6.5081: Locks

App wants to multiple cores
Kernel must handle parallel system calls
Access shared data structures in parallel

⇒ Locks for correct sharing
Locks can limit performance
35 YEARS OF MICROPROCESSOR TREND DATA

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore
Why locks?  Avoid race conditions.

Before

CPU

After

CPU1

Lost page
Lock abstraction

One process can acquire lock, struct lock?

acquire(\alpha l)

\text{Critical section}

release(\alpha l)

Programs have many locks \Rightarrow more parallelism
When to lock?

Conservative rule: 2 processes access a shared data structure, one is writer $\Rightarrow$ lock data structure

too strict: lock-free programming

too loose: points("", "")
Could locking be atomic:

every start has a lock → - locking

rename('d1/x', 'd2/y')

1 lock d1; erase x; release d1

2 lock d2; add y; release d2

Need: lock d1+d2; erase + add; release d2+d3
Lock perspectives

1) Locks help avoid lost updates
2) Locks make multi-step ops atomic
3) Locks help maintain invariant
Deadlock

acquire(x1)

acquire(x2)

deadlock

CPU1

rename(d1,x,d2,y)

acquire(c1)

Solution: order locks

CPU2

rename(b2,b1,d1,b1)

acquire(d1)

acquire(d2)

deadlock

Solutions: order locks in order
Locks vs. modularity

Lock ordering \rightarrow \text{global}

M_1.g() \rightarrow M_2.f()

locks M_2 uses

Internals of M_2 in terms of locks must be visible to M_1
Locks vs. performance

1. Start with coarse-grained locks
2. Measure clock is completed

Need to split update structures
Best split is a challenge
May need to rewrite to code too

= lot of work!
Case study var: w prints

lock roles

1) protect this database
2) tail end is in flight
3) hw registers have one writer
Broken: acquire(serialize lock +1) 

while (1) {
    if (true) {
        if (locked = 0) {
            return;
        }
        locked = 1;
    } else {
        B
    }
}
Hardware test and set support. 

clmoswap addr, r1, r2 

lock addr 

\[ \text{tmp} \leftarrow \ast \text{addr} \]

\[ \ast \text{addr} \leftarrow r1 \]

\[ r2 \leftarrow \text{tmp} \]

unlock

impl detail dependent on mem system
Memory address

Locked ≠ φ

concurrent execution

Wrong

Single core

Ok
Wrap up:

- Locks good for correctness can be bad for perf

- Locks complicate programming

- Don't share if you don't have to

- Start with coarse-grained - use race detector
Figure 1: FPU40-C000 top-level block diagram.