School of Computer Science & Engineering

COMP3891/9283 Extended Operating Systems

2025 T2 Week 03

Process Abstractions III: Scheduler Activations

Gernot Heiser

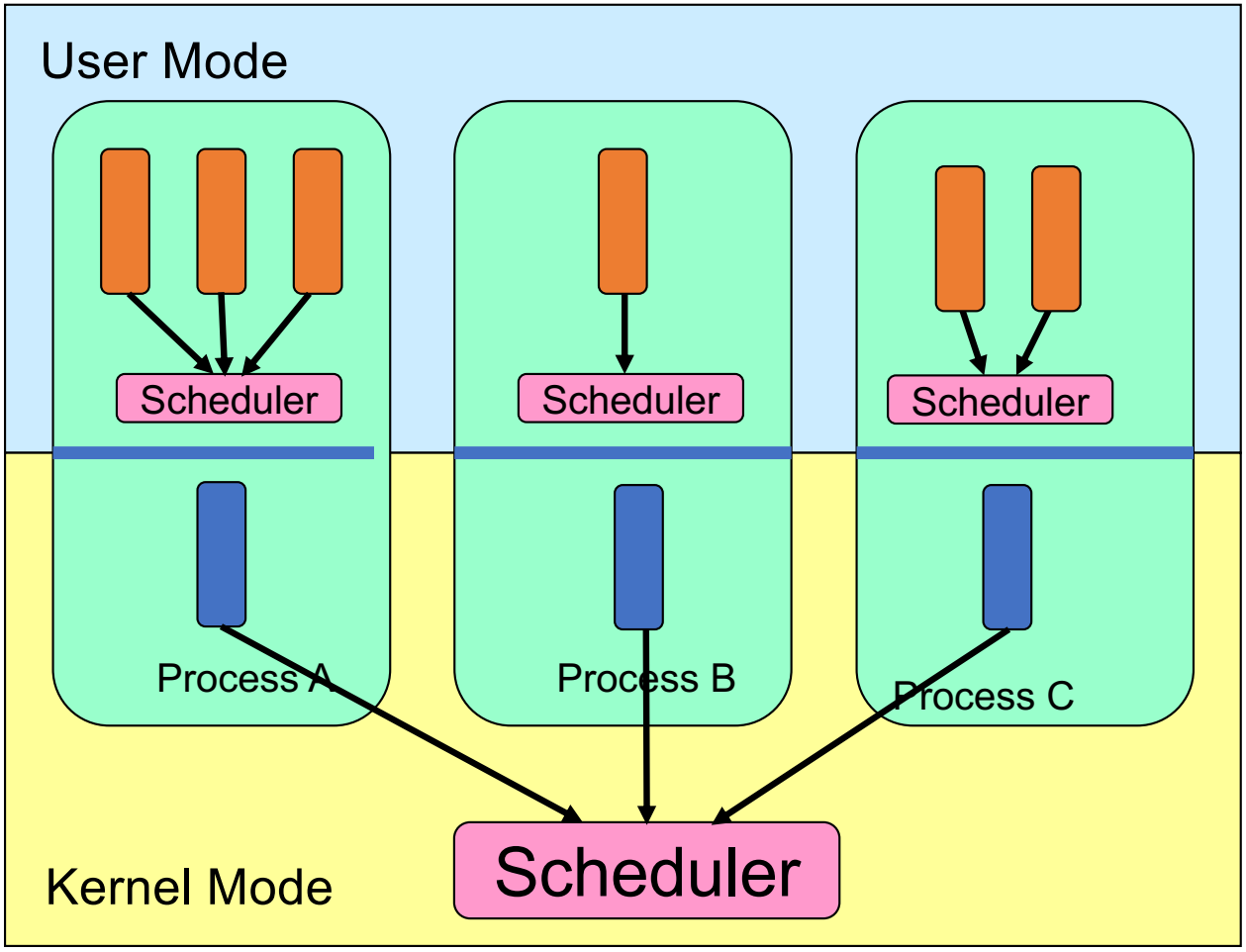
Copyright Notice

Learning Outcomes

- An understanding of hybrid approaches to thread implementation
- A high-level understanding of scheduler activations, and how they overcome the limitations of user-level and kernel-level threads.

