Speed-Up and Easier Programming with Transactional Memory

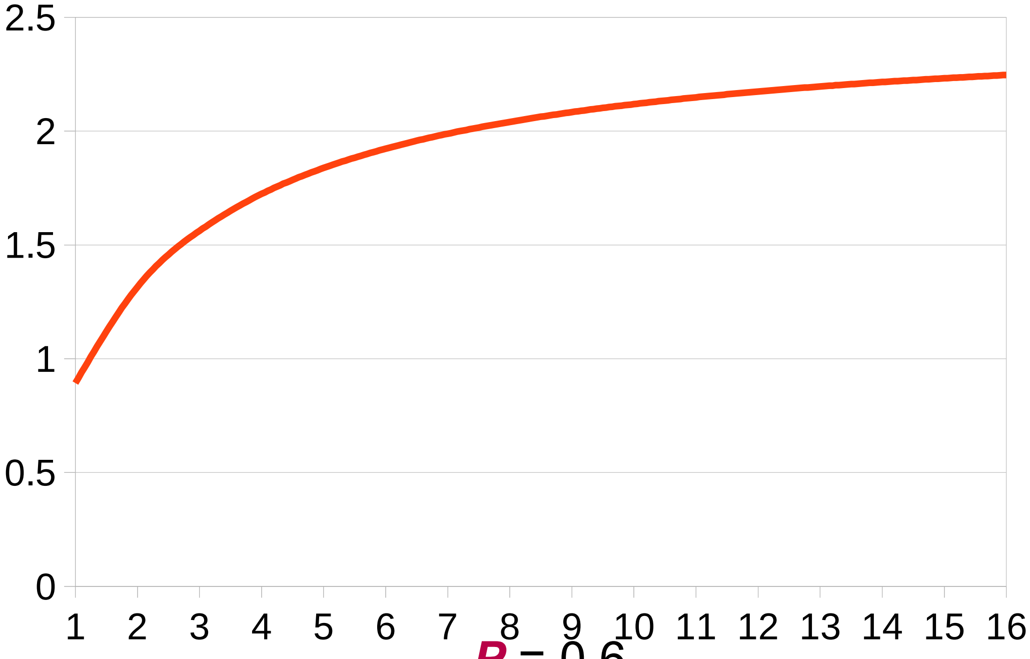
Ulrich Drepper
<drepper@gmail.com>

Programmer's Delight, New York, December 10th 2013

Why? Amdahl!

My More Realistic Version of Amdahl's Law

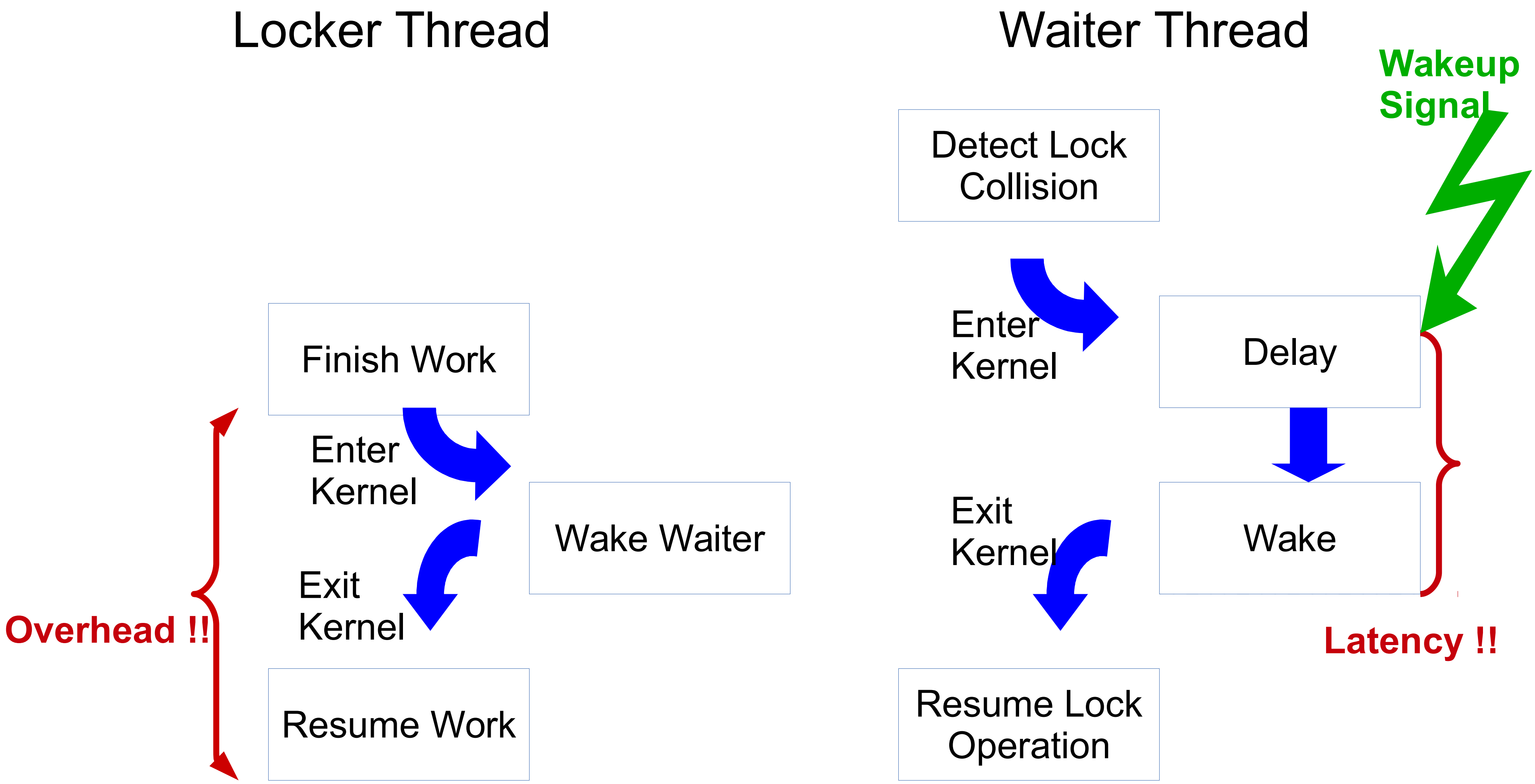
$$S = \frac{1}{(1-P)(1+O_s) + \frac{P}{N}}$$



