ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Unlocking Energy

Babak Falsafi, Rachid Guerraoui, Javier Picorel, Vasileios Trigonakis

Conference Dilemma: To Sleep or Not to Sleep?

Do not sleep for such short duration

Energy Efficiency Through Synchronization (Locking)

Why lock-based synchronization?

1. Concurrent systems synchronize with locks
2. Locks are well-defined abstractions
   → lock() / unlock()
3. Locking strategies affect power consumption

busy lock

lock()

wait!
(waste time
and energy)

Lock Waiting Techniques

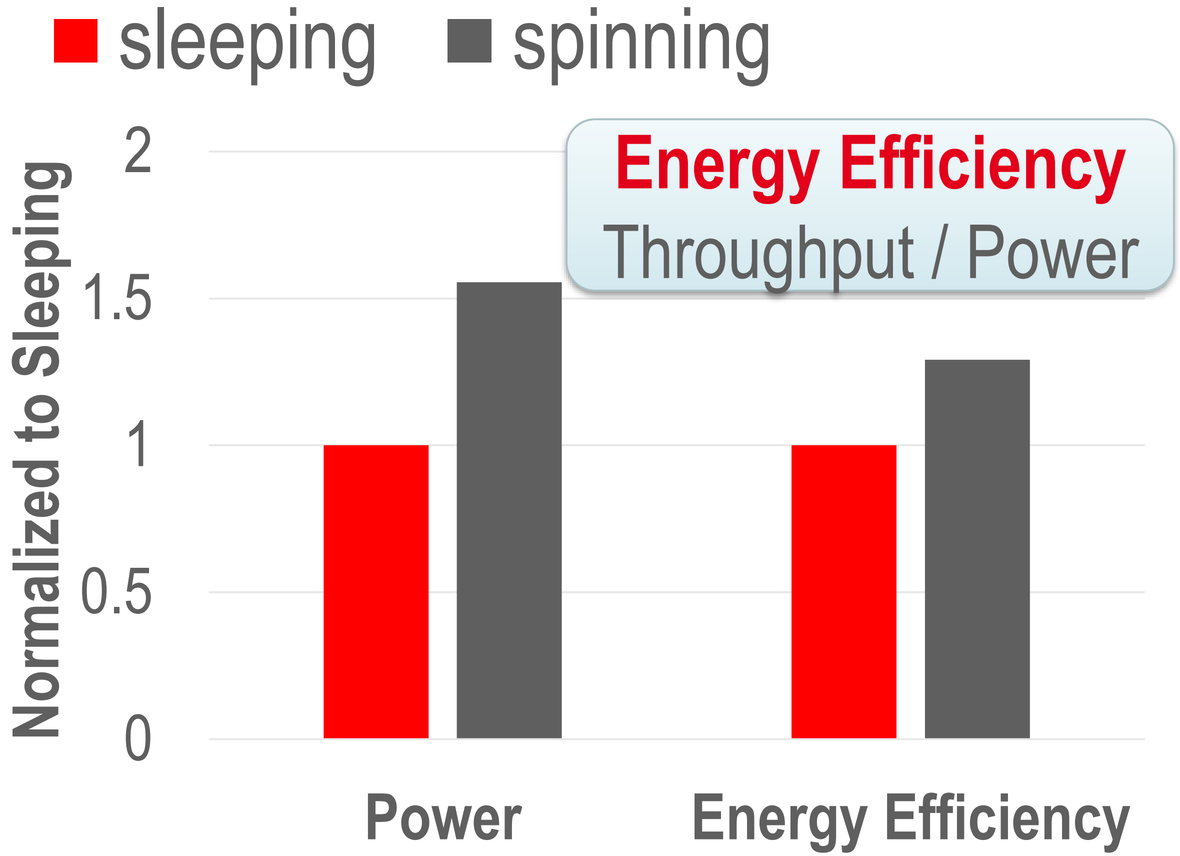
busy lock

sleeping (blocking)

busy waiting (spinning)

Q. Is sleeping energy-friendly?

Java CopyOnWriteArrayList



Locking is a good candidate for reducing energy consumption

Energy Efficiency By Improving Locking

1. Concurrent systems synchronize with locks
2. Locks are well-defined abstractions
3. Locking strategies affect power consumption
4. ***Energy efficiency* and *throughput* go *hand in hand***
   in the context of lock algorithms

Outline

- Motivation

busy lock

busy waiting → hurts power consumption

spin-then-sleep "cleverly"

