Valor: Efficient, Software-Only Region Conflict Exceptions

Swarnendu Biswas, Minjia Zhang, and
Michael D. Bond
Ohio State University
Brandon Lucia
Carnegie Mellon University

OOPSLA 2015

A Simple C++ Program

```
X* x = NULL;
bool done= false;
```

Thread T1

```
x = new X();
done = true;
```

Thread T2

```
if (done) {
  x->func();
}
```

A Simple C++ Program

ad T2

Data Races

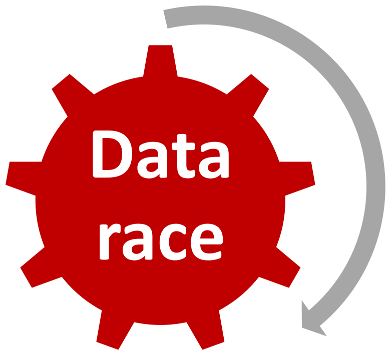
```
X* x = NULL;
bool done= false;
```

Thread T1

```
x = new X();
done = true;
```

Thread T2

```
if (done) {
  x->func();
}
```



Data Races are Evil

No semantic guarantees	Lack of semantic guarantees make software unsafe
Complicates language specifications	Challenging to reason about correctness for racy executions
Indicates other concurrency errors	Leads to atomicity, order or sequential consistency violations

Catch-Fire Semantics

C++ treats data races as errors

```
  X* x = NULL;
bool done= false;
```

Thread T1

```
x = new X();
done = true;
```

Thread T2

```
if (done) {
  x->func();
}
```

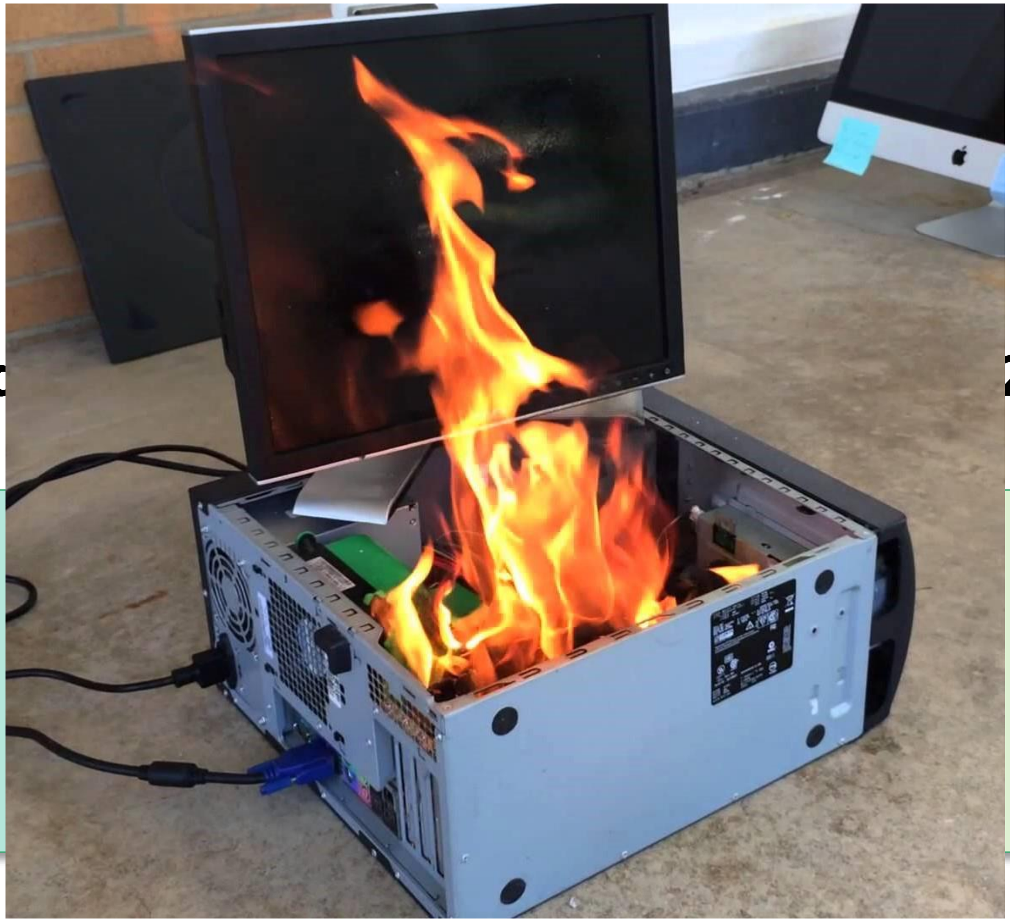
Catch-Fire Semantics

C++ treats data races as errors

Threa

2

```
x = new X();
done = true;
```



Catch-Fire Semantics

C++ treats data races as errors

Threa

```
x = new X();
done = true;
```

A Java Example

Thread T1

```
X = new Object();
done = true;
```

Thread T2

```
while (!done) {}
X.compute();
```

A Java Example

Java tries to assign semantics, which are unsatisfactory

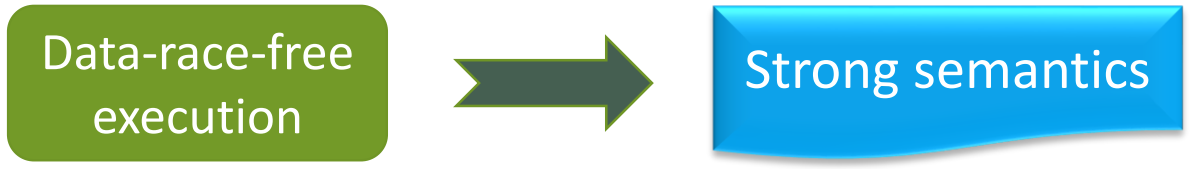
Thread T1

```
X = new Object();
done = true;
```

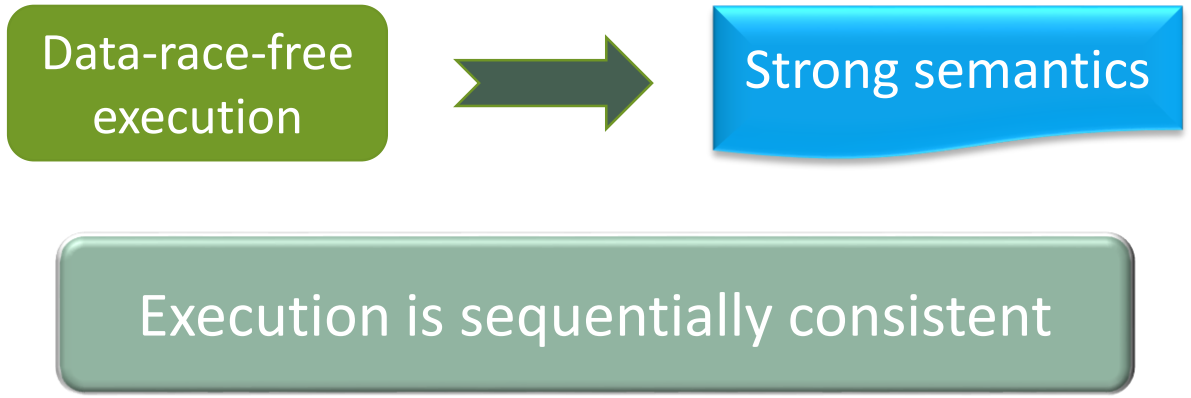
Thread T2

```
while (!done) {}
X.compute();
```

C++ and Java Memory Models



C++ and Java Memory Models



C++ and Java Memory Models

Data-race-free execution → Strong semantics

Execution is sequentially consistent

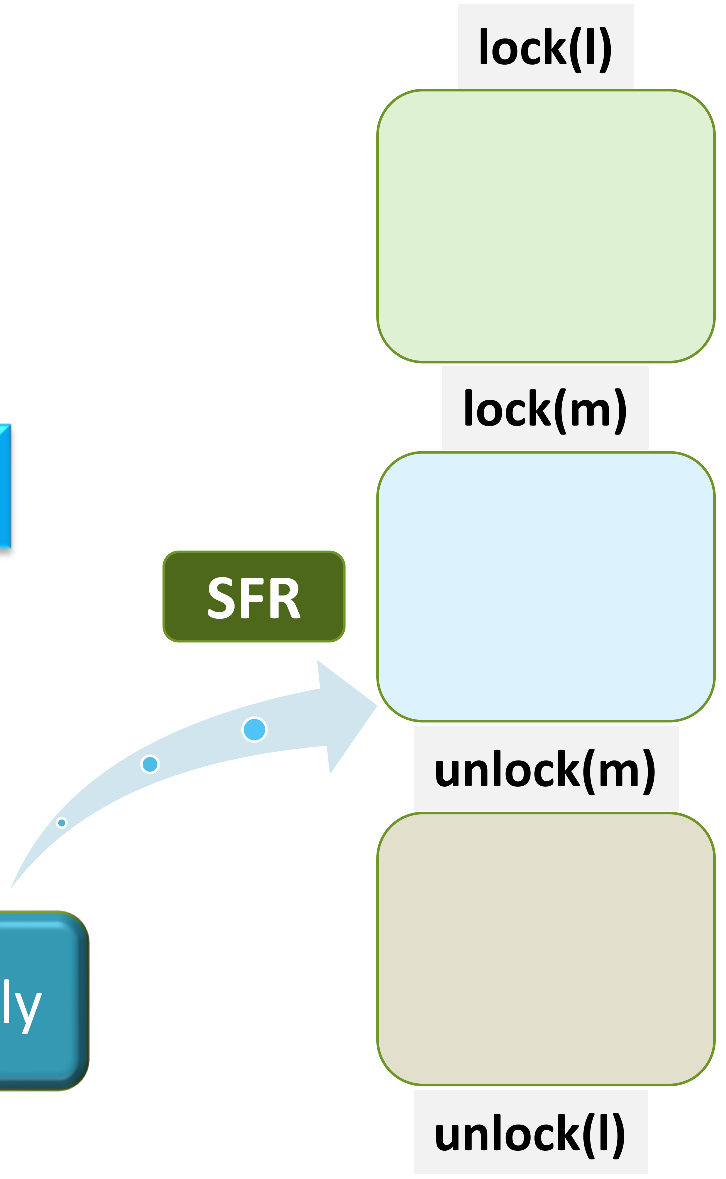
Synchronization-free regions execute atomically

C++ and Java Memory Models

Data-race-free execution → Strong semantics

Execution is sequentially consistent

Synchronization-free regions execute atomically

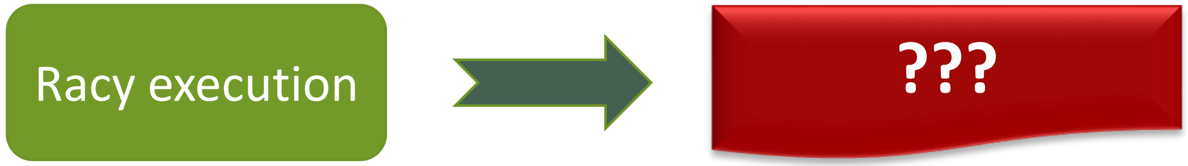


C++ and Java Memory Models

But what about data races?

C++ and Java Memory Models

But what about data races?



Need for Stronger Memory Models

Adve and Boehm, CACM 2010

"The inability to define reasonable semantics for programs with data races is not just a theoretical shortcoming, but a fundamental hole in the foundation of our languages and systems."

Need for Stronger Memory Models

Adve and Boehm, CACM 2010

“The inability to define reasonable semantics for programs with data races is not just a theoretical shortcoming, but a fundamental hole in the foundation of our languages and systems.”

“We call upon software and hardware communities to develop languages and systems that enforce data-race-freedom, ...”

Outline

- Programming language memory models and data races
- **Data race and region conflict exceptions model**
- **Valor: Our contribution**
- **Evaluation**

Data Race

Thread T1

```
X = new Object();
done = true;
```

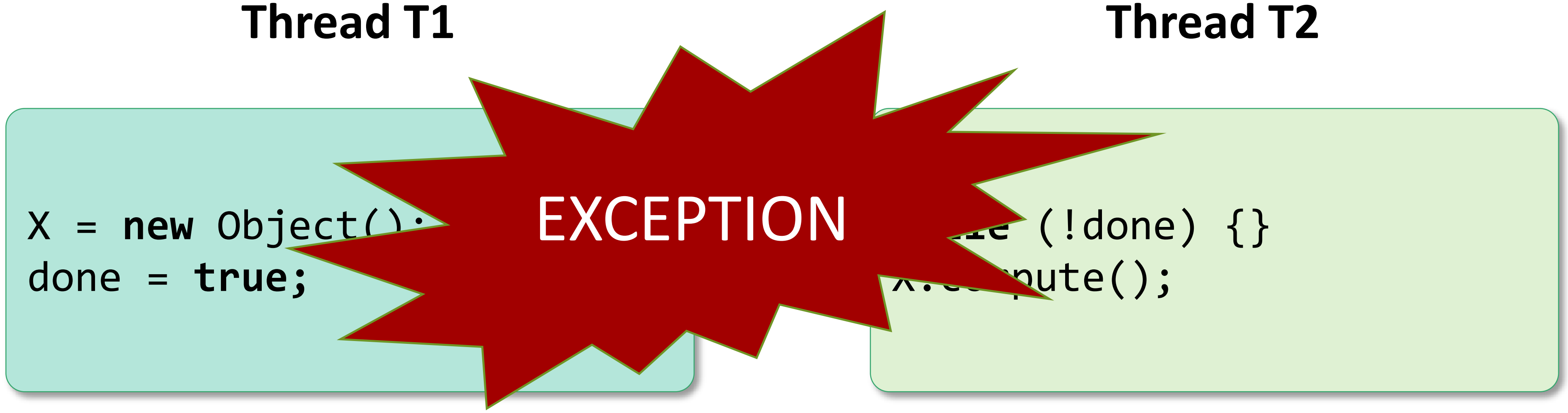
Thread T2

```
while (!done) {}
X.compute();
```

Data Race Exceptions

Thread T1

Thread T2

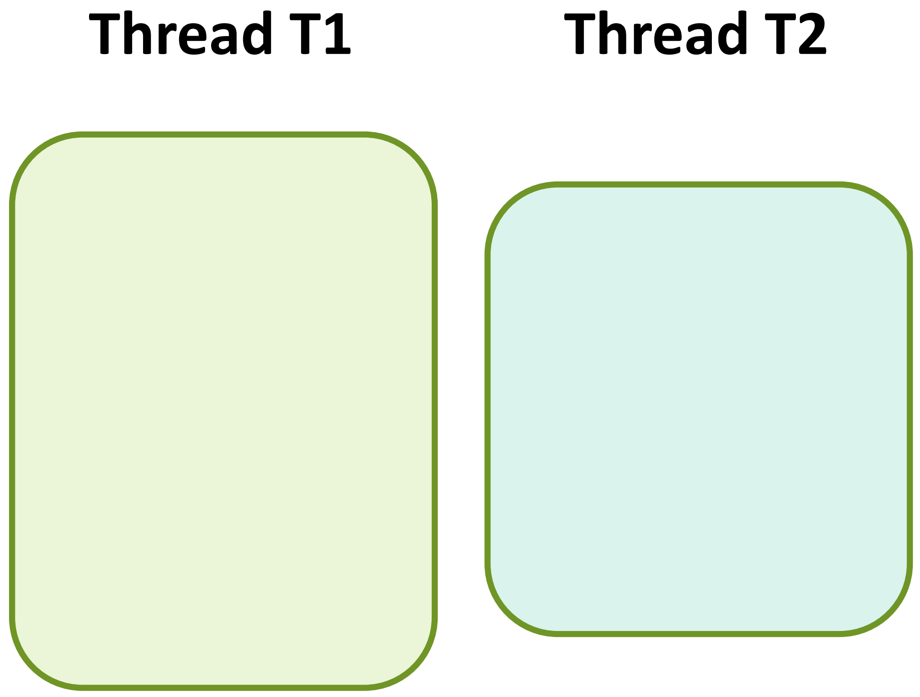


```
X = new Object();
done = true;
```

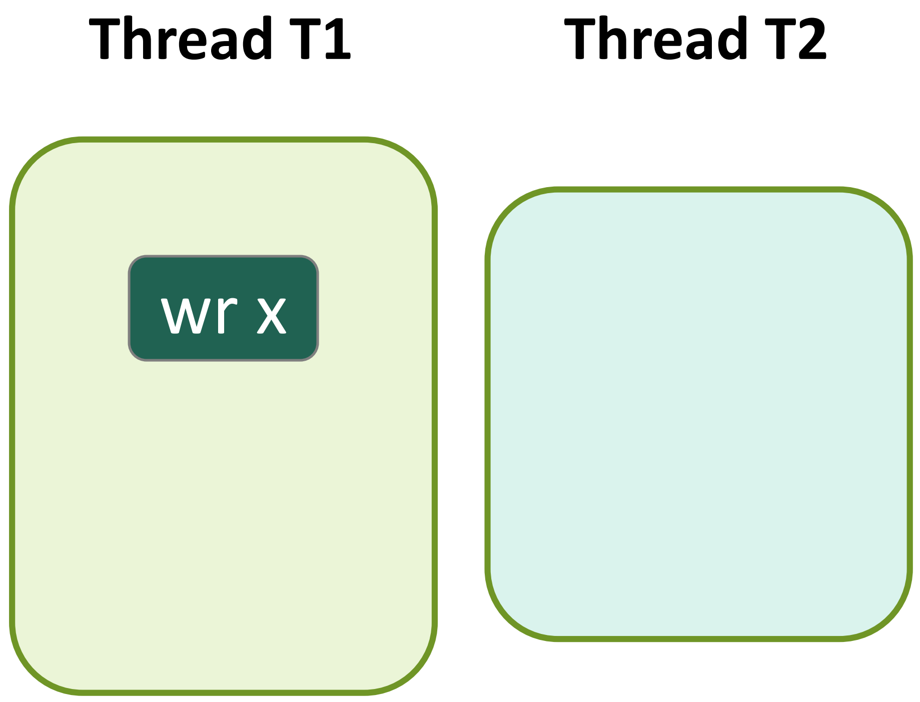
```
le (!done) {}
X.[illegible]pute();
```

