departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

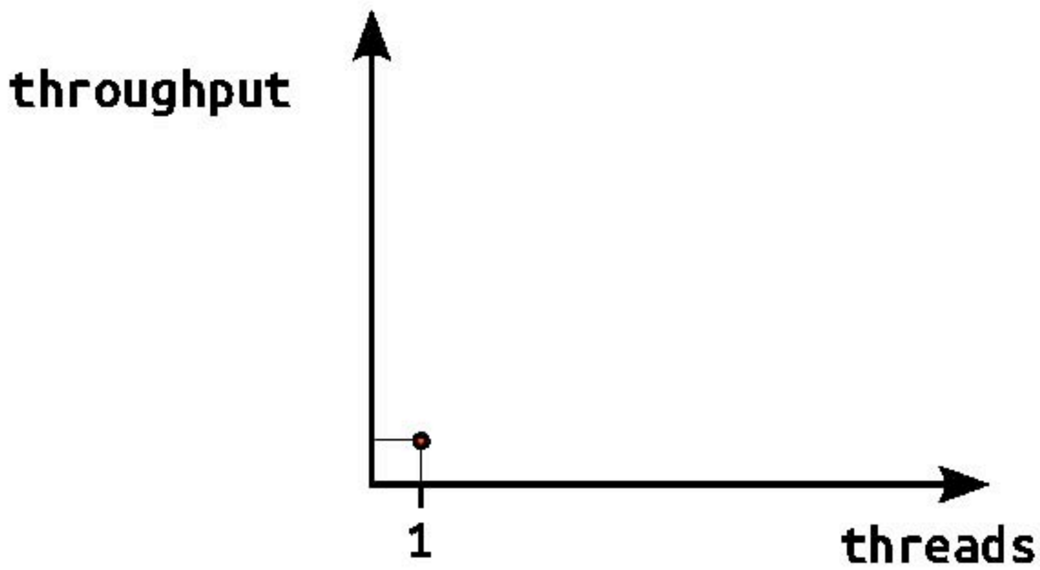
Concurrency and Parallelism
(Concorrência e Paralelismo – CP 11158)