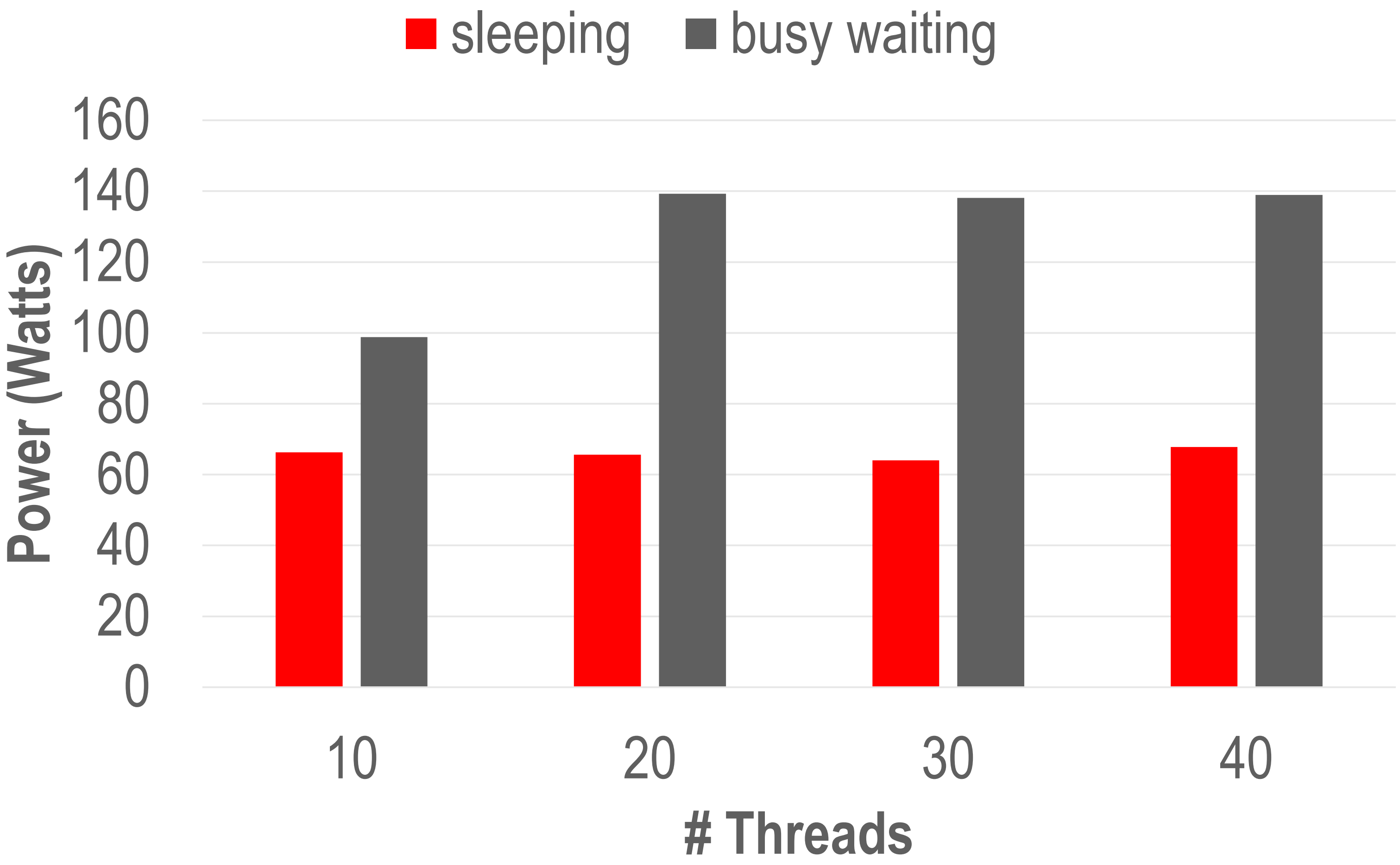
sleeping → saves power, but hurts throughput

