

A lock-free cache invalidation protocol for the ATOM system

Hugo Rito, João Cachopo

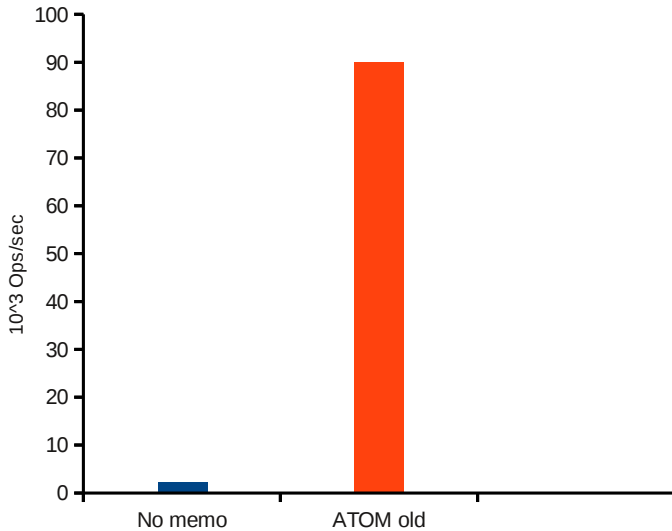
INESC-ID
Instituto Superior Técnico

October 23, 2011

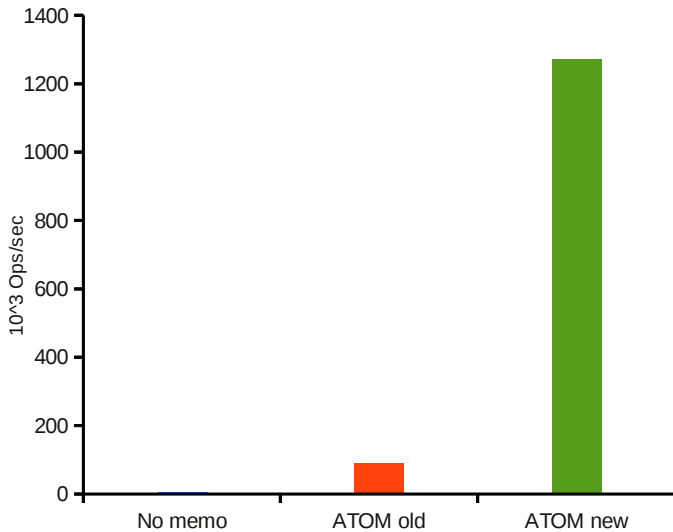
Overview

Memoization

Overview



Overview



The ATOM system

A

T

O

M

The ATOM system

A**T****O****M**emoization

```
public class Foo {  
    public int shared;  
  
    int foo(int arg) {  
        return shared * arg;  
    }  
}
```

The ATOM system

Automatic

T

O

Memoization

```
public class Foo {  
    public int shared;  
  
    @Memo  
    int foo(int arg) {  
        return shared * arg;  
    }  
}
```

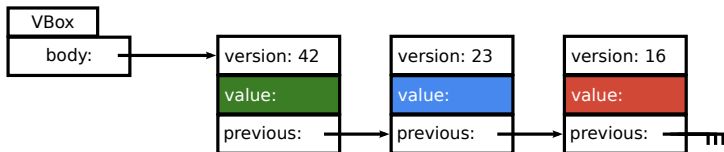
The ATOM system

Automatic **T**ransaction-**O**riented **M**emoization

```
public class Foo {  
    public int shared;  
    Transaction.begin()  
    @Memo  
    int foo(int arg) {  
        return shared * arg;  
    }  
    Transaction.commit()  
}
```