Lecture 9

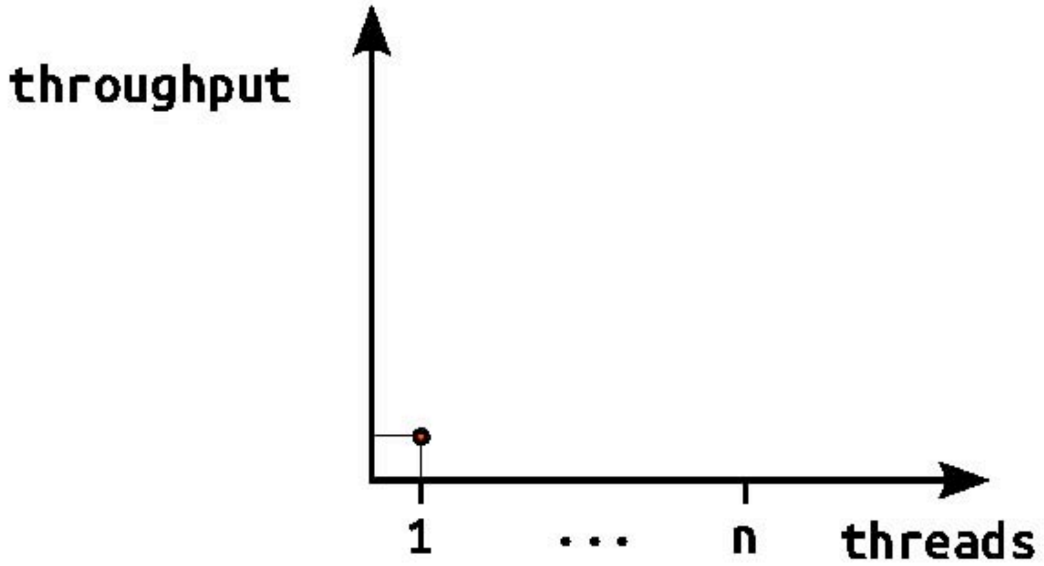
— Implementing Transactional Memory —

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Motivation

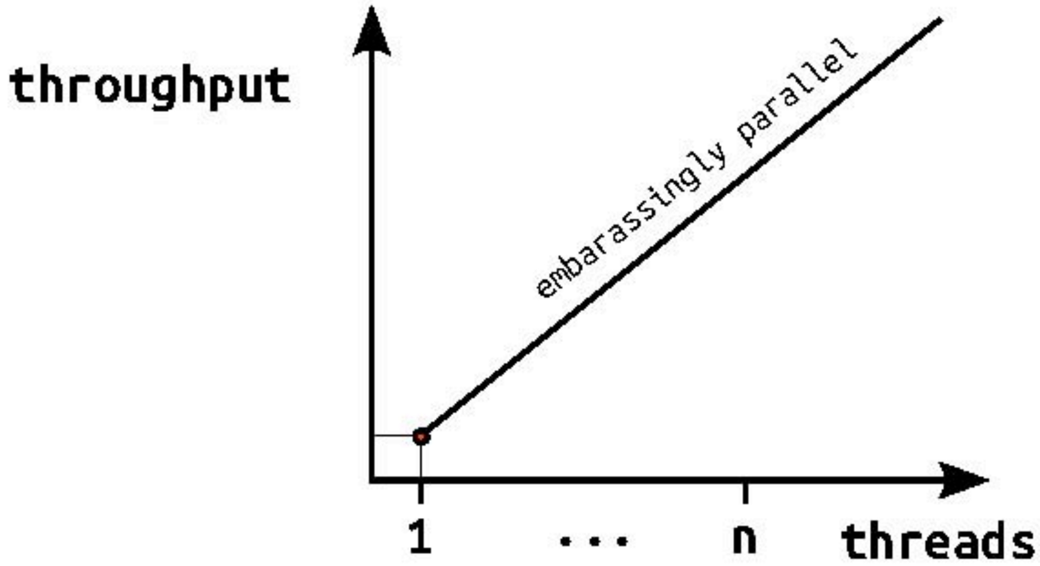


Motivation

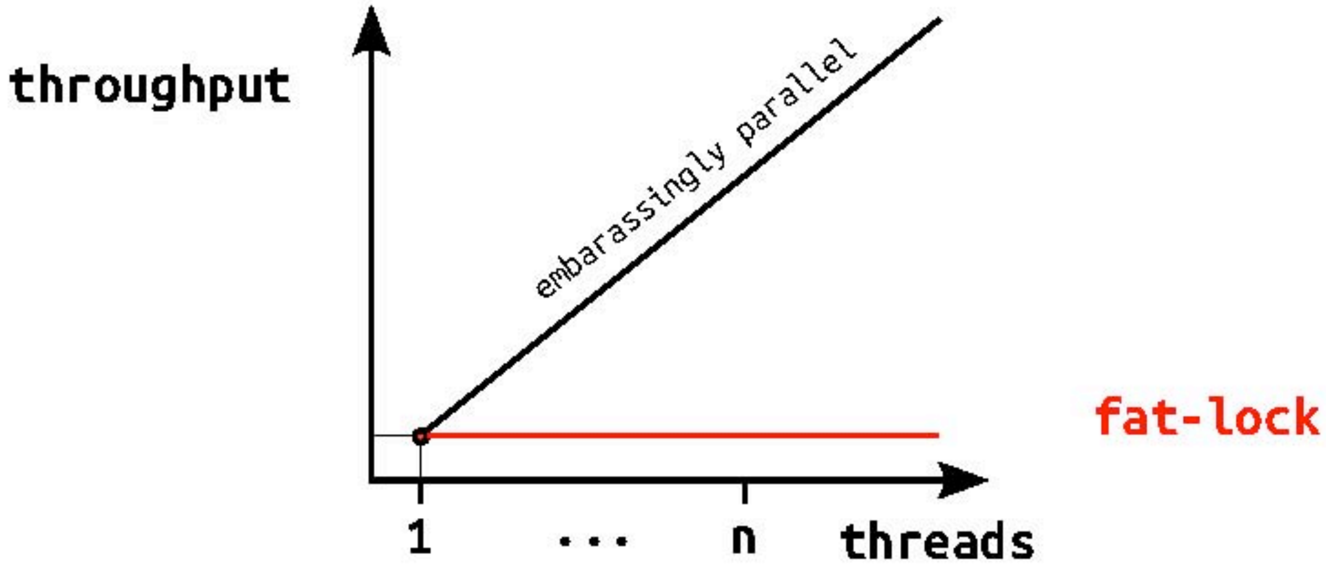


departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Motivation

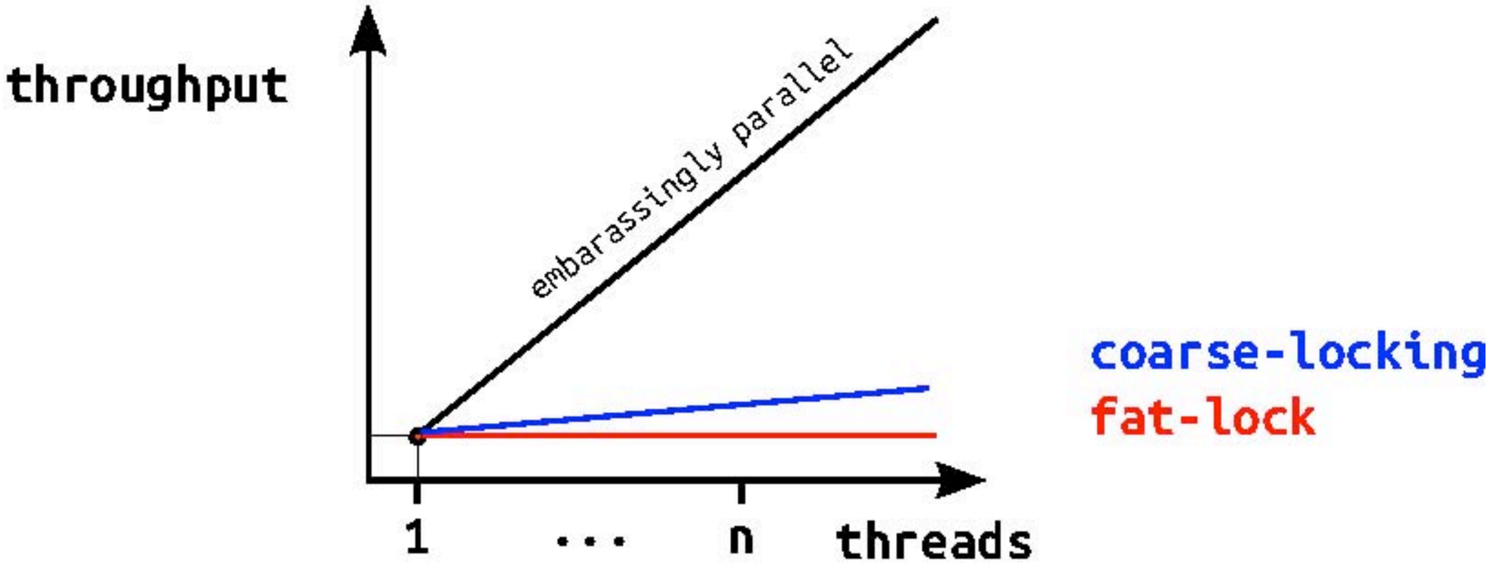


Motivation



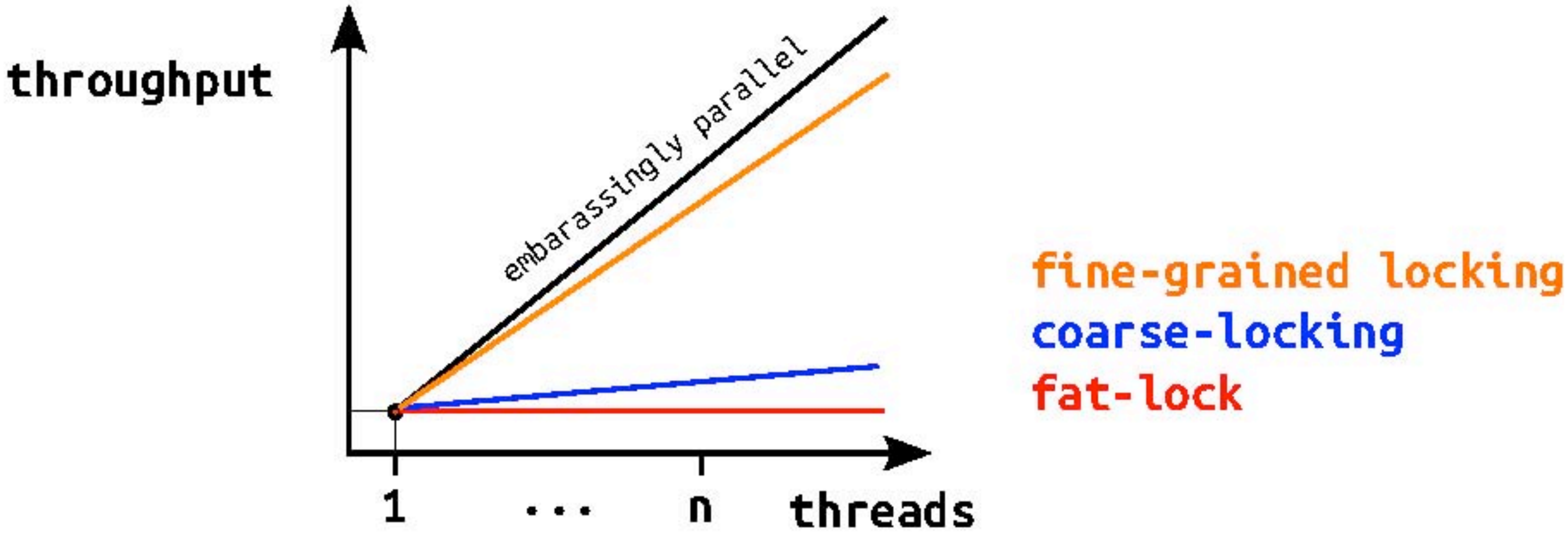
departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Motivation

