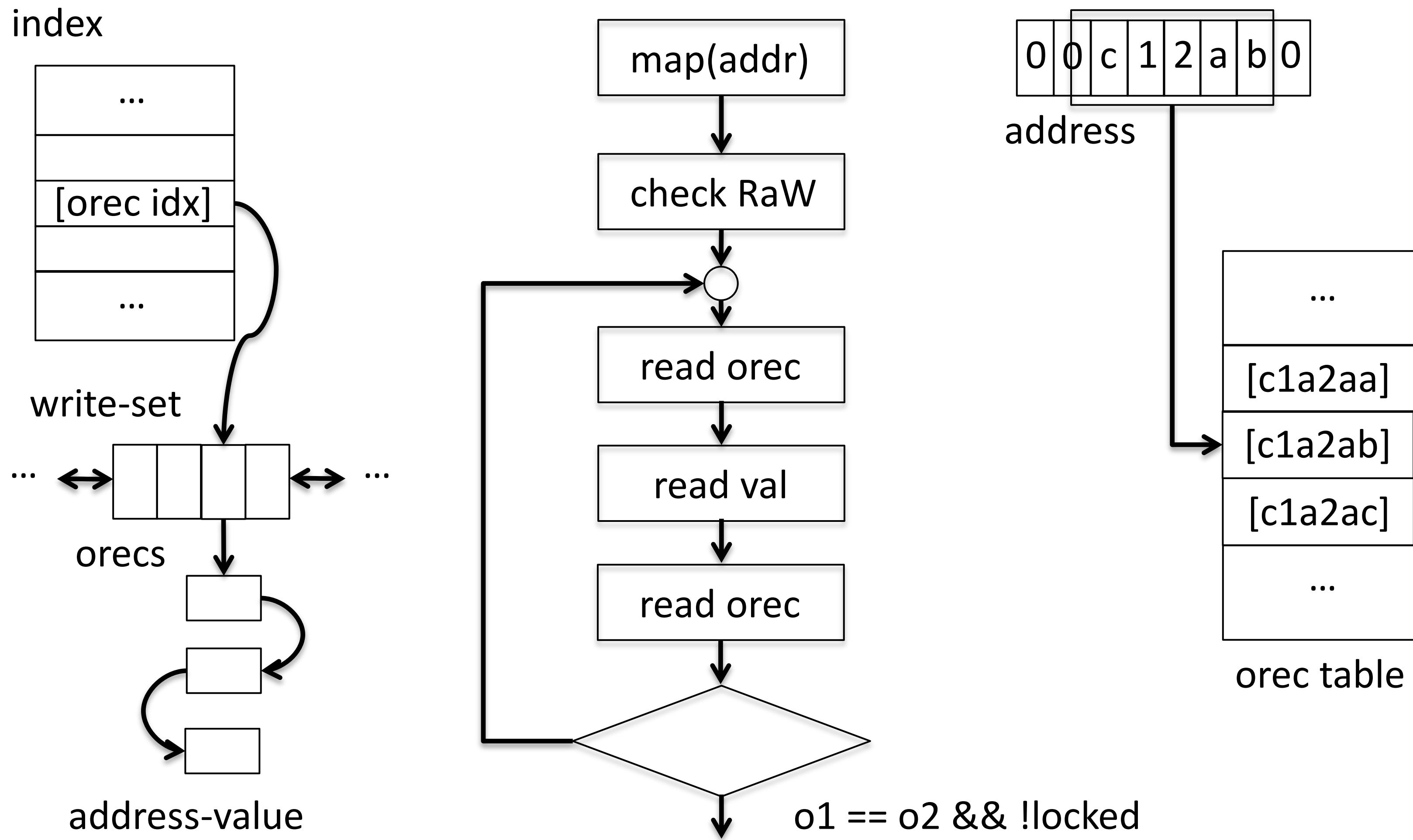






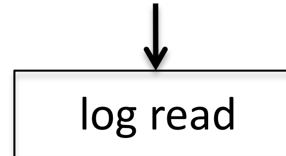
Microsoft Research

STM read



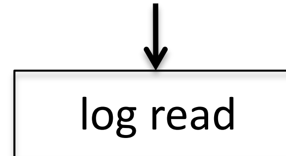
Microsoft Research

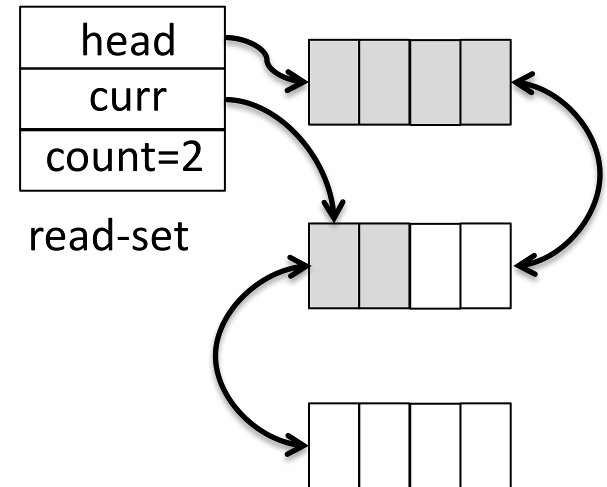
STM read (cont’d)



Microsoft Research

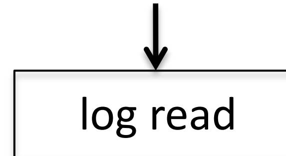
STM read (cont'd)

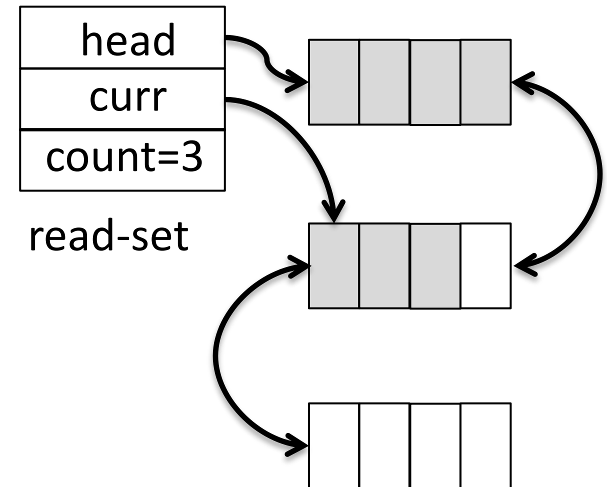




Microsoft Research

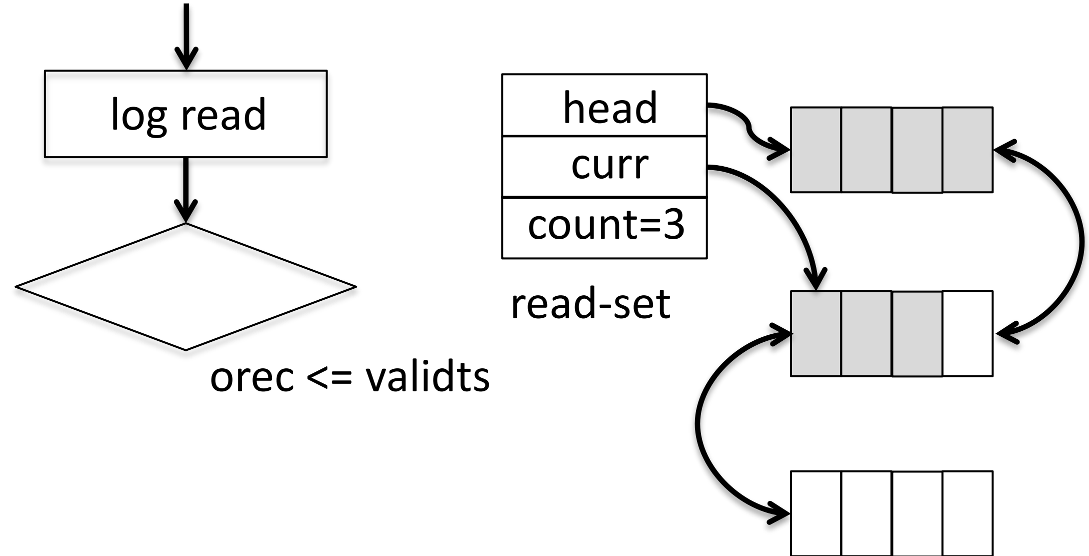
STM read (cont'd)





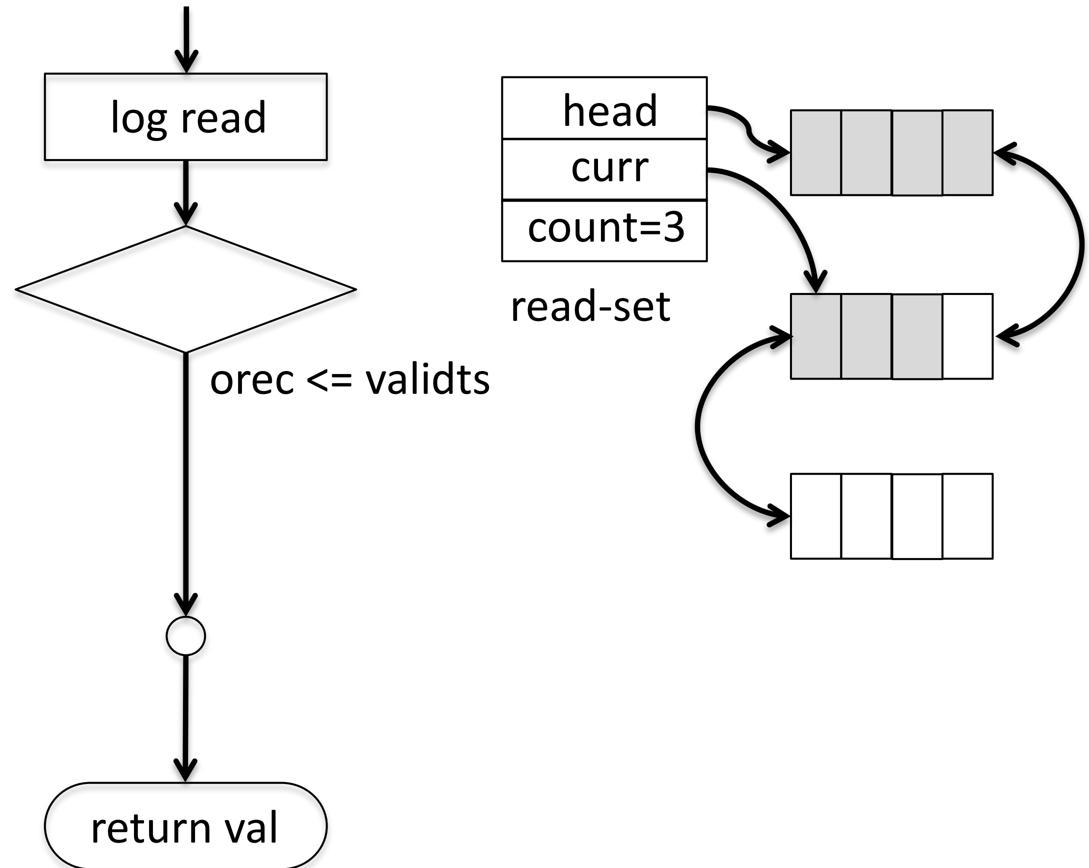
Microsoft Research

STM read (cont'd)



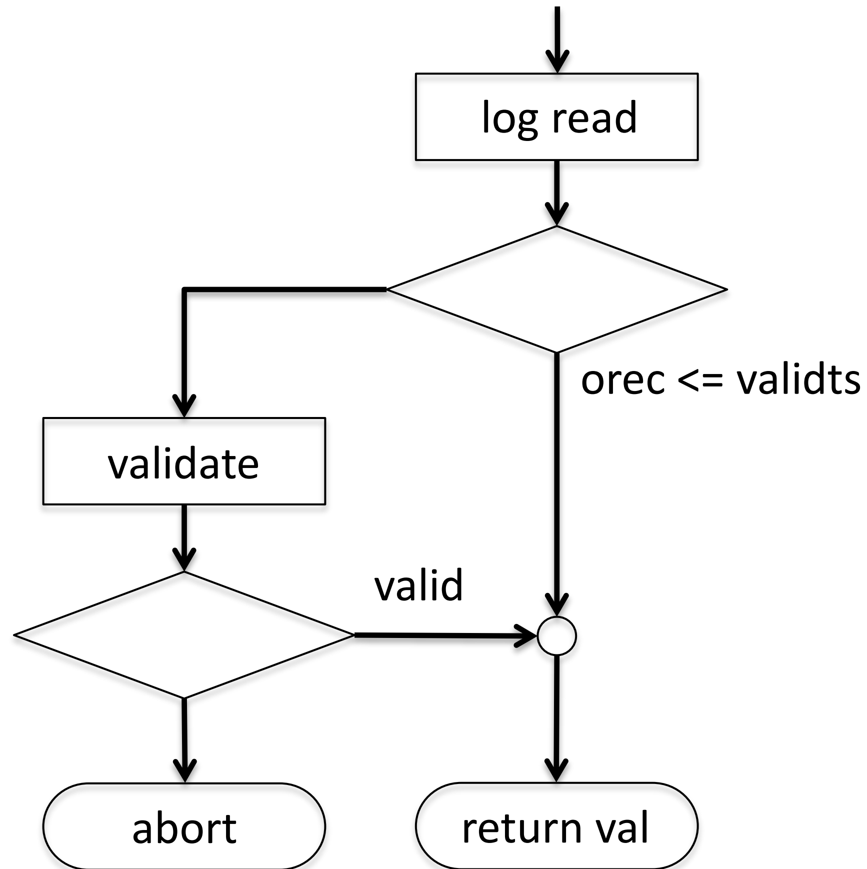
Microsoft Research

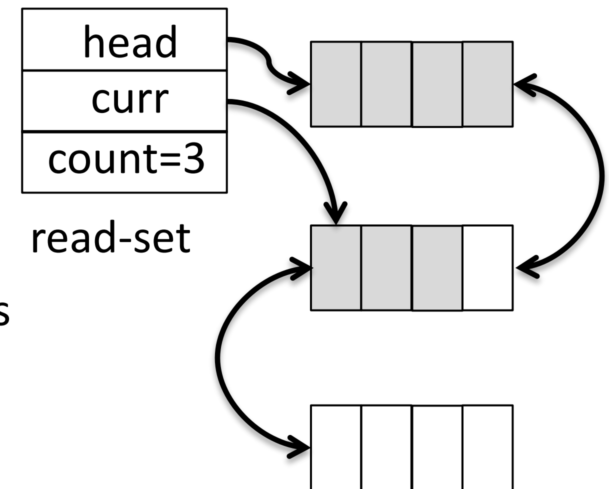
STM read (cont'd)



Microsoft Research

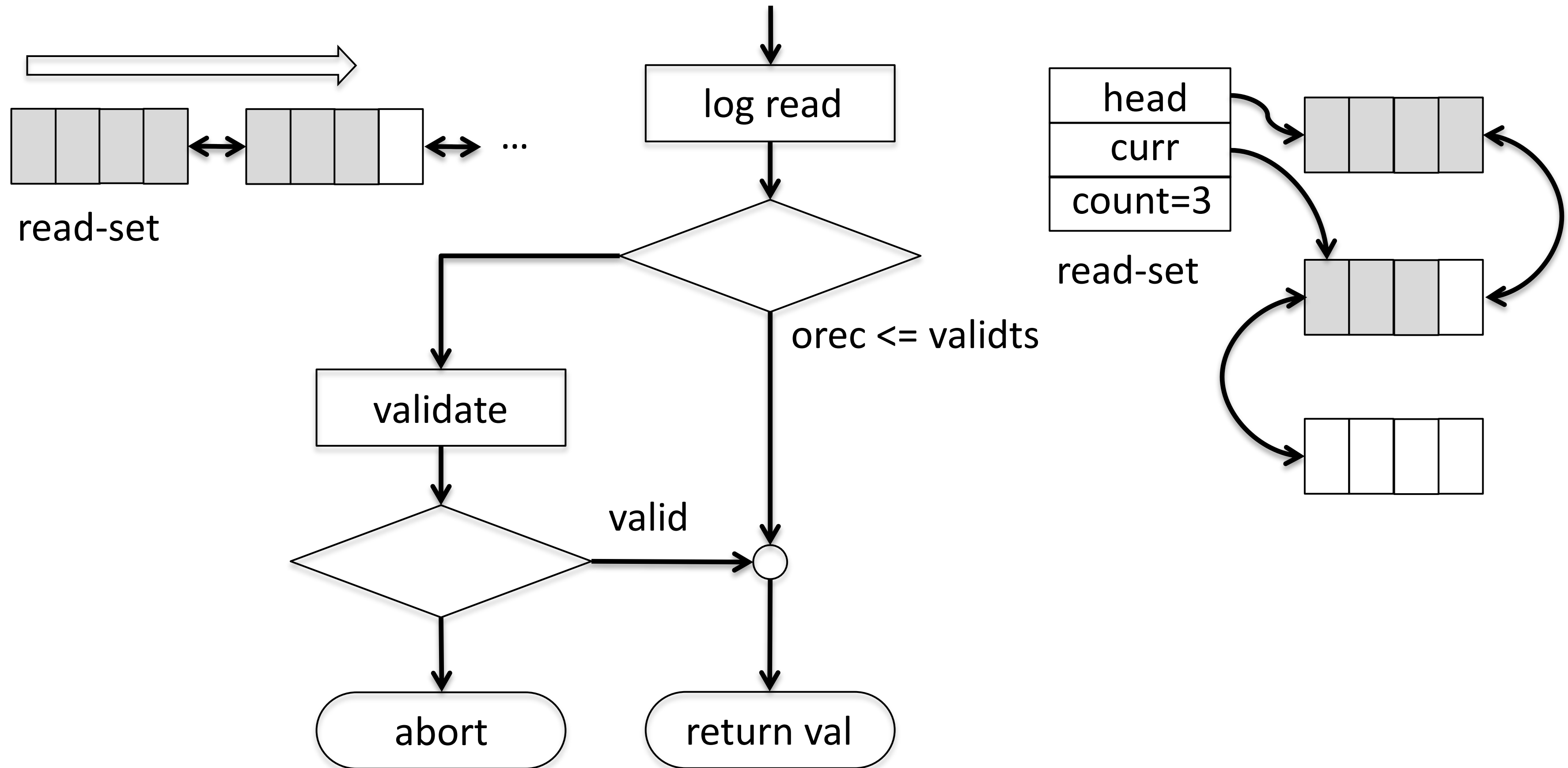
STM read (cont'd)





Microsoft Research

STM read (cont'd)



Microsoft Research

STM read fast-path

- Significantly more expensive than CPU read instructions
  - >10 instructions
- Writes have comparable costs
  - Incurs a compare-and-swap at commit time

Microsoft Research

Overview

- STM overheads
- SpecTM
  - Short-transaction API
  - Collocating data and meta-data
  - In-word meta-data
- Evaluation

Microsoft Research

Overview

- STM overheads
- SpecTM
  - Short-transaction API
  - Collocating data and meta-data
  - In-word meta-data
- Evaluation

Microsoft Research

Short transactions

- Short read-write transactions
- Short read-only transactions
- Single-access transactions
  - Read, Write, CAS

Microsoft Research

Short read-write transactions

```
word_t TxRWRead_1(word_t *addr_1);
word_t TxRWRead_2(word_t *addr_2);
word_t TxRWRead_3(word_t *addr_3);
...

void TxRWCommit_1(word_t val);
void TxRWCommit_2(word_t v1, word_t v2);
void TxRWCommit_3(word_t v1,
                  word_t v2,
                  word_t v3);
...
```


Microsoft Research

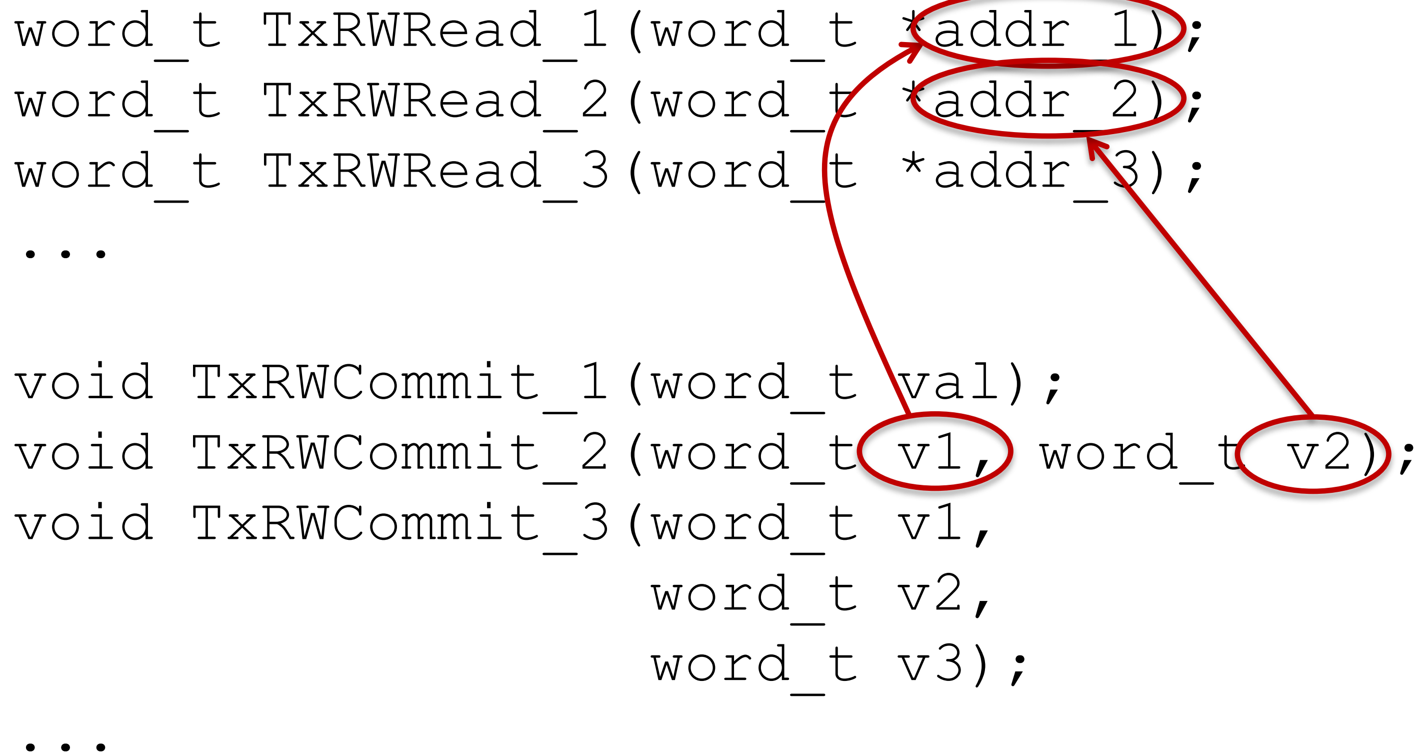
Short read-write transactions

```
word_t TxRWRead_1(word_t *addr_1);
word_t TxRWRead_2(word_t *addr_2);
word_t TxRWRead_3(word_t *addr_3);
...

void TxRWCommit_1(word_t val);
void TxRWCommit_2(word_t v1, word_t v2);
void TxRWCommit_3(word_t v1,
                  word_t v2,
                  word_t v3);
...
```