P = 0.6

Why? Latency!

Mutual Exclusion versus Wait-Free Data Structures



Locker Thread

Finish Work

Enter Kernel

Wake Waiter

Exit Kernel

Resume Work

Overhead !!

Waiter Thread

Detect Lock Collision

Enter Kernel

Delay

Wake

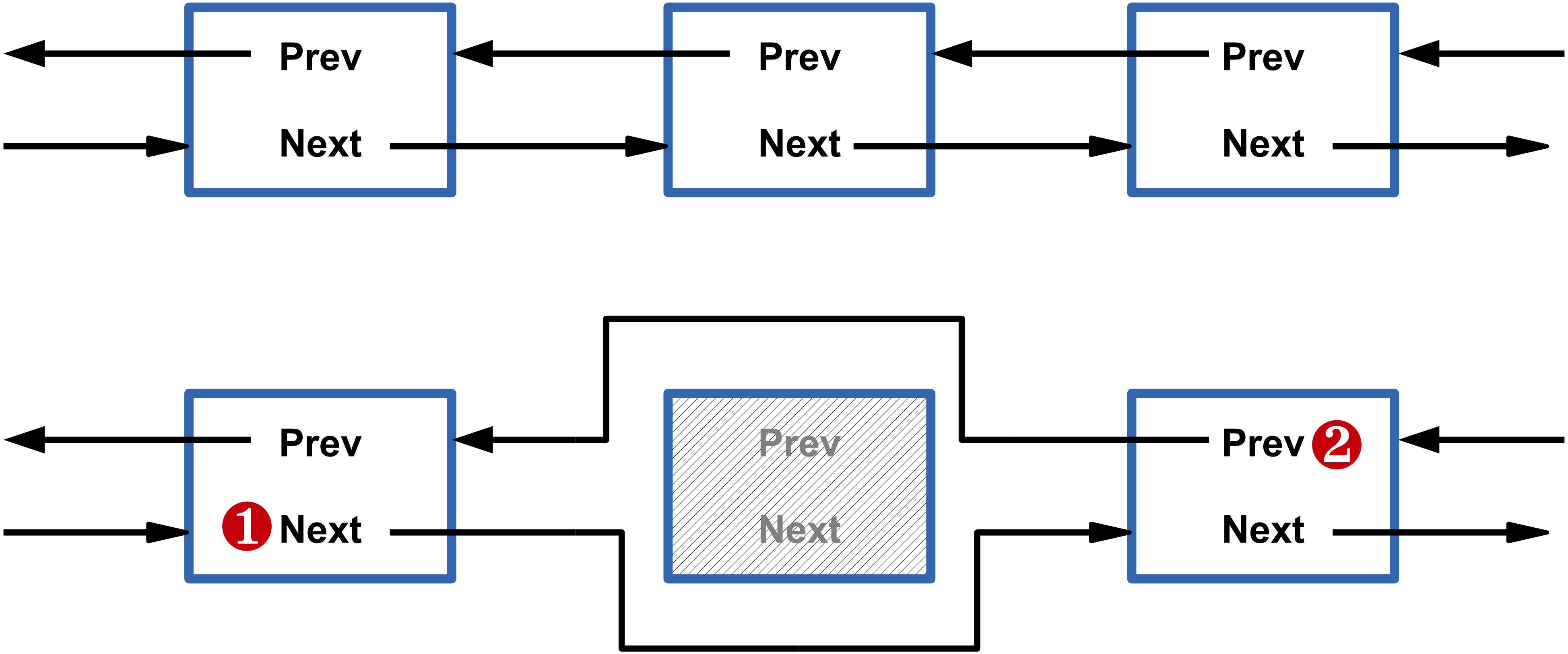
Exit Kernel

Resume Lock Operation

Wakeup Signal

Latency !!