User-level Threads

Thread management cheap
- create, delete, switch, sync

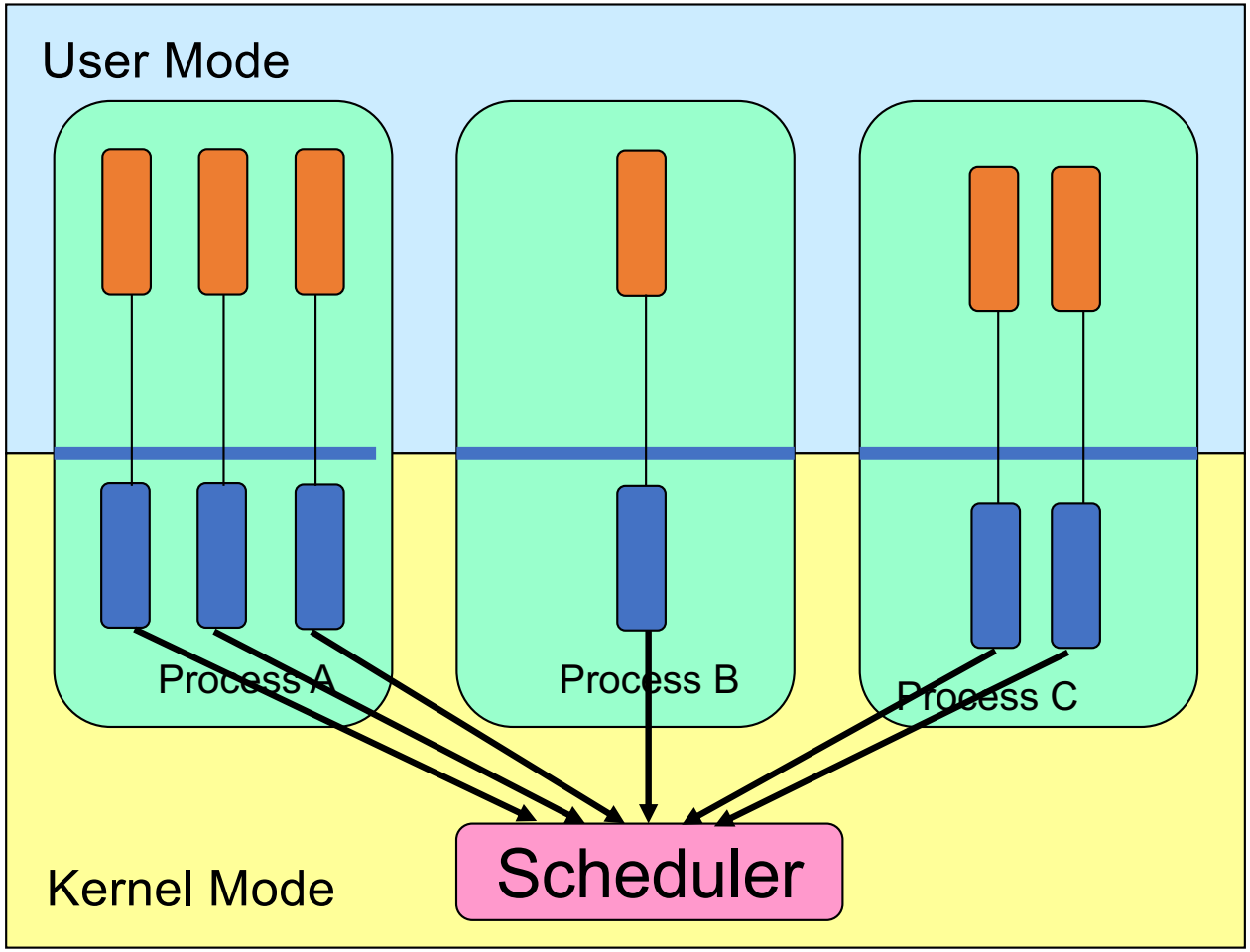


Blocking thread blocks process
- syscalls, page faults

No parallelism on multiprocessor

Kernel-Level Threads

Thread management expensive
- create, delete, switch, sync



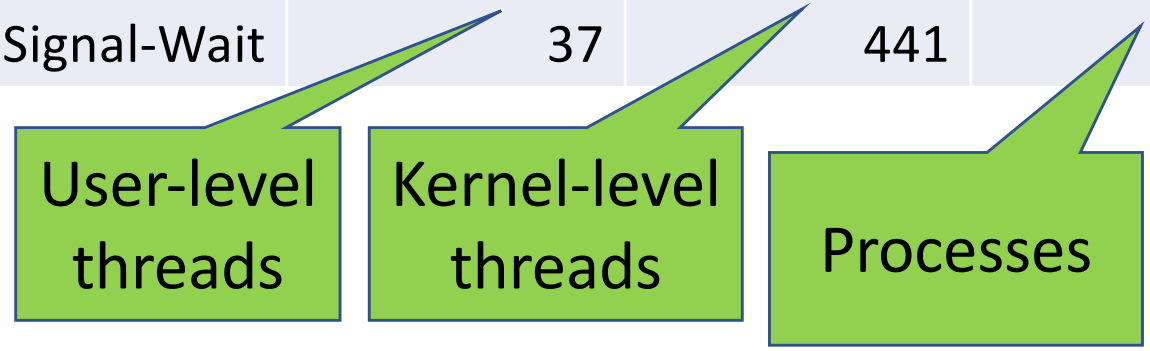
Blocking thread switches to other thread

Parallelism on multiprocessor

Performance

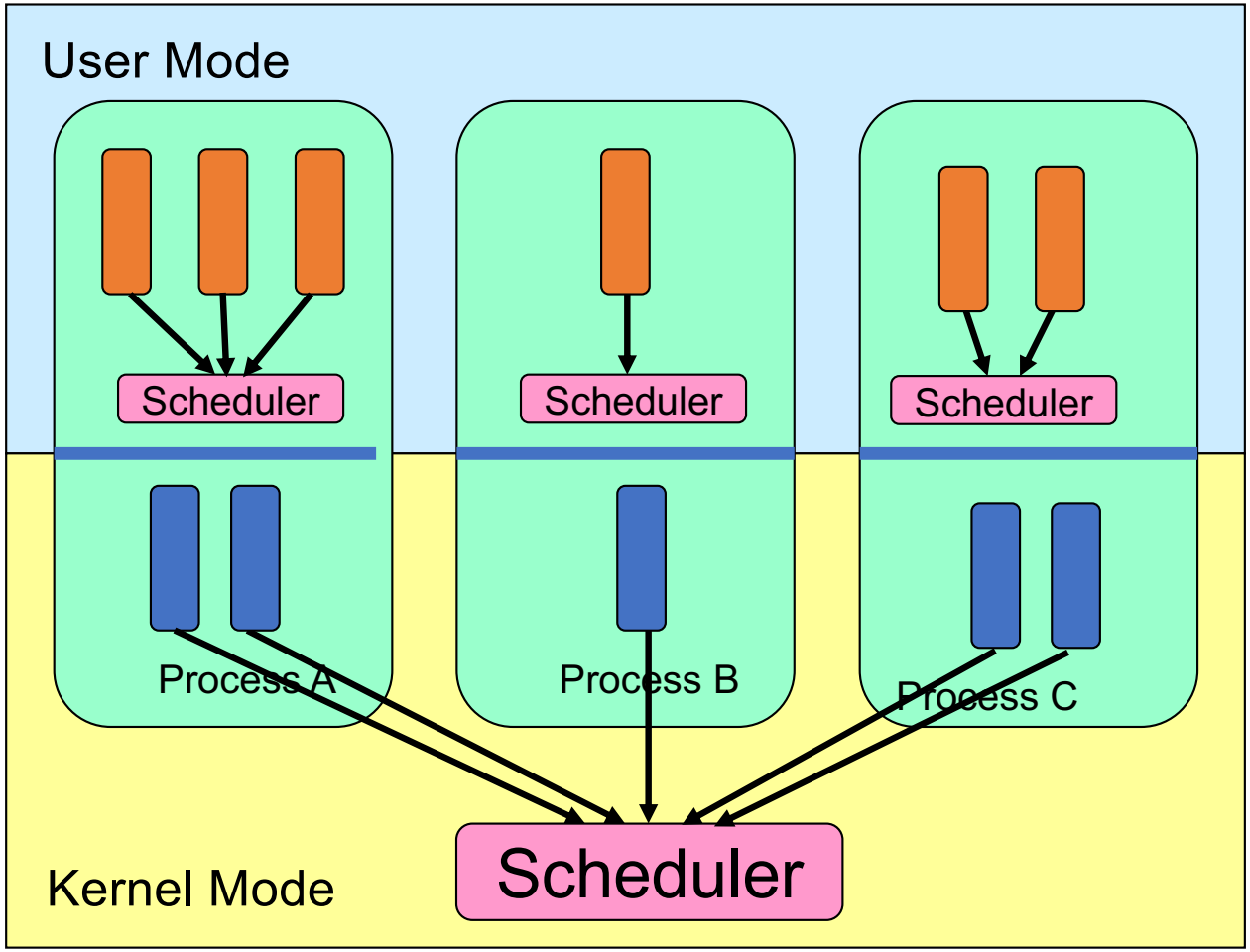
Thread operation latencies (μs) on CVAX.

Operation	FastThreads	Topaz Threads	Ultrix Processes
Null Fork	34	948	11300
Signal-Wait	37	441	1840



Hybrid Multithreading

Thread management cheaper
- create, delete, switch, sync



Blocking can still be a problem!