Target platform
2-socket Intel Ivy Bridge
20 cores, 40 hyper-threads

- Improving the energy efficiency of systems

Power Consumption of Waiting

1 lock, never released



Observations

1. Sleeping power-friendly
2. Busy waiting

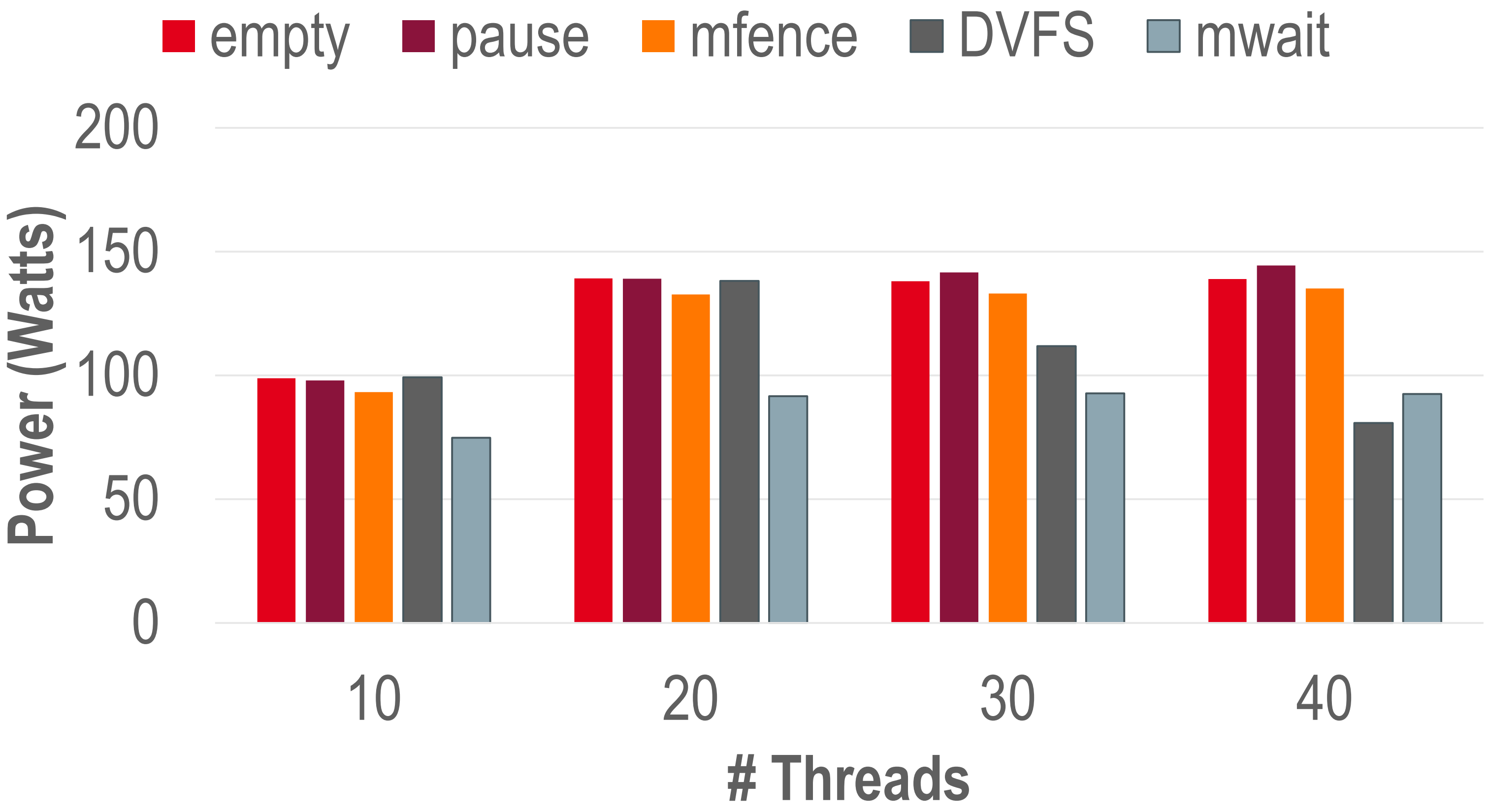
```
while (*lock != FREE) {}
```

Busy waiting is very power hungry

Reducing Power of Busy Waiting

1 lock, never released

while (*lock != FREE) { ?? }



Observations

1. empty: 1 iteration / cycle
2. Intel docs: use pause
3. pause not ideal
4. mfence > pause
5. Still, mfence ~5% better
6. DVFS and mwait are **not practical** (details in the paper)

