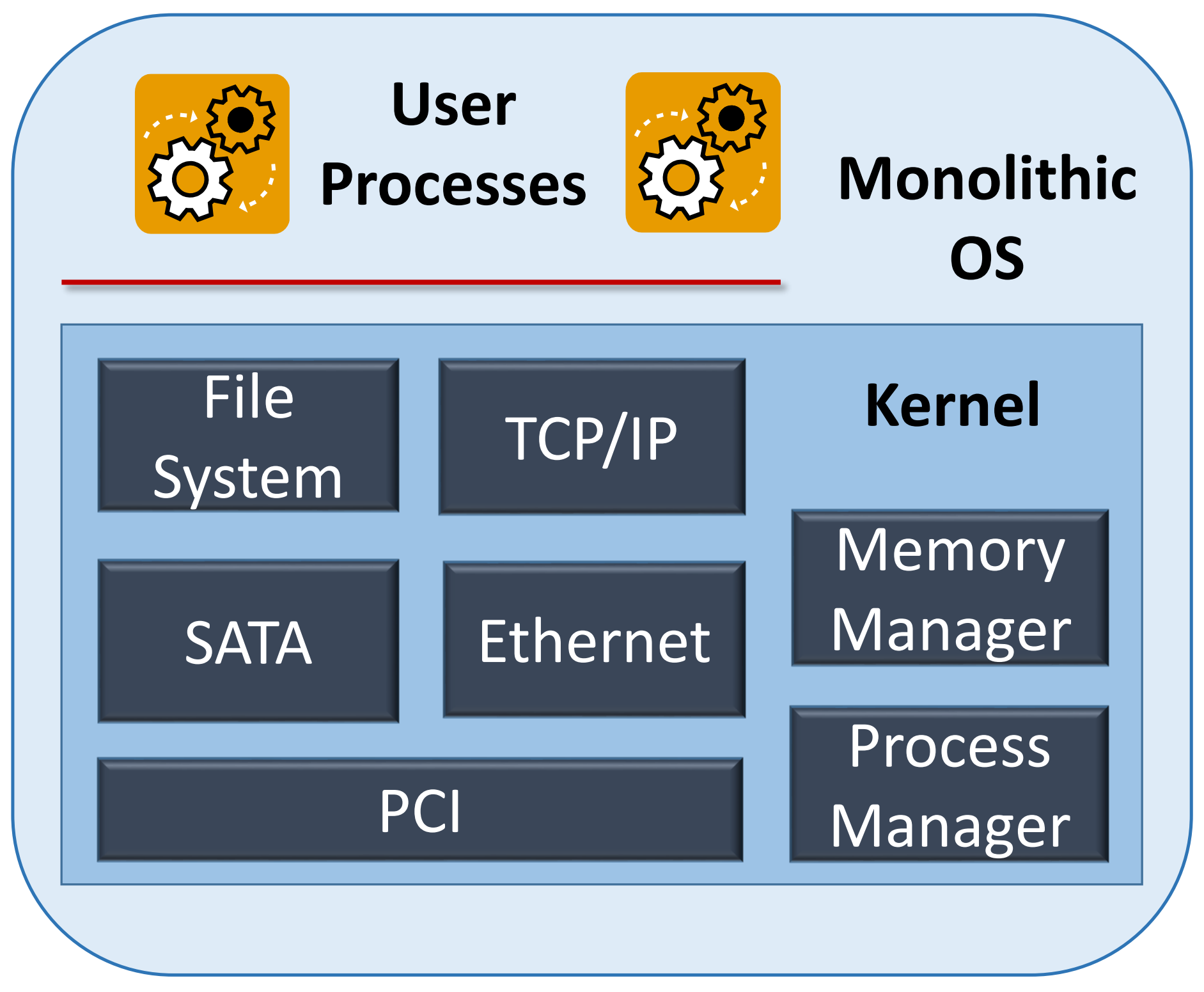
VirtuOS: an operating system with kernel virtualization

Ruslan Nikolaev, Godmar Back
Virginia Tech, Blacksburg

Motivation

Problem: Lack of isolation and protection for core systems code in monolithic OS.