Parallelism on multiprocessor

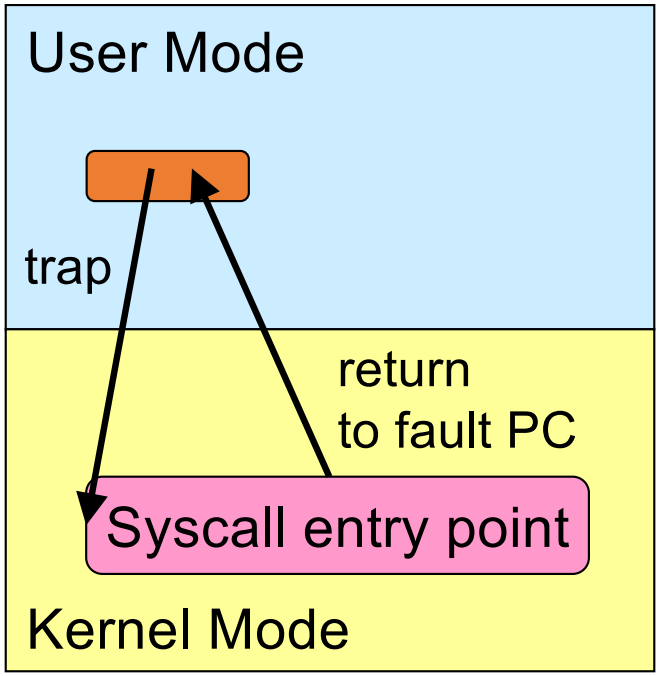
Scheduler Activations

Idea: Both schedulers co-operate

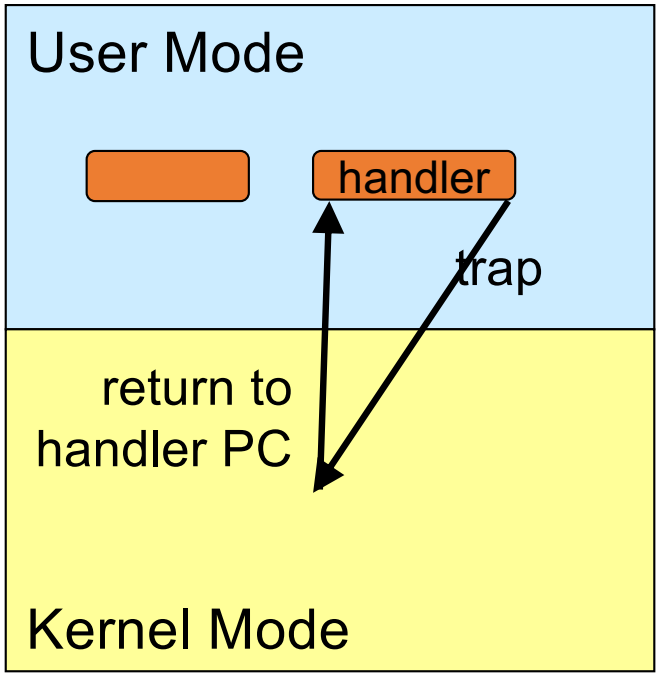
- User scheduler uses system calls
- Kernel scheduler uses upcalls!

First proposed by:
Thomas Anderson, Brian Bershad, Edward Lazowska, and Henry Levy. *Scheduler Activations: Effective Kernel Support for the User-Level management of Parallelism.* ACM Transactions on Computer Systems 10(1), February 1992, pp. 53-79

Syscalls vs Upcalls



System Call
(Downcall)



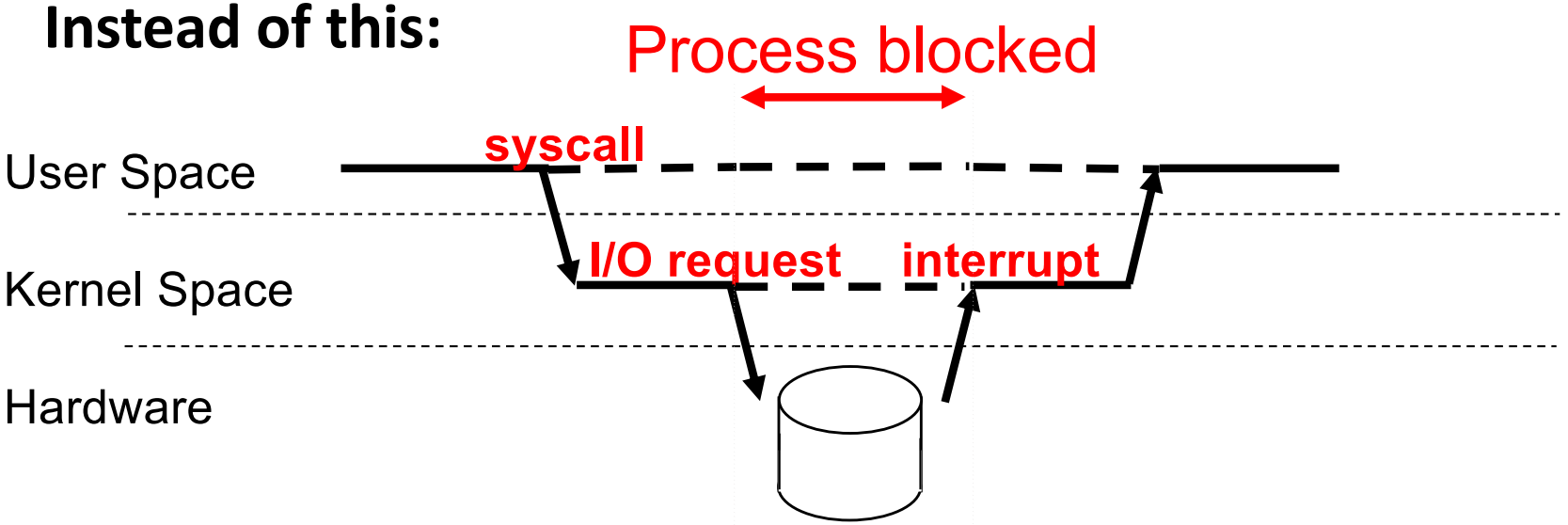
System Call
(Downcall)

