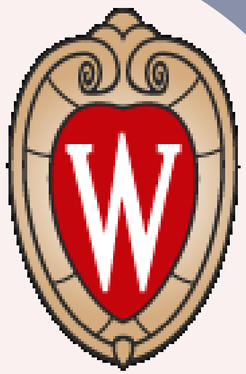


Avoiding Scheduler Subversion using Scheduler-Cooperative Locks

Yuvraj Patel, Leon Yang, Leo Arulraj,
Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, Michael Swift

University of Wisconsin-Madison

Once upon a time ....

Processes

P0 P1

Accessing a kernel service

Once upon a time ....

Processes

P0 P1

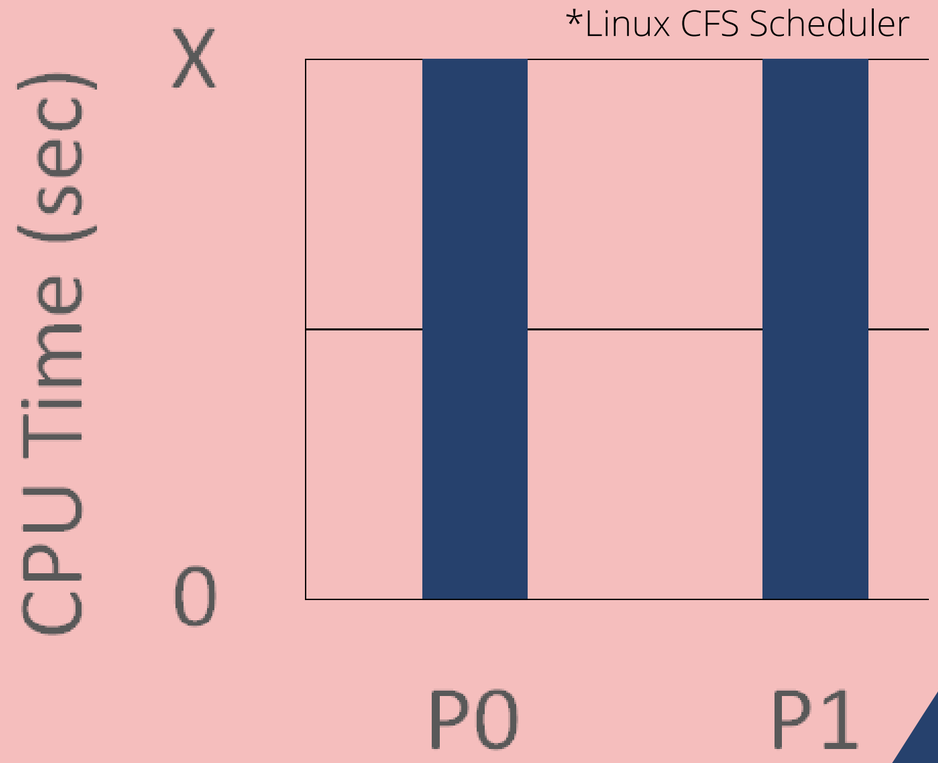
Default priority

Accessing a kernel service

Once upon a time ....

Processes

P0 P1



Accessing a kernel service

Default priority

Schedulers are important for resource sharing

Let us now introduce a lock....

A ticket lock reduces wait-time and avoids starvation

If P0 holds the lock twice as long as P1

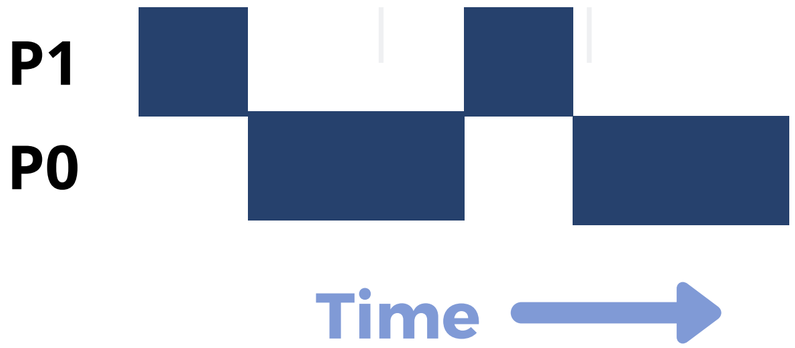
If P0 holds the lock twice as long as P1