REGION CONFLICT EXCEPTIONS MODEL

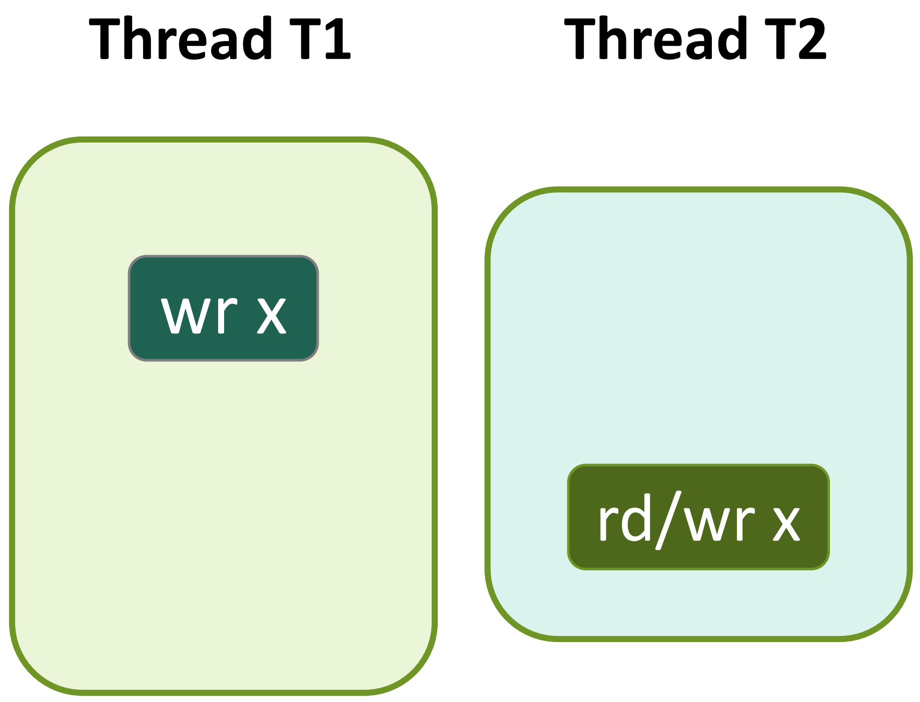
Region Conflict



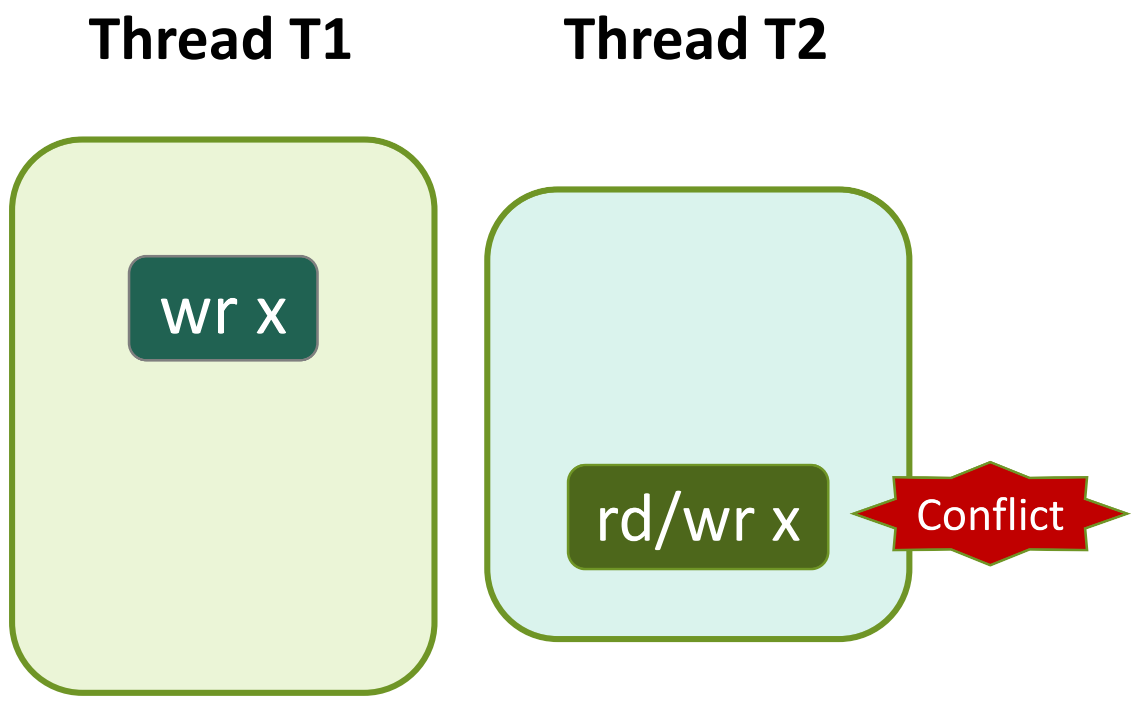
Region Conflict



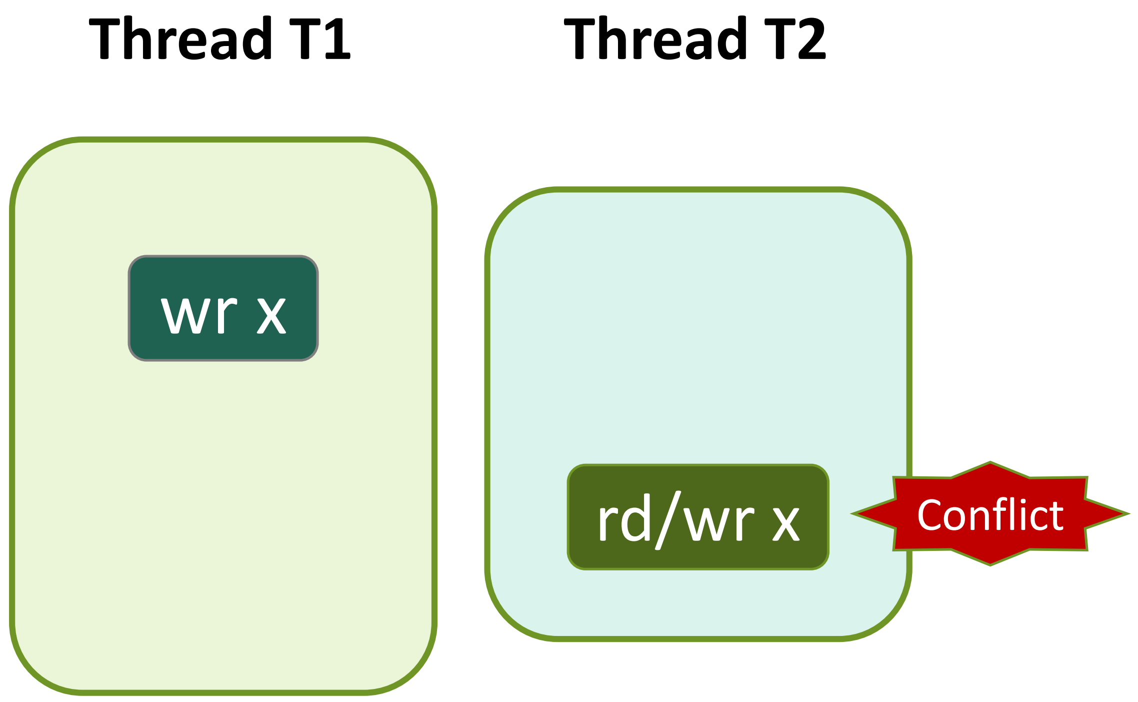
Region Conflict



Region Conflict

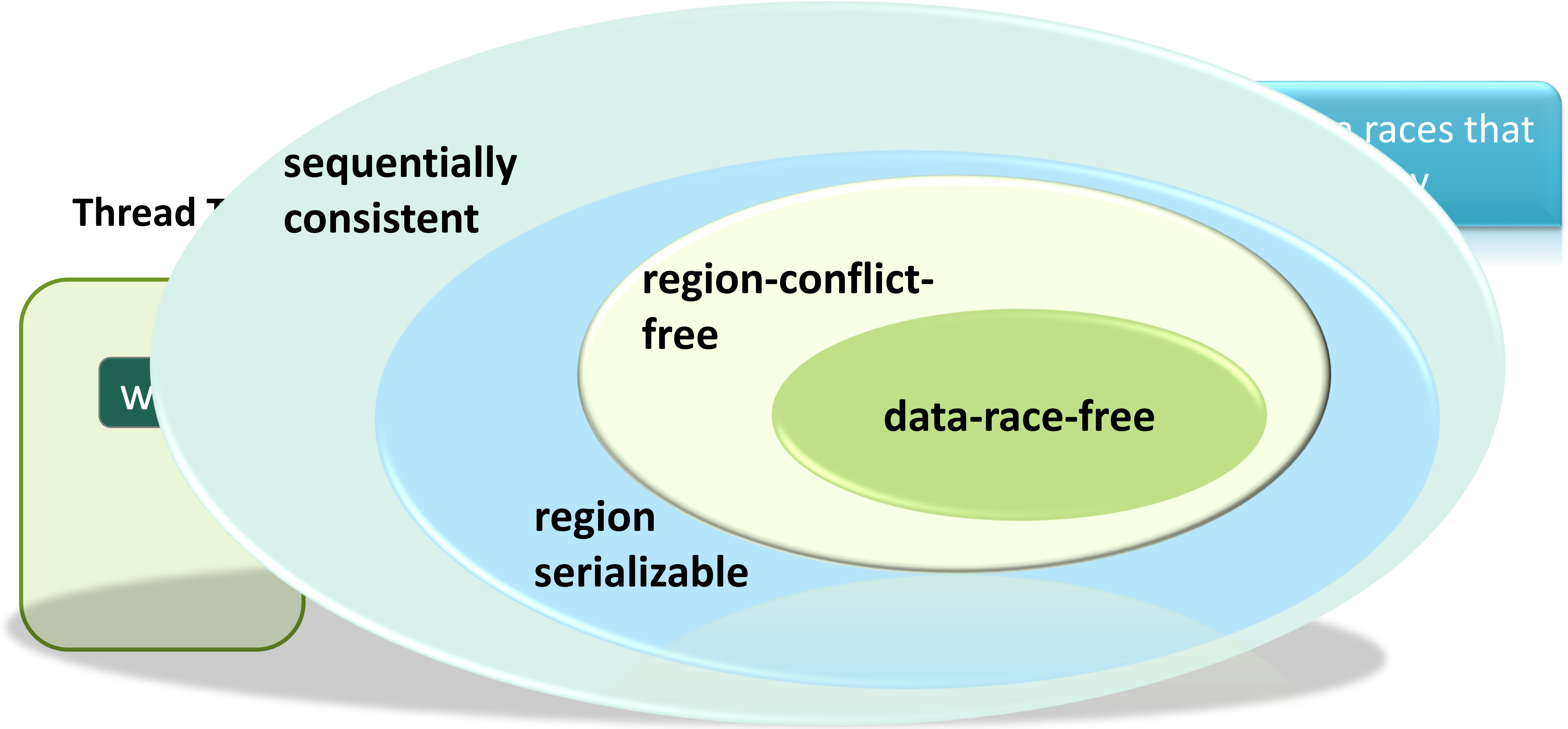


Region Conflict



Reports a subset of true data races that violate region serializability

Execution Models



Region Conflict Exception Model

GOAL

Develop a practical region conflict detection technique

Region Conflict Detection

Hardware customizations required for good performance

- ❖ Limited by resources and applicability
    - Needs extensive modifications and is unscalable[1]
    - Detects serializability violations of bounded regions[2]

1. Lucia et al. Conflict Exceptions: Simplifying Concurrent Language Semantics With Precise Hardware Exceptions for Data-Races. ISCA 2010.
2. Marino et al. DRFx: A Simple and Efficient Memory Model for Concurrent Programming Languages. PLDI 2010.

Outline

- Programming language memory models and data races
- Data race and region conflict exceptions model
- **Valor: Our contribution**
- **Evaluation**

Valor: Efficient, Software-Only Region Conflict Detector

Elides tracking last readers, only tracks last writer

Detect read-write conflicts lazily

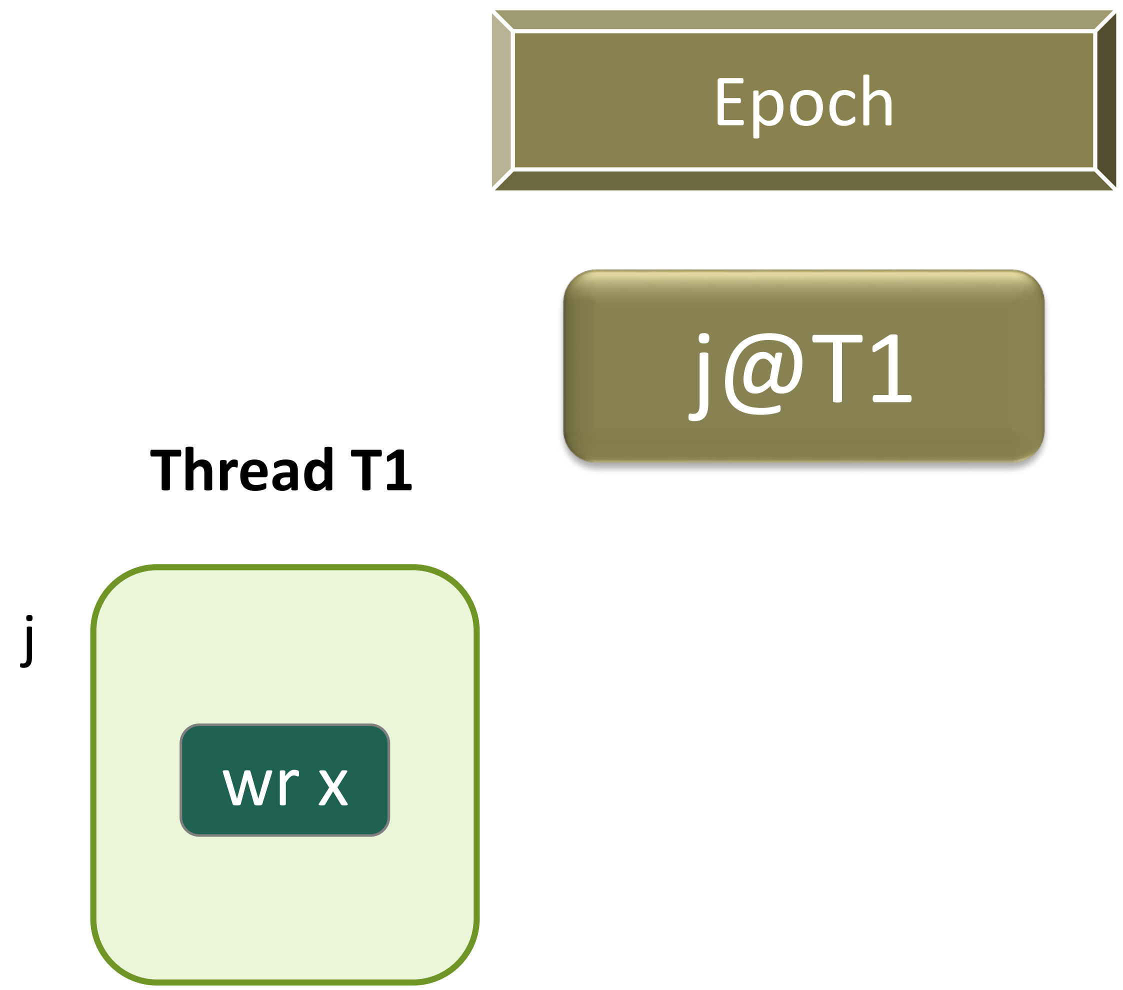
Valor: Efficient, Software-Only Region Conflict Detector

- Tracking last writers
- Detecting read-write conflicts lazily
- Impact of lazy conflict detection

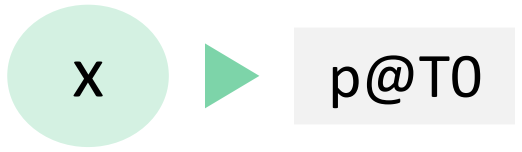
Tracking Last Writer

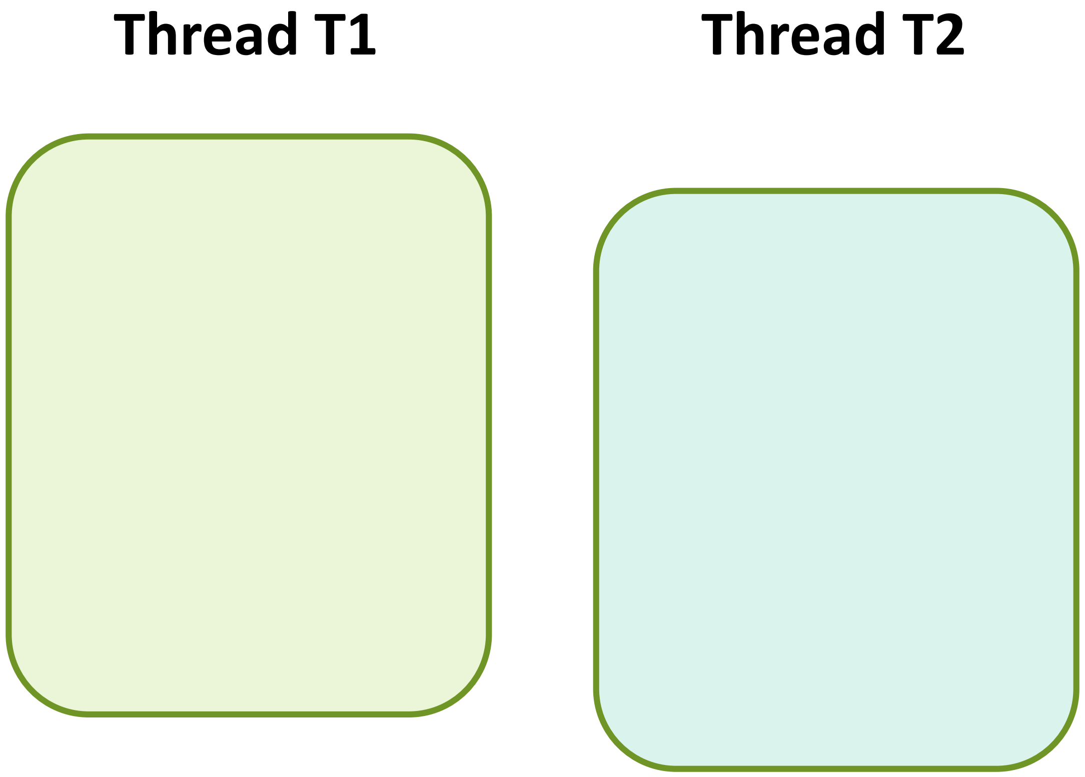
Per-Variable Metadata

Has an ongoing region updated x?

