Understanding Microquanta Process Scheduling for Cloud Applications

Erfan Sharafzadeh, Alireza Sanaee, Peng Huang, Gianni Antichi, Soudeh Ghorbani

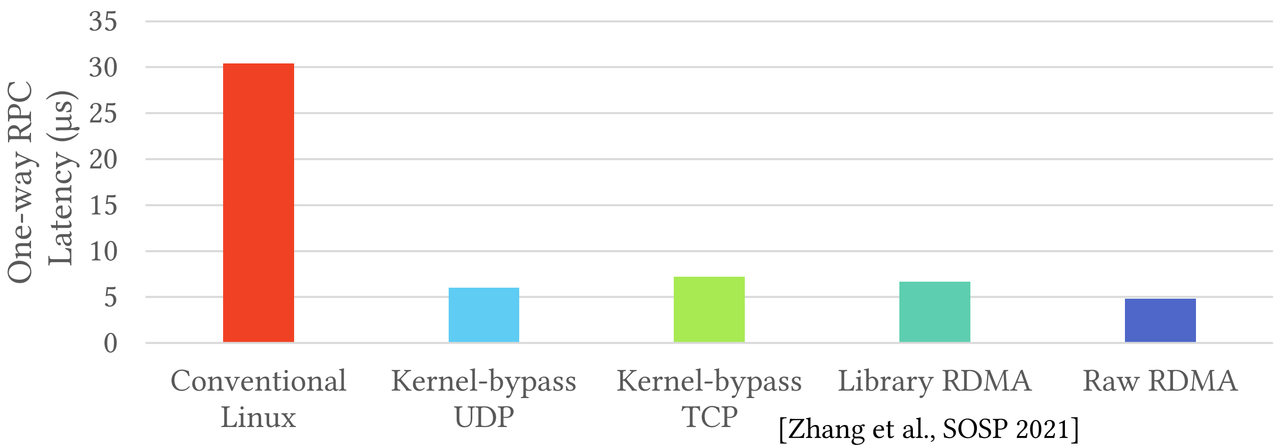
Johns Hopkins University

Queen Mary University of London

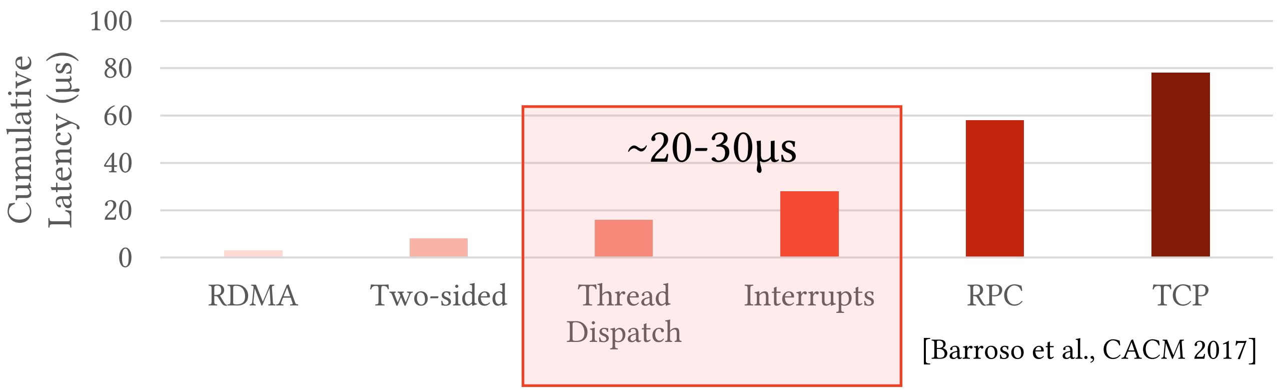
December 2022

The Need for Low Latency in Data Centers

- The call for µs-scale and ns-scale processing
- Emerging userspace networking runtimes
- **Thread-dispatch** and **interrupt handling** are culprits!