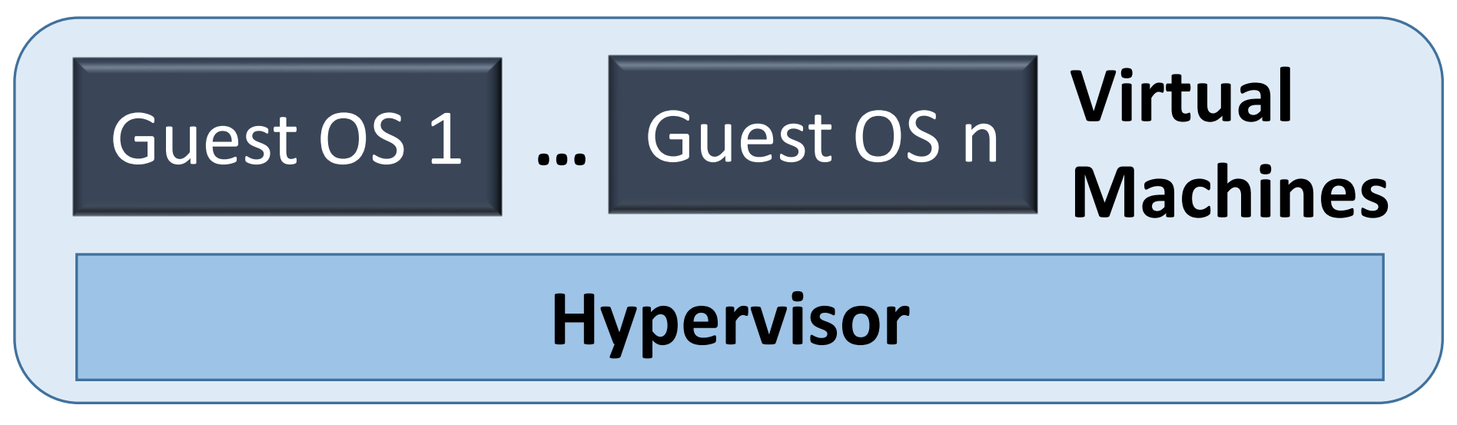
- Well-known problem – numerous studies & experience have indicated reliability problems, largely with 3rd party code.

