Legato: Bounded Region Serializability Using Commodity Hardware Transactional Memory

Aritra Sengupta
Man Cao
Michael D. Bond
and
Milind Kulkarni

Programming Language Semantics?

- Data Races
  - C++ no guarantee of semantics – "catch-fire" semantics
  - Java provides weak semantics

Weak Semantics

```
Data data = null;
boolean done= false;
```

T1

```
data = new Data();
done = true;
```

T2

```
if (done)
  data.foo();
```

Weak Semantics

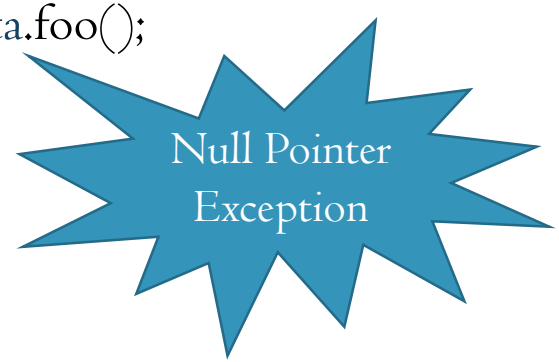
```
Data data = null;
boolean done= false;
```

T1

```
data = new Data();
done = true;
```

T2

```
if (done)
  data.foo();
```



Weak Semantics

```
Data data = null;
boolean done= false;
```

T1

```
data = new Data();
done = true;
```

No data dependence: Reordering effect

T2

```
if (done)
  data.foo();
```

Null Pointer Exception

Need for Stronger Memory Models

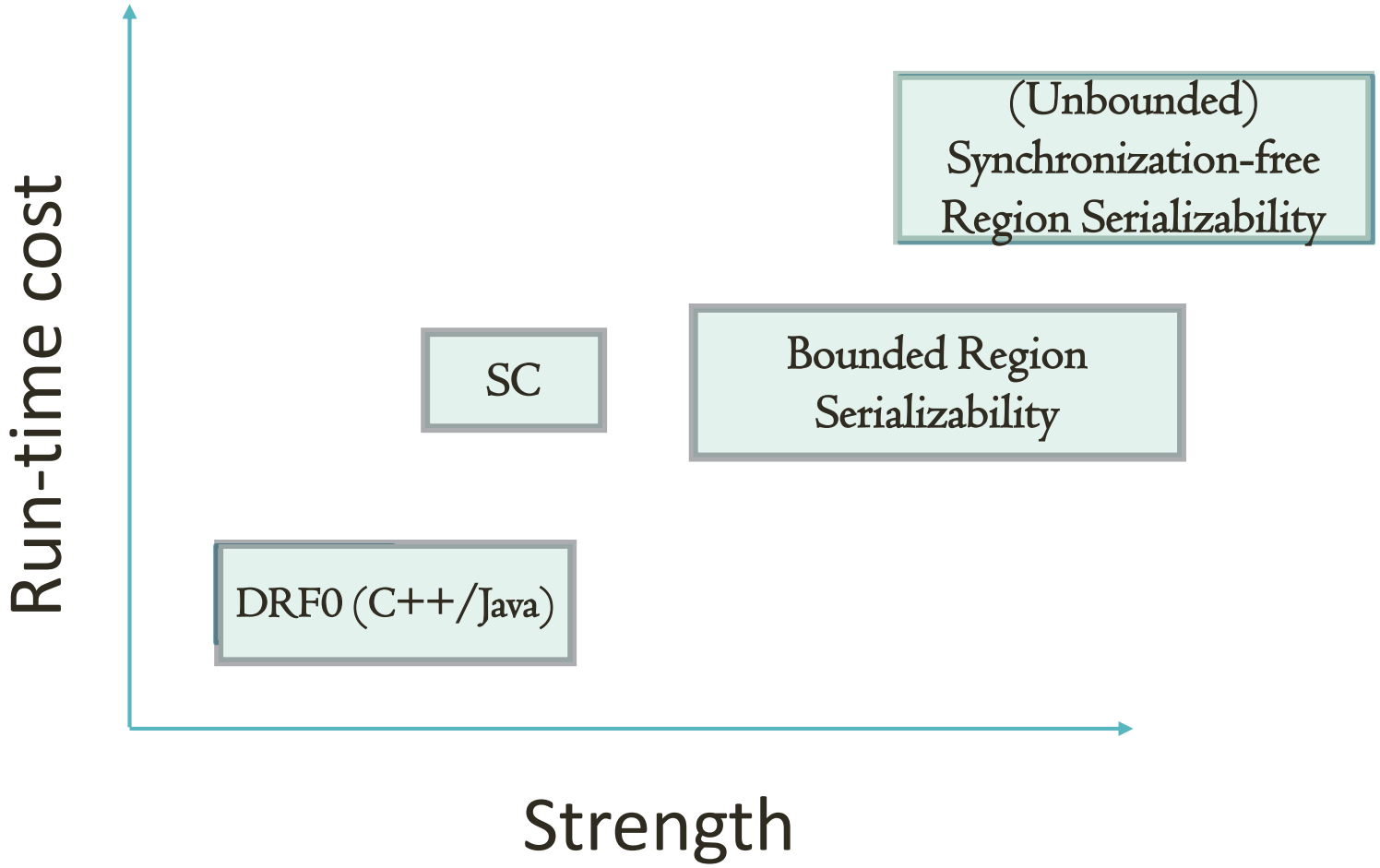
"The inability to define reasonable semantics for programs with data races is not just a theoretical shortcoming, but a fundamental hole in the foundation of our languages and systems…"

Give better semantics to programs with data races

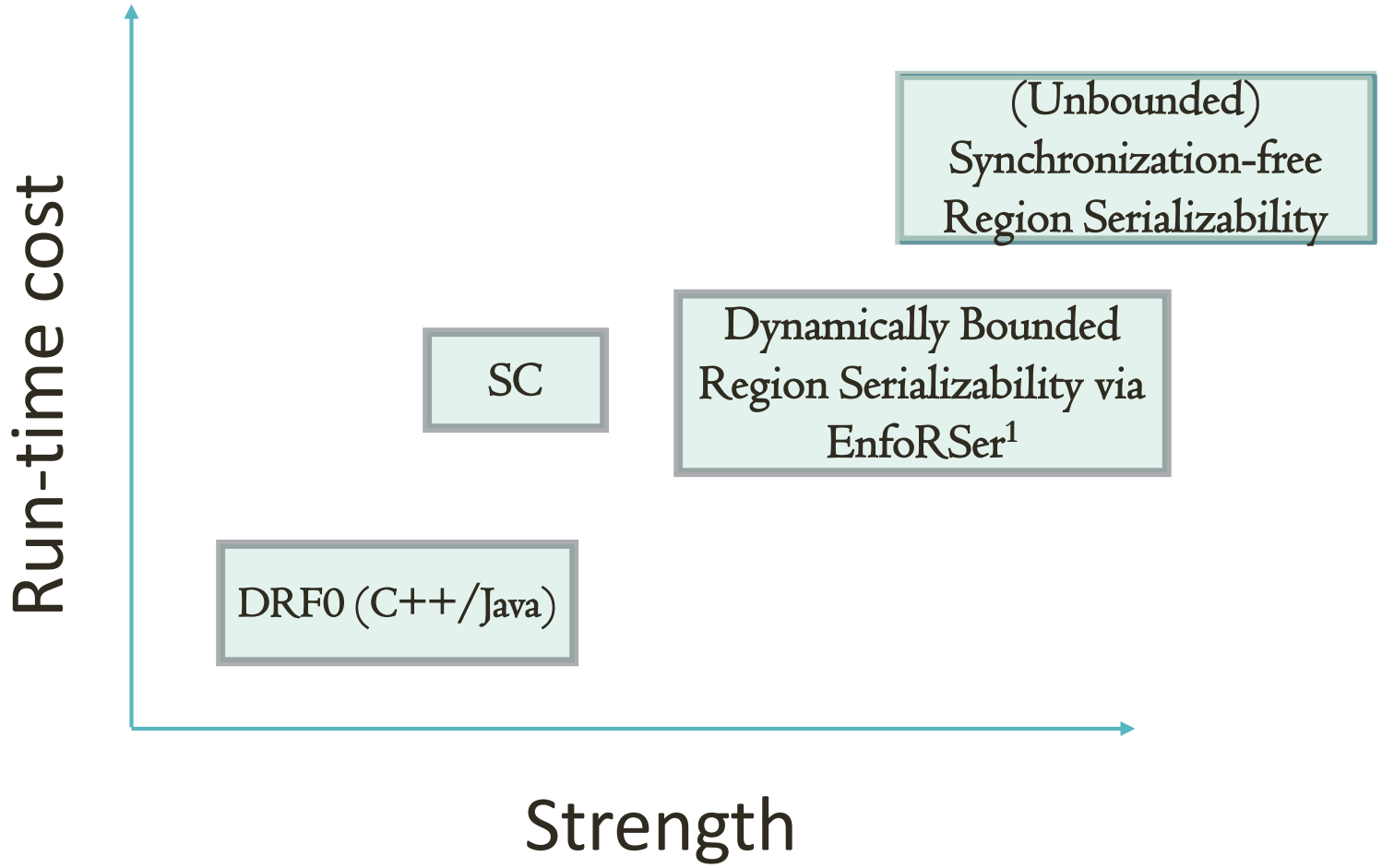
Stronger memory models

– Adve and Boehm, CACM, 2010

End-to-End Memory Models: Run-time cost vs Strength

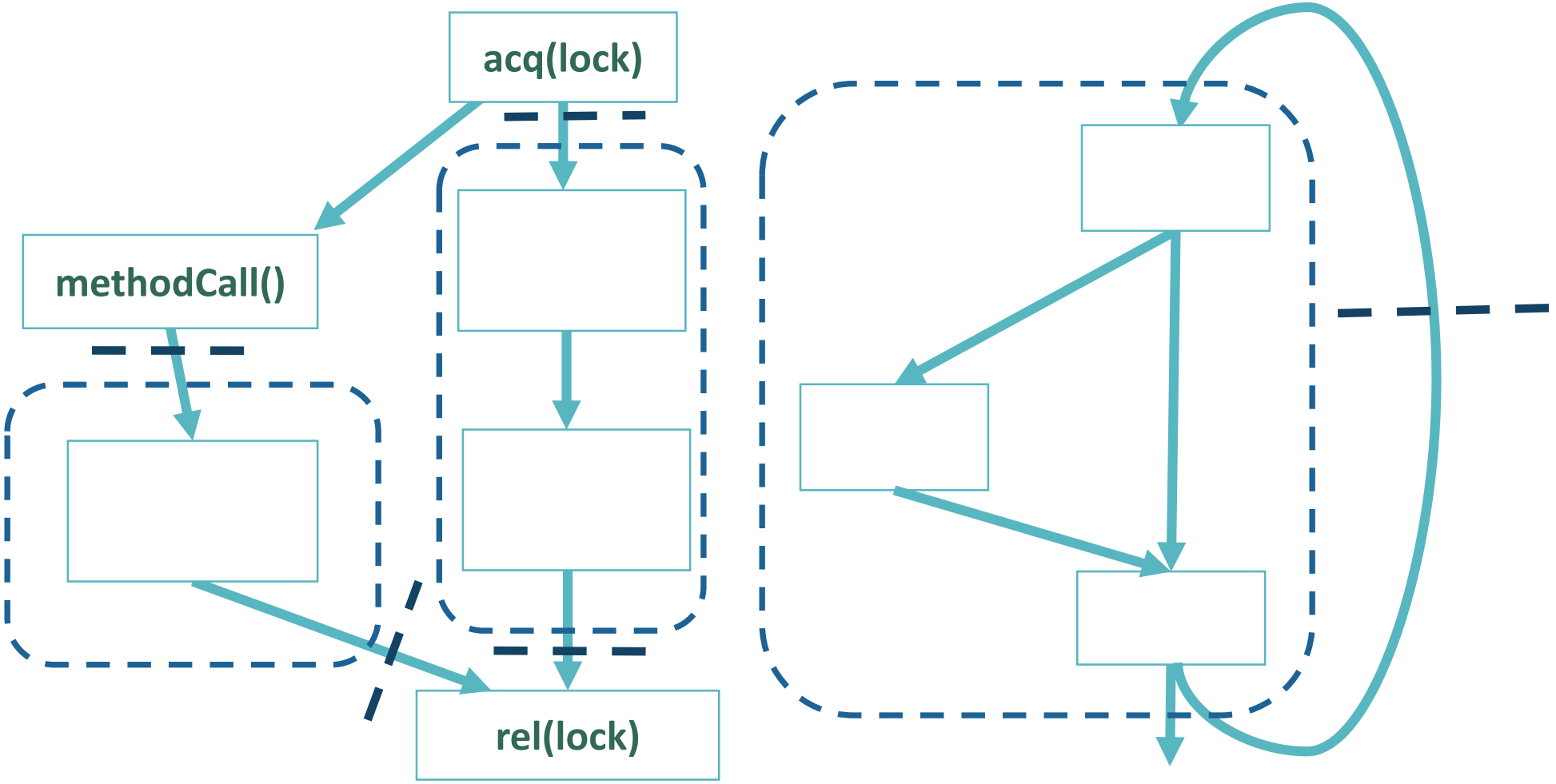


End-to-End Memory Models: Run-time cost vs Strength



1. Sengupta et al. Hybrid Static-Dynamic Analysis for Statically Bounded Region Serializability. ASPLOS, 2015.

Dynamically Bounded Region Serializability (DBRS)



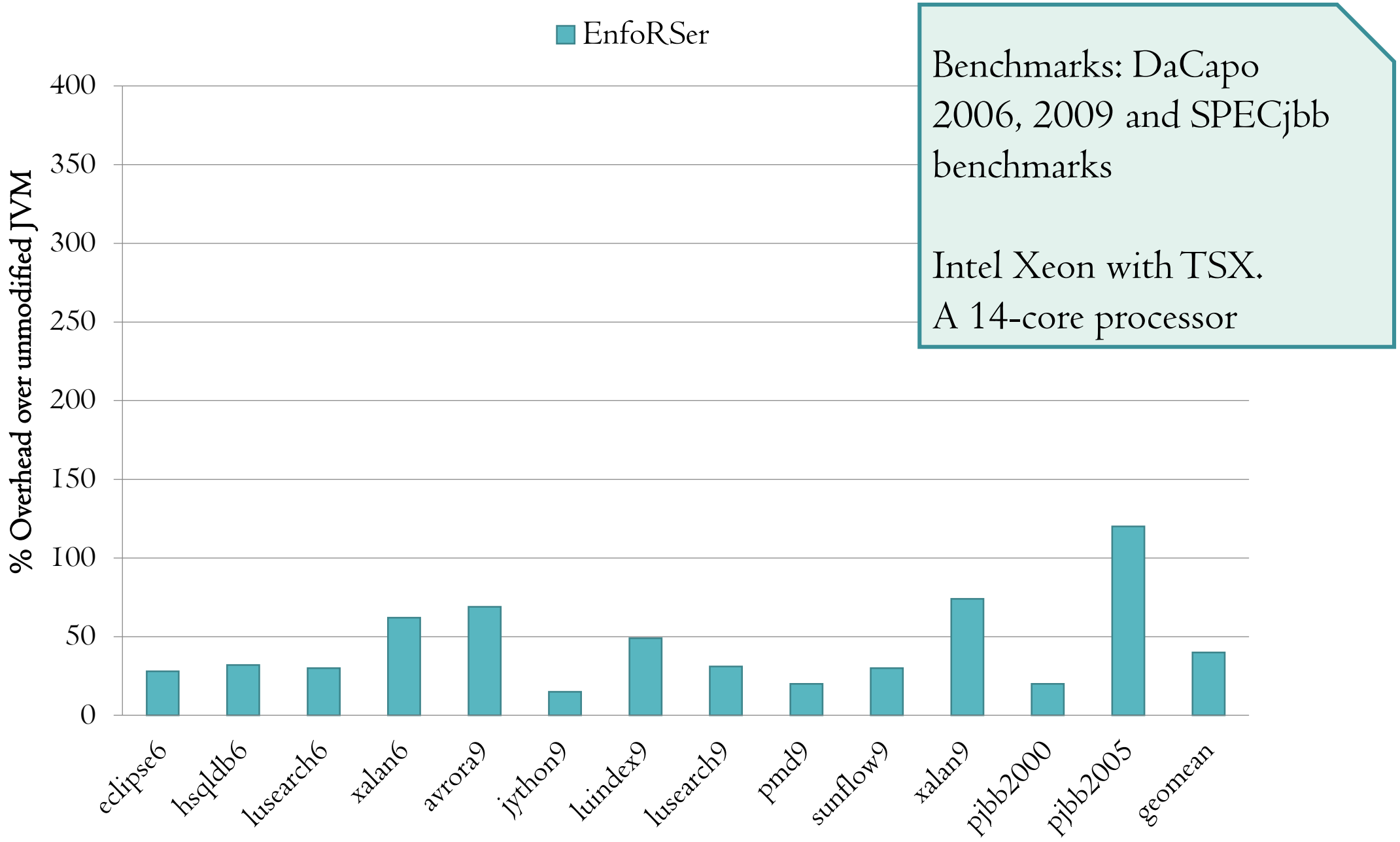
Dynamically Bounded Region Serializability

SC+Atomicity of bounded regions[1]

- Eliminates SC violation
- Eliminates some atomicity violations

	DRF0	SC	DBRS
hsqldb6	Infinite loop	Correct	Correct
sunflow9	Null pointer exception	Correct	Correct
jbb2000	Corrupt output	Corrupt output	Correct
jbb2000	Infinite loop	Correct	Correct
sor	Infinite loop	Correct	Correct
lufact	Infinite loop	Correct	Correct
moldyn	Infinite loop	Correct	Correct
raytracer	Fails validation	Fails validation	Correct