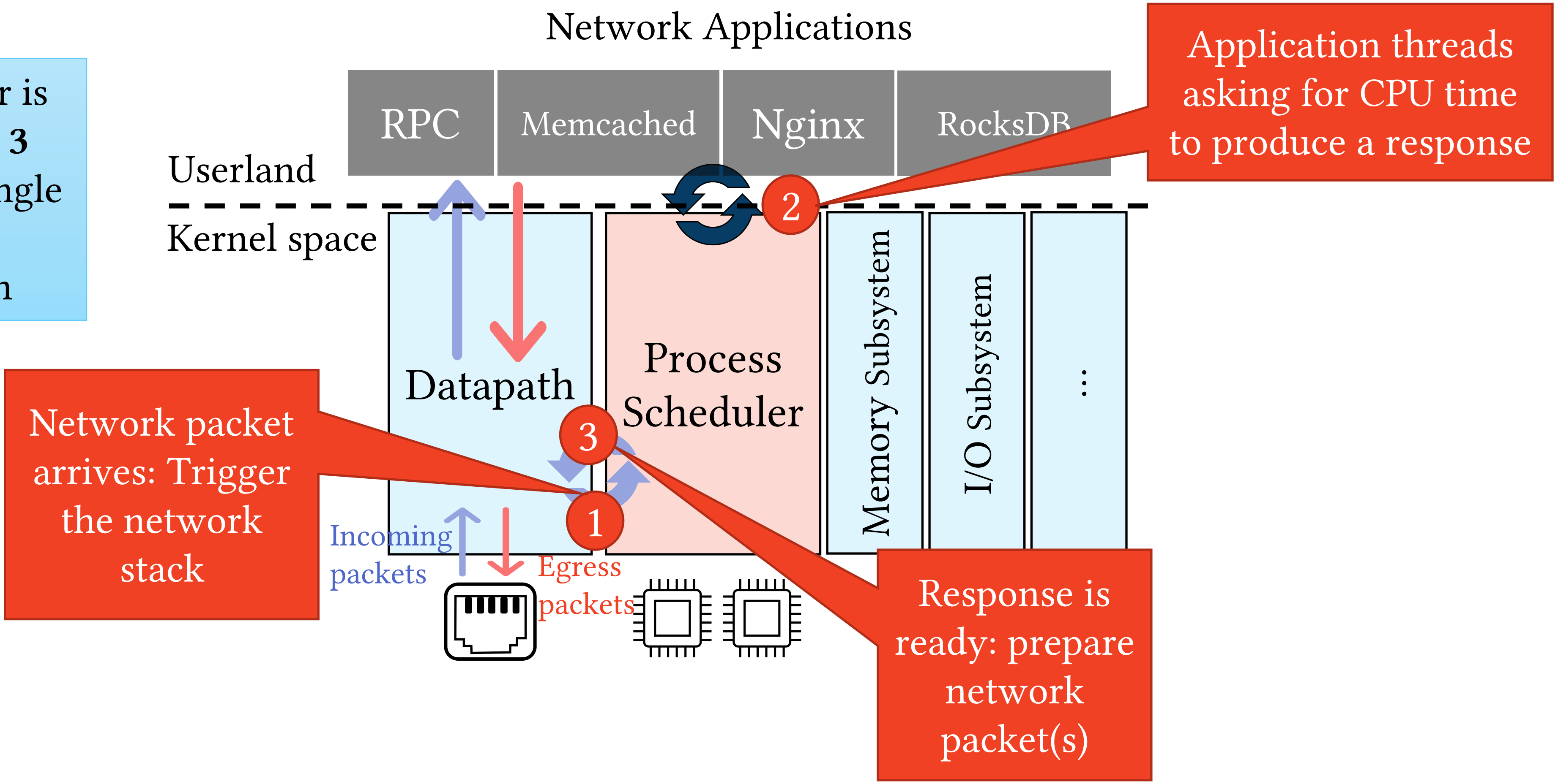
[Zhang et al., SOSP 2021]



