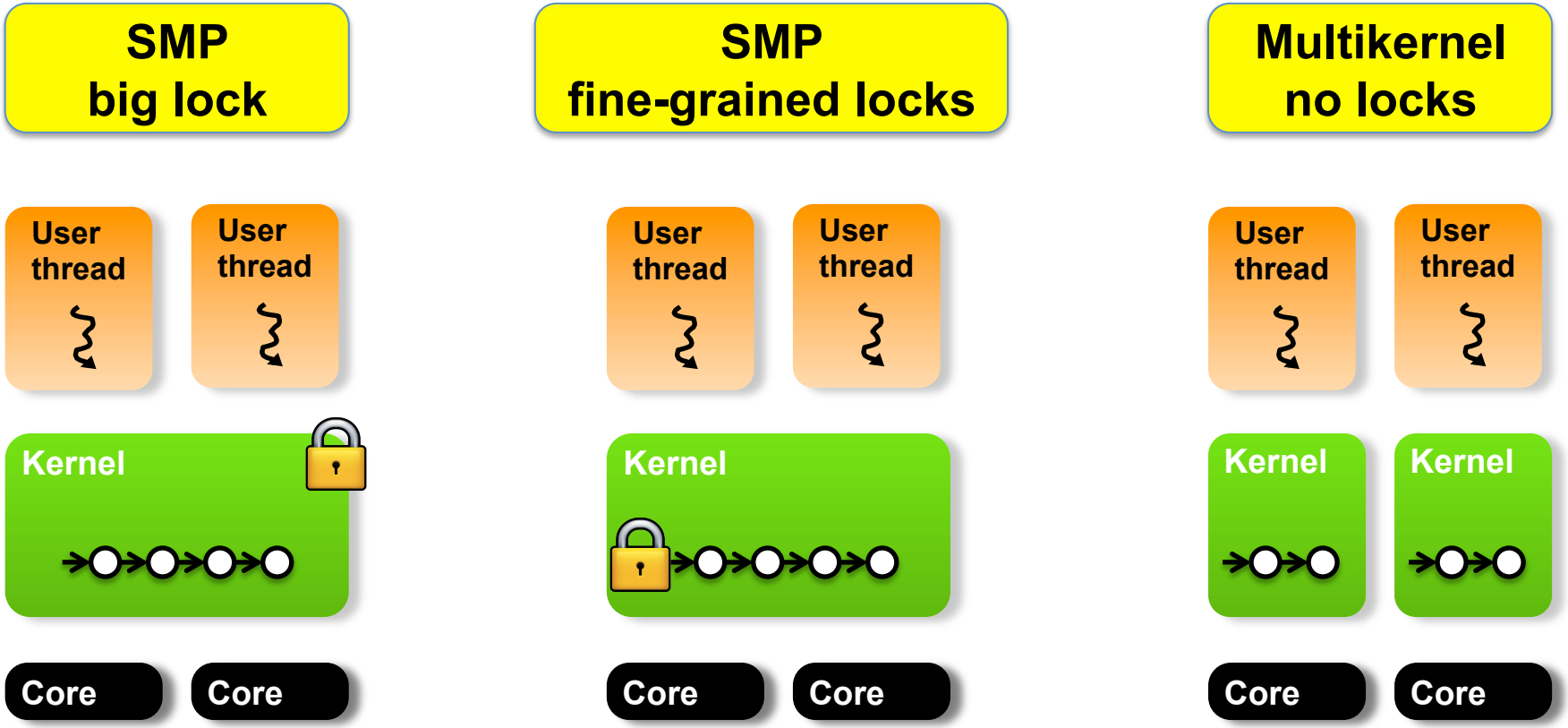

# For a Microkernel, a ***BIG*** Lock is *fine* !

Sean Peters, Adrian Danis, Kevin Elphinstone, *Gernot Heiser*

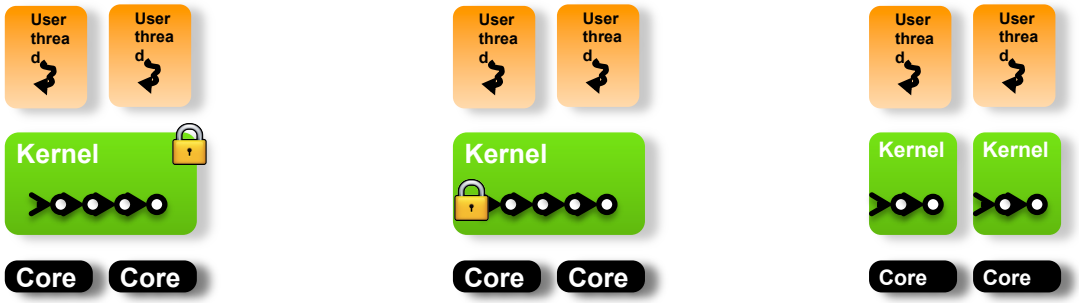
NICTA and UNSW, Sydney, Australia



# Approaches for Multicore Kernels



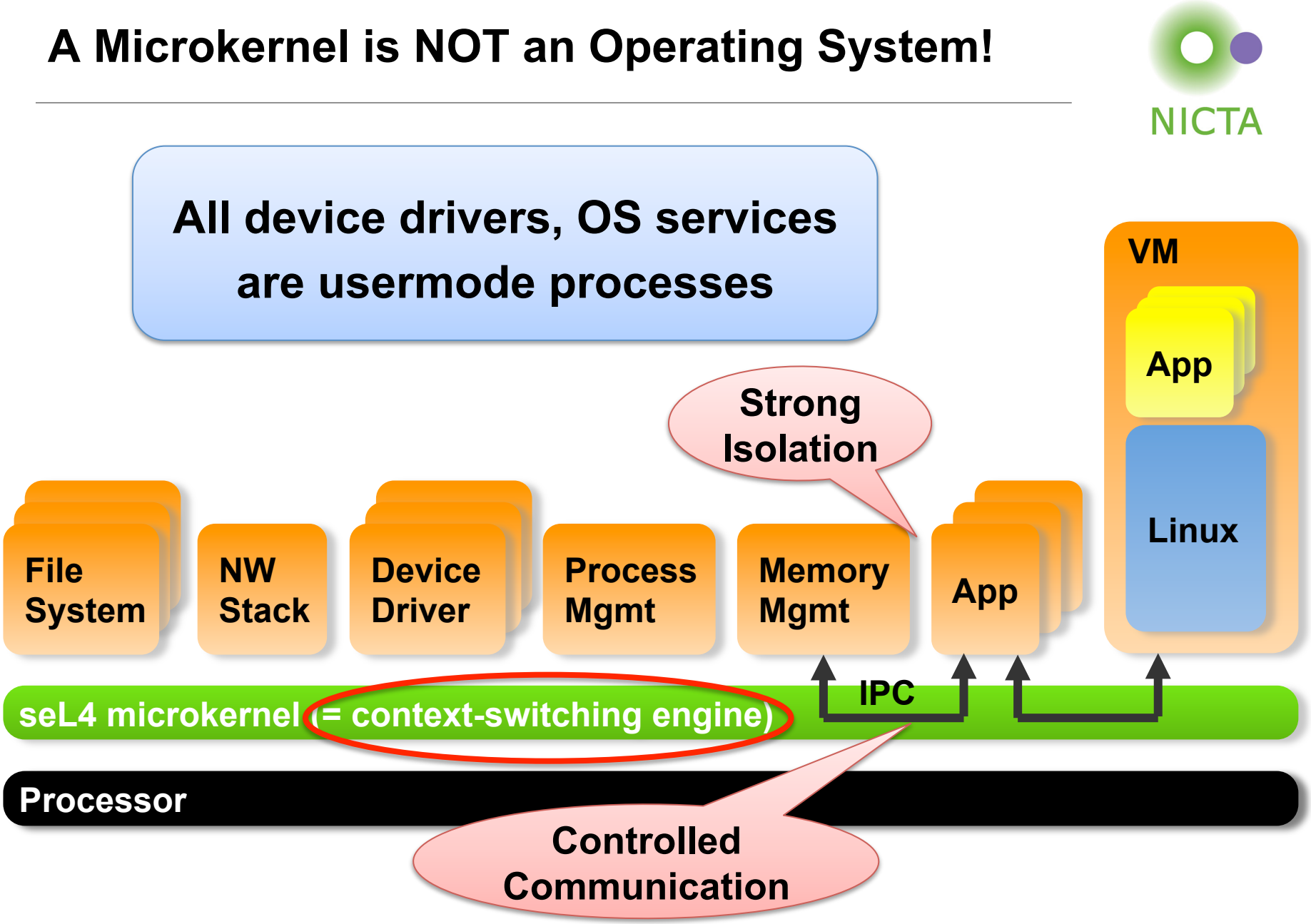