Microsoft Research

Short read-write transactions



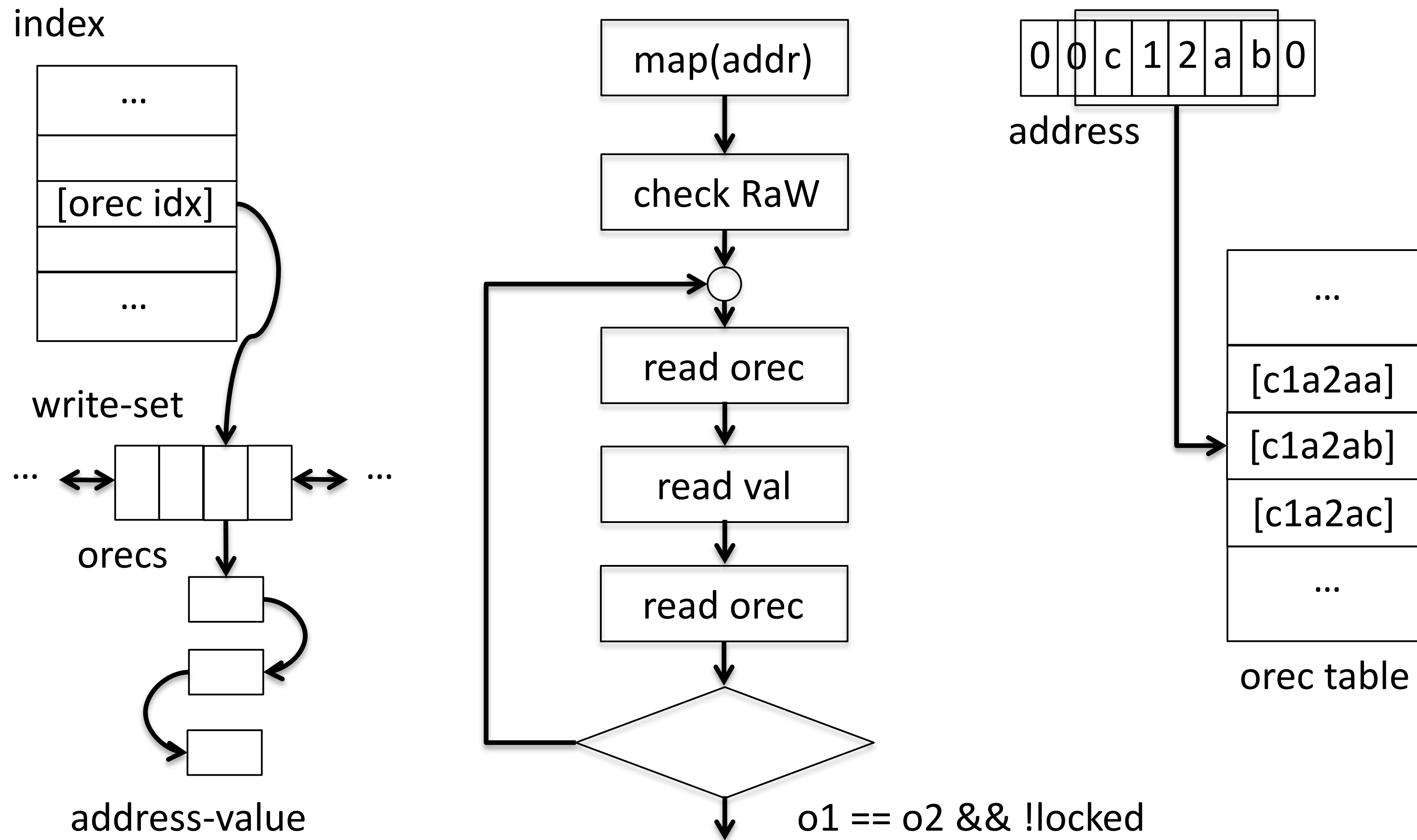
Short read-write transactions

```
word_t TxRWRead_1(word_t *addr_1);
word_t TxRWRead_2(word_t *addr_2);
word_t TxRWRead_3(word_t *addr_3);
...

void TxRWCommit_1(word_t val);
void TxRWCommit_2(word_t v1, word_t v2);
void TxRWCommit_3(word_t v1,
                  word_t v2,
                  word_t v3);
...
```