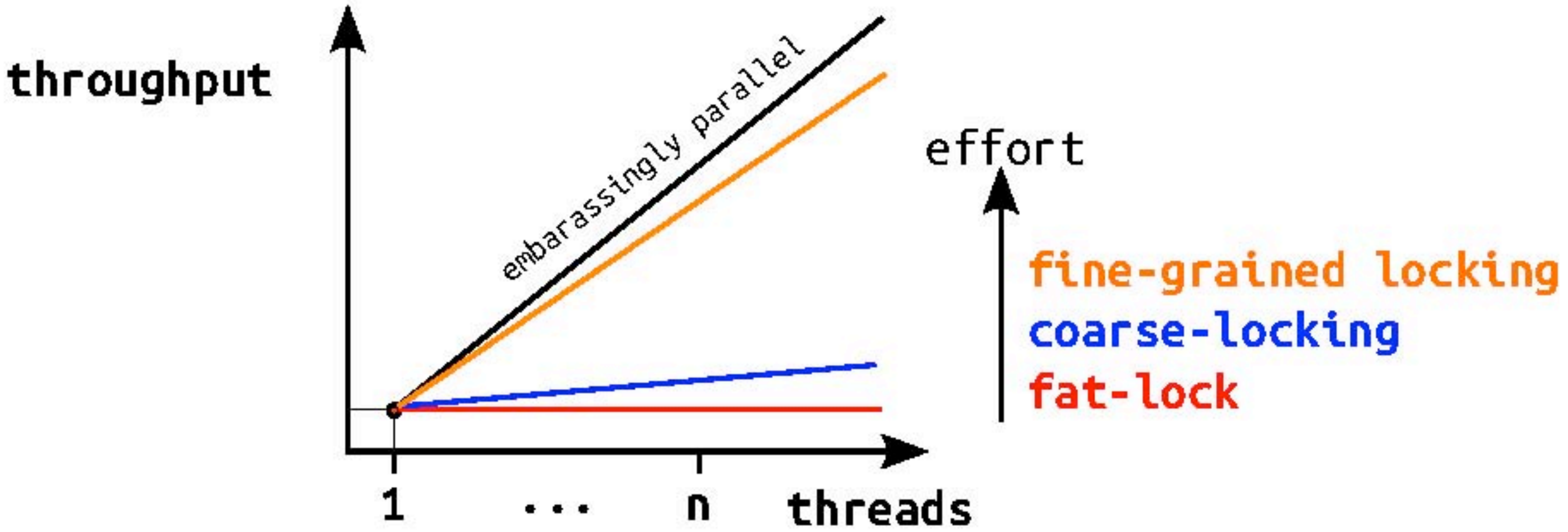


departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

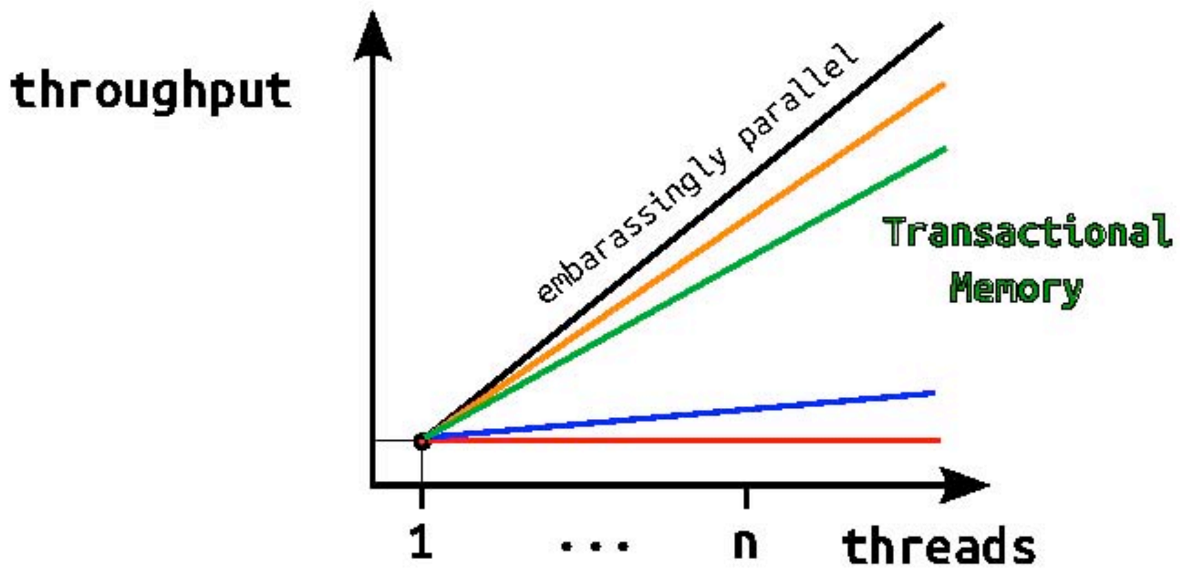
Motivation



Motivation


throughput
embarassingly parallel
effort
fine-grained locking
coarse-locking
fat-lock
1
...
n
threads

Motivation


throughput
embarassingly parallel
Transactional
Memory
1
...
n
threads

Privatization

```
// 'x' and 'y' are transactional variables

int lowBound = x;
int upBound = y;

while ( lowBound < upBound ) {
    // some computation
    lowBound ++;
}
```

Opacity

```
// atomic variables
X=0;  y=0;

while (true) {
    atomic {
        x++;
        y++;
    }
}
```

```
while (true) {
    if (x != y){
        // this should never happen
        exit(1);
    }
}
```