If P0 holds the lock twice as long as P1

While in critical section, P0 uses twice much CPU as P1

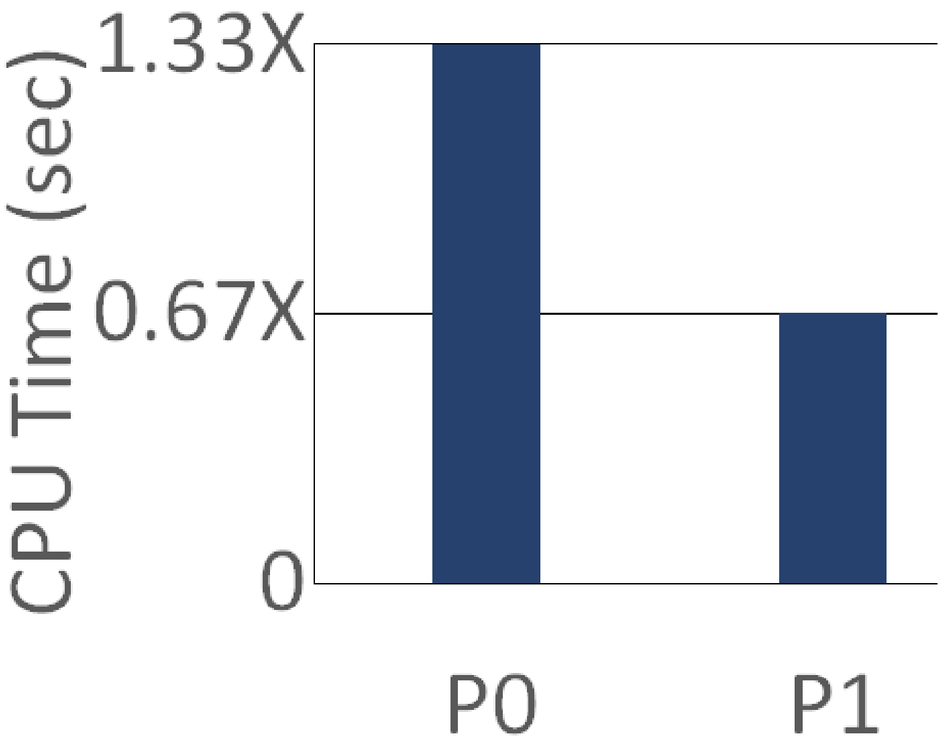
If P0 holds the lock twice as long as P1

If P0 holds the lock twice as long as P1



If P0 holds the lock twice as long as P1

Every time P0 enters critical section, it uses twice as much CPU as P1



Despite 50:50 allocation, P0 uses twice as much CPU as P1's

Scheduler Subversion

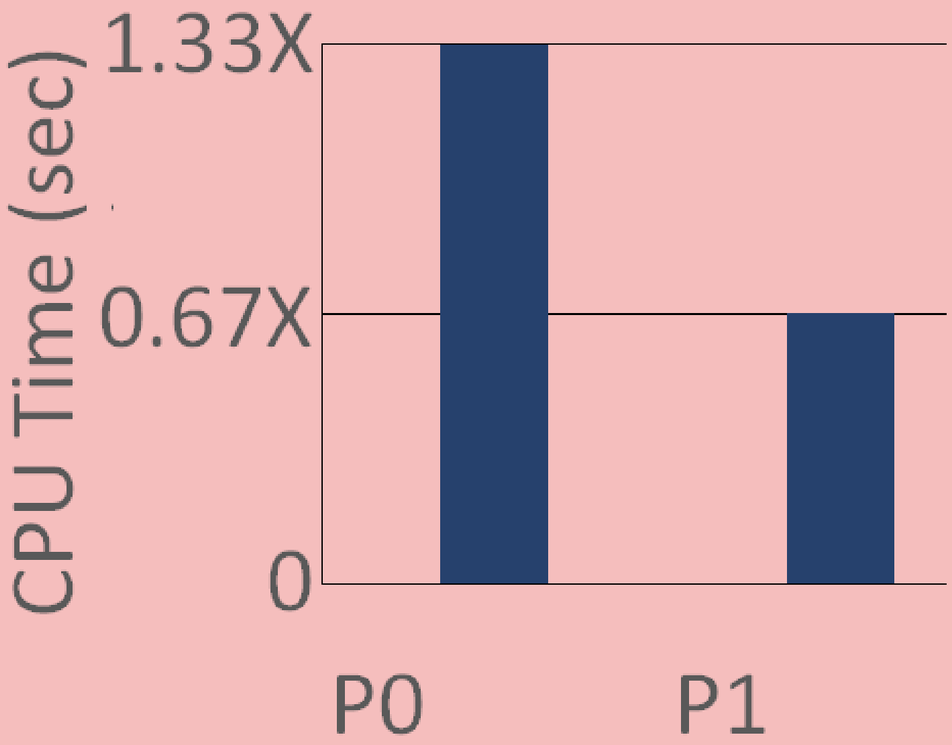
Locks determine the CPU allocation instead of the scheduler