1. Sengupta et al. Hybrid Static-Dynamic Analysis for Statically Bounded Region Serializability. ASPLOS, 2015.

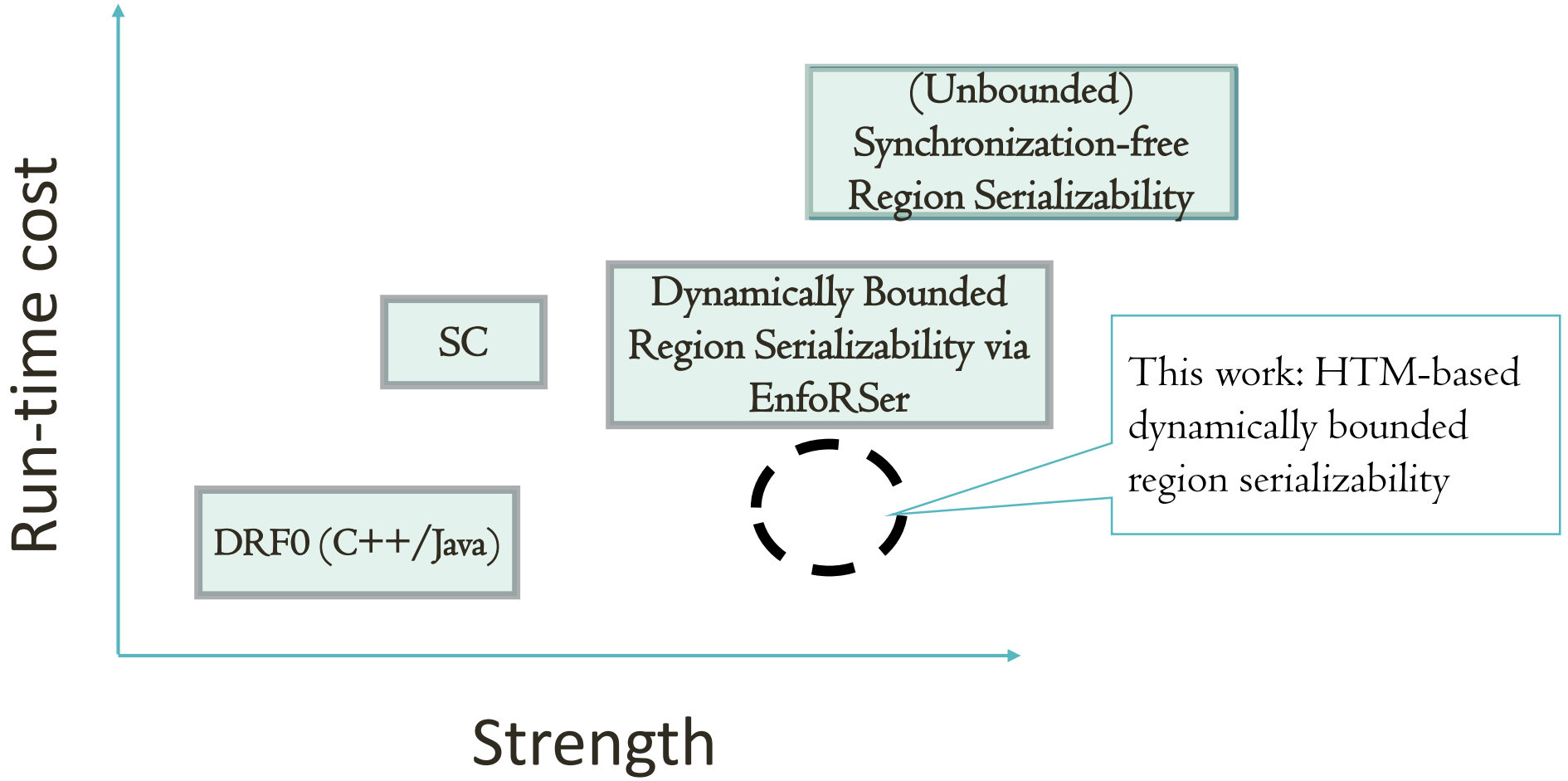


Benchmarks: DaCapo 2006, 2009 and SPECjbb benchmarks

Intel Xeon with TSX.
A 14-core processor

1. Sengupta et al. Hybrid Static-Dynamic Analysis for Statically Bounded Region Serializability. ASPLOS, 2015.

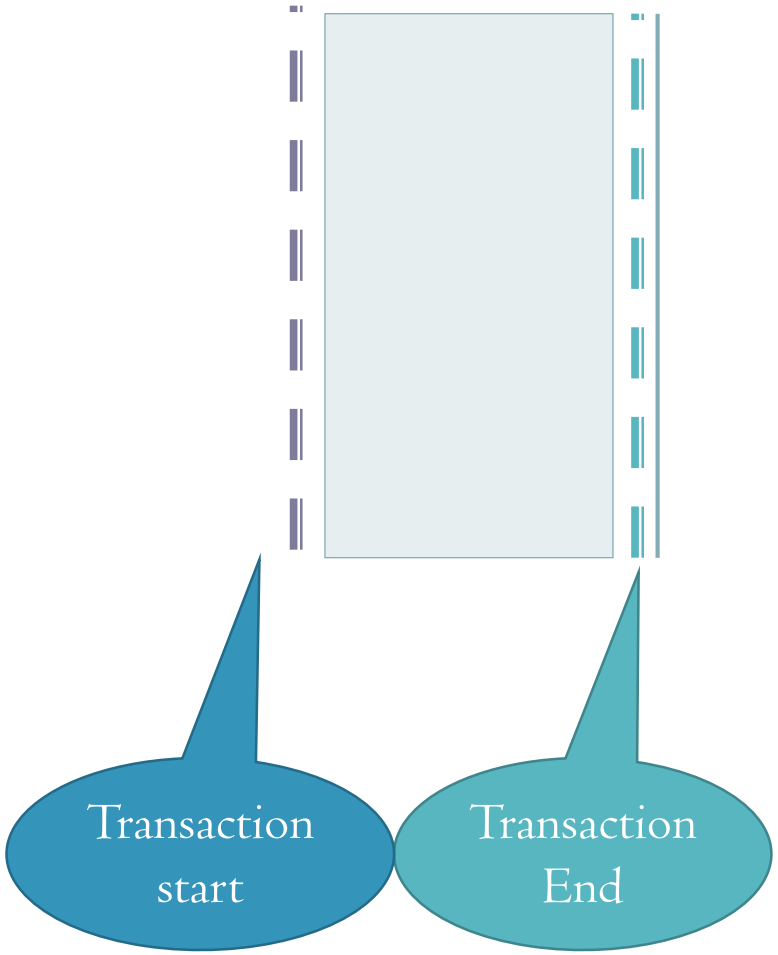
End-to-End Memory Models: Run-time cost vs Strength



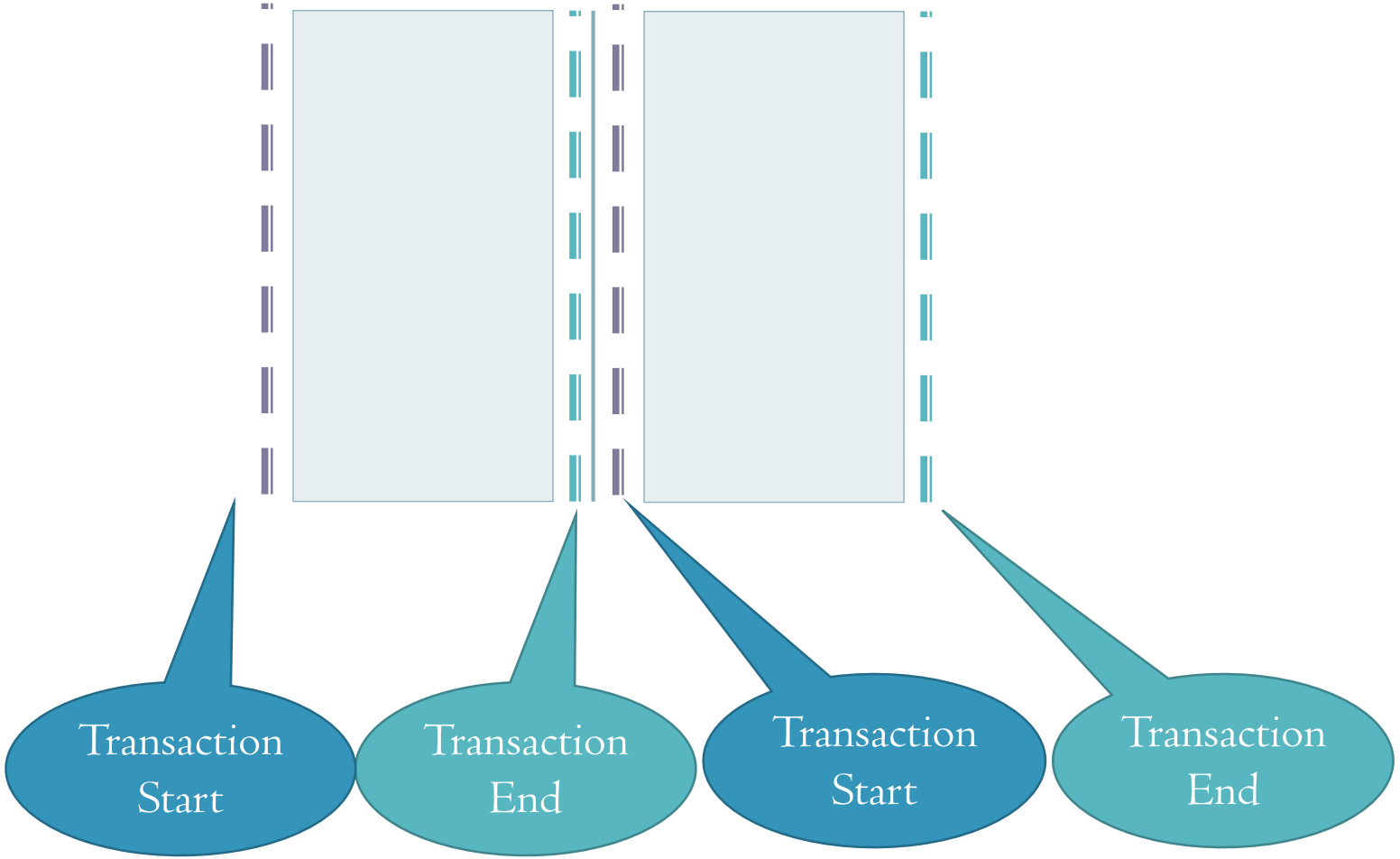
Outline

- Challenges
  - Naïve implementation with hardware transactional memory
  - Limitations of using HTM for DBRS
- Approach
  - Overcoming limitations
  - Our approach to DBRS enforcement: **Legato**
- Evaluation

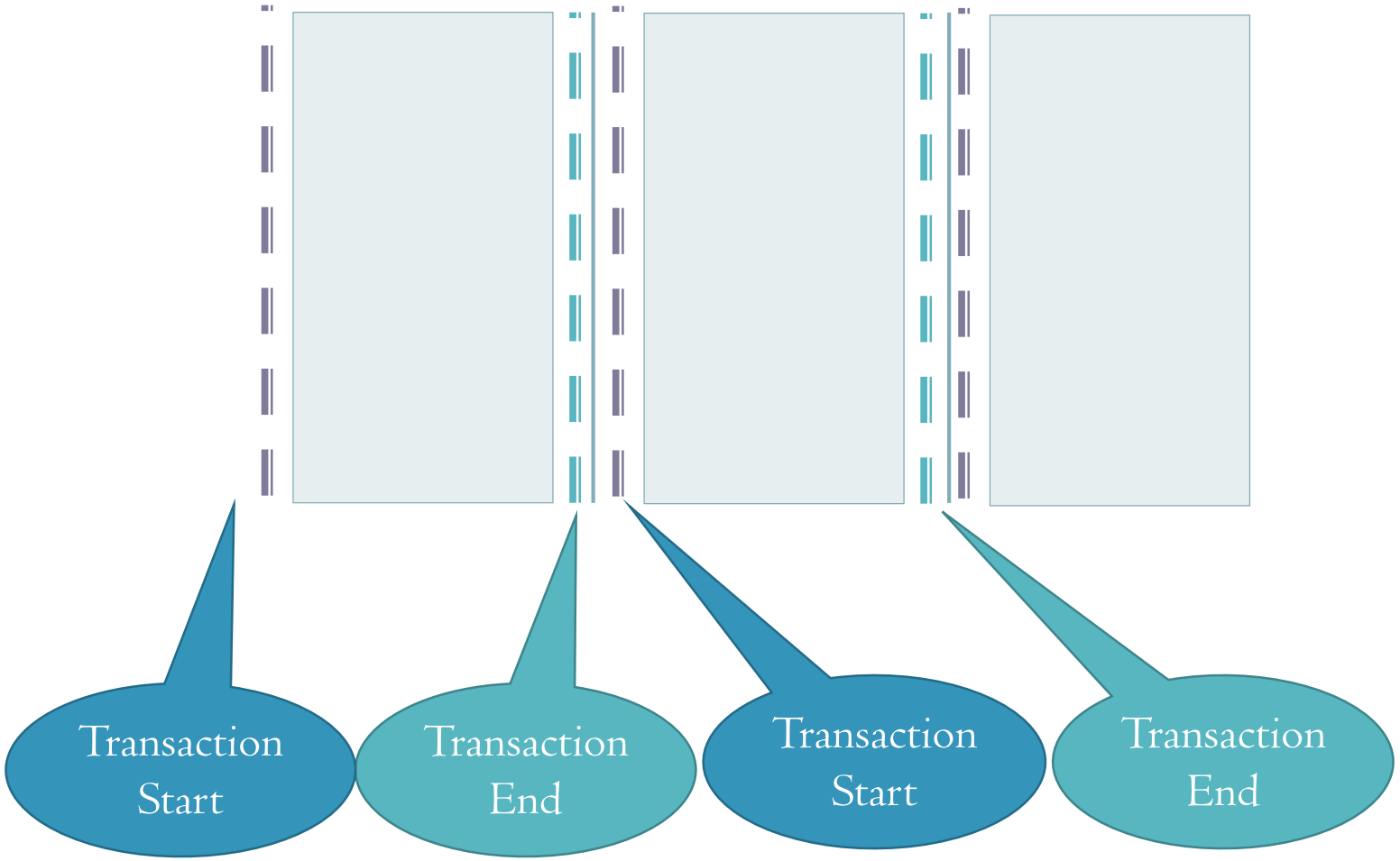
Enforcing DBRS with Commodity Hardware Transactional Memory (HTM)