[Barroso et al., CACM 2017]

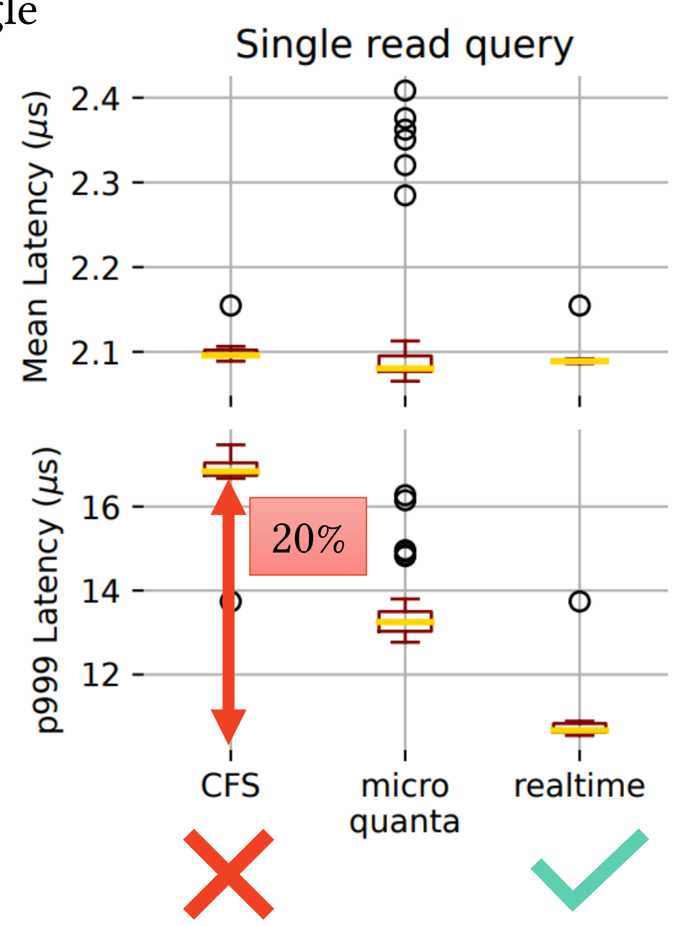
Process Scheduling Involved Everywhere!

Process scheduler is invoked at least **3 times** during a single network communication



Conventional Linux Schedulers Falling Short

- Running **RocksDB** benchmark on a single machine under three process schedulers

