School of Computer Science & Engineering

COMP3891/9283 Extended Operating Systems

2025 T2 Week 04

Scheduler Activations

Gernot Heiser

Copyright Notice

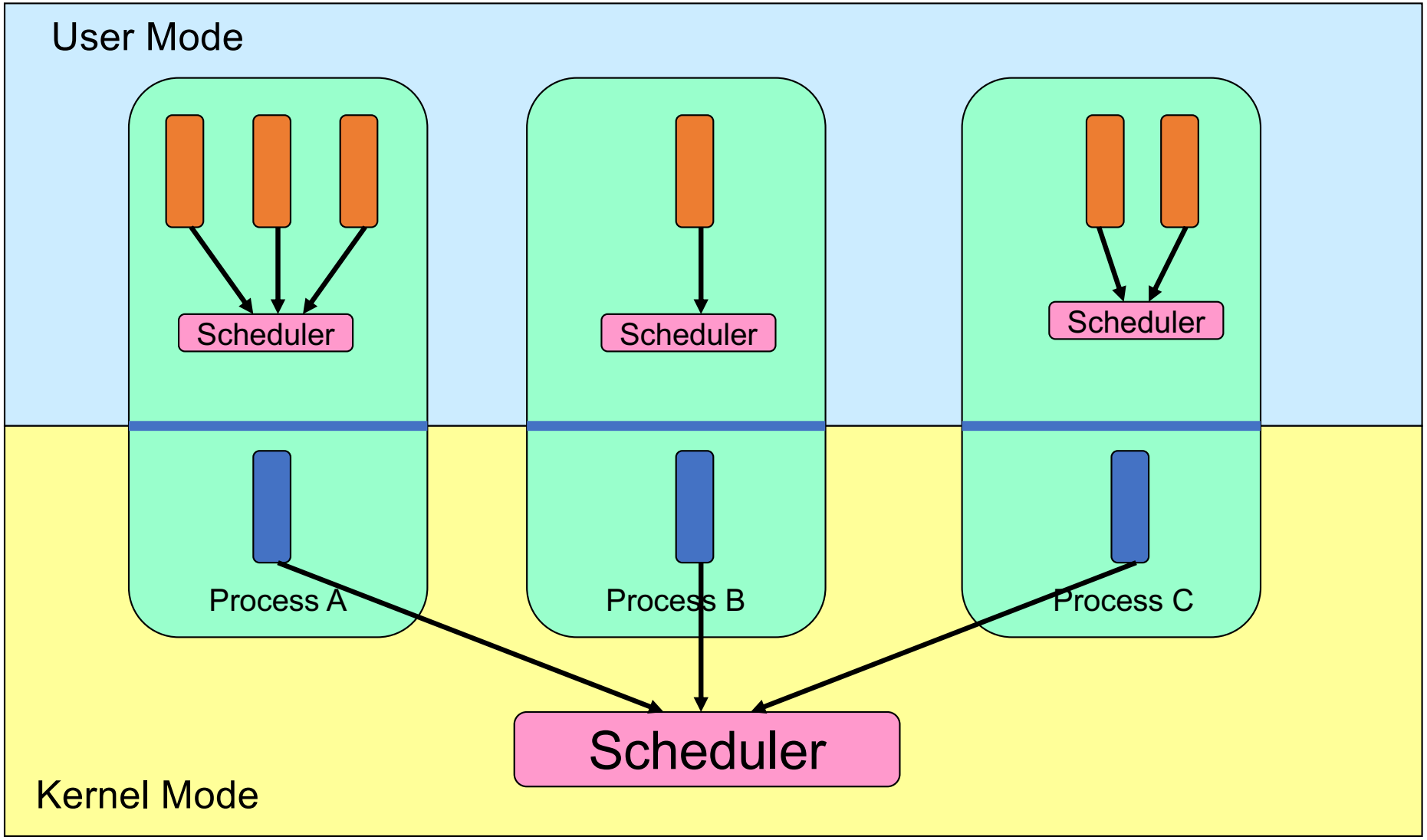
Learning Outcomes

- 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.

Scheduler Activations: The Paper

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.