When does scheduler subversion happen?

Two reasons

Different critical section lengths

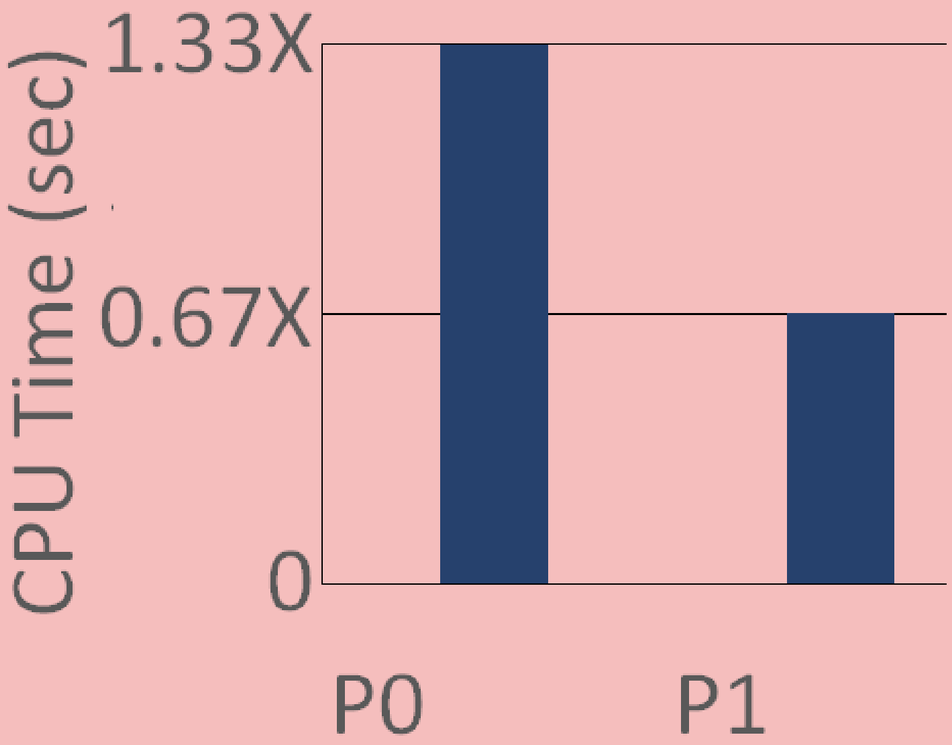
Different critical section lengths



Thread dwelling longer in critical section becomes dominant user of CPU

Majority locked run time

Majority locked run time



Threads have to spend a lot of time in the critical section and it is often a case [1, 2, 3, 4]

1. Lock–Unlock: Is That All? A Pragmatic Analysis of Locking in Software Systems. ACM Trans. Comput. Syst.,36(1), March 2019.
2. Remote Core Locking: Migrating Critical-Section Execution to Improve the Performance of Multithreaded Applications. USENIX ATC 2012
3. Understanding Manycore Scalability of File Systems, USENIX ATC 2016
4. Non-scalable locks are dangerous. Linux Symposium, 2012