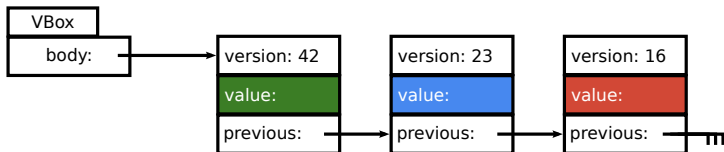
The Java Versioned STM

Versioned box



The Java Versioned STM

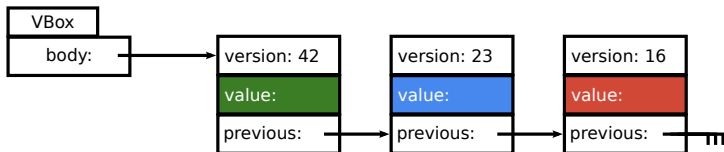
- Versioned box



- Read-set

The Java Versioned STM

- Versioned box

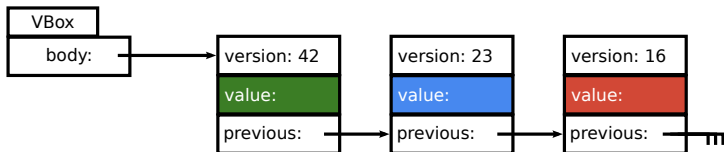


- Read-set

- Write-set

The Java Versioned STM

- Versioned box



- Read-set

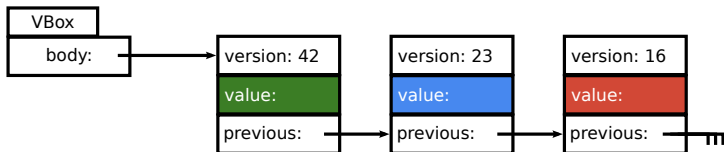


Relevant state

- Write-set

The Java Versioned STM

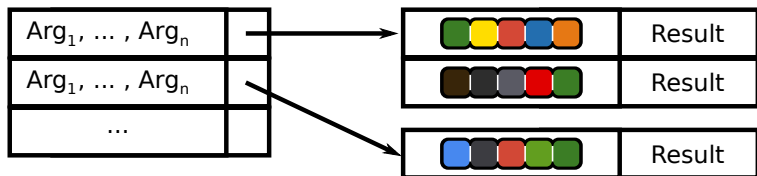
- Versioned box



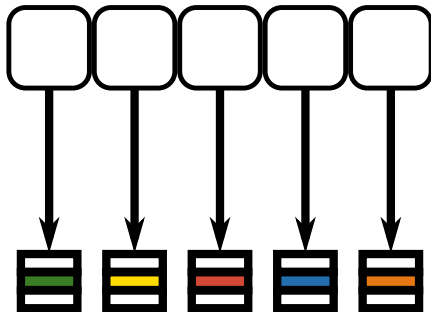
- Read-set → Relevant state

- Write-set → Safe to memoize

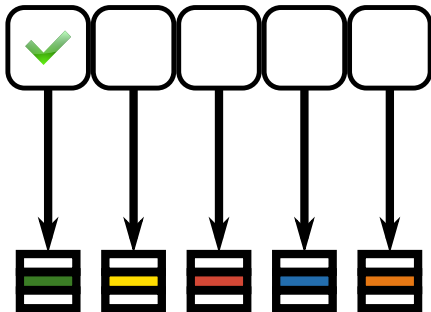
ATOM's memo cache



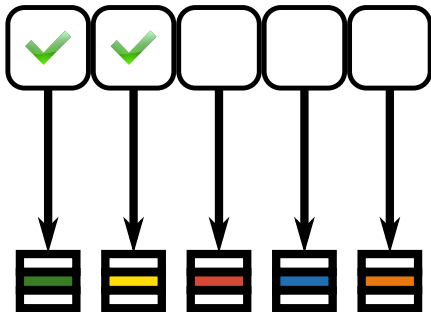
Relevant state validation



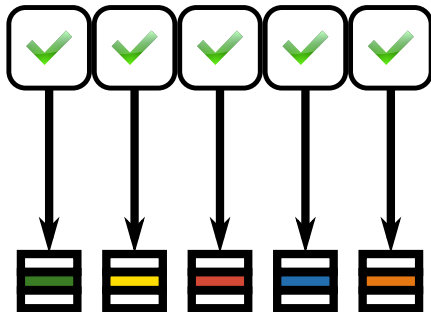
Relevant state validation



Relevant state validation



Relevant state validation



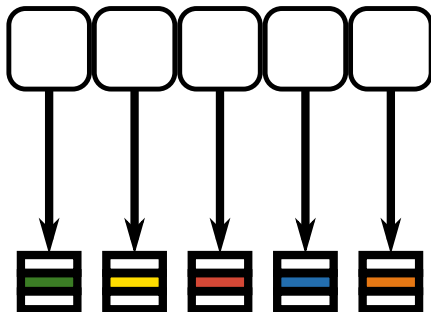