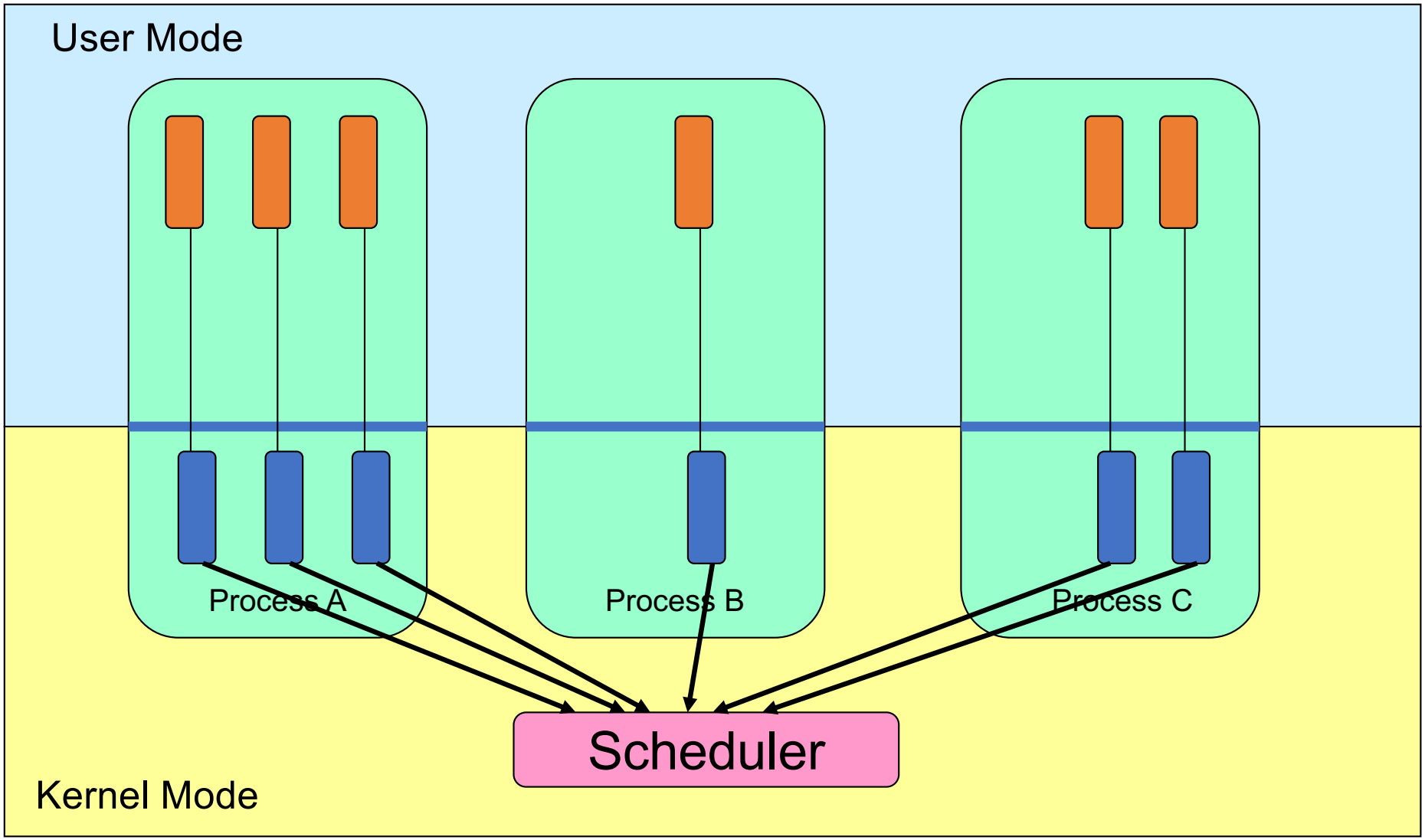
User-level Threads



User-level Threads: Pros & Cons

- ✓ Fast thread management (creation, deletion, switching, synchronisation…)
- ✗ Blocking blocks all threads in a process
  - Syscalls
  - Page faults
- ✗ No thread-level parallelism on multiprocessor

Kernel-Level Threads



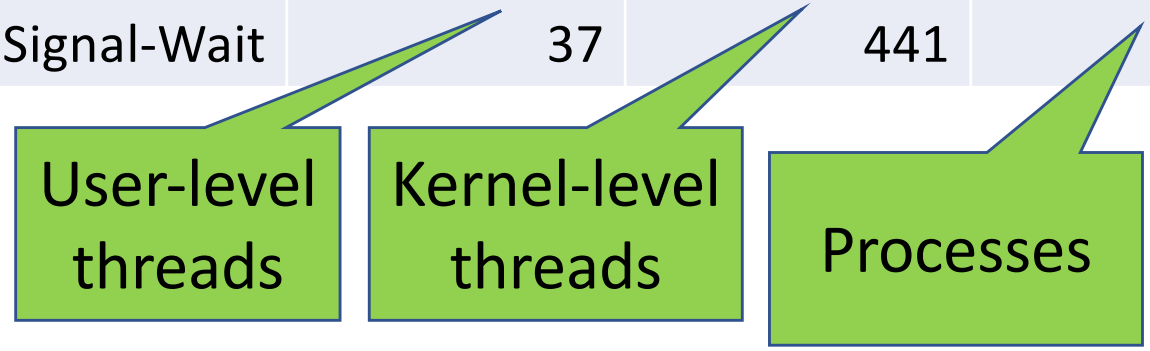
Kernel-level Threads: Pros & Cons

- ✘ Slow thread management (creation, deletion, switching, synchronisation…)
  - System calls
- ✓ Blocking blocks only the responsible thread in a process
- ✓ Thread-level parallelism on multiprocessor

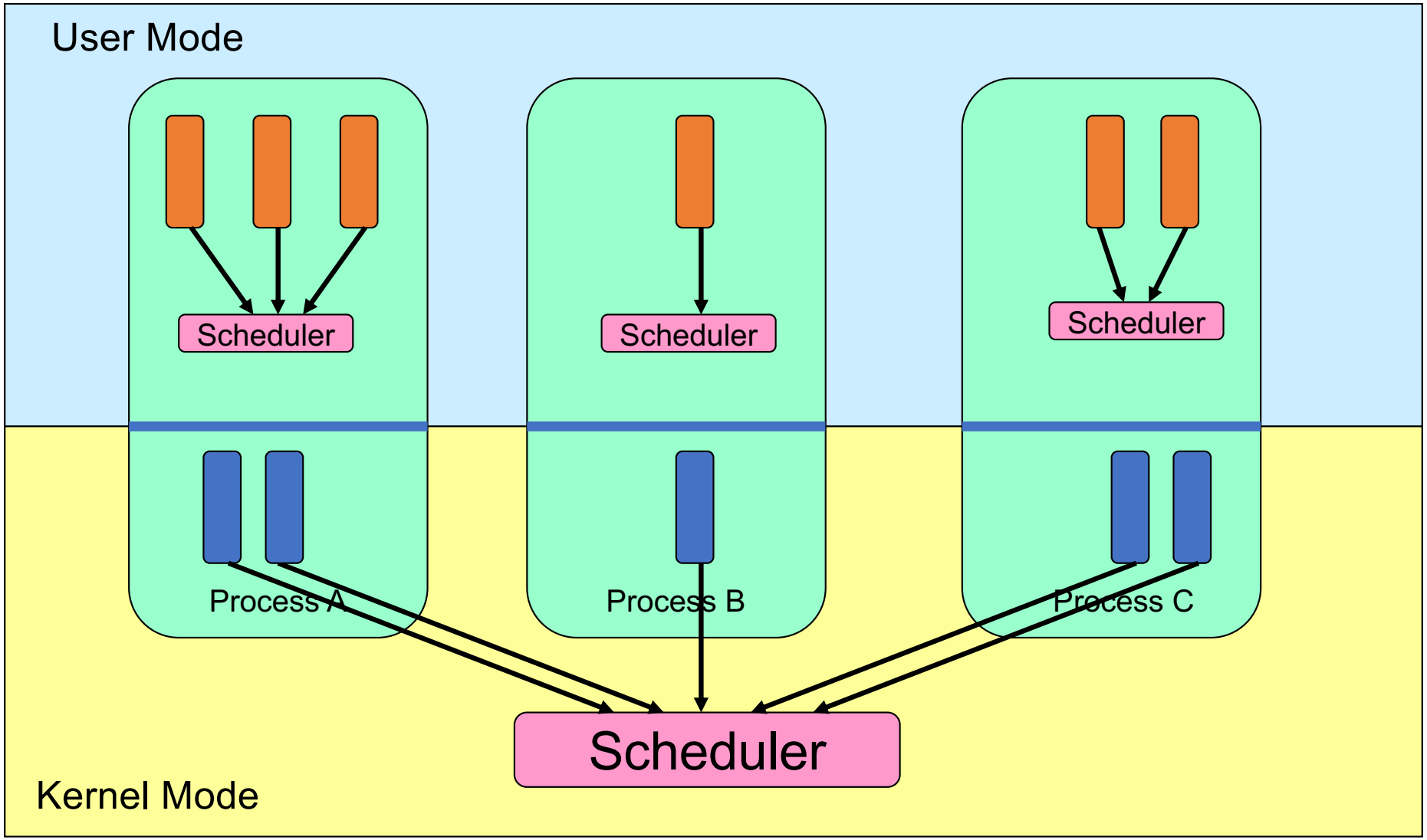
Performance

Thread operation latencies (μs) on CVAX.