# Multicore Kernel Trade-Offs



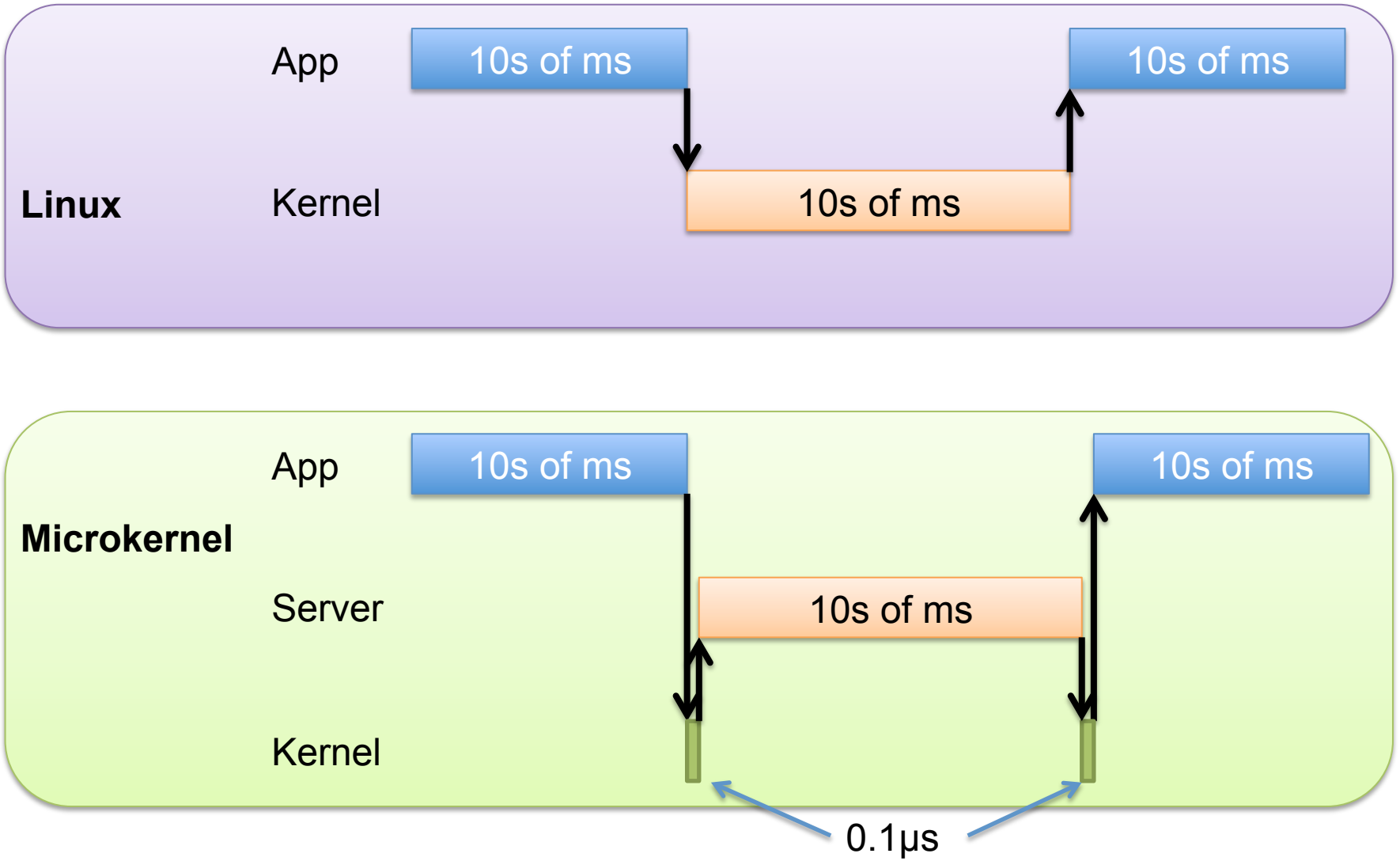

Really?

| Property | Big Lock | Fine-grained Locking | Multikernel |
|---|---|---|---|
| **Data structures** | shared | shared | distributed |
| **Scalability** | poor | good | excellent |
