# Rethinking seL4 Multicore Architecture
# SMP vs. Multikernel

Guangtao Zhu, Rihui Wu, Julia Vassiliki,
Peter Chubb, and Gernot Heiser
**UNSW Sydney**

## Abstract

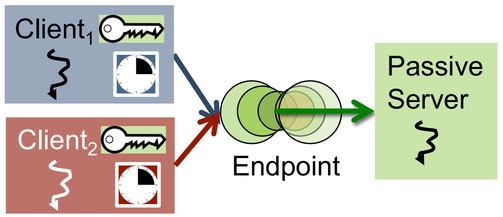
Peters et al. [2015] claimed a big kernel lock (BKL) is fine for a fast microkernel on a few cores sharing cache. We revisit this design issue, comparing the big lock on such a system to a Multikernel (MK) version, and find that MK outperforms BKL on all IPC operations.

### Breakdown Analysis of SMP Cost

• Round-trip IPC latency (cycles)

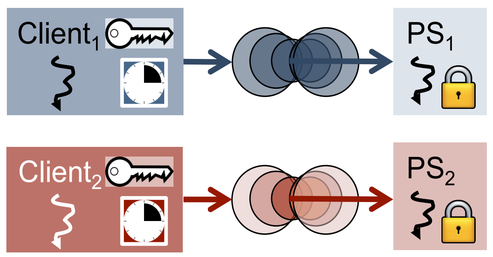
| Mode | UniCore (UC) | BKL | UC + BKL | SMP orig. | SMP optim. |
|---|---|---|---|---|---|
| EL1 | 643 | 89 | 732 | 877 | 783 |
| EL2 | 671 | 148 | 819 | 1091 | 921 |

## 1. Passive Server on SMP vs MK



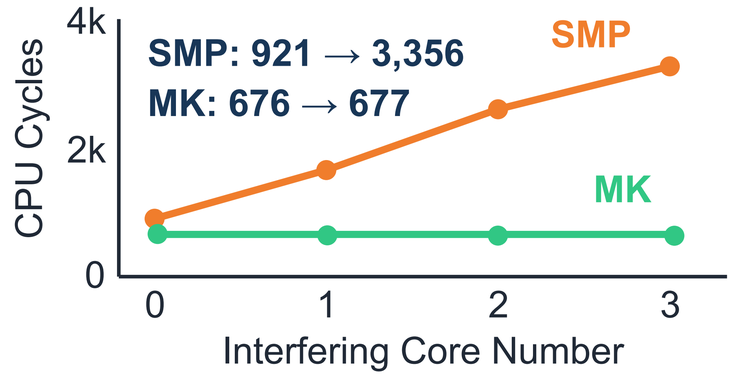

**SMP**
**migrating server**

**MK**
**core-local servers**
- Identical page-table structures
- Synchronise with a user-level lock




## 2. True Kernel Parallelism on MK

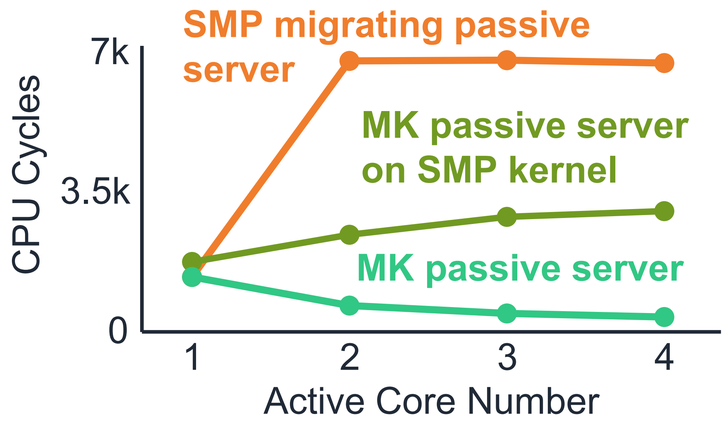
Local IPC ping-pong pair runs beside 0–3 interfering cores using the same setup.




**MK stays flat; SMP latency rises**

## 3. Average Server Invocation Latency

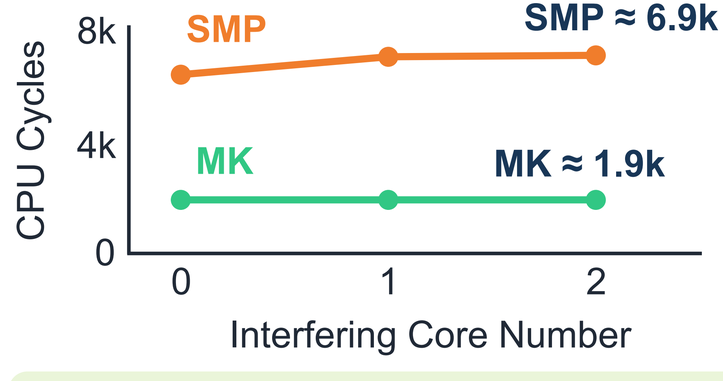
Clients from 1~4 cores invoke the same passive server for 1,000,000 rounds in sum.




**MK approaches perfect scaling**

## 4. Cross-core Notifications: > 3× Faster

The latency of cross-core Notification ping-pong with 0-2 interfering cores.



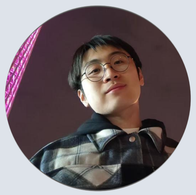

**MK is 3× SMP faster without interference**

## What We Learn

- MK local IPC is the same as UC
- Contention widens the MK lead
- MK is more minimal as kernel complexity is moved up
- Cost: replicated kernel objects (e.g., memory, page-table)

Github repo QR Code

I am not at the summit. Gernot will present this for me instead and if he is not in front on our poster come and find him to talk about our work.



**Guangtao Zhu**
UNSW Sydney