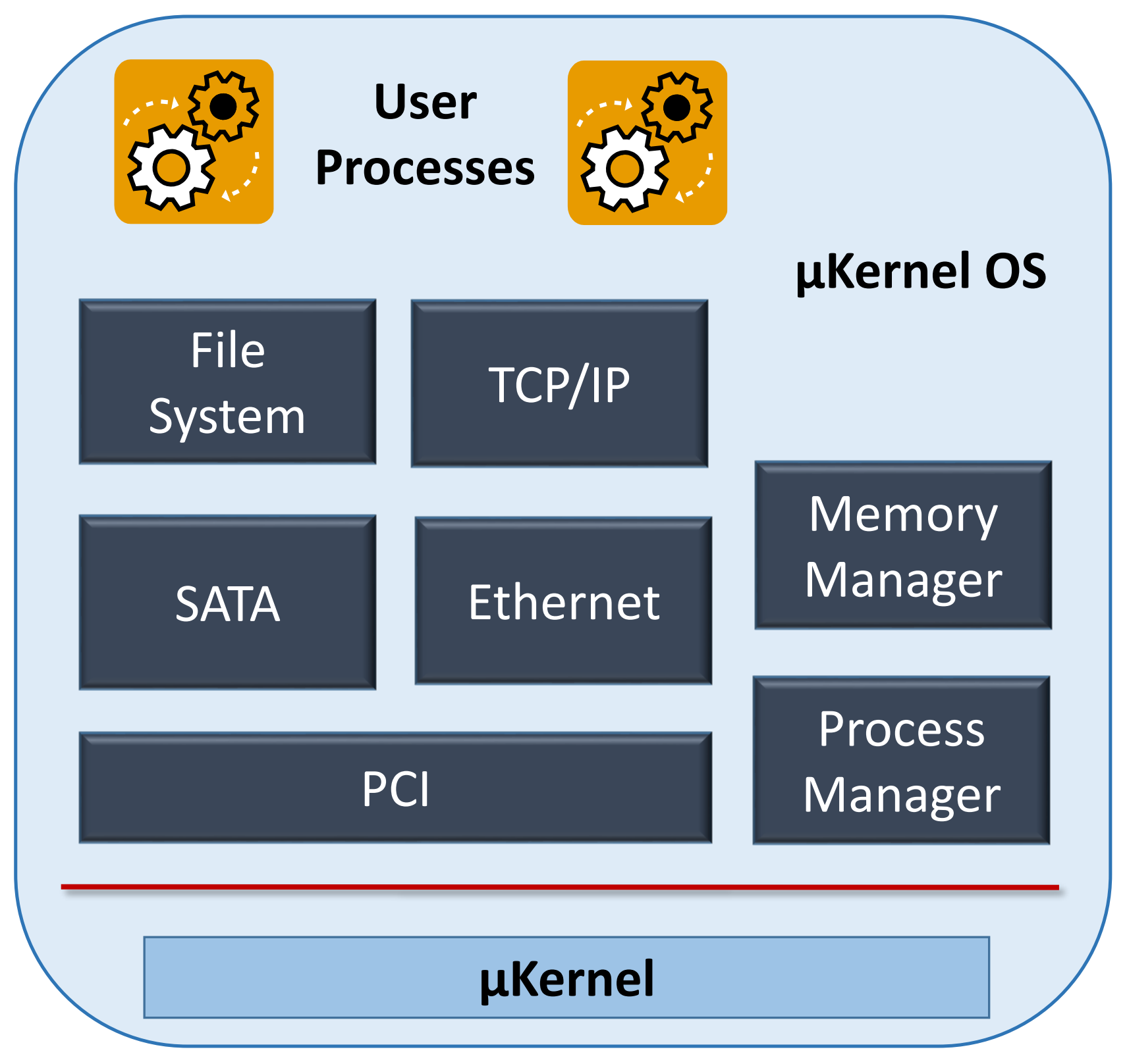


Approach: Decompose & Isolate Components.

Related Work

Existing Designs:
Rely on privilege separation and protection domains.
Examples: μ-Kernels, User-level drivers and file systems, VM-Based Isolation