| **Concurrency in kernel** | zero | high | zero |
| **Kernel complexity** | low | high | low |
| **Resource management** | centralised | centralised | distributed |

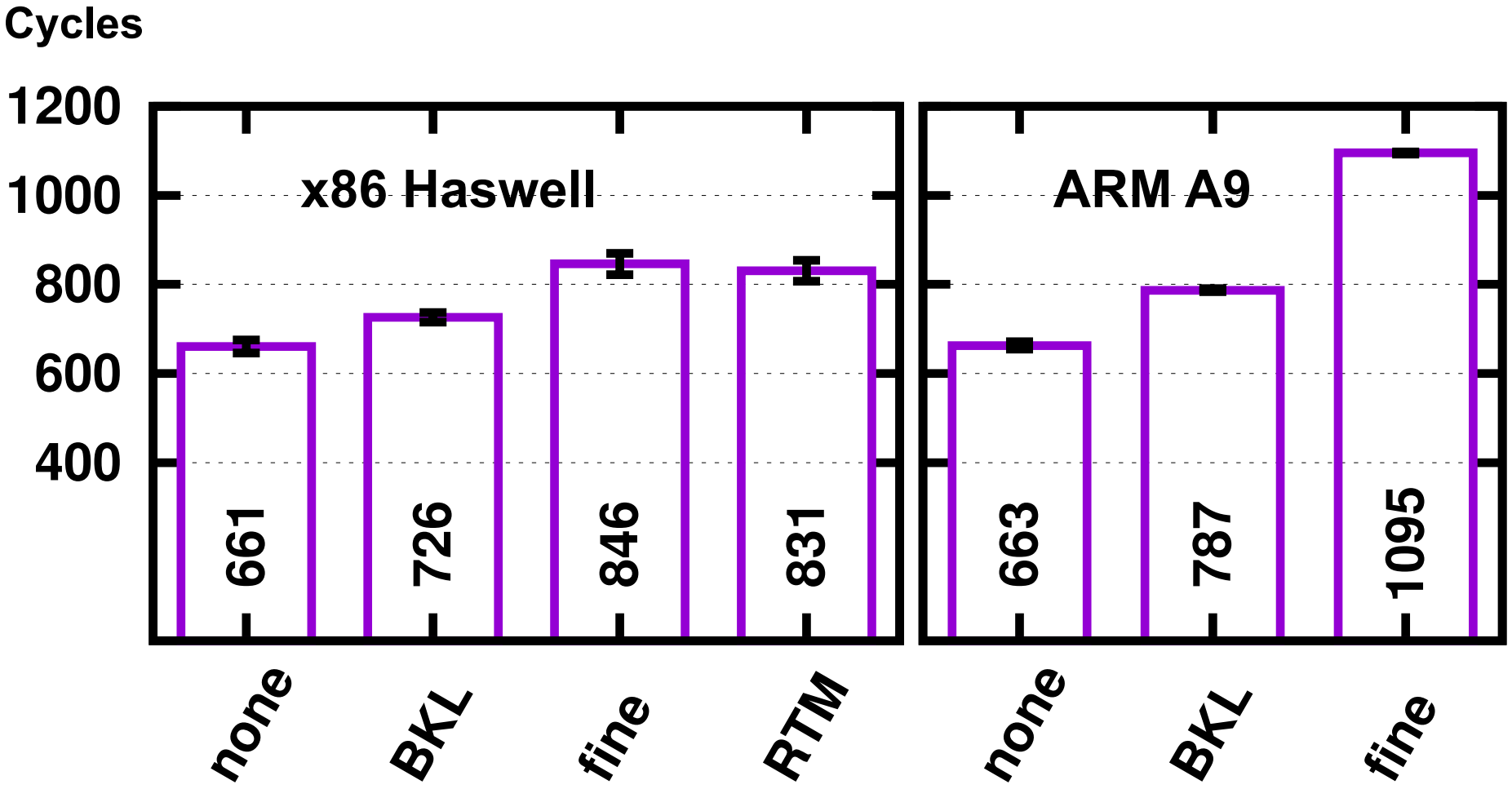
# A Microkernel is NOT an Operating System!