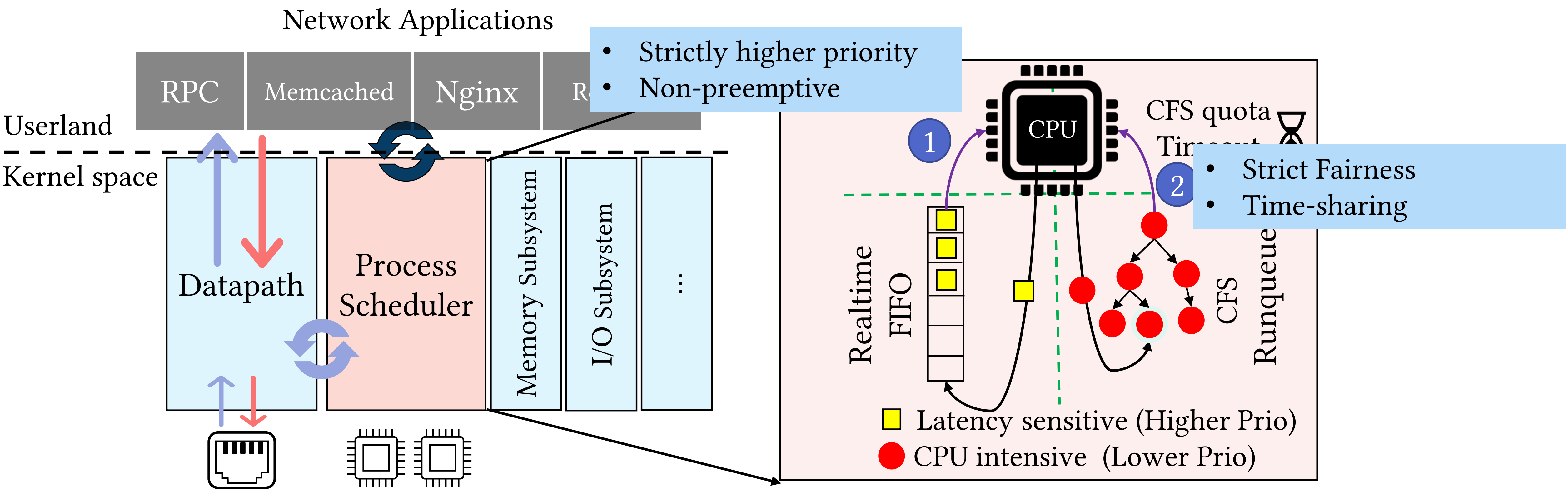


Microquanta holds a middle-ground but raises its own issues!

Non-skewed workloads can benefit from Realtime scheduling by minimizing the **interference**!

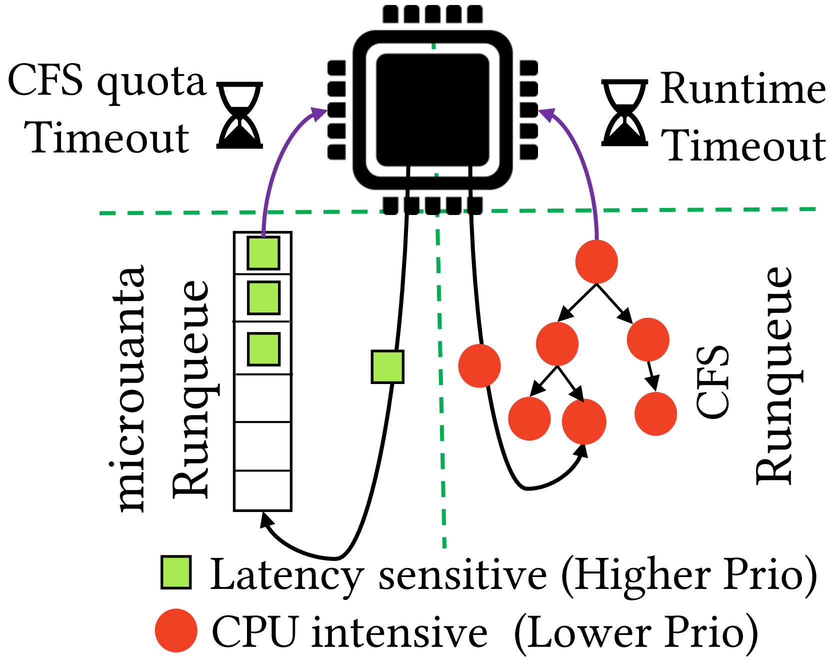
Non-preemptive realtime scheduling is unfit for skewed workloads due to **HoL blocking**!