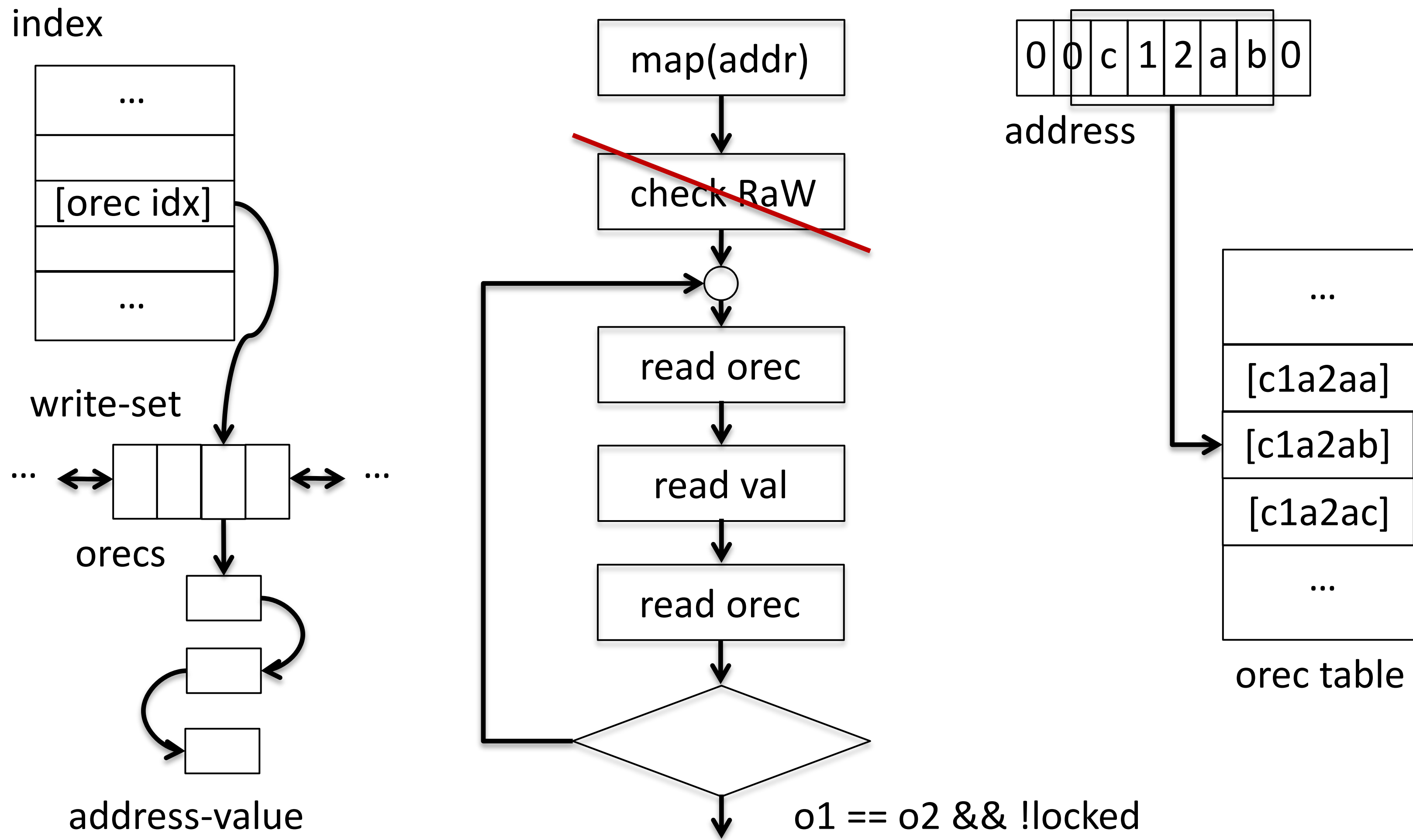

Microsoft Research

Short transaction optimizations



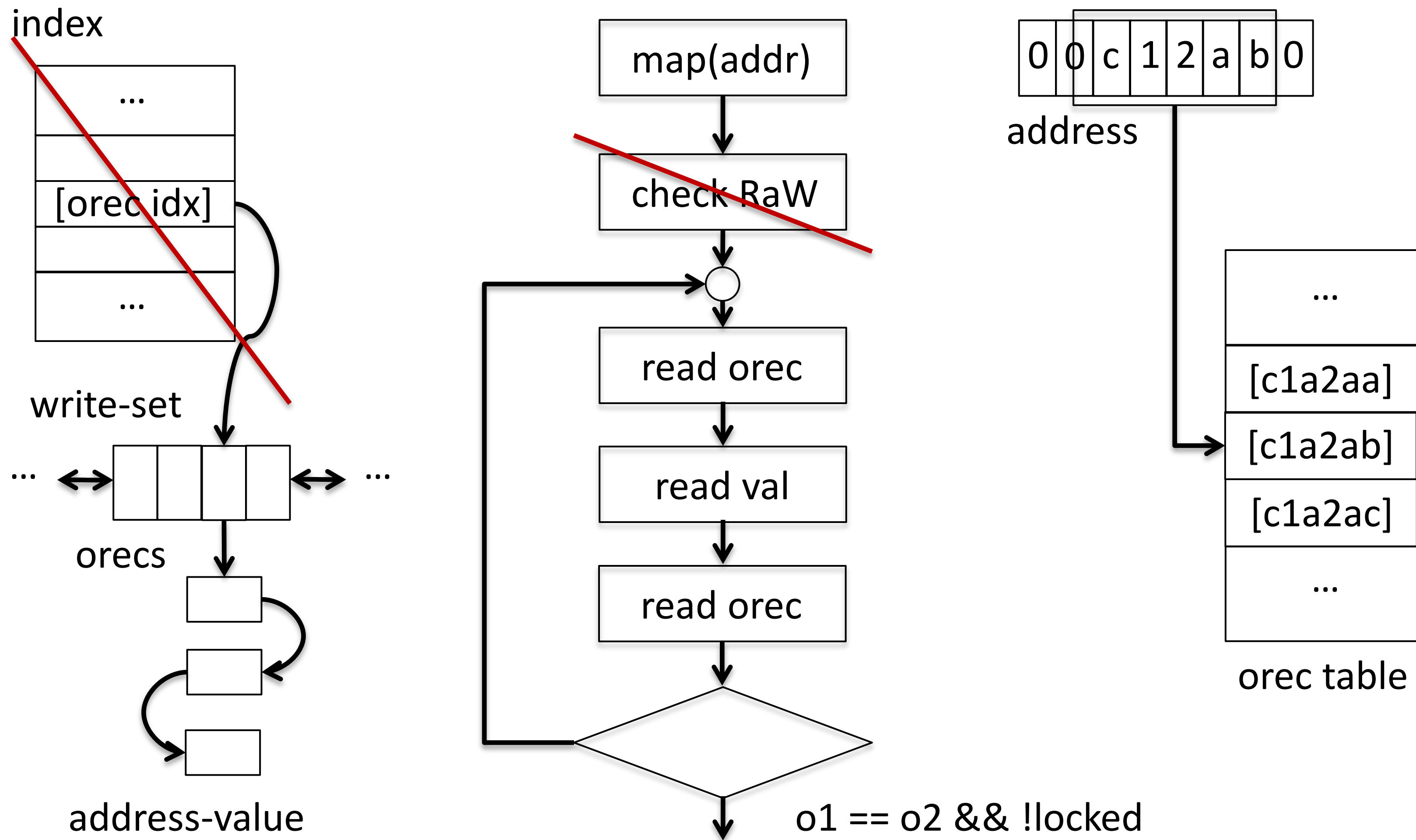
Microsoft Research

Short transaction optimizations



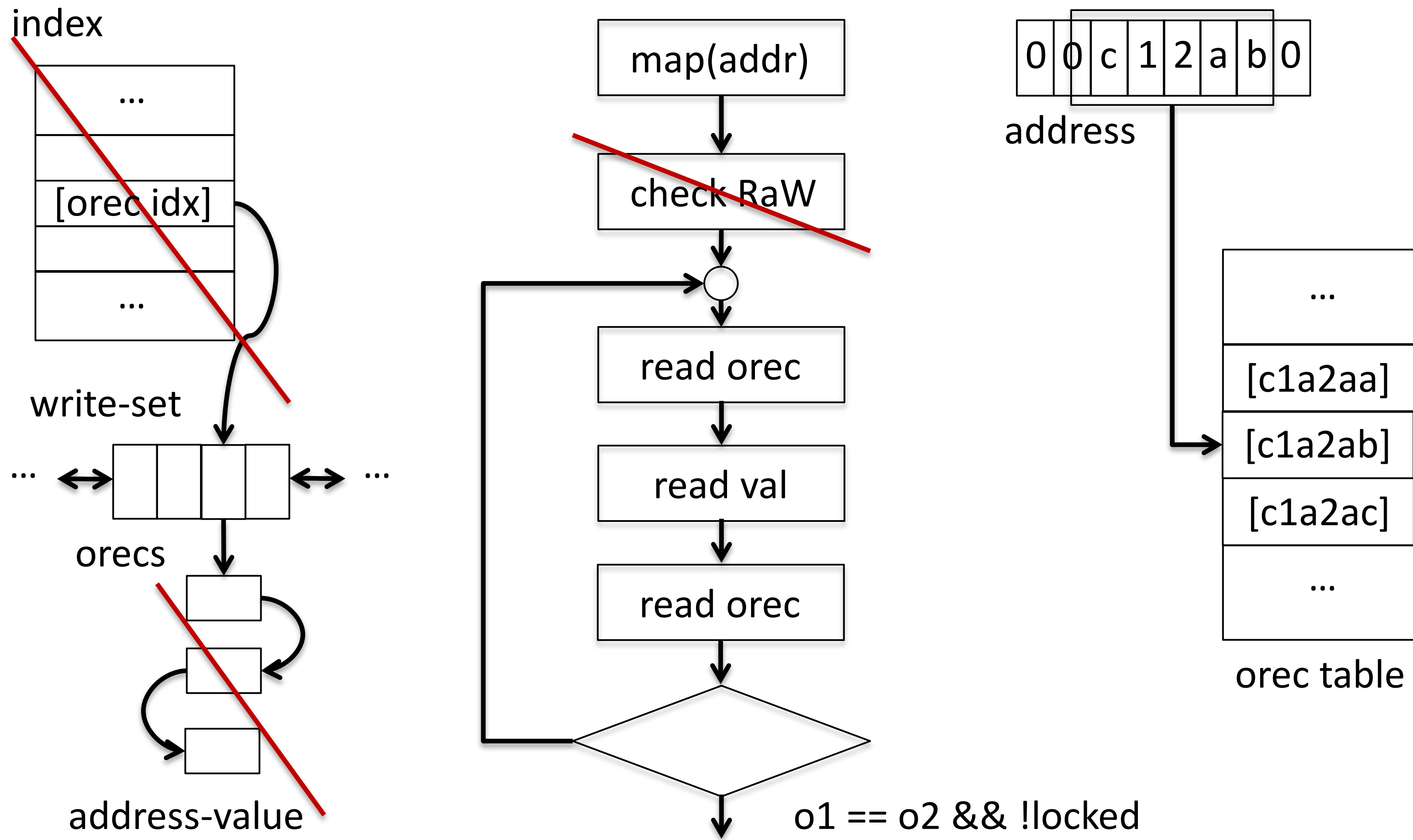
Microsoft Research

Short transaction optimizations



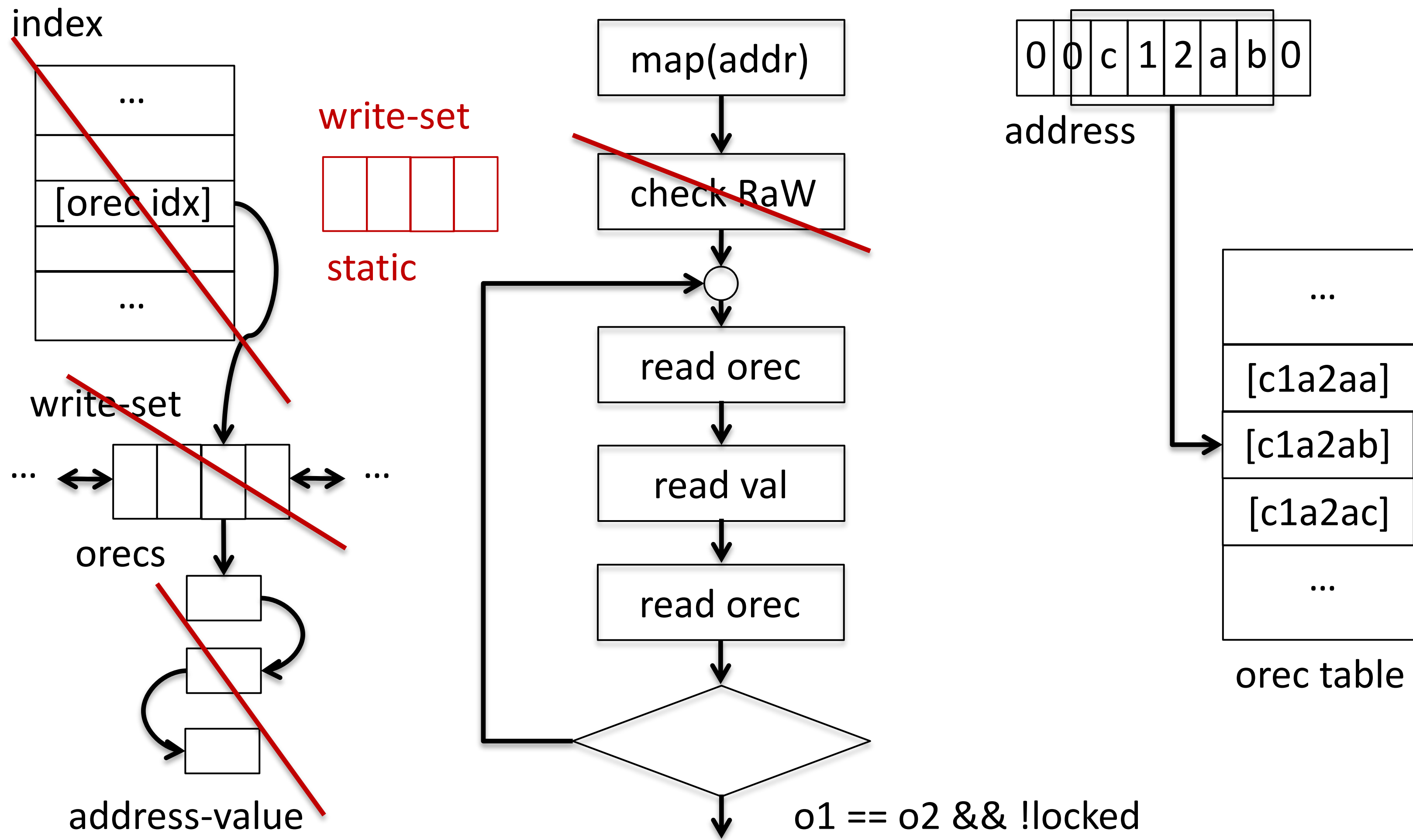
Microsoft Research

Short transaction optimizations



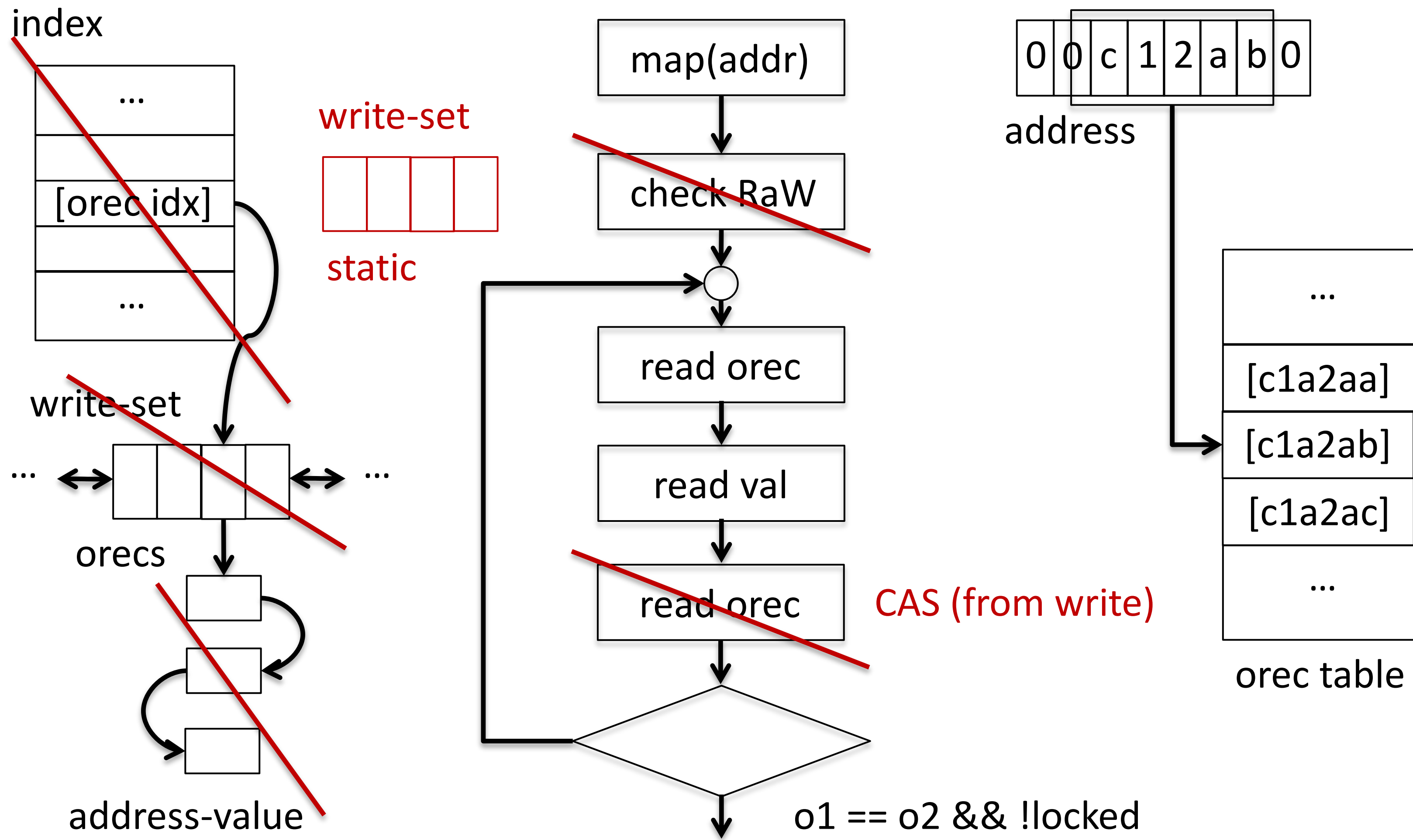
Microsoft Research

Short transaction optimizations



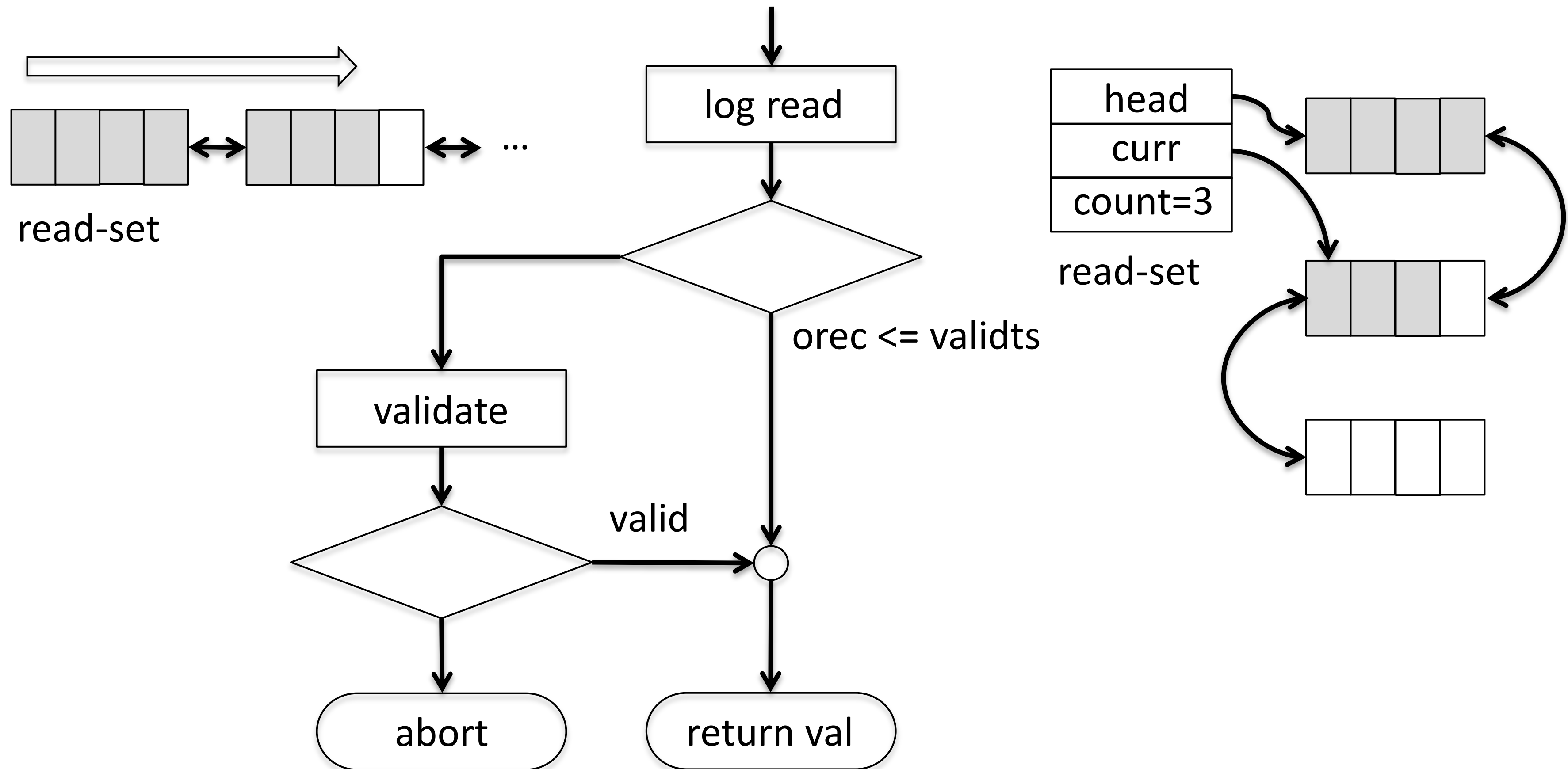
Microsoft Research

Short transaction optimizations



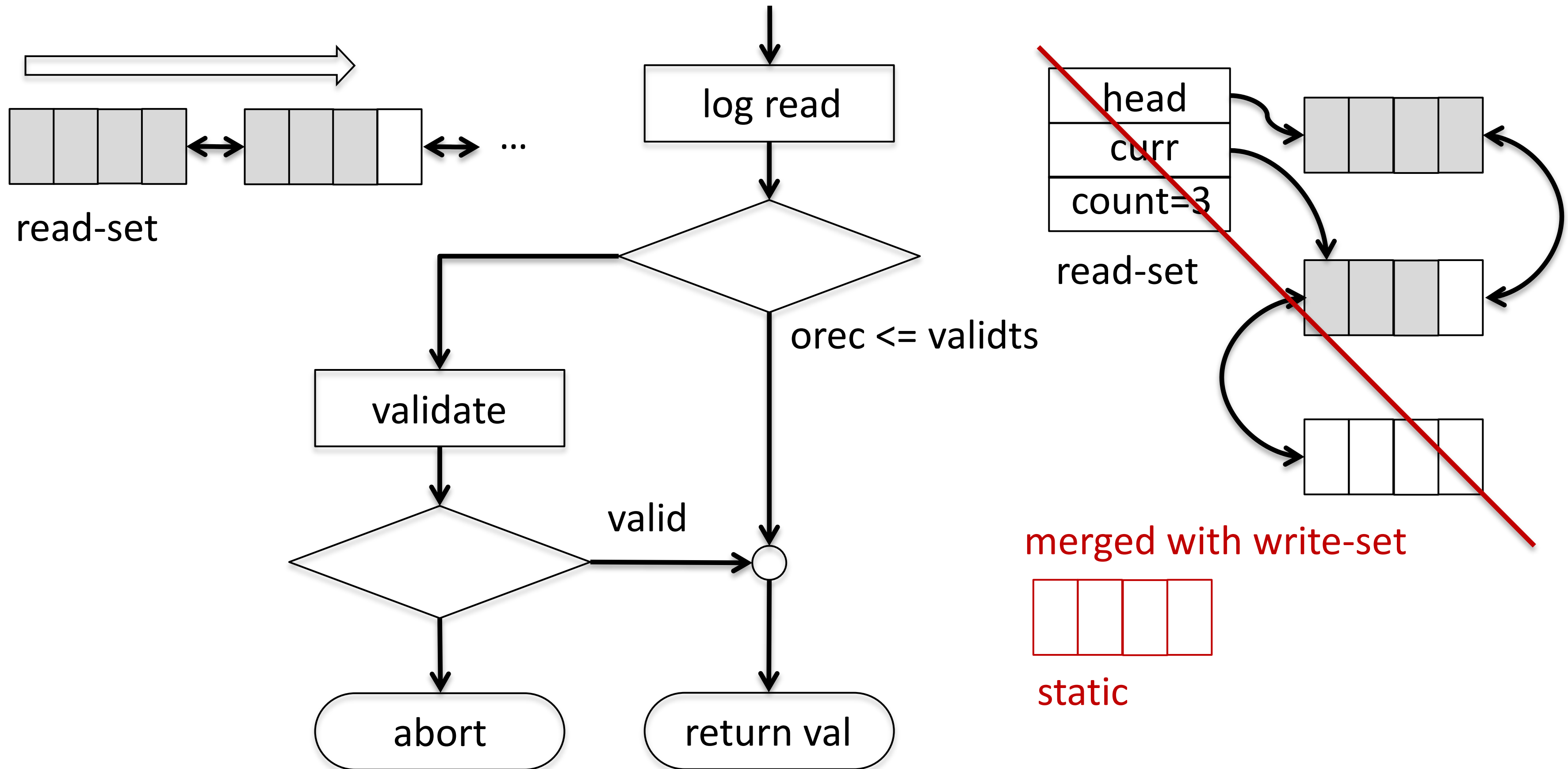
Microsoft Research

Short transaction optimizations (cont'd)



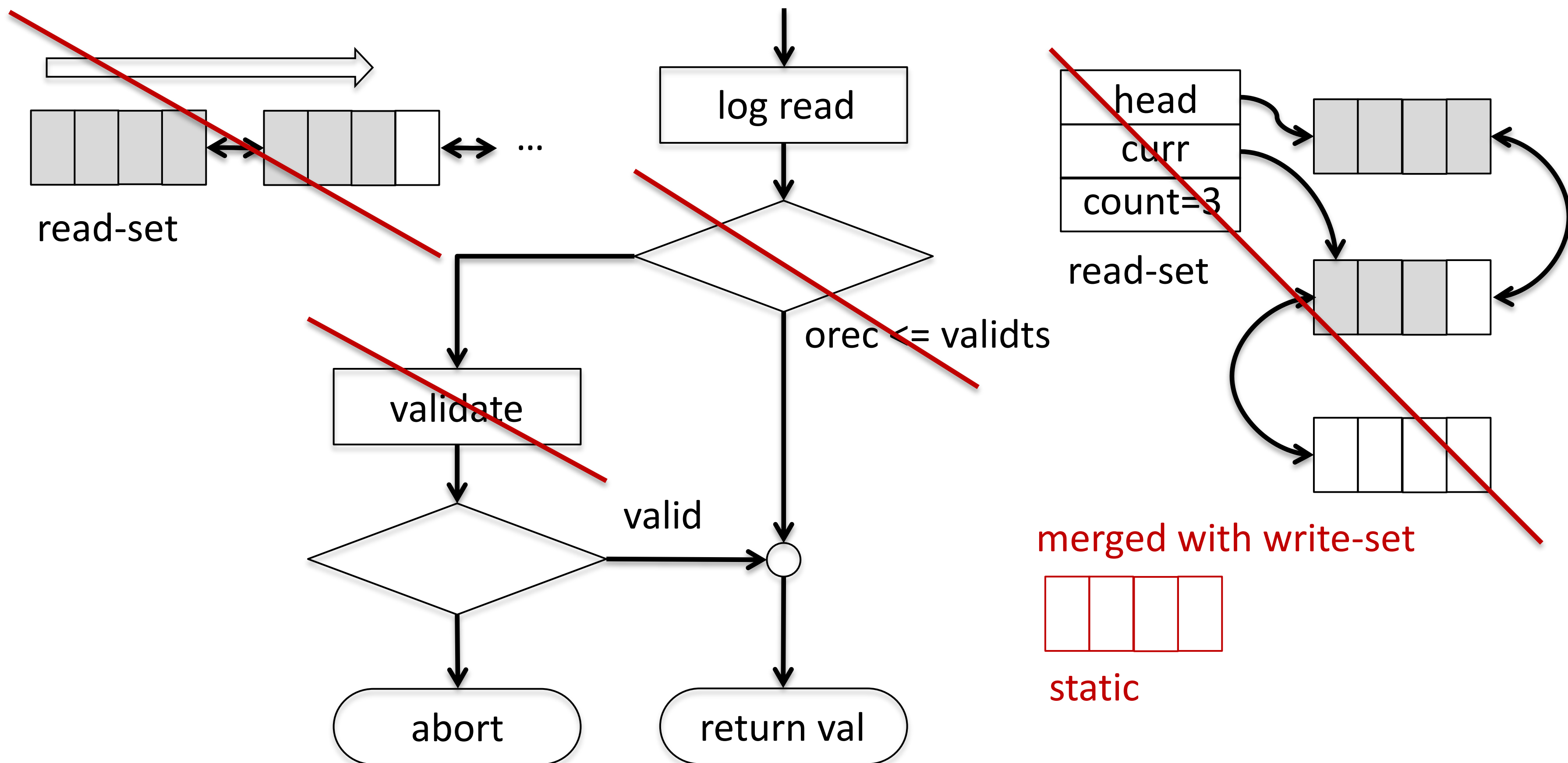
Microsoft Research

Short transaction optimizations (cont’d)



Microsoft Research

Short transaction optimizations (cont'd)



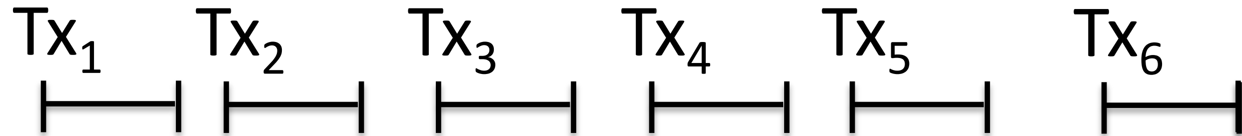
Microsoft Research

Using short transactions

With “traditional” transactions the whole operation is a single transaction

Microsoft Research

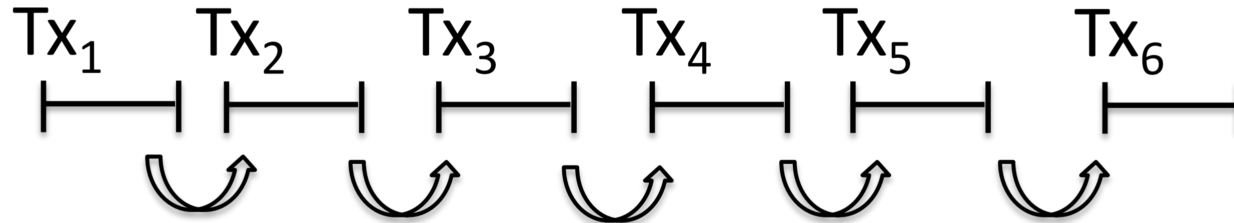
Using short transactions



Split operation into a sequence of short transactions

Microsoft Research

Using short transactions

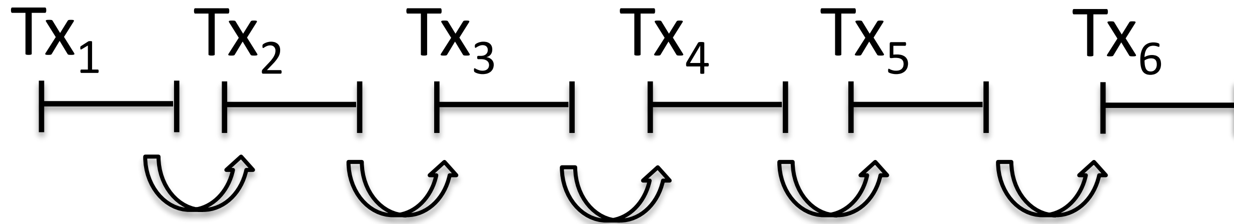


“Glue” them together using techniques similar to lock-free algorithms (e.g. pointer marking)

Microsoft Research

Using short transactions

Is this simpler than using CAS directly?



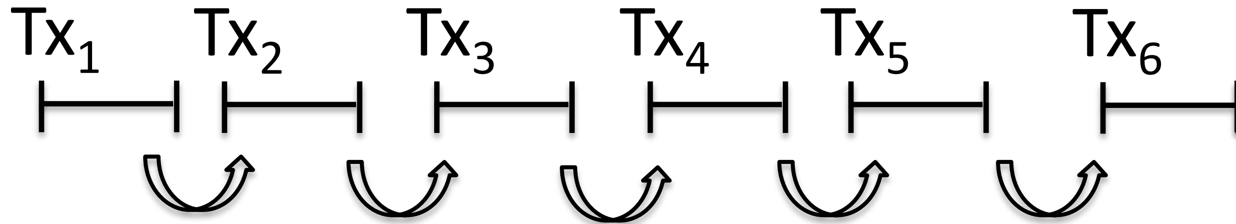
"Glue" them together using techniques similar to lock-free algorithms (e.g. pointer marking)

Microsoft Research

Using short transactions

Is this simpler than using CAS directly?

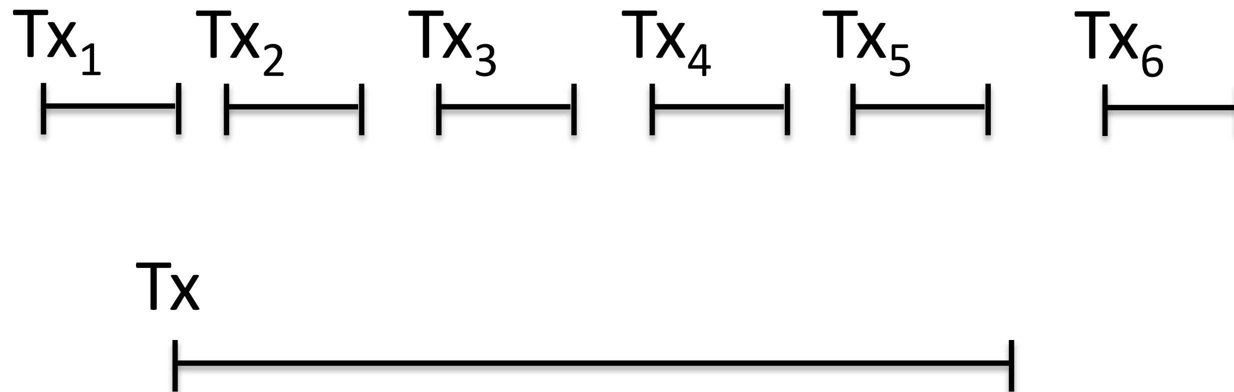
Yes: transactions eliminate tricky races



"Glue" them together using techniques similar to lock-free algorithms (e.g. pointer marking)