# Microkernel vs Linux Execution



# Cost of Locking: Round-Trip Intra-Core IPC

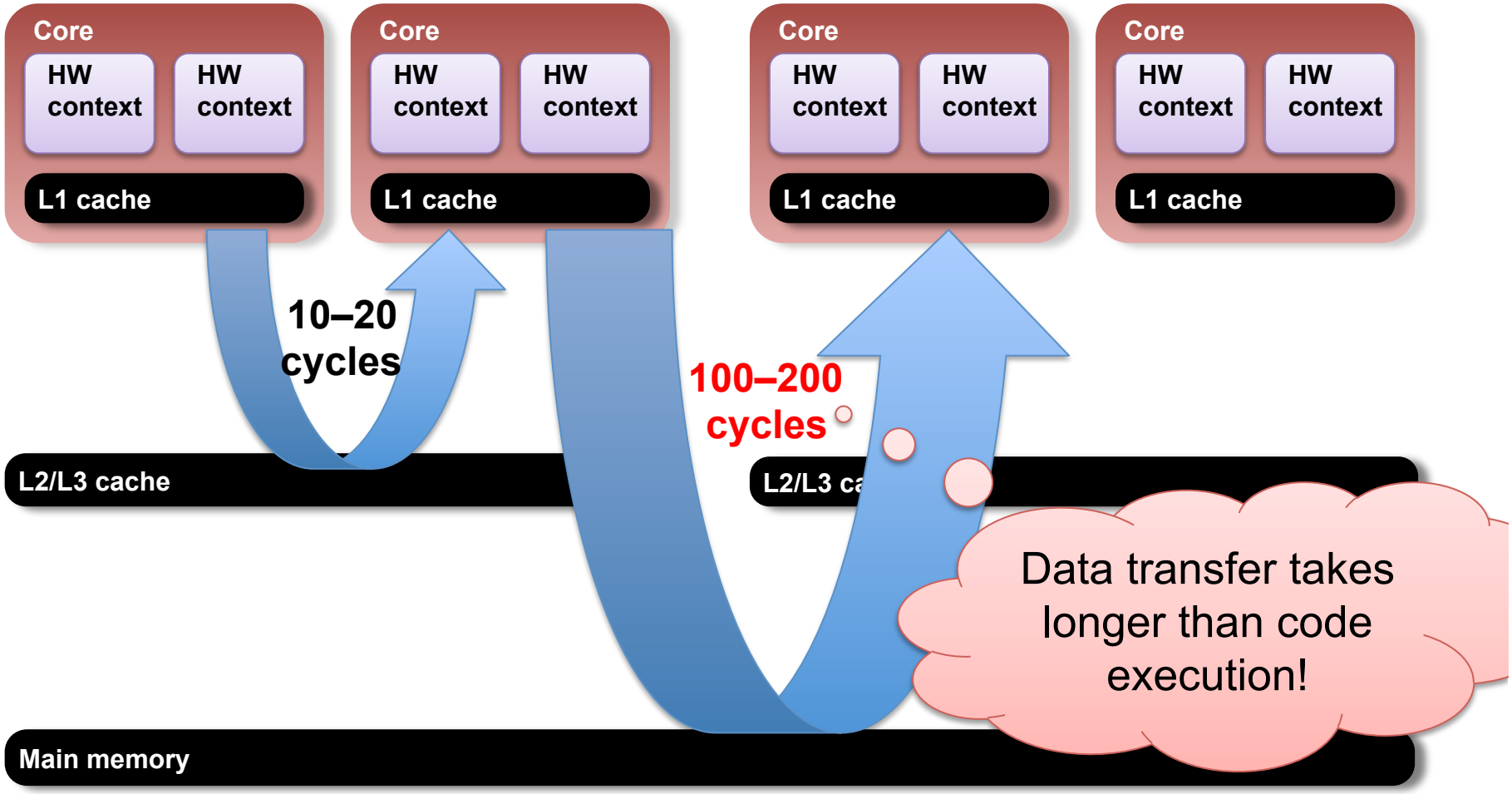


# Microkernel Multicore Design


**Assertion 1: Minimise locks, not locked code**

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to fine- to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short ⇒ contention is low