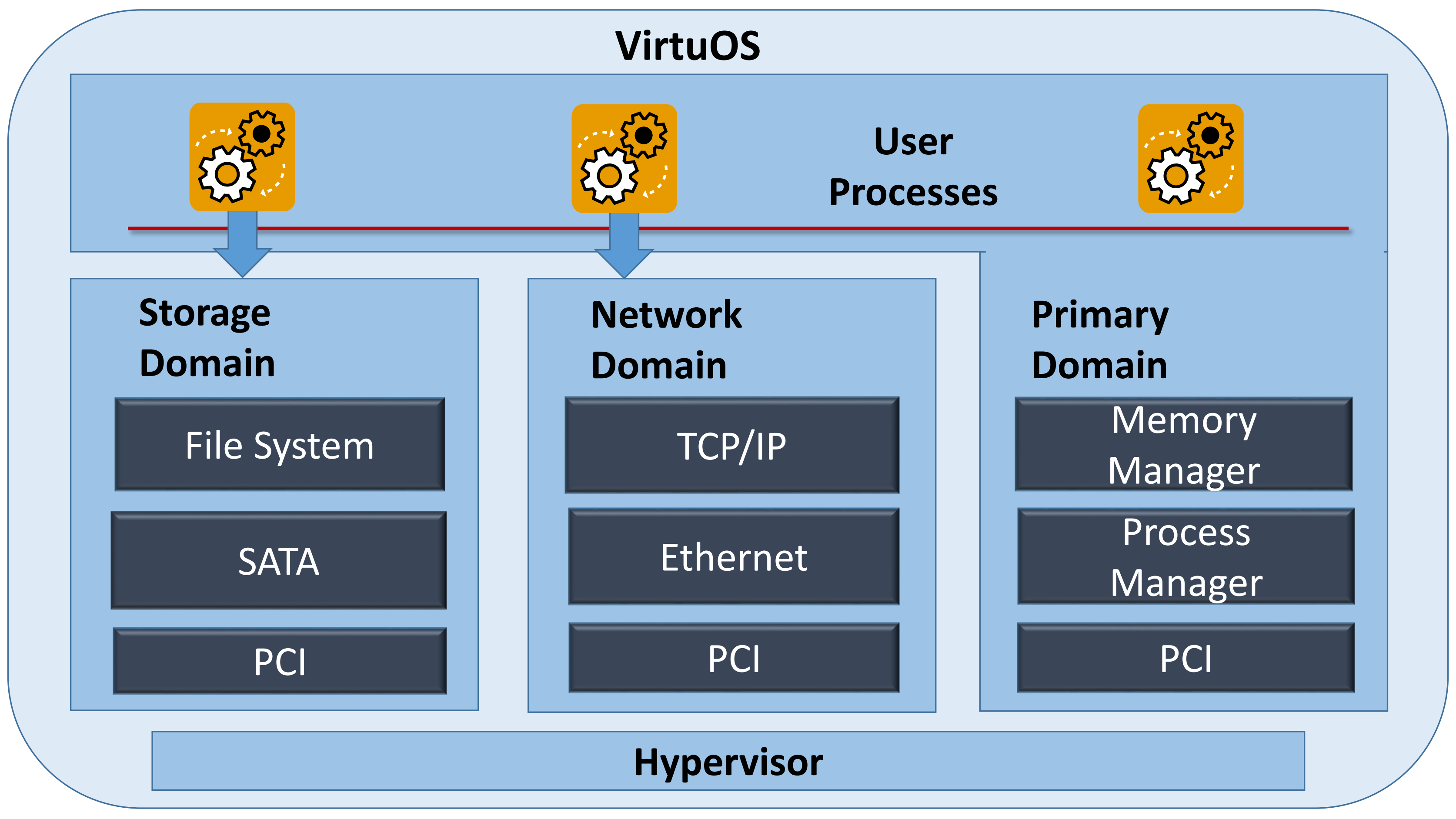




VirtuOS Design Characteristics

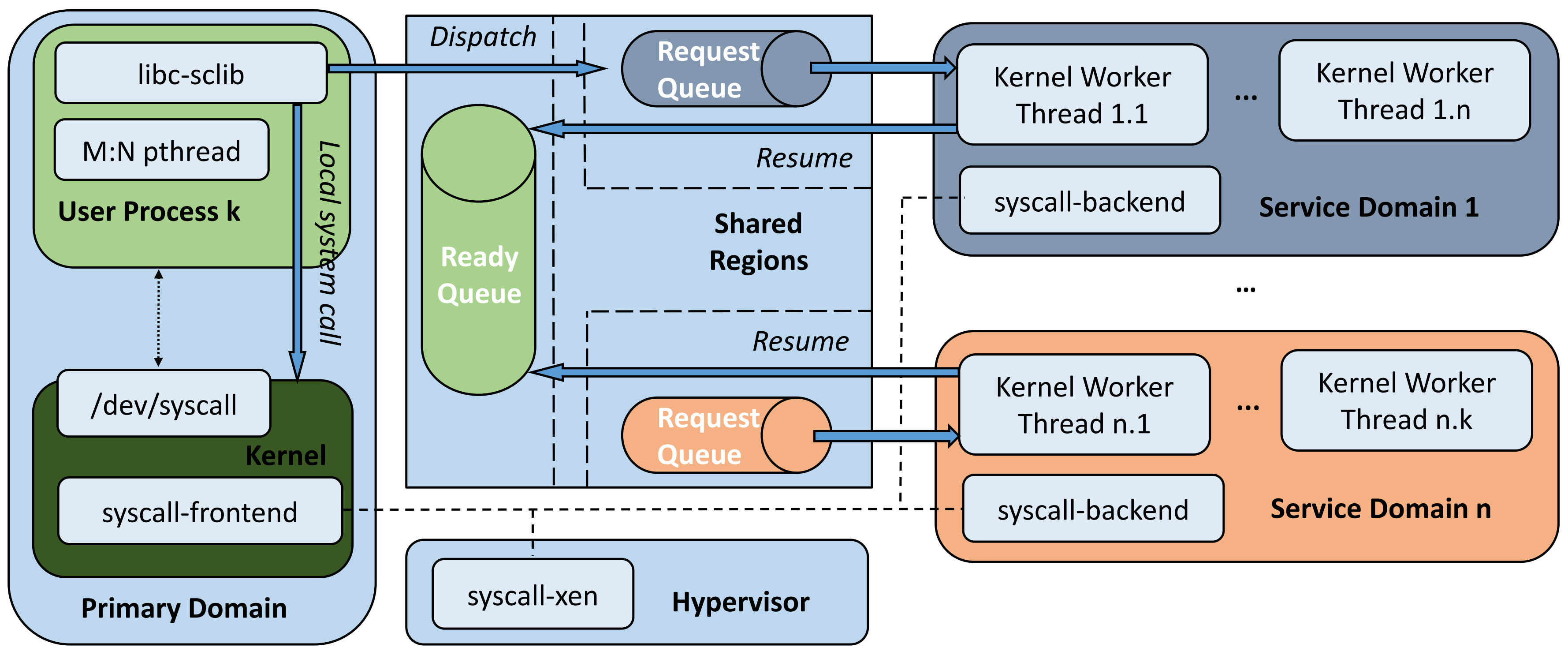
Flexible Decomposition of vertical slices of a monolithic kernel into service domains
Strong Isolation & Device Protection through hardware-supported virtual machines
Separate Failure & Recovery of service domains
Transparent to kernel code
Compatible with POSIX application code
Good Performance due to fast interdomain communication