Every concurrent system
can be prone to
scheduler subversion

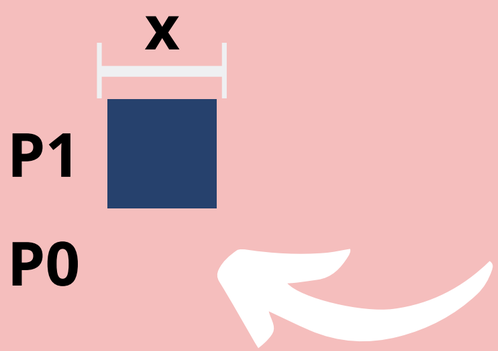
Remedy

Three design components

- Track lock usage of users

Three design components

Example

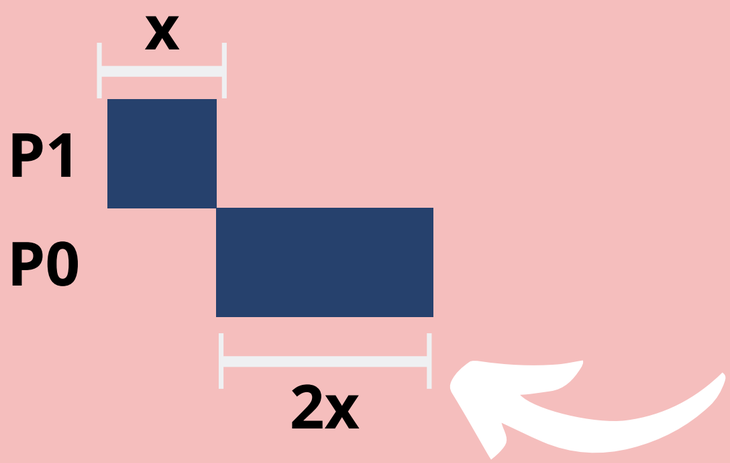


Track lock usage of users

Track the time spent by each process in critical section

Three design components

Example

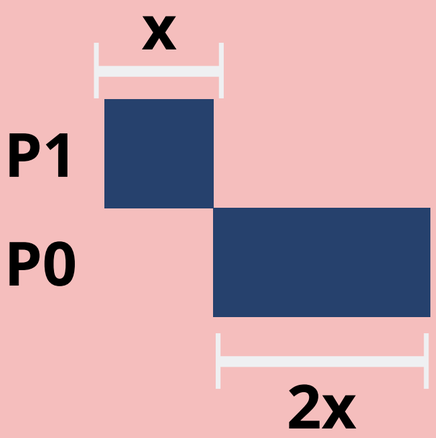


Track lock usage of users

Track the time spent by each process in critical section

Three design components

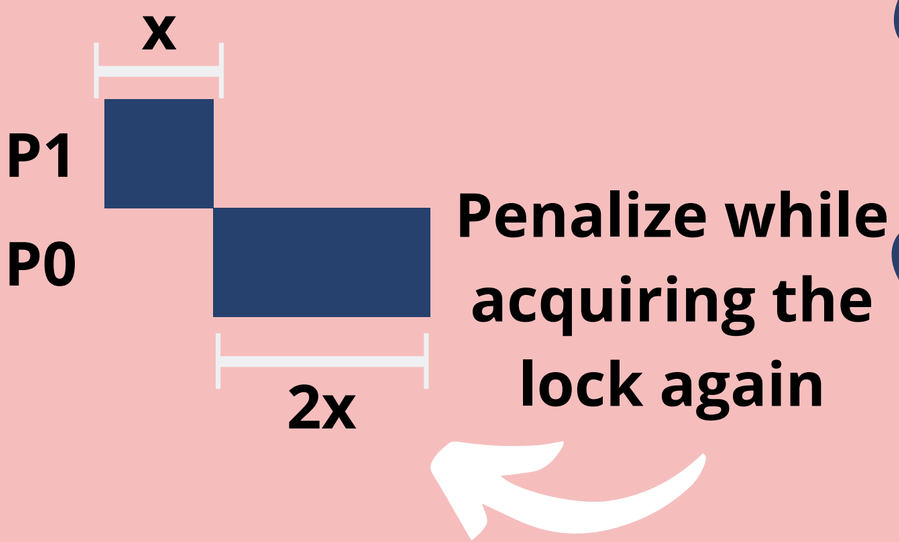
Example



- Track lock usage of users
- Penalize dominant users

Three design components