# Cache Line Migration Latencies



# Microkernel Multicore Design

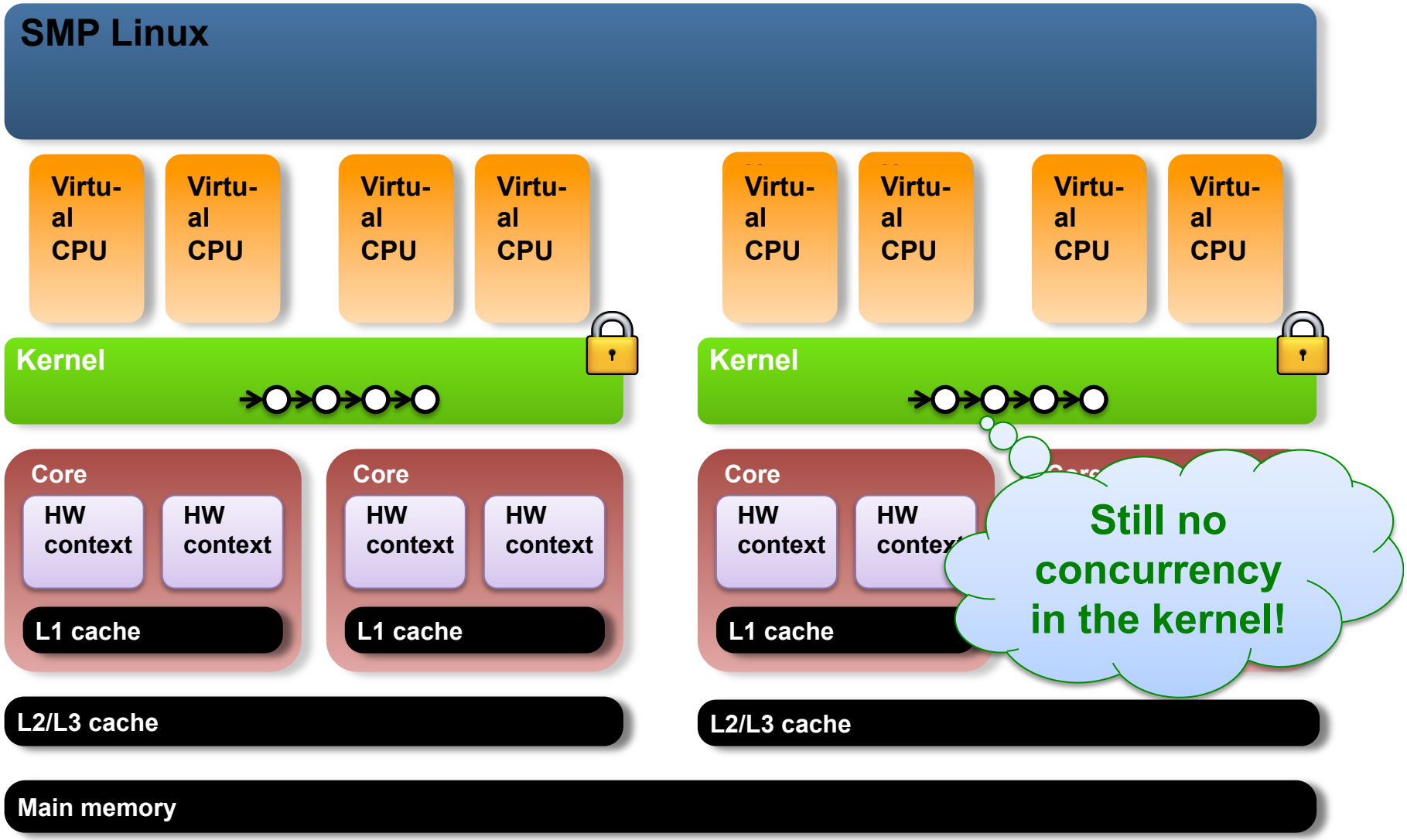

**Assertion 1: Minimise locks, not locked code**

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short ⇒ contention is low

**Assertion 2: Don’t share mirokernel data without shared cache**

- Migrating only a few cache lines takes longer than rest of syscall