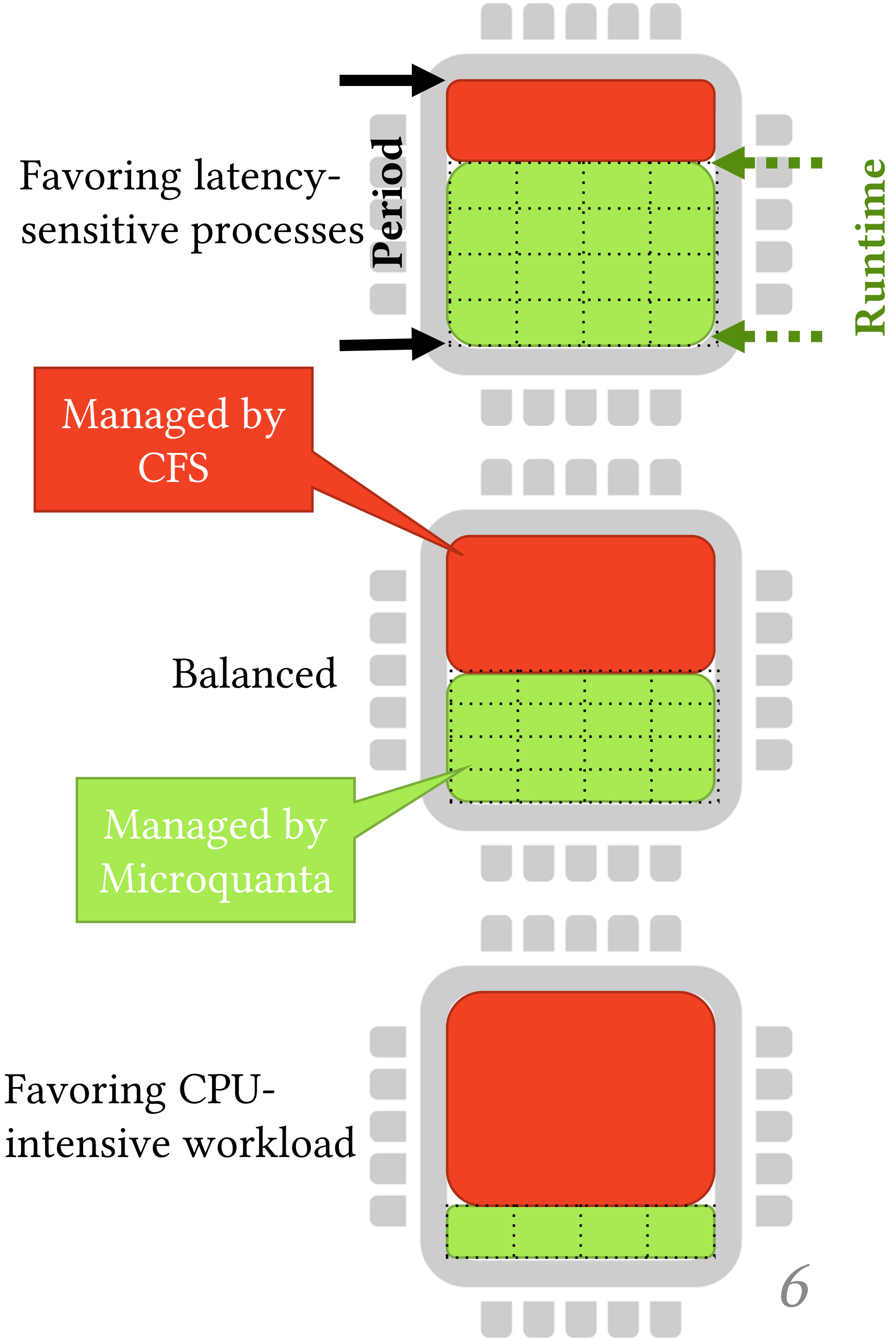
Introducing Three Representative Schedulers