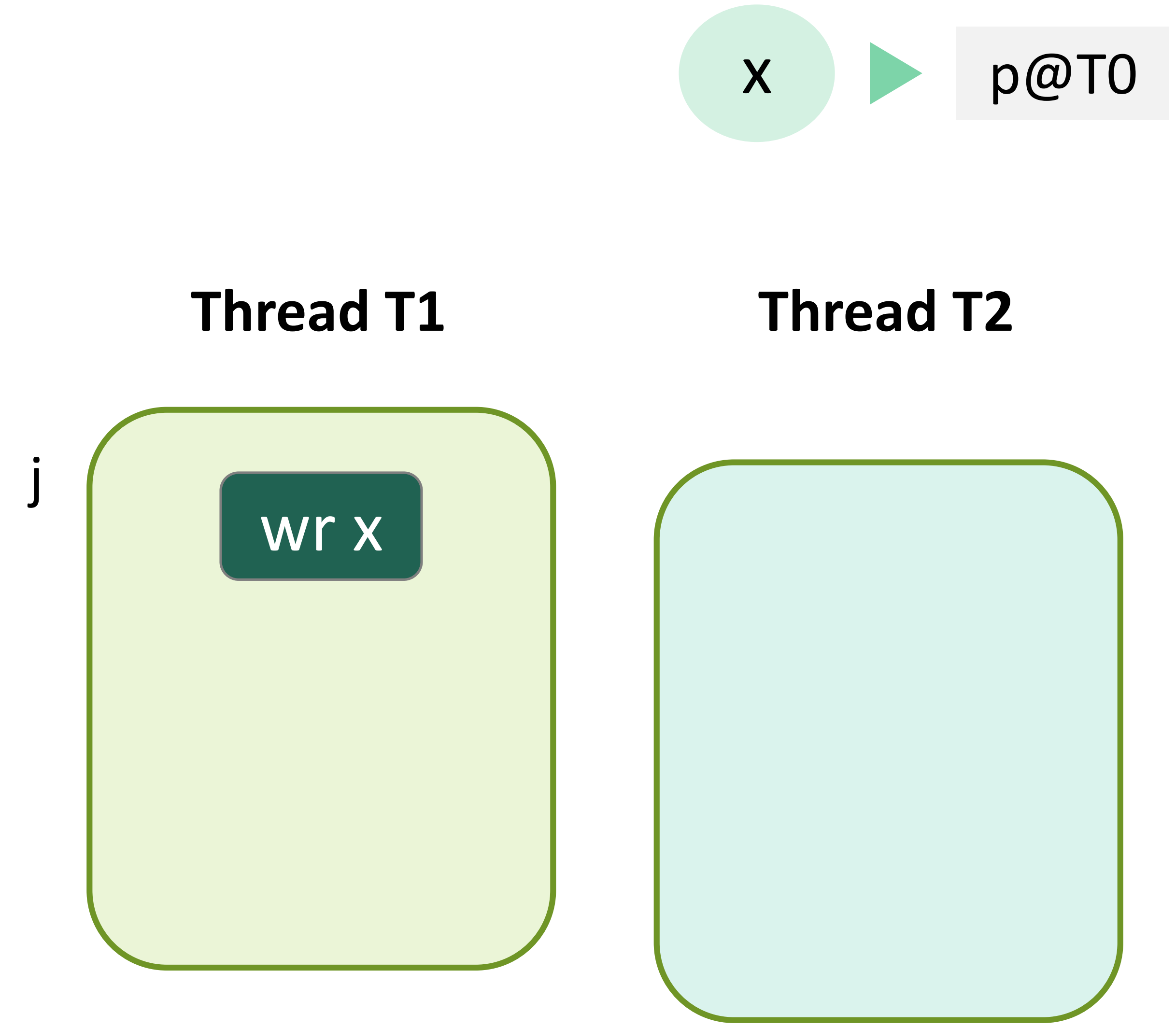


Tracking Last Writer



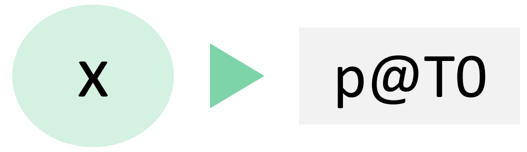


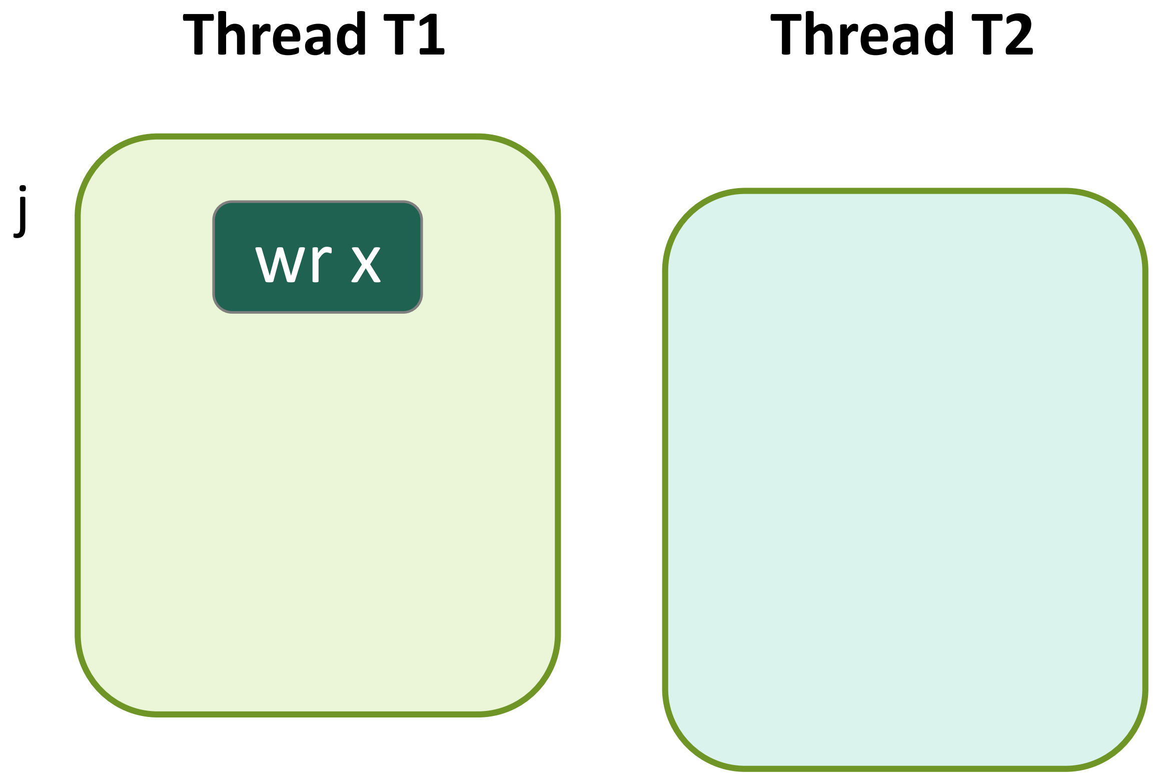
Tracking Last Writer



Tracking Last Writer

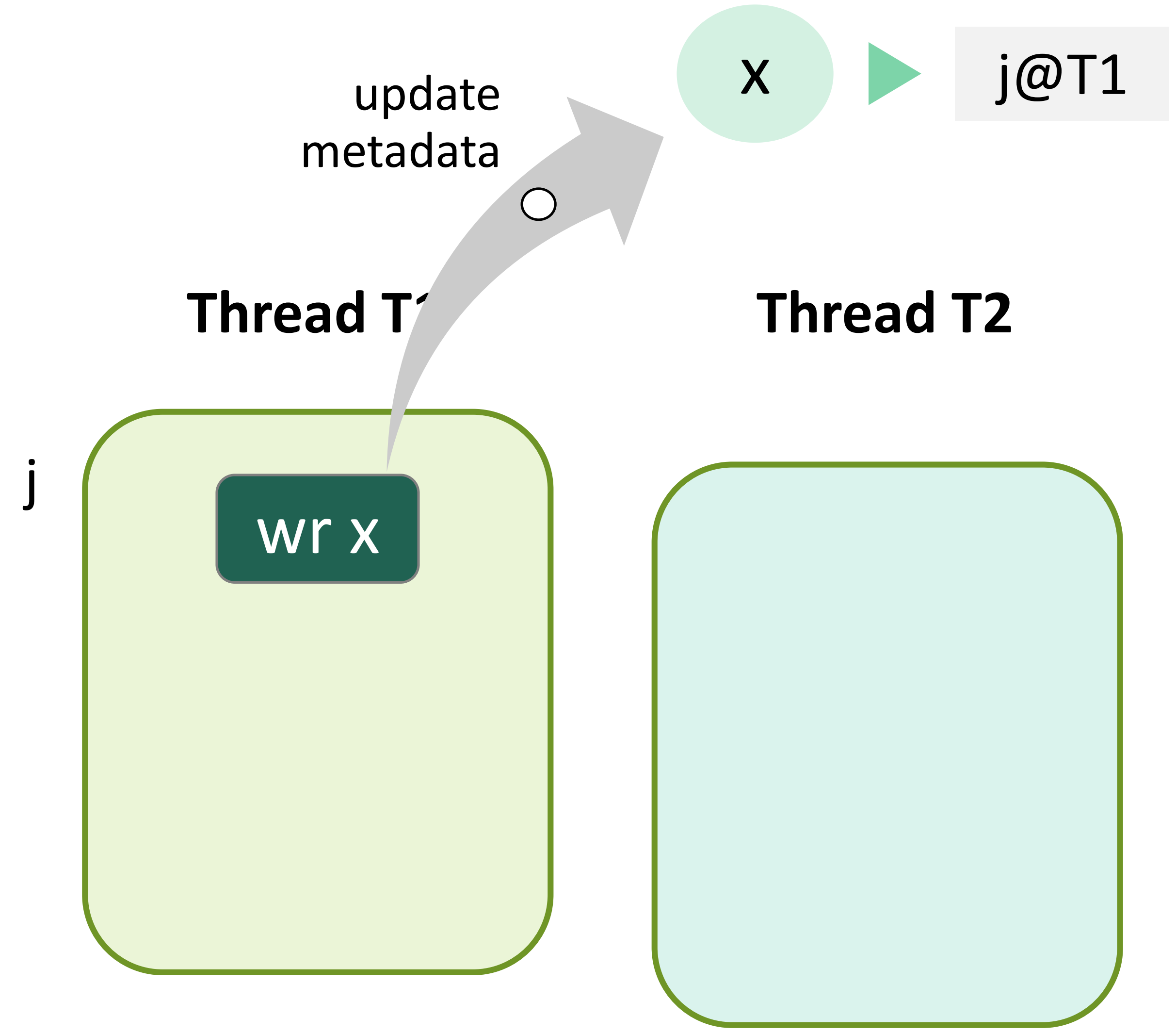
Track last writer



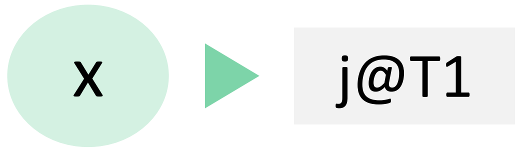


Tracking Last Writer

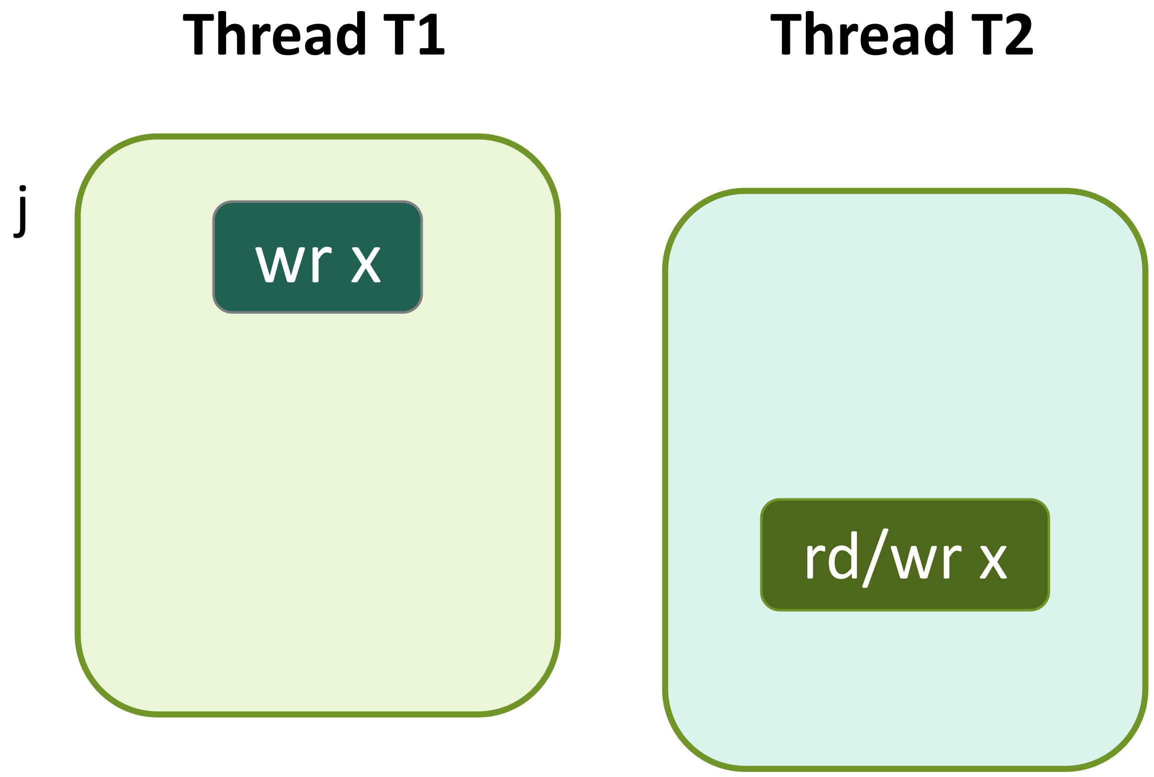
Track last writer



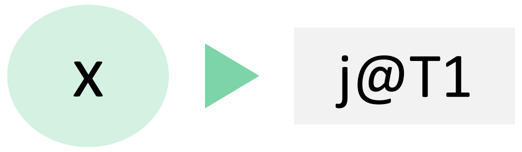
Tracking Last Writer



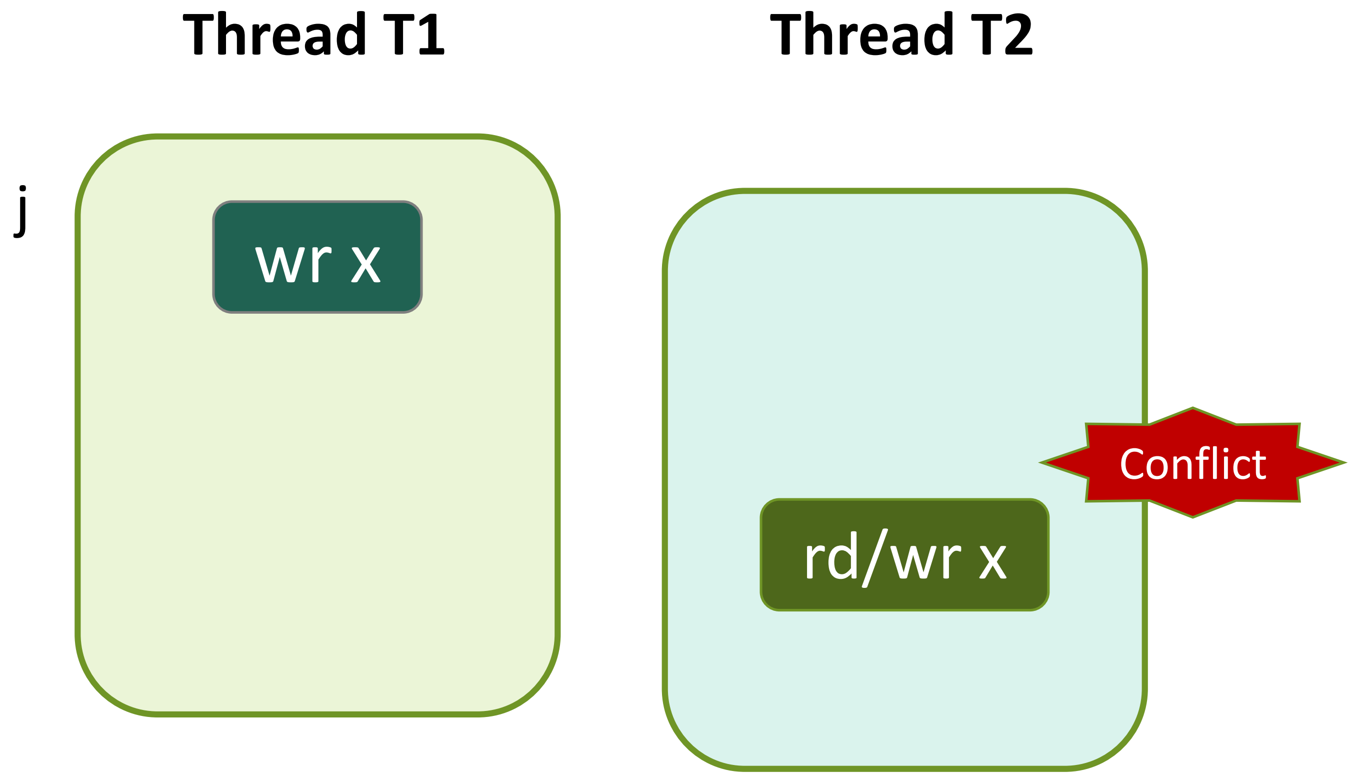
Track last writer



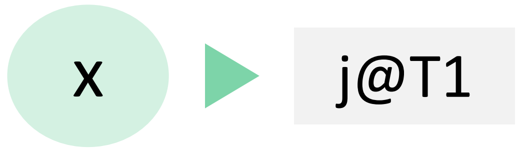
Tracking Last Writer



Track last writer

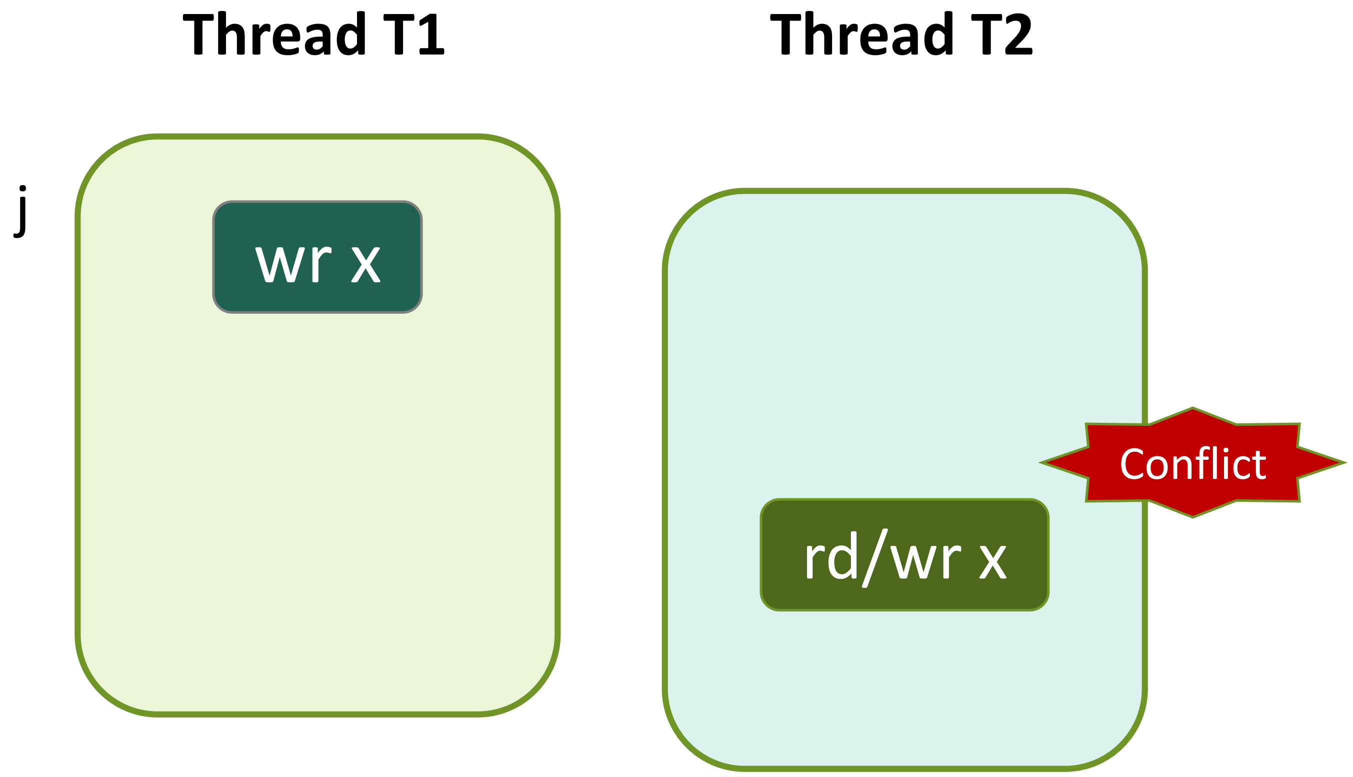


Tracking Last Writer



Track last writer

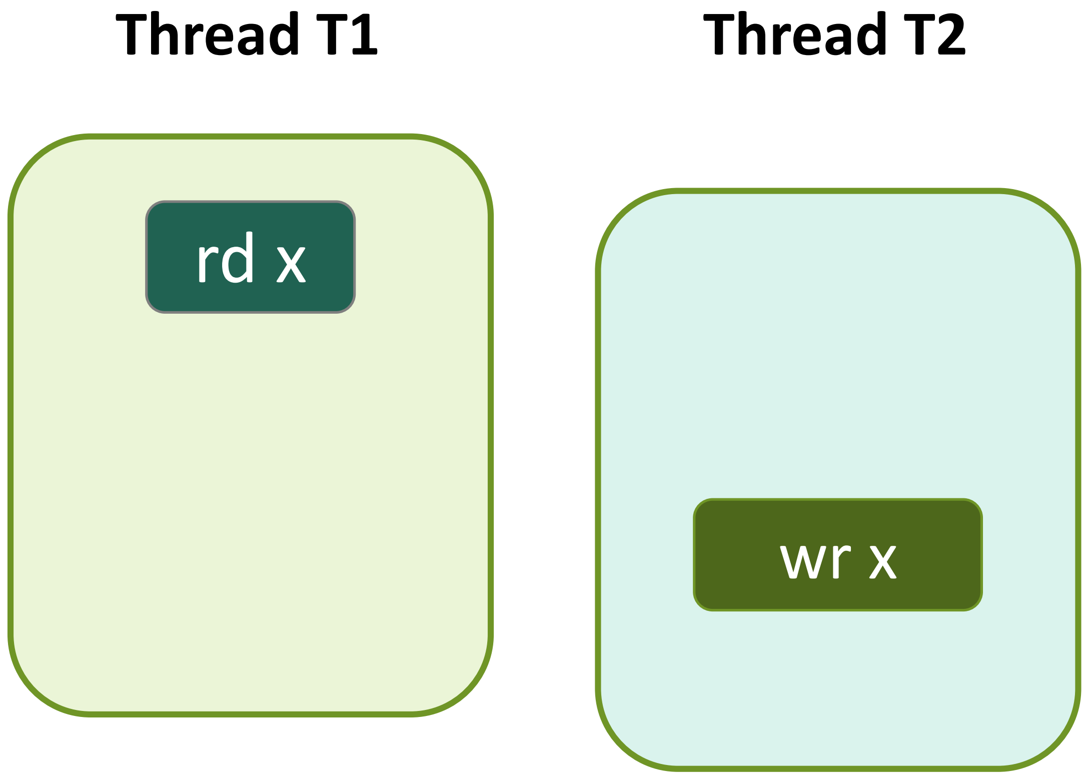
- ❖ Allows precisely detecting write-write and write-read conflicts



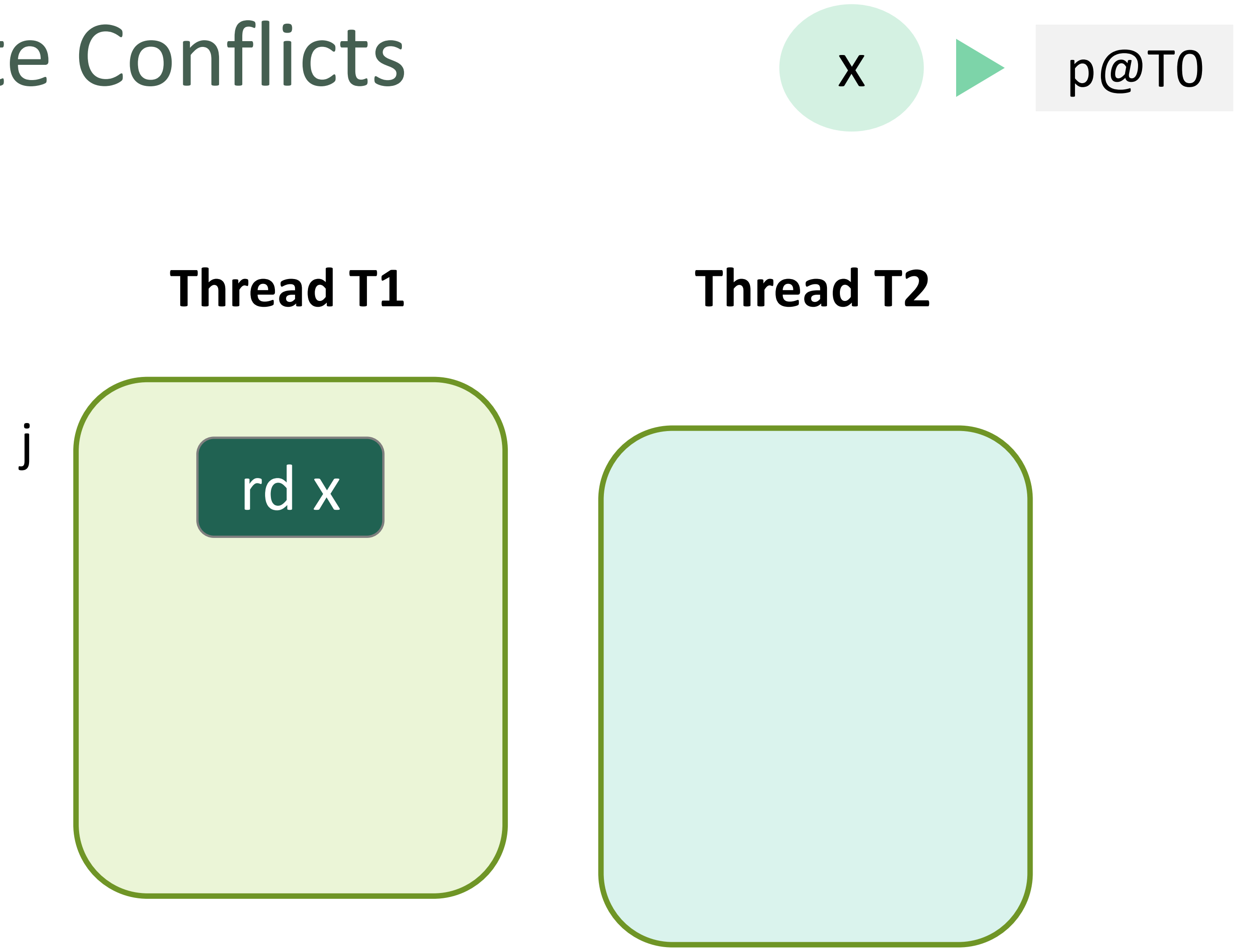
Valor: Efficient, Software-Only Region Conflict Detector