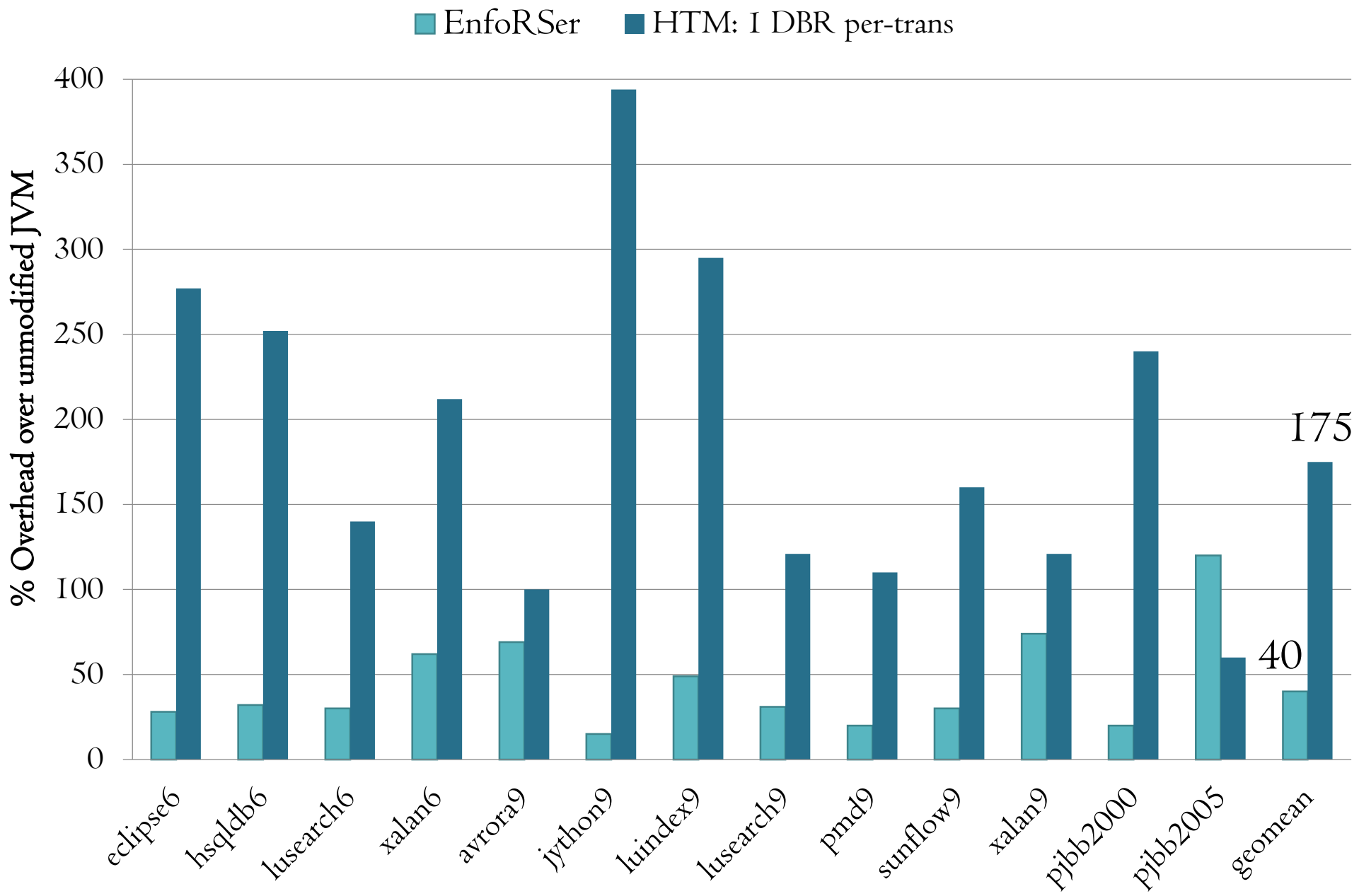


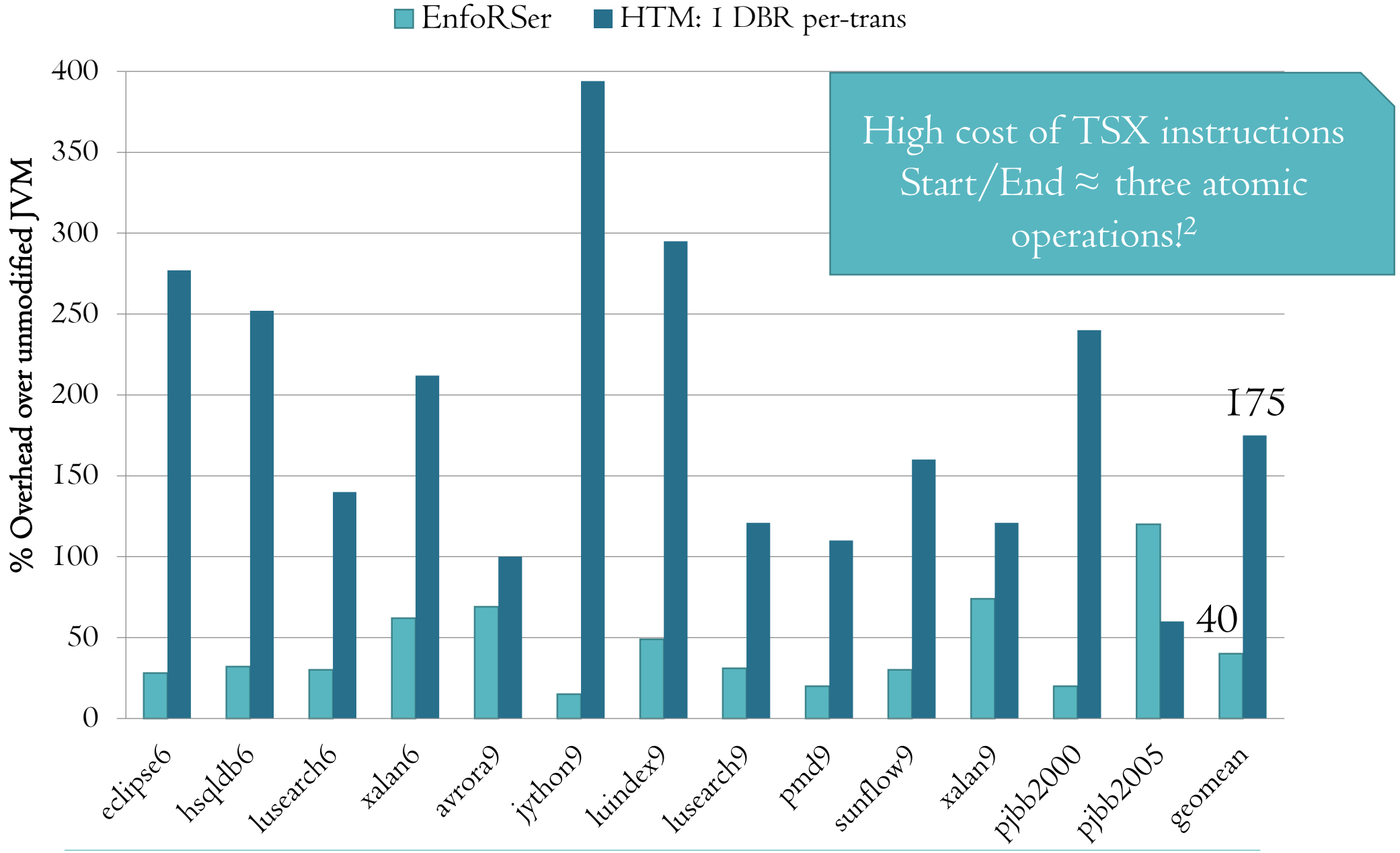
Enforcing DBRS with Commodity Hardware Transactional Memory (HTM)



Enforcing DBRS with Commodity Hardware Transactional Memory (HTM)




EnfoRSer
HTM: 1 DBR per-trans
% Overhead over unmodified JVM
400
350
300
250
200
150
100
50
0
175
40
eclipse6
hsqldb6
lusearch6
xalan6
avrora9
jython9
luindex9
lusearch9
pmd9
sunflow9
xalan9
pjbb2000
pjbb2005
geomean

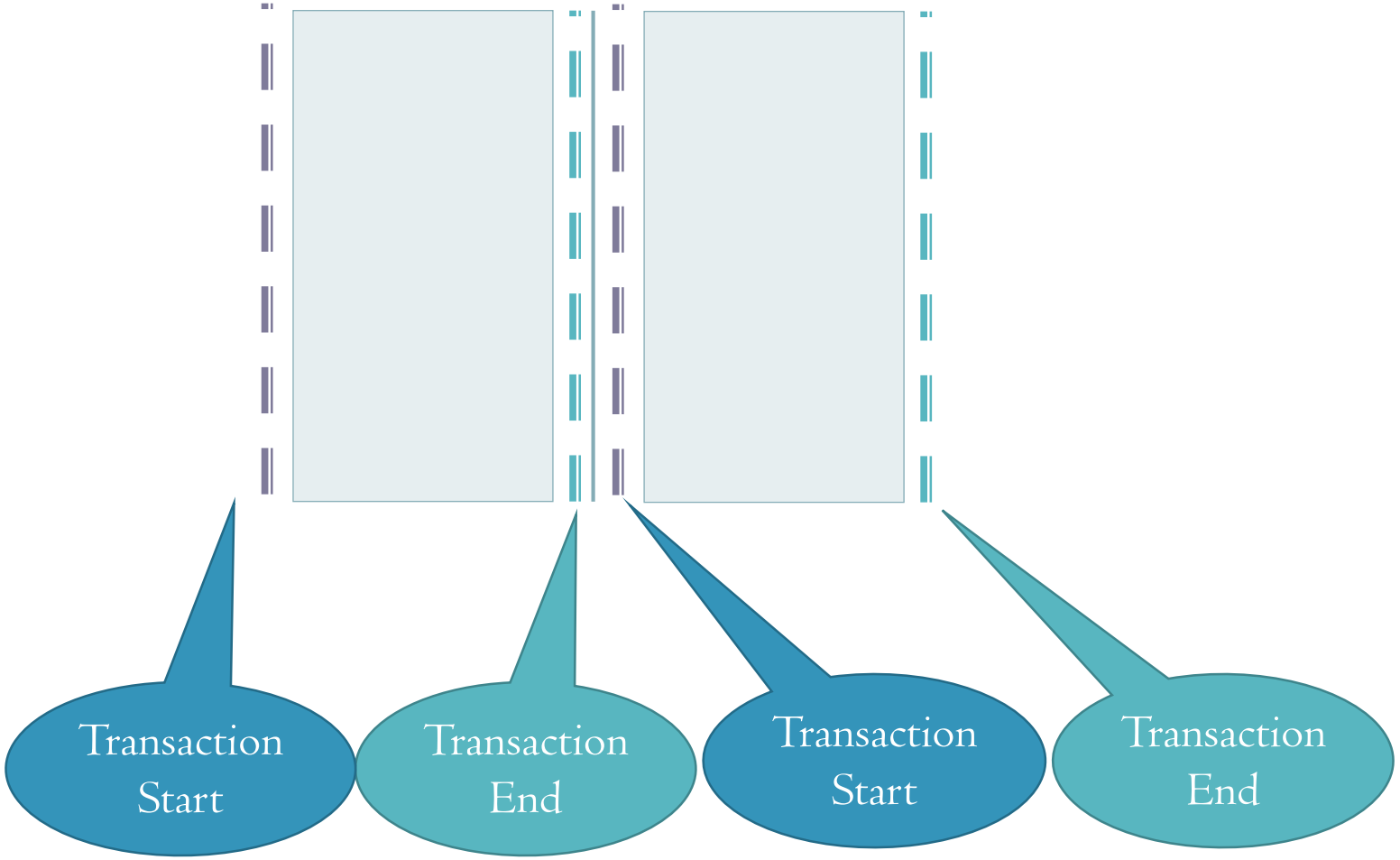


2. Yoo et al. Performance Evaluation of Intel TSX for High-Performance Computing, SC 2013

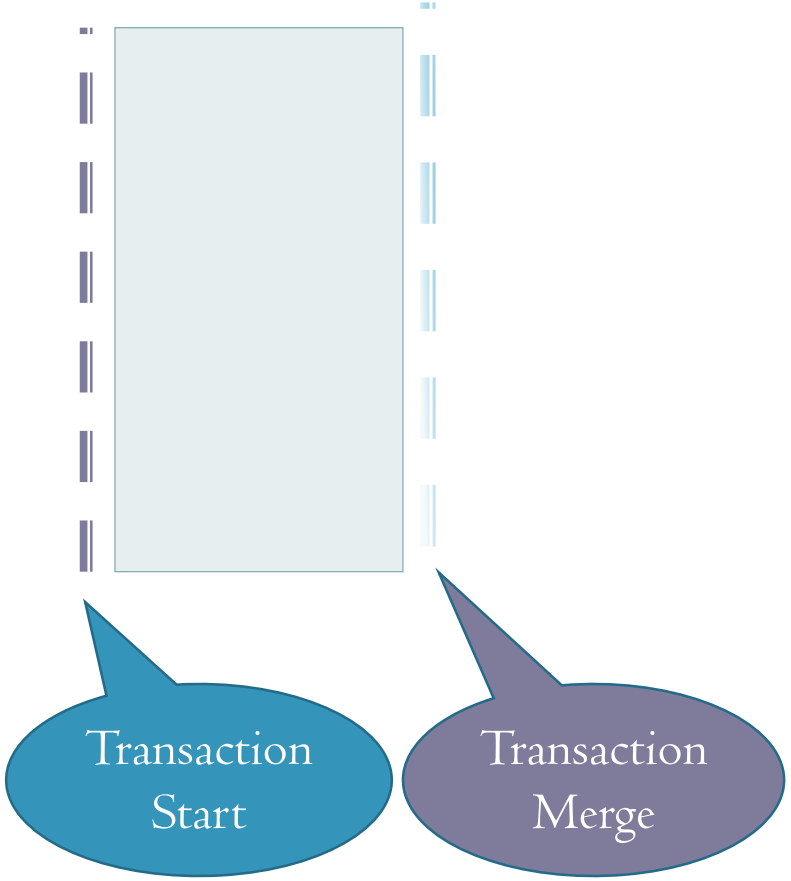
Legato: Key Idea

Merge several regions into a single transaction:
amortize the cost of starting and stopping a transaction