Architecture: Primary & Service Domains



VirtuOS Implementation Highlights

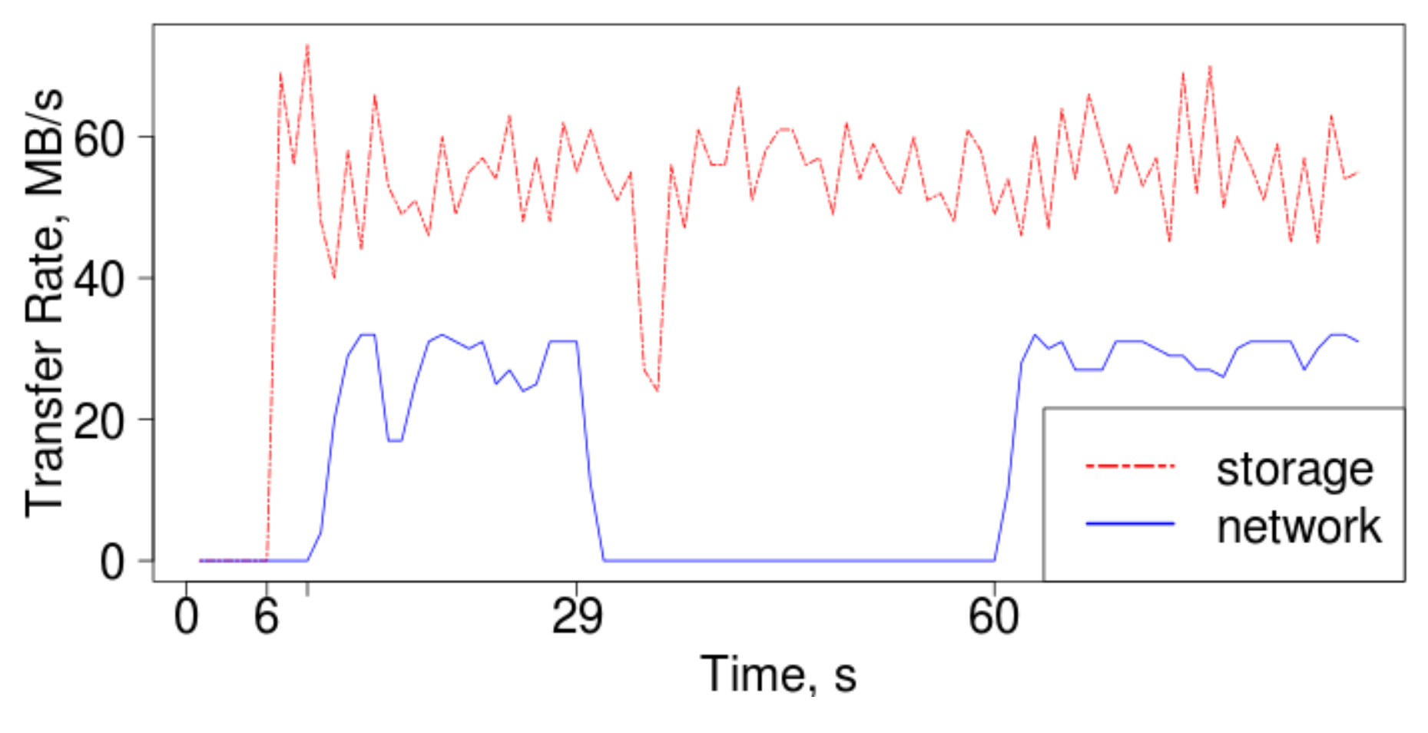
- Direct system call handling by remote domains via **exceptionless** system call dispatch
- Integrated with **M:N threading** to avoid interdomain signaling
- **Lock free** request and ready queues for dispatch & wakeup
- Supports all of **POSIX** (including polling & signals)



