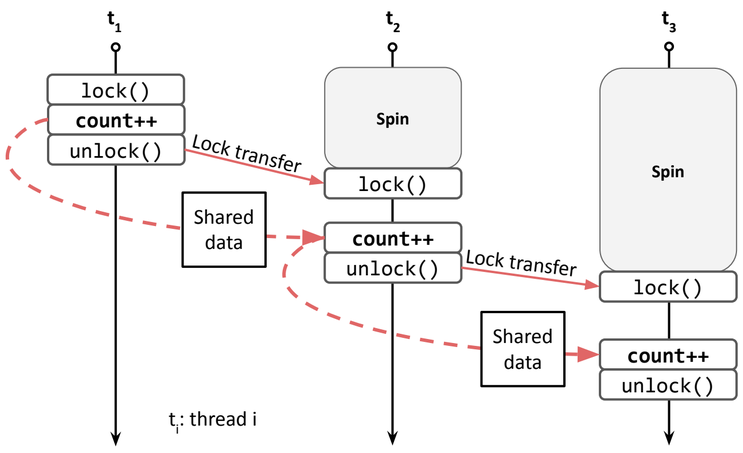
Ship your Critical Section Not Your Data: Enabling Transparent Delegation with TCLocks

Vishal Gupta Kumar Kartikeya Dwivedi Yugesh Kothari
Yueyang Pan Diyu Zhou Sanidhya Kashyap

Existing lock design: Either no locality of shared data or require application modification

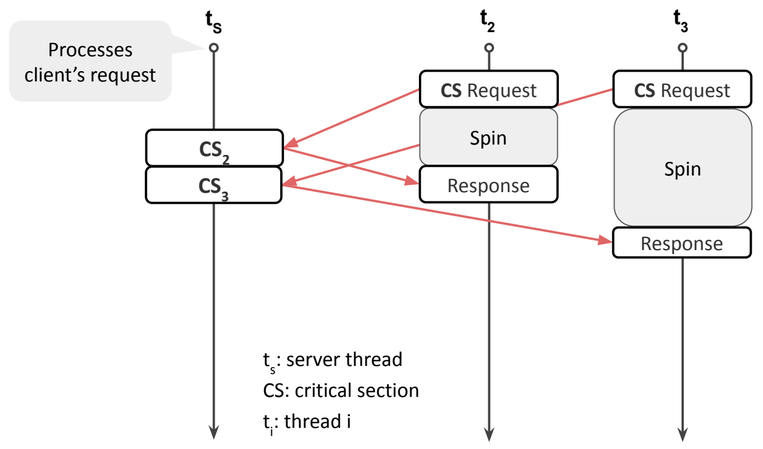
Traditional lock design: Move data to computation



Existing lock design:

- ❏ Use traditional lock/unlock APIs
- ❏ Require movement of shared data

Delegation-style lock design: Move computation to data

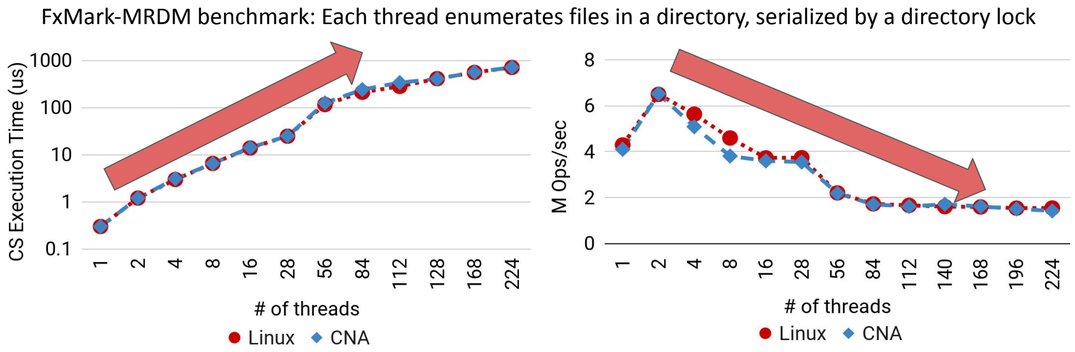


Existing lock design:

- ❏ Provide locality for shared data
- ❏ Require application rewrite

Traditional lock design: No locality of shared data

FxMark-MRDM benchmark: Each thread enumerates files in a directory, serialized by a directory lock

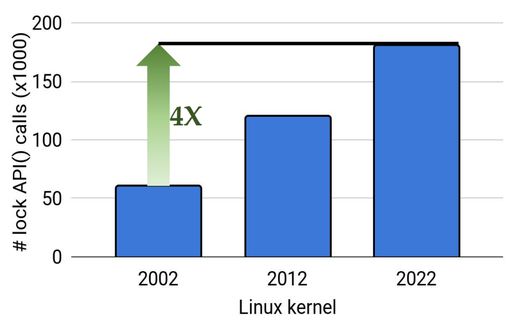


Shared Data Movement → CS Execution Time → Application performance

Delegation-style lock design: Requires application rewrite

```
lock()
count++
unlock()
```

```
void incr_func() =
    count++

send_req_to_server(&incr_func)
```

