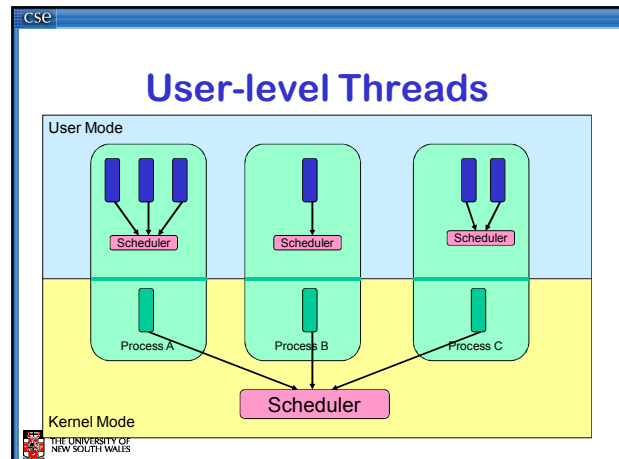


CSE

Thread Models Recap

With some slides modified from Raymond Namyst, U. Bordeaux

THE UNIVERSITY OF NEW SOUTH WALES

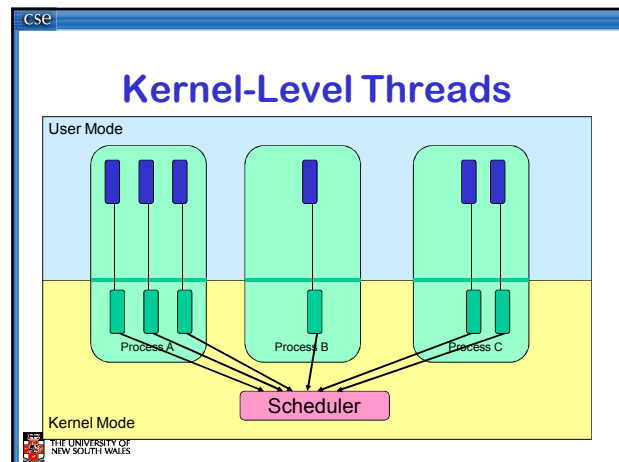


CSE

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

THE UNIVERSITY OF NEW SOUTH WALES

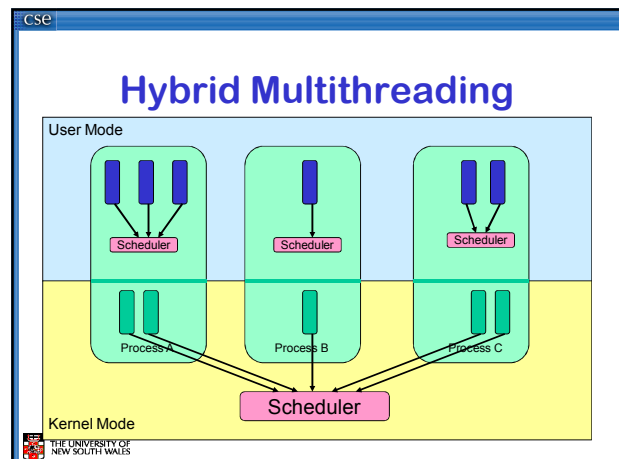


CSE

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

THE UNIVERSITY OF NEW SOUTH WALES




Hybrid Multithreading

- ✓ Can get real thread parallelism on multiprocessor
- ✗ Blocking still a problem!!!



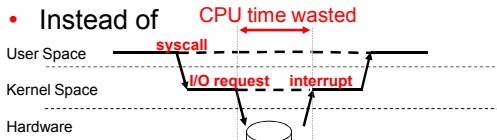
Scheduler Activations

- 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

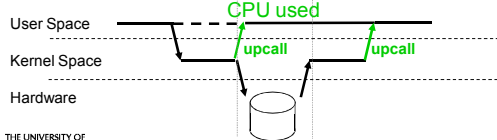



Scheduler Activations

- Instead of CPU time wasted




- ...rather use the following scheme:

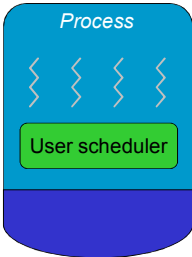

Upcalls to User-level scheduler

- 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



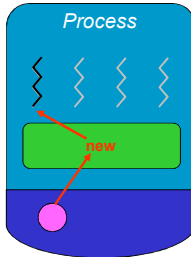

Working principle

- Blocking syscall scenario on 2 processors

Working principle

- Blocking syscall scenario on 2 processors

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

Preempt

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

Blocking syscall

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Working principle

- Blocking syscall scenario on 2 processors

Process

Now blocked

THE UNIVERSITY OF NEW SOUTH WALES

cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram shows a blue rounded rectangle labeled "Process". Inside, there are two green horizontal bars representing threads. Below the threads are two pink circles representing processors. Lines connect the threads to the processors. A label "I/O completion" with an arrow points to the right 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 right processor is now highlighted in red. A red arrow points from the word "Unblocked" to the right processor.

THE UNIVERSITY OF NEW SOUTH WALES

cse

Working principle

- Blocking syscall scenario on 2 processors

The diagram is similar to the previous ones, but the right processor is now highlighted in red with a lightning bolt symbol, indicating a new activation.

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