DCAS

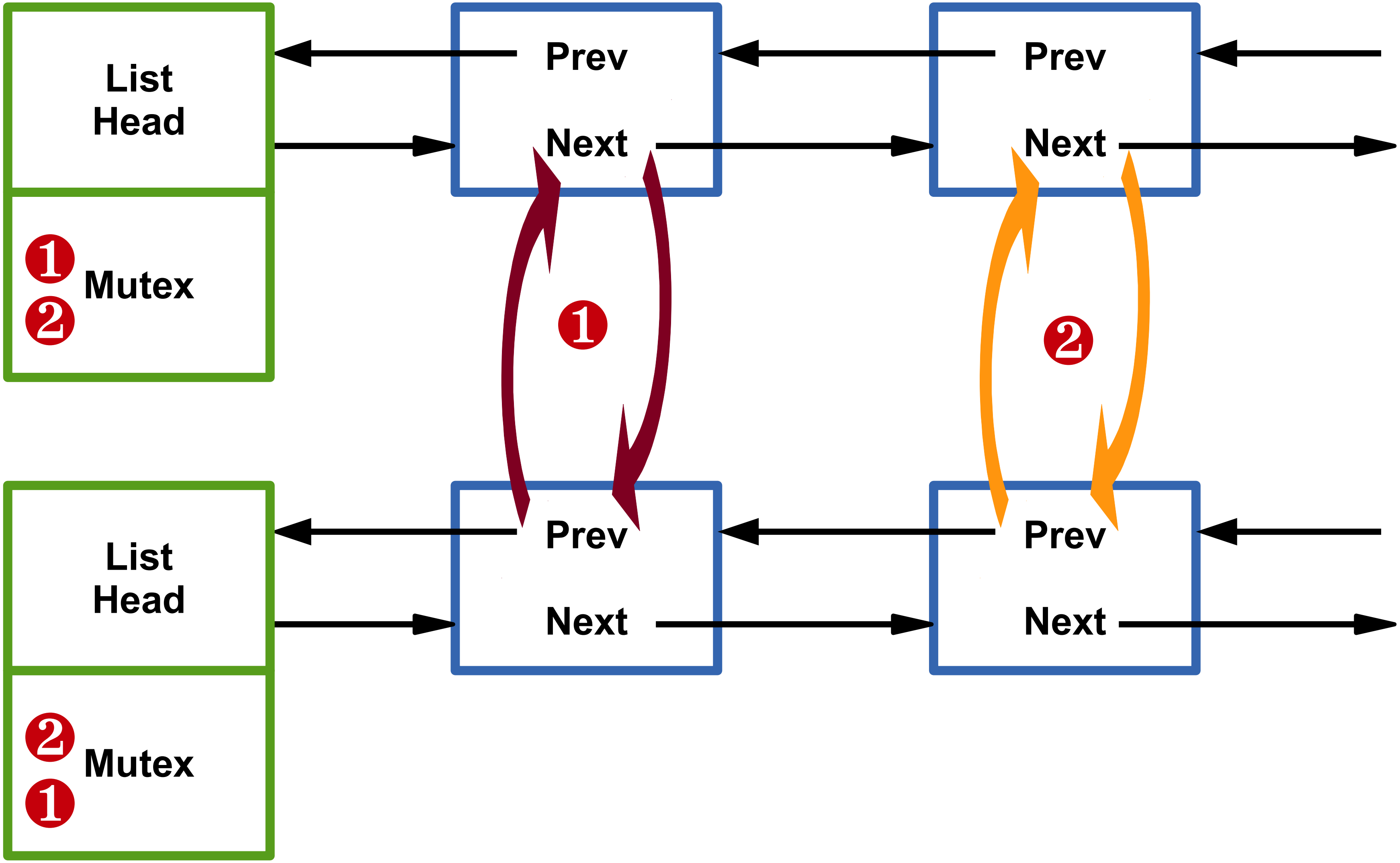

Prev
Next
Prev
Next
Prev
Next
Prev
1 Next
Prev
Next
Prev 2
Next

(Non-)Conflict

Most Common!