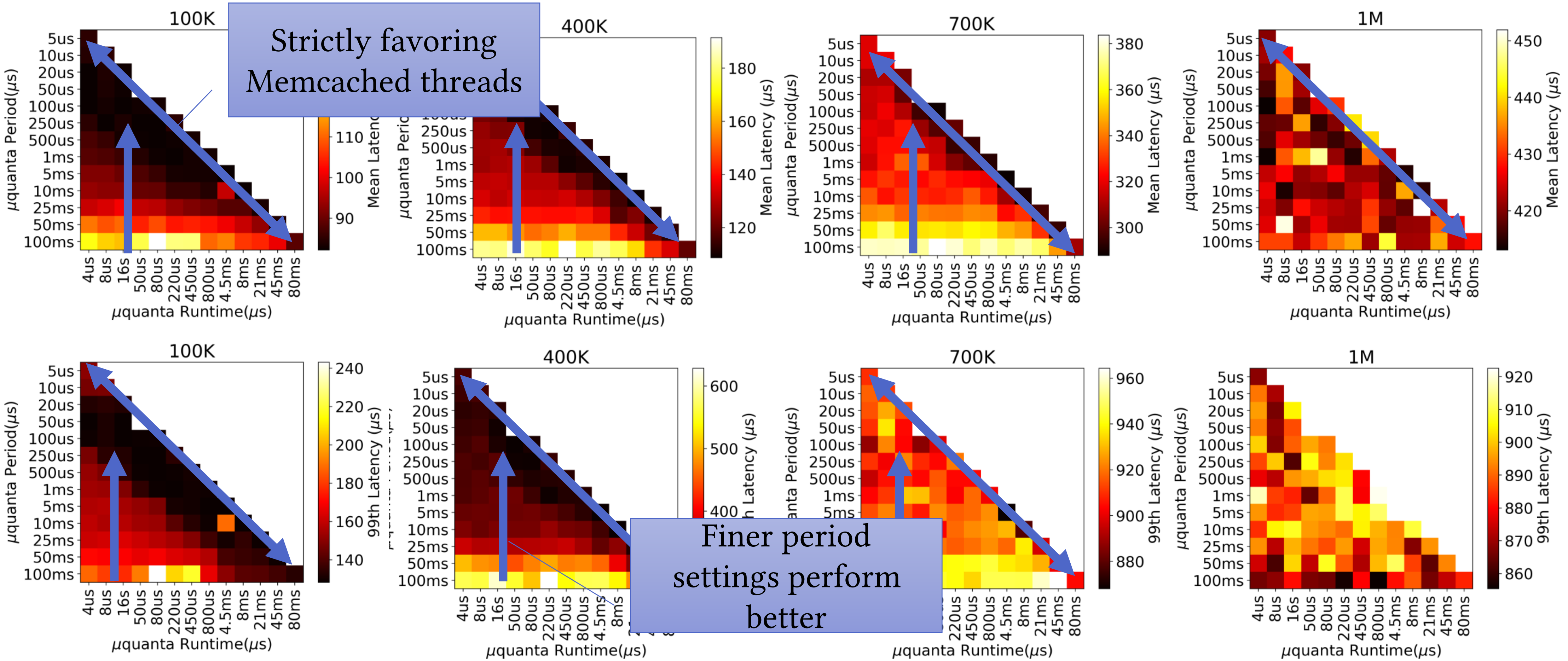
Microquanta Scheduling

- Per-CPU FIFO queues
- Microsecond-scale scheduling between processes
- Tunable CPU allocation via **Runtime** and **Period** Parameters