# seL4 Multicore Design: Clustered Multikernel



# Microkernel Multicore Design


**Assertion 1: Minimise locks, not locked code**

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short ⇒ contention is low

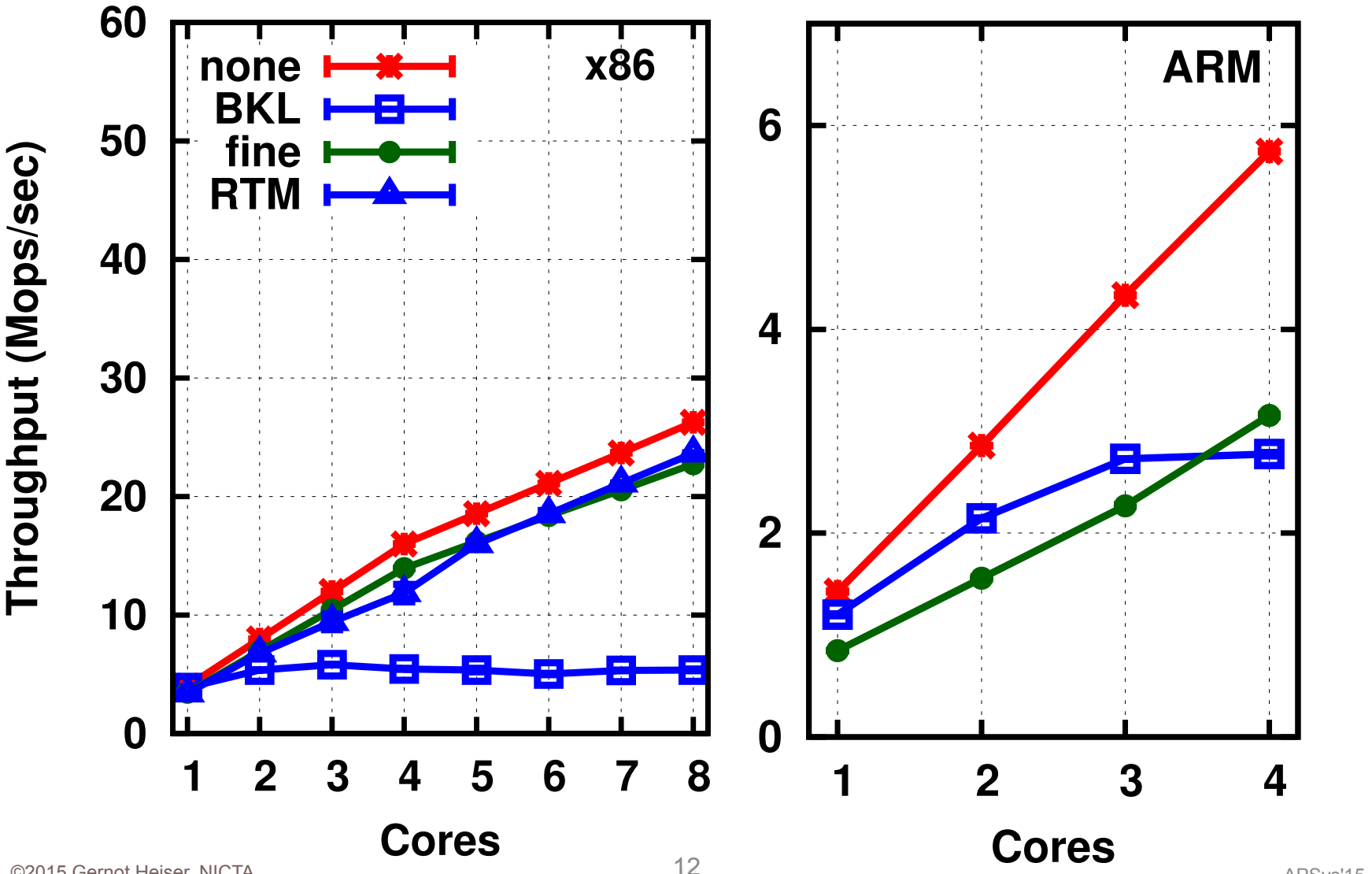
**Assertion 2: Don't share mirokernel data without shared cache**

- Migrating only a few cache lines takes longer than rest of syscall

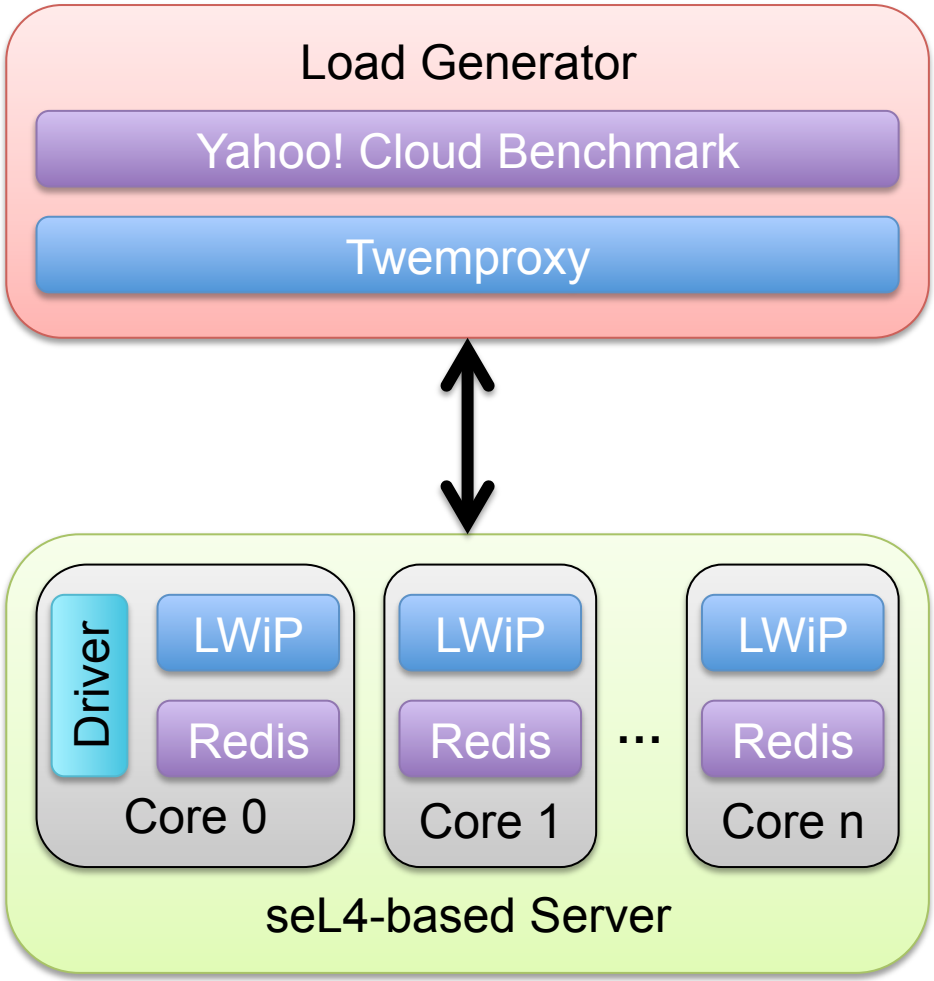
**Assertion 3: Big lock will perform for closely-coupled cores**