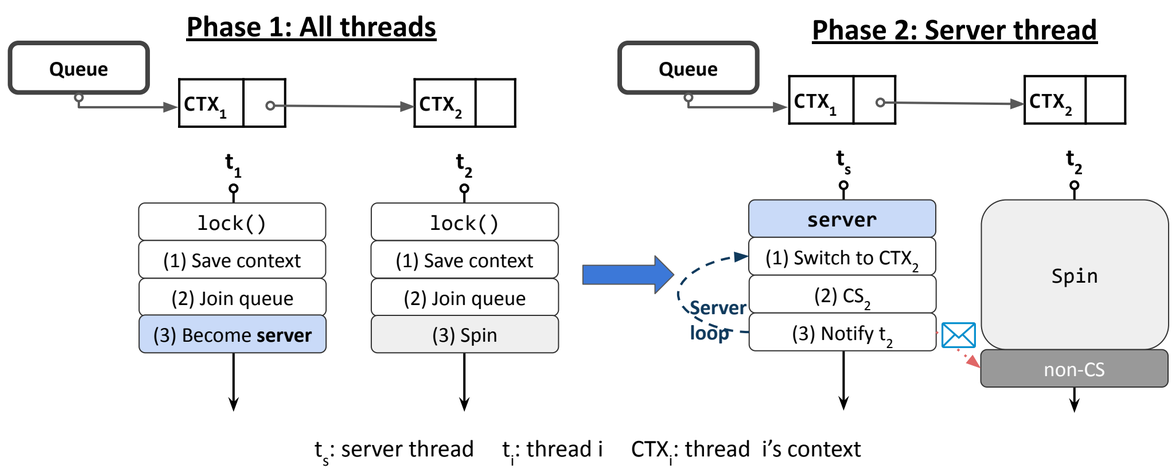


TCLocks: Provide minimal shared data movement without application modification

Key ideas

- **How to capture the thread's context?**
  - Instruction pointer + stack pointer + general-purpose registers
- **Where to capture the thread's context?**
  - Start and end of lock/unlock API
- **Does the waiter's thread modify its context?**
  - No, lock waiter busy waits to acquire the lock

Algorithm



Practical considerations

- Algorithmic support:
  - Delegation-based blocking lock
  - Phase-based reader-writer lock
  - NUMA-aware policy
- Lock usage:
  - Nested locking
  - OOO unlocking
  - Special execution contexts
  - per-CPU variables
- Performance optimization:
  - Reduced context-switch overhead
  - Stack prefetching

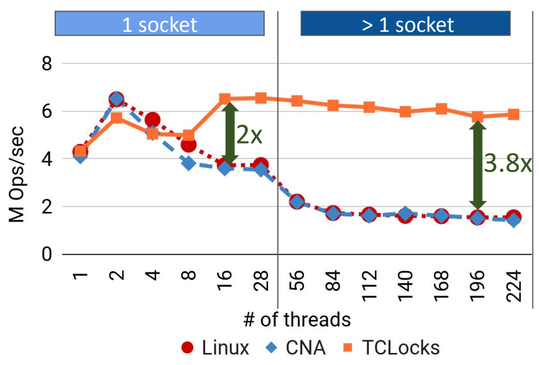
fs/dcache.c

```
// Update the dcache to reflect the
move of a file name
static void __d_move(struct dentry *dentry,
struct dentry *target, bool exchange) {
  ……..
  spin_lock(&target->d_parent->d_lock);
  spin_lock_nested(&old_parent->d_lock, 1);
  spin_lock_nested(&dentry->d_lock, 2);
  spin_lock_nested(&target->d_lock, 3);

  …….critical section……..

  spin_unlock(&target->d_parent->d_lock);
  spin_unlock(&old_parent->d_lock);
  spin_unlock(&target->d_lock);
  spin_unlock(&dentry->d_lock);
}
```

Evaluation

Kernel-space: FxMark

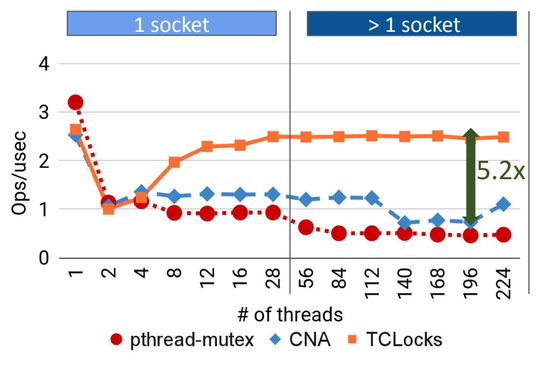
Benchmark: Each thread enumerates files in a directory, serialized by a directory lock



- **Within a socket:**
  - Minimal shared data movement

User-space: LevelDB

Benchmark: Key-value store, threads contend on the pthread lock



- **Across socket:**
  - NUMA-aware policy

TCLocks provides locality for shared data without application modification