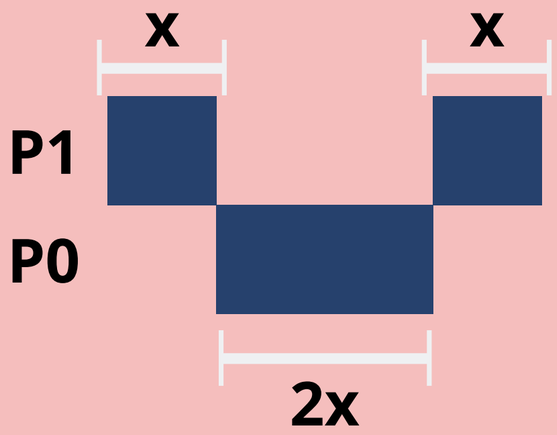
Example



- Track lock usage of users
- Penalize dominant users

Three design components

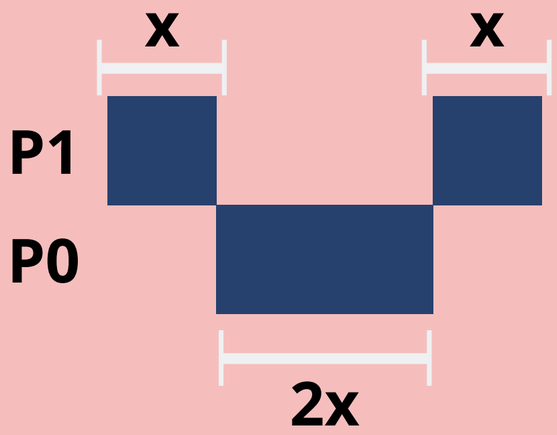
Example



- Track lock usage of users
- Penalize dominant users

Three design components

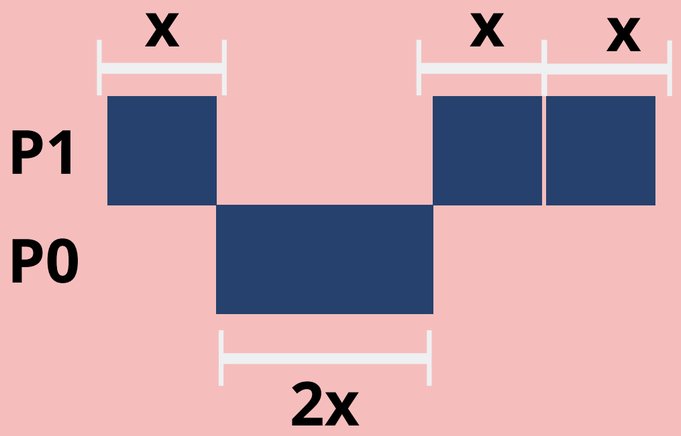
Example



- Track lock usage of users
- Penalize dominant users
- Provide appropriate opportunity to each user

Three design components

Example

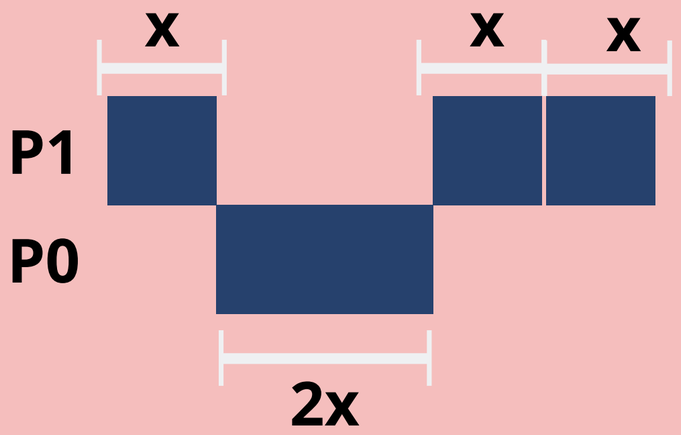


Penalizing P0 creates enough opportunity to let P1 acquire the lock multiple times

- Track lock usage of users
- Penalize dominant users
- Provide appropriate opportunity to each user