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



Hybrid Multithreading



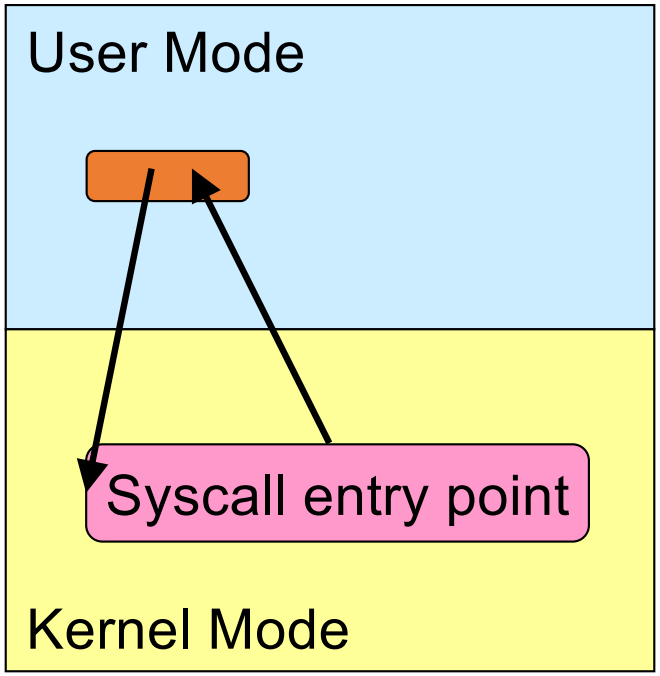
Hybrid Multithreading: Pros & Cons

- ✓ Can get real thread parallelism on multiprocessor
- ✘ Blocking can still be a problem!!!

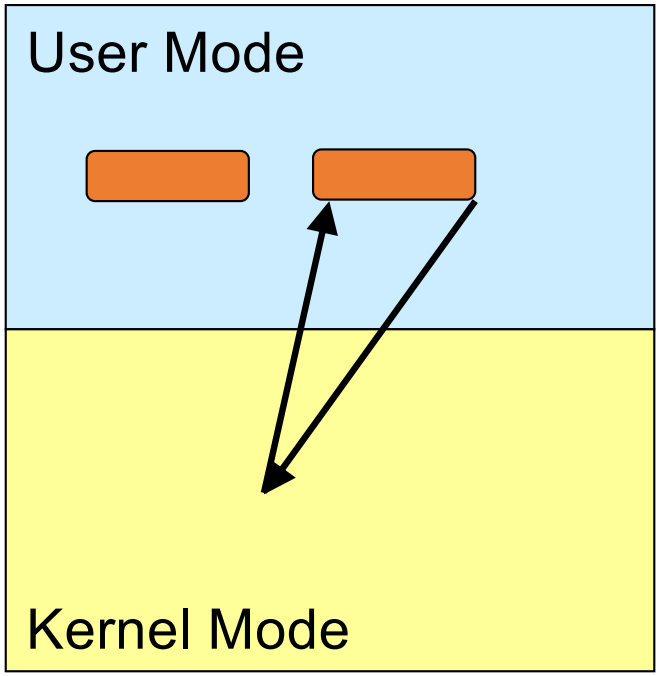
Scheduler Activations

- First proposed by [Anderson et al. 91]
- 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
        - As many running activations as (allocated) processors
        - Kernel controls activation creation and destruction

Syscalls vs Upcalls



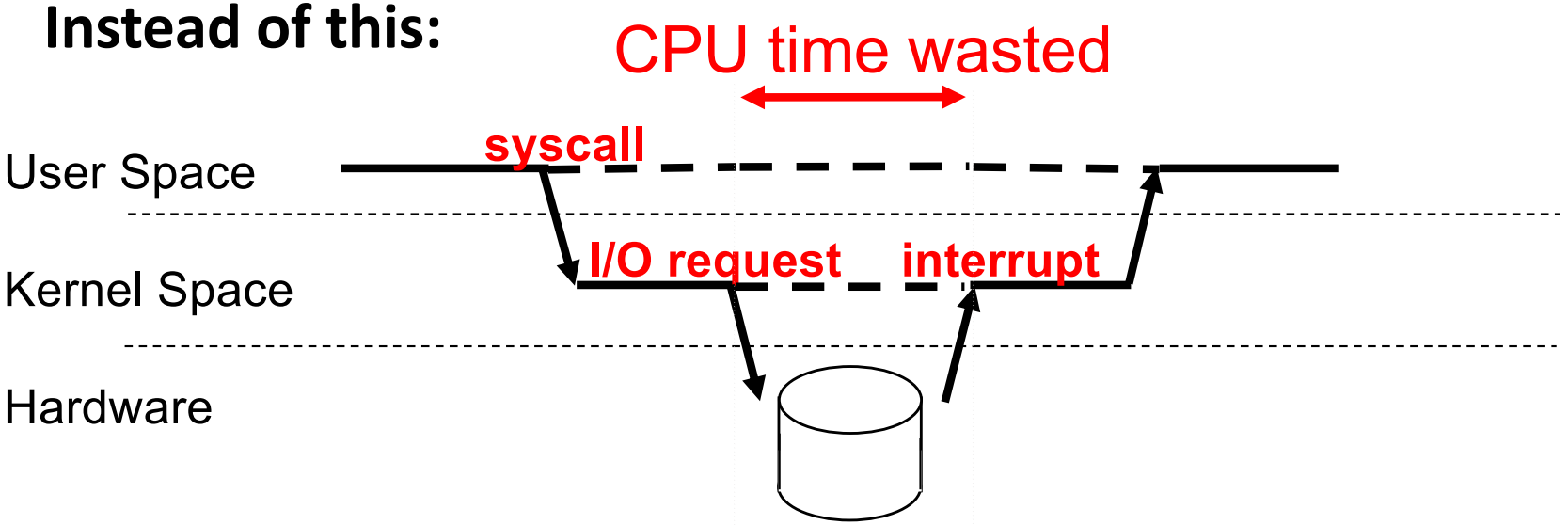
System Call
(Downcall)