Scheduler Activations

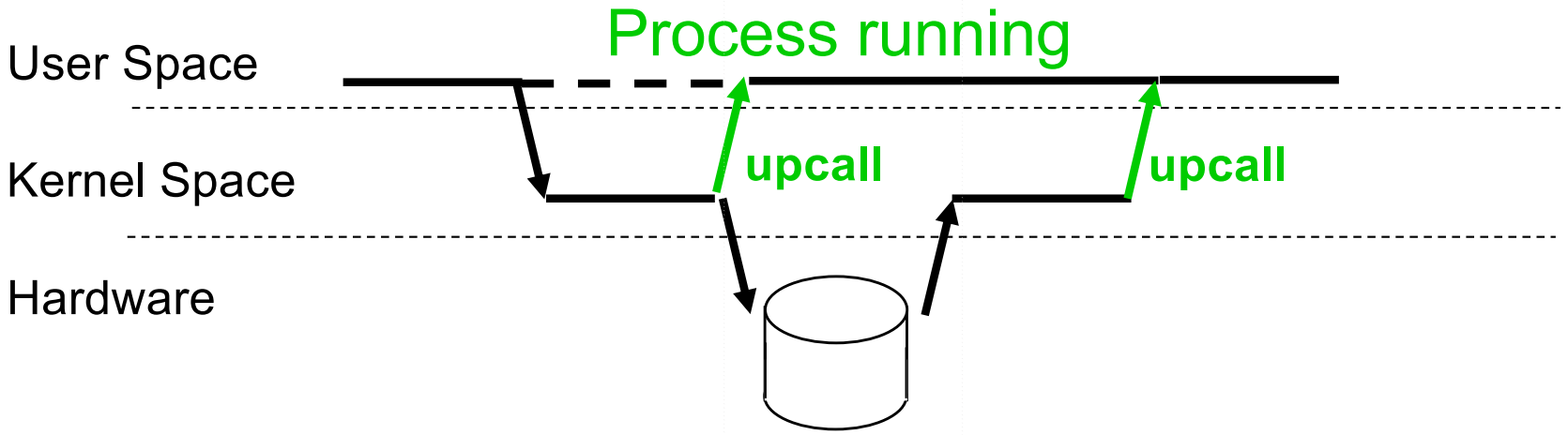
- Idea: Both schedulers co-operate
  - User scheduler uses system calls
  - Kernel scheduler uses upcalls!
- Two important concepts
  - Upcalls
    - Notify user-level of kernel scheduling events
  - Activations
    - A new structure to support upcalls and execution
      - approximately a kernel thread
    - One activations per (allocated) processor
    - Kernel controls activation creation and destruction

Scheduler Activations

Instead of this:



Do this:

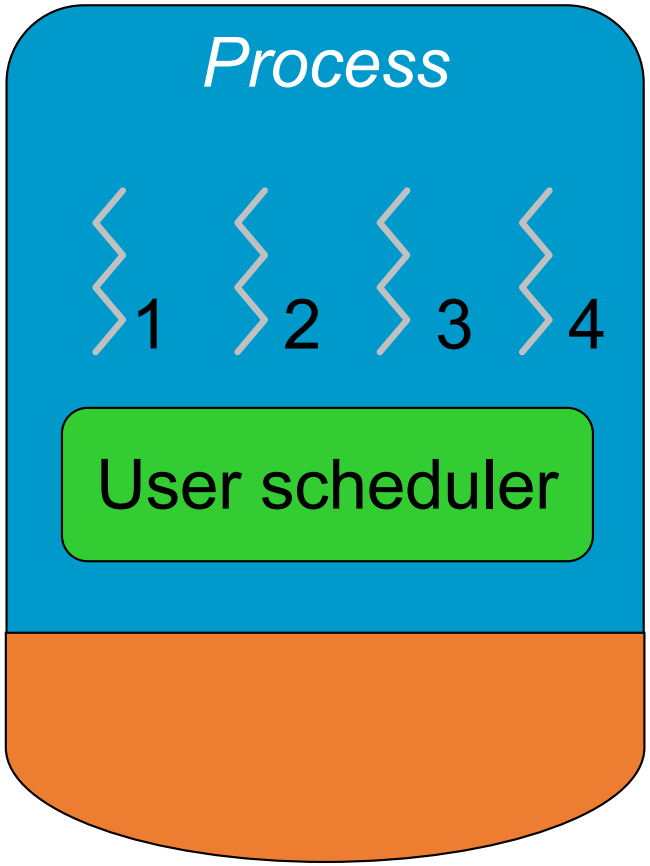


Upcalls to User-level scheduler

- New (processor #)
  - New virtual CPU allocated
  - Can schedule a user-level thread
- Preempted (activation # and its machine state)
  - Virtual CPU deallocated
  - Can schedule one less thread
- Blocked (activation #)
  - Notification of thread blocked
  - Can schedule another user-level thread
- Unblocked (activation # and its machine state)
  - Notification a thread has become runnable
  - Must decide to continue current or unblocked thread