Microsoft Research

Mix short and ordinary transactions

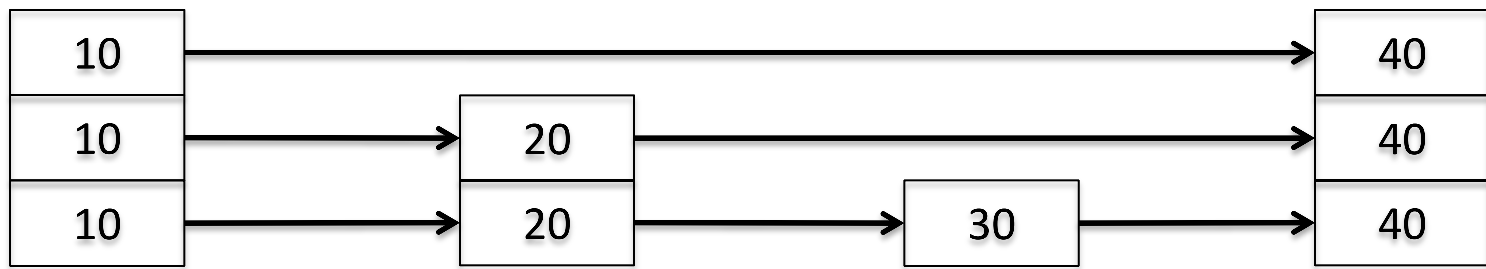


Implement corner cases using ordinary transactions

Microsoft Research

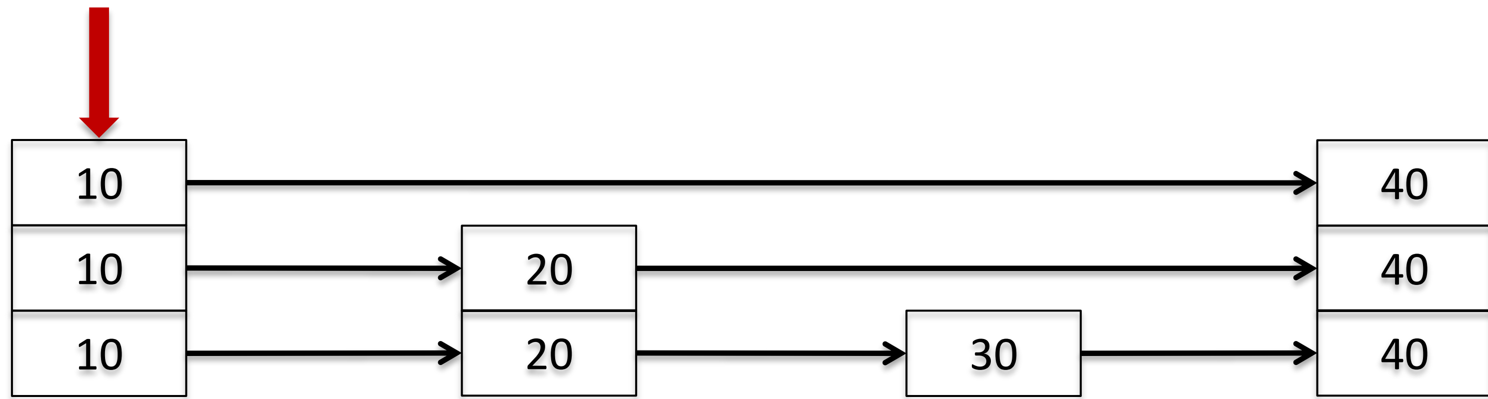
Example

Remove 20



Microsoft Research

Example

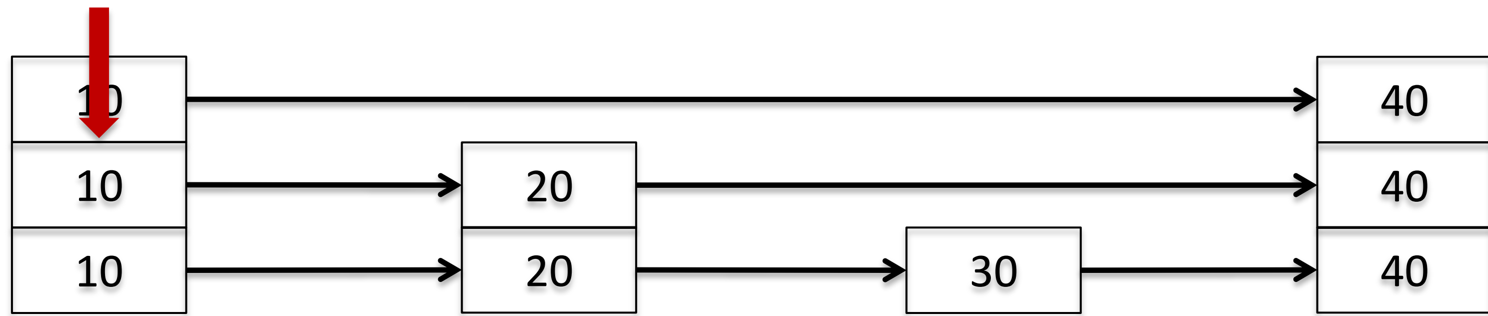


Microsoft Research

Example

Remove 20

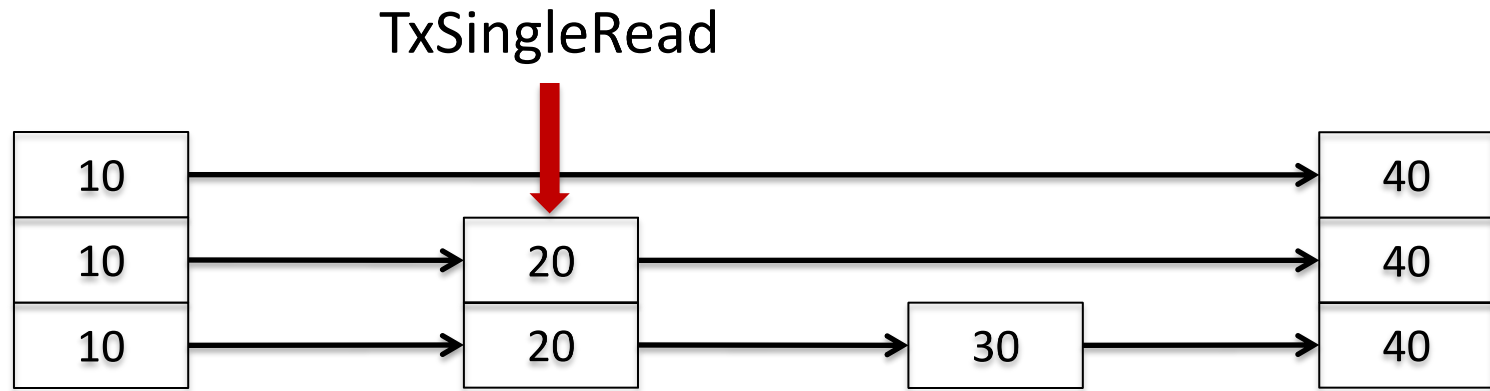
TxSingleRead



Microsoft Research

Example

Remove 20

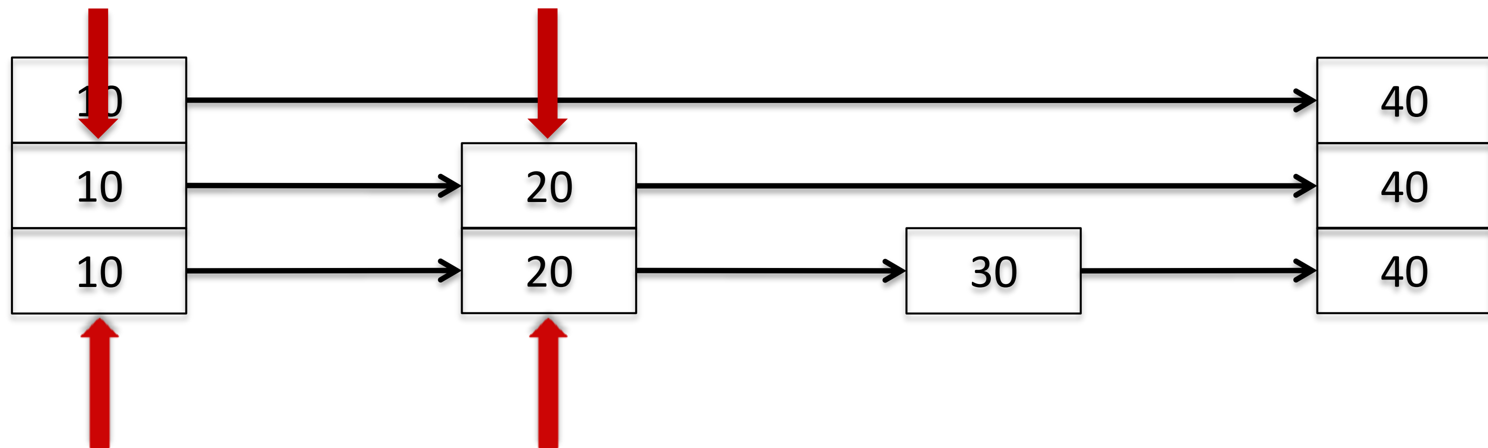


Microsoft Research

Example

Remove 20

Short read-write transaction to mark the node and unlink it

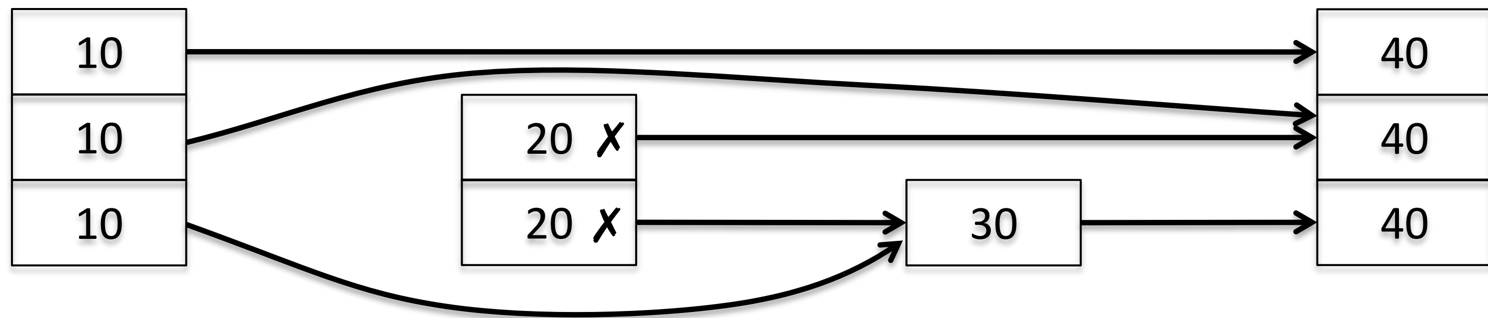


Microsoft Research

Example

Remove 20

Short read-write transaction to mark the node and unlink it

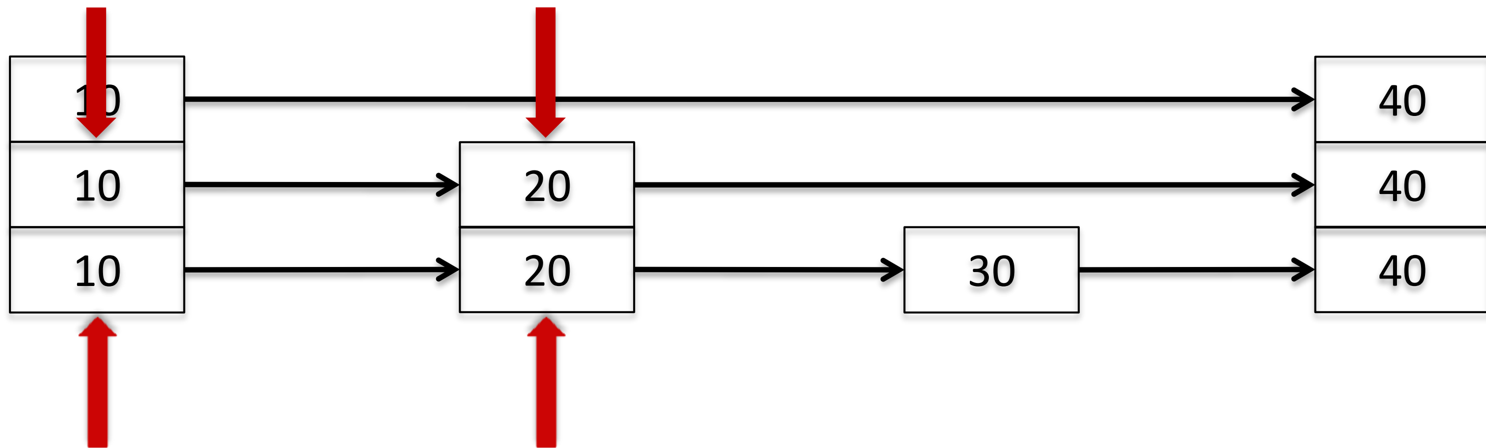


Microsoft Research

Example

Remove 20

This becomes trickier when using CAS directly

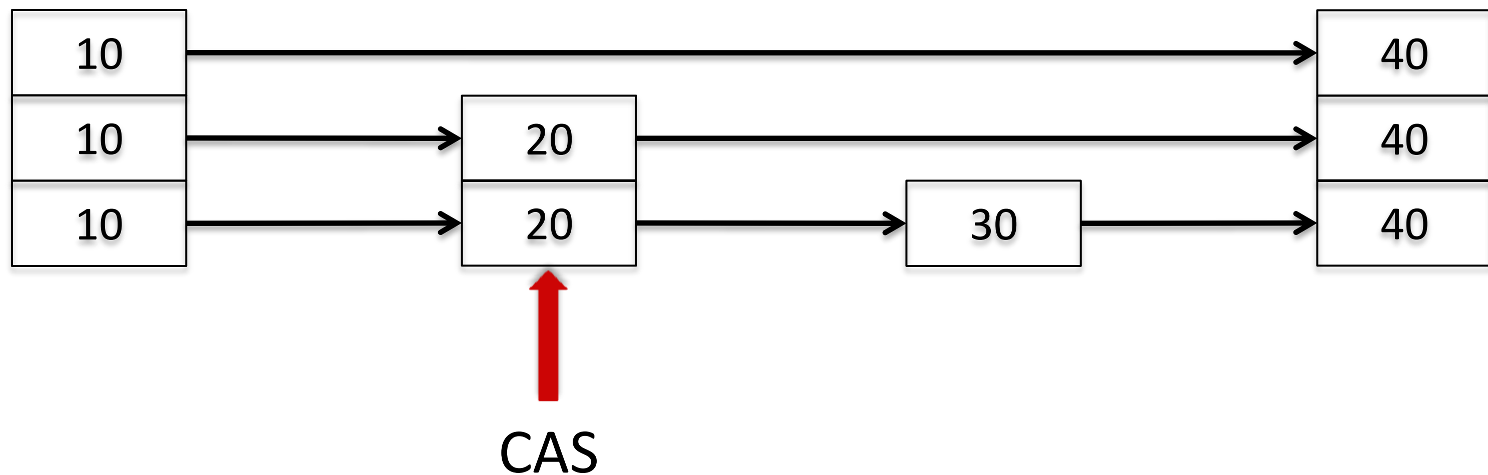


Microsoft Research

Example

Remove 20

We have to use several CASes

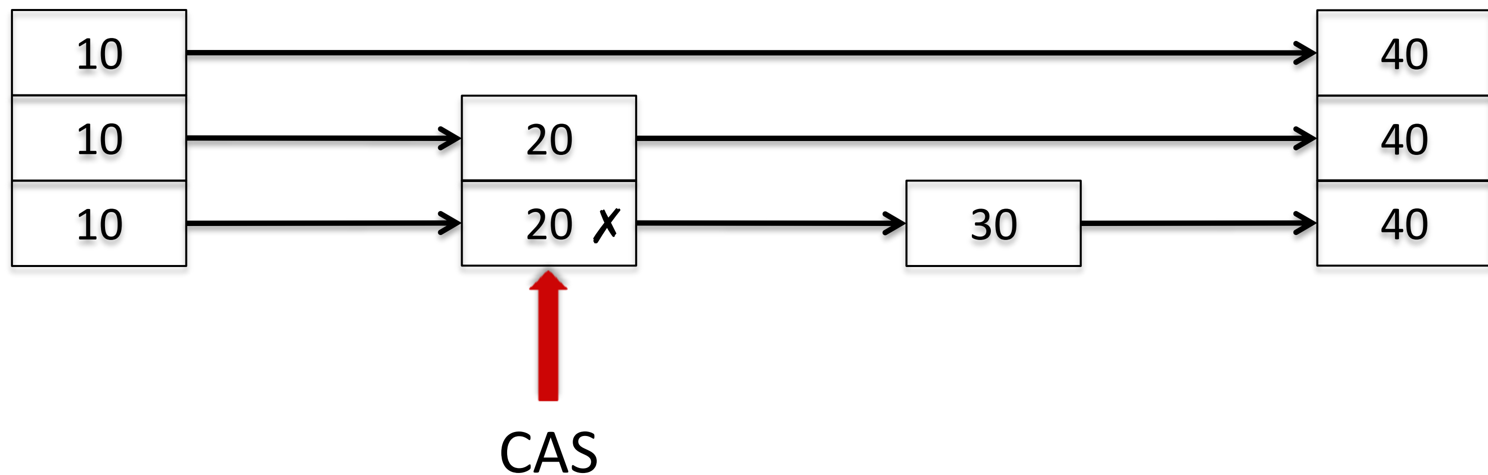


Microsoft Research

Example

Remove 20

We have to use several CASes

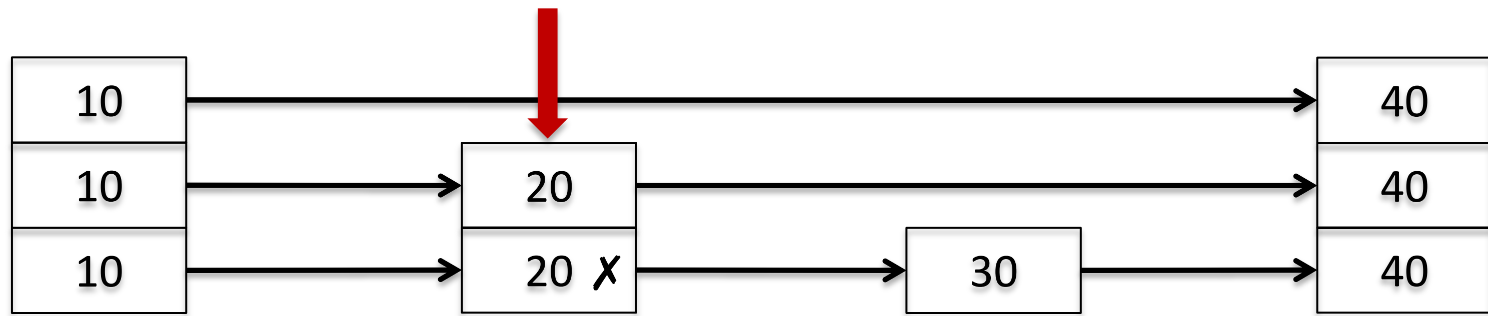


Microsoft Research

Example

Remove 20

We have to use several CASes

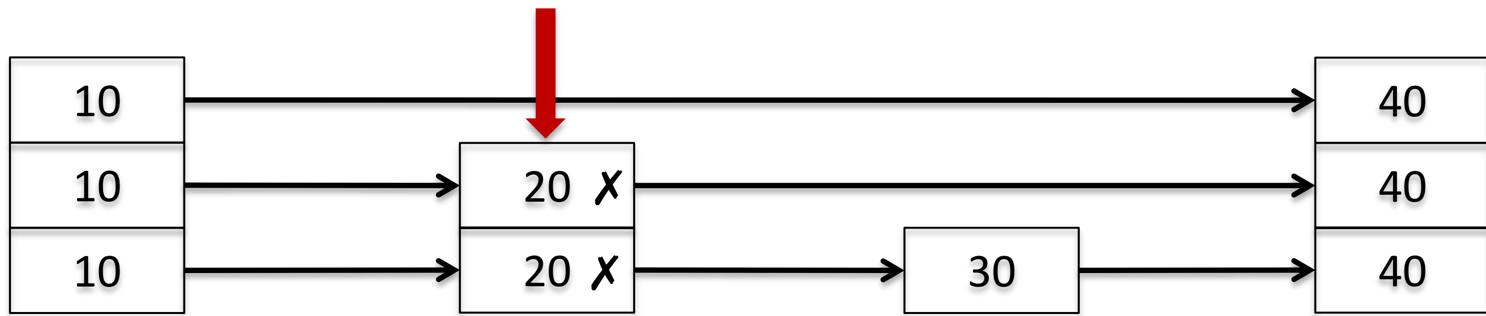


Microsoft Research

Example

Remove 20