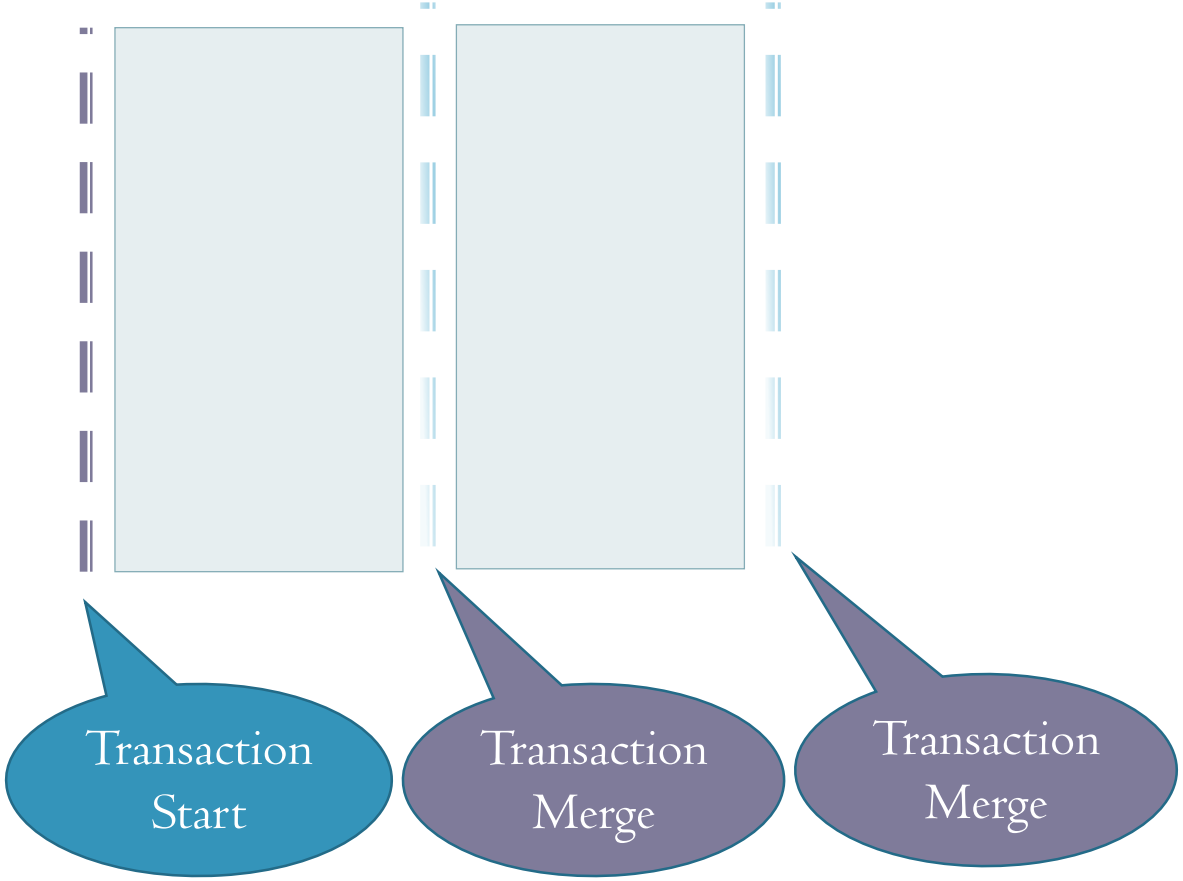
Legato: Key Idea



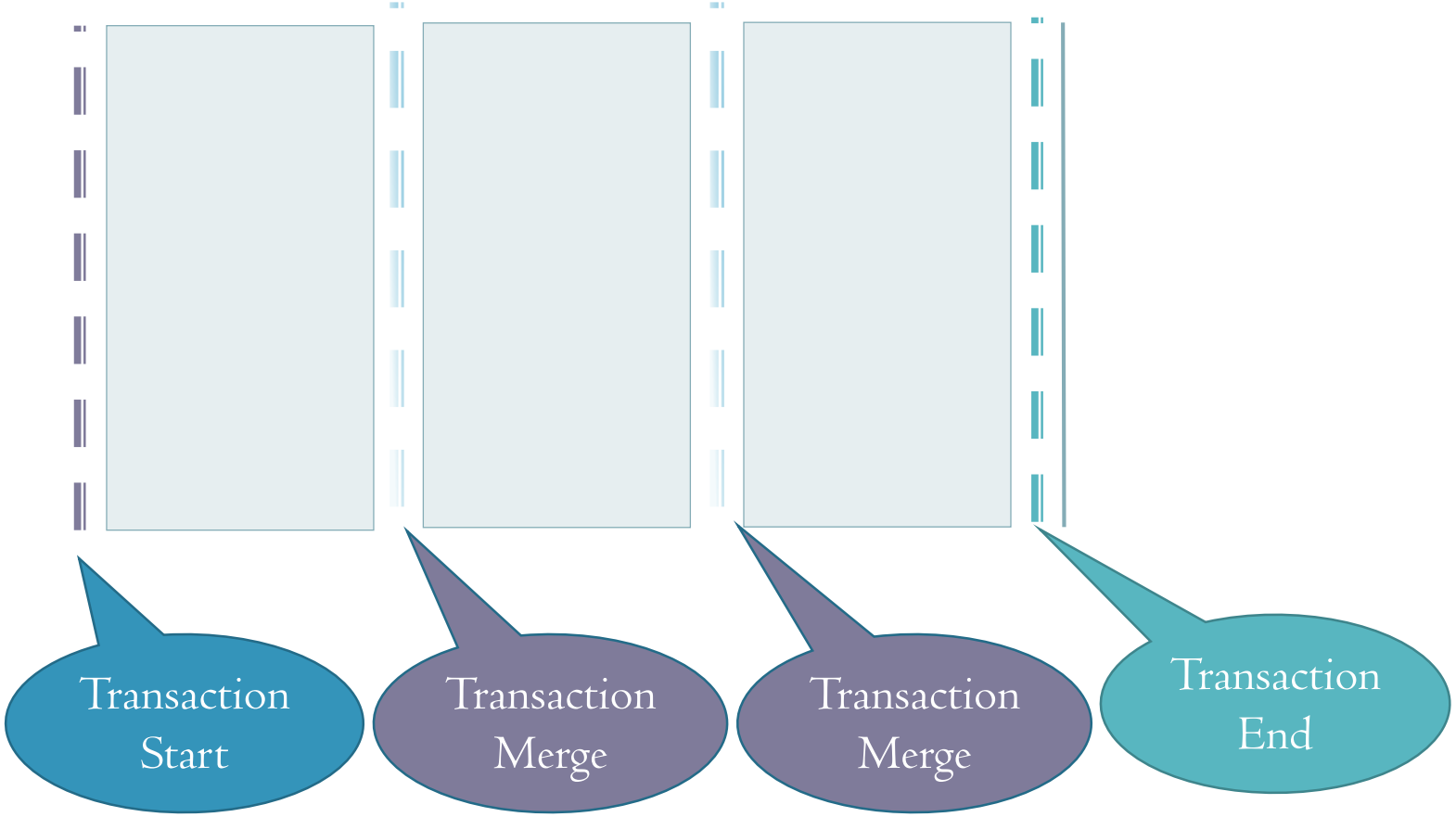
Legato: Key Idea



Legato: Key Idea



Legato: Key Idea



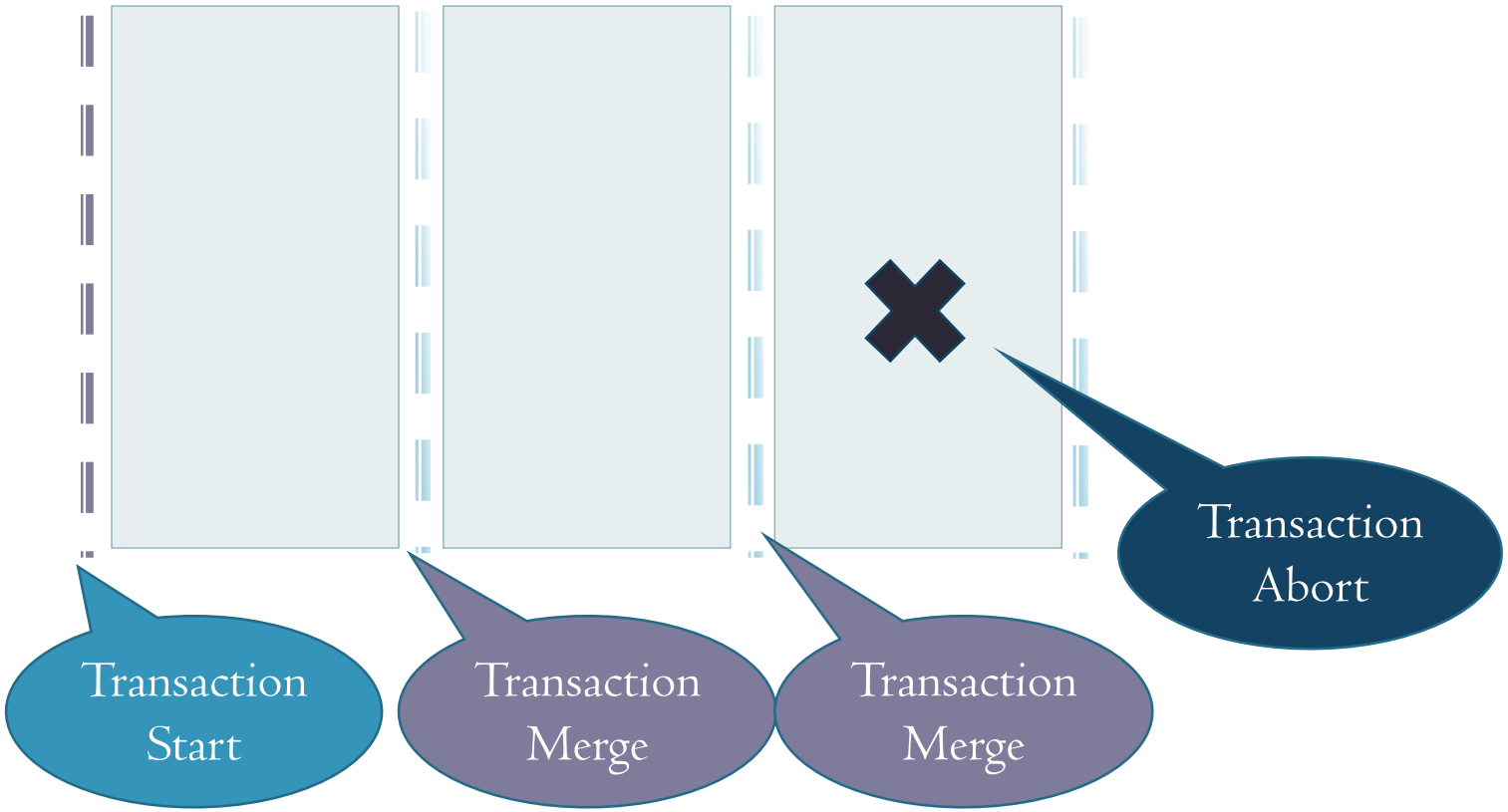
Challenges

Conflicts abort transactions: wasted work

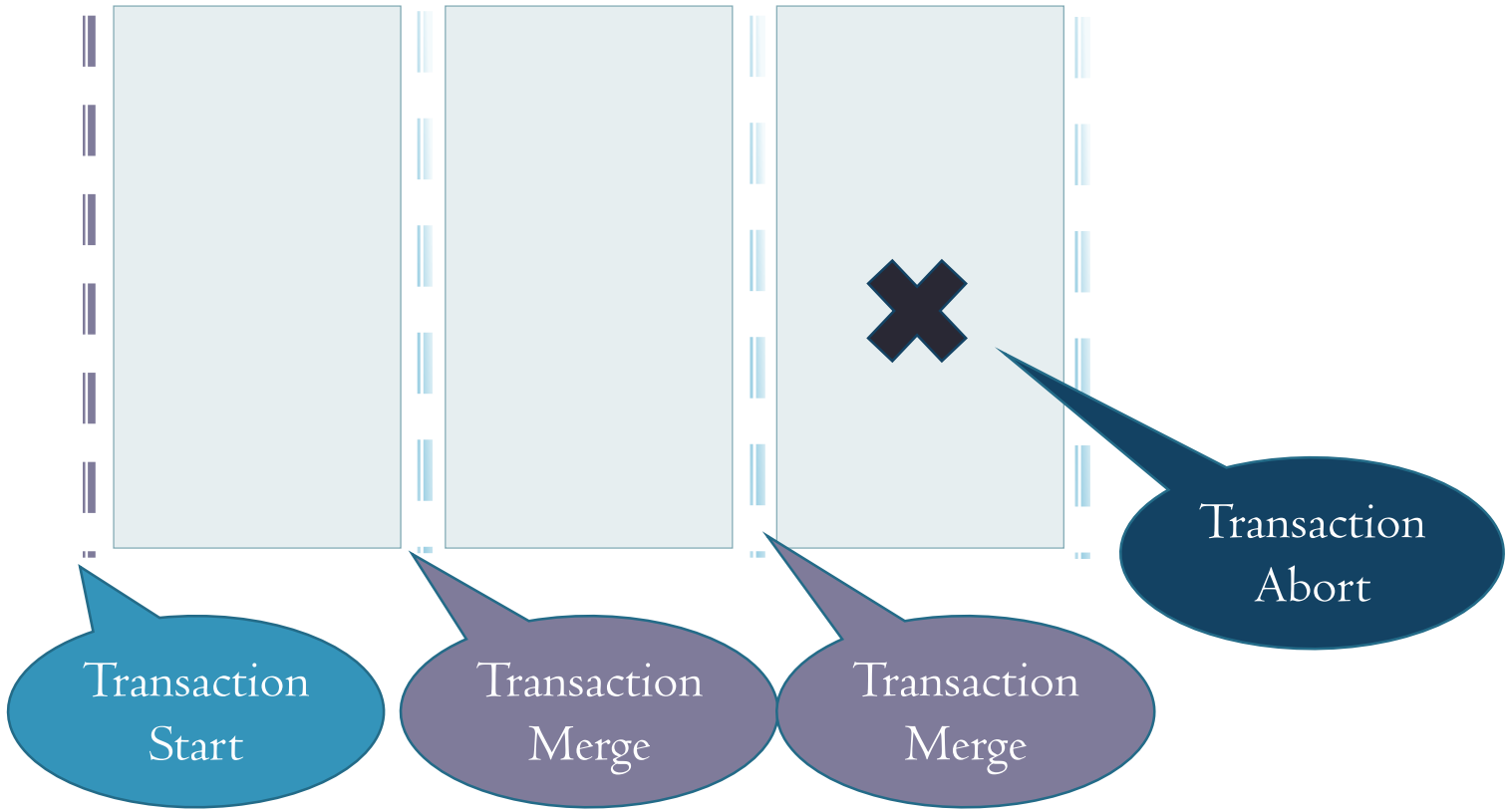
Capacity aborts: larger transactions have **larger footprint**, **unknown *a priori***

HTM-unfriendly operations: hardware interrupts, page faults etc.

Per-transaction cost vs Abort cost



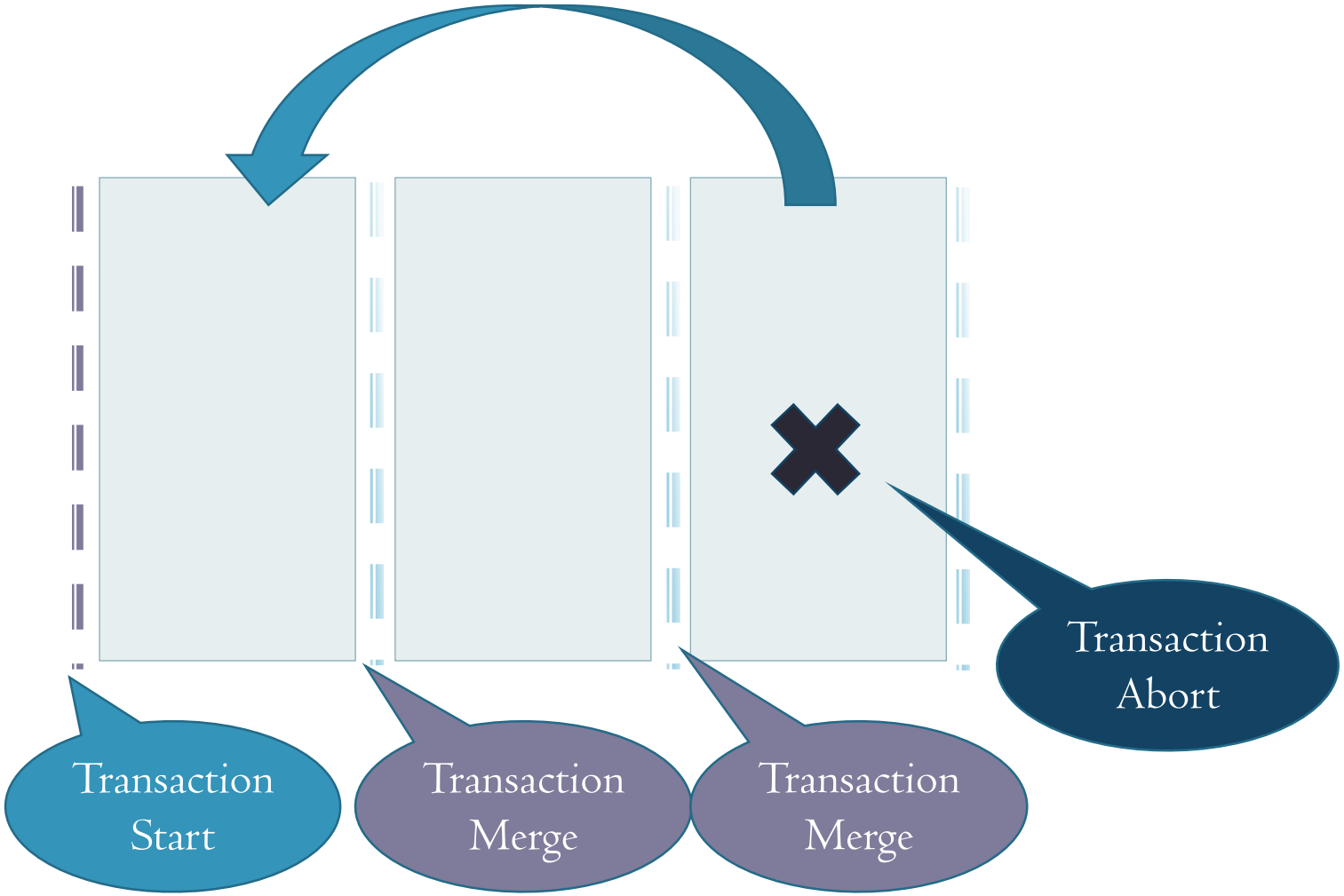
Per-transaction cost vs Abort cost



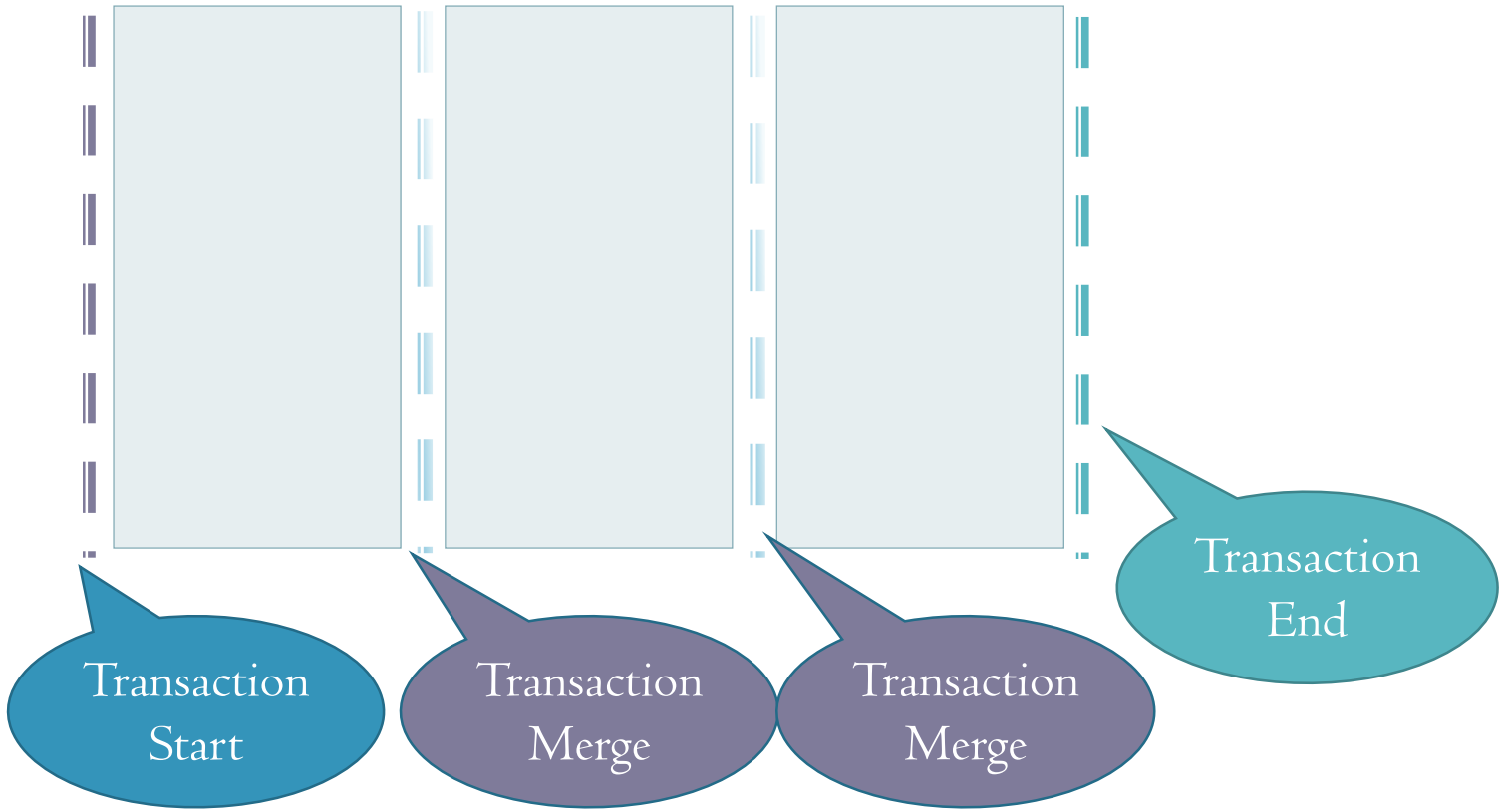
Transient cause: conflicts, hardware interrupts
=>
Retry transaction