- Tracking last writers
- Detecting read-write conflicts lazily
- Impact of lazy conflict detection

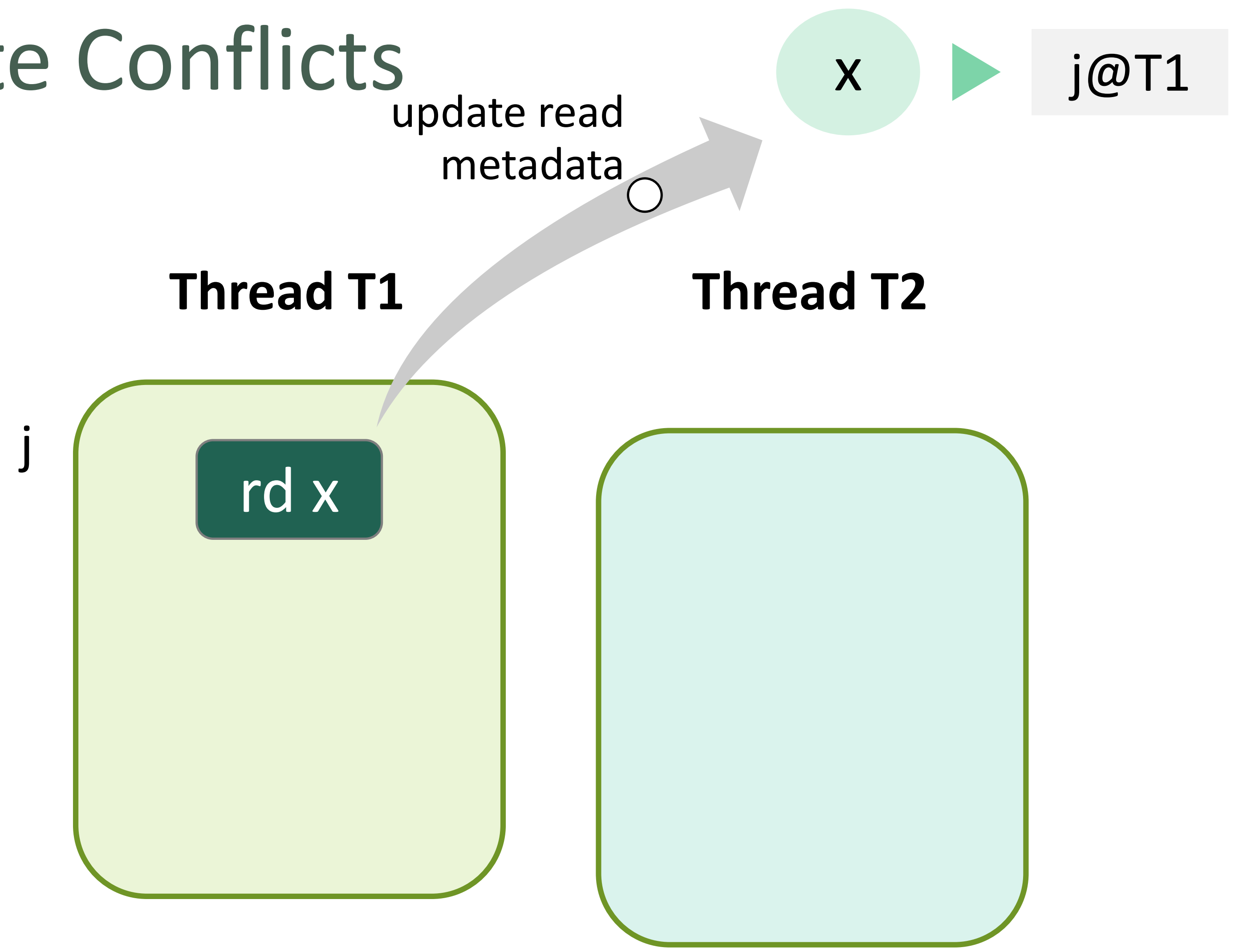
Detecting Read-Write Conflicts



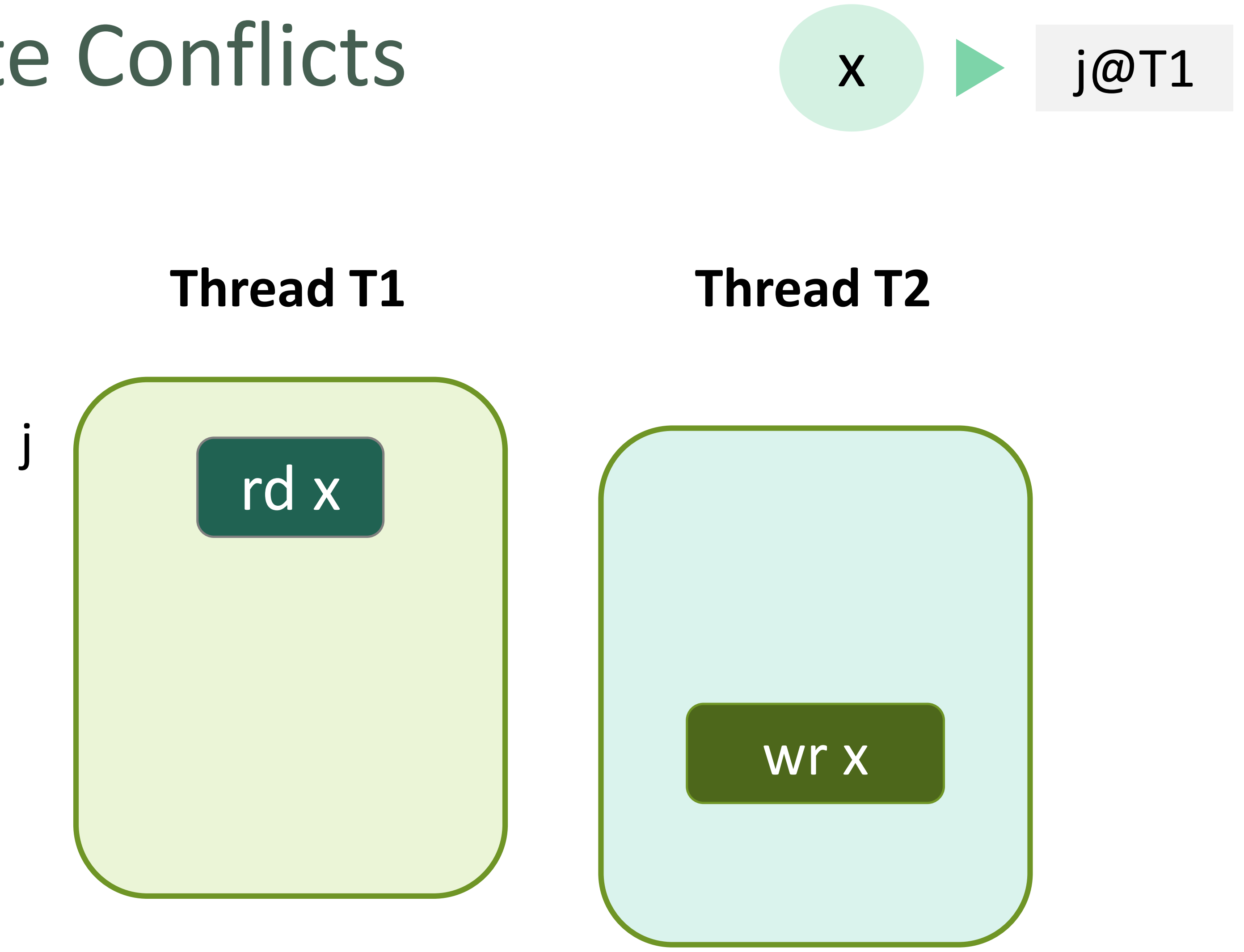
Detecting Read-Write Conflicts



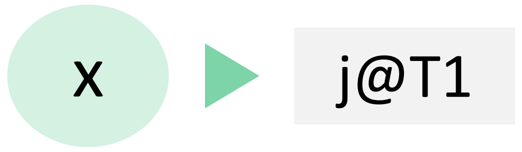
Detecting Read-Write Conflicts

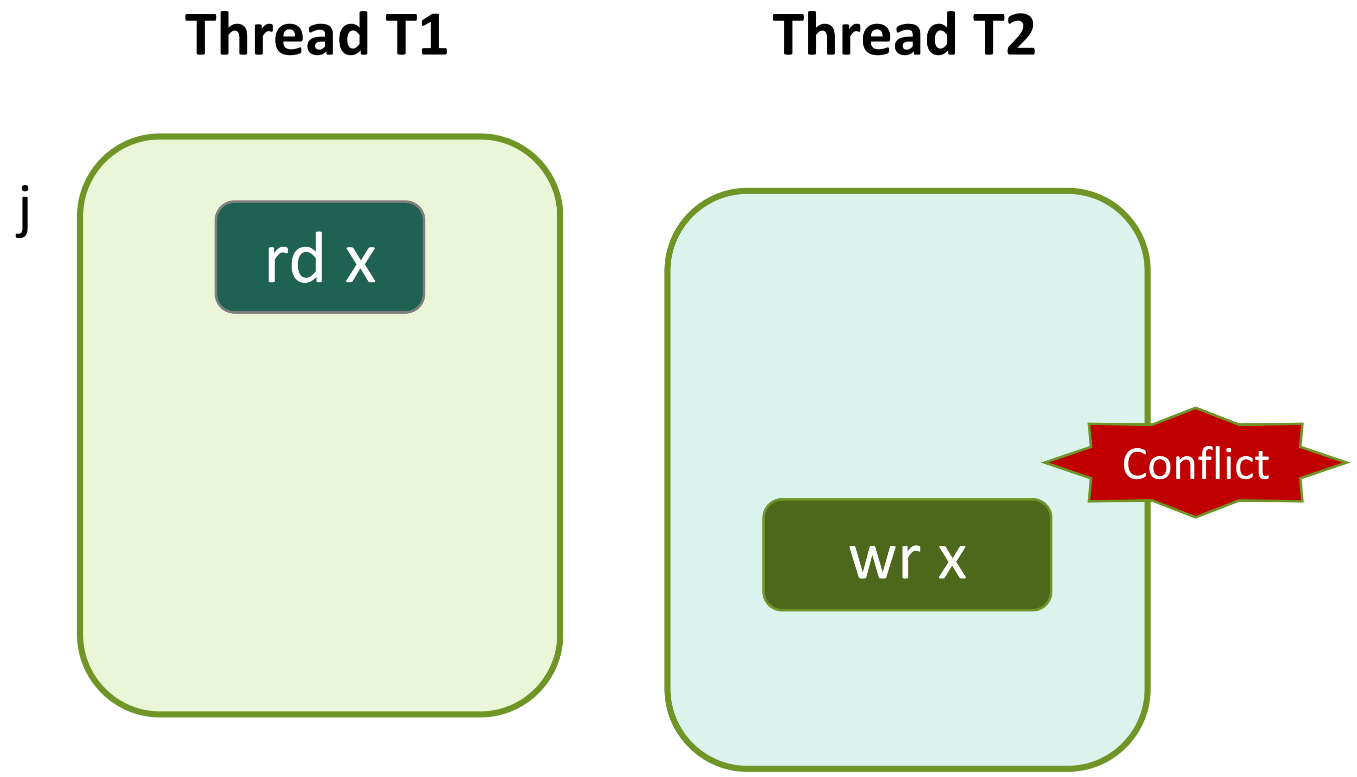


Detecting Read-Write Conflicts

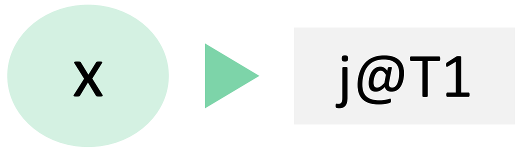


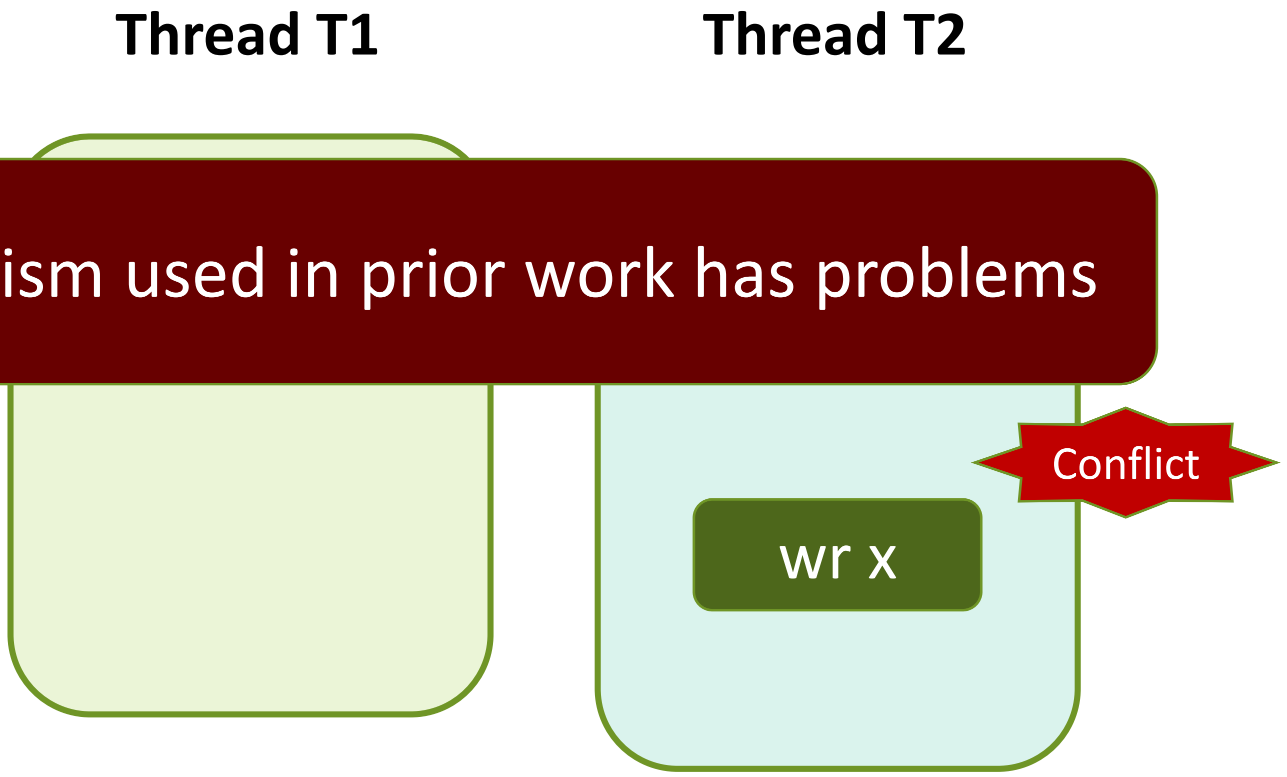
Detecting Read-Write Conflicts





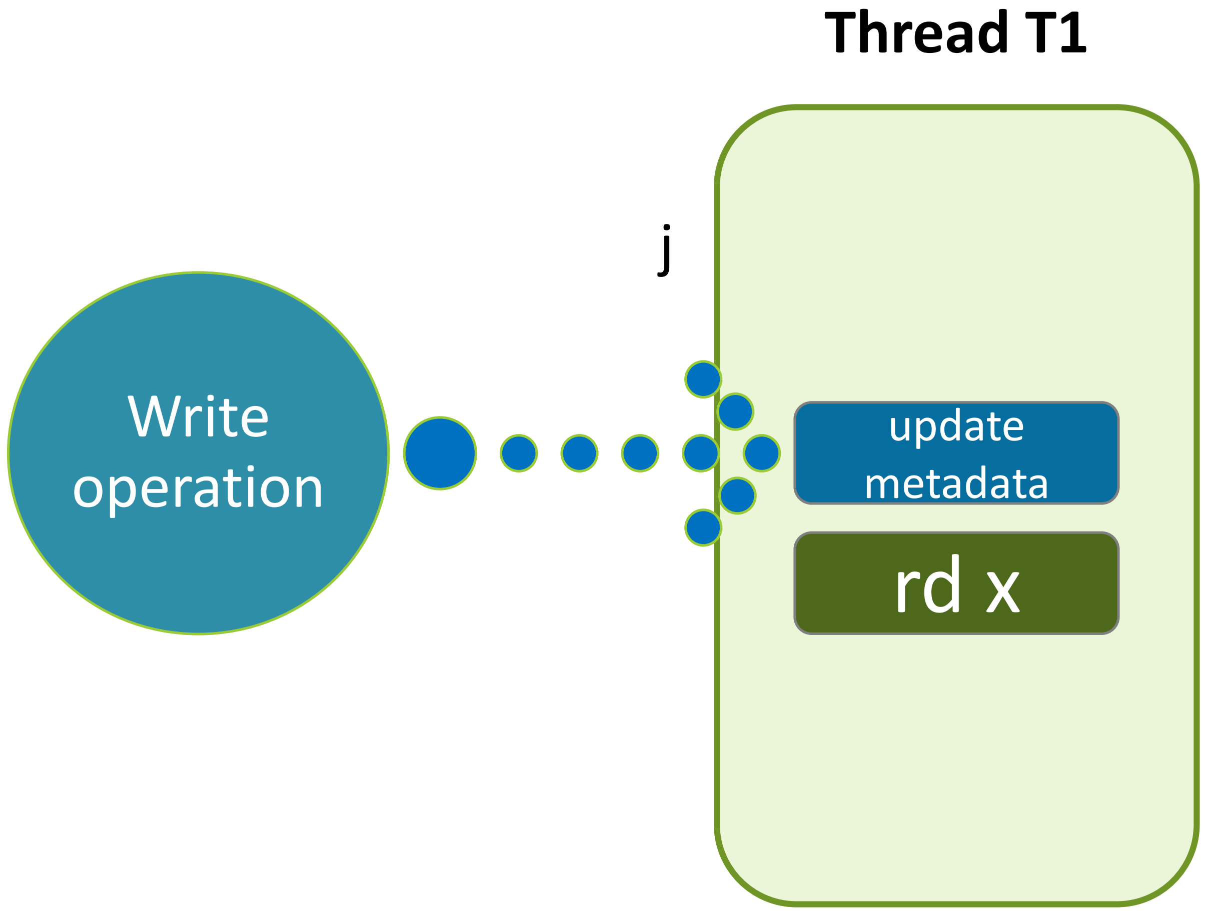
Detecting Read-Write Conflicts





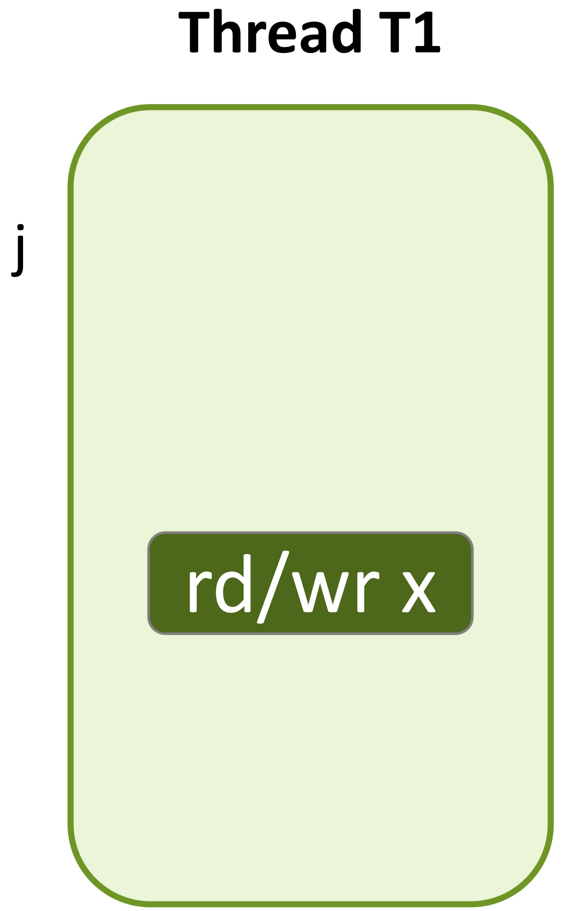
This simple mechanism used in prior work has problems