We have to use several CASes

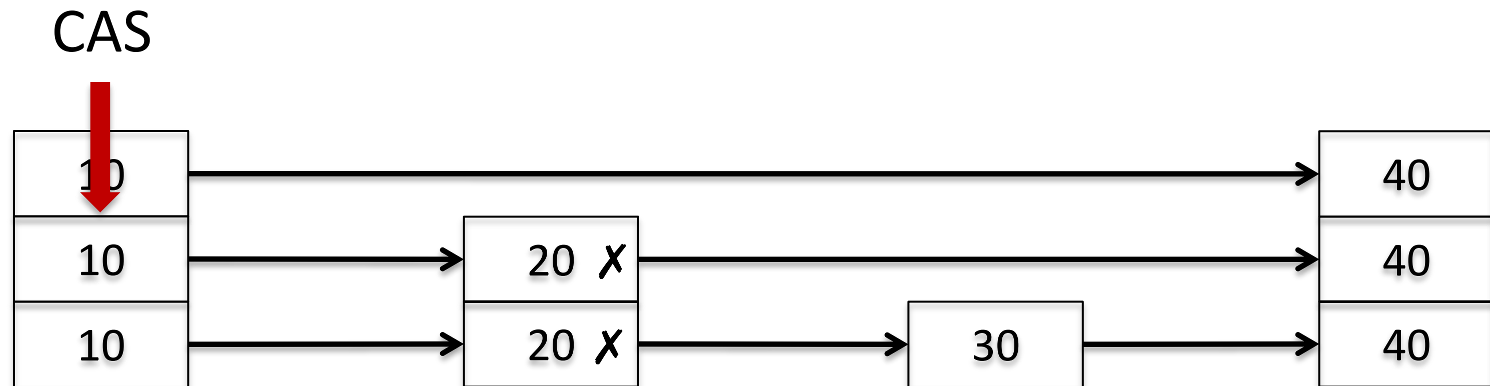


Microsoft Research

Example

Remove 20

We have to use several CASes

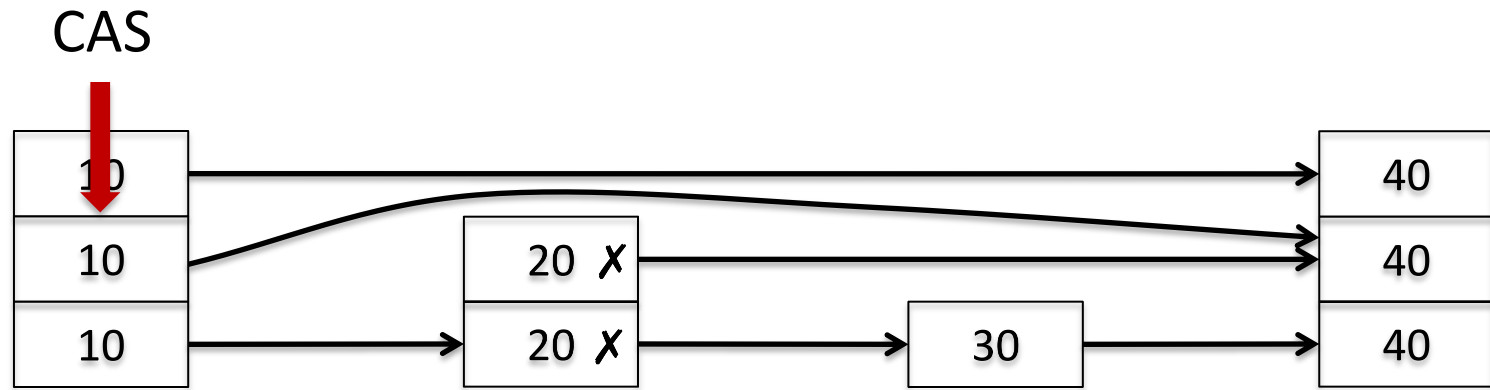


Microsoft Research

Example

Remove 20

We have to use several CASes

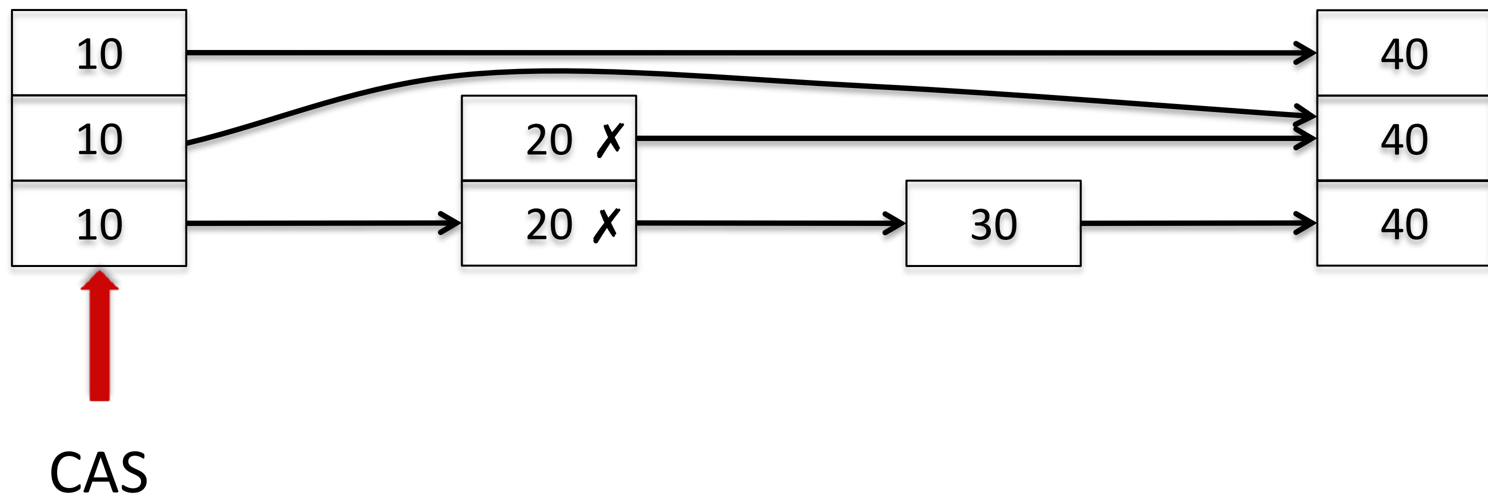


Microsoft Research

Example

Remove 20

We have to use several CASes

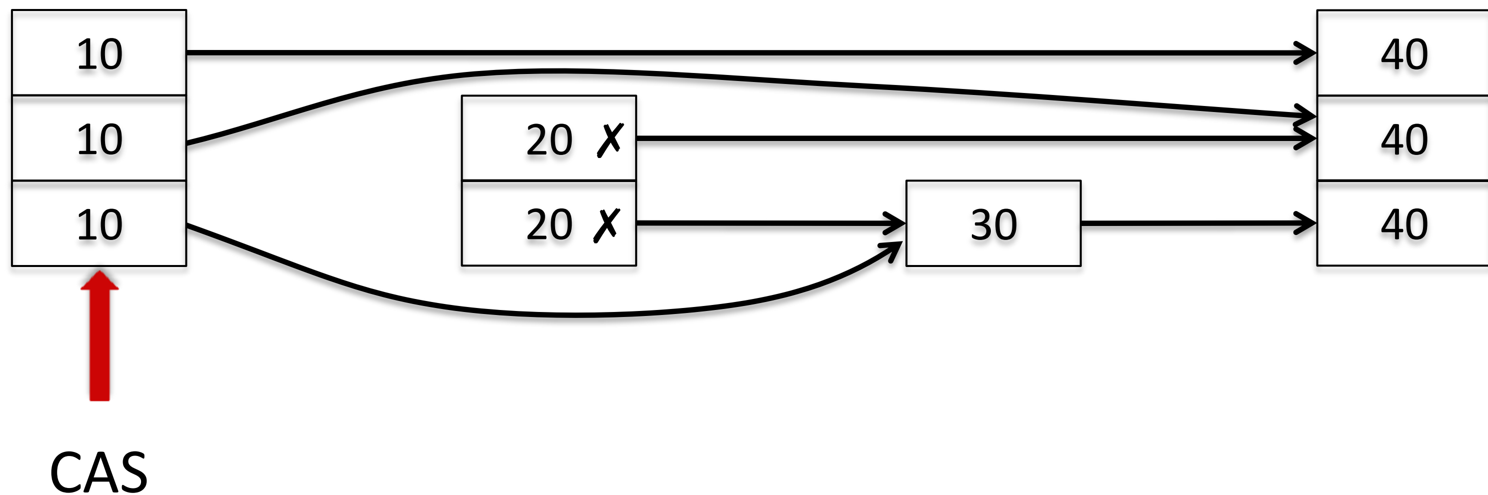


Microsoft Research

Example

Remove 20

We have to use several CASes

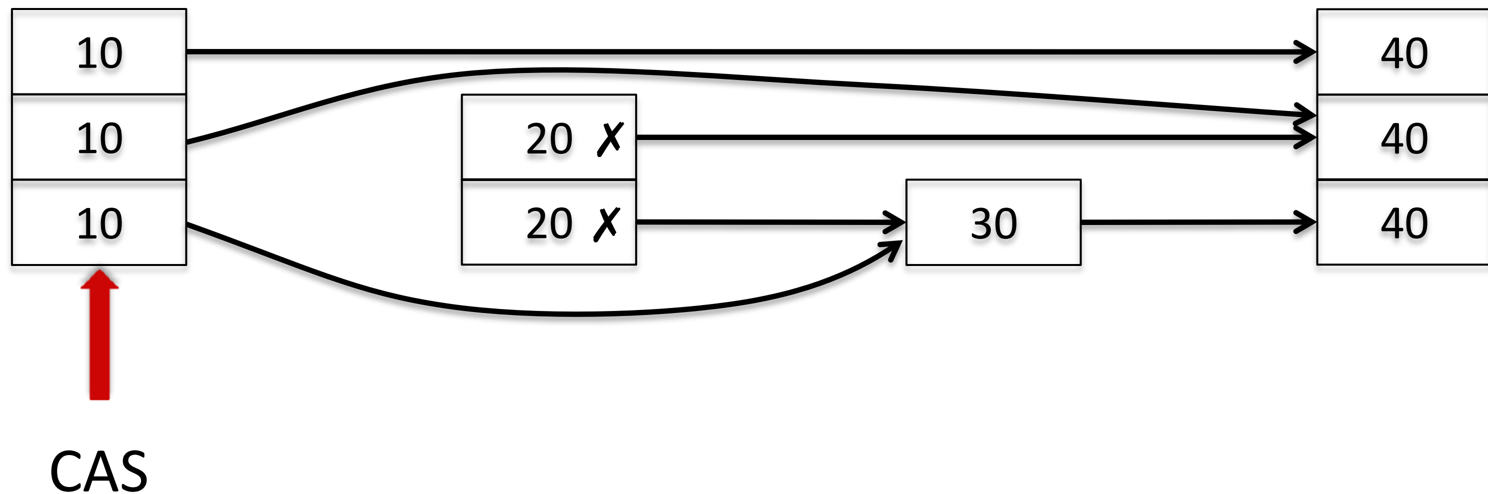


Microsoft Research

Example

Remove 20

There might be a race at each of these steps

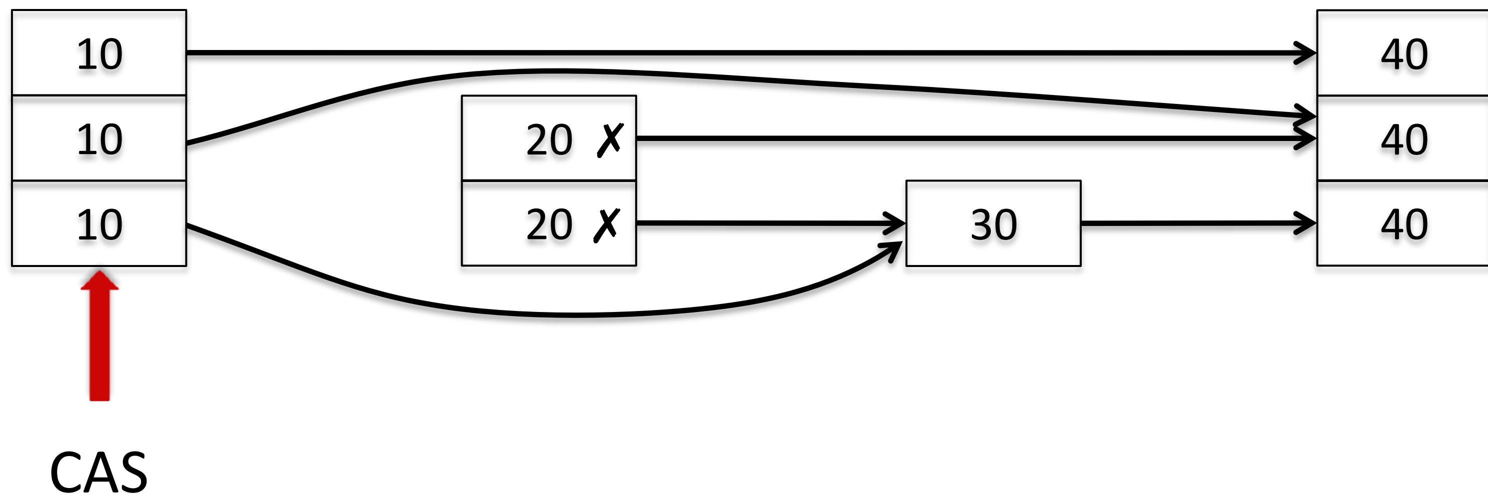


Microsoft Research

Example

Remove 20

We need to make sure other operations clean-up for us

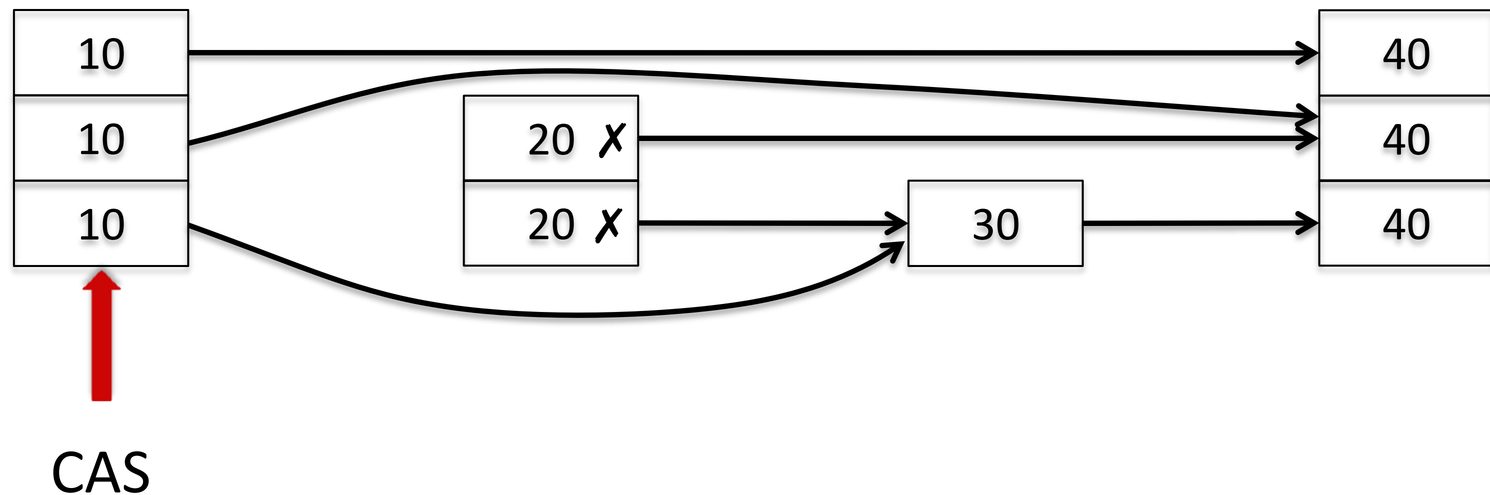


Microsoft Research

Example

Remove 20

All interactions between operations have to be handled correctly

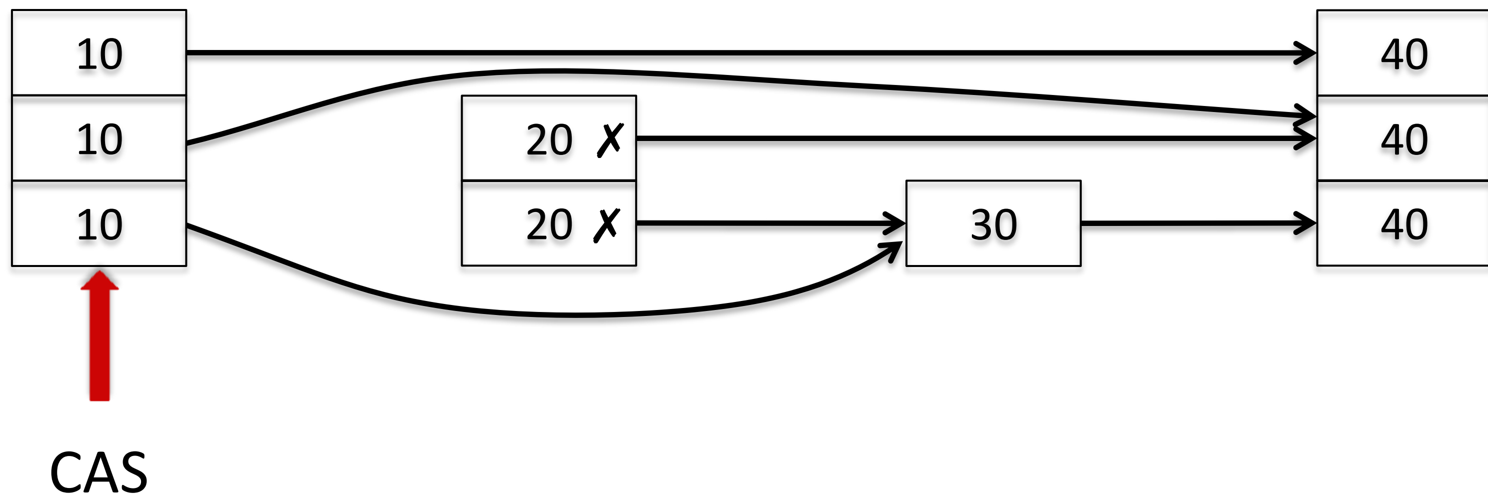


Microsoft Research

Example

Remove 20

E.g. removal of an element that is still being added to the list



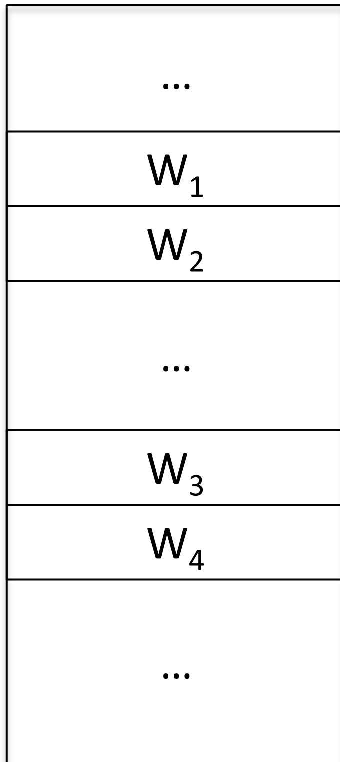
Microsoft Research

Overview

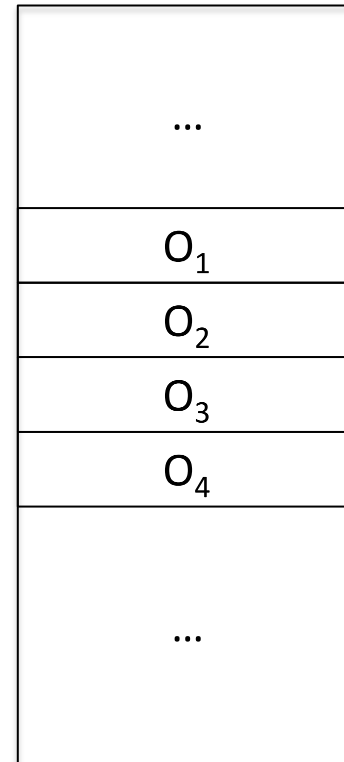
- STM overheads
- SpecTM
  - Short-transaction API
  - Collocating data and meta-data
  - In-word meta-data
- Evaluation