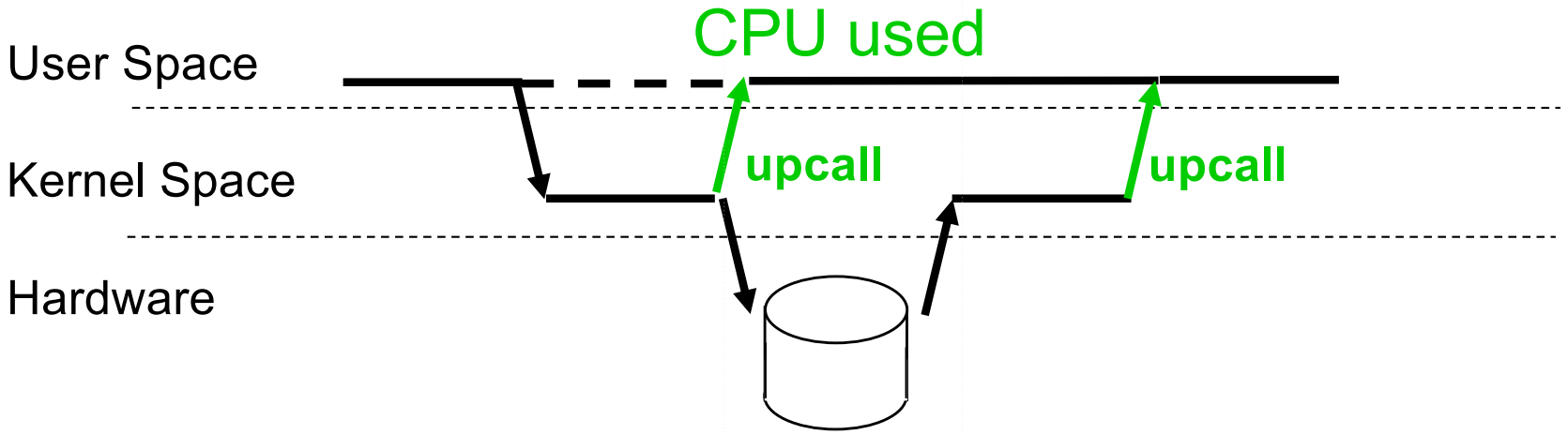
System Call
(Downcall)

Scheduler Activations

Instead of this:



Do this:

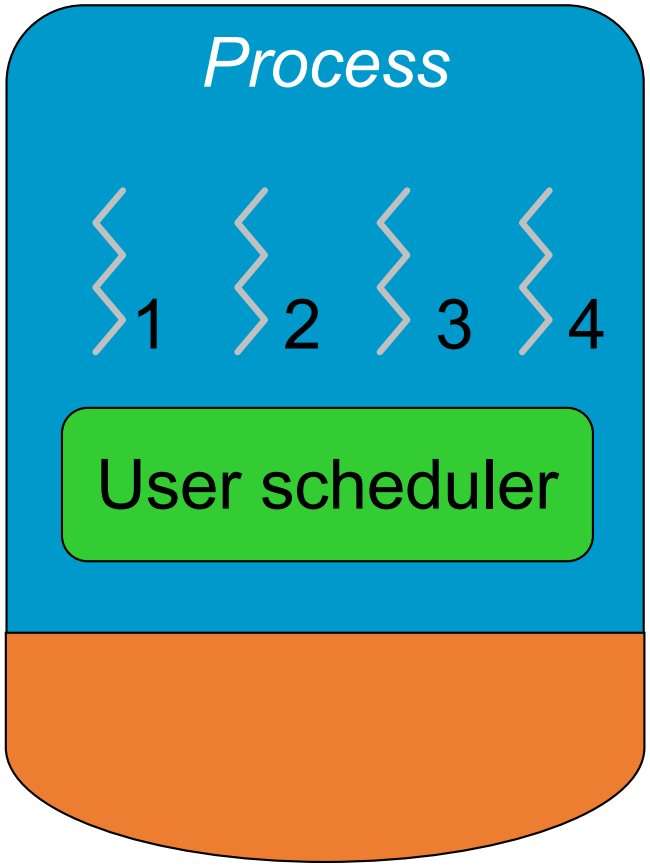


Upcalls to User-level scheduler

- New (processor #)
  - Allocated a new virtual CPU
  - Can schedule a user-level thread
- Preempted (activation # and its machine state)
  - Deallocated a virtual CPU
  - Can schedule one less thread
- Blocked (activation #)
  - Notifies thread has blocked
  - Can schedule another user-level thread
- Unblocked (activation # and its machine state)
  - Notifies a thread has become runnable
  - Must decided to continue current or unblocked thread