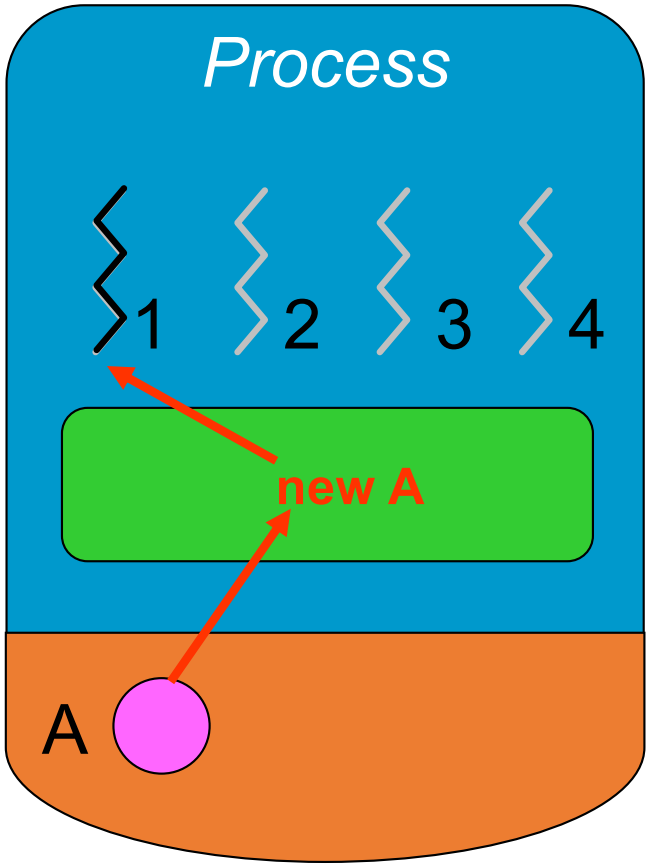
Working principle

Assume 2 processors



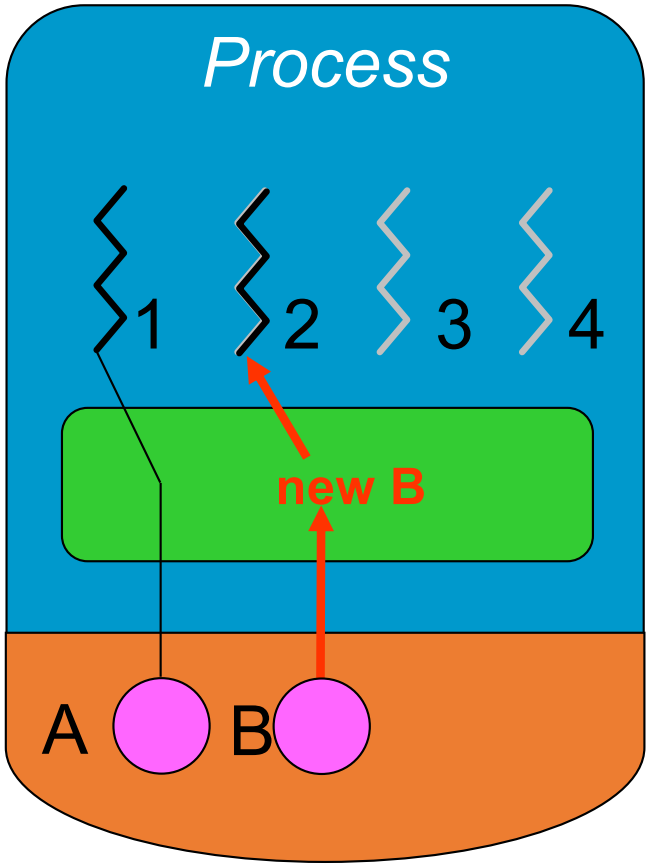
Working principle

Assume 2 processors



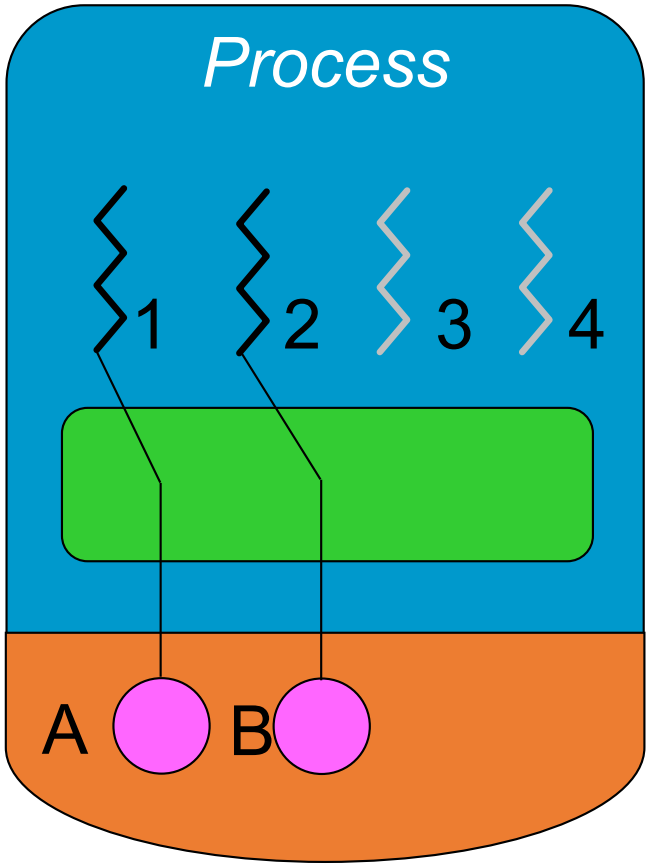
Working principle

Assume 2 processors



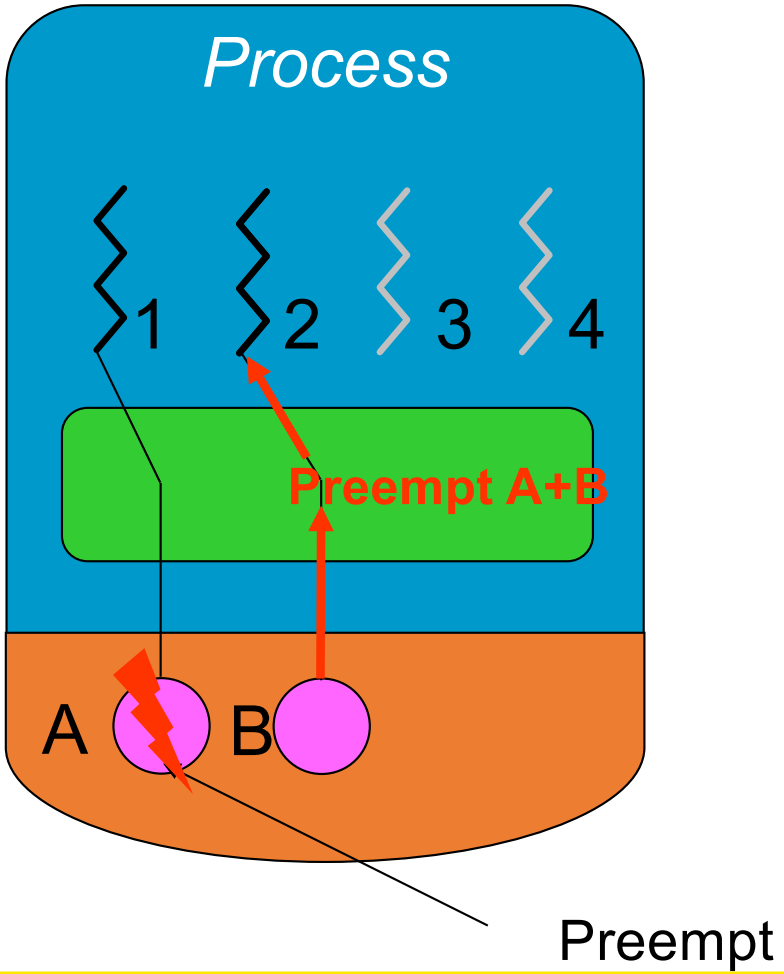
Working principle

Assume 2 processors



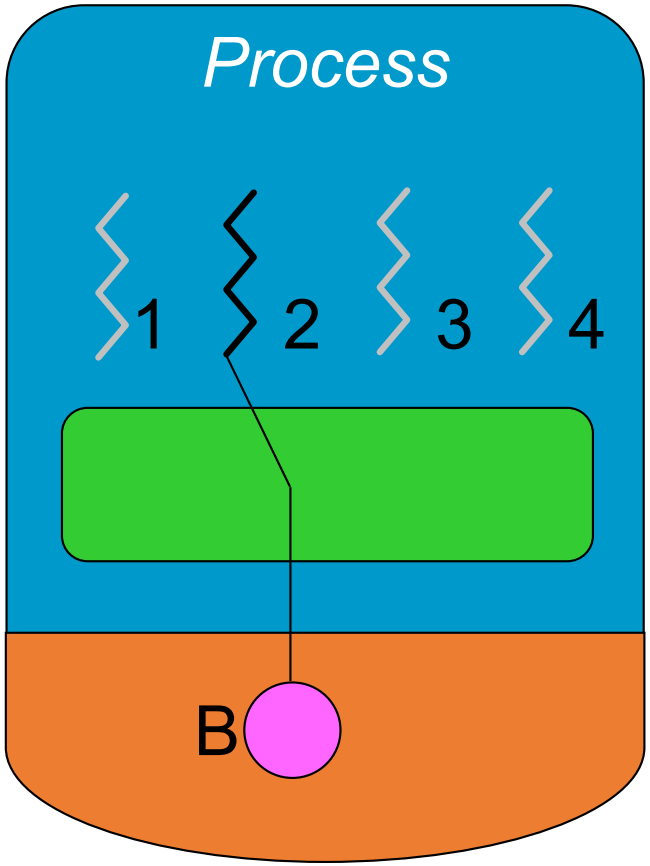
Working principle

Preemption scenario on 2 processors



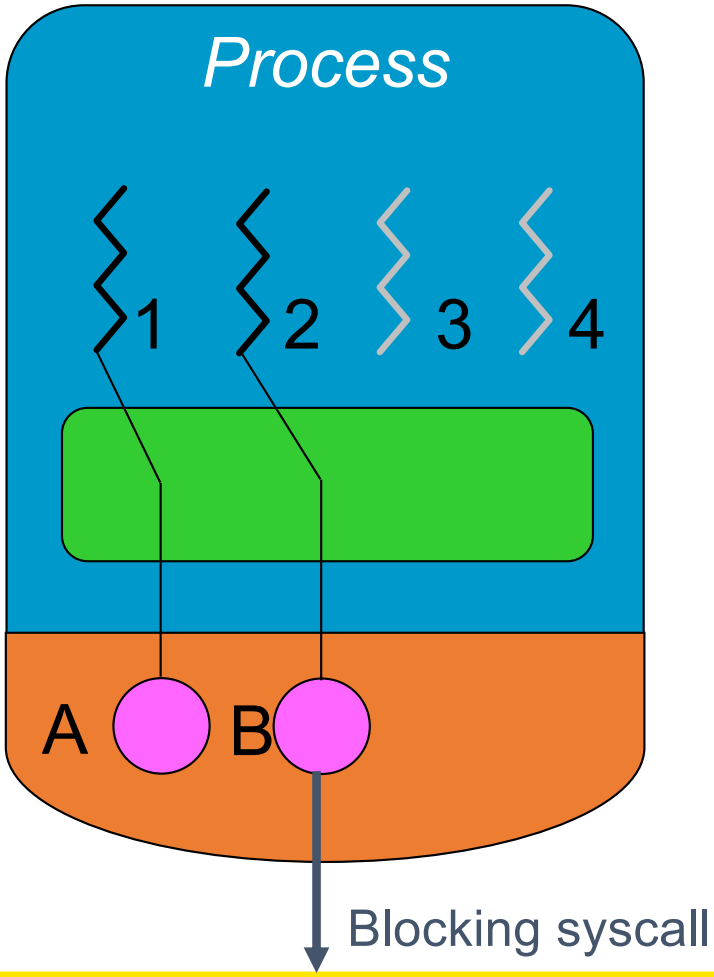
Working principle