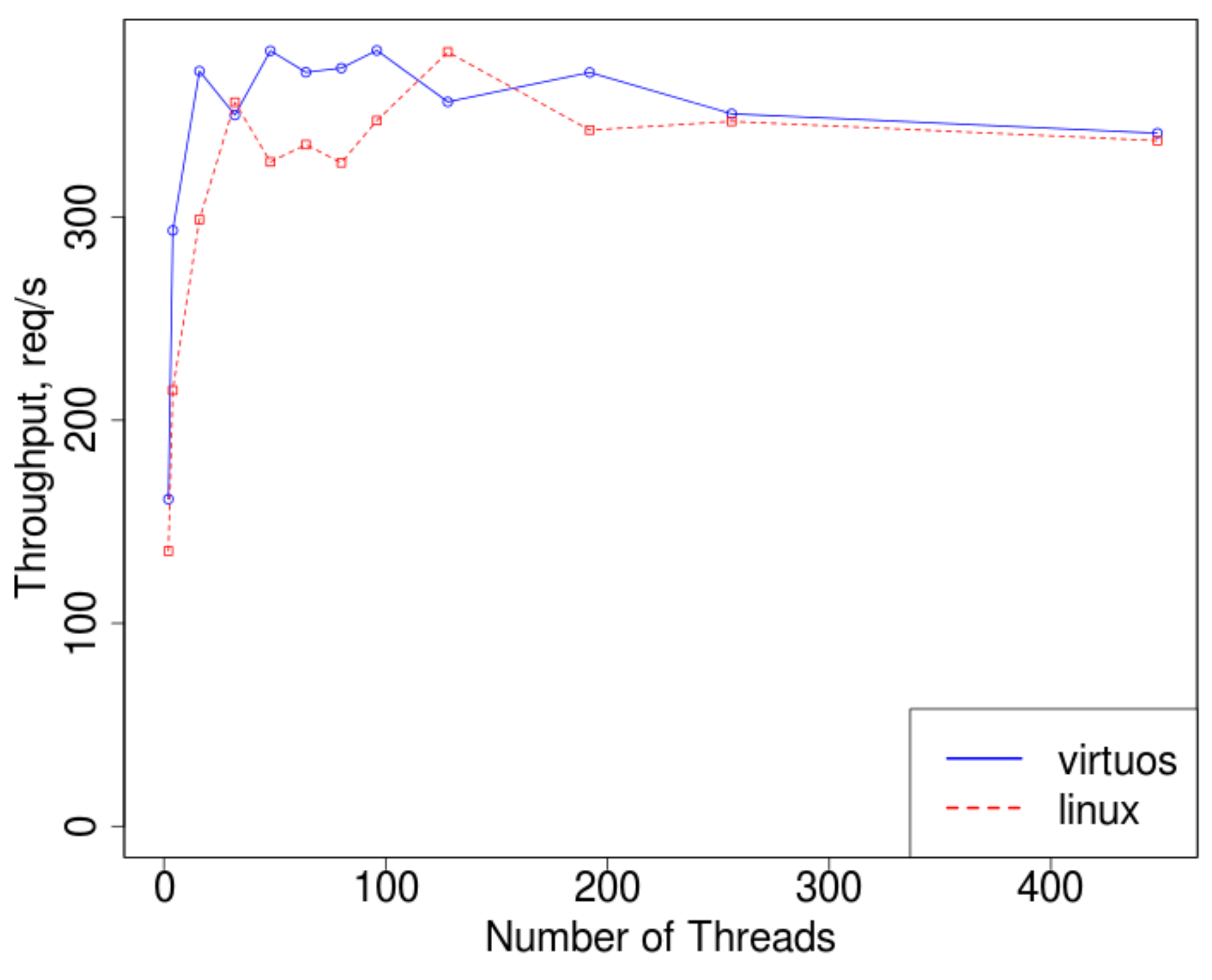
Source Code available at:
people.cs.vt.edu/~rnikola