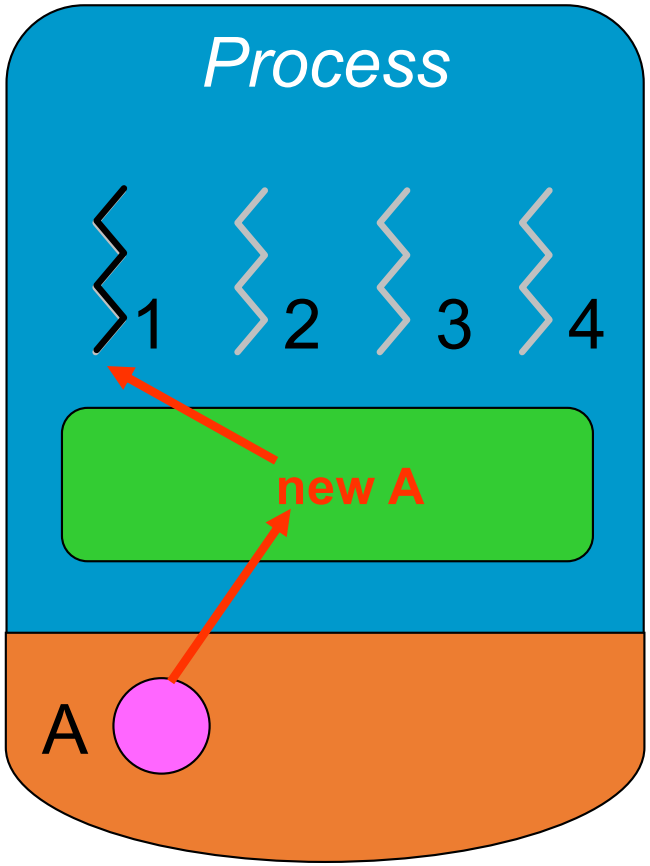
Working principle

Preemption scenario on 2 processors



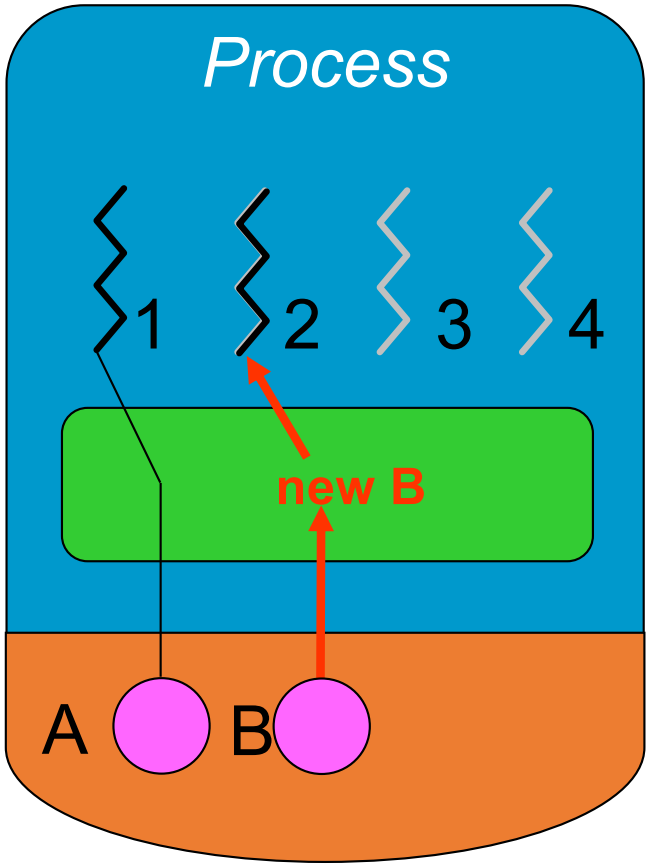
Working principle

Preemption scenario on 2 processors



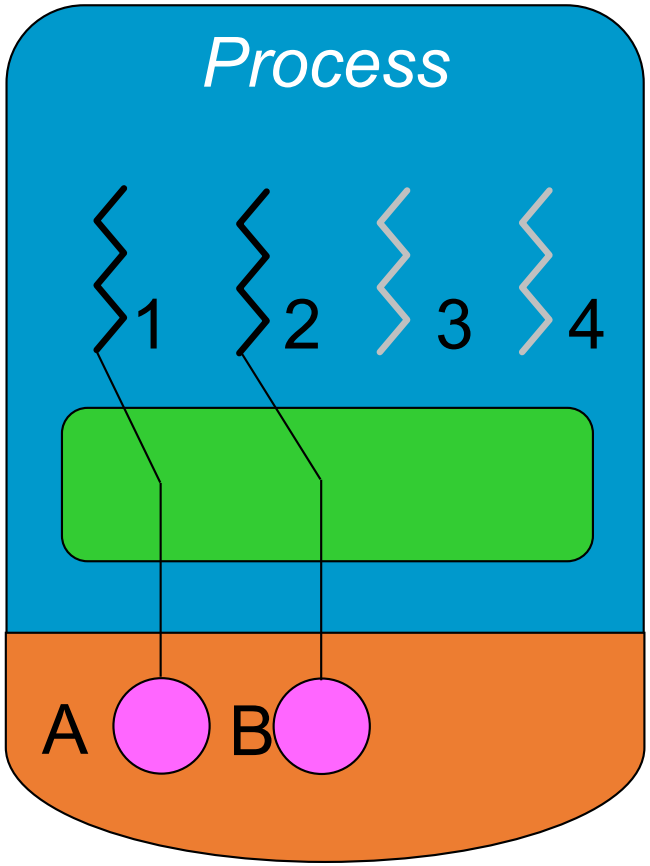
Working principle

Preemption scenario on 2 processors



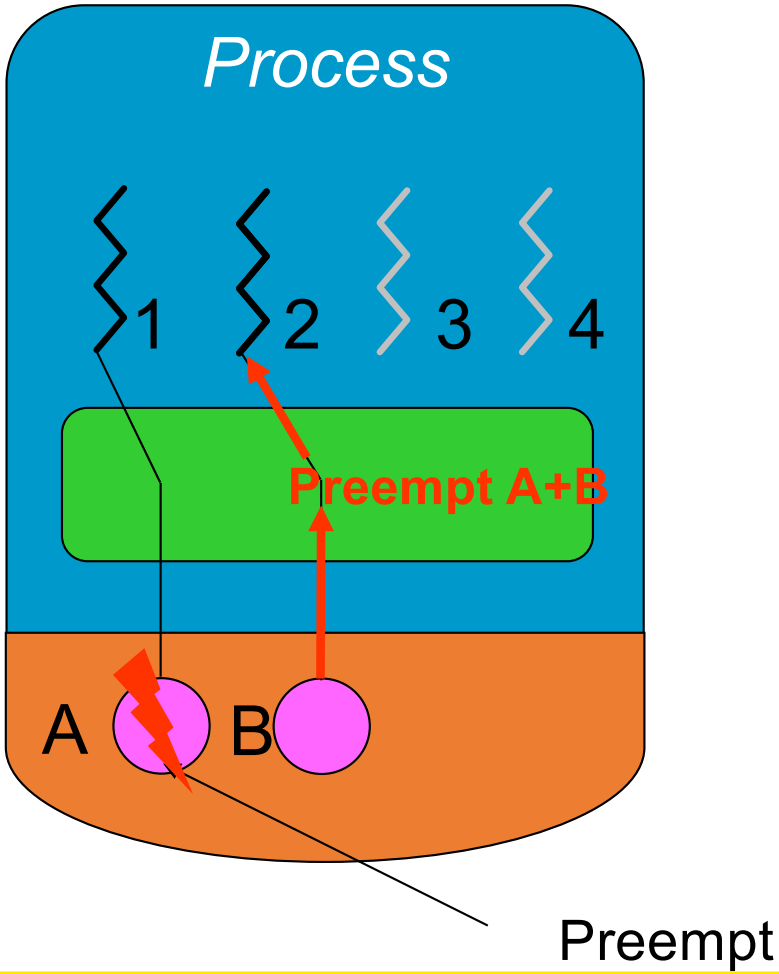
Working principle

- **Preemption** scenario on 2 processors



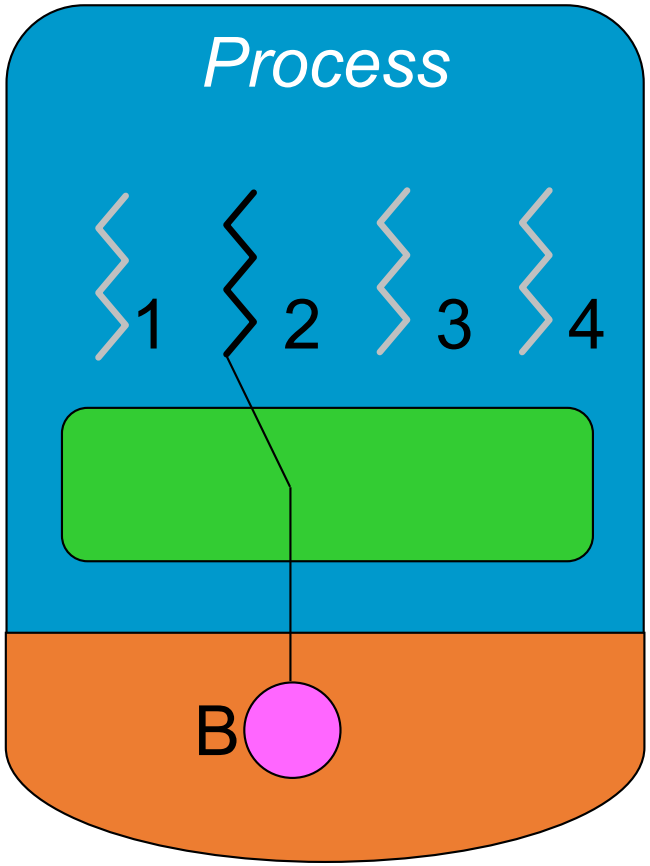
Working principle

Preemption scenario on 2 processors