Power consumption of busy waiting cannot be practically reduced

Outline

- Motivation

busy lock

busy waiting → hurts power consumption

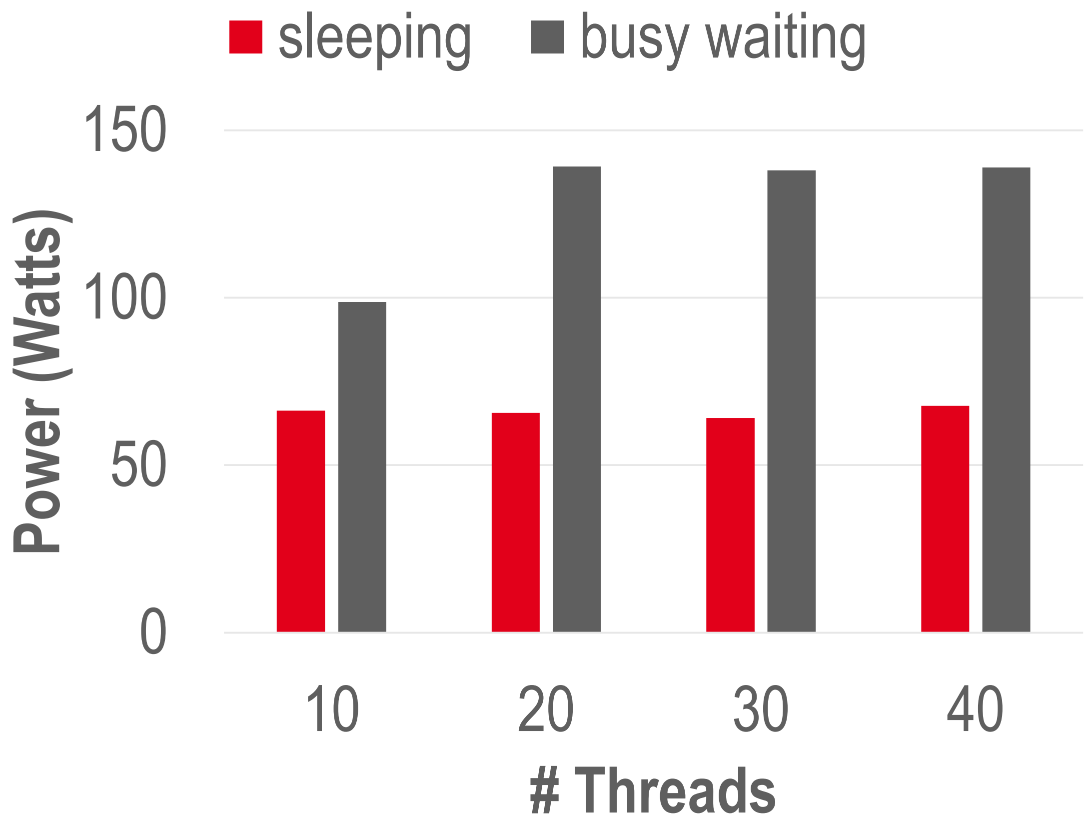
spin-then-sleep "cleverly"

sleeping → saves power, but hurts throughput

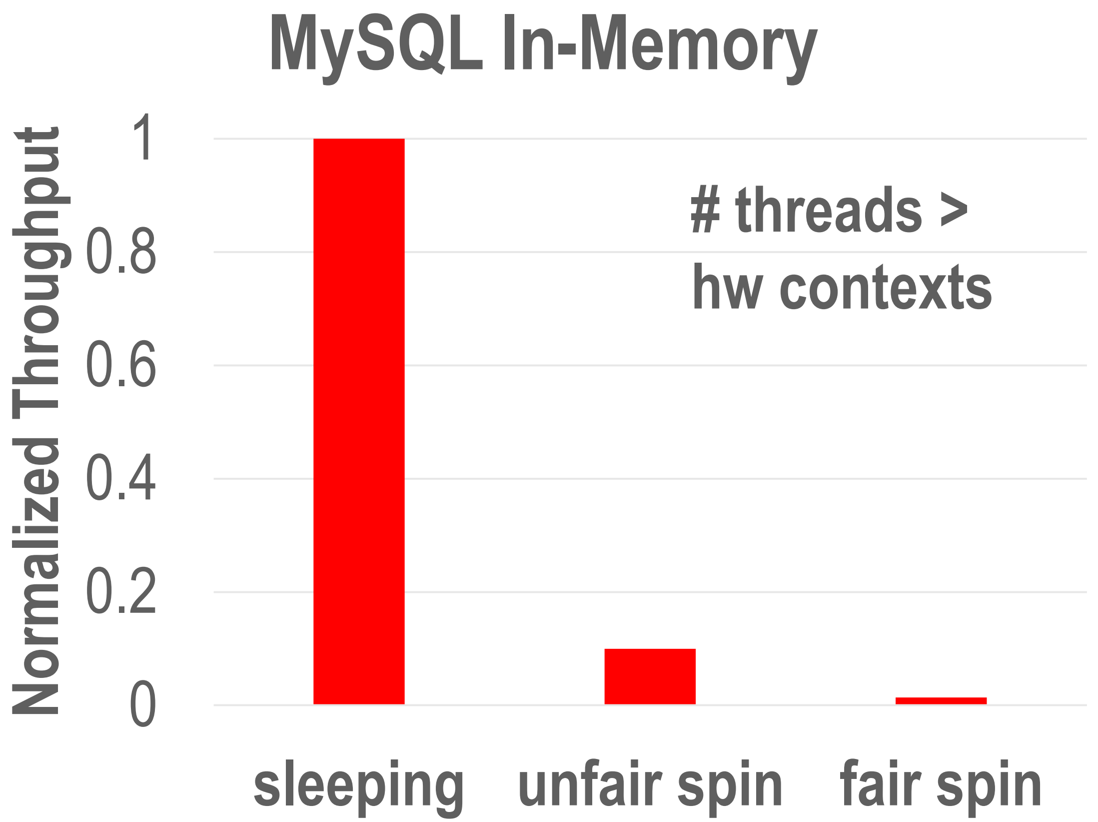
- Improving the energy efficiency of systems