Preemption scenario on 2 processors



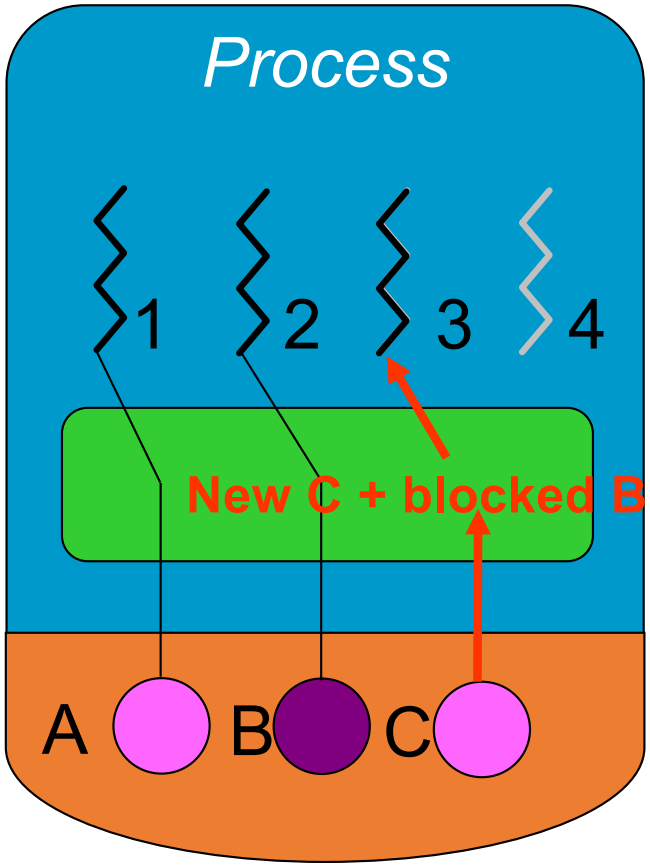
Working principle

Blocking syscall scenario on 2 processors



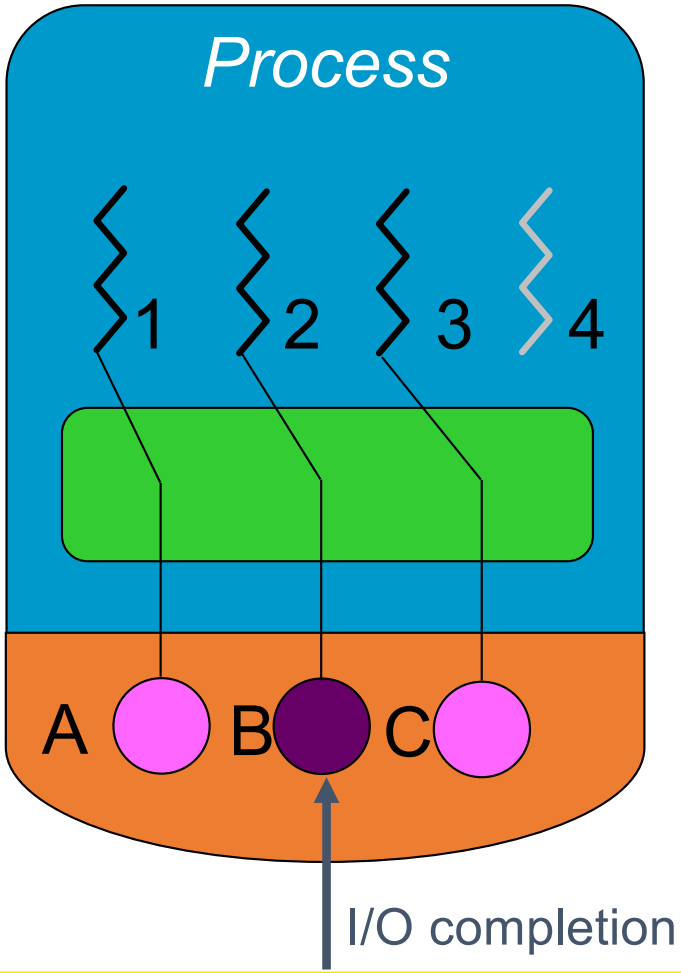
Working principle

Blocking syscall scenario on 2 processors



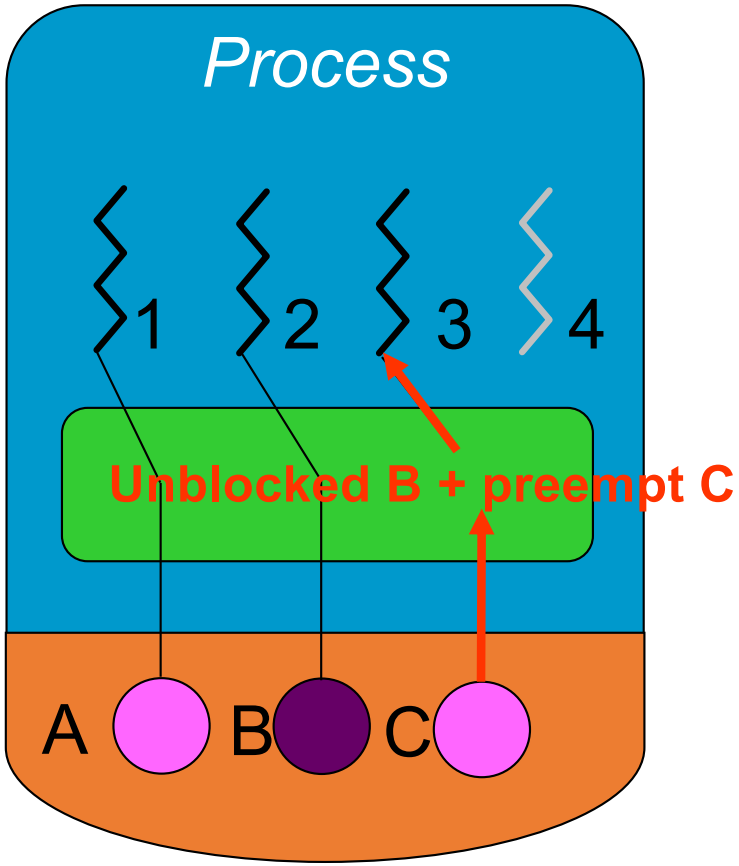
Working principle

Blocking syscall scenario on 2 processors



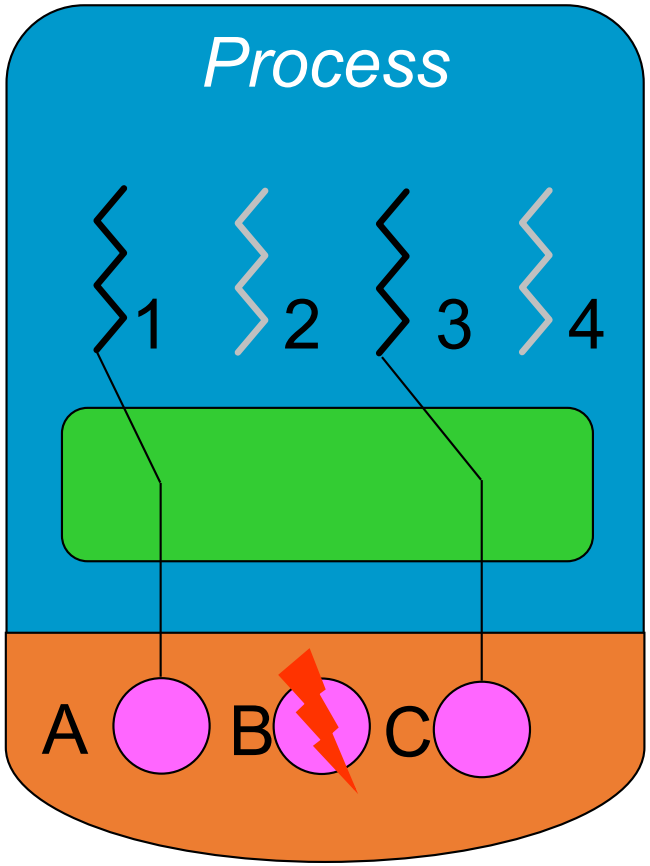
Working principle

Blocking syscall scenario on 2 processors



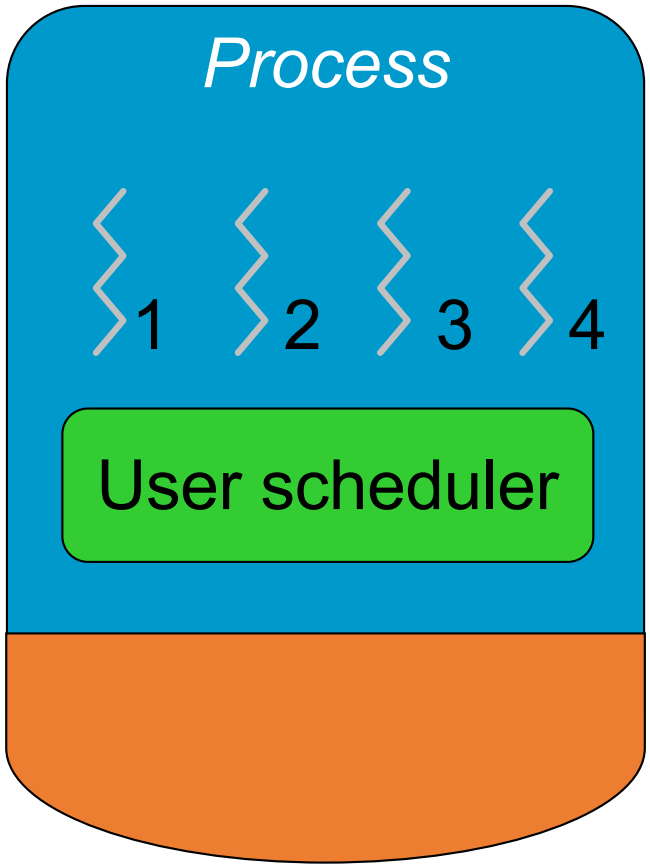