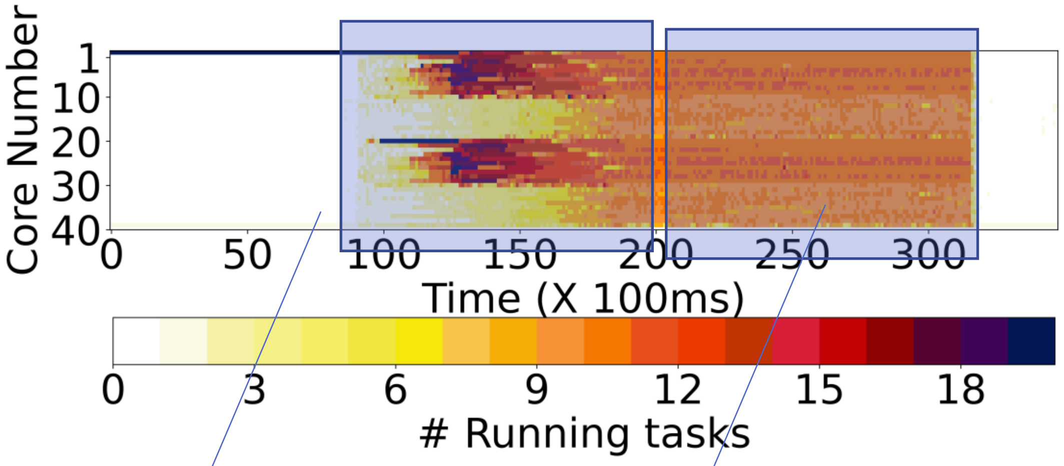


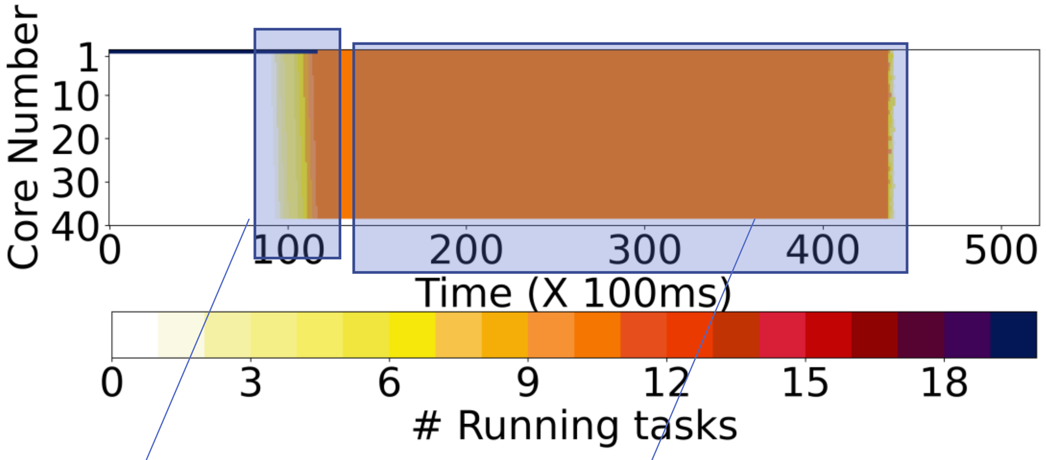
Impact of Microquanta Parameter Setting on Application Performance



Microquanta and Fast Load-Balancing

- 500 benchmark threads pinned to core #1 -> Released on 10th second
- The schedulers start distributing threads





10-second convergence time for CFS!

Non uniform load distribution, hot zones still exist

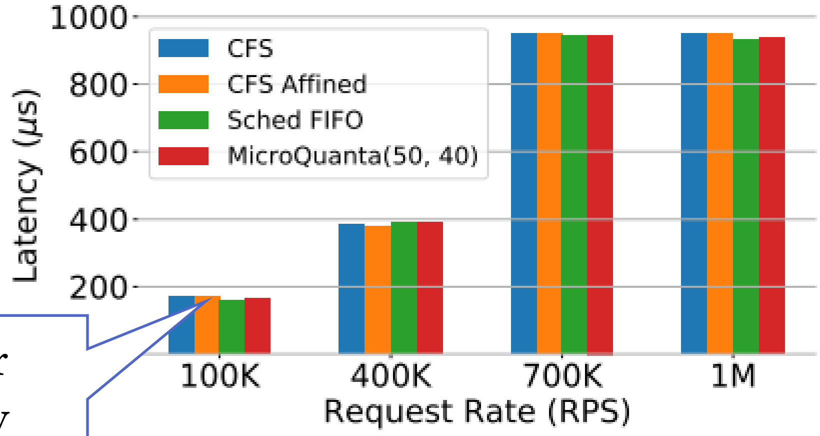
Very fast convergence for Microquanta ~1s