Sleeping Might Be Necessary (For Two Reasons)

1 Power Consumption—Waiting



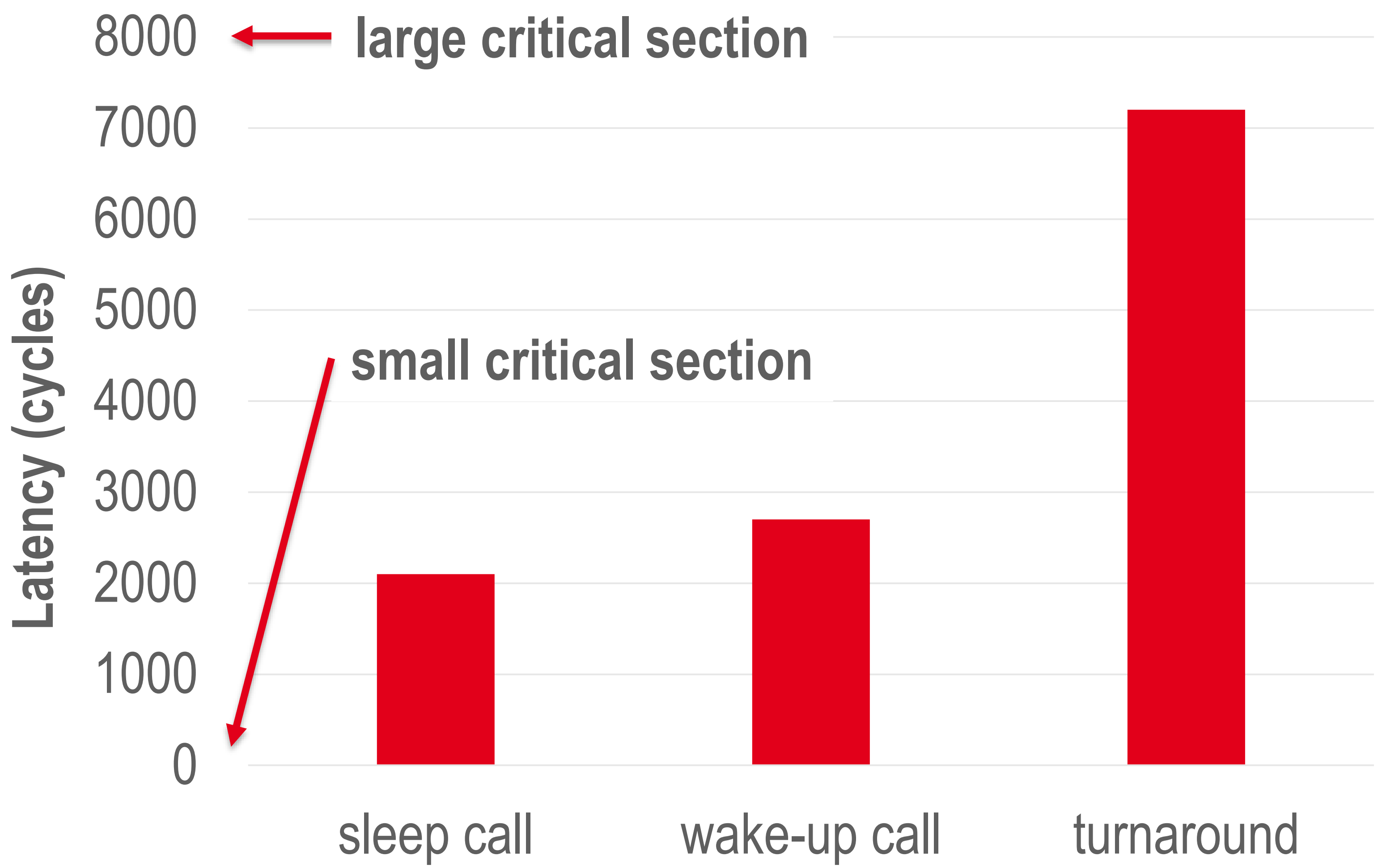
2 Locks with Multiprogramming



Sleeping can reduce power consumption (and more)

Latency: The Price of Sleeping

2 threads invoke futex
1 sleeps,1 wakes up



Observations

1. Sleep call:
   release context
2. Wake-up call:
   to handover the lock
3. Turnaround latency ≈
   lock handover latency