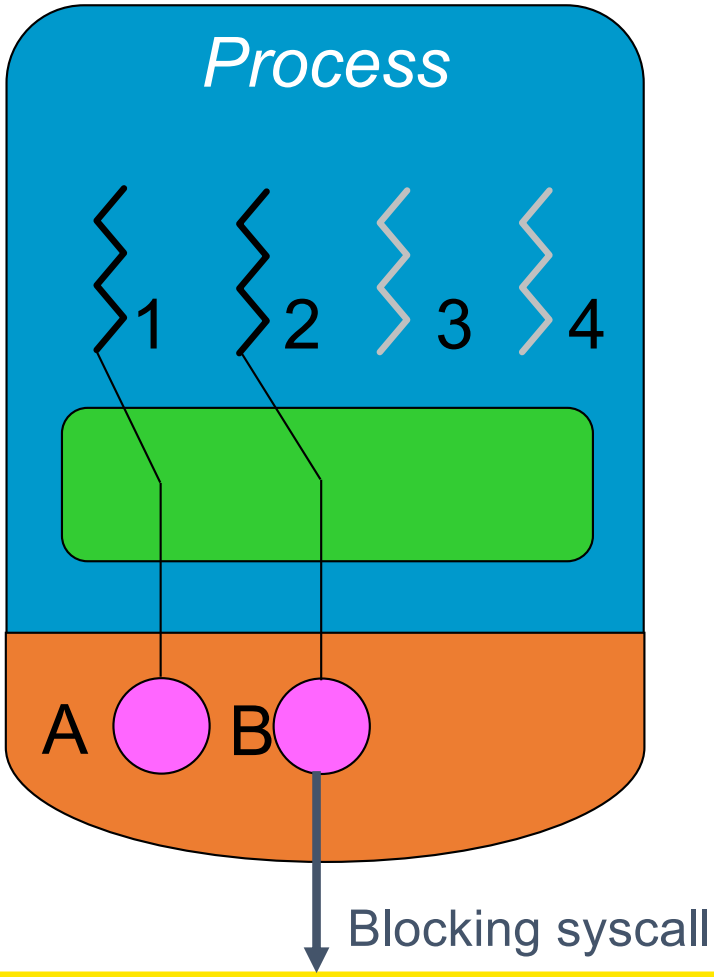
Working principle

Preemption scenario on 2 processors



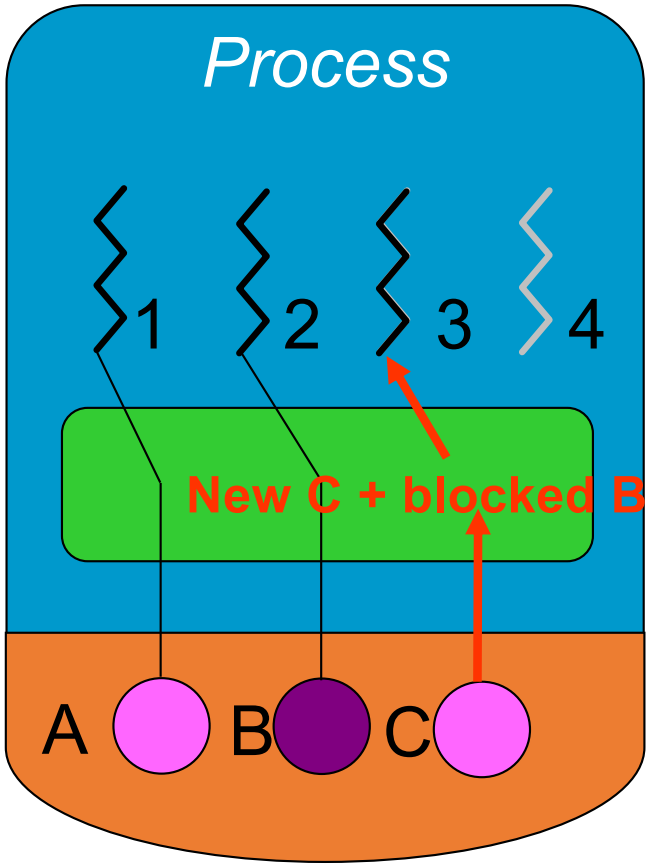
Working principle

Blocking syscall scenario on 2 processors



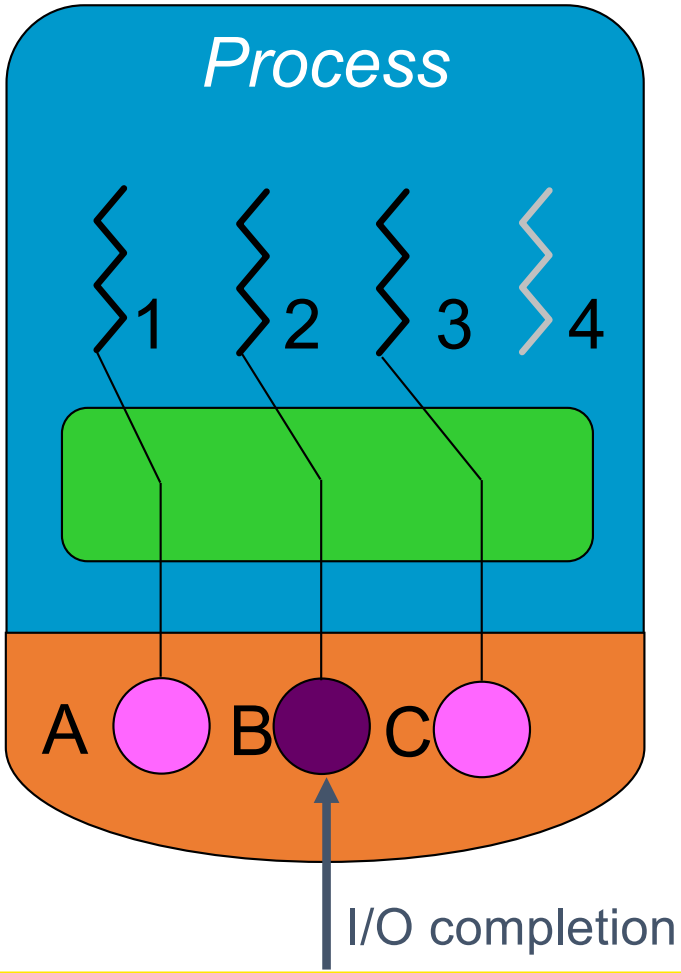
Working principle

Blocking syscall scenario on 2 processors



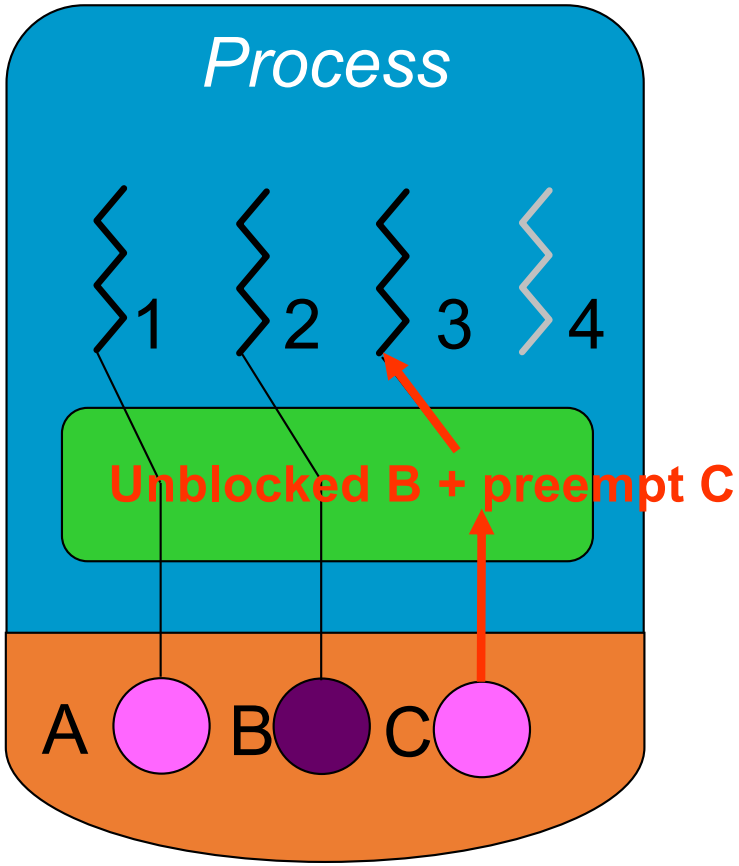
Working principle

Blocking syscall scenario on 2 processors



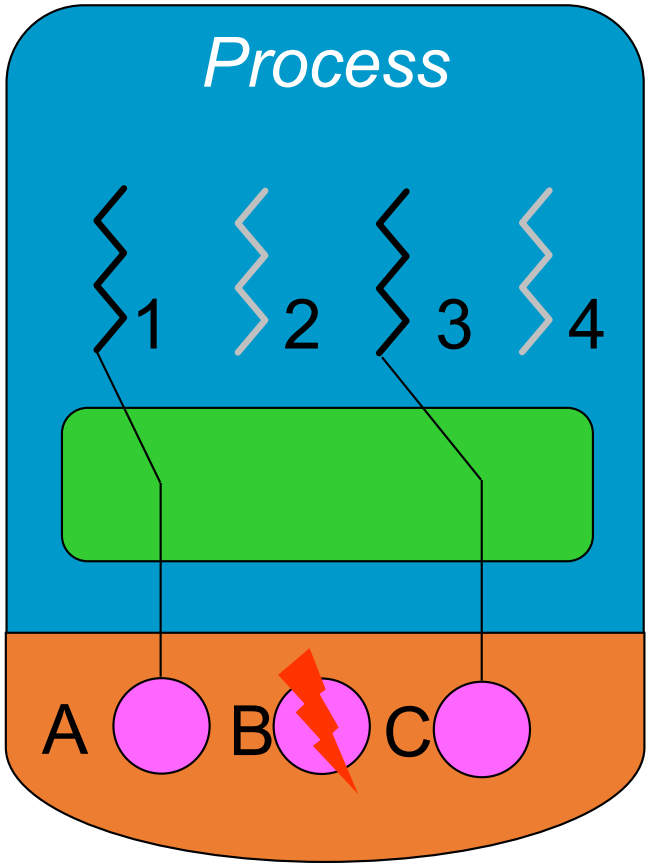
Working principle

Blocking syscall scenario on 2 processors



Working principle

Blocking syscall scenario on 2 processors



Scheduler Activations

- Thread management at user-level
  - Fast
- Real thread parallelism via activations
  - Number of activations (virtual CPUs) can equal CPUs
- Blocking (syscall or page fault) creates new activation
  - User-level scheduler can pick new runnable thread.
- Fewer stacks in kernel
  - Blocked activations + number of virtual 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