	R	W
R	✓	✗
W	✗	✗

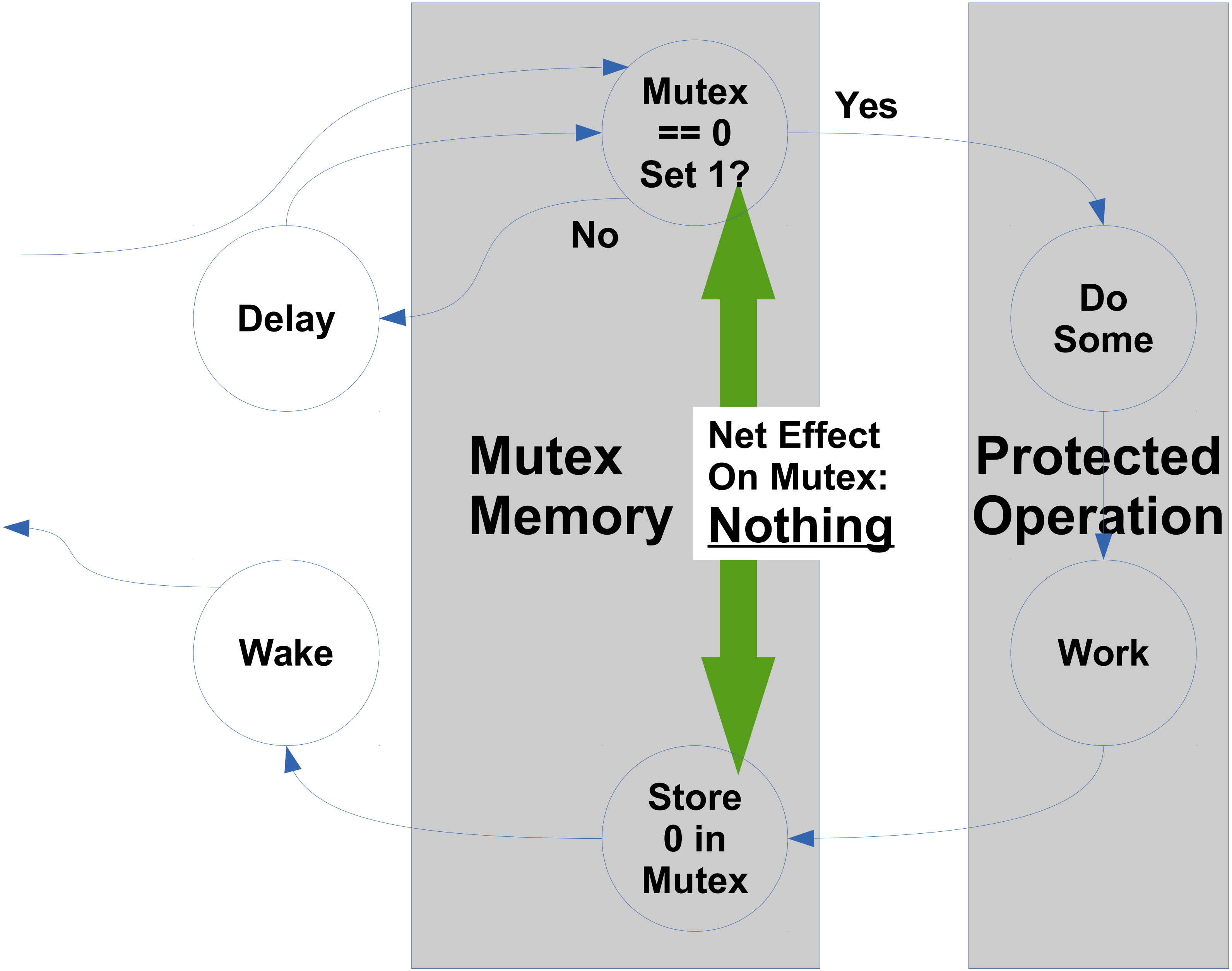
AB/BA


List Head
❶ ❷ Mutex
Prev
Next
Prev
Next
❶
❷
List Head
❷ ❶ Mutex
Prev
Next
Prev
Next

Problems

- Parallel Programming is hard
- Locking is expensive in locker and waiter
- Fine-grained locking needed for more concurrency
  - Makes parallel programming even harder
- Parallelism ***potential*** vs incurred cost