Experimental Results

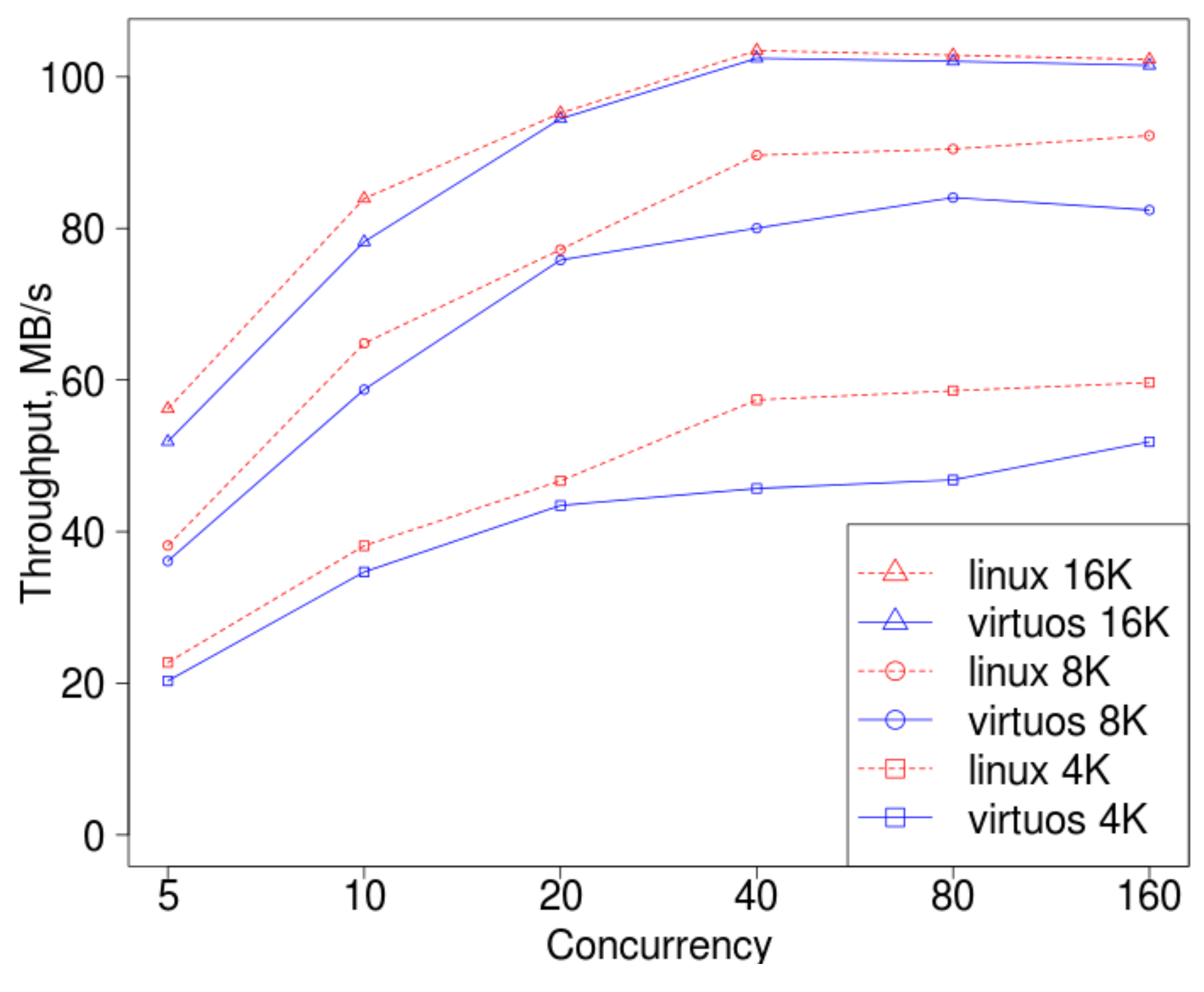
Evaluated **Overhead** due copying, coordination, evaluated **Failure Recovery** when service domains fail, and **Performance** for multithreaded & multiprocess workloads.