Capacity abort: end before culprit region
=>
Start new transaction

Per-transaction cost vs Abort cost



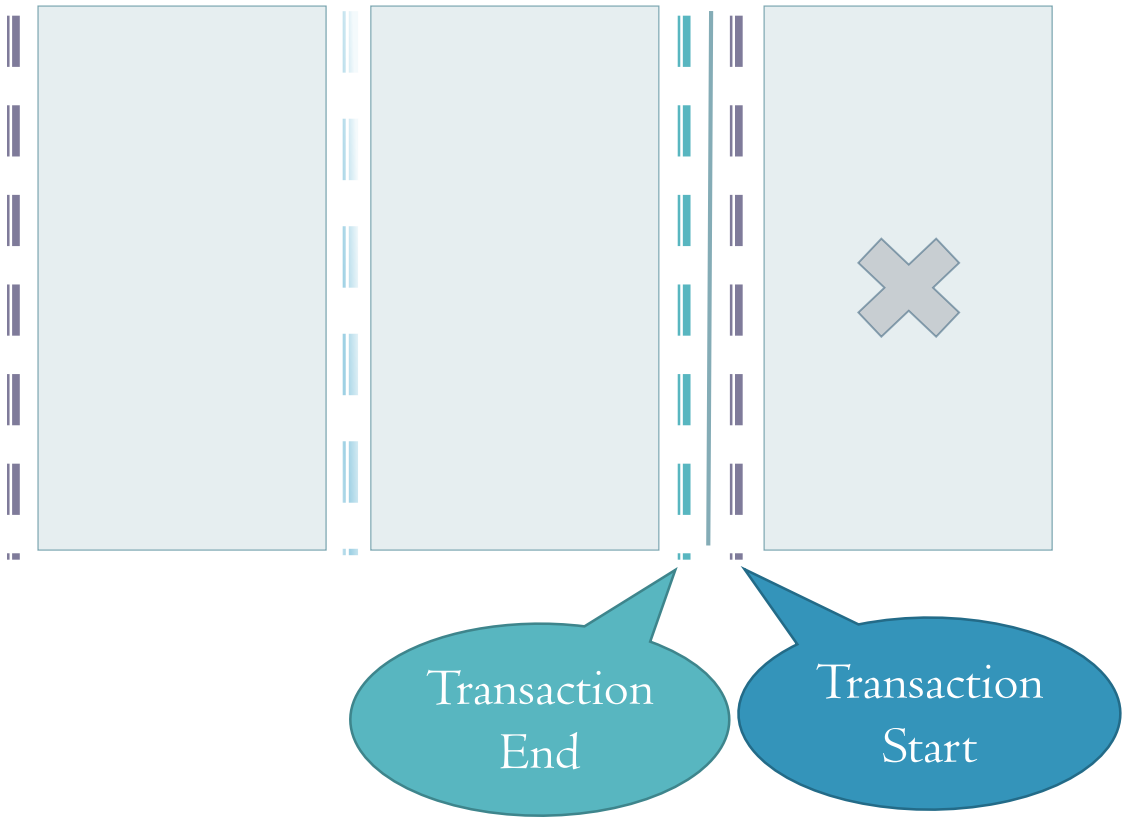
Per-transaction cost vs Abort cost



Transient cause: conflicts, hardware interrupts
=>
Retry transaction

Capacity abort: end before culprit region
=>
Start new transaction

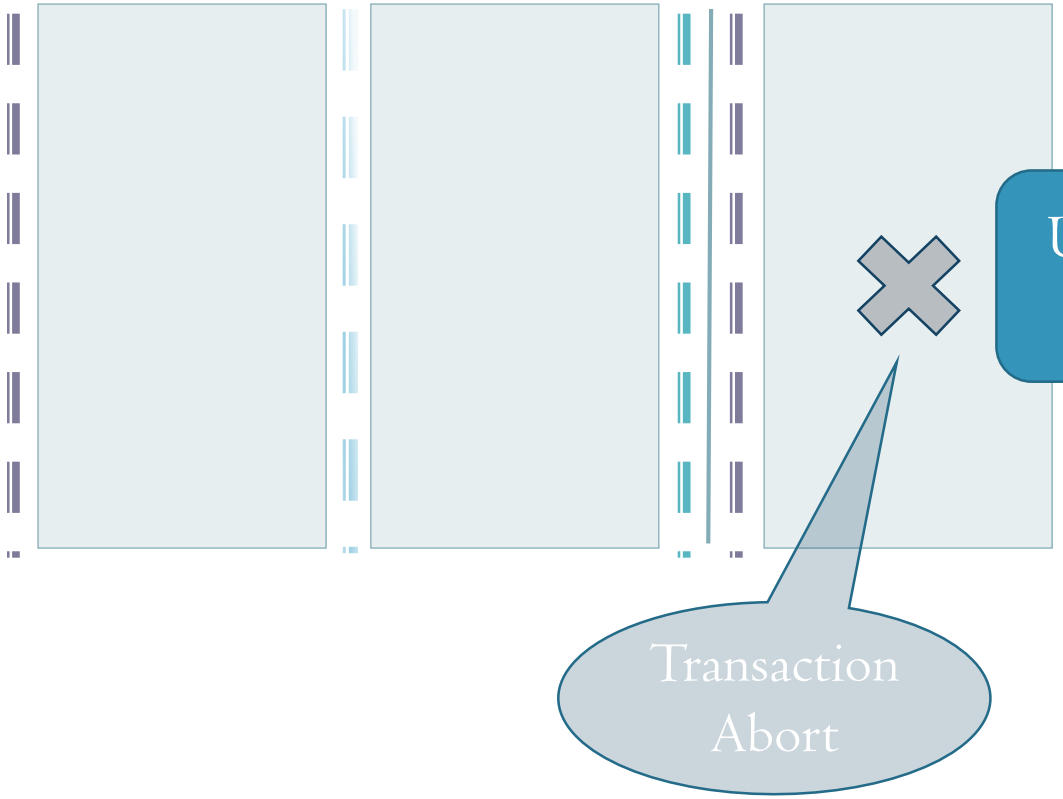
Per-transaction cost vs Abort cost



Transient cause: conflicts, hardware interrupts
=>
Retry transaction

Capacity abort: end before culprit region
=>
Start new transaction

Per-transaction cost vs Abort cost

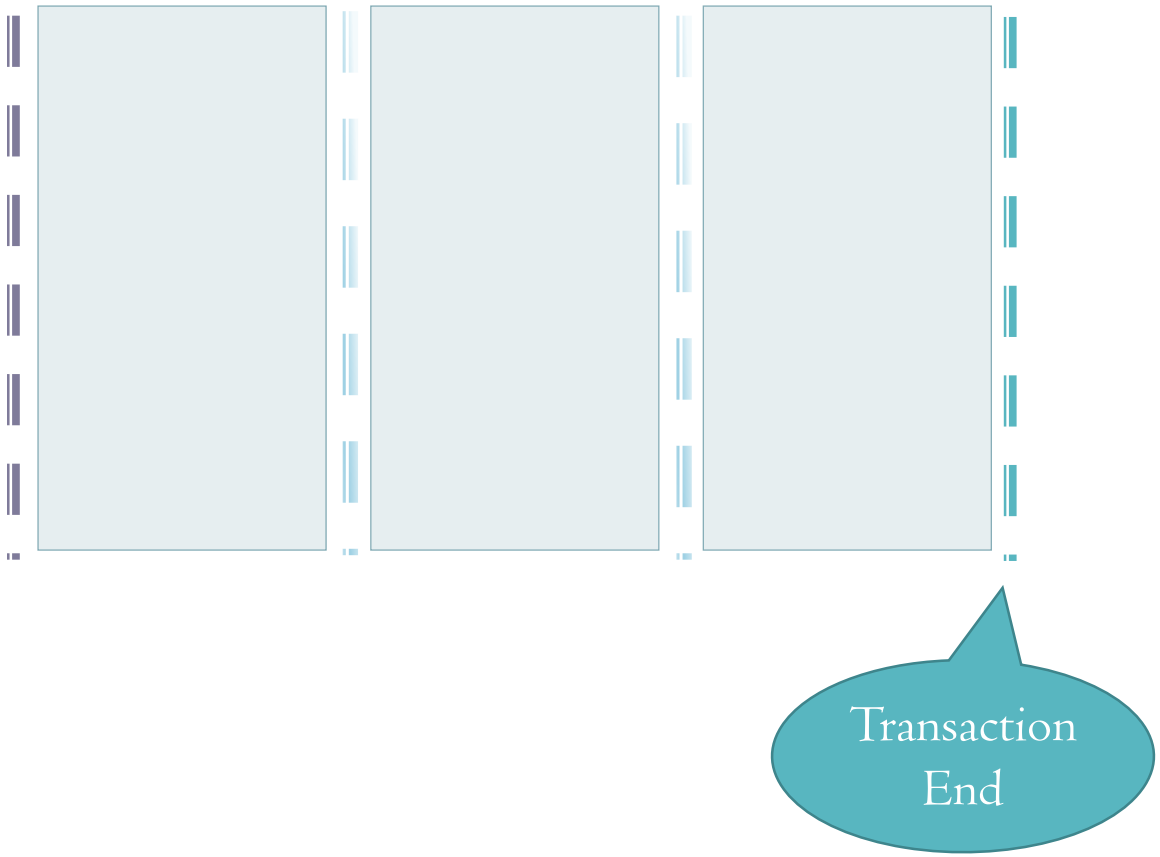


Unknown abort location

Transient cause: conflicts, hardware interrupts
=>
Retry transaction

Capacity abort: end before culprit region
=>
Start new transaction

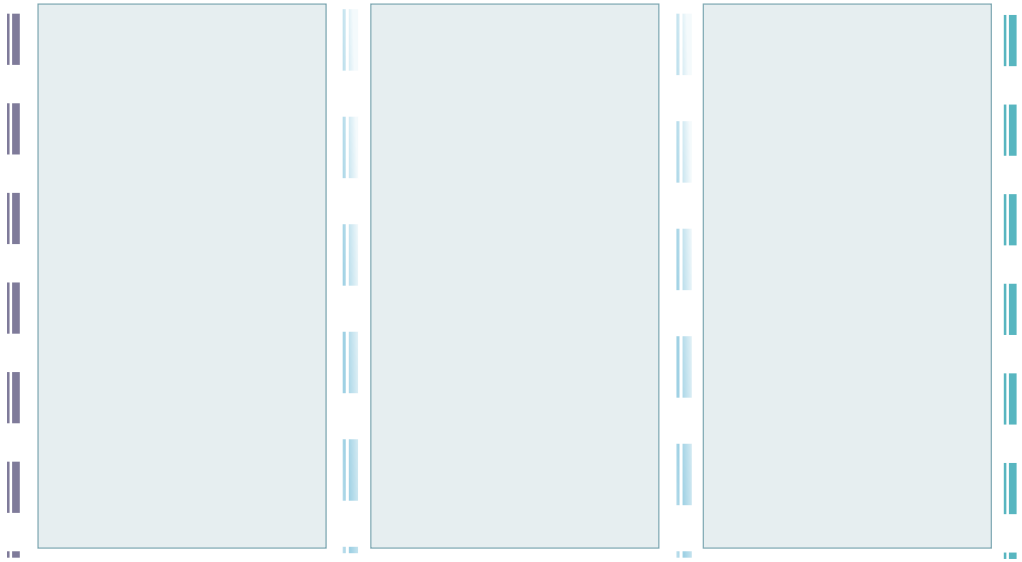
Per-transaction cost vs Abort cost



Transient cause: conflicts, hardware interrupts
=>
Retry transaction

Capacity abort: end before culprit region
=>
Start new transaction

Per-transaction cost vs Abort cost



Transaction End

Minimal Learning

Transient cause: conflicts, hardware interrupts
=>
Retry transaction

Capacity abort: end before culprit region
=>
Start new transaction

Future transaction behavior unknown

Legato: Our Approach

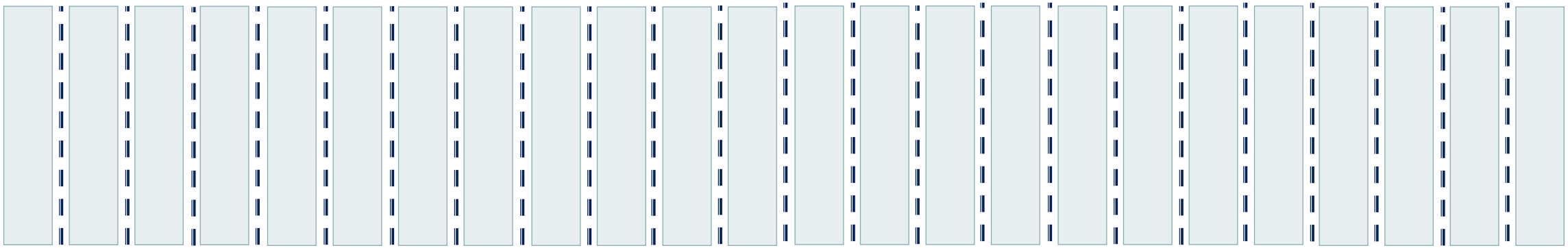
Decide on a merge target: use **history** of previous transaction

1. **Temporary target** changes **rapidly:** capture transient effects

2. **Setpoint** or **"steady state"** target changes **slowly**: capture program phases