Observation on Locking



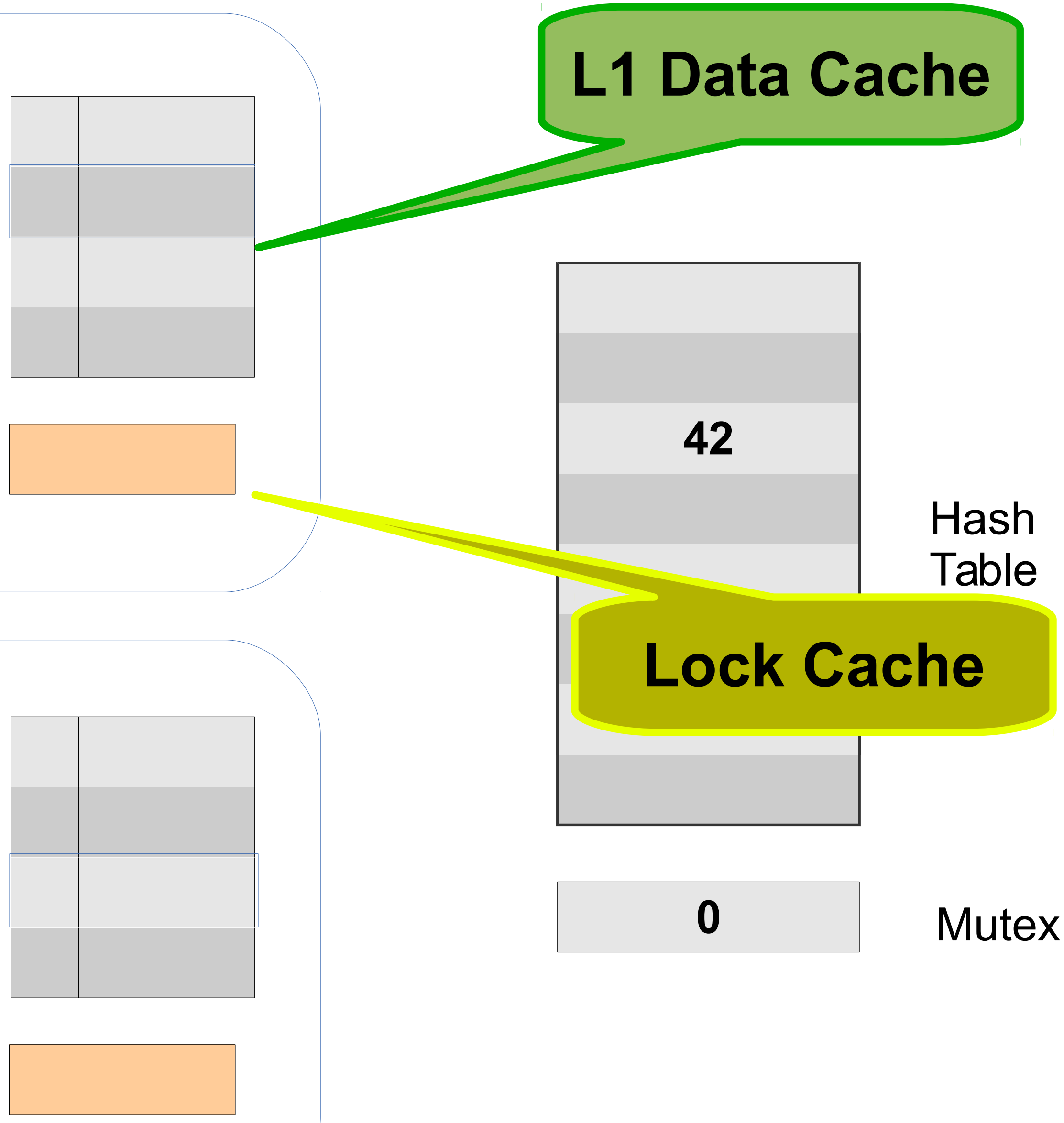
Reader/Writer Workload

Thread 1

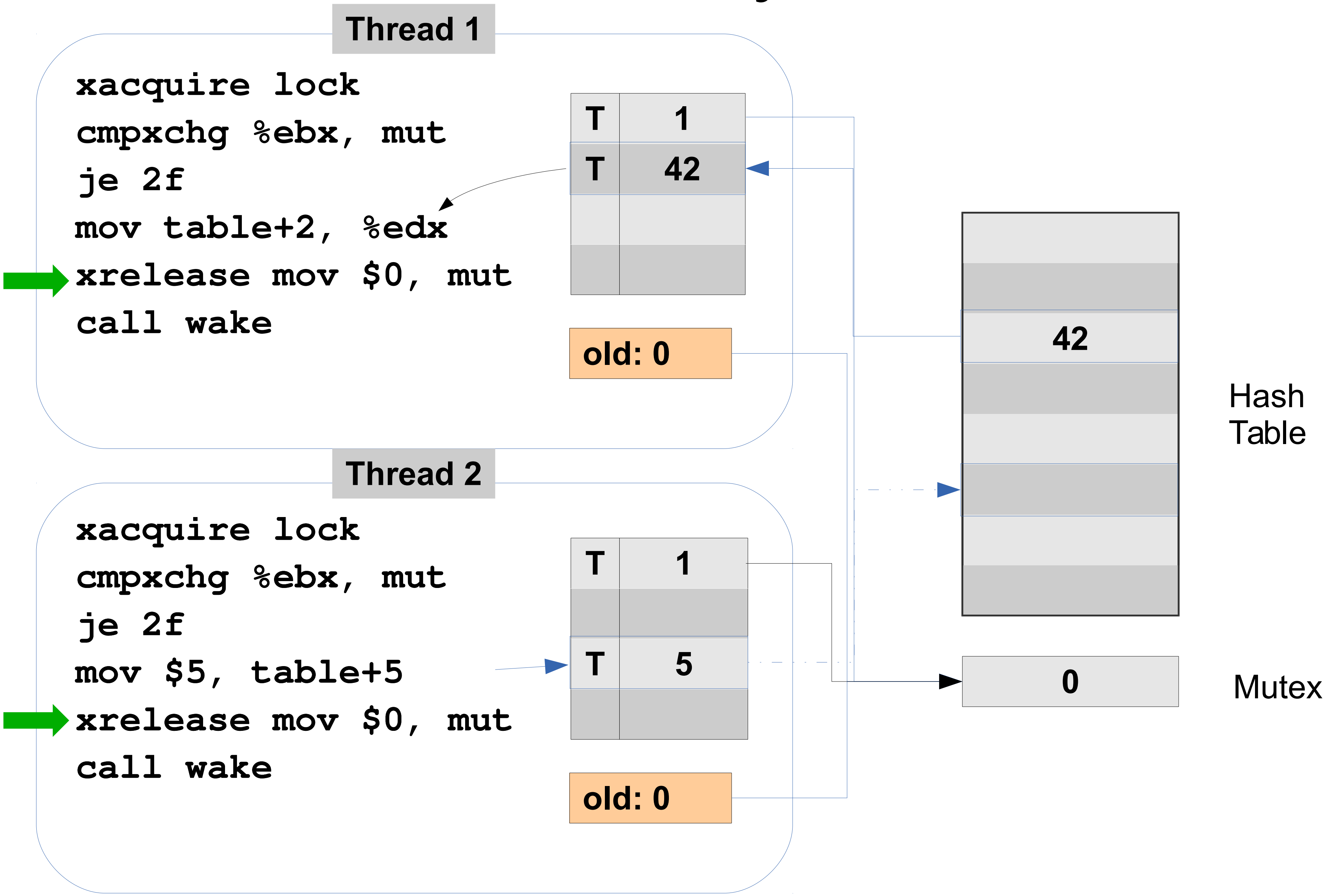
```
xacquire lock
cmpxchg %ebx, mut
je 2f
mov table+2, %edx
xrelease mov $0, mut
call wake
```