Microsoft Research

“Traditional” STM data layout



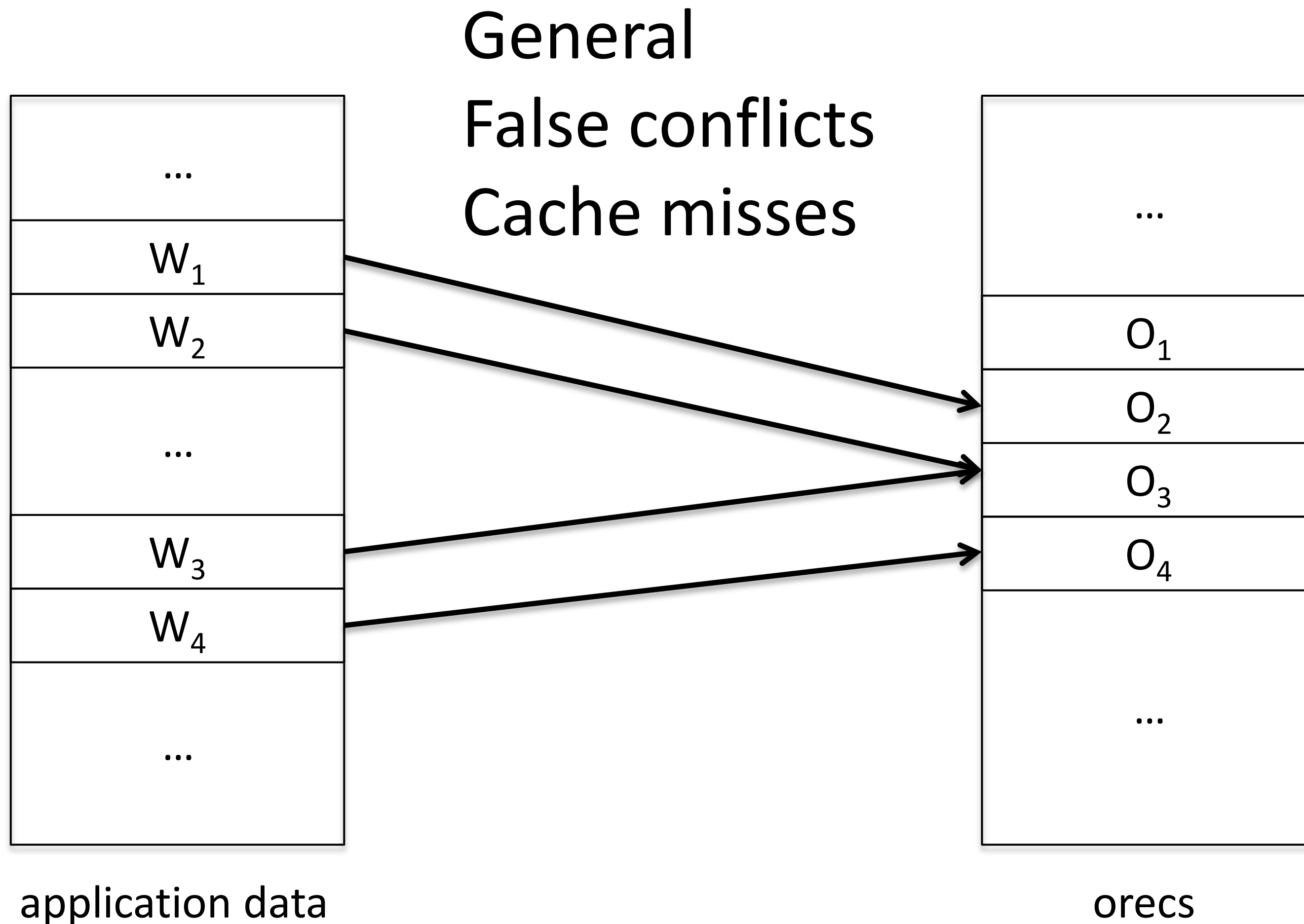
application data



orecs

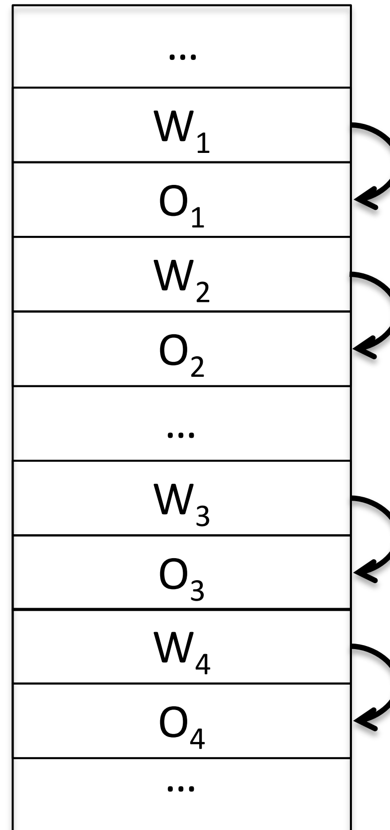
Microsoft Research

“Traditional” STM data layout



Microsoft Research

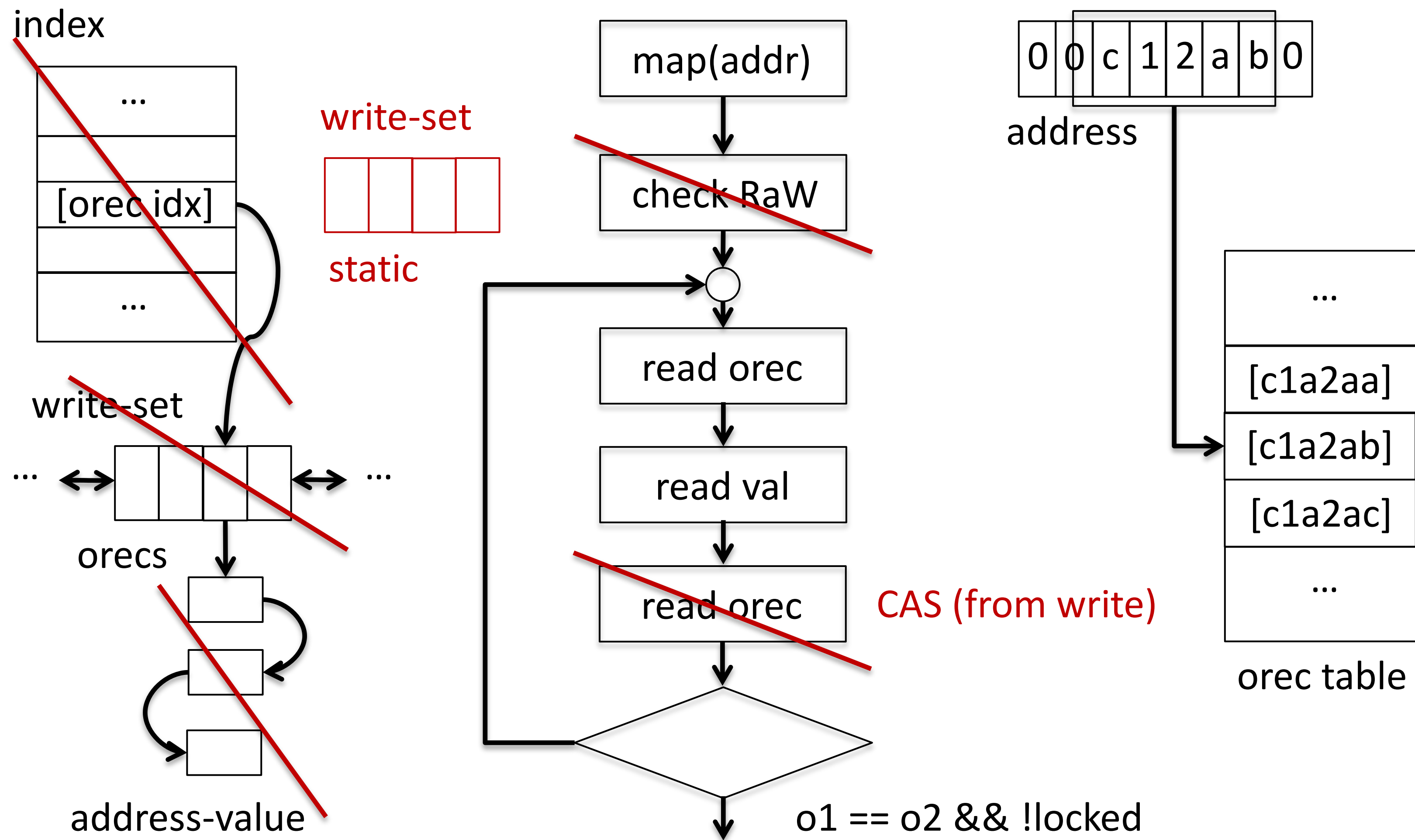
Collocate data and meta-data



Simplify mapping
No false conflicts
Same cache line

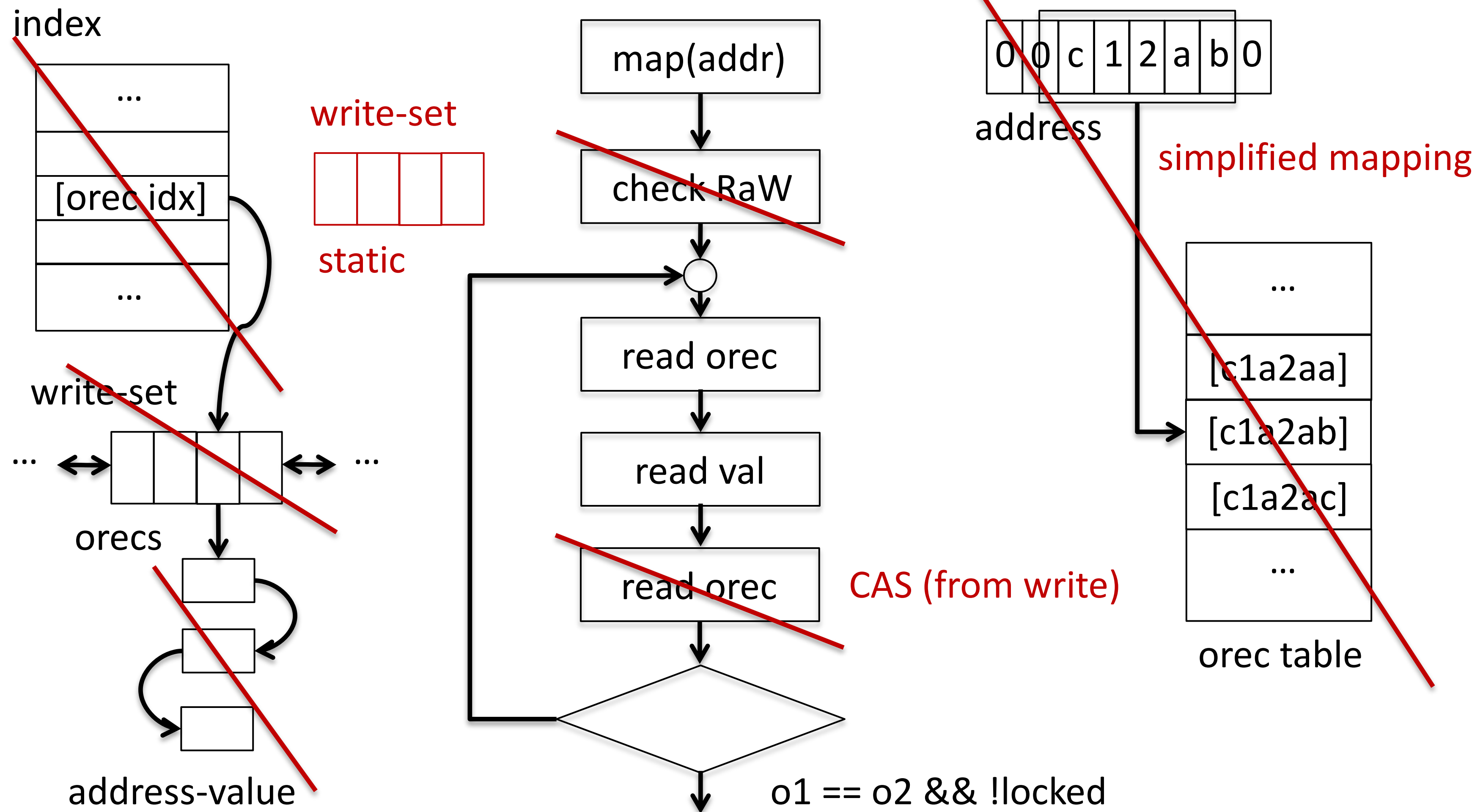
Microsoft Research

Collocated meta-data optimizations



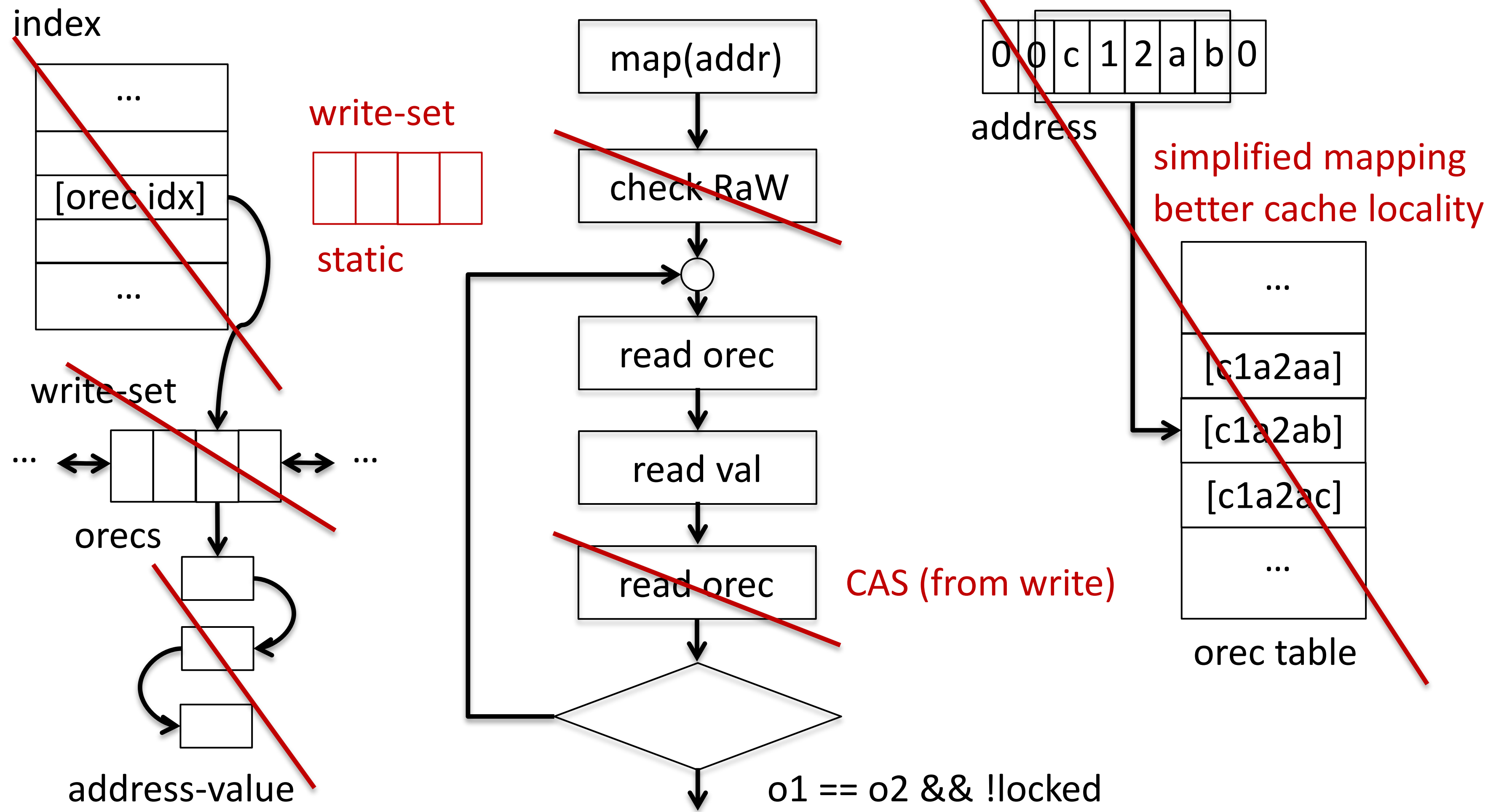
Microsoft Research

Collocated meta-data optimizations



Microsoft Research

Collocated meta-data optimizations



Microsoft Research

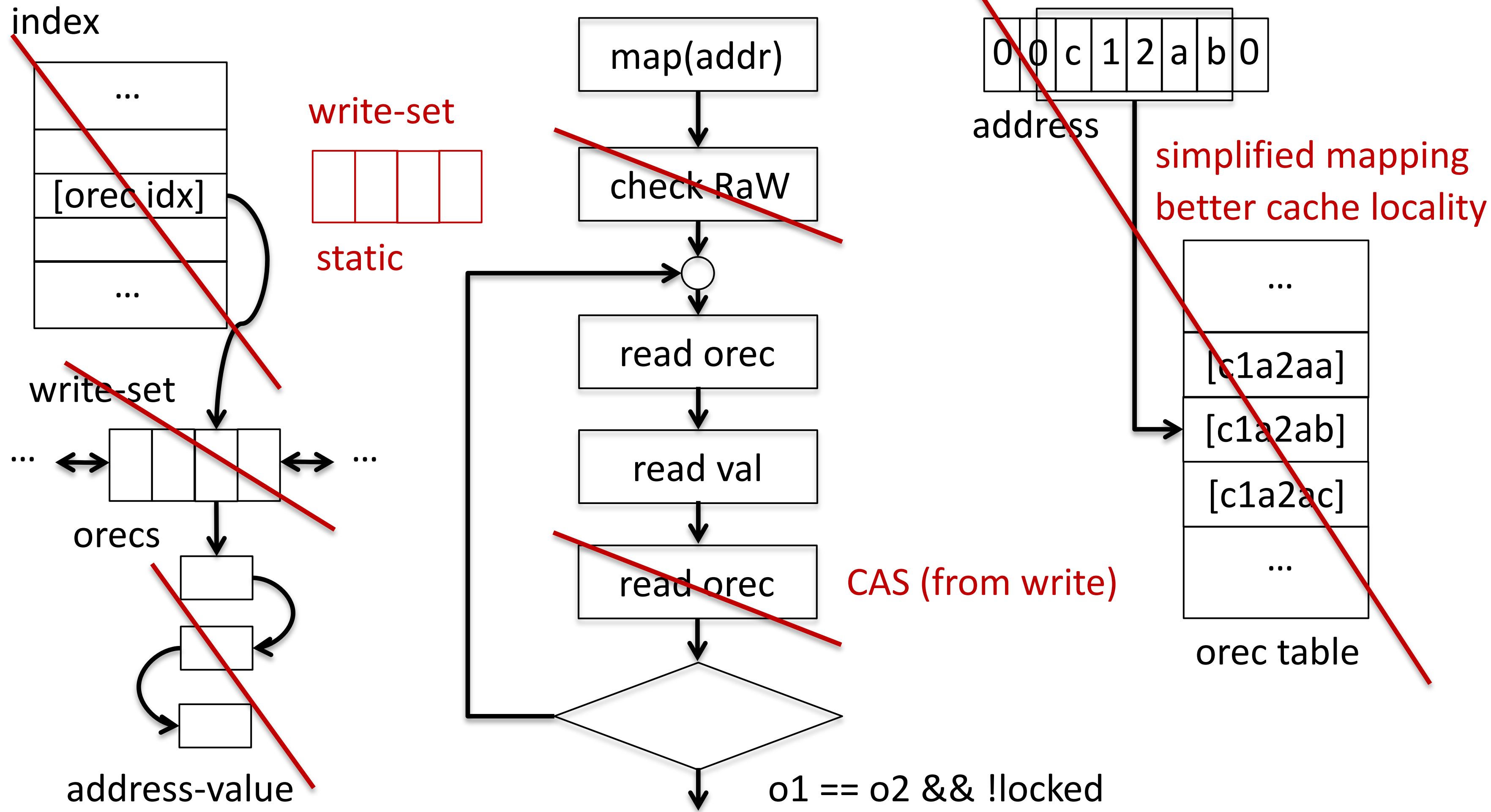
Pure value-based validation

…
W_1
W_2
…
W_3
W_4
…

Restrict values
Restrict access patterns
Single-word operations

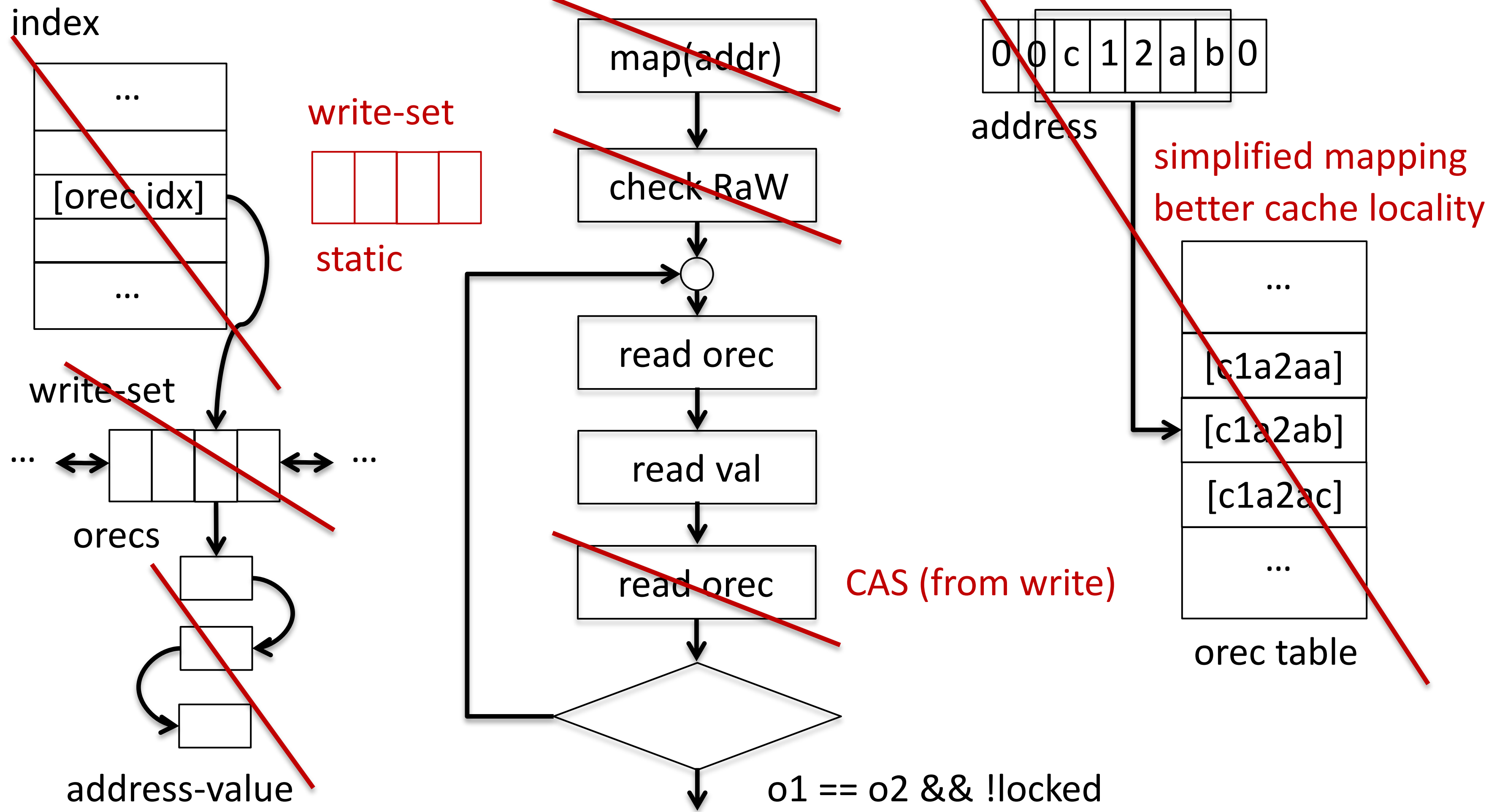
Microsoft Research

Value-based optimizations



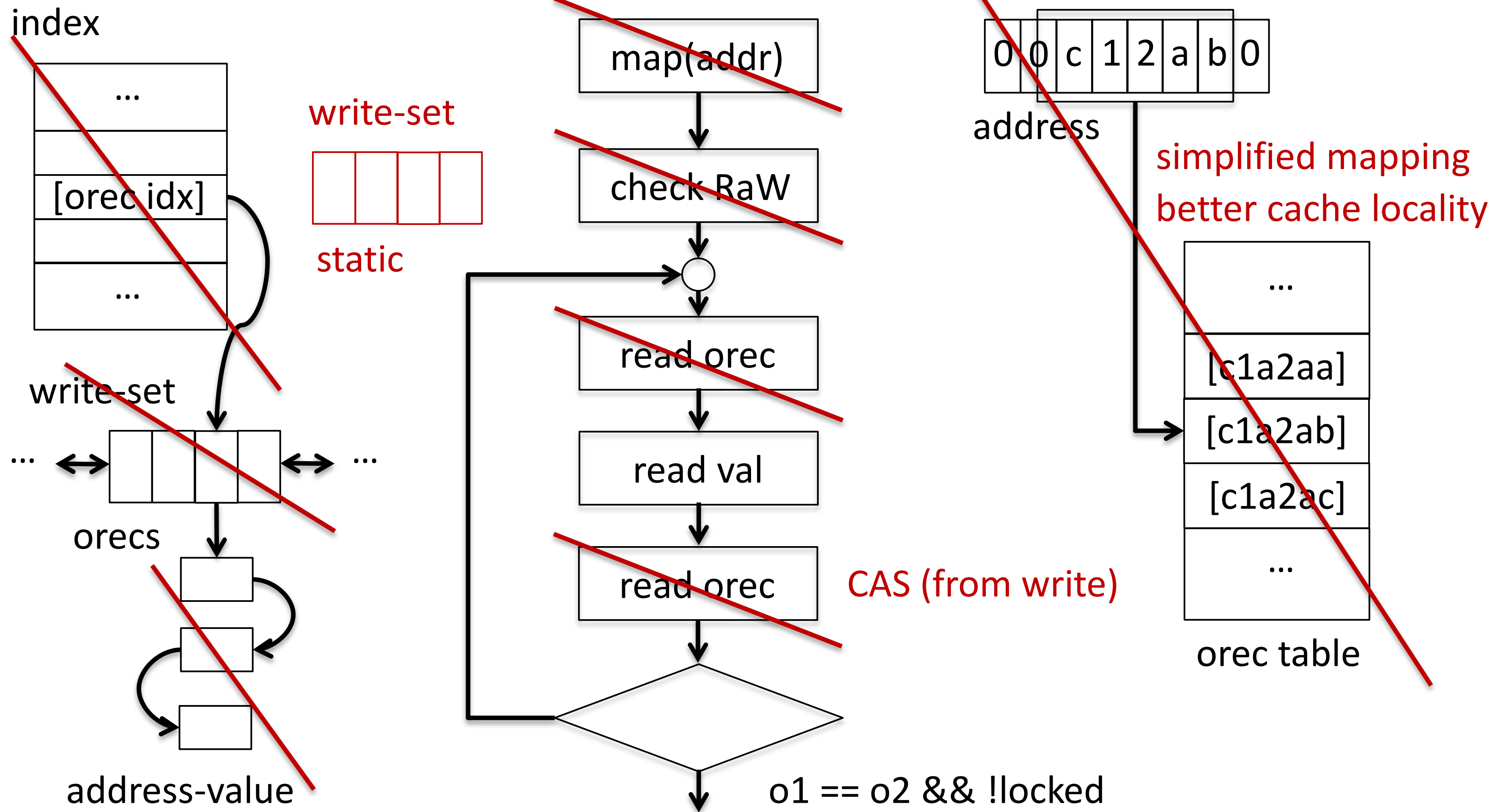
Microsoft Research

Value-based optimizations



Microsoft Research

Value-based optimizations



Microsoft Research

API changes

Microsoft Research

API changes

```
word_t ReadWord(TVar *addr);
void WriteWord(TVar *addr, word_t val);
```

```
word_t TVarToWord(TVar addr);
TVar WordToTVar(word_t val);
```


Microsoft Research

API changes

Transactions access TVar, not word_t

```
word_t ReadWord(TVar *addr);

void WriteWord(TVar *addr, word_t val);
```

```
word_t TVarToWord(TVar addr);

TVar WordToTVar(word_t val);
```


Microsoft Research

API changes

Transactions access TVar, not word_t

```
word_t ReadWord(TVar *addr);
void WriteWord(TVar *addr, word_t val);
```

Calls to update TVar non-transactionally

```
word_t TVarToWord(TVar addr);
TVar WordToTVar(word_t val);
```