Remote Cache Misses Due to Tracking of Metadata

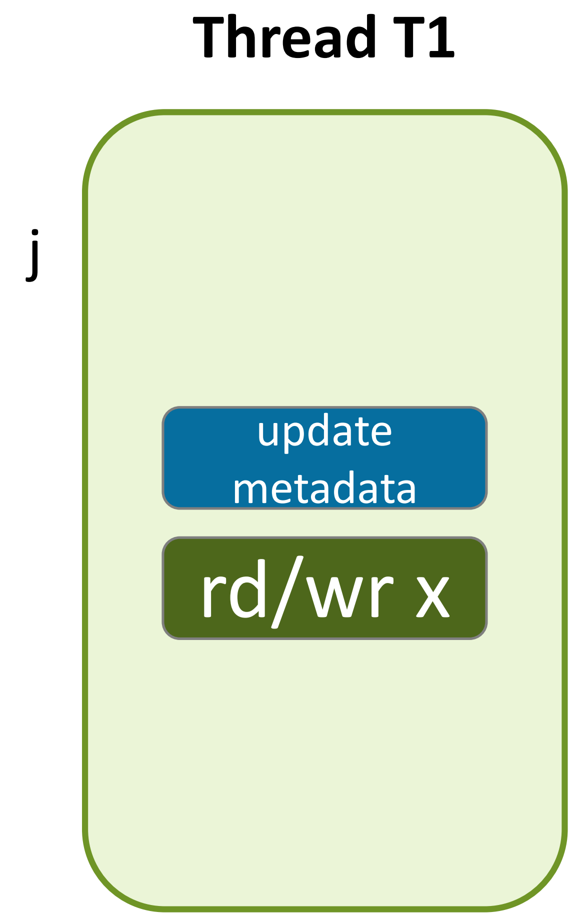


Leads to remote cache misses

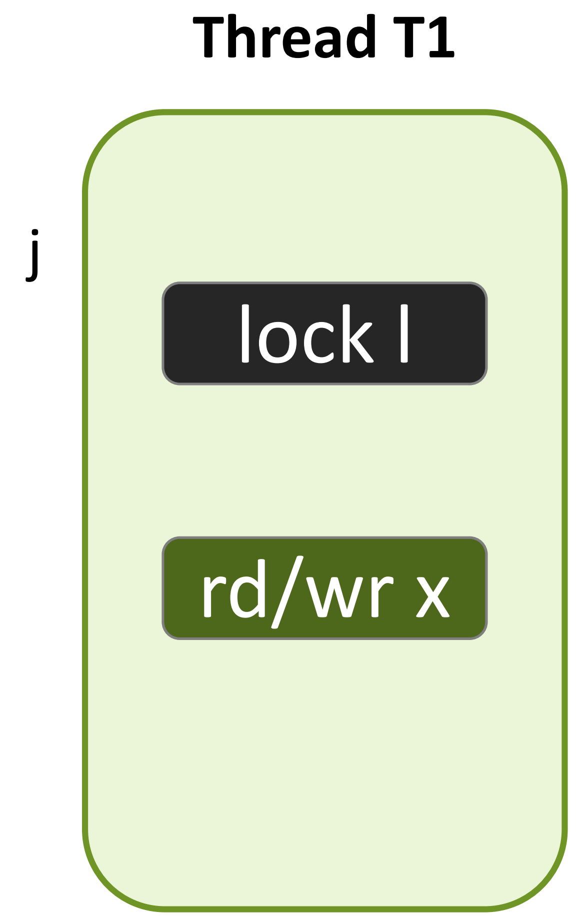
Metadata Updates



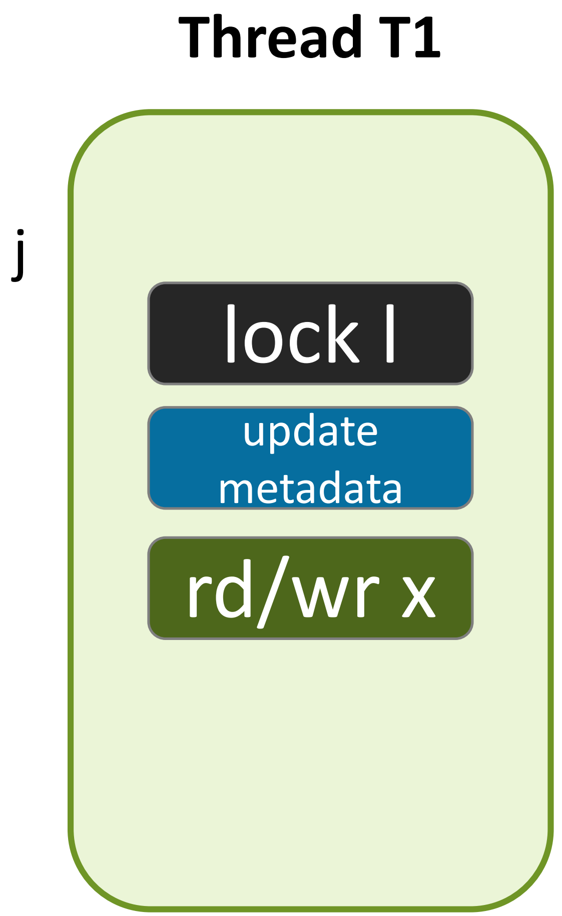
Metadata Updates



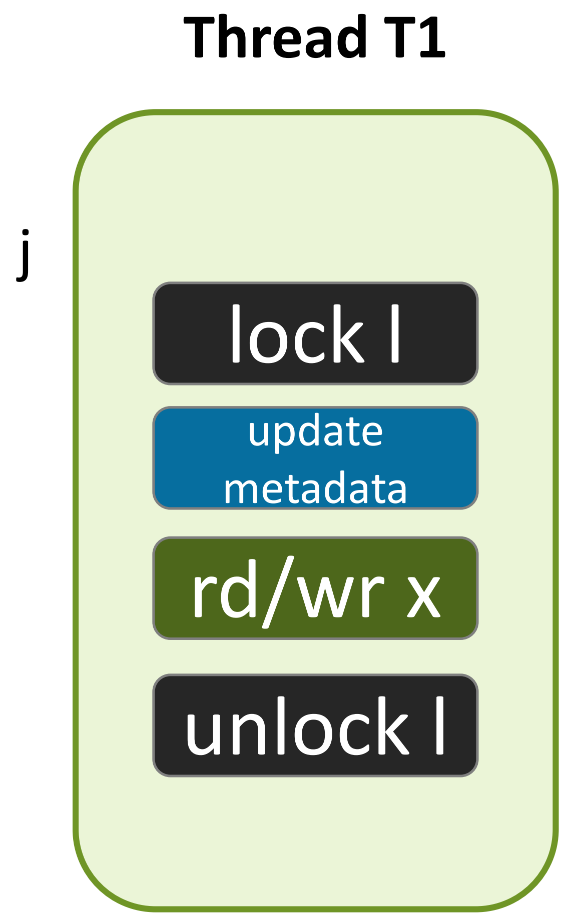
Synchronization on Metadata Updates



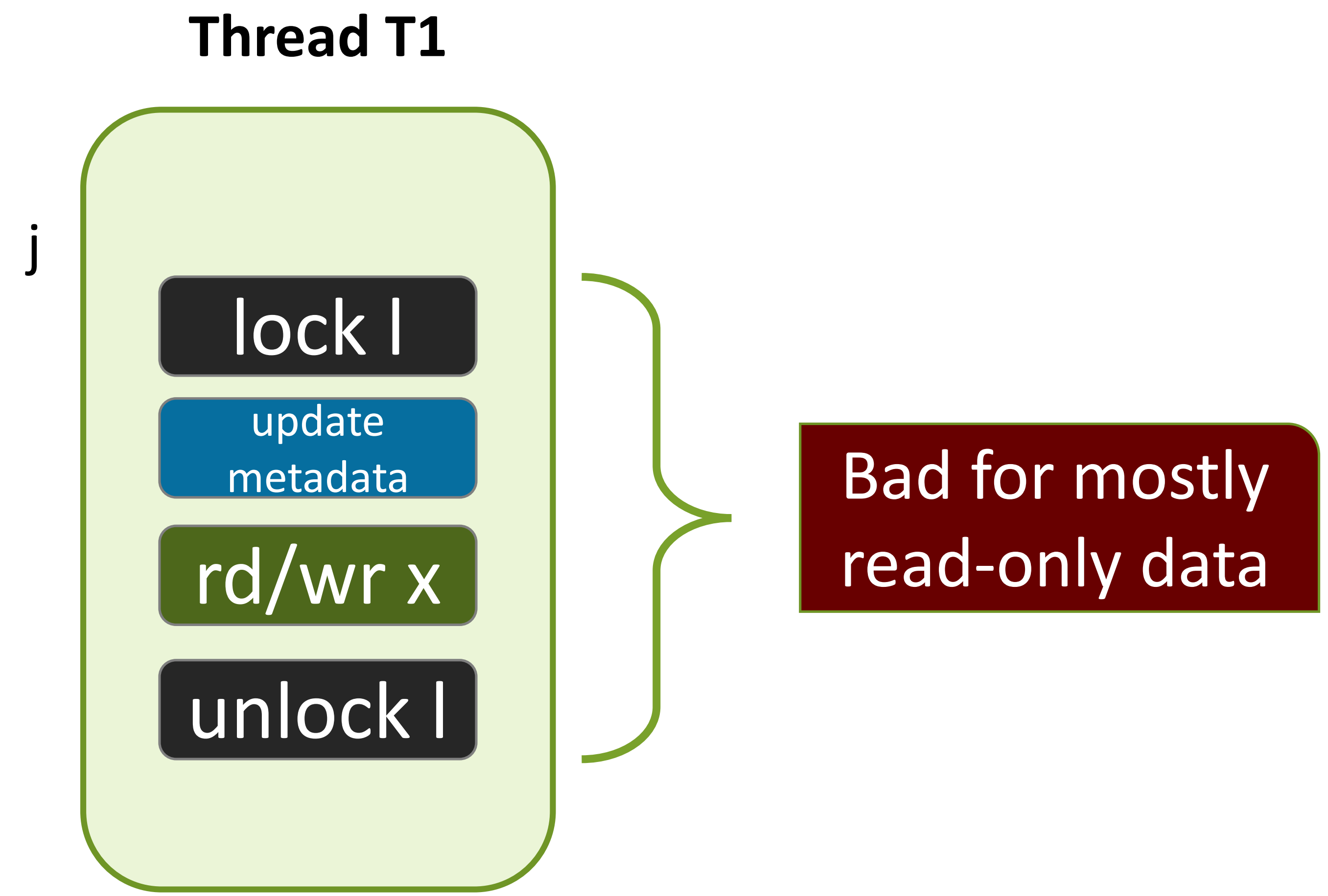
Synchronization on Metadata Updates



Synchronization on Metadata Updates

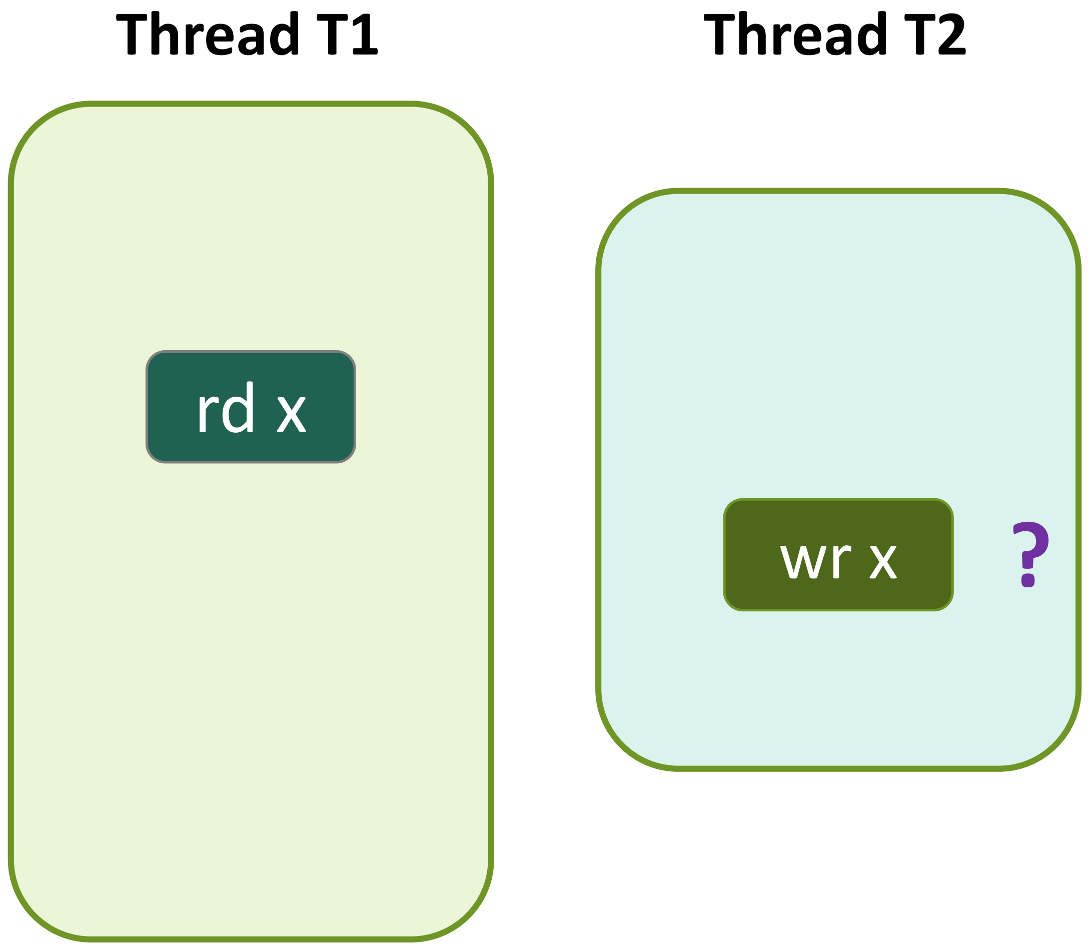


Synchronization on Metadata Updates



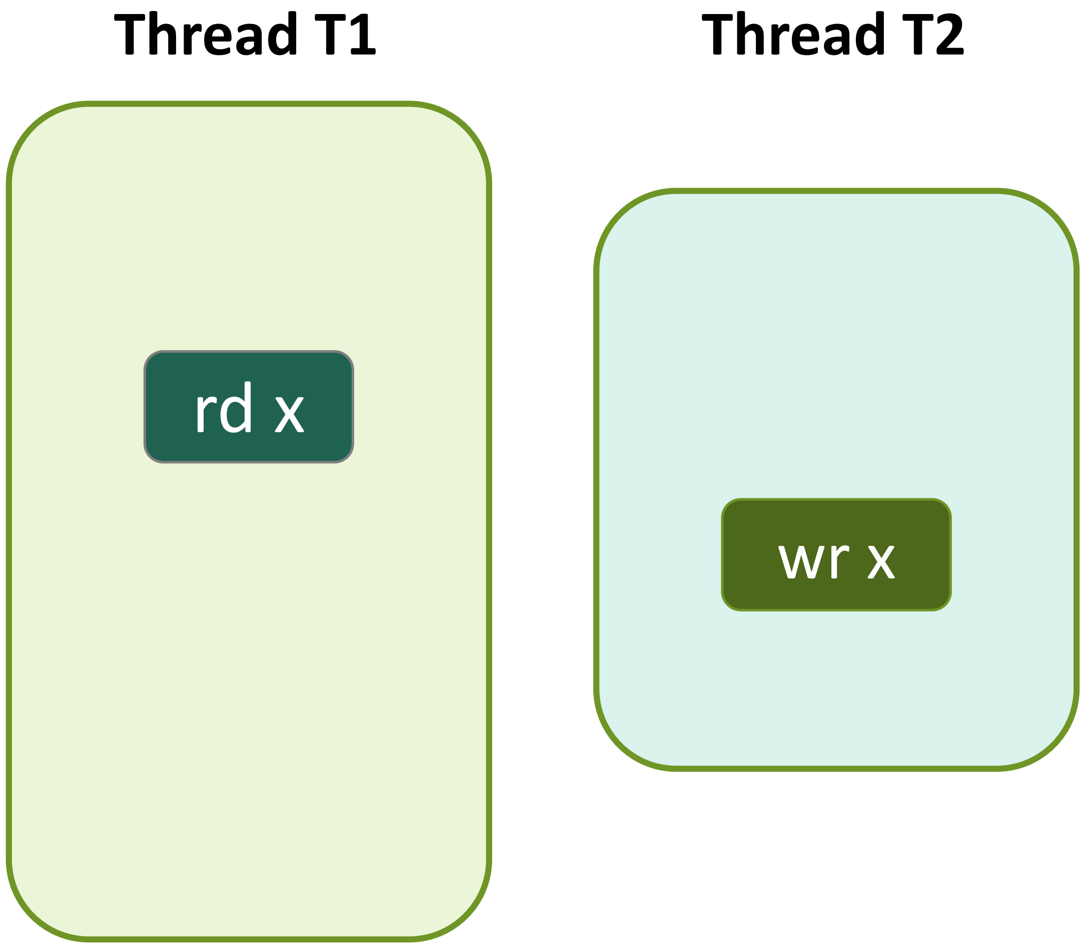
Elide Tracking Last Readers

Elide Tracking Last Readers



Elide Tracking Last Readers

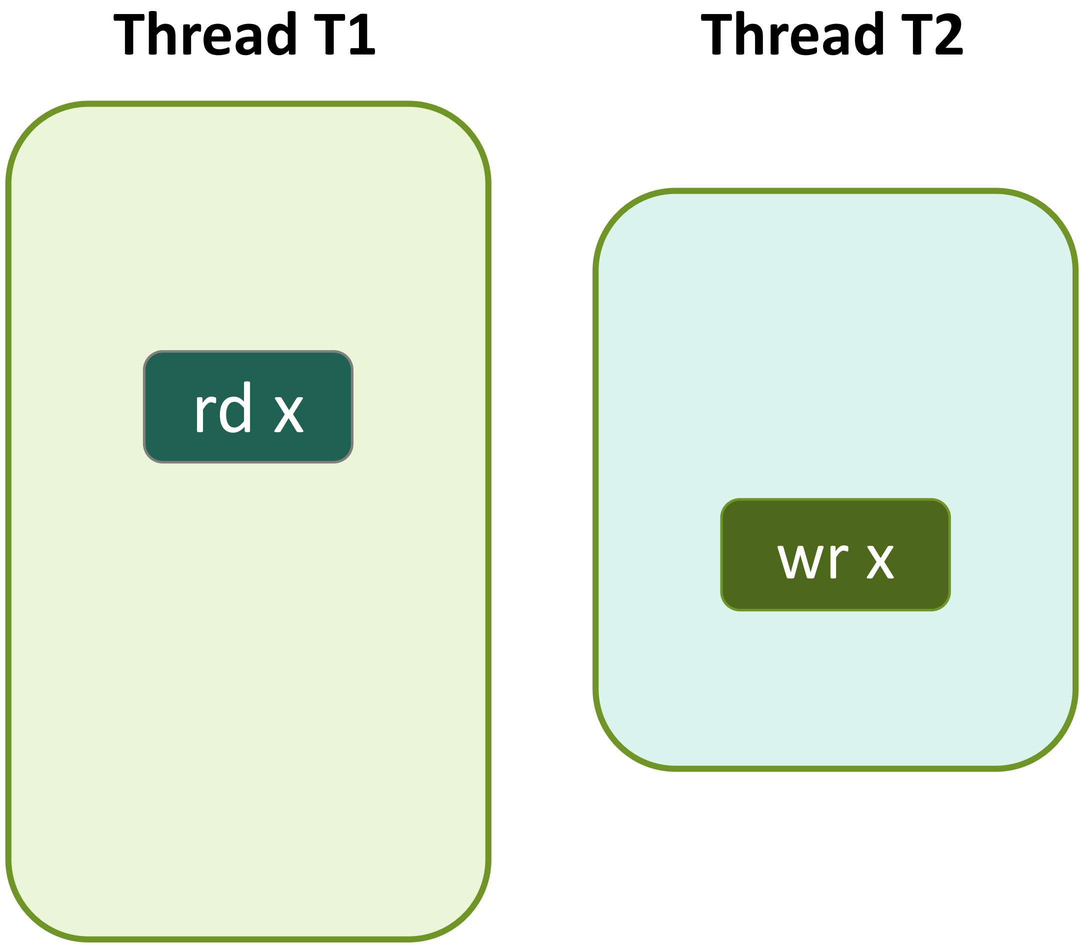
Detect read-write conflicts lazily



Elide Tracking Last Readers

Detect read-write conflicts lazily

- Log read accesses in thread-local buffers
- Validate reads at region boundaries



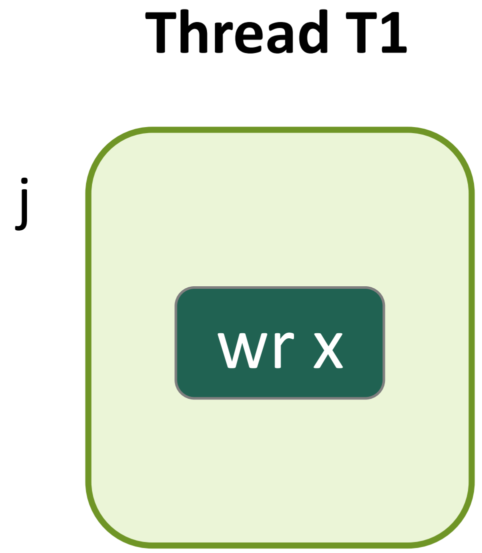
1. Saha et al. McRT-STM: A High Performance Software Transactional Memory System for a Multi-Core Runtime. PPoPP 2006.

Per-Variable Metadata

Has an ongoing region updated x?

