An Introduction to Concurrent Programming

Richard Tsai

Microsoft Student Technology Club of SYSU

Overview

Overview

- Response time

Overview

- Response time
- Parallelize IO operations

Overview

- Response time
- Parallelize IO operations
- Make full use of hardware resources

Overview

- Response time
- Parallelize IO operations
- Make full use of hardware resources
- How does the OS schedule tasks?

Overview

- Response time
- Parallelize IO operations
- Make full use of hardware resources
- How does the OS schedule tasks?
- “Concurrent” v.s. “Parallel”

Process-level concurrency

Process-level concurrency

- Pros and cons

Process-level concurrency

- Pros and cons
- Implementations

Process-level concurrency

- Pros and cons
- Implementations
  - The `fork` and `waitpid` system calls

Process-level concurrency

- Pros and cons
- Implementations
  - The `fork` and `waitpid` system calls
  - Python's `multiprocessing` package

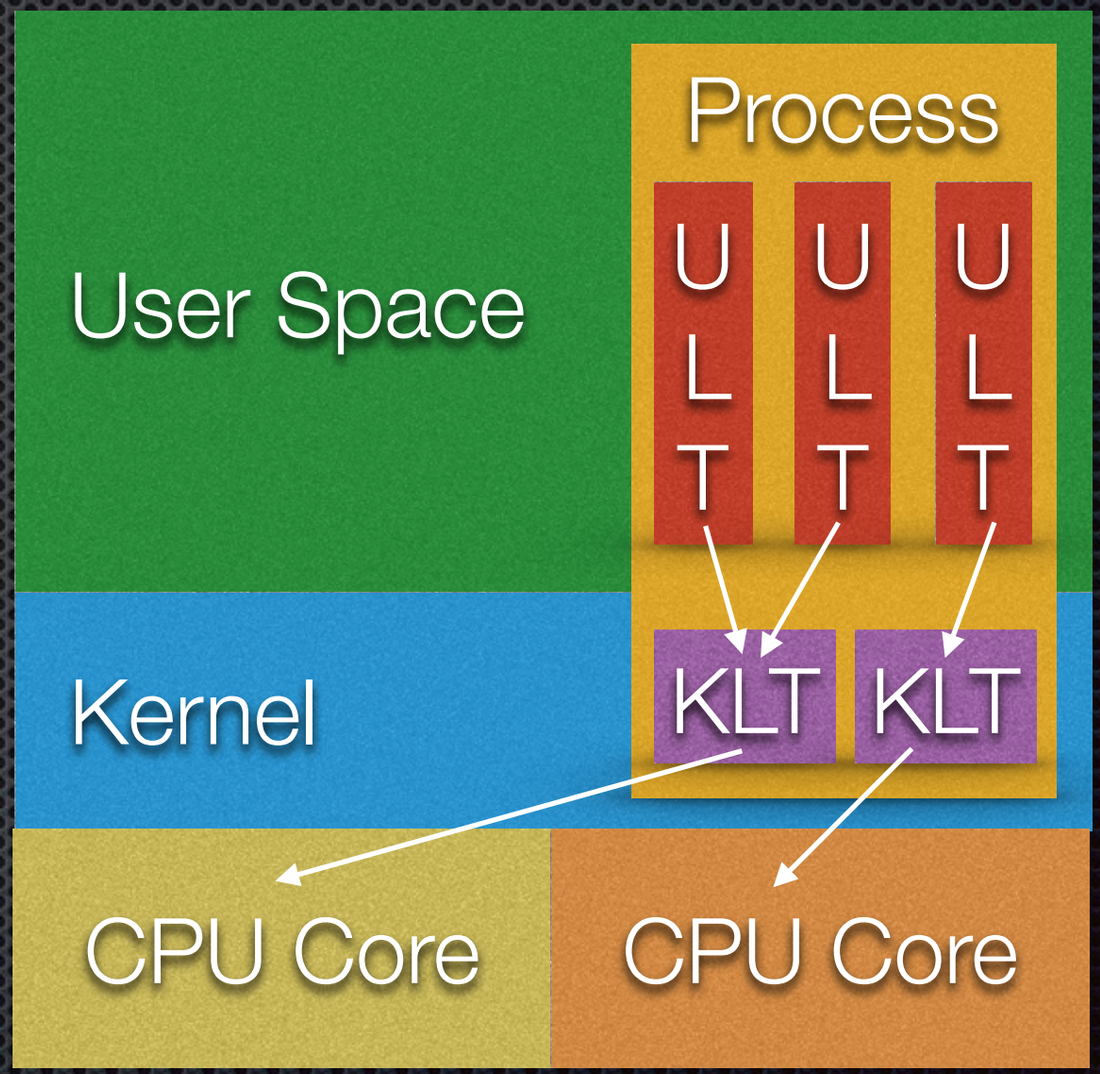
Process-level concurrency

- Pros and cons
- Implementations
  - The `fork` and `waitpid` system calls
  - Python's `multiprocessing` package
  - MPI implementations