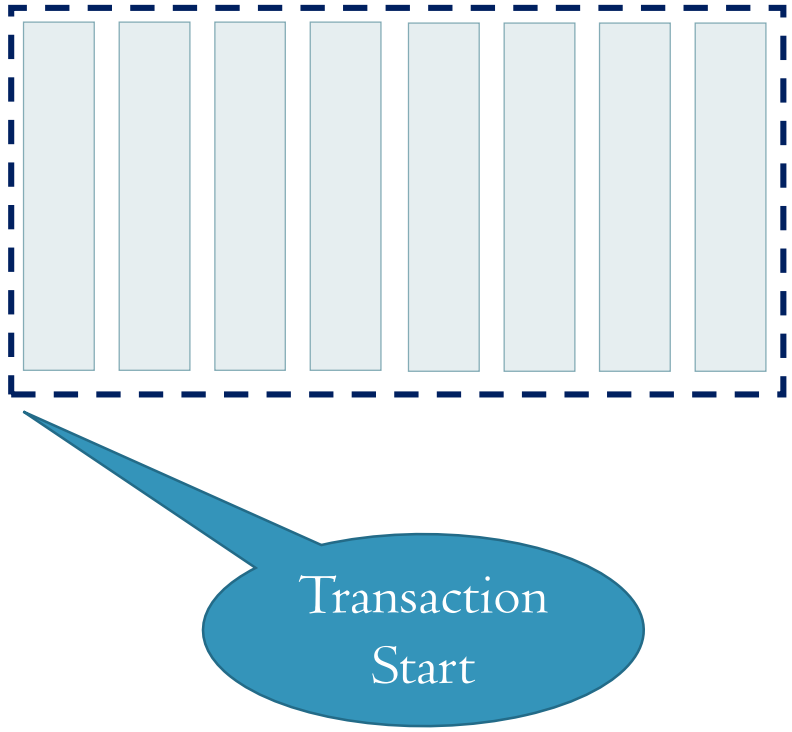
Merging Algorithm

Curr Target: 8
Setpoint: 8



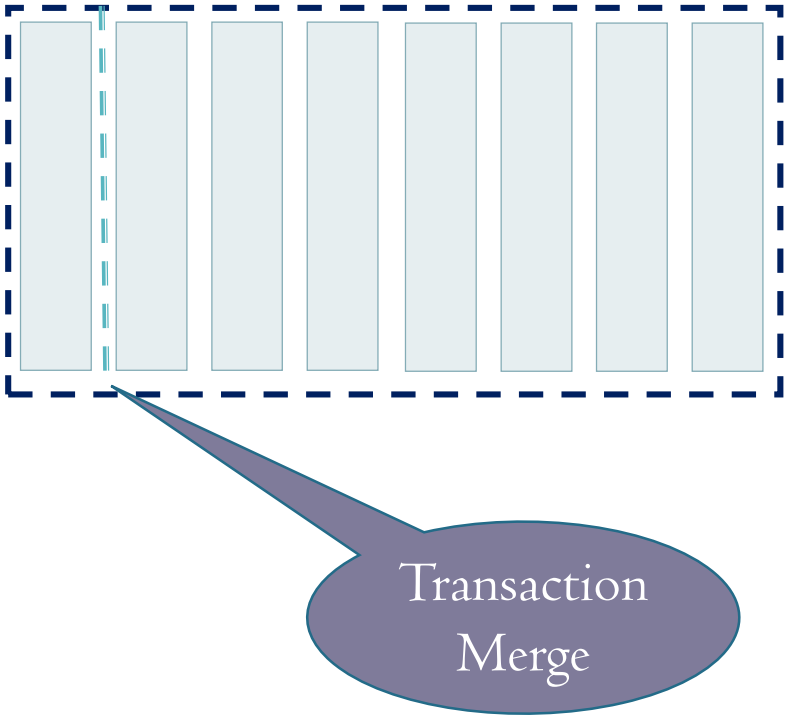
Merging Algorithm: Initial Phase

Curr Target: 8
Setpoint: 8



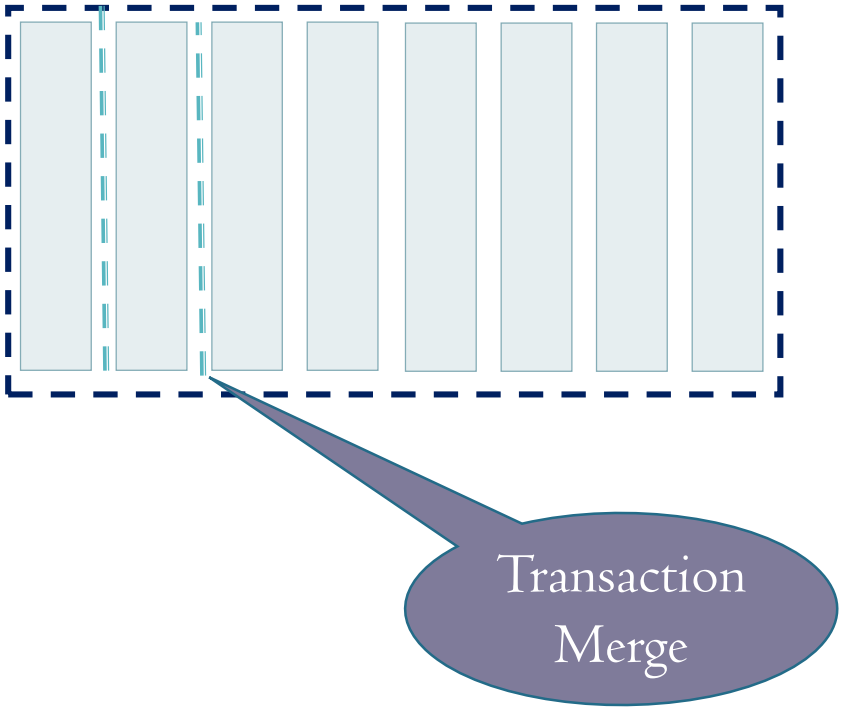
Merging Algorithm: Initial Phase

Curr Target: 8
Setpoint: 8

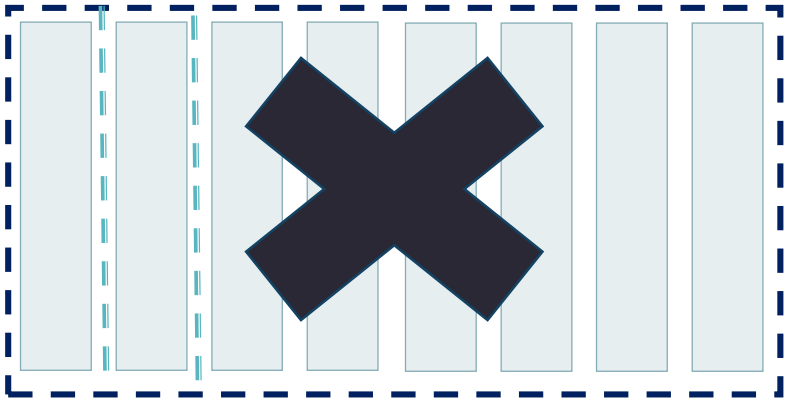


Merging Algorithm: Initial Phase

Curr Target: 8
Setpoint: 8



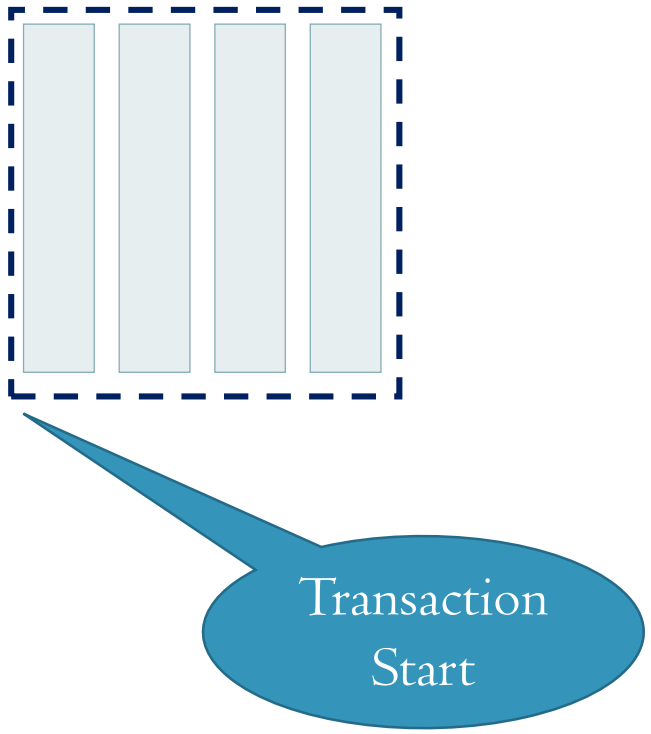
Merging Algorithm: Recover from Transient Error



Curr Target: $8/2 = 4$
Setpoint: 8

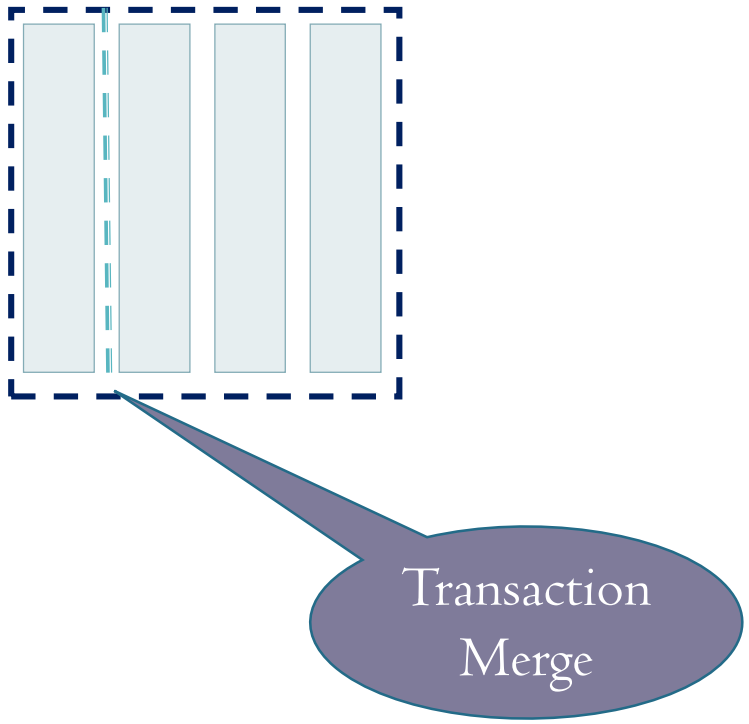
Rapid action:

Merging Algorithm: Recover from Transient Error



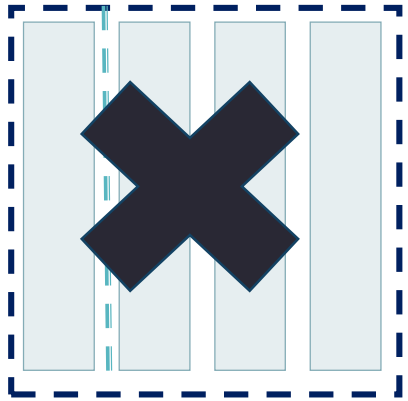
Curr Target: 4
Setpoint: 8

Merging Algorithm: Recover from Transient Error



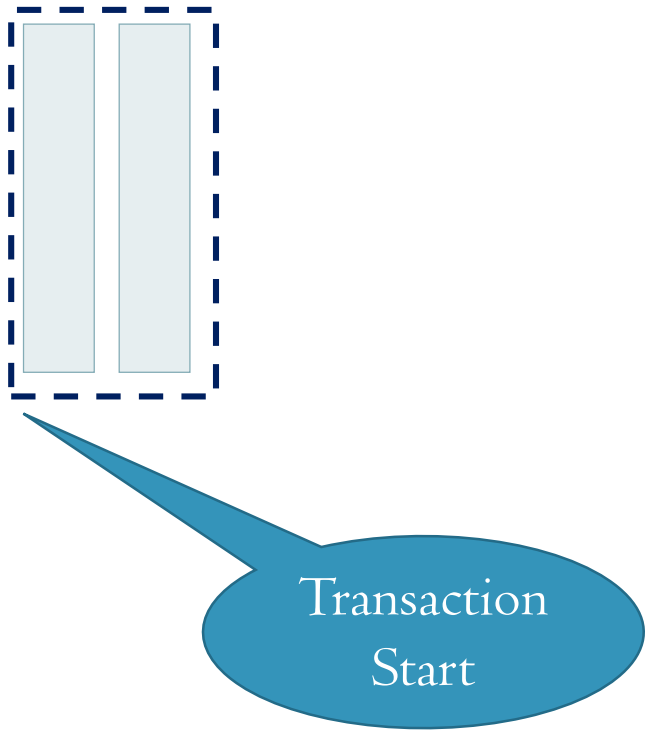
Curr Target: 4
Setpoint: 8

Merging Algorithm: Recover from Transient Error



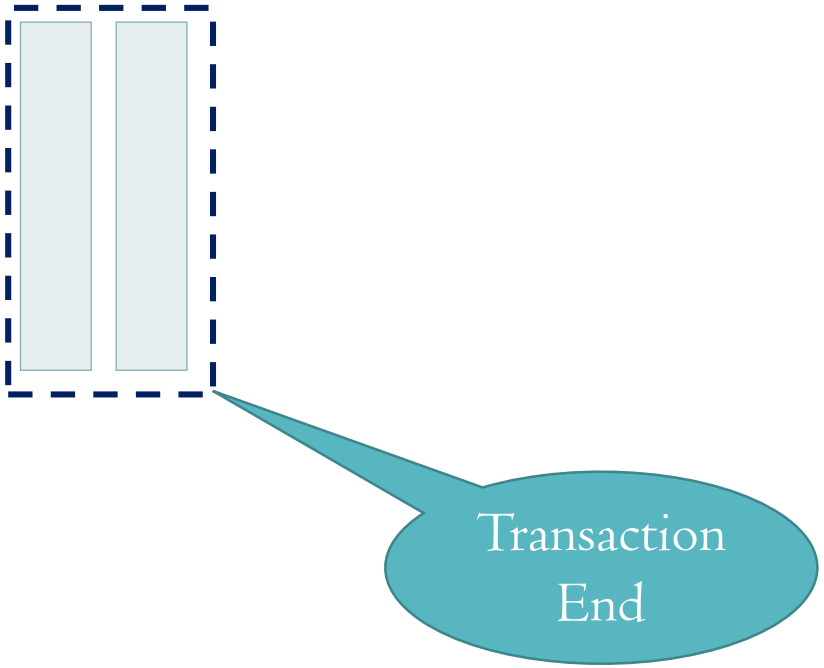
Curr Target: $4/2 = 2$
Setpoint: 8

Merging Algorithm: Recover from Transient Error



Curr Target: 2
Setpoint: 8

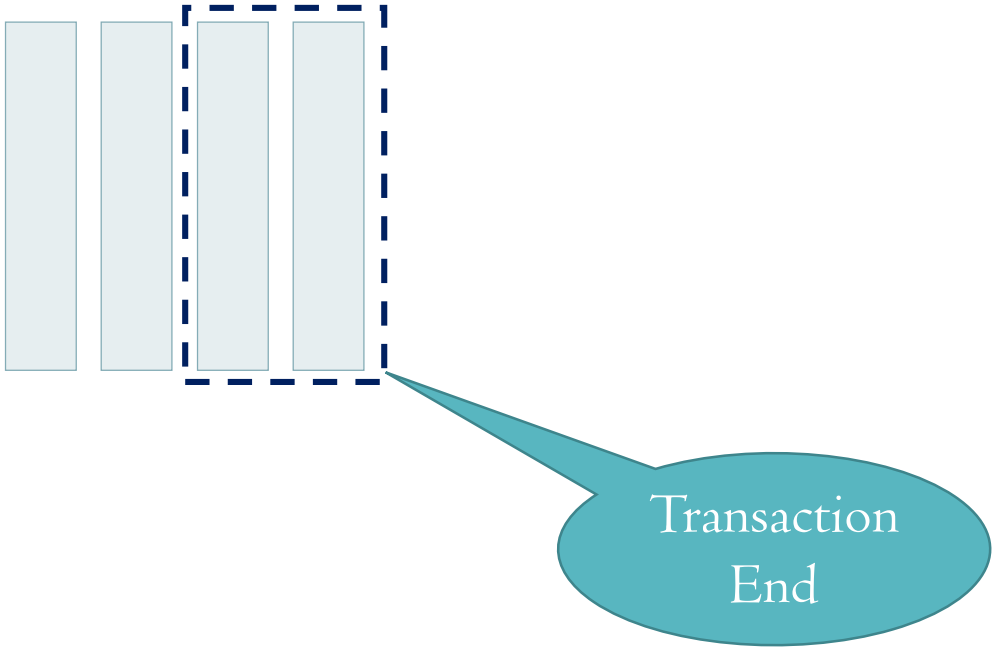
Merging Algorithm: Recover from Transient Error