Improving the ATOM

Large read-sets

Improving the ATOM

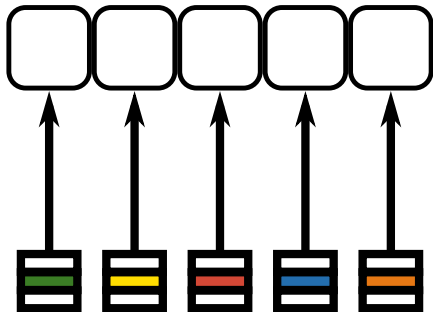
State rarely changes

Relevant state validation



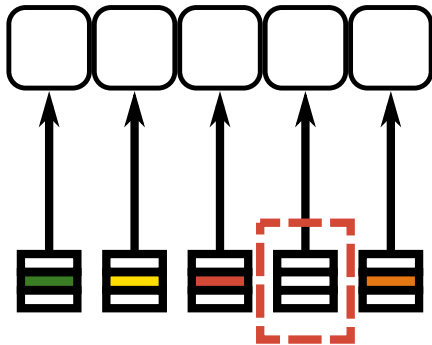
Relevant state **invalidation**

State: 



Relevant state **invalidation**

State: INVALID




Design drivers

Highly concurrent

Design drivers

Write-dominated

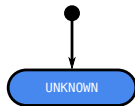
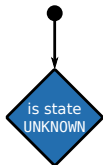
The InvalidationRelevantState

- Create:
 - Lazy initialization
 -  state

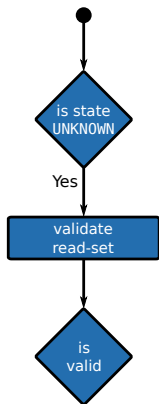
The InvalidationRelevantState

- Create:
 - Lazy initialization
 -  state
- Search:
 - Slow path: helps register
 - Fast path: reads state

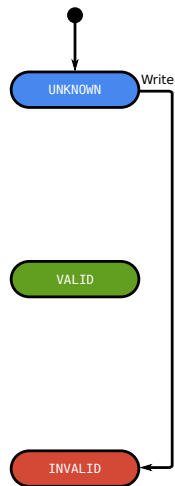
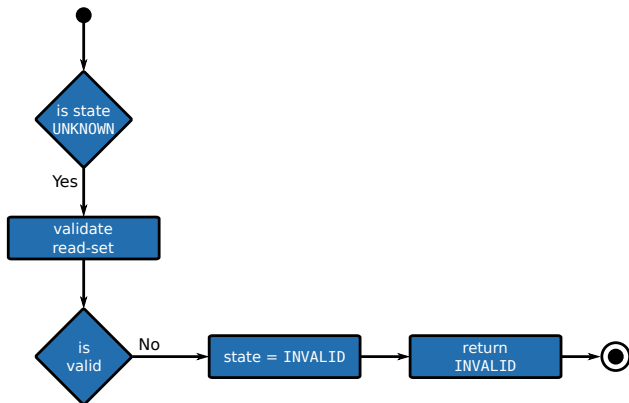
Cache search



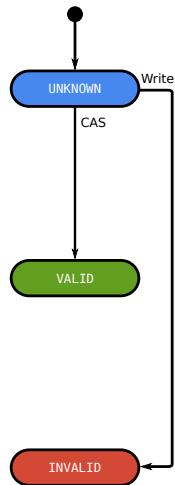
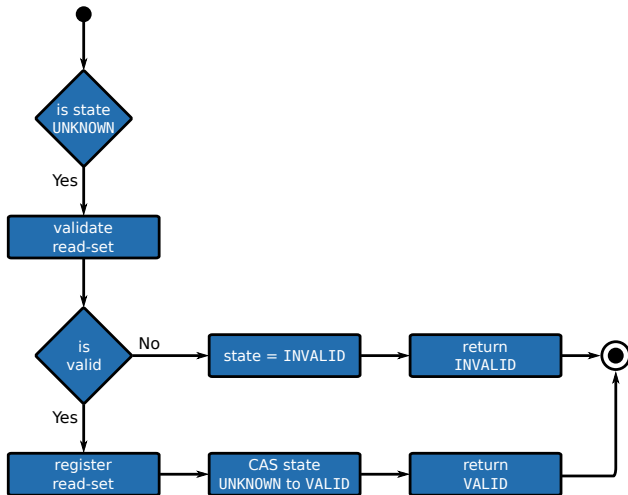
Cache search