How to avoid those hazards

- Sandboxing
  - Mask signal handlers
  - Periodic validation to avoid infinite loops | on loop back-edge
  - Very ad-hoc, incomplete, affects performance
  - Always ensure consistent reads (this paper's contribution)

TL2

- Commit-time locking
- Write buering
- Timestamped transactional locations
- Global clock (updated by write txs on their commit)

TL2 – Read-only Tx

- Identified statically
- On start:
  - tx.rv := globalClock
- On read(var):
  - lock := lockArray[var]
  - value := var
  - repeatLock := lockArray[var]
- Abort if:
  - isLocked(lock) $\vee$ timestamp(lock) > tx.rv $\vee$ lock ≠ repeatLock
- No read-set maintenance
- If it reaches commit operation, proceeds trivially

TL2 – Read-Write Tx

- On start:
  - Tx.rv := globalClock
- On write(var, value):
  - writeSet := writeSet U (var, value)
  - bloomFilter.update(var)
- On read(var):
  - Check for read-after-write using bloomFilter
  - Or proceed as Read-Only tx,
    plus readSet := readSet U (lockArray[var])

TL2 – Commit

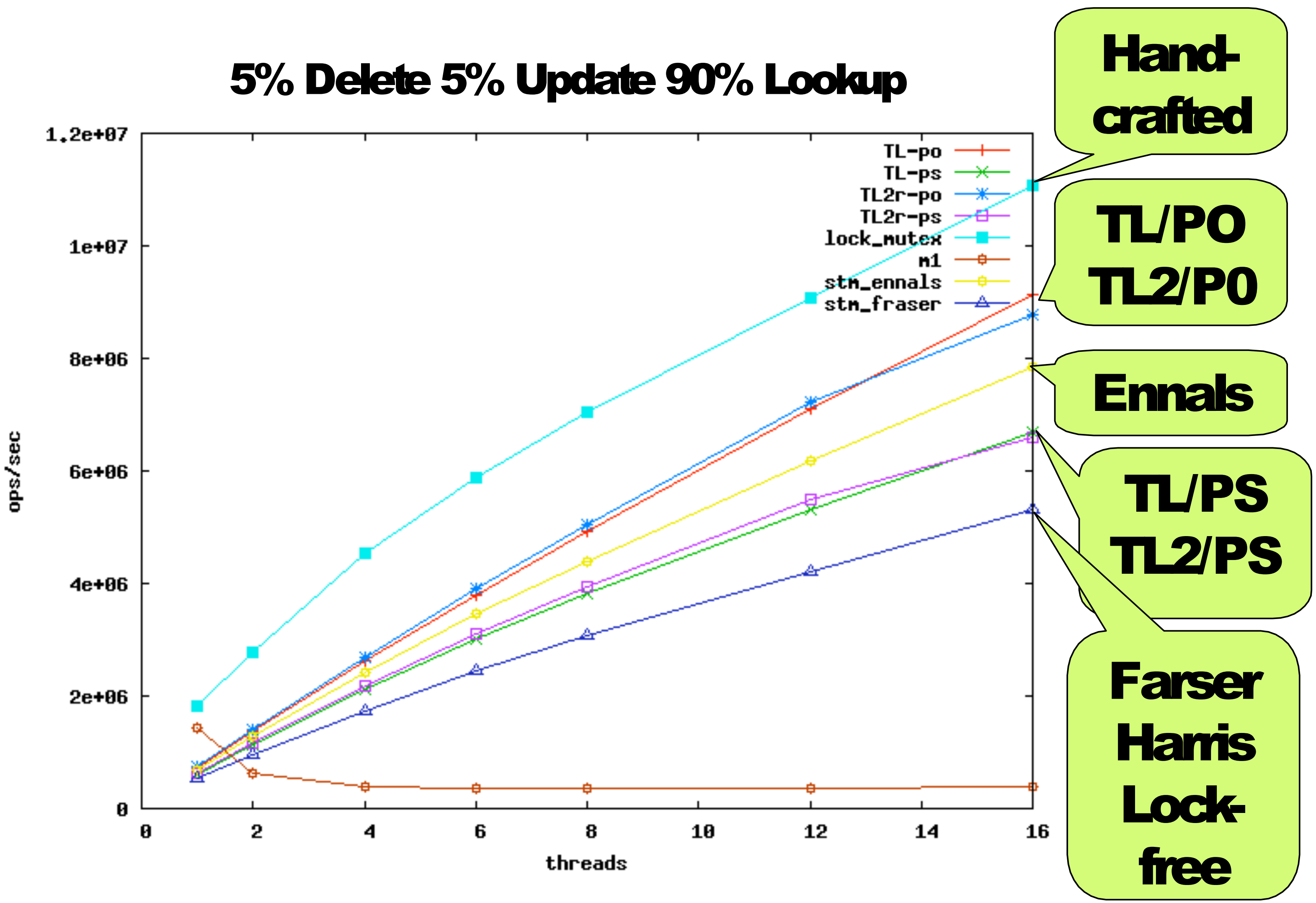
- Acquire locks of variables written
  - Use bounded spinning locks to avoid deadlocks
  - False aborts due to timeouts
  - Cheaper than establishing a locking order
- Increment-and-fetch global clock
- Validate all reads
  - As previously described for read operation
  - Skip this step if no concurrent commit since *tx.rv*
- For all written variables:
  - Write buffered value in variable
  - Release lock by writing bit to 0 and new timestamp