Version, Epoch

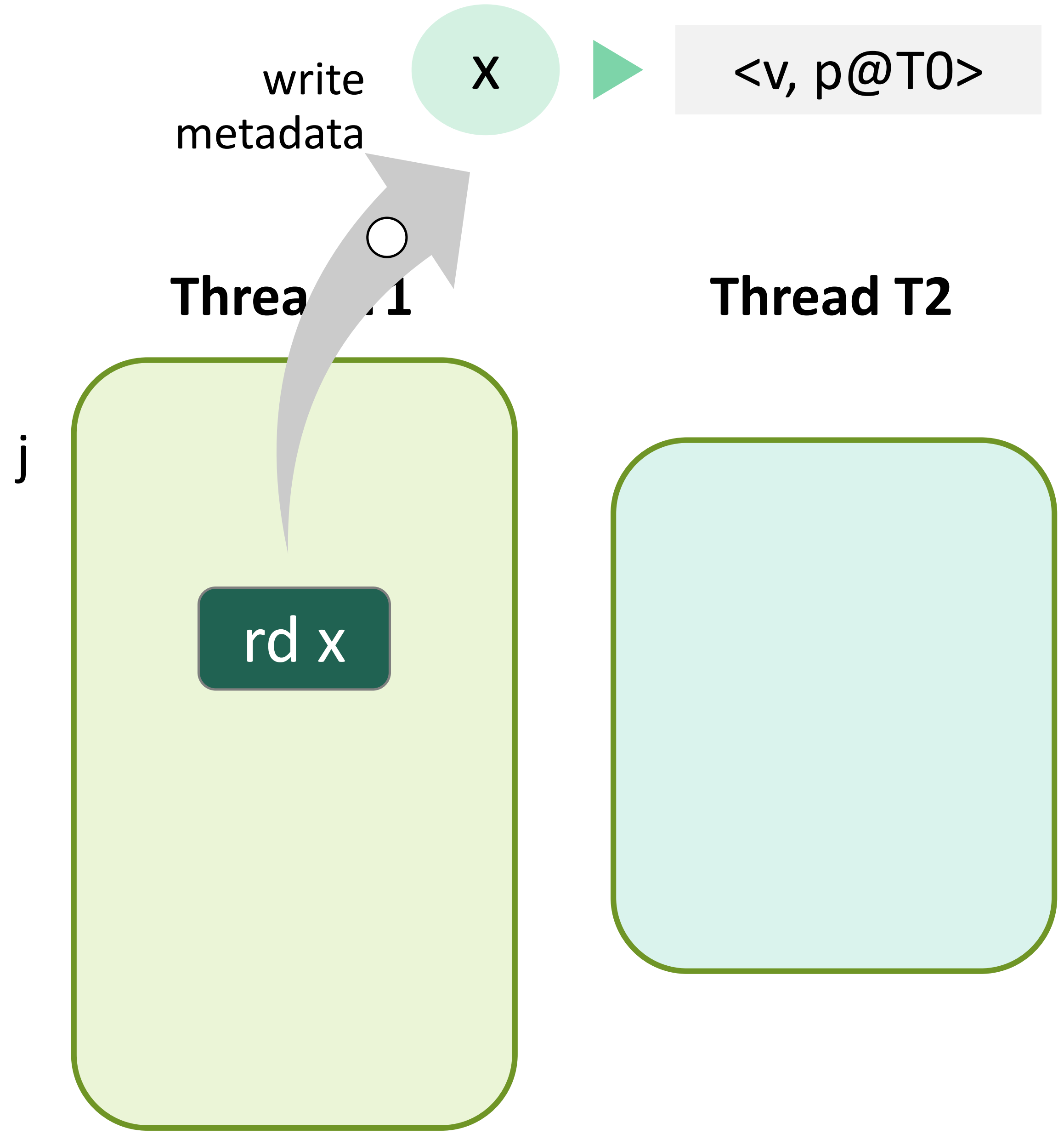
<v, j@T1>



Elide Tracking Last Readers

Detect read-write conflicts lazily

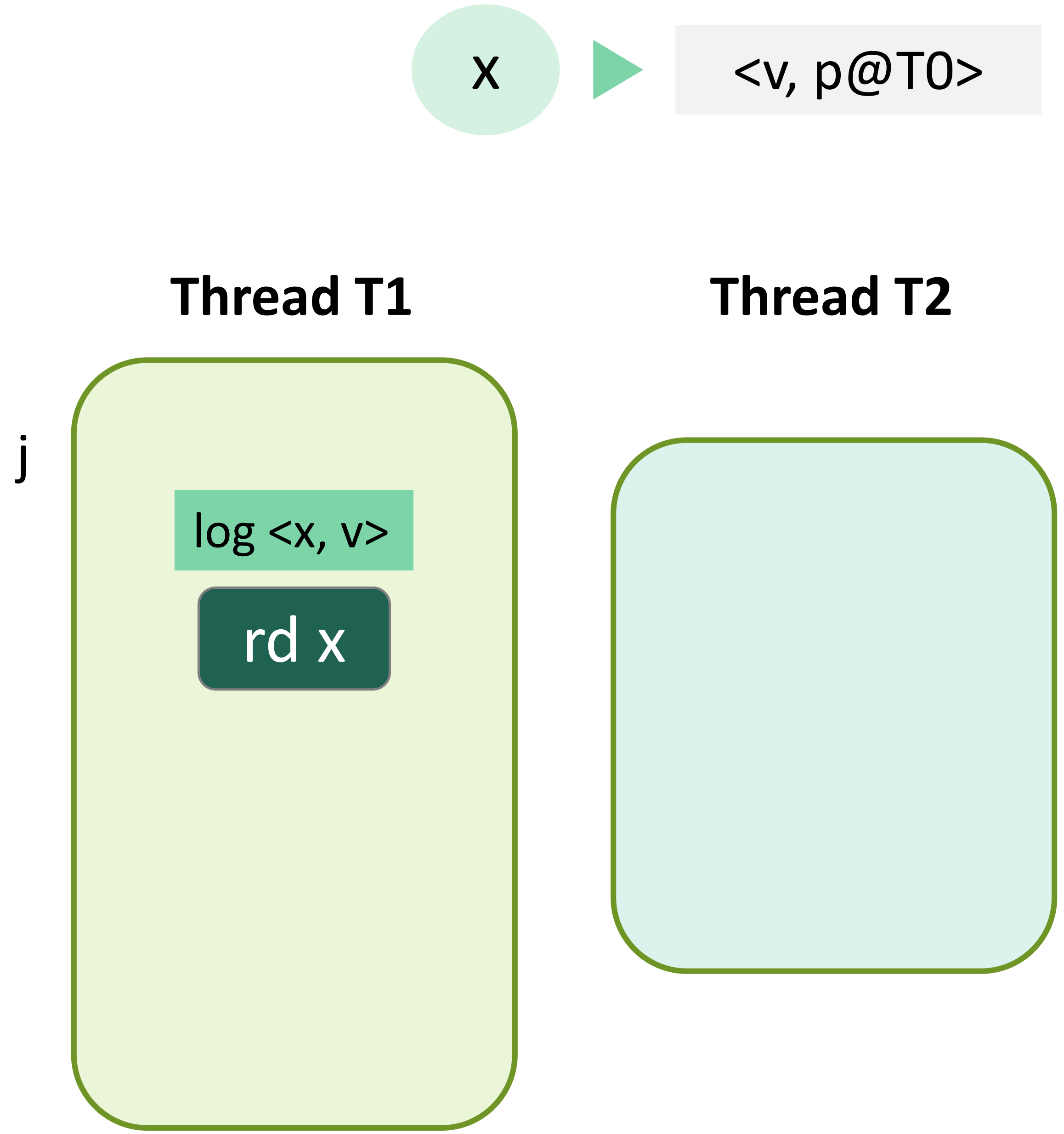
- Log read accesses in thread-local buffers
- Validate reads at region boundaries



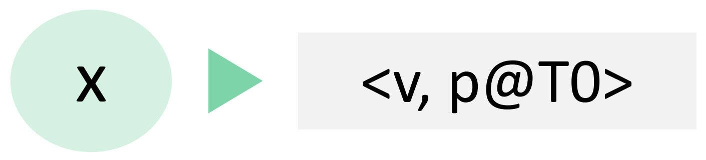
Elide Tracking Last Readers

Detect read-write conflicts lazily

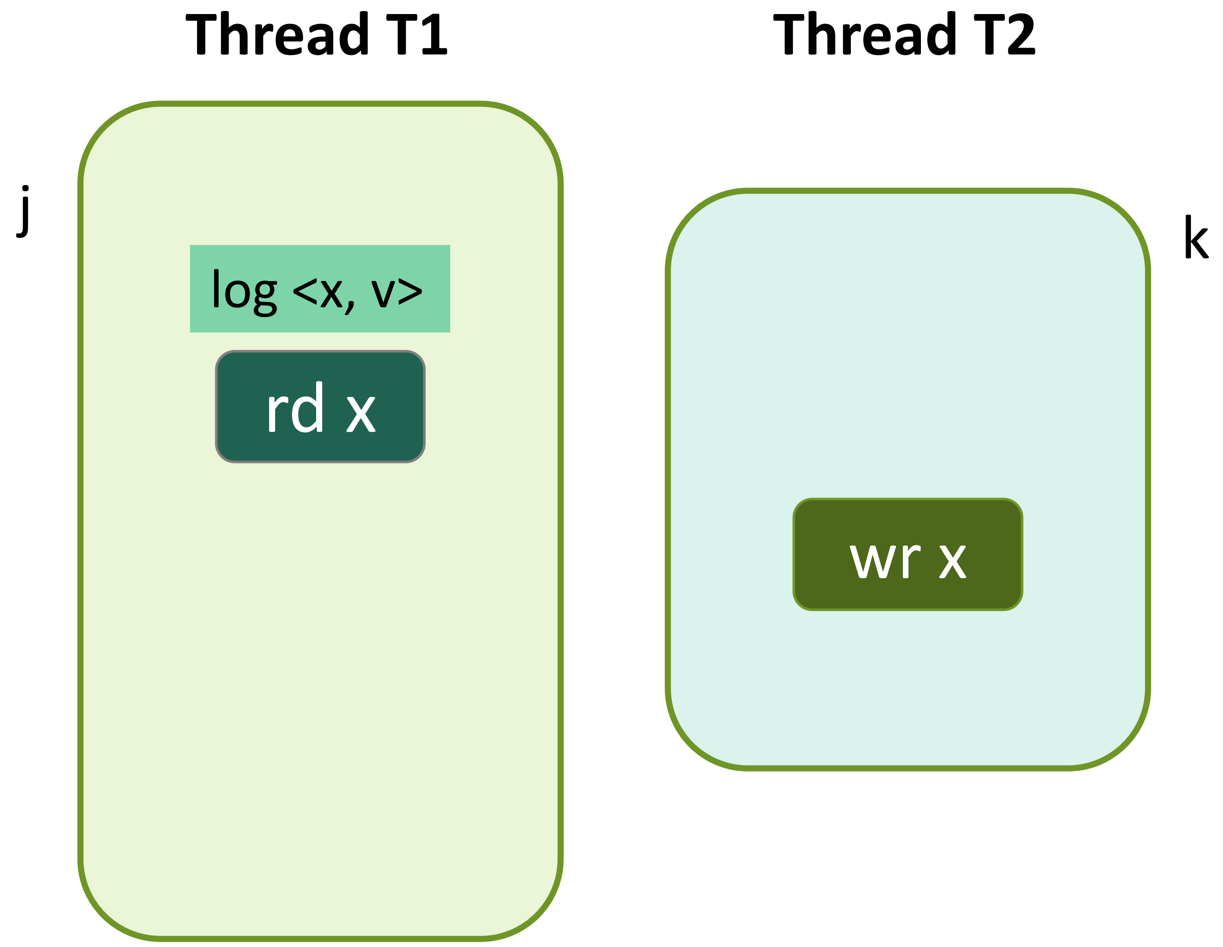
- Log read accesses in thread-local buffers
- Validate reads at region boundaries