Frequent sleep/wake-up calls reduce throughput without saving energy

Outline

- Motivation

busy lock

busy waiting → hurts power consumption

spin-then-sleep "cleverly"

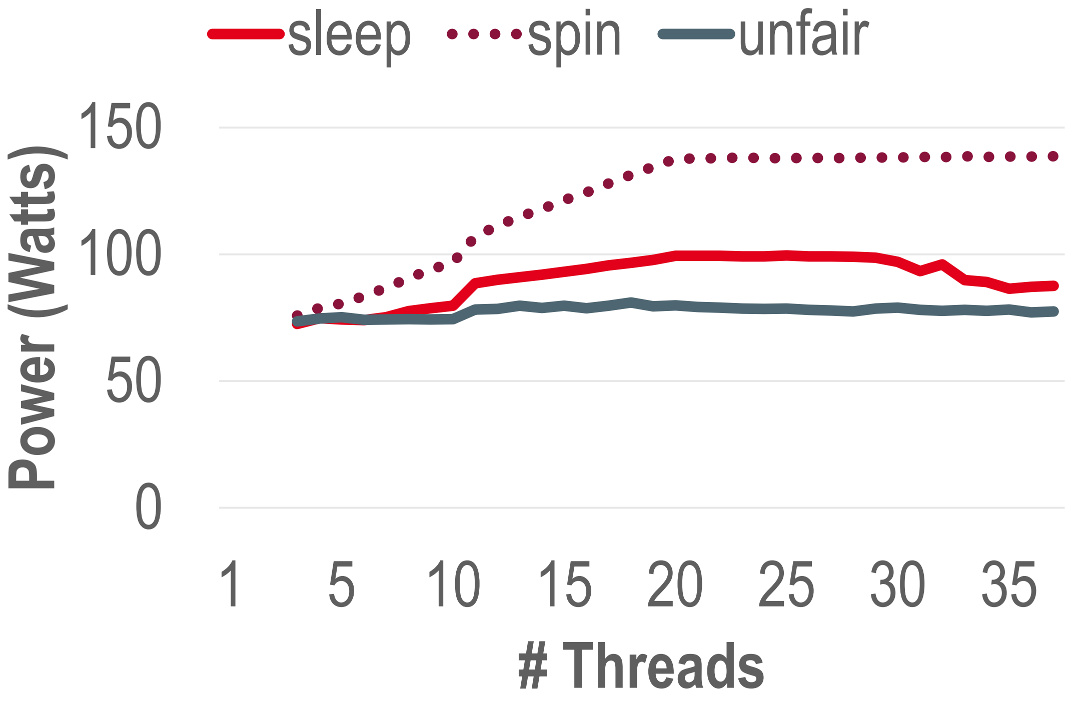
sleeping → saves power, but hurts throughput

- Improving the energy efficiency of systems

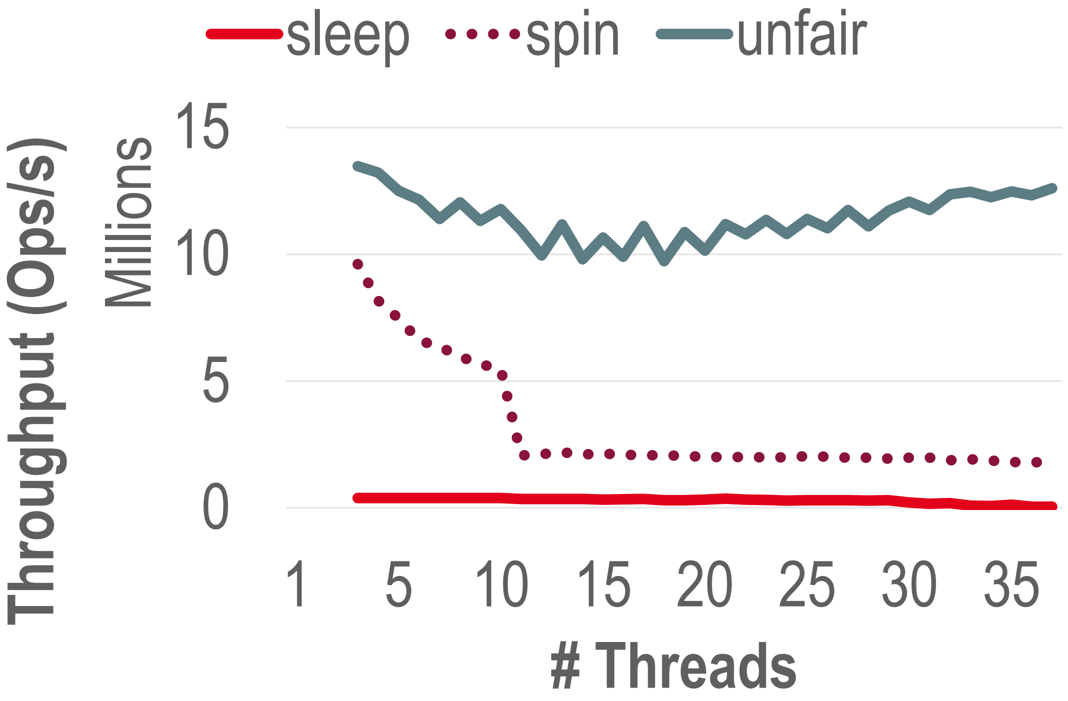
Reducing Fairness: Sleeping for Long Durations