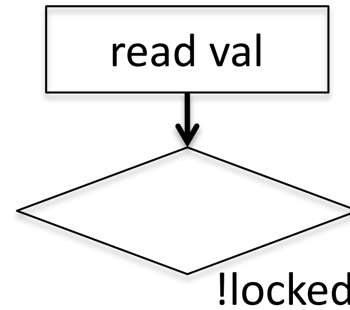

Microsoft Research

Pure value-based short read-write

read val

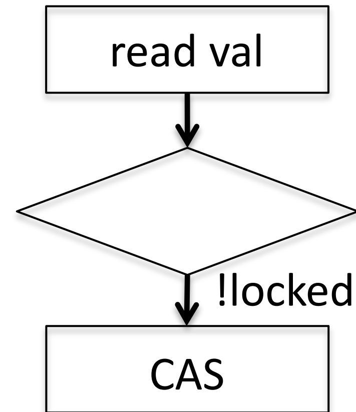
Microsoft Research

Pure value-based short read-write



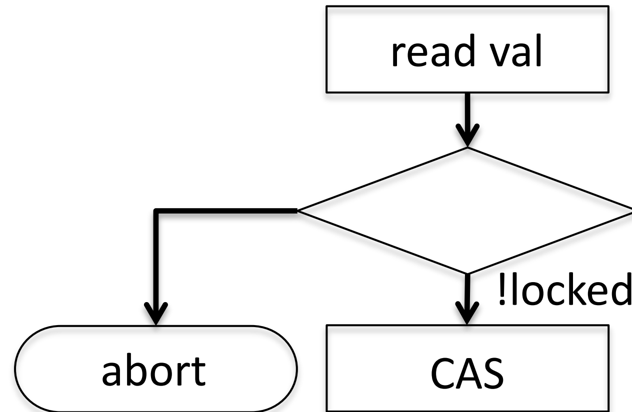
Microsoft Research

Pure value-based short read-write



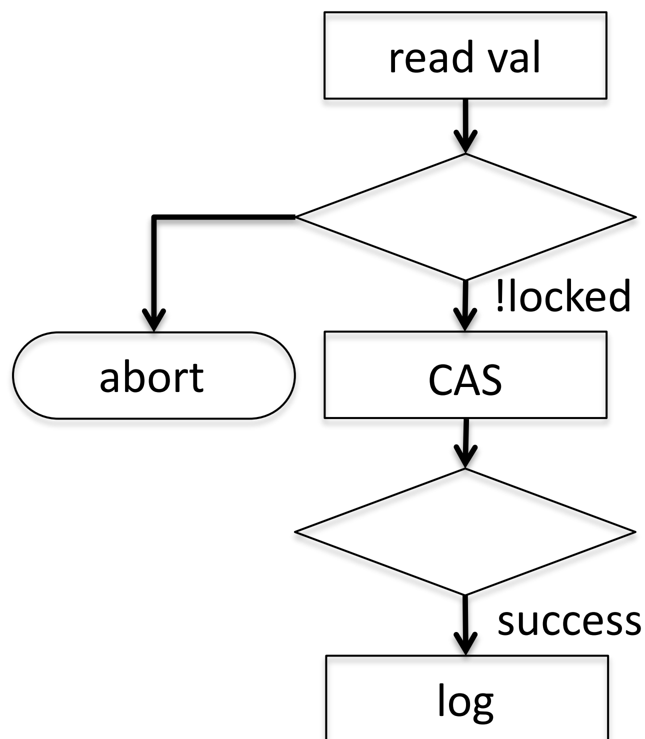
Microsoft Research

Pure value-based short read-write



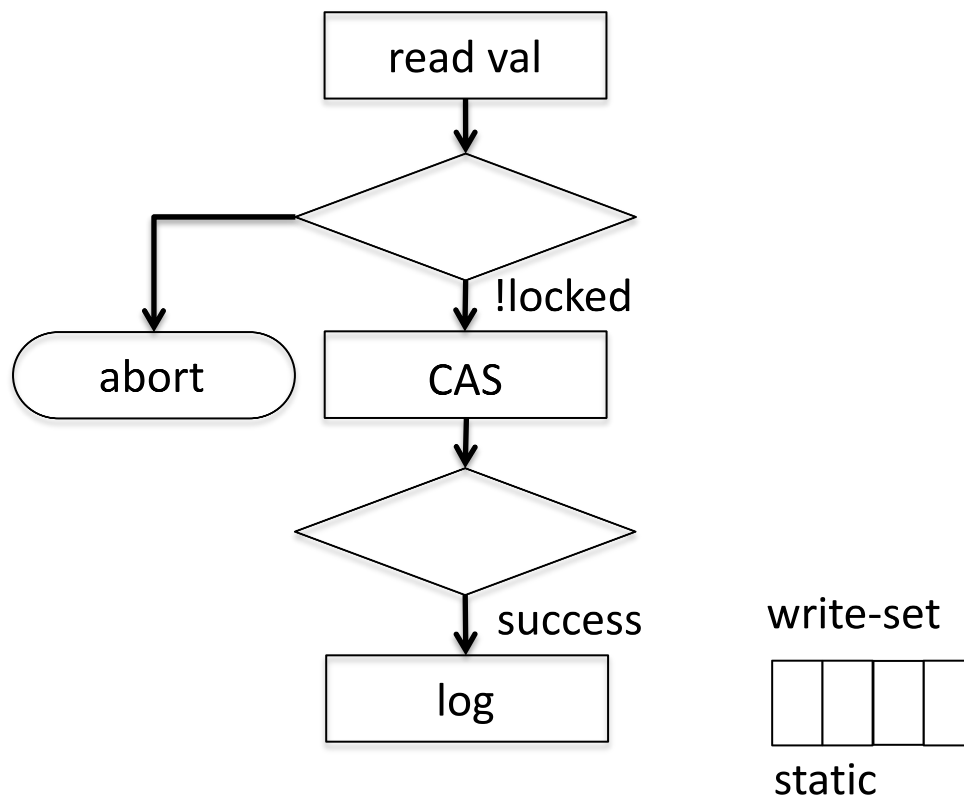
Microsoft Research

Pure value-based short read-write



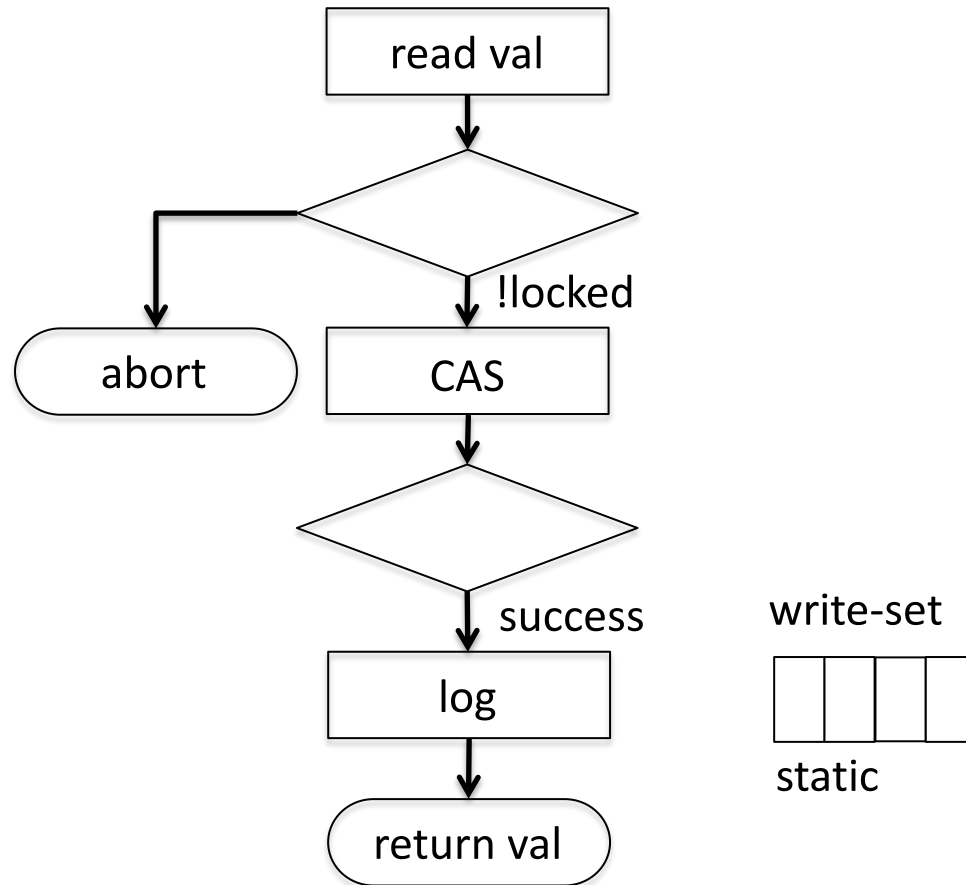
Microsoft Research

Pure value-based short read-write



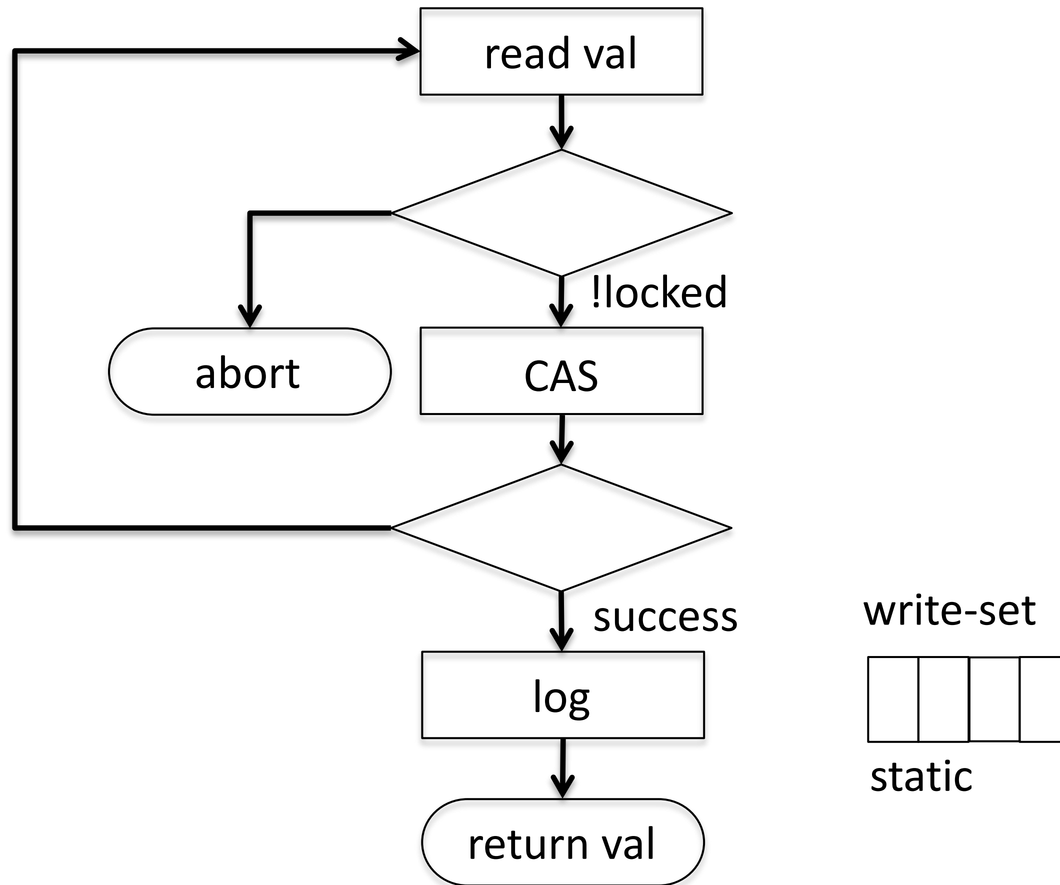
Microsoft Research

Pure value-based short read-write



Microsoft Research

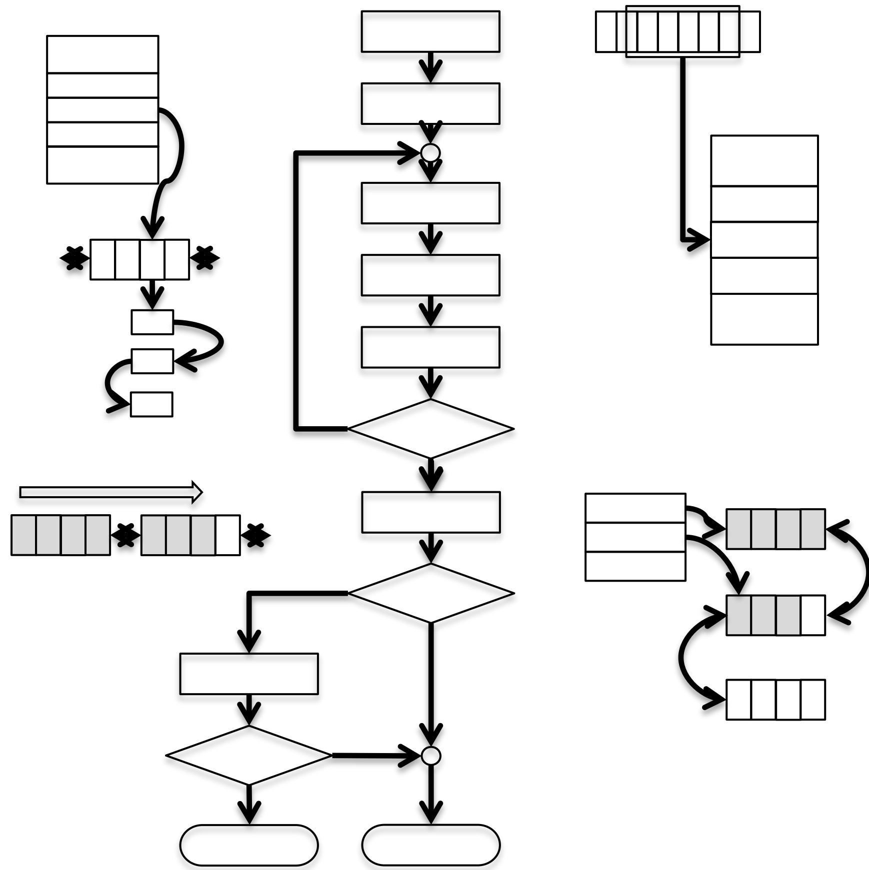
Pure value-based short read-write



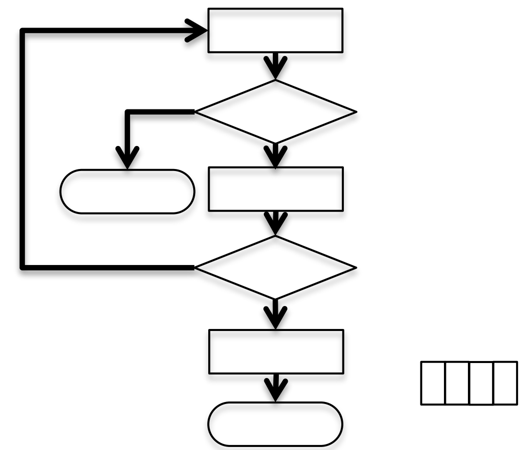
Microsoft Research

STM vs. SpecTM

STM



SpecTM



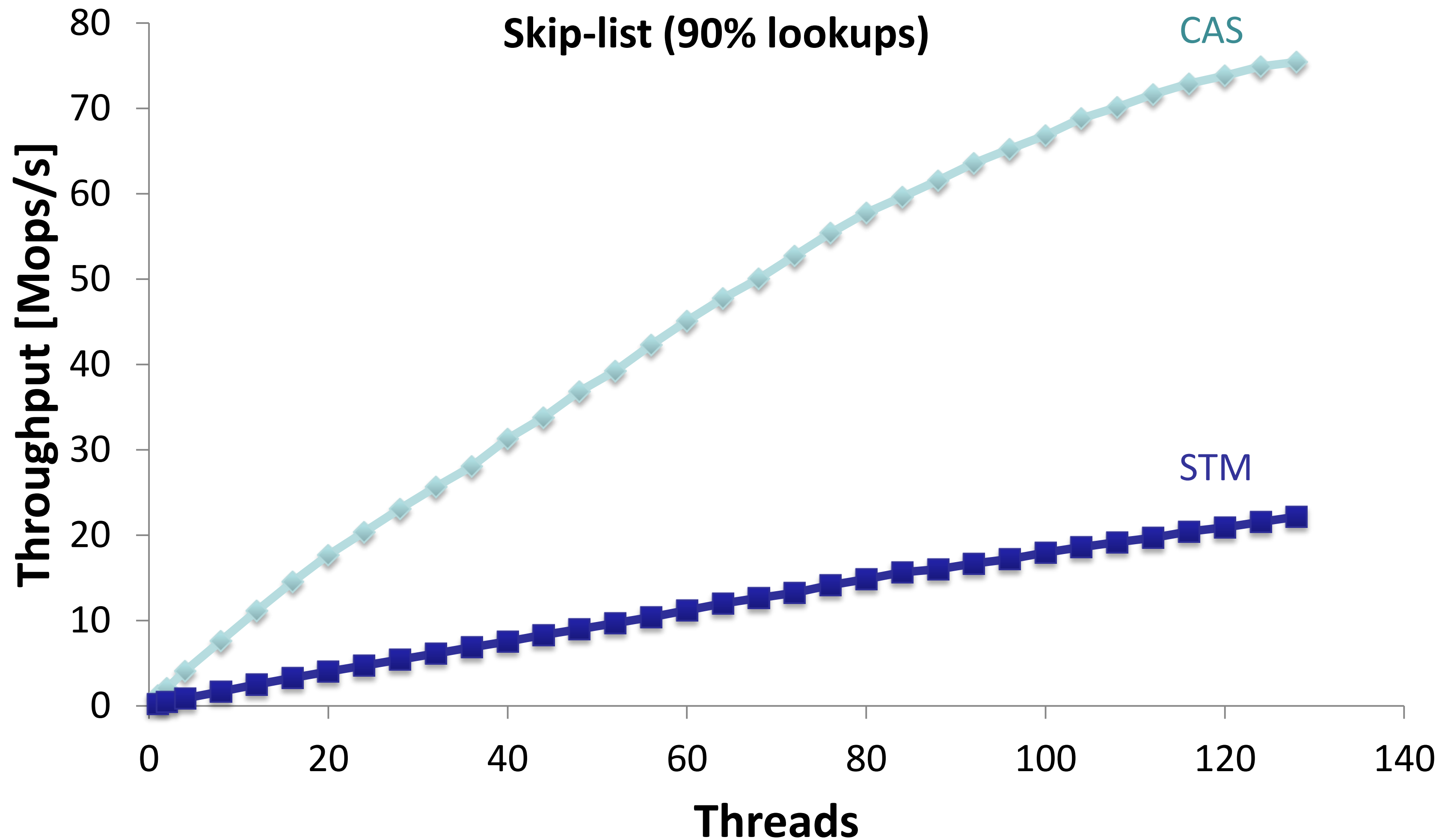
Microsoft Research

Overview

- STM overheads
- SpecTM
  - Short-transaction API
  - Collocating data and meta-data
  - In-word meta-data
- Evaluation