Curr Target: 2
Setpoint: 8

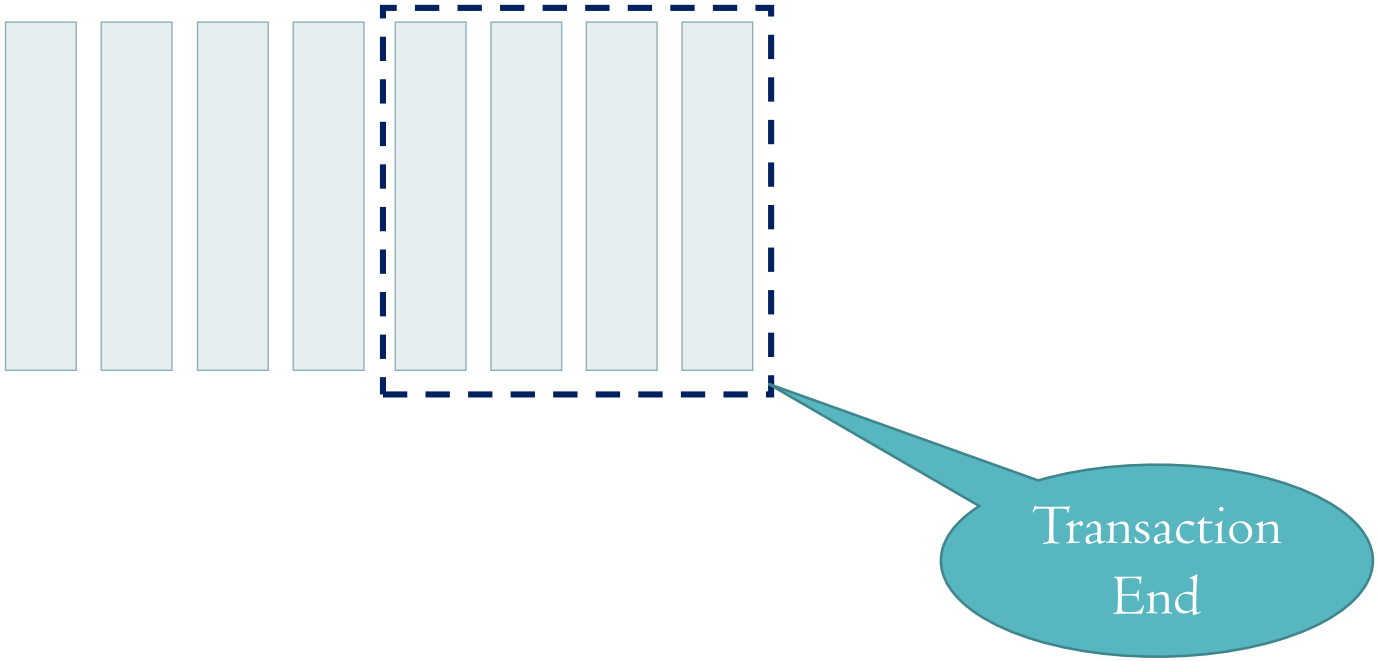
No Action:
Skeptical

Merging Algorithm: Build up to Setpoint



Curr Target: 2*2 = 4
Setpoint: 8

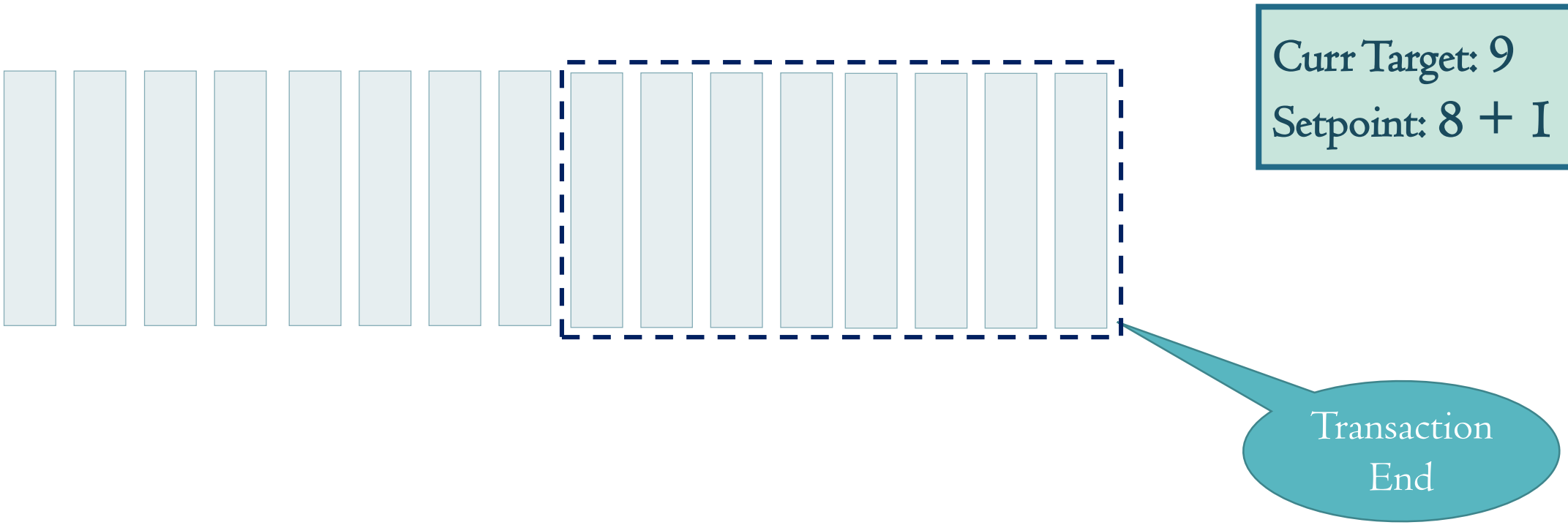
Merging Algorithm: Build up to Setpoint



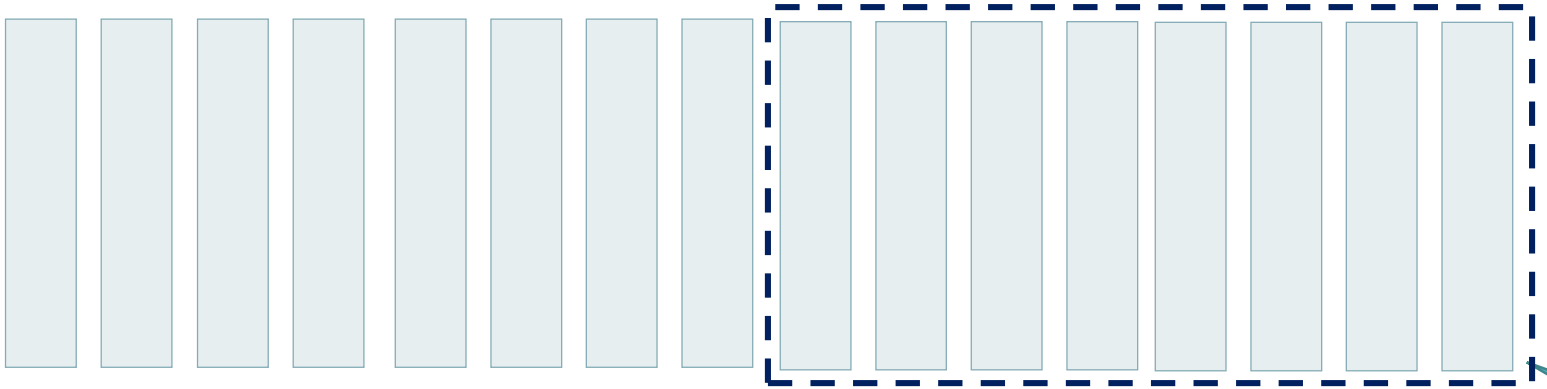
Curr Target: 4*2 = 8
Setpoint: 8

Aggressive

Merging Algorithm: Change of Program Phase



Merging Algorithm: Change of Program Phase



Curr Target: 9

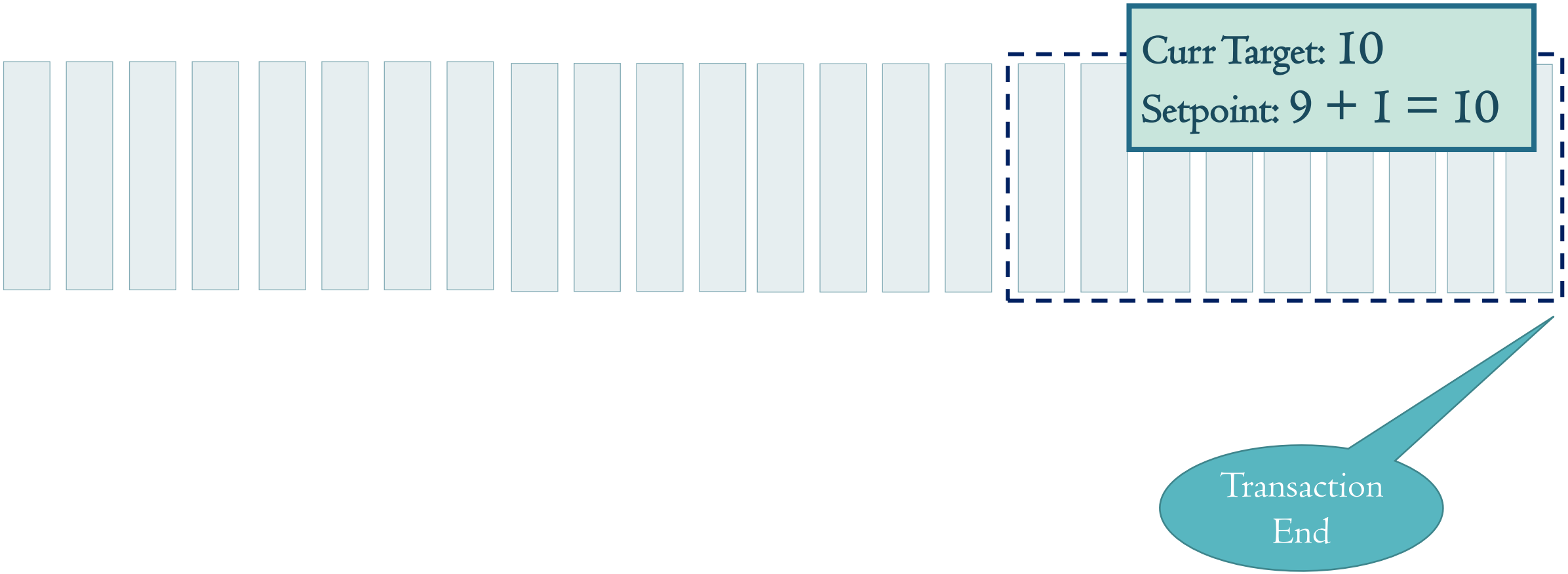
Setpoint: 8 + 1 = 9

Transaction End

Enter a low-conflict program phase

=> more aggressive merging

Merging Algorithm: Change of Program Phase



Implementation and Evaluation

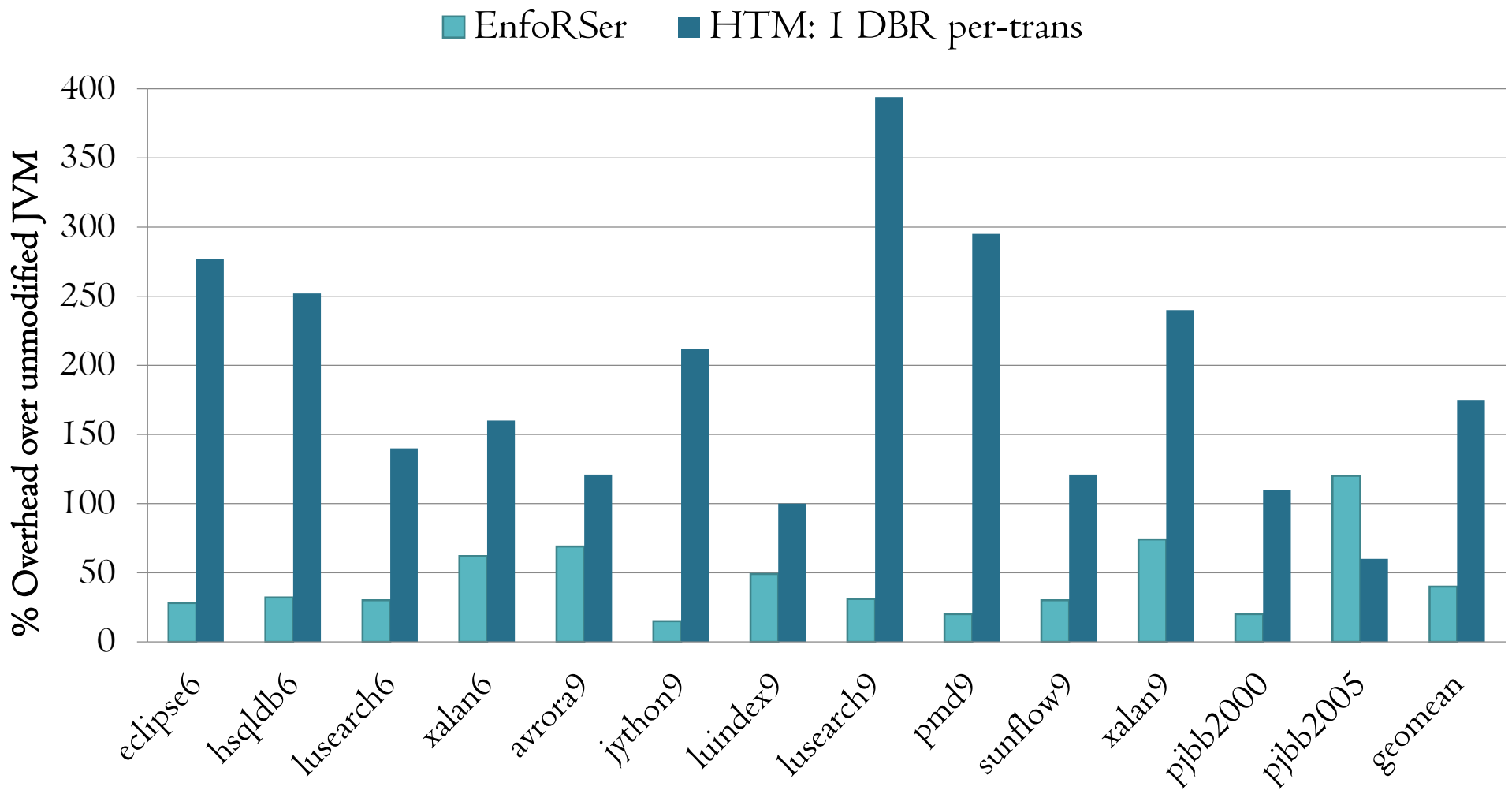
Implementation

Developed in **Jikes RVM 3.1.3**

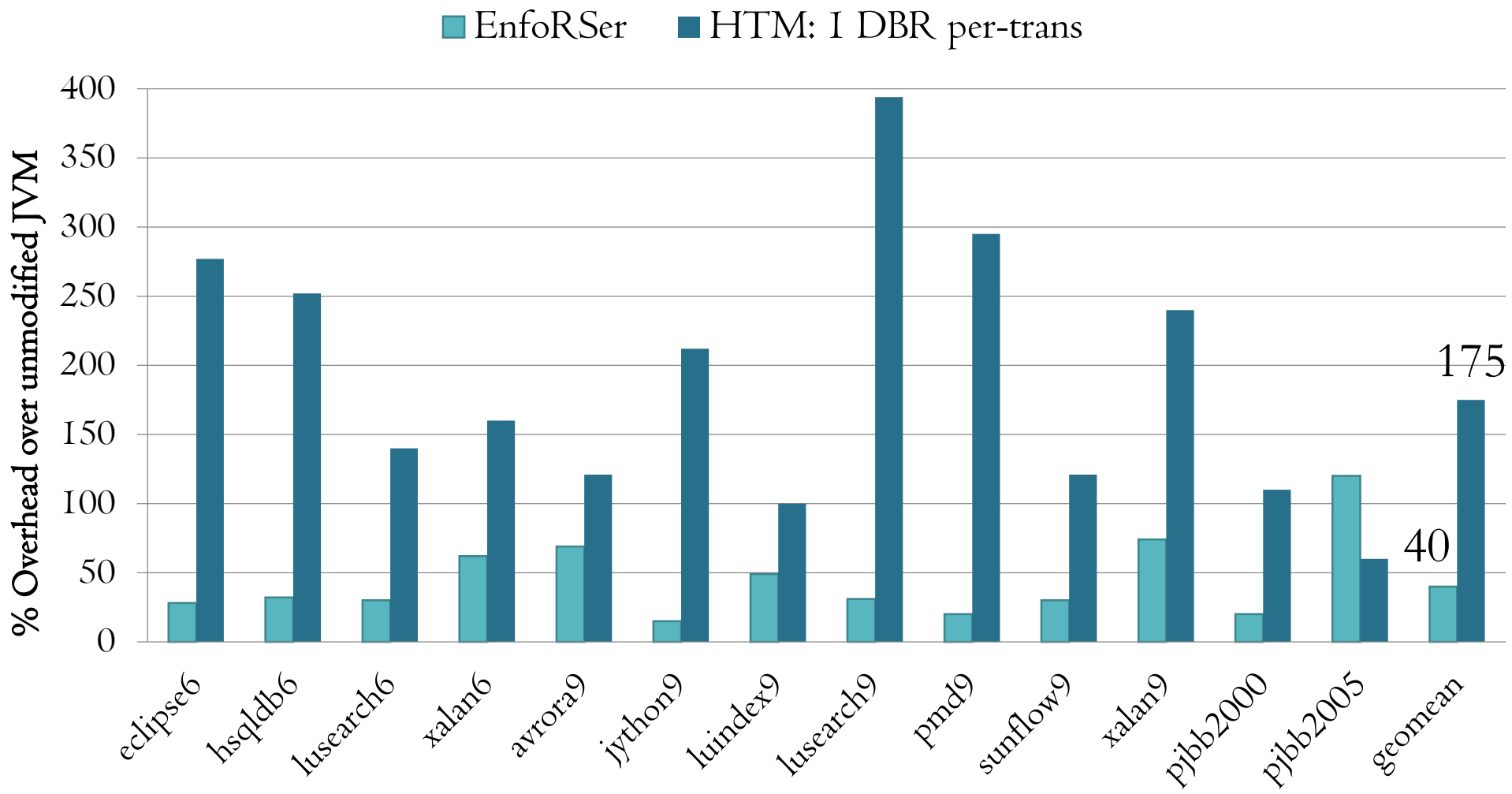
Code publicly available in **Jikes RVM Research Archive**