Passing a “token” from thread to thread

Power Consumption



Communication Throughput



unfair: 1000:1 spin-to-sleep ratio (while 2 threads spin, the rest sleep)

Trade fairness for energy efficiency

How can we use these results in designing locks?

Problems of Pthread MUTEX lock

lock()

MUTEX
For up to 100 attempts
spin with pause
if still busy, sleep

1. **Spins less than sleep latencies**
2. **Spins with pause**

unlock()

MUTEX
release in user space
wake up a thread

3. **Always wakes up a thread**

Pthread MUTEX does not take into account the sleep overheads

MUTEXEE: An Optimized MUTEX Lock

lock()

MUTEX	MUTEXEE
For up to 100 attempts	For up to ~8000 cycles
spin with pause	spin with mfence
if still busy, sleep	

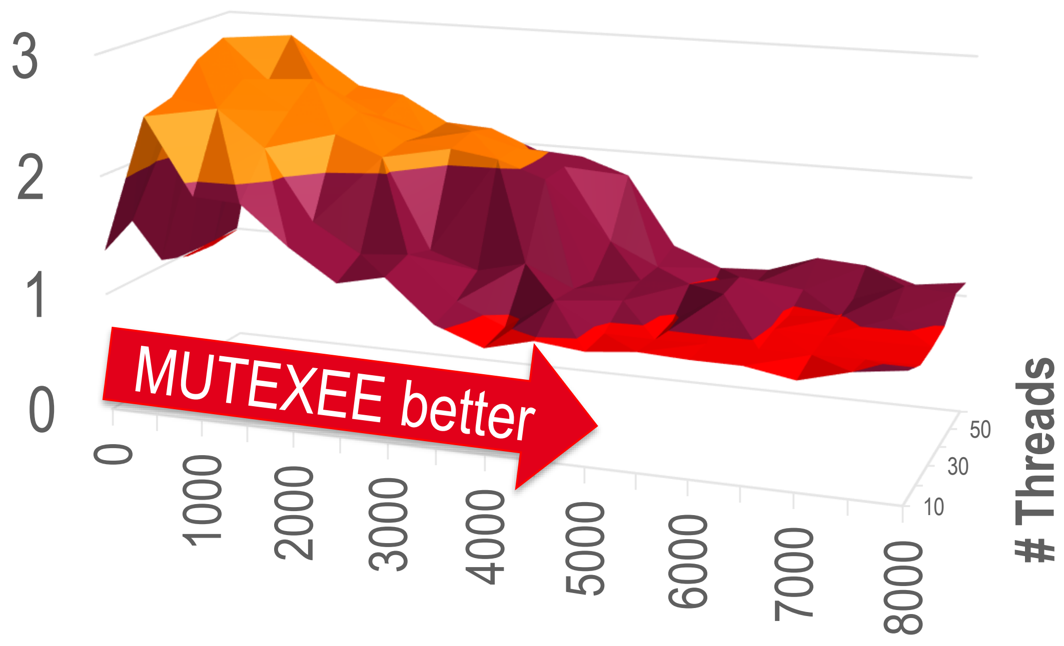
unlock()

MUTEX	MUTEXEE
release in user space (lock->locked = 0)	
	wait in user space (~300 cycles)
wake up a thread	