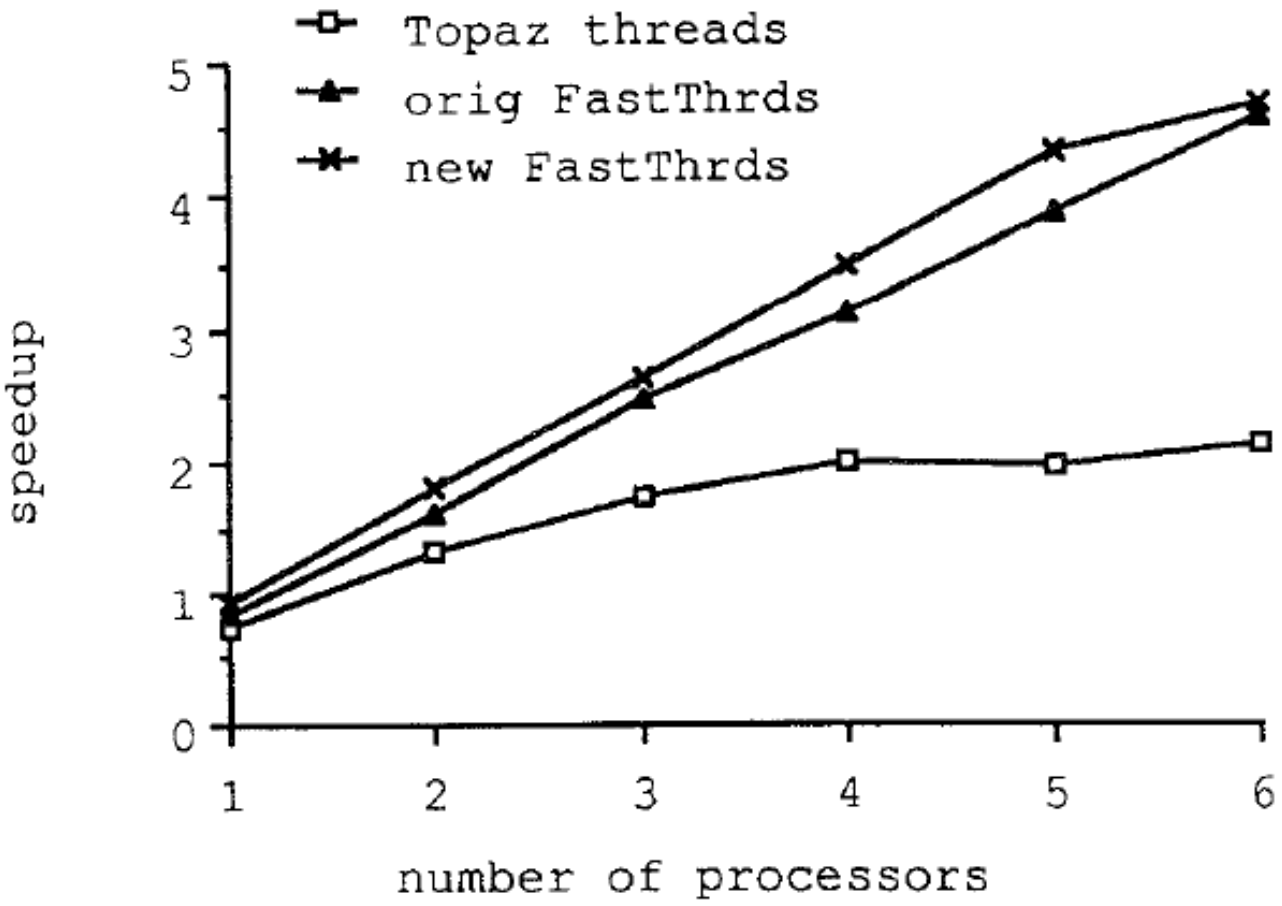


Fig. 2. Speedup of N-Body application versus number of processors, 100% of memory available.

Performance: I/O-Bound

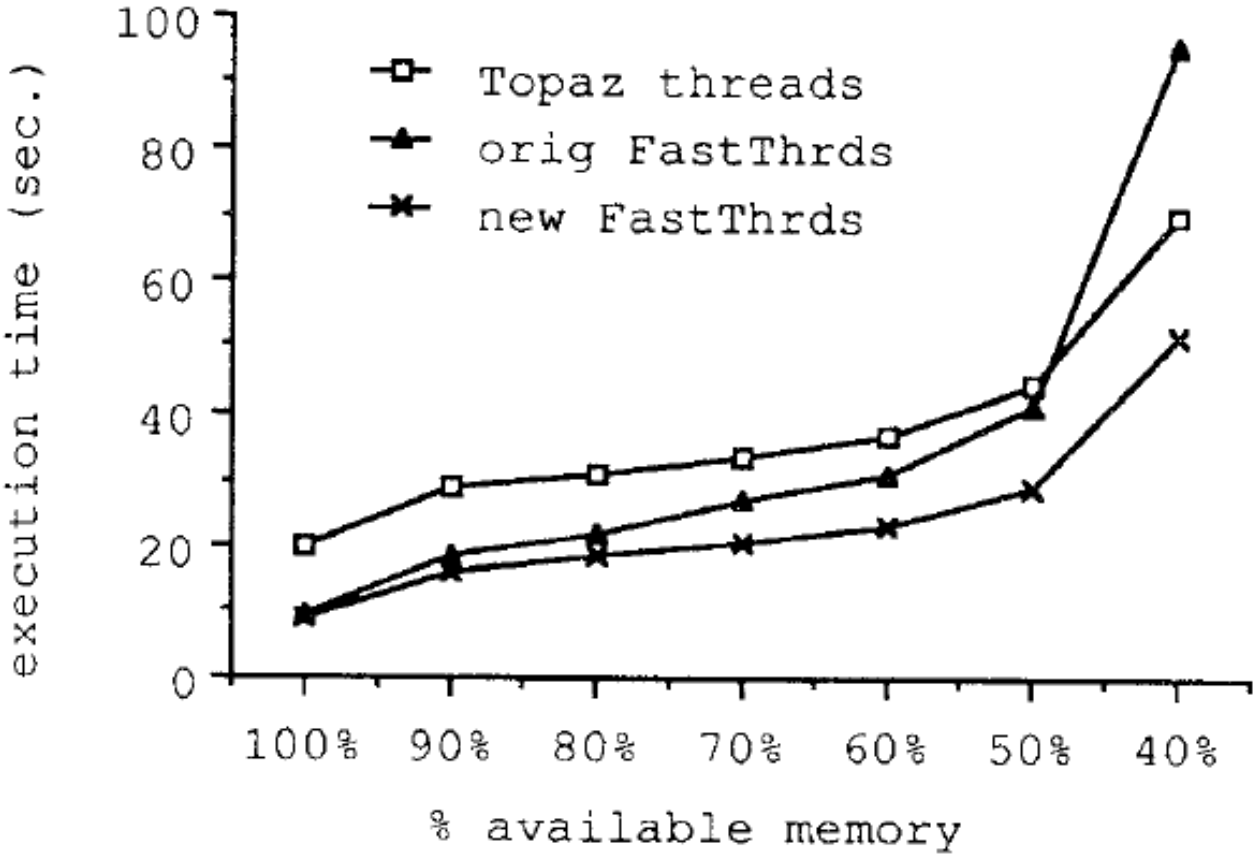


Fig. 3. Execution time of N-Body application versus amount of available memory, 6 processors.

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*
- Linux -> kernel threads