Thread 2

```
xacquire lock
cmpxchg %ebx, mut
je 2f
mov $5, table+5
xrelease mov $0, mut
call wake
```



No Concurrency Problem



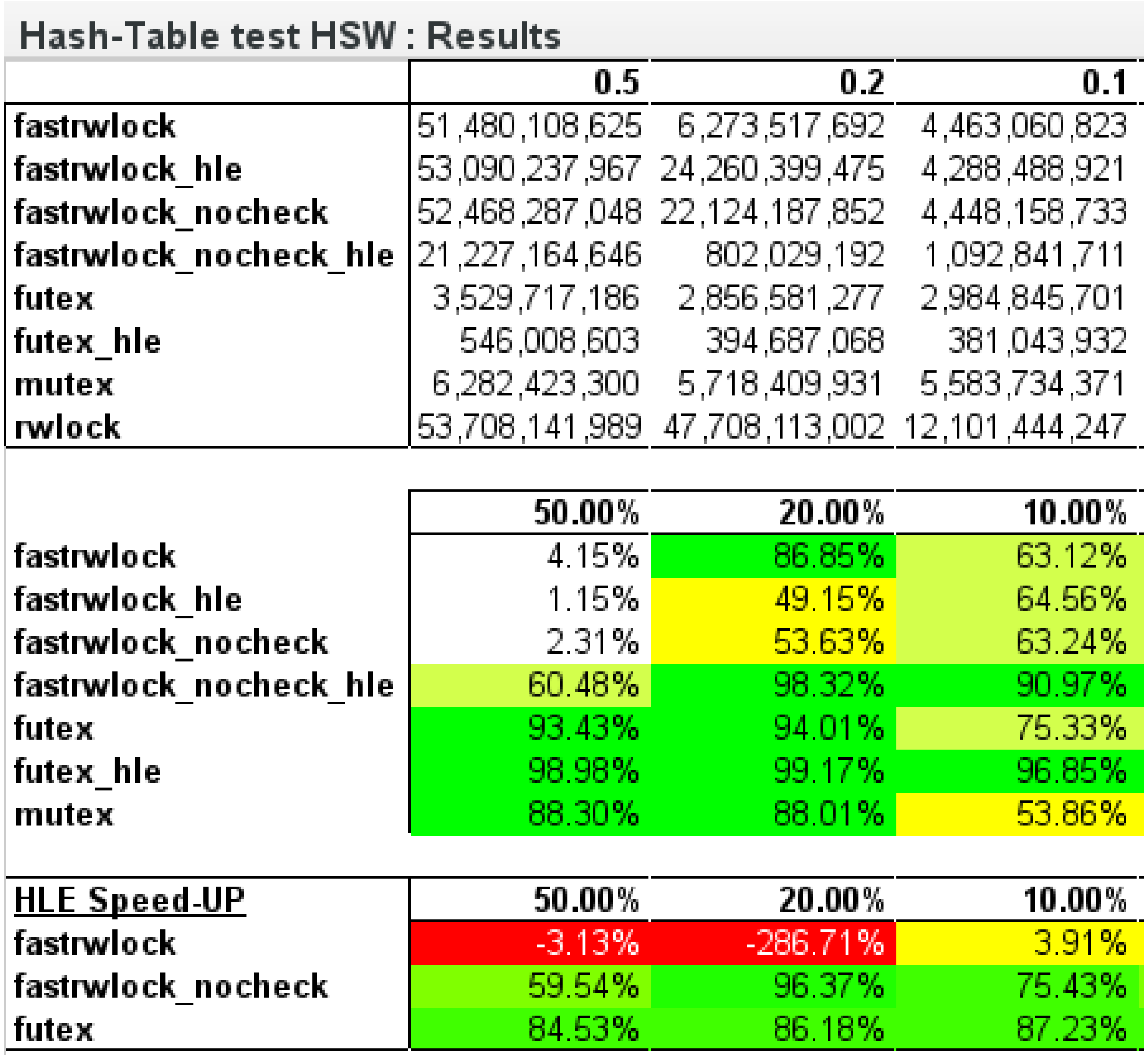
Thread 1

```
xacquire lock
cmpxchg %ebx, mut
je 2f
mov table+2, %edx
xrelease mov $0, mut
call wake
```

Thread 2

```
xacquire lock
cmpxchg %ebx, mut
je 2f
mov $5, table+5
xrelease mov $0, mut
call wake
```