Three design components

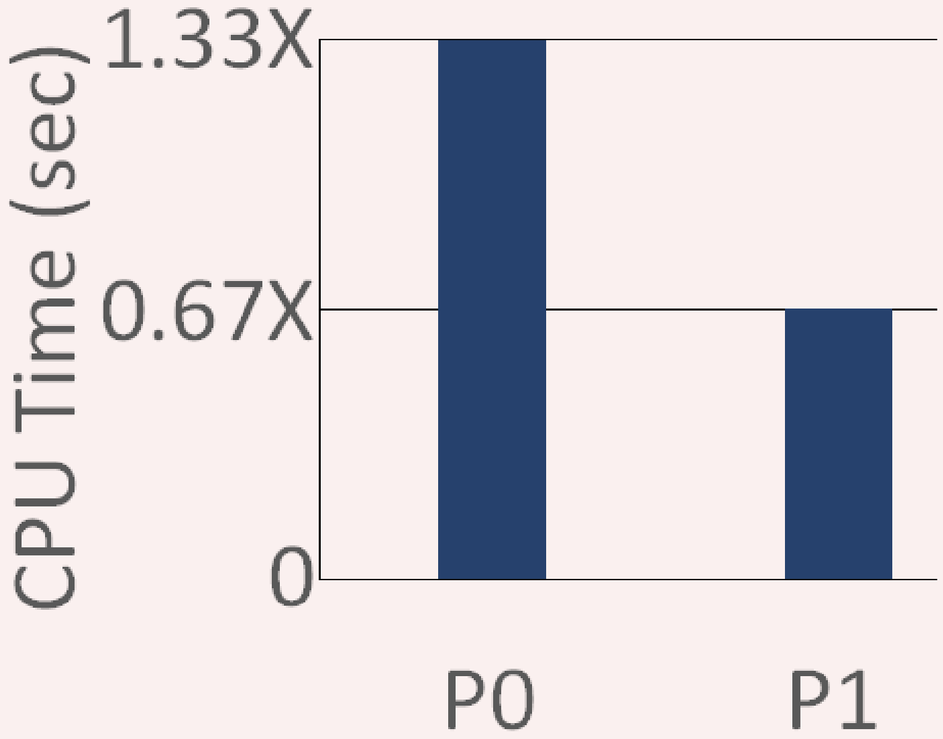
Example



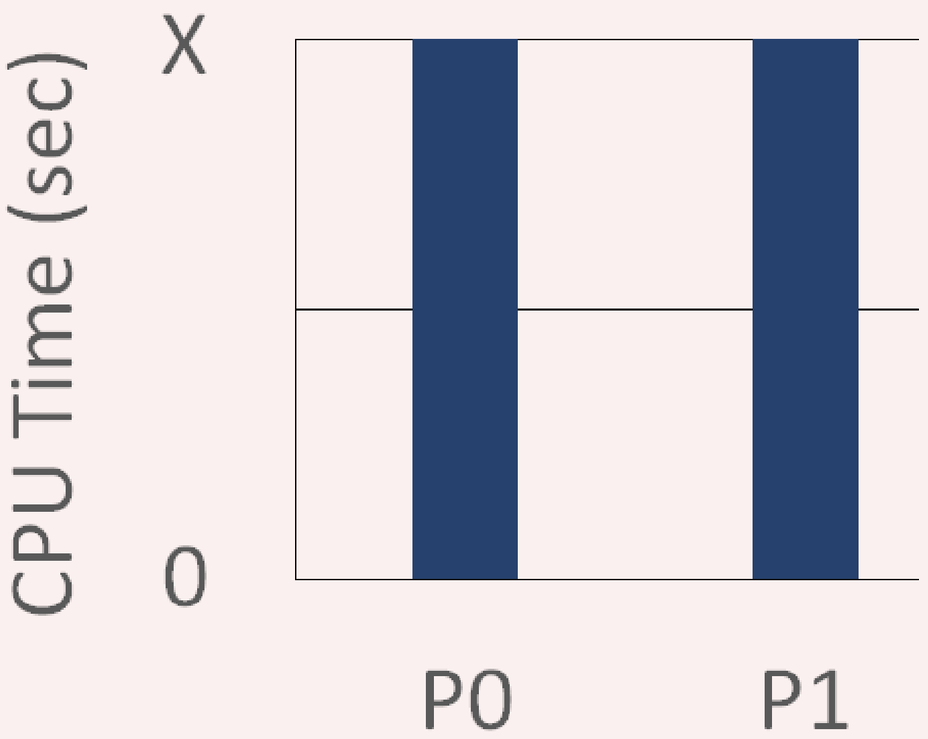
Opportunity depends on the scheduling goals

- Track lock usage of users
- Penalize dominant users
- Provide appropriate opportunity to each user

Scheduler-Cooperative Locks (SCL)



Ticket Lock



SCL

Conclusion

We introduce the scheduler subversion problem

We introduce scheduler-cooperative locks (SCL) to mitigate scheduler subversion

Thank you!!!
Queries??