- Shared caches presently have moderate core counts
- Big lock in a *well-designed* microkernel will scale there

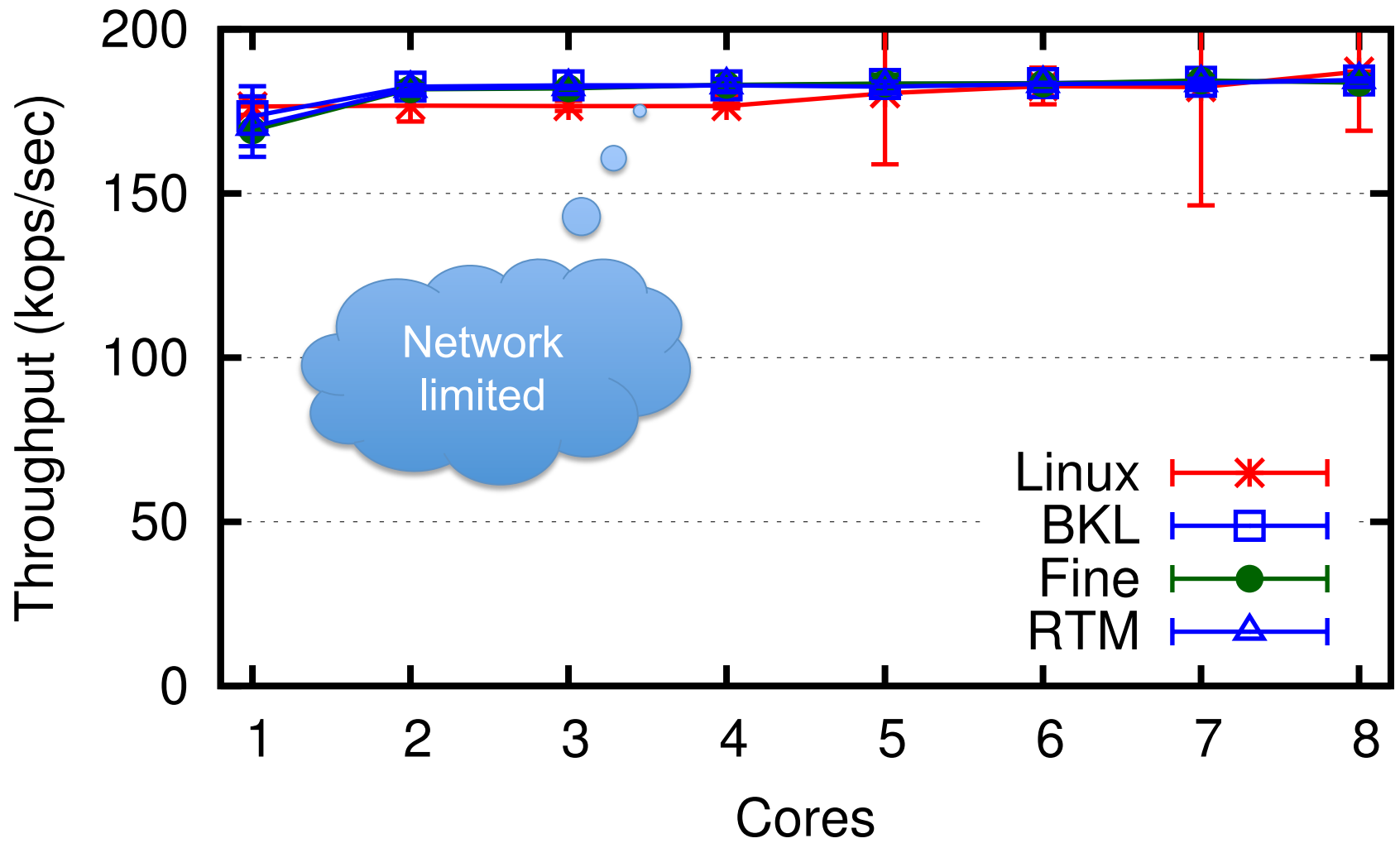
# Pathological Case: Intra-Core IPC Ping-Pong