Hash-Table with HLE: Results

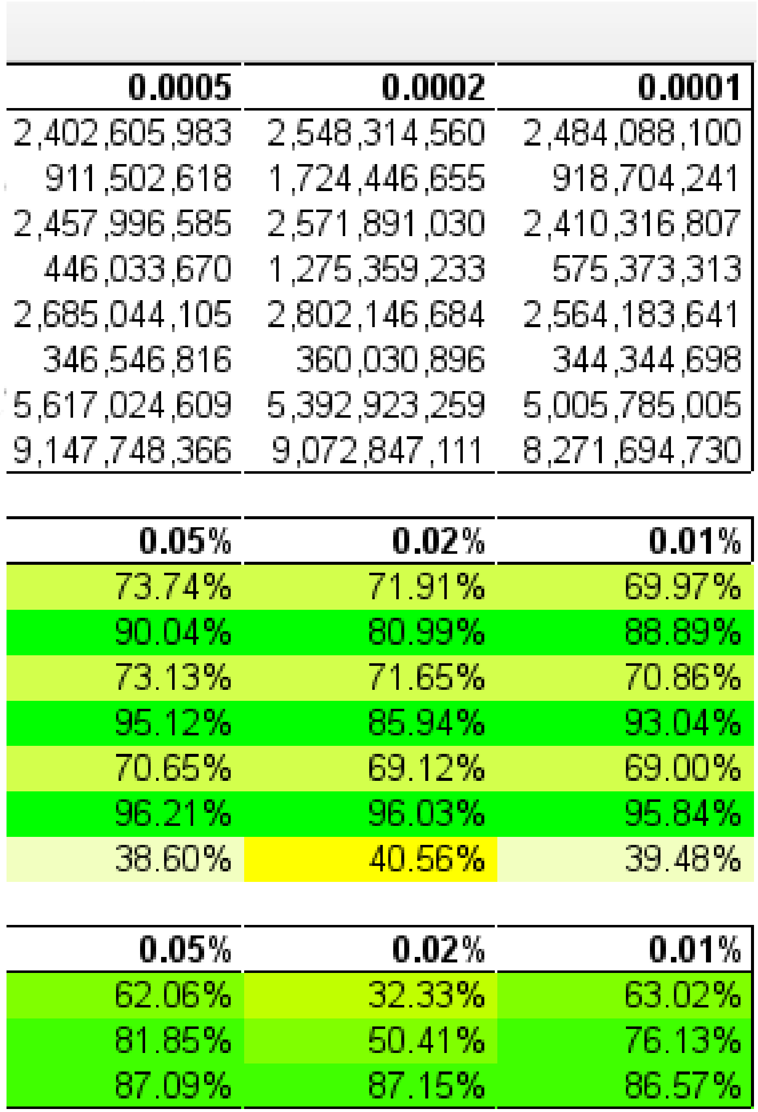


Hash-Table test HSW : Results

	0.5	0.2	0.1
fastrwlock	51,480,108,625	6,273,517,692	4,463,060,823
fastrwlock_hle	53,090,237,967	24,260,399,475	4,288,488,921
fastrwlock_nocheck	52,468,287,048	22,124,187,852	4,448,158,733
fastrwlock_nocheck_hle	21,227,164,646	802,029,192	1,092,841,711
futex	3,529,717,186	2,856,581,277	2,984,845,701
futex_hle	546,008,603	394,687,068	381,043,932
mutex	6,282,423,300	5,718,409,931	5,583,734,371
rwlock	53,708,141,989	47,708,113,002	12,101,444,247

	50.00%	20.00%	10.00%
fastrwlock	4.15%	86.85%	63.12%
fastrwlock_hle	1.15%	49.15%	64.56%
fastrwlock_nocheck	2.31%	53.63%	63.24%
fastrwlock_nocheck_hle	60.48%	98.32%	90.97%
futex	93.43%	94.01%	75.33%
futex_hle	98.98%	99.17%	96.85%
mutex	88.30%	88.01%	53.86%

HLE Speed-UP	50.00%	20.00%	10.00%
fastrwlock	-3.13%	-286.71%	3.91%
fastrwlock_nocheck	59.54%	96.37%	75.43%
futex	84.53%	86.18%	87.23%



0.0005	0.0002	0.0001
2,402,605,983	2,548,314,560	2,484,088,100
911,502,618	1,724,446,655	918,704,241
2,457,996,585	2,571,891,030	2,410,316,807
446,033,670	1,275,359,233	575,373,313
2,685,044,105	2,802,146,684	2,564,183,641
346,546,816	360,030,896	344,344,698
5,617,024,609	5,392,923,259	5,005,785,005
9,147,748,366	9,072,847,111	8,271,694,730

0.05%	0.02%	0.01%
73.74%	71.91%	69.97%
90.04%	80.99%	88.89%
73.13%	71.65%	70.86%
95.12%	85.94%	93.04%
70.65%	69.12%	69.00%
96.21%	96.03%	95.84%
38.60%	40.56%	39.48%

0.05%	0.02%	0.01%
62.06%	32.33%	63.02%
81.85%	50.41%	76.13%
87.09%	87.15%	86.57%

Complete Table: https://www.akkadia.org/drepper/hsw.html

Combining: Old Style

```
if (from == to)
  lists[from].lock();
else if (from < to) {
  lists[from].lock();
  lists[to].lock();
} else {
  lists[to].lock();
  lists[from].lock();
}
```