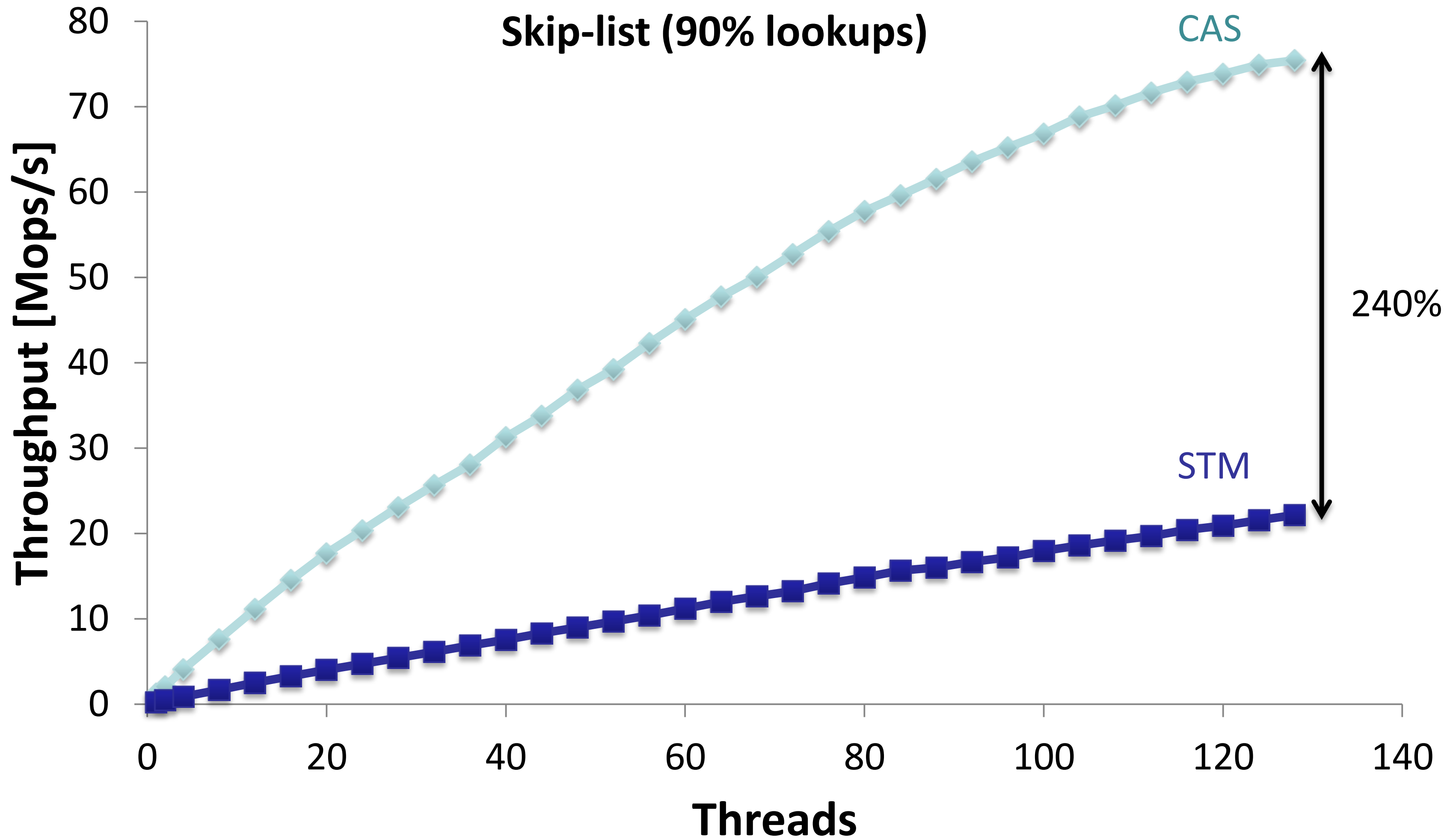
Microsoft Research

Evaluation



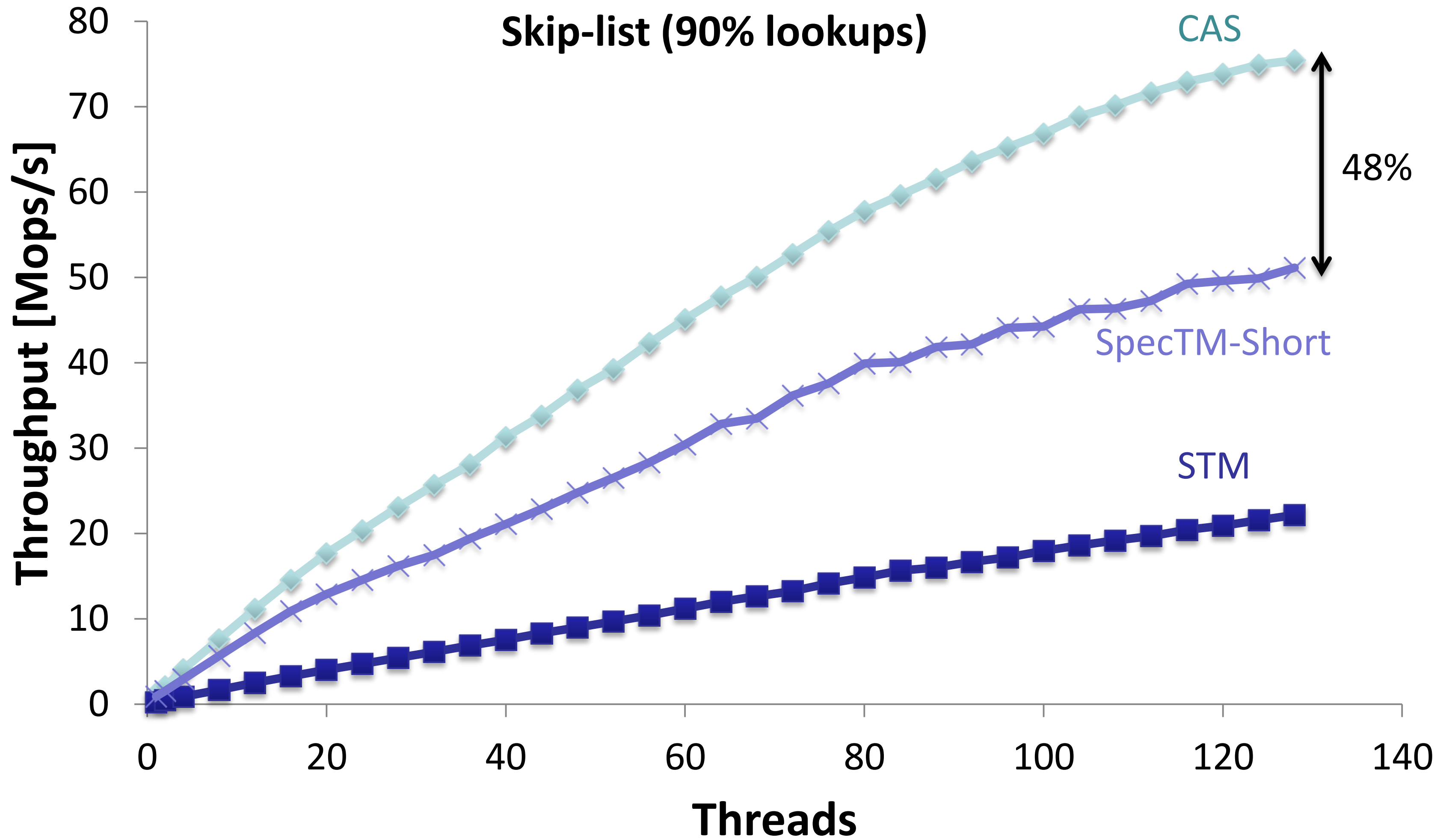
Microsoft Research

Evaluation



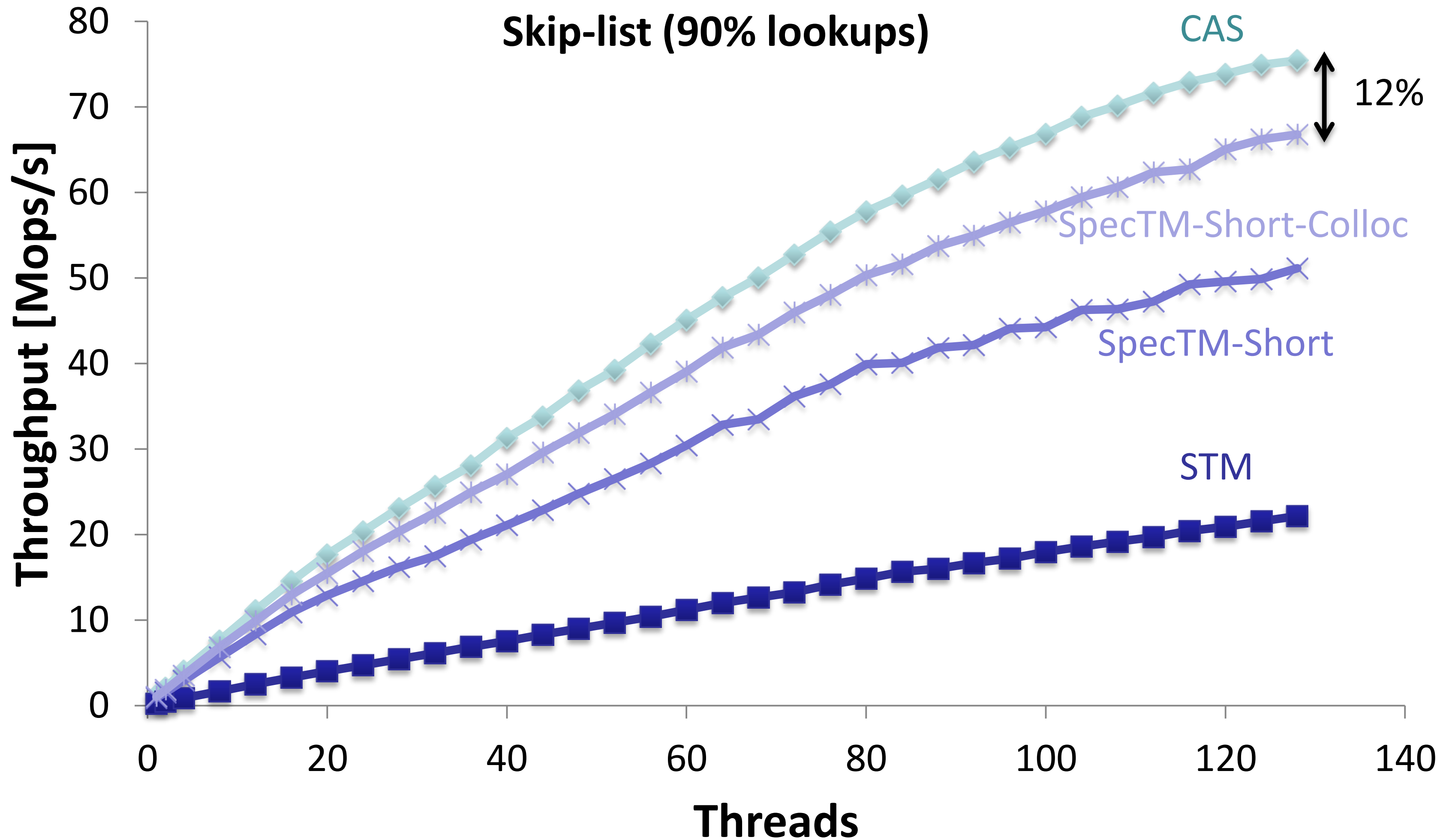
Microsoft Research

Evaluation



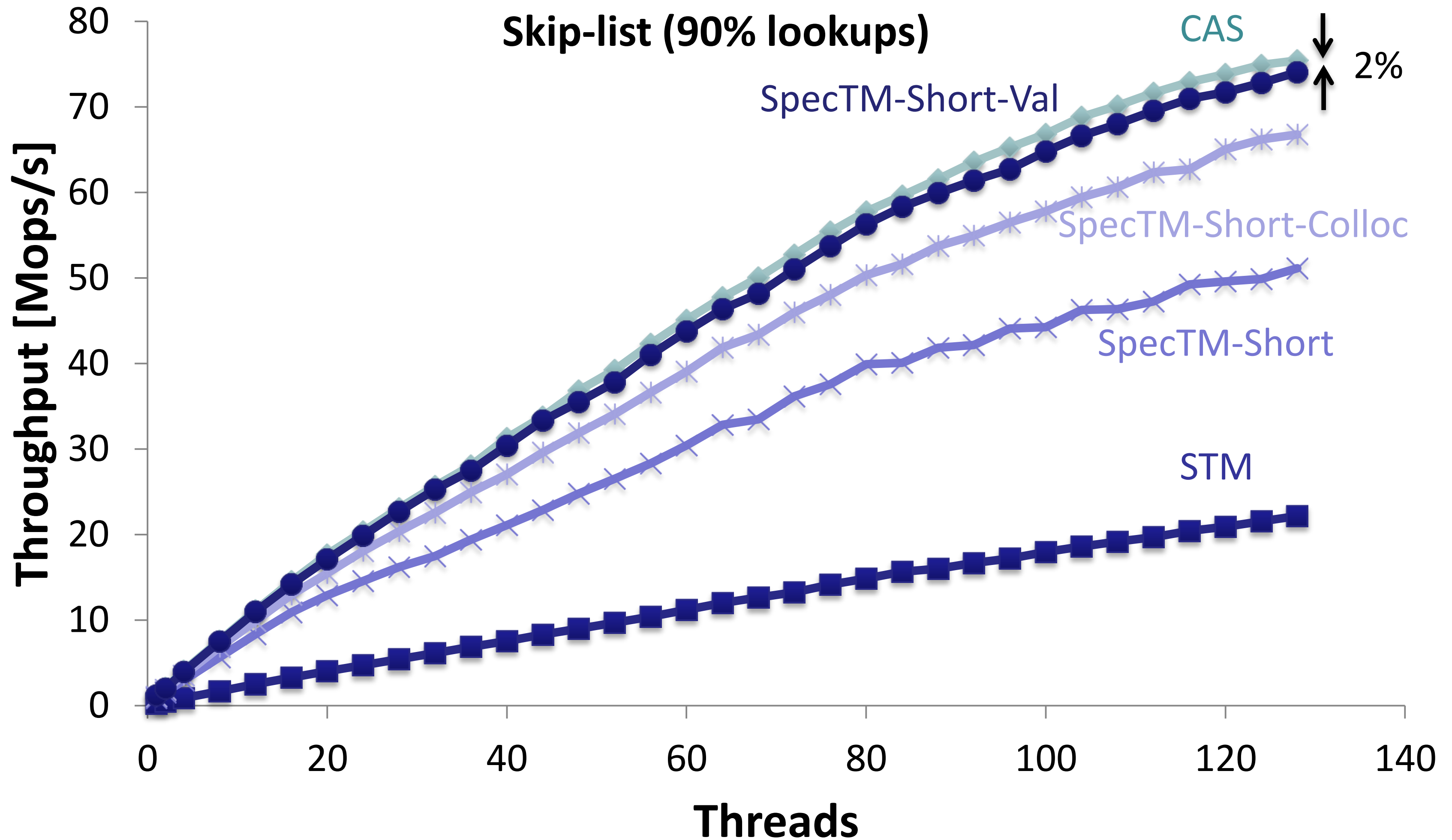
Microsoft Research

Evaluation



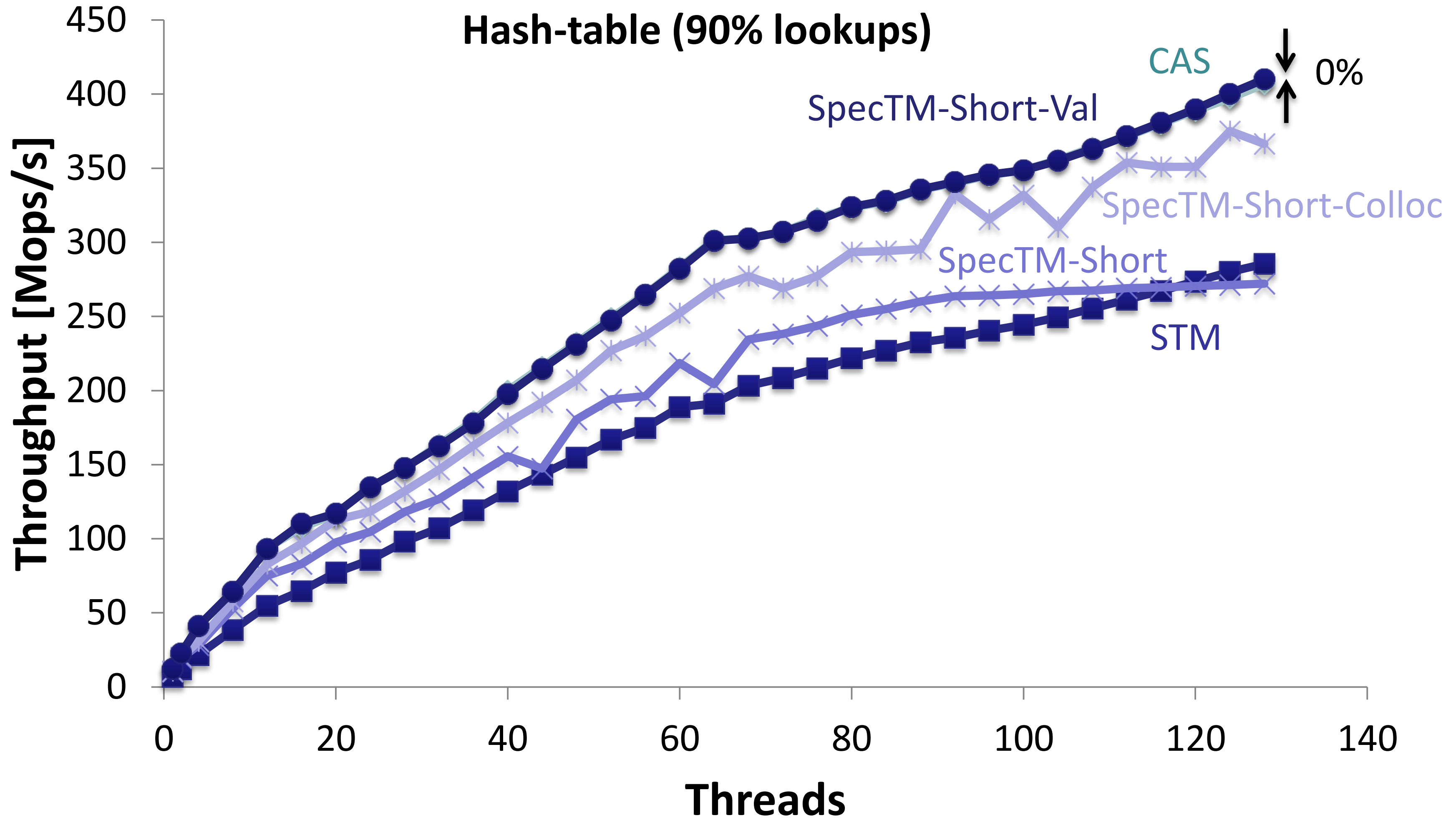
Microsoft Research

Evaluation



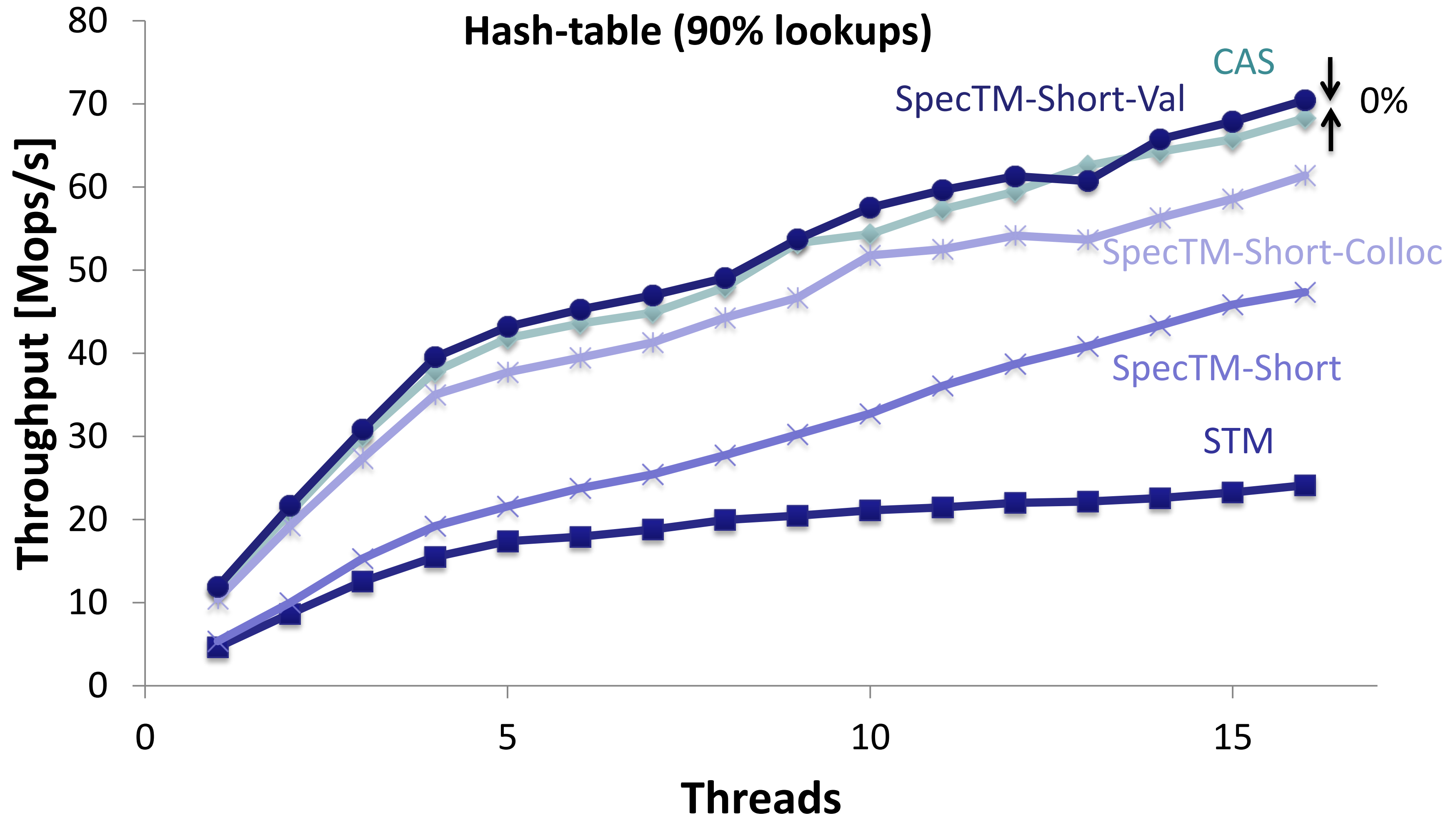
Microsoft Research

Evaluation



Microsoft Research

Evaluation



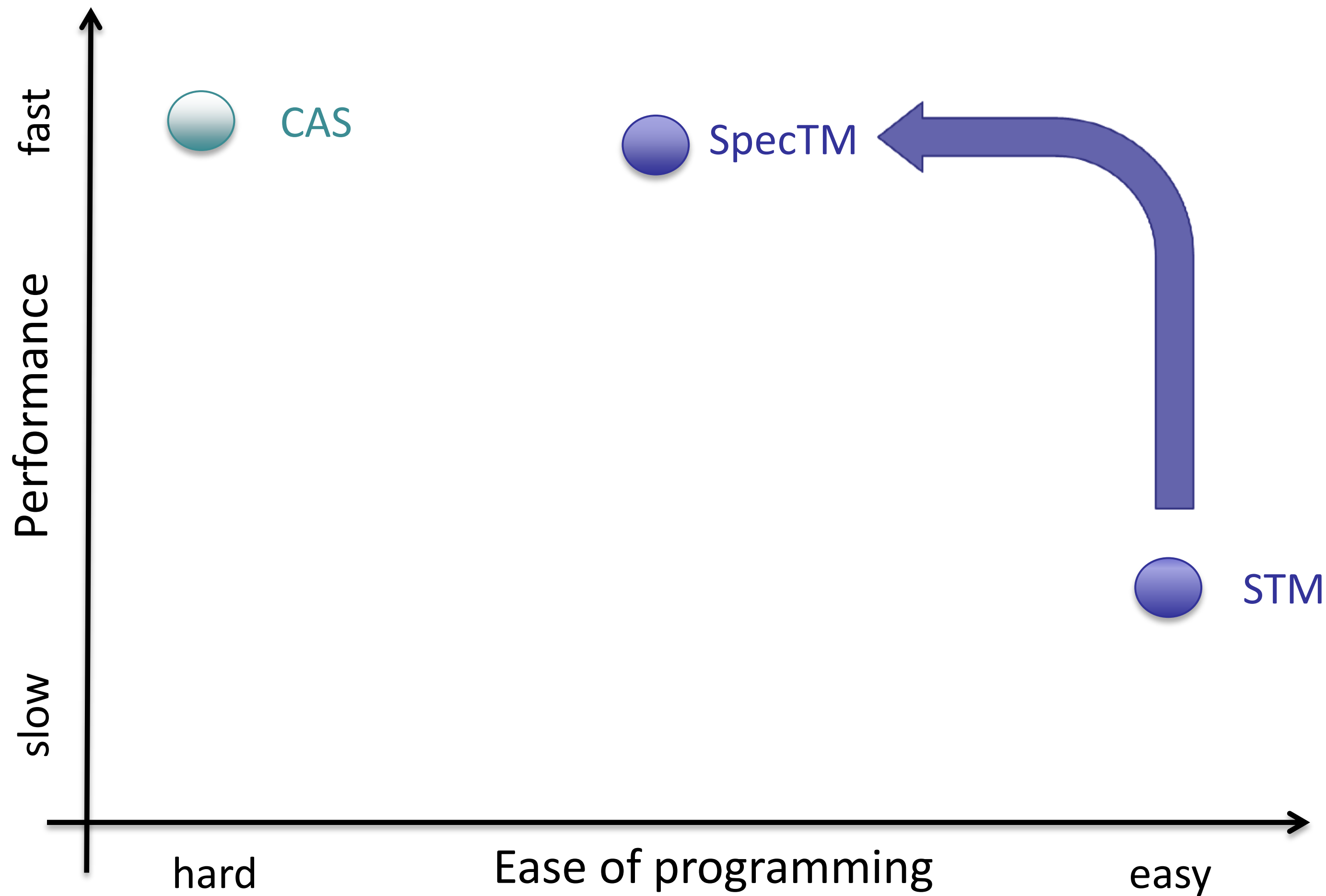
Microsoft Research

Conclusions

- Trade generality for performance in SpecTM
  - Restrict STM API
  - Control data layout
- STM-based data structures perform as well as the ones built directly from CAS
- Do we need SpecTM with upcoming Intel TSX?
  - Best-effort limited-size transactions

Microsoft Research

SpecTM: Specialized STM for Concurrent Data Structures



Microsoft Research

©2011 Microsoft Corporation. All rights reserved.