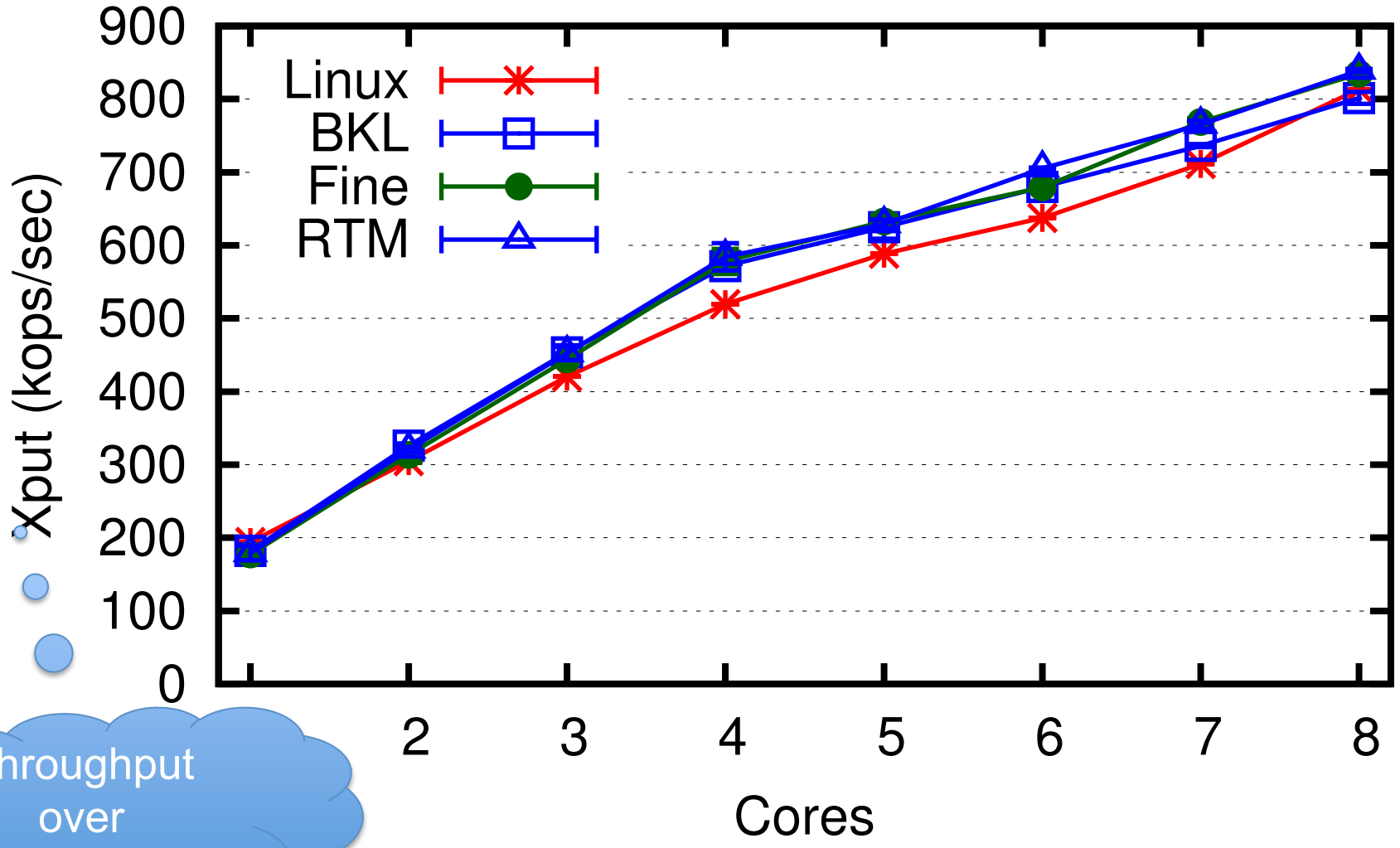
# Realistic Case: Syscall-Intensive Macrobenchmark



# Realistic Case: Syscall-Intensive Macrobenchmark



# Macrobenchark: Correct for Load




Throughput over CPU utilization

# Microkernel Multicore Design


**Assertion 1: Minimise locks, not locked code**

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short ⇒ contention is low

**Assertion 2: Don't share mirokernel data without shared cache**

- Migrating only a few cache lines takes longer than rest of syscall

**Assertion 3: Big lock will perform for closely-couple cores**

- Shared caches presently have moderate core counts
- Big lock in a *well-designed* microkernel will scale there
- Seems to work fine for 8 cores, should still work for 16

# Summary


1. Big lock fine-grained enough for a *well-designed* microkernel
   - Short system calls ⇒ low contention
   - Big lock performs as well as fine-grained for closely-coupled cores
   - Complexity of fine-grained locking is not worth it!

2. Should not even *try* to share a kernel without shared caches

**http://seL4.systems**

**gernot@nicta.com.au**

**http://microkerneldude.wordpress.com**

**@GernotHeiser**