Elide Tracking Last Readers

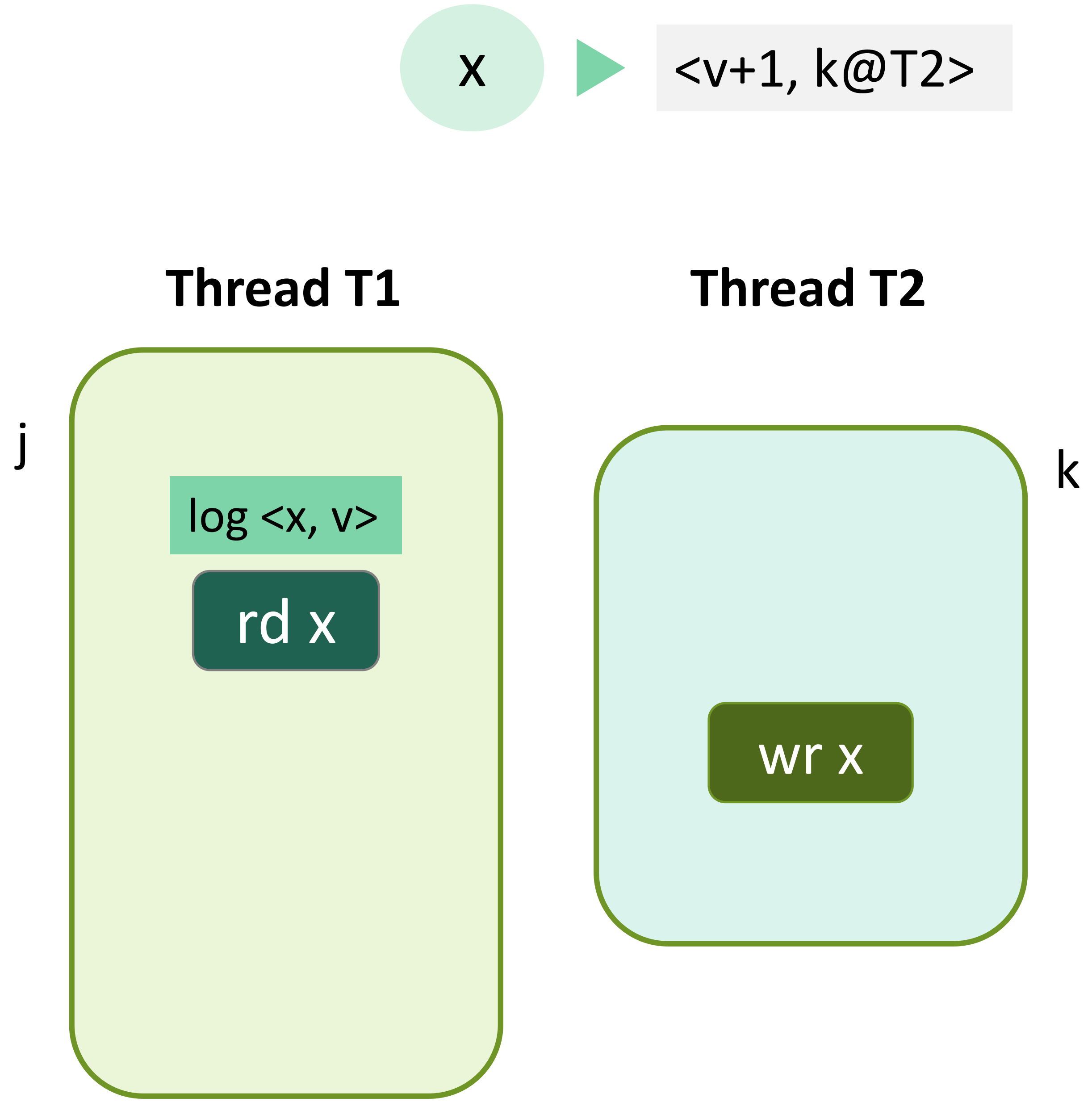


Detect read-write conflicts lazily

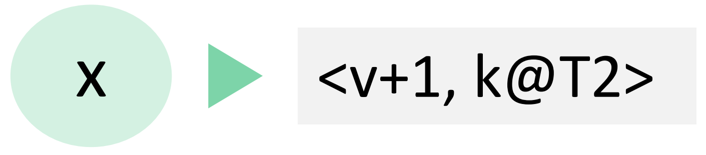


Elide Tracking Last Readers

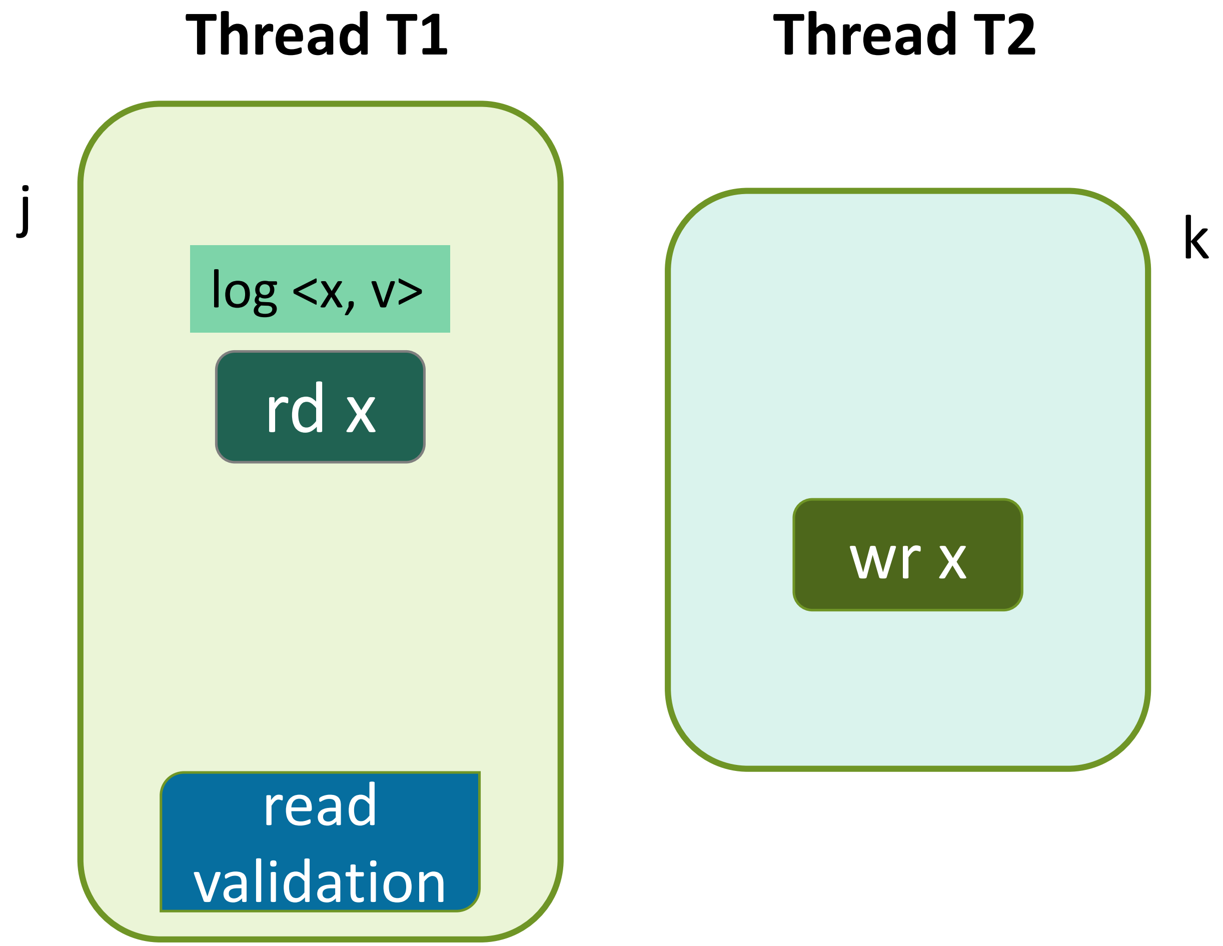
Detect read-write conflicts lazily



Elide Tracking Last Readers



Detect read-write conflicts lazily

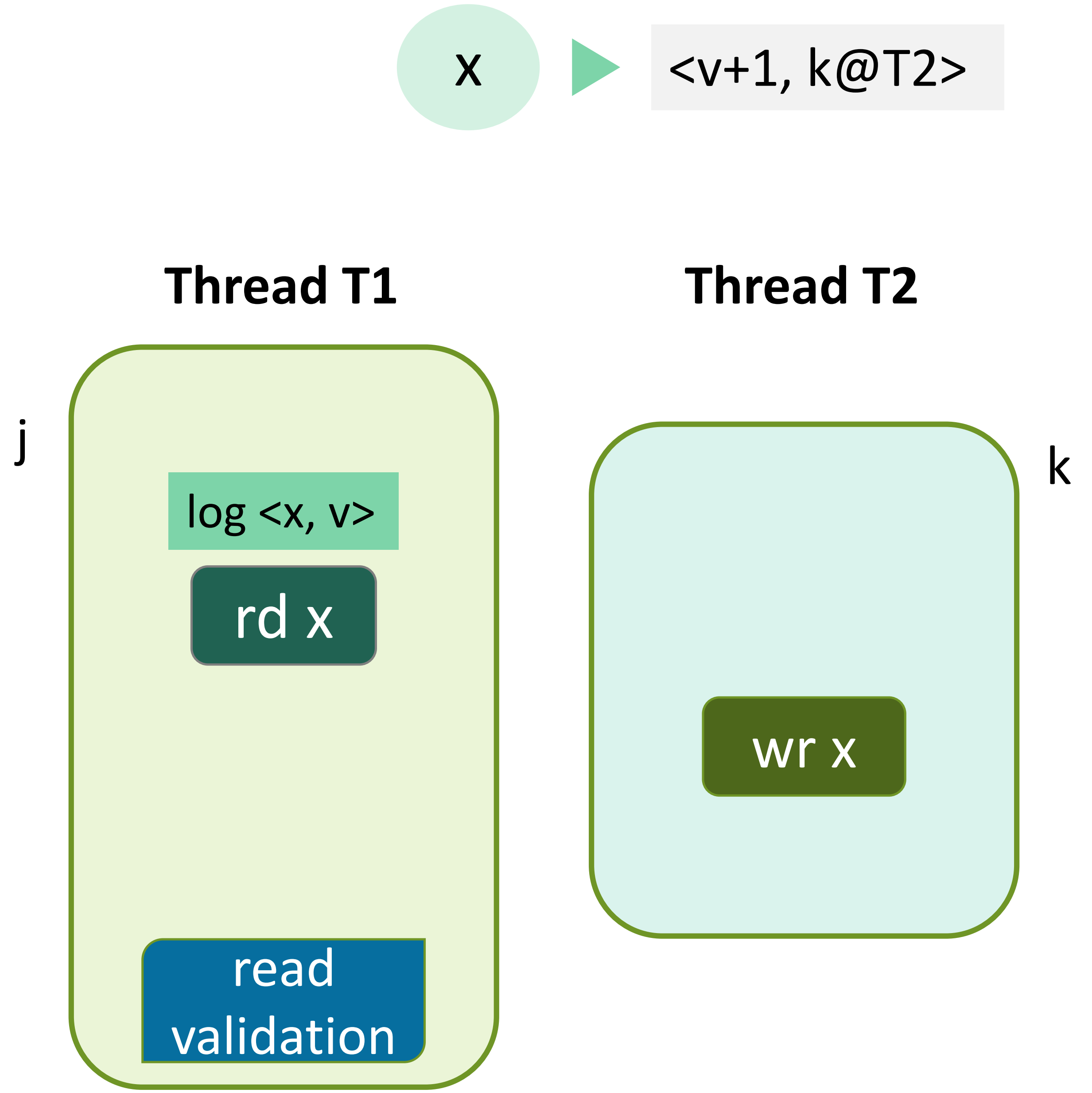


Elide Tracking Last Readers

Detect read-write conflicts lazily

T1 is not the last writer

Version read is outdated

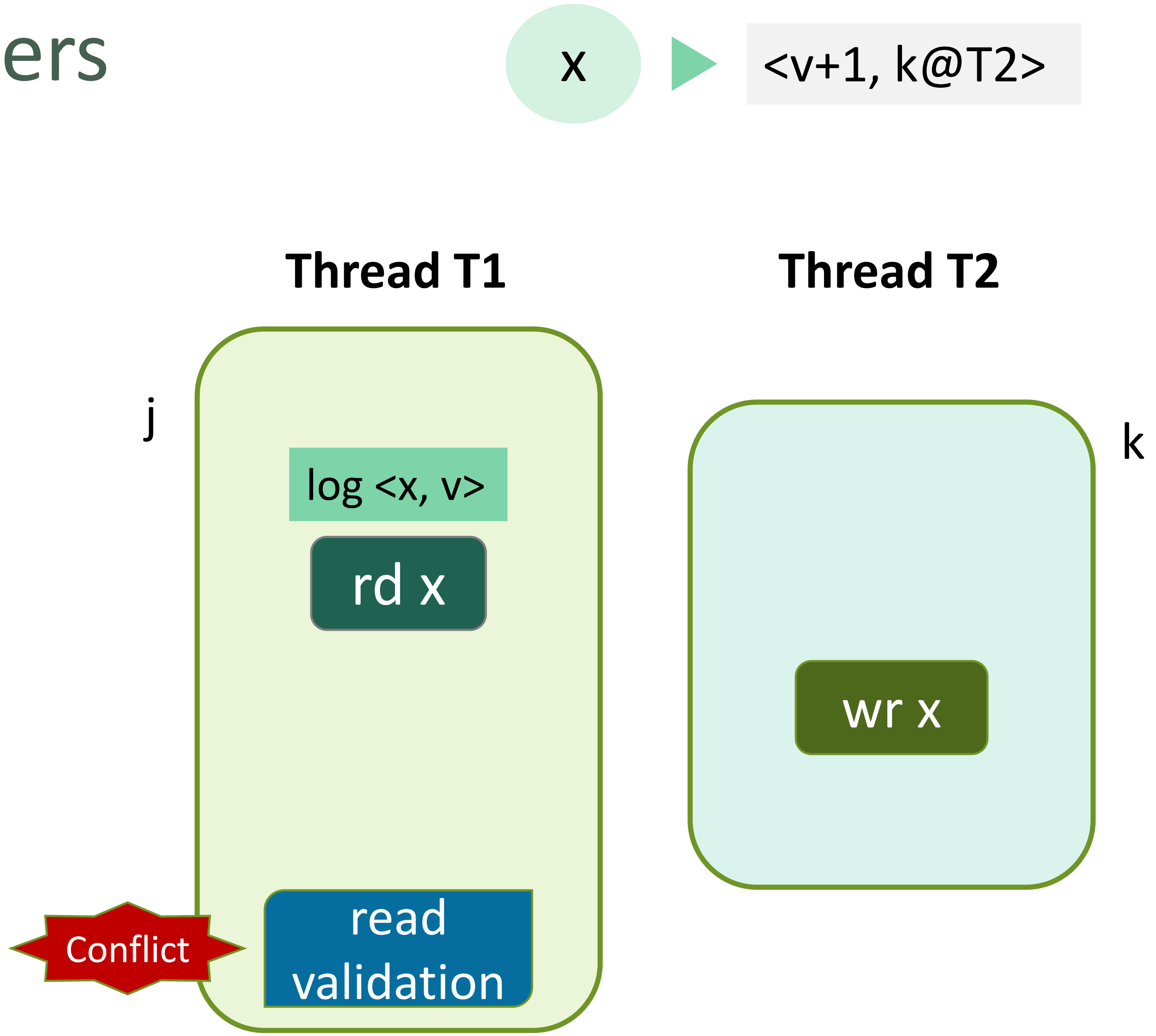


Elide Tracking Last Readers

Detect read-write conflicts lazily

T1 is not the last writer

Version read is outdated



Elide Tracking Last Readers

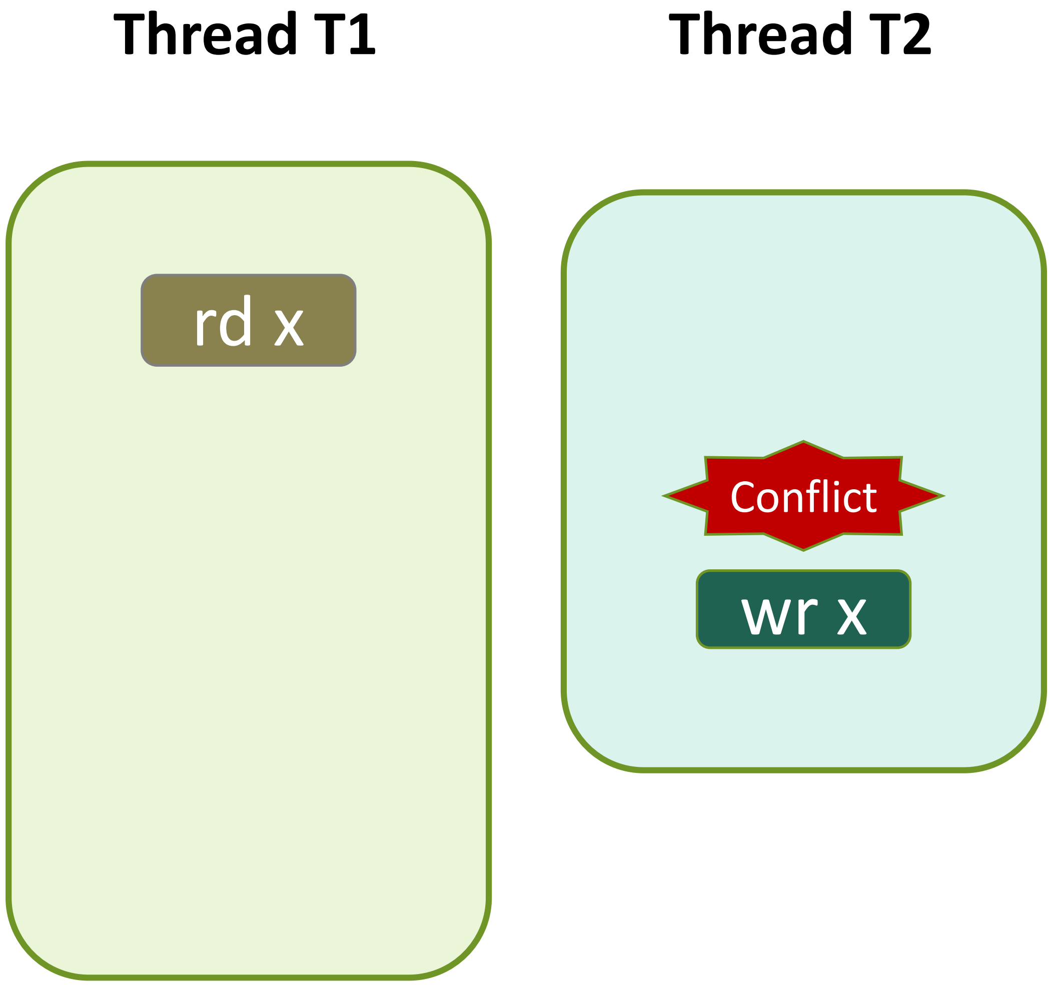
Avoids

- Remote cache misses
- Synchronization overhead

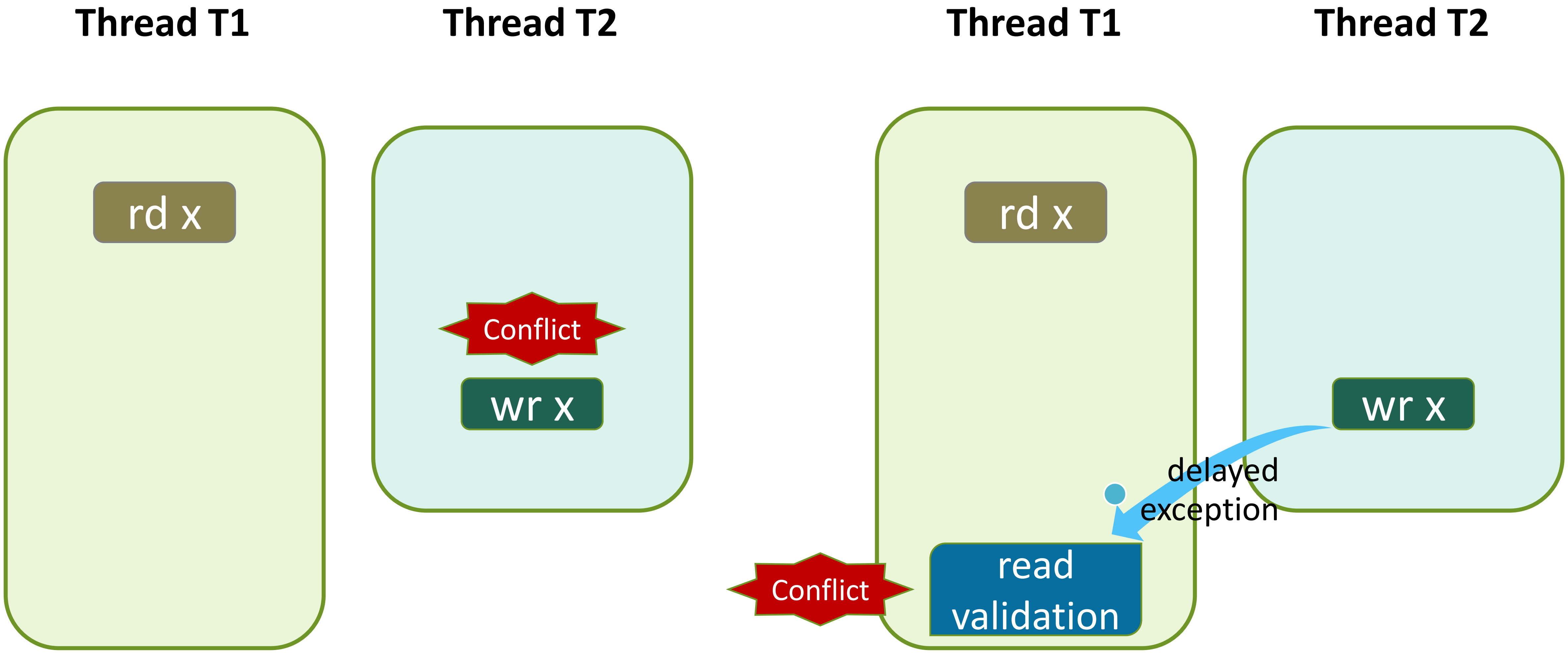
Valor: Efficient, Software-Only Region Conflict Detector

- Tracking last writers
- Detecting read-write conflicts lazily
- Impact of lazy conflict detection

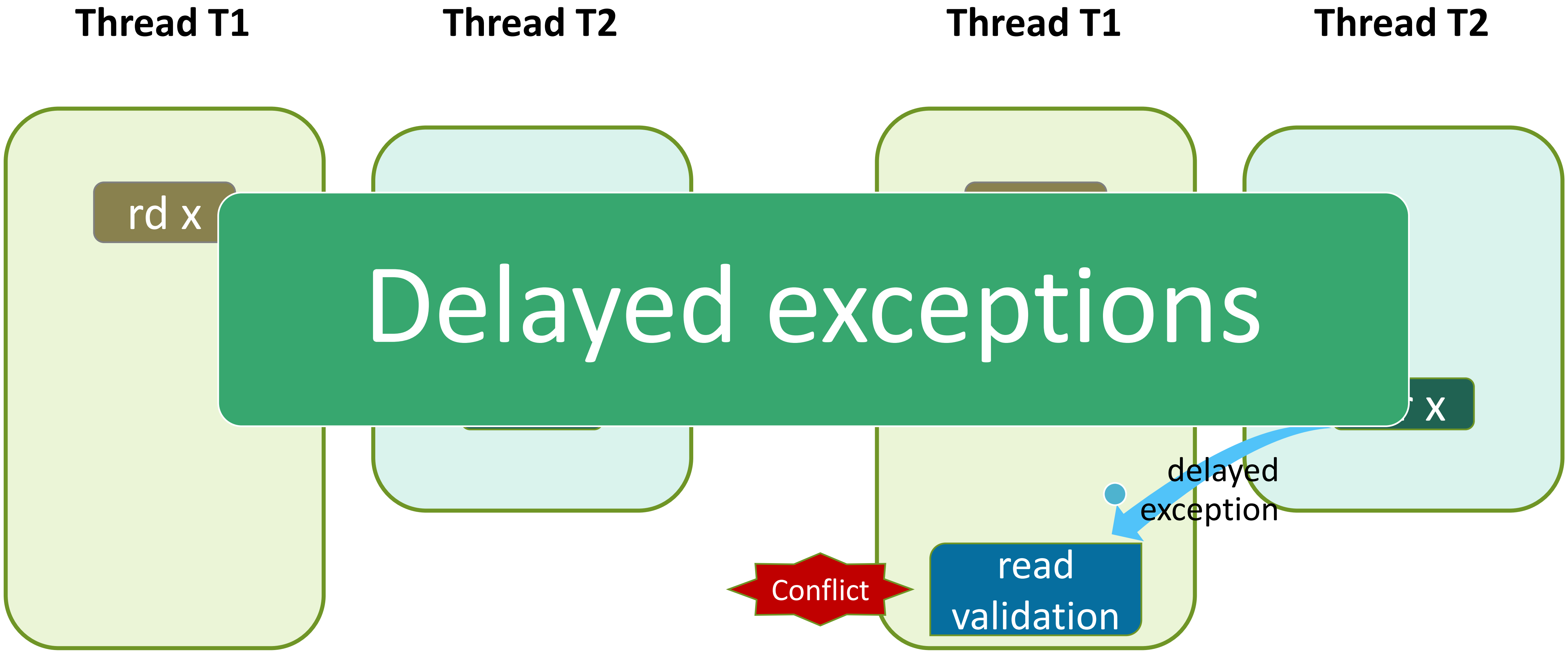
Precise Conflict Detection



Precise vs Lazy Conflict Detection

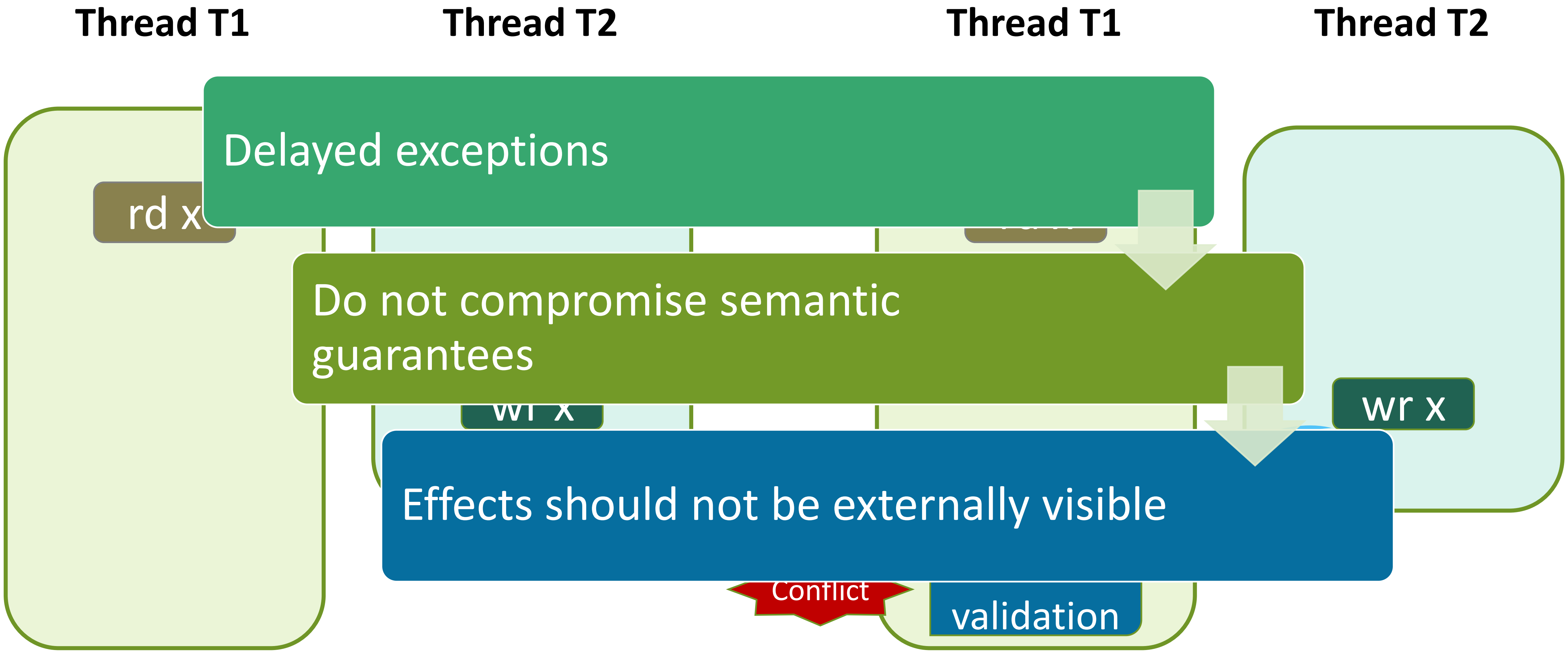


Precise vs Lazy Conflict Detection



Delayed exceptions

Delayed Exceptions

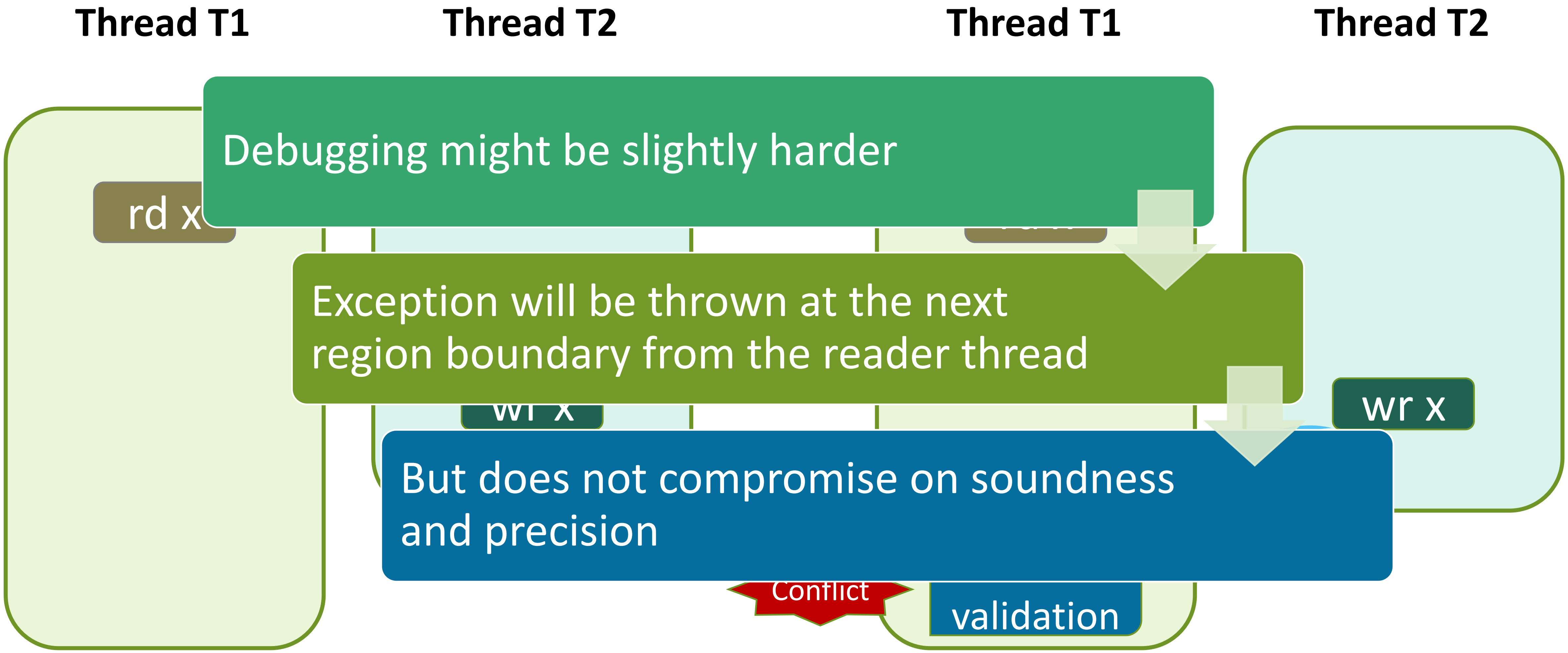


Delayed exceptions

Do not compromise semantic guarantees

Effects should not be externally visible

Delayed Exceptions



Debugging might be slightly harder

Exception will be thrown at the next region boundary from the reader thread

But does not compromise on soundness and precision

Outline

- Programming language memory models and data races
- Data race and region conflict exceptions model
- Valor: Our contribution
- **Evaluation**