AB/BA-controlled locking

```
if (from != to) {
  auto it1 = lists[from].begin();
  auto it2 = lists[to].begin();
  advance(it1, fromidx);
  advance(it2, toidx);

  std::swap(it1, it2);
}
```

```
if (from == to)
  lists[from].unlock();
else if (from < to) {
  lists[from].unlock();
  lists[to].unlock();
} else {
  lists[to].unlock();
  lists[from].unlock();
}
```

AB/BA-controlled unlocking

Combining: New Style

```
__transaction_atomic {
```

AB/BA-controlled locking

```
if (from != to) {
  auto it1 = lists[from].begin();
  auto it2 = lists[to].begin();
  advance(it1, fromidx);
  advance(it2, toidx);

  std::swap(it1, it2);
}
```

```
}
```

AB/BA-controlled unlocking

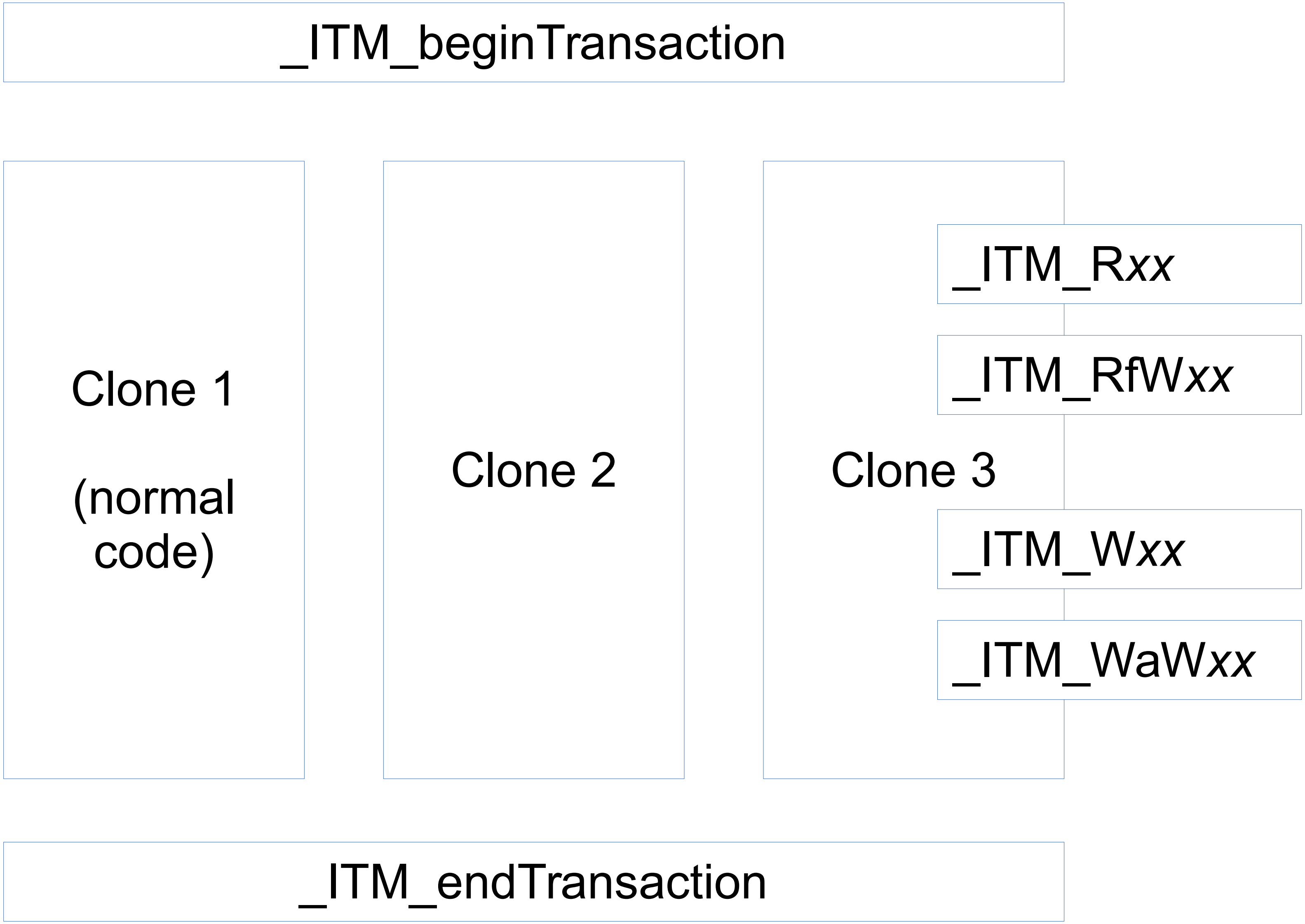
Not Slow (on HSW)

- 2 threads
- 3 lists
- 32 elements
- Swap random elements from random lists

  with locking: 464 cycles per round

  with TM: 403 cycles per round (-13%)

Slower on pre-HSW



_ITM_beginTransaction

Clone 1

(normal code)

Clone 2

Clone 3

_ITM_R*xx*

_ITM_RfW*xx*

_ITM_W*xx*

_ITM_WaW*xx*

_ITM_endTransaction

Questions?