Uniform load-distribution

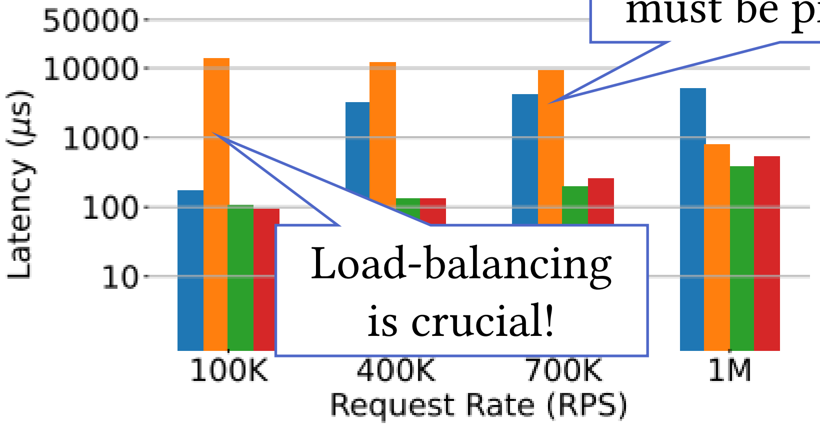
Application Performance Comparison

Network application must be prioritized!

Uncontested



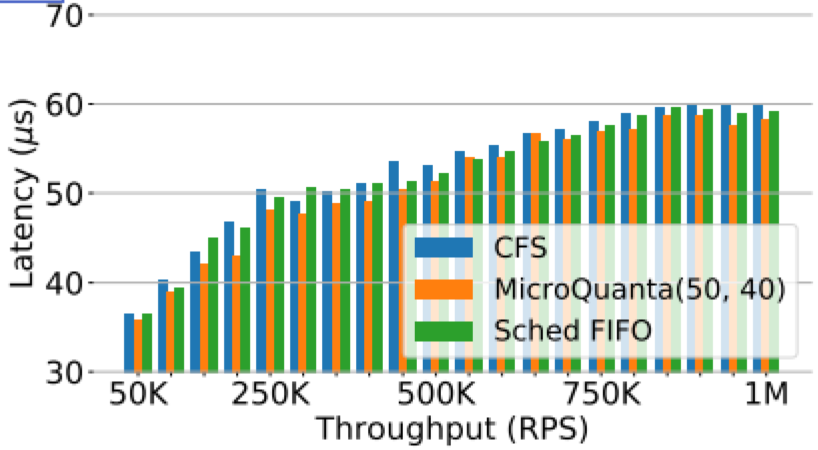
Contested



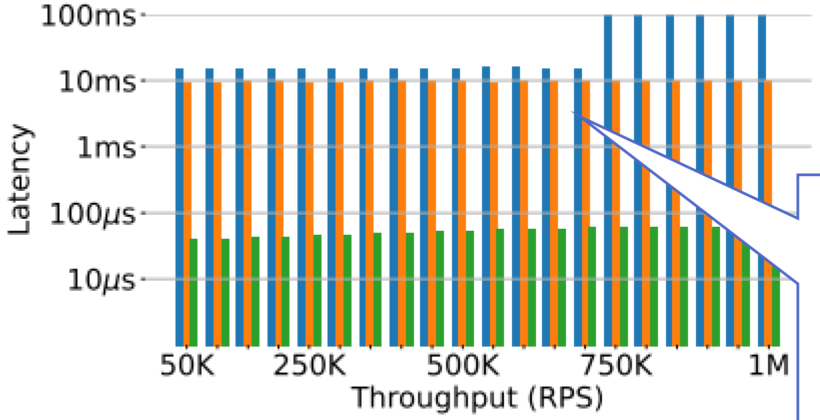
Load-balancing is crucial!

Similar latency performance

Uncontested



Contested



Microquanta cannot benefit the busy-polling threads of IO-kernel

The Future of Process Scheduling

- Linux process scheduling is challenged by skewed workloads
- Parameter-based scheduling faces tuning issues
- Design space of process schedulers
  - Schedulers that can **learn and adapt** to workload changes
  - Schedulers that are **tied to applications logic**
    - Kernel-bypass runtimes (Shinjuku, Caladan)
    - Userspace thread-management (Arachne)
    - In-application scheduling (Ghost, Peafowl)

Microquanta Kernel Repository: https://github.com/erfan111/linux_uquanta

Contact: erfan@cs.jhu.edu