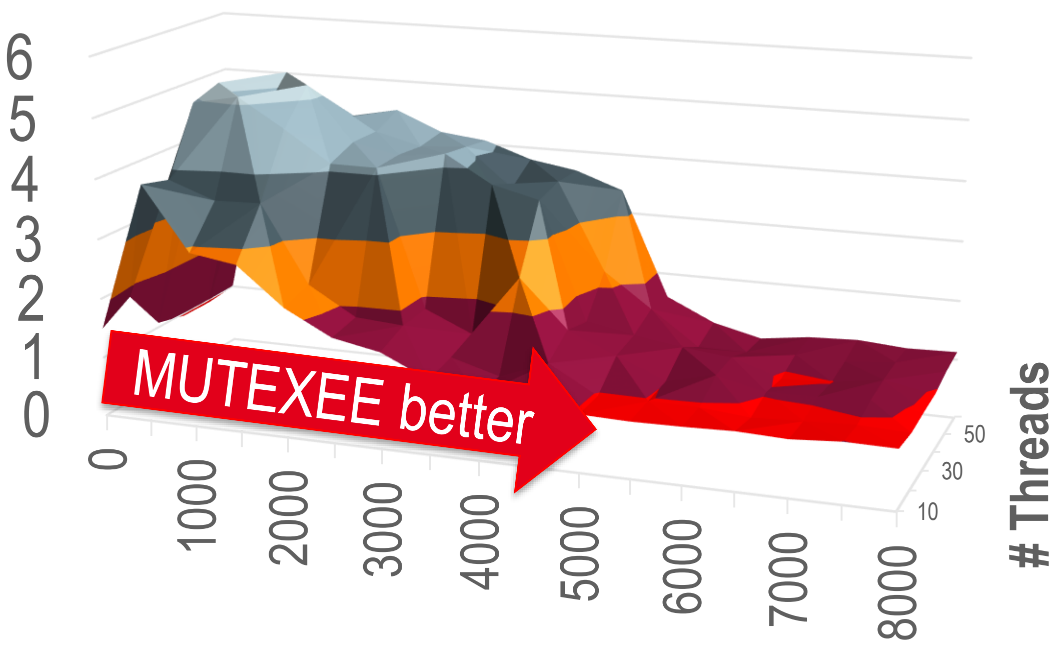
Performance of MUTEXEE over MUTEX

One lock

Throughput



Energy Efficiency



Fairness: results and analysis in the paper

MUTEXEE fixes the problematic cases of MUTEX

Outline

- Motivation

busy lock

busy waiting → hurts power consumption

spin-then-sleep "cleverly"

sleeping → saves power, but hurts throughput

- Improving the energy efficiency of systems

Evaluation: Improving Energy Efficiency of Systems Through Locks

Six modern software systems

– Overload pthread mutex

HamsterDB embedded database

Kyoto Cabinet

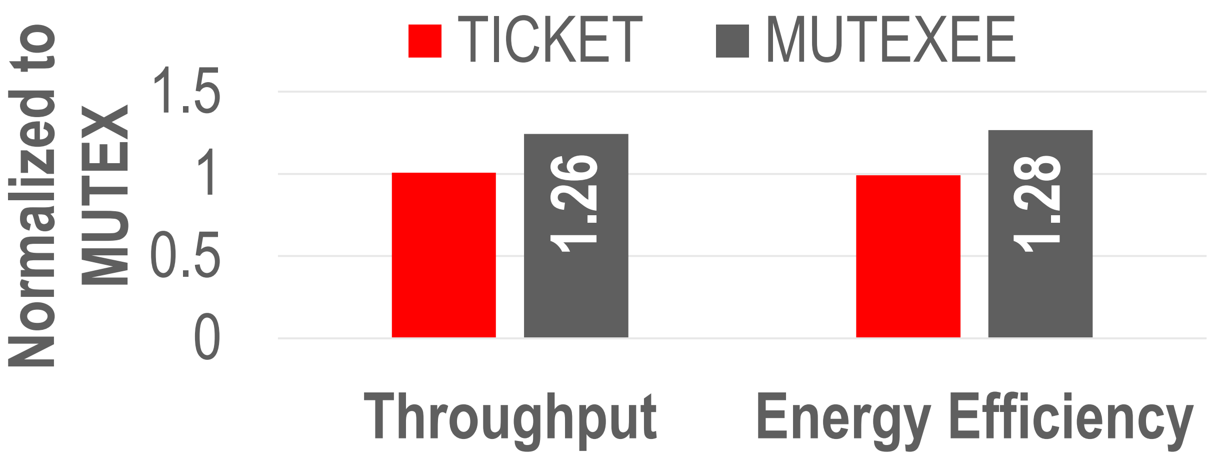
MEMCACHED

RocksDB

MySQL

SQLite

Average Throughput and Energy Efficiency



Results

1. **Benefits**: Avoid sleeping
2. Sleeping is sometimes necessary
3. Throughput-driven benefits
4. MUTEXEE >> MUTEX

Locking can indeed be used to improve the energy efficiency of systems

Concluding Remarks

- An analysis of the energy efficiency of lock-based synchronization
  - Energy efficiency of locks goes hand in hand with throughput
  - MUTEXEE: an optimized MUTEX lock

→ **Locking can be used to improve the energy efficiency of systems**

- LOCKIN: https://github.com/LPD-EPFL/lockin

THANK YOU! QUESTIONS?