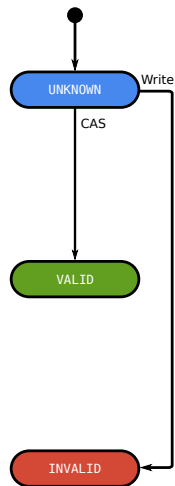
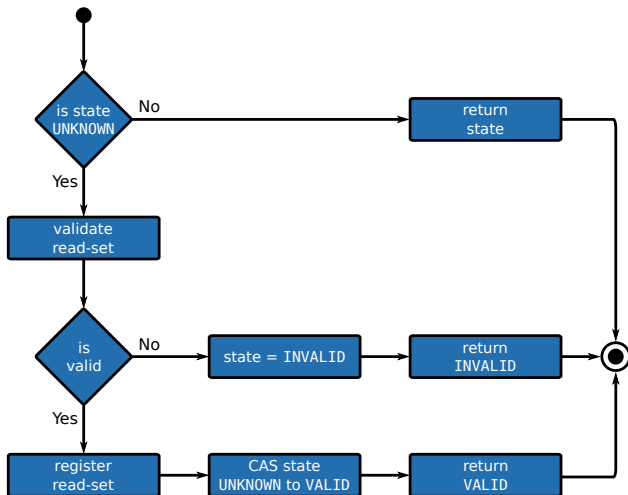
Cache search



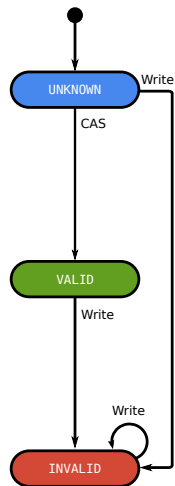
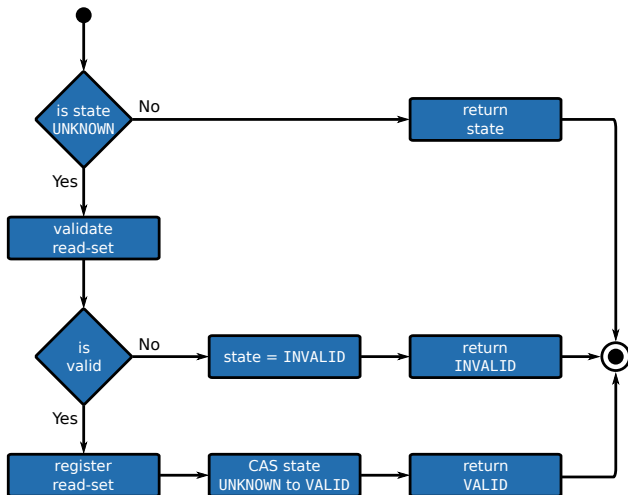
Cache search



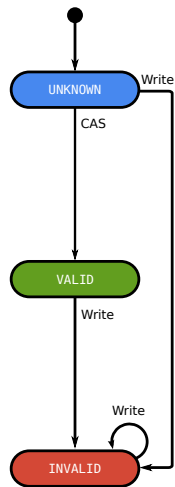
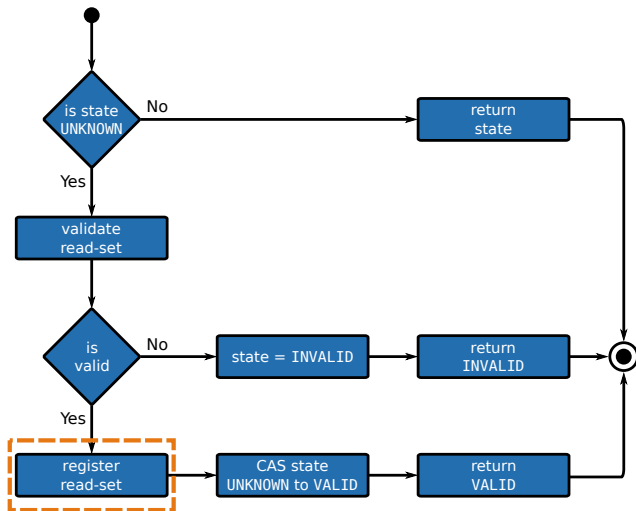
Cache search