Lock granularity

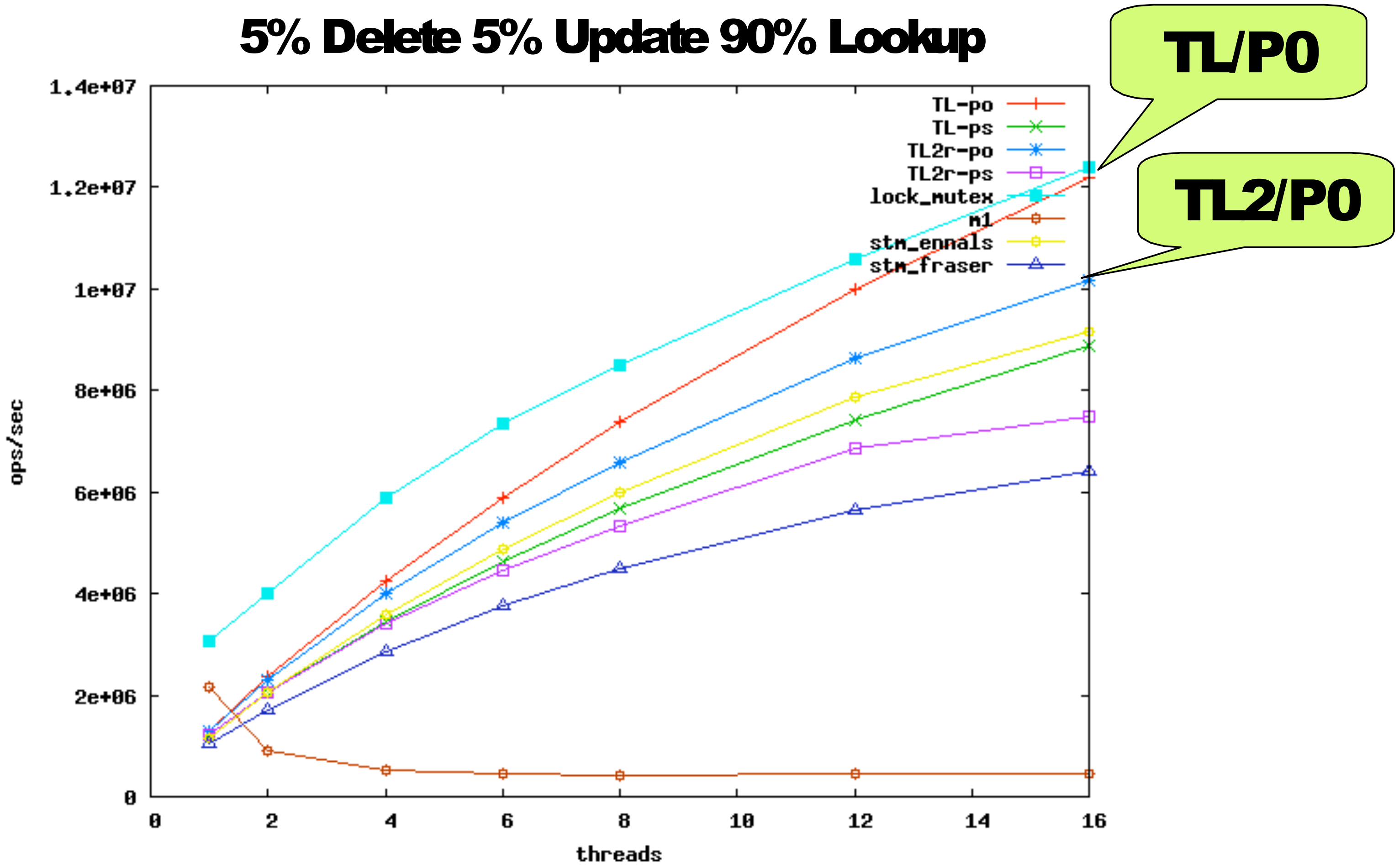
- A lot of flexibility:
  - Per address — too much space overhead
  - Per stripe (cache line for instance) — false conflicts
  - Per object (header) — precludes transparent memory management