Working principle

Blocking syscall scenario on 2 processors



Scheduler Activations

Fast user-level thread management



Blocking creates new activation
- allows user-level scheduler to pick new thread

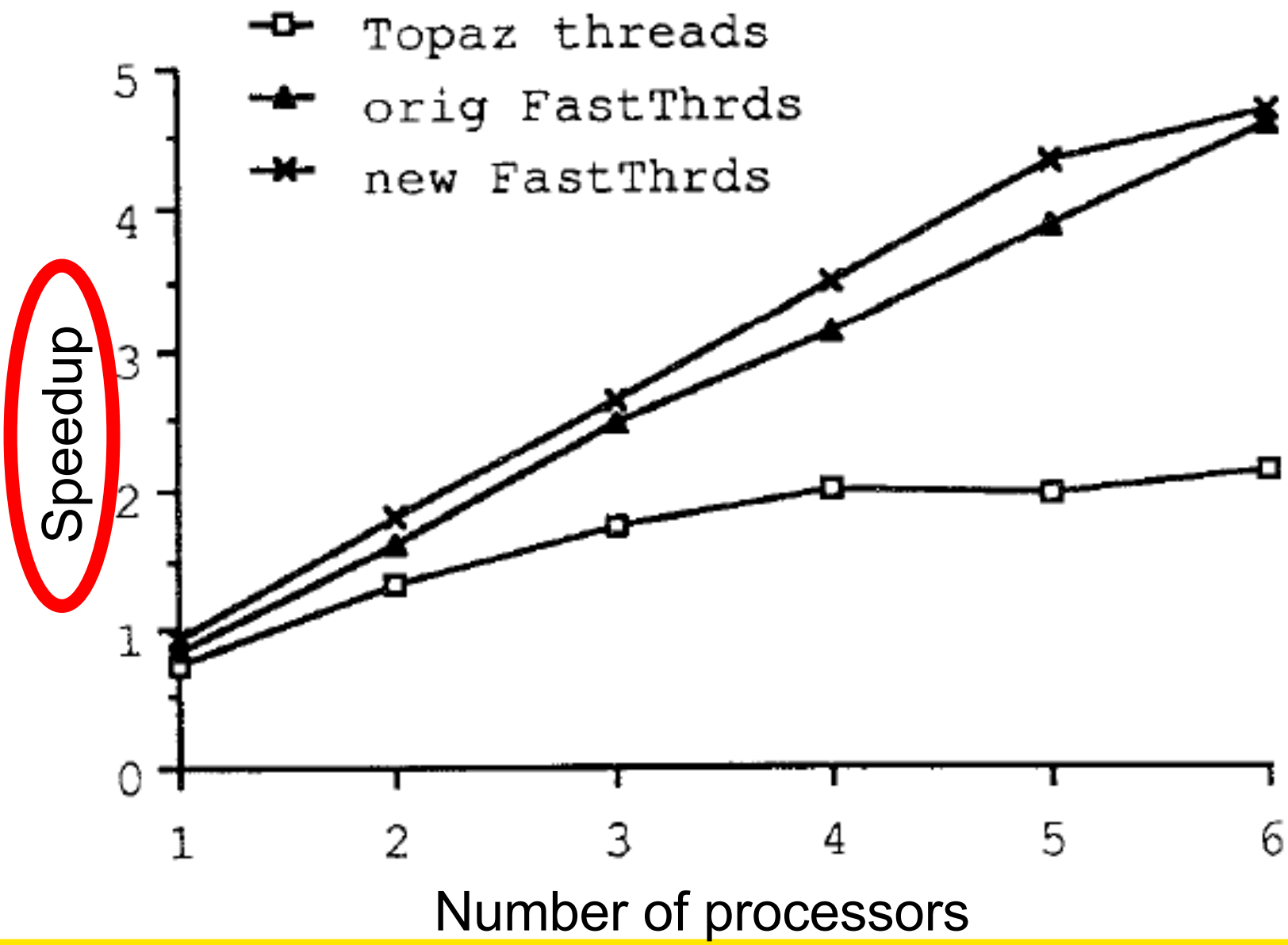
Real parallelism via activations
- Activations are virtual CPUs
- Can have as many as real CPUs

Performance

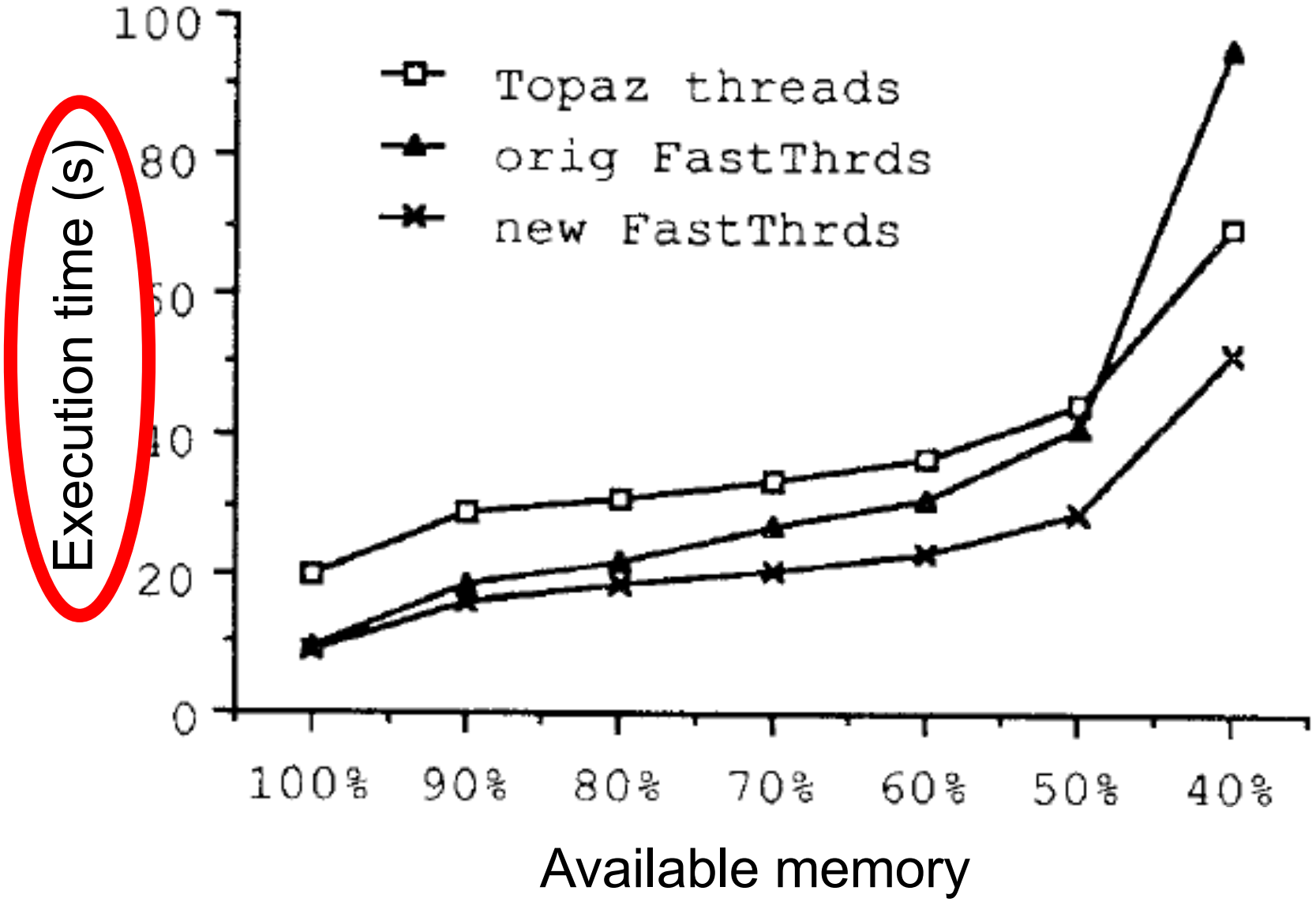
Thread operation latencies (μs) on CVAX.

Operation	FastThread on Topaz Threads	FastThread on Sched. Activ.	Topaz Threads	Ultrix Processes
Null Fork	34	37	948	11300
Signal-Wait	37	42	441	1840

Performance: Compute-Bound



Performance: I/O-Bound



Adoption

- Adopters
  - BSD “Kernel Scheduled Entities”
    - Reverted back to kernel threads
  - Variants in Research OSs: K42, Barrelfish
  - Digital UNIX
  - Solaris
  - Mach
  - Windows 64-bit *User-Mode Scheduling*
  - Research OS Composite
  - seL4 MCS timeout exceptions
- Linux -> kernel threads