Cache search



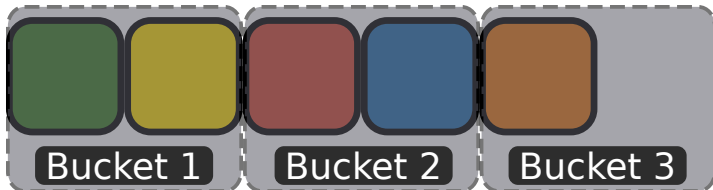
Cache search



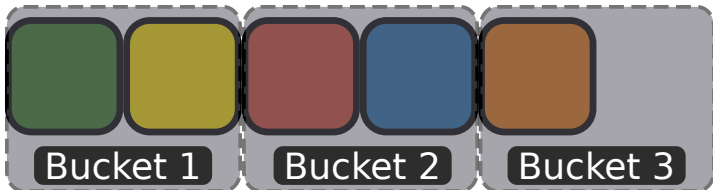
Helping register read-set



Helping register read-set



Helping register read-set



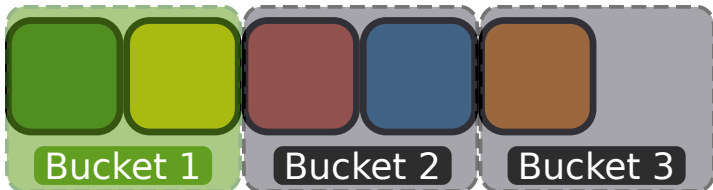
Thread #1

Bucket 1

Bucket 3

Bucket 2

Helping register read-set



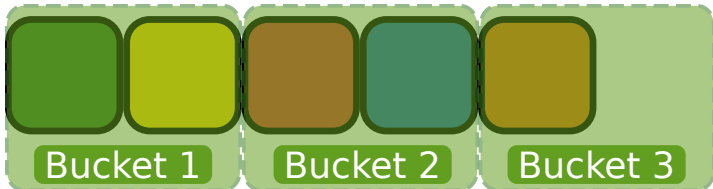
Thread #1

Bucket 1
Bucket 3
Bucket 2

Thread #2

Bucket 2
Bucket 3
Bucket 1

Helping register read-set



Thread #1

Bucket 1

Bucket 3

Bucket 2

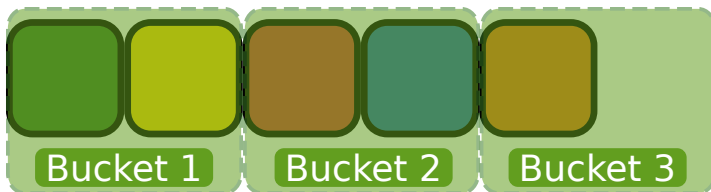
Thread #2

Bucket 2

Bucket 3

Bucket 1

Helping register read-set



Thread #1

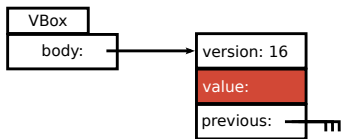
Bucket 1
Bucket 3
Bucket 2



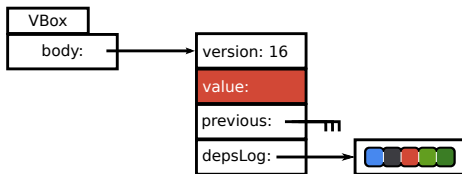
Thread #2

Bucket 2
Bucket 3
Bucket 1

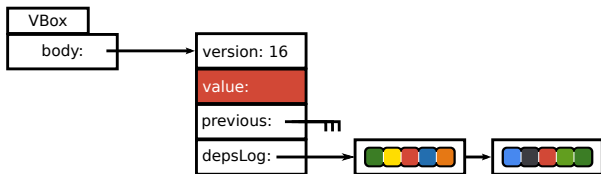
The MemoVBoxBody



The MemoVBoxBody



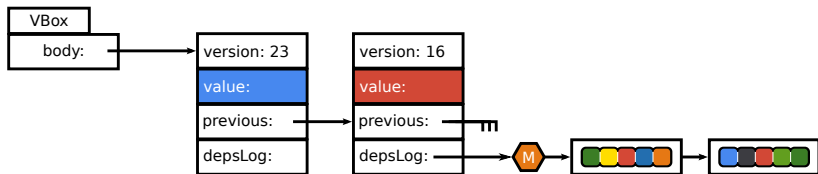
The MemoVBoxBody



Register:

- CAS depsLog head to new dependent

The MemoVBoxBody



Register:

- CAS depsLog head to new dependent

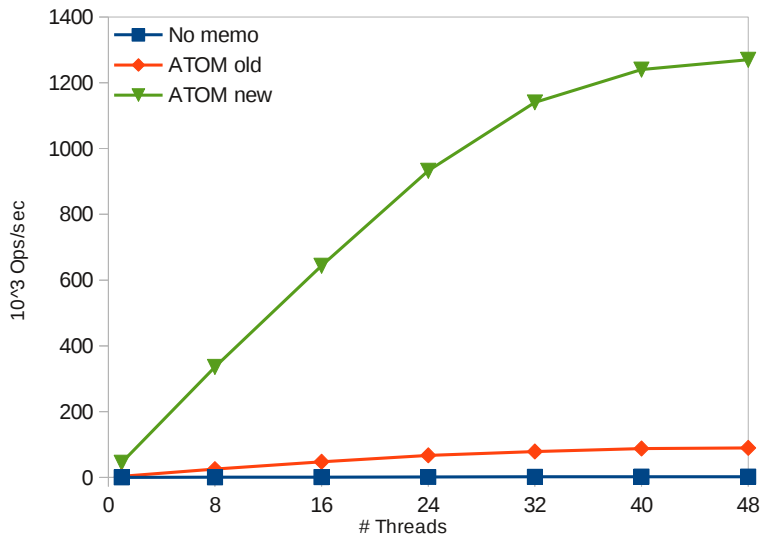
Invalidate:

- New body invalidates old body (previous)
- CAS depsLog head to INVALID_MARK
- Helping bucket-based

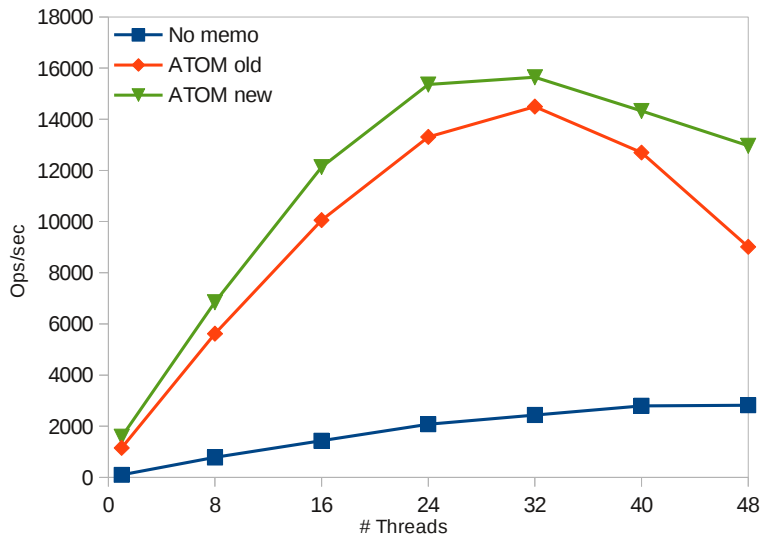
Experimental setup

- STMBench7 benchmark
- 4xAMD Opteron 6168 (48 cores)
- 1 → 48 threads
- Workloads
 - Read-only
 - Read-dominated
 - Read-write

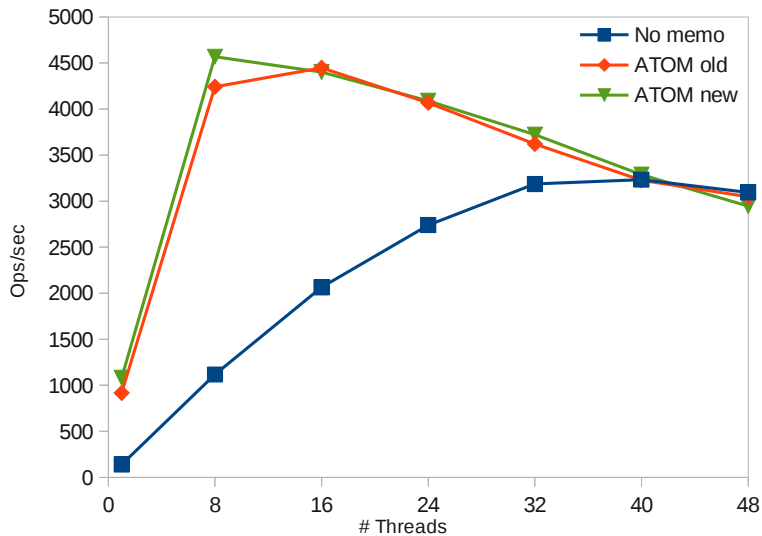
Read-only (0% writes)



Read-dominated (10% writes)



Read-write (40% writes)



Conclusions

Invalidation memo strategy:

- State rarely changes → Penalise writes
- Large read-sets → Single read
- Highly concurrent → Parallel registration/invalidation
- Write-dominated → Lazy initialization

Thank you,

Q U E S T I O N S