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Uncontended Large RB-Tree

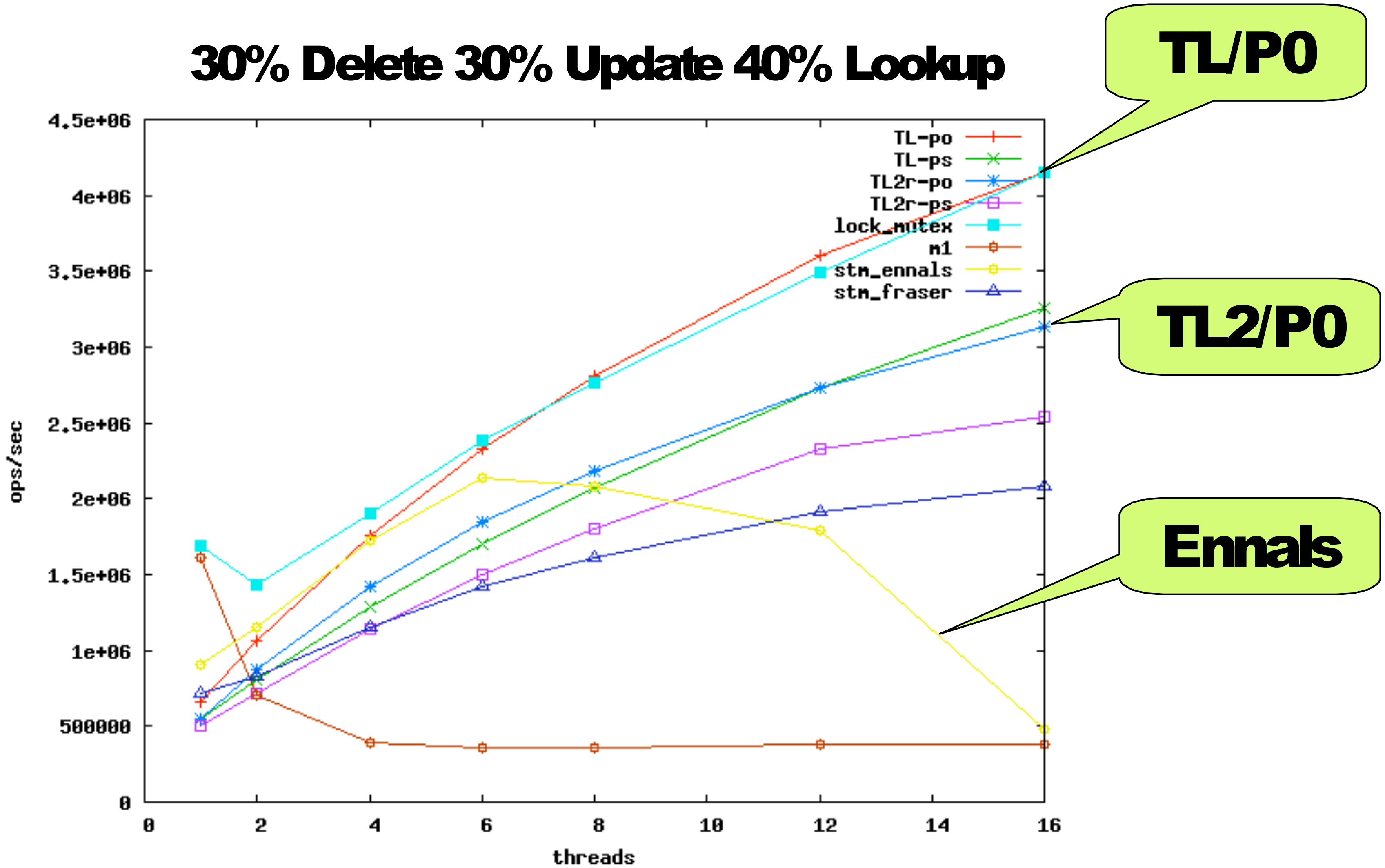


Uncontended Small RB-Tree



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

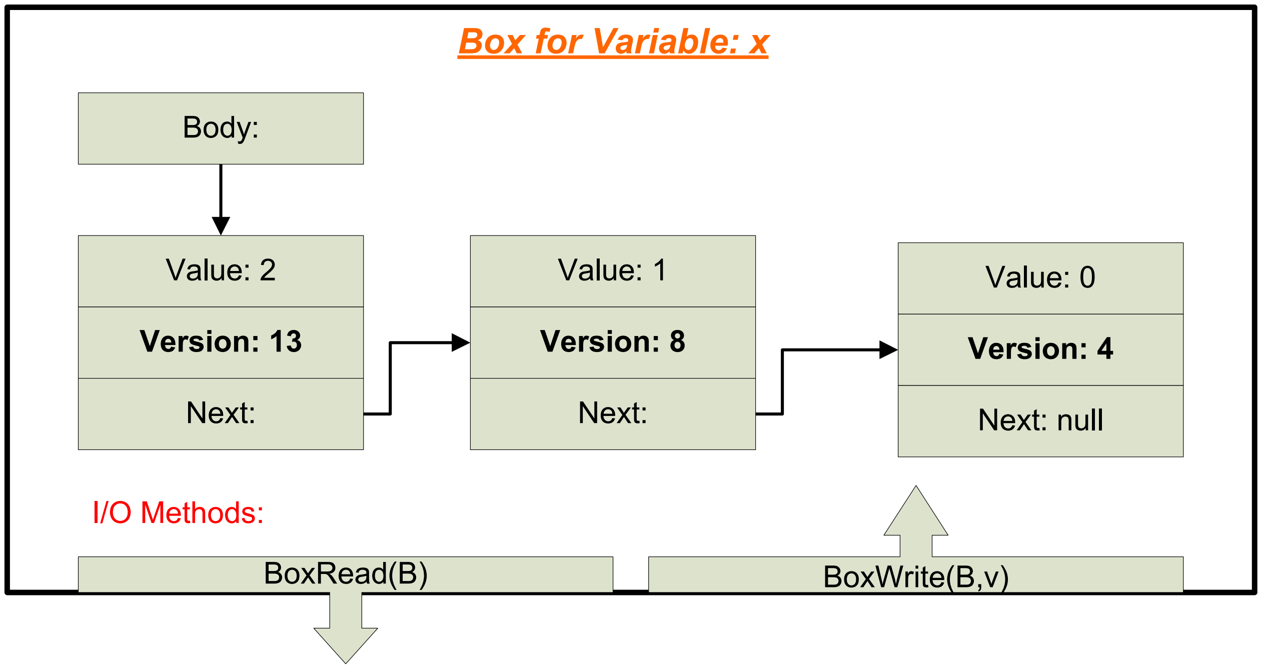
Contended Small RB-Tree



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

JVSTM

- **Variables** are modeled has objects called "**BOXES**"
- Boxes store different versions of their update history inside

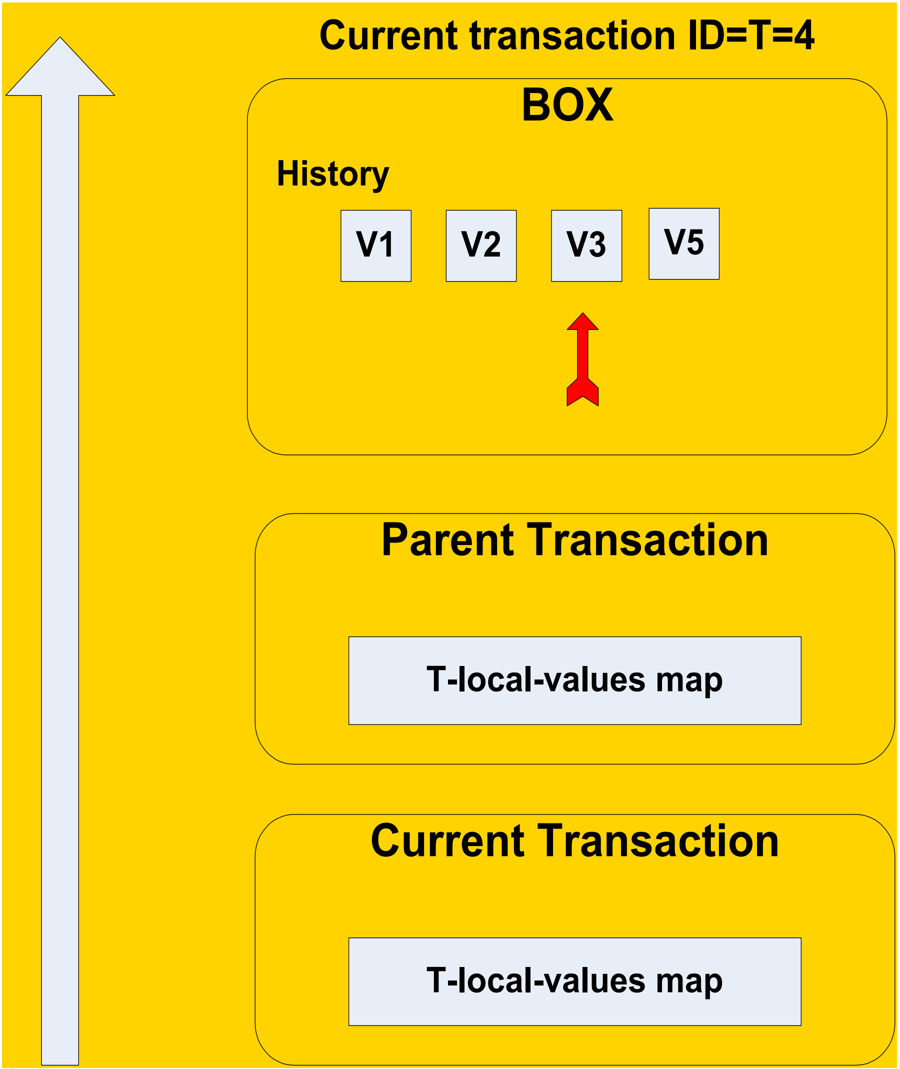


JVSTM – Transactional Data

- Transaction ID
- T-Read Set
  - Boxes used in the transaction
- Local-Values Map
  - Assignments/updates to the variables/boxes
  - BoxWrite()-dosen't changes the value of the Boxes. It adds a mapping from a value to the box in the Local-Values Map.
  - Only when the transaction finishes the inner values of the transaction are placed in the history of the object.

JVSTM – Reading

- Reading is done in a bottom-up direction
- This includes the situation of nested transactions



The End