IMPLEMENTATION

Implementation

Developed on top of Jikes RVM 3.1.3

Artifact * AEC * Evaluated * OOPSLA *

Consistent * Complete * Well Documented * Easy to Reuse *

Implementation

Developed on top of Jikes RVM 3.1.3

Implemented FastTrack, state-of-art happens-before analysis based data race detector

1. Flanagan and Freund. FastTrack: Efficient and Precise Dynamic Data Race Detection. PLDI 2009.

Implementation

Developed on top of Jikes RVM 3.1.3

Implemented FastTrack, state-of-art happens-before analysis based data race detector

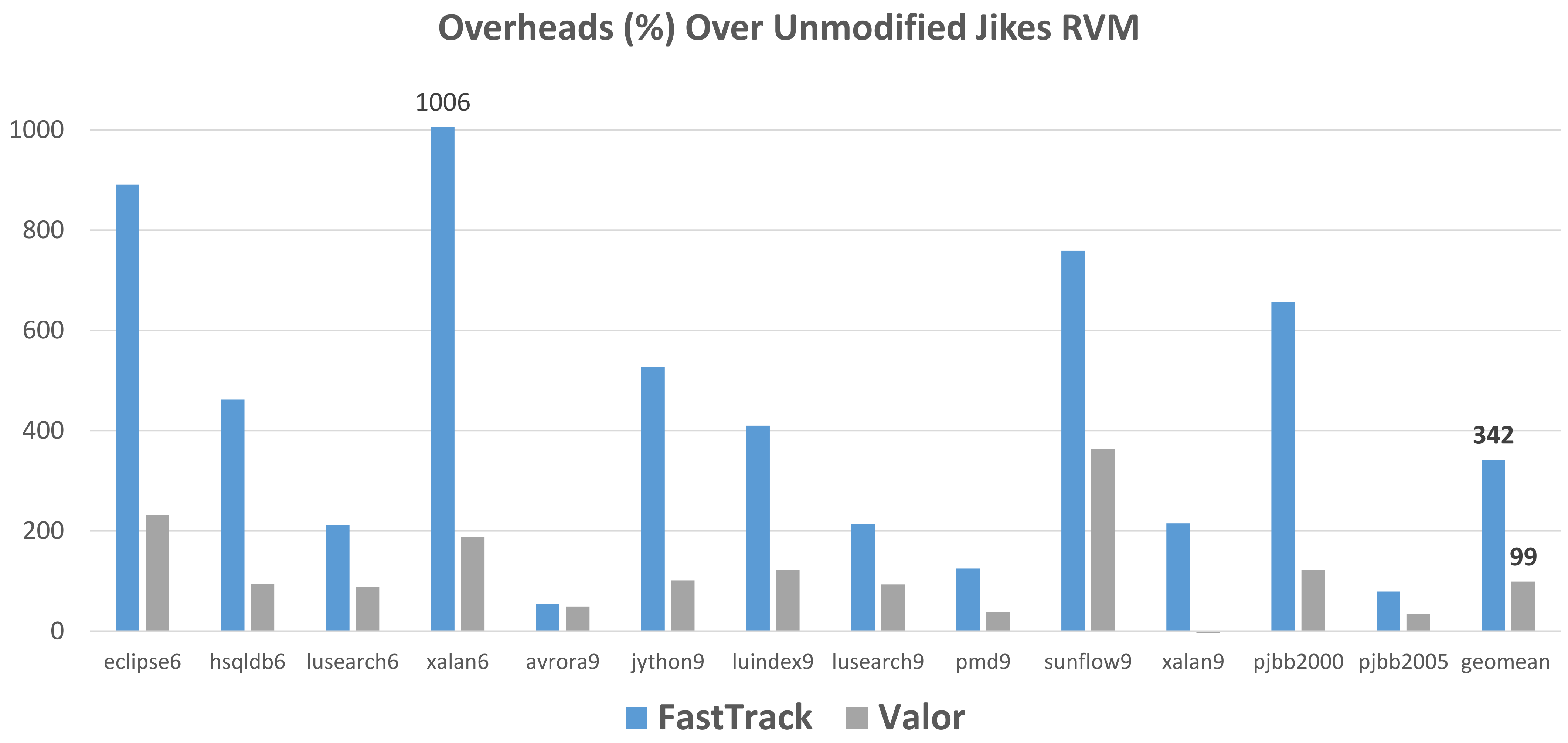
Shared on Jikes RVM Research Archive and ACM DL

EVALUATION

Experimental Methodology

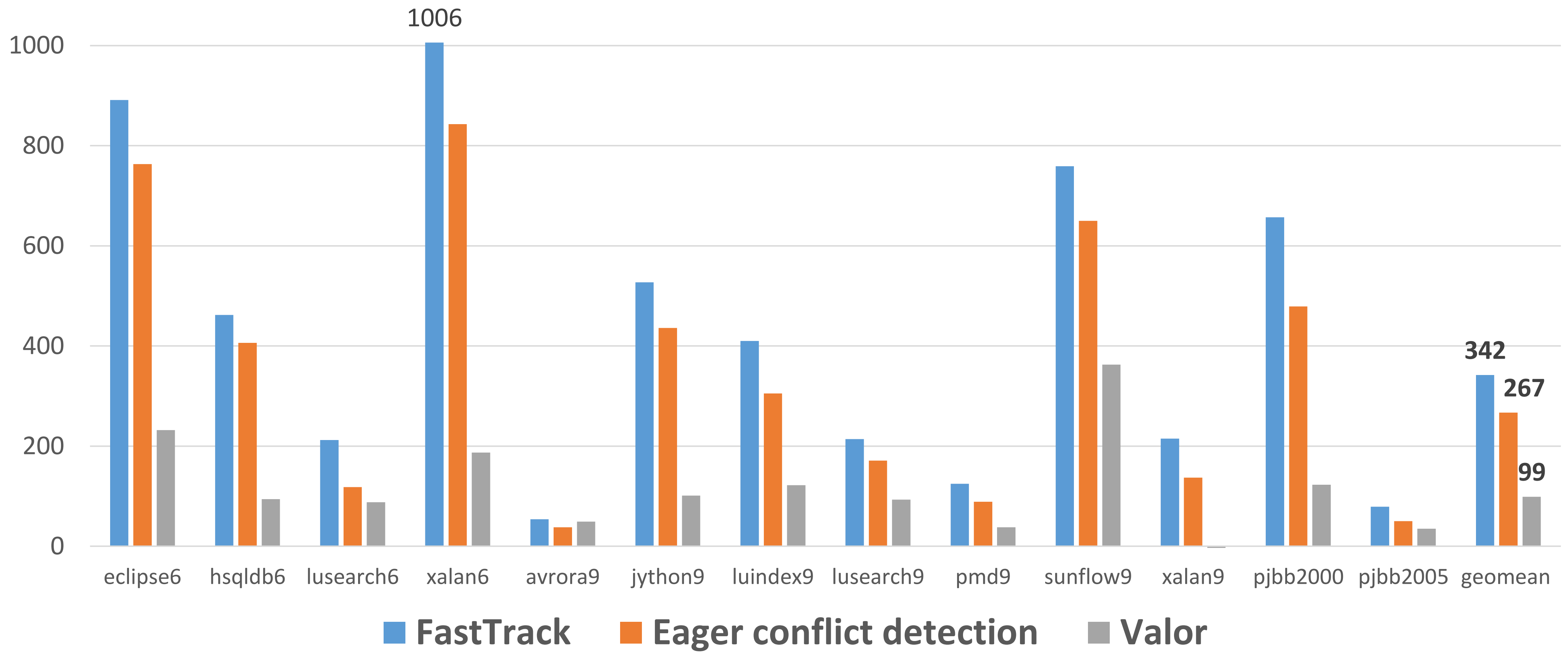
- **Benchmarks**
  - Large workload sizes of DaCapo 2006 and 9.12-bach suite
  - Fixed-workload versions of SPECjbb2000 and SPECjbb2005
- **Platform**
  - 64-core AMD Opteron 6272

Performance Comparison

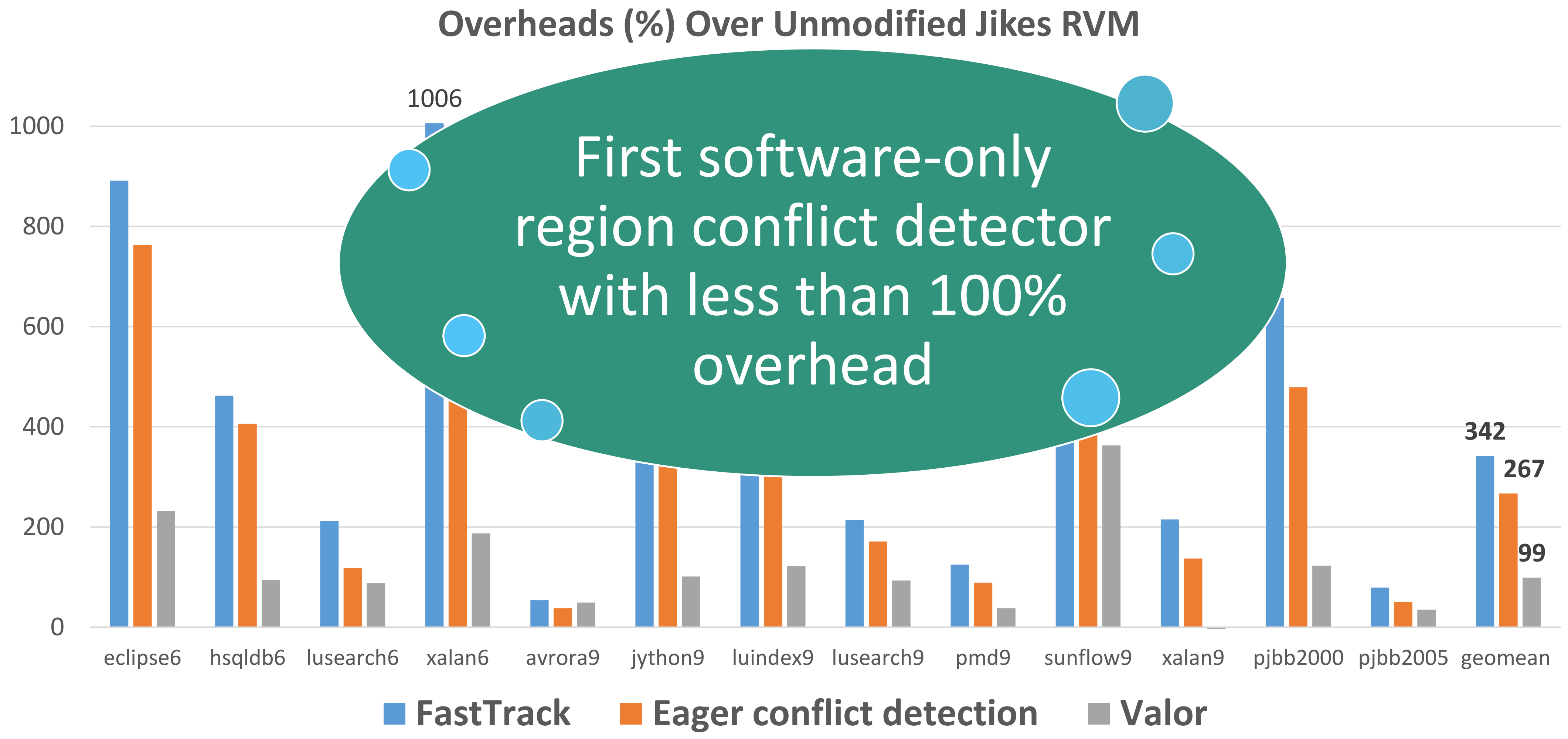


Performance Comparison

Overheads (%) Over Unmodified Jikes RVM

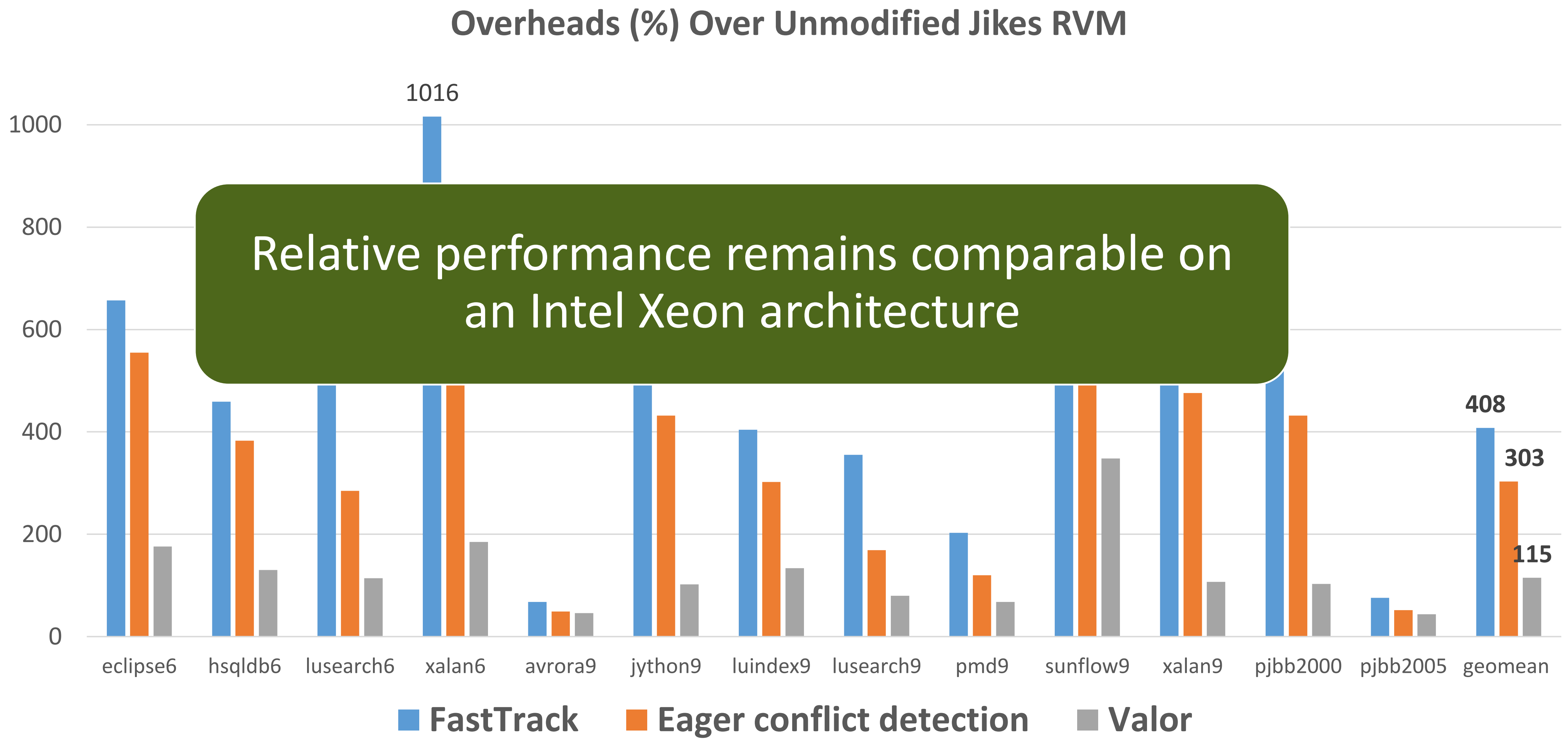


Performance Comparison



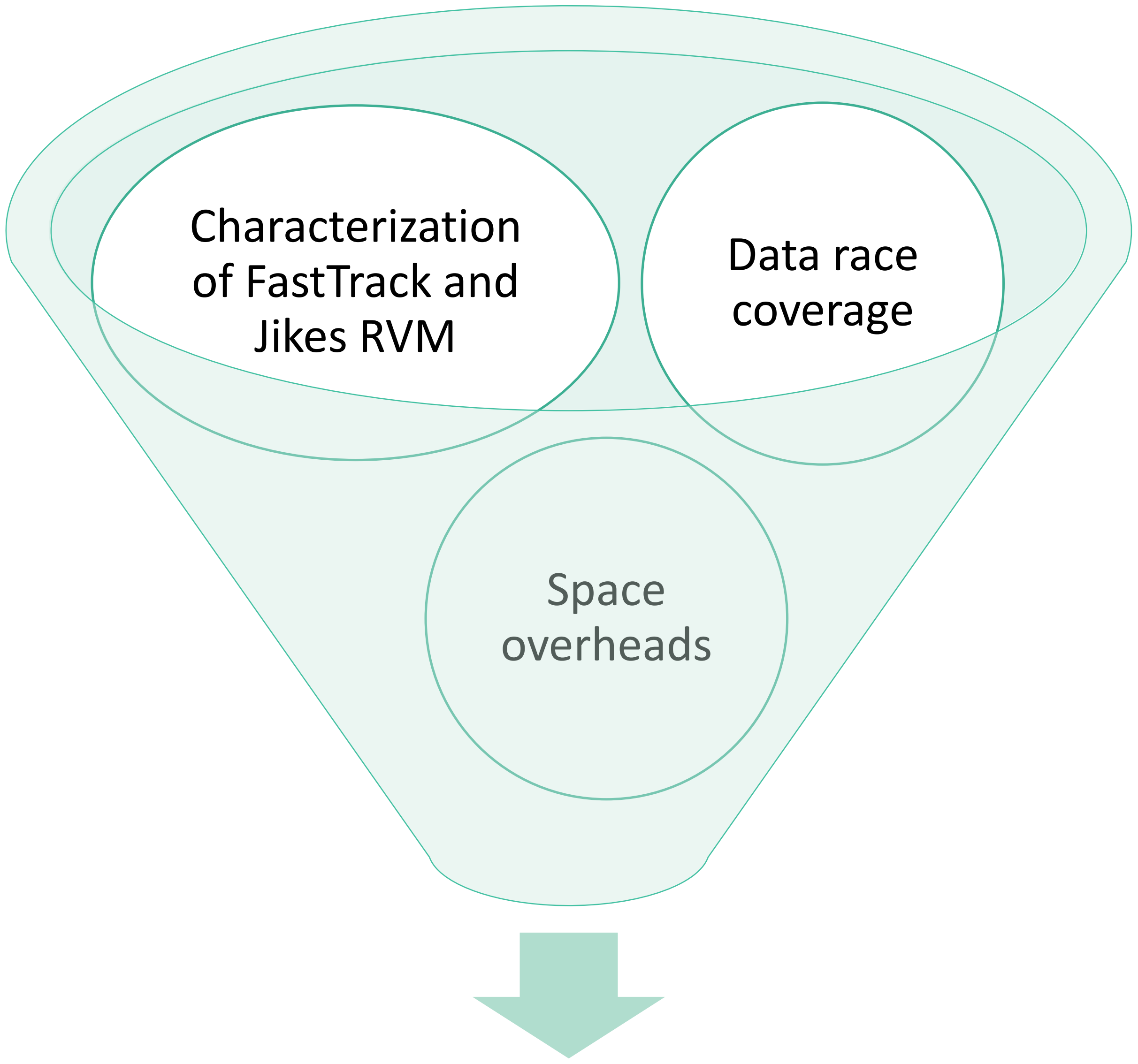
First software-only region conflict detector with less than 100% overhead

Performance Comparison: Intel Xeon



Relative performance remains comparable on an Intel Xeon architecture

Additional Experiments



Please check the paper

Valor: Contributions

Strong execution guarantees in software

Detects all violations of region serializability

$$\begin{tikzcd}\text{Exception-free execution} \arrow[r] & \text{Strong semantics}\end{tikzcd}$$

Advances state-of-art

- ❖ Provides strong semantics in software at less than 100% overhead

New Opportunities with Valor

Semantic guarantees	Language runtimes could **integrate** this
Debugging	Can be used to detect **problematic** data races
Conflict exceptions	**All-the-time monitoring** in certain environments
Aggressive optimizations	**Reorder and eliminate** redundant loads and stores within synchronization-free regions

Valor: Efficient, Software-Only Region Conflict Exceptions

Swarnendu Biswas, Minjia Zhang, and
Michael D. Bond
Ohio State University
Brandon Lucia
Carnegie Mellon University

OOPSLA 2015