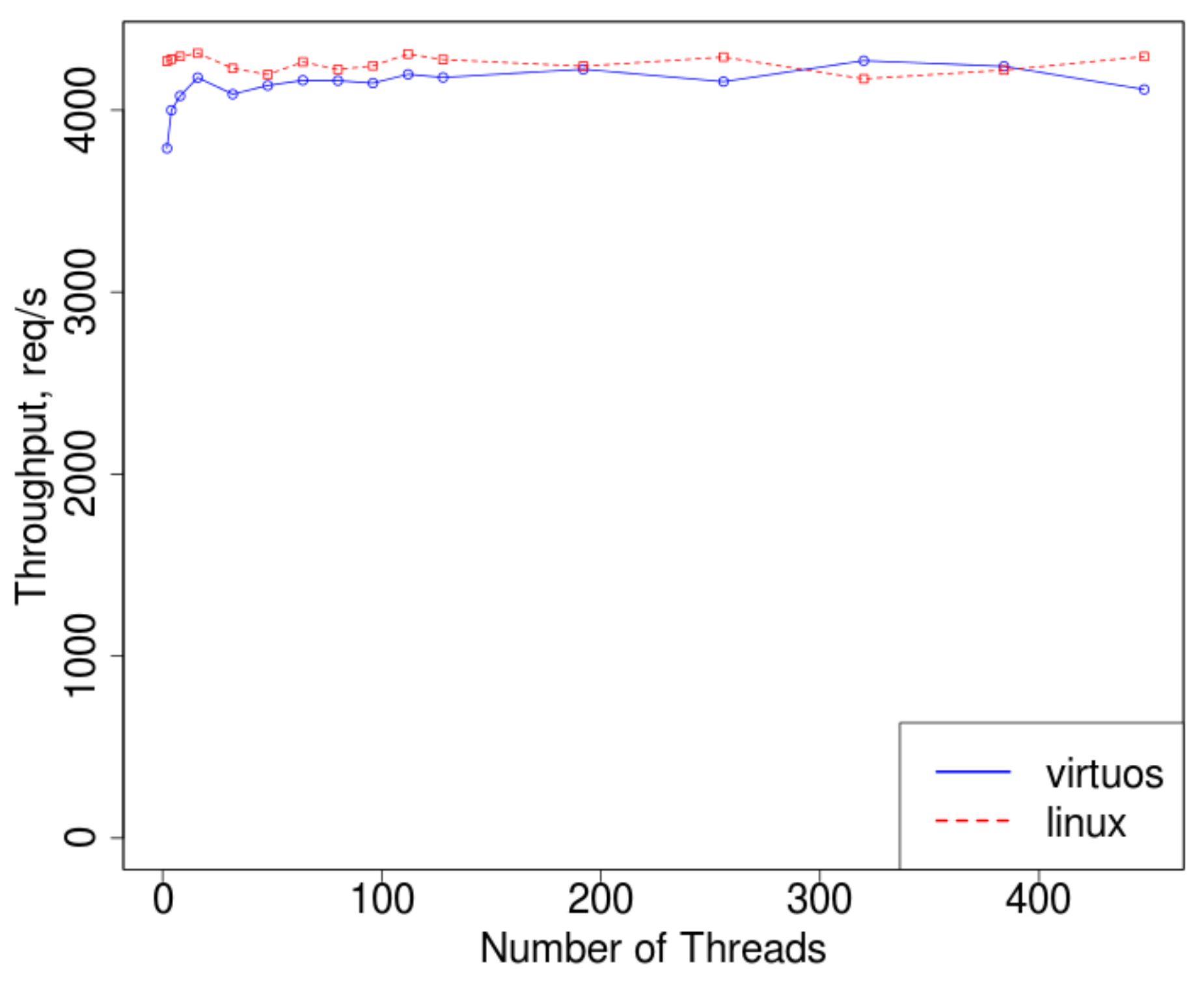
Failure recovery



OLTP/Sysbench mySQL



Apache throughput



FileIO/Sysbench