Run-time Performance

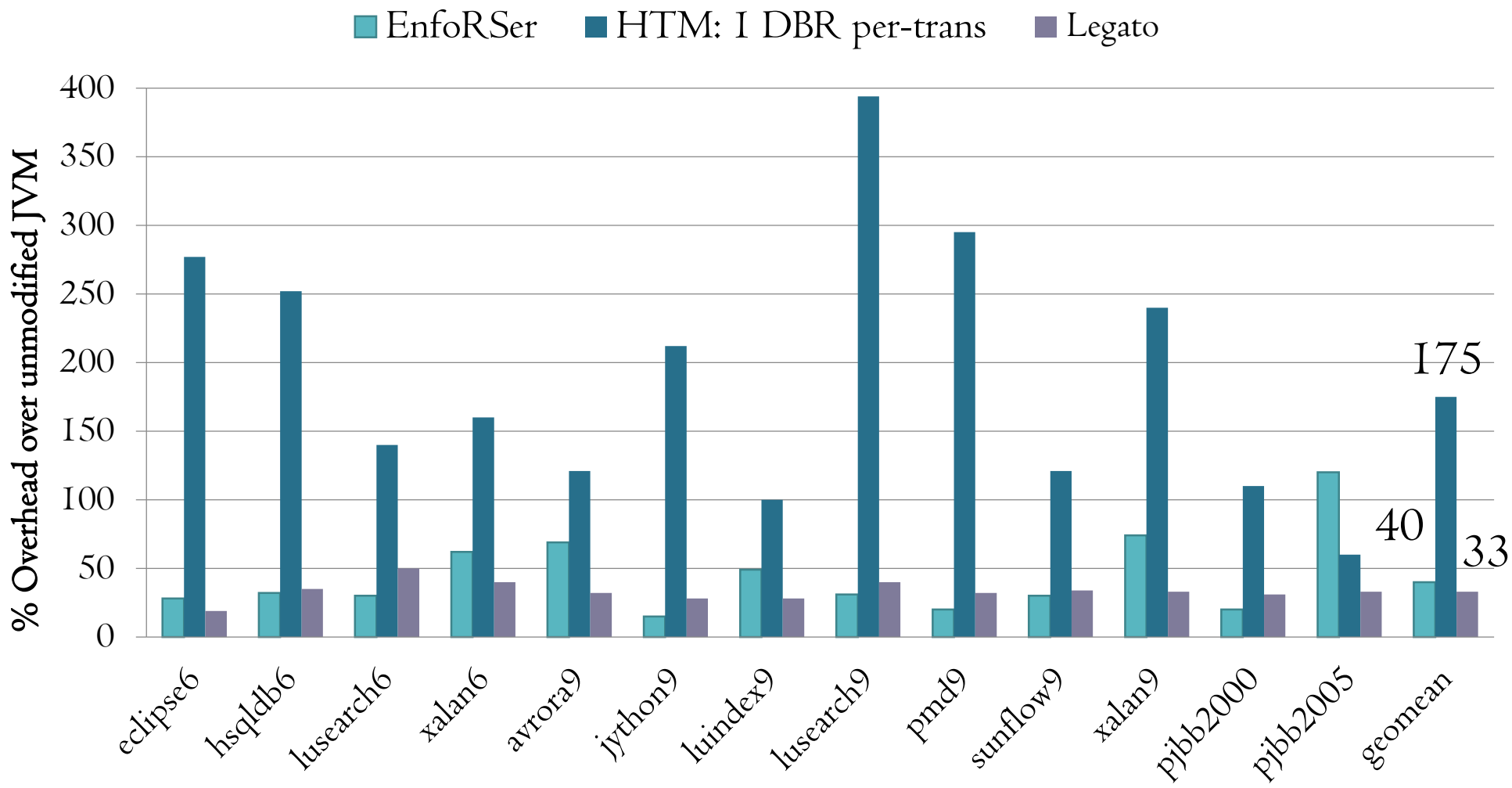
Run-time Performance



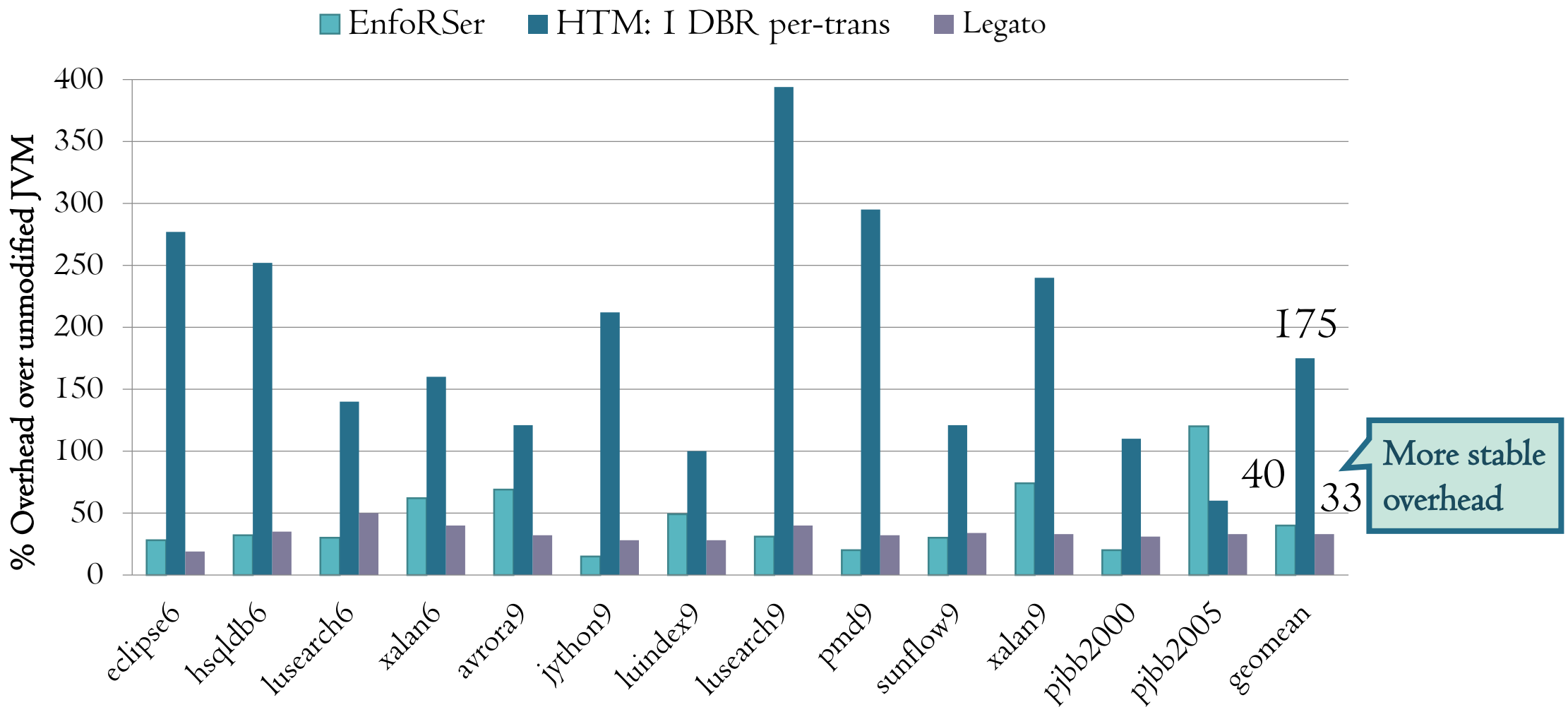
Run-time Performance



Run-time Performance

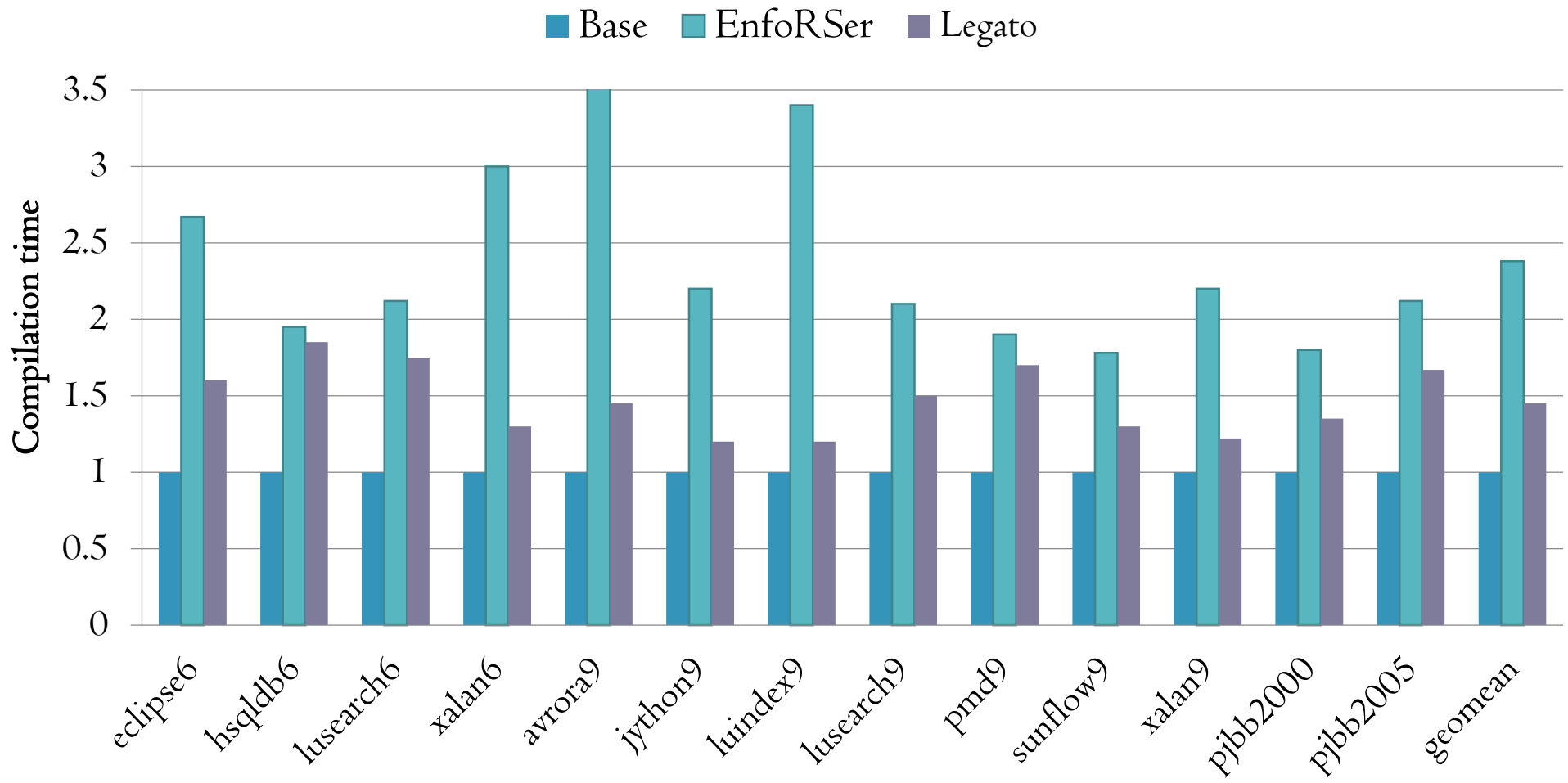


Run-time Performance

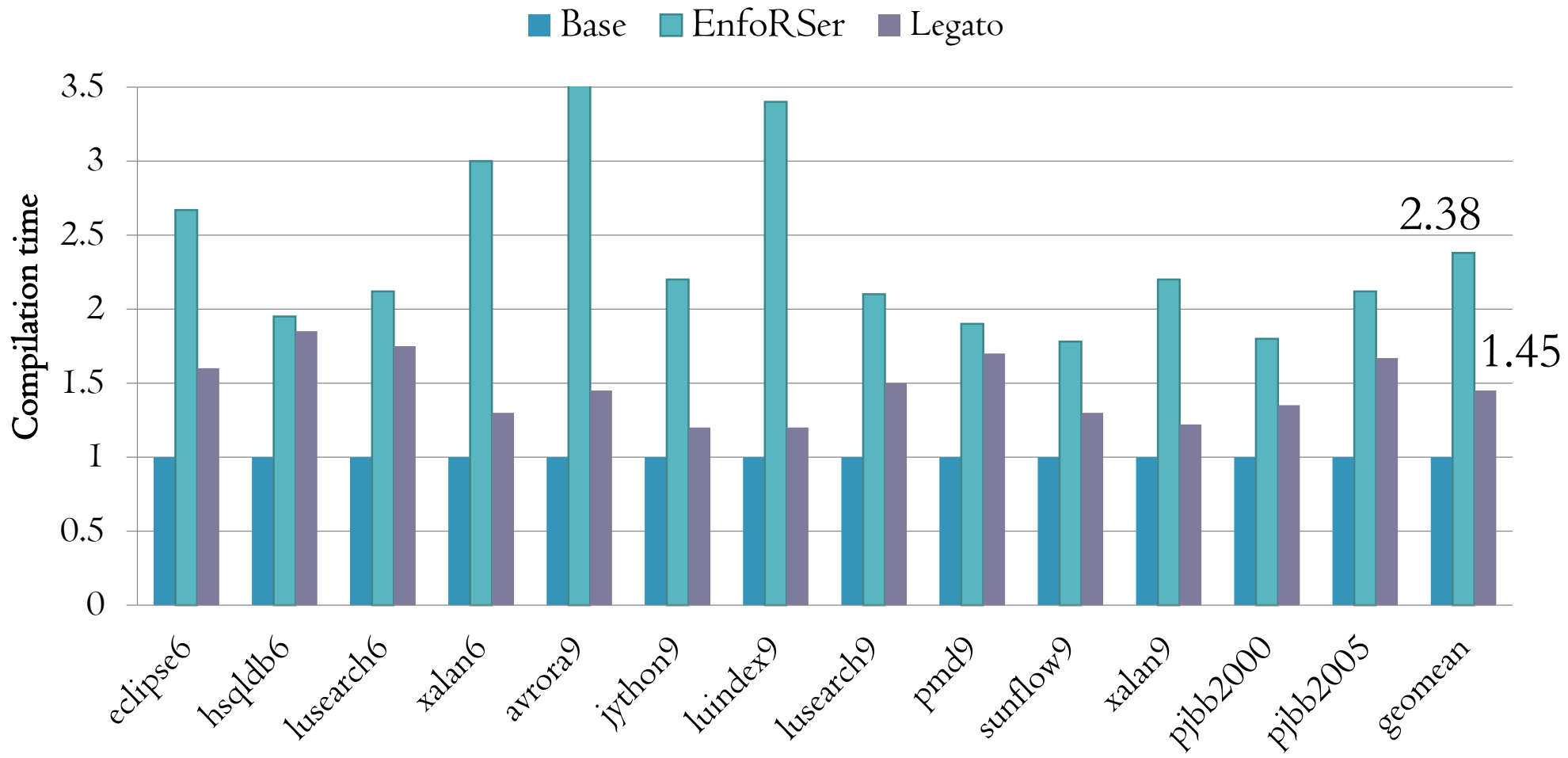


JIT Compilation Time

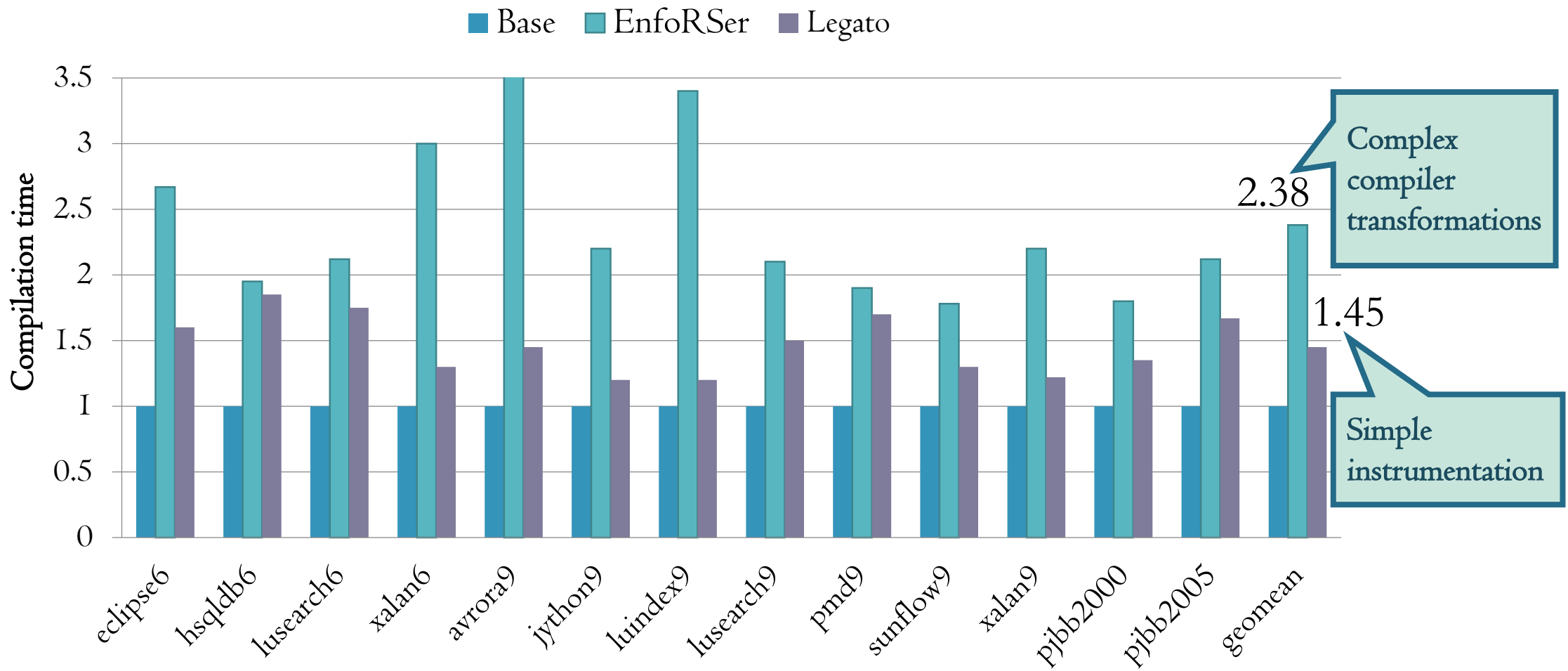
JIT Compilation Time



JIT Compilation Time



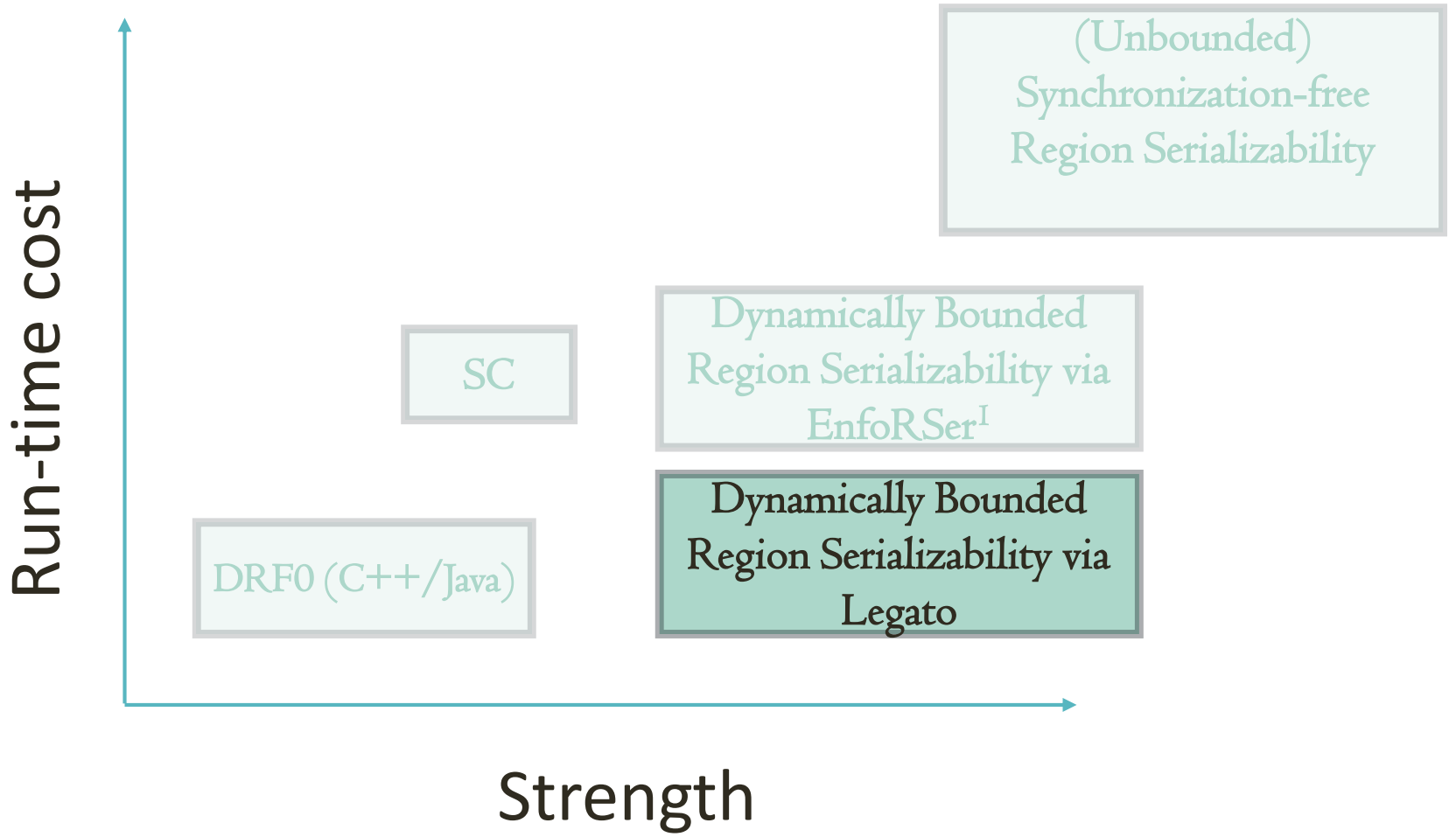
JIT Compilation Time



Related Work

- *Checks conflicts in bounded region*
  DRFx, Marino et al., PLDI 2010
- *Checks conflicts in synchronization-free regions*
  Conflict Exceptions, Lucia et al., ISCA 2010
- *Enforces atomicity of bounded regions*
  BulkCompiler, Ahn et al., MICRO 2009
  Atom-Aid, Lucia et al., ISCA 2008
- *Reducing dynamic misspeculations*
  BlockChop, Mars and Kumar, ISCA 2012

Memory Models: Run-time cost vs Strength



1. Sengupta et al. Hybrid Static-Dynamic Analysis for Statically Bounded Region Serializability. ASPLOS, 2015.