

CSE

Scheduler Activations

With some slides modified from Raymond Namyst, U. Bordeaux

CSE

Handwritten notes:

10 10 10 10 10 1

E	R	ALL	WB	?
D	D	D	D	D
E	R	ALL	WB	?

Annotations: "instr" with arrows pointing to the first two rows. "Sys" with a vertical line on the right. "K" and "M" with a vertical line on the right.

CSE

User-level Threads

User Mode

Kernel Mode

CSE

User-level Threads

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

CSE

Kernel-Level Threads

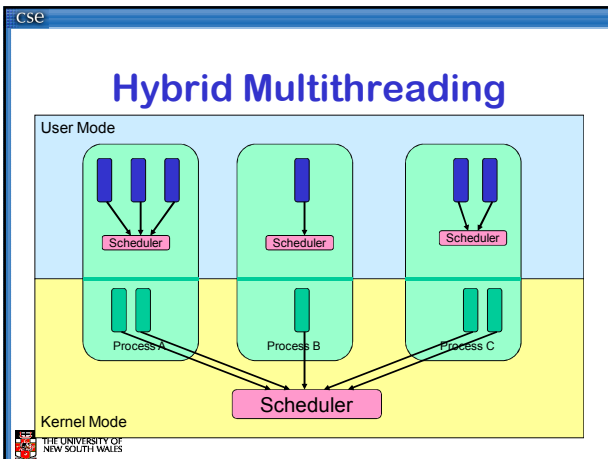
User Mode

Kernel Mode

CSE

Kernel-level Threads

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

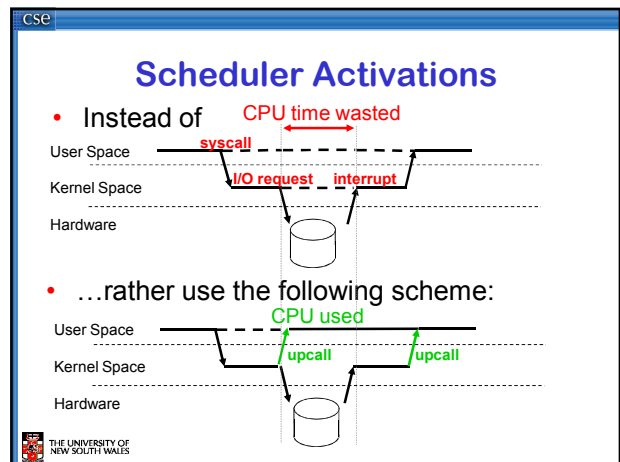


Hybrid Multithreading

- ✓ Can get real thread parallelism on multiprocessor
- ✗ Blocking still 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 the 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



Upcalls to User-level scheduler

- New
 - Allocated a new virtual CPU
 - Can schedule a user-level thread
- Preempted
 - Deallocated a virtual CPU
 - Can schedule one less thread
- Blocked
 - Notifies thread has blocked
 - Can schedule another user-level thread
- Unblocked
 - Notifies a thread has become runnable
 - Must decided to continue current or unblocked thread

Working principle

- Blocking syscall scenario on 2 processors

Cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram shows a blue rounded rectangle labeled "Process" divided into a top light blue section and a bottom dark blue section. Inside the top section are four vertical wavy lines. Below them is a green horizontal bar. A red arrow labeled "syscall" points from the green bar to a pink circle representing a processor. The bottom dark blue section contains another pink circle representing a processor.

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram is similar to the previous one, but the bottom dark blue section now contains two pink circles representing processors. A red arrow labeled "syscall" points from the green bar to the right processor.

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram shows the process on two processors. The green bar is now empty. Two black lines connect the top of the green bar to the two processors. A red arrow labeled "preempt" points from the right processor back to the green bar.

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram is similar to the previous one, but the green bar now contains a red arrow labeled "preempt" pointing to the right processor. A black arrow labeled "Preempt" points from the right processor to the green bar.

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram shows the process on a single processor. The green bar is empty. A black line connects the top of the green bar to the processor. The bottom dark blue section contains one pink circle representing a processor.

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram shows the process on two processors. The green bar is empty. A black line connects the top of the green bar to the right processor. The bottom dark blue section contains two pink circles representing processors. A black arrow labeled "Blocking syscall" points to the right processor.

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

New + blocked

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

I/O completion

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

Unblocked

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Scheduler Activations

- Thread management at user-level
 - Fast
- Real thread parallelism via activations
 - Number of activations (virtual CPU) 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

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Adoption

- Adopters
 - BSD "Kernel Scheduled Entities"
 - K42
 - Digital UNIX
 - Solaris
 - Mach
- Linux -> kernel threads

THE UNIVERSITY OF NEW SOUTH WALES