Multithreading

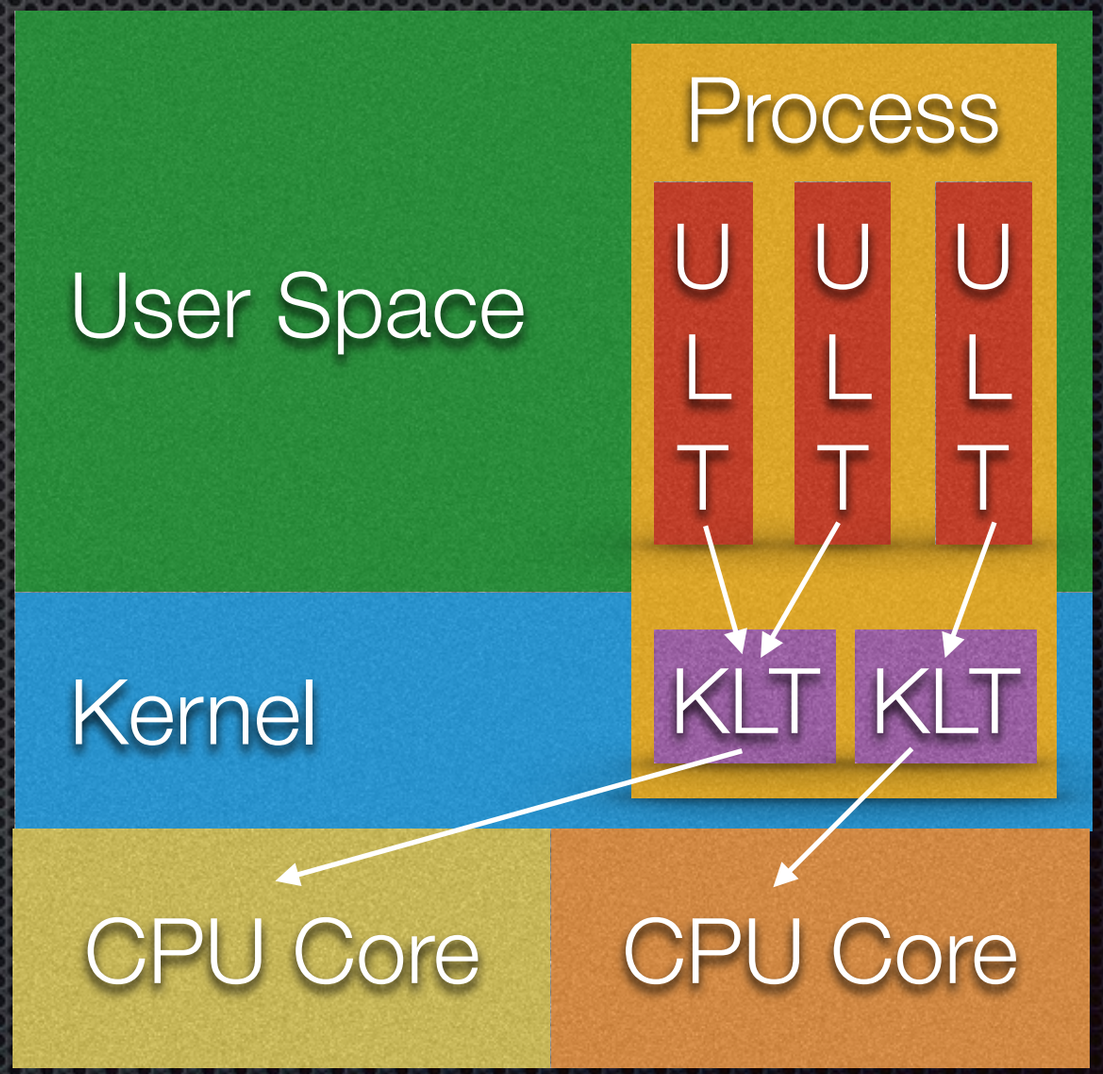
Multithreading

- Process v.s. thread



Multithreading

- Process v.s. thread
- KLT v.s. ULT



Multithreading

- 1:1 Multithreading (Pure KLT)

Multithreading

- 1:1 Multithreading (Pure KLT)
  - Most of the **`pthread`** implementations (e.g. NPTL in Linux)

Multithreading

- 1:1 Multithreading (Pure KLT)
  - Most of the `pthread` implementations (e.g. NPTL in Linux)
  - Simple wrappers for kernels' threading facilities (e.g. Python's `threading` package)

Multithreading

- N:1 Multithreading (Pure ULT)

Multithreading

- N:1 Multithreading (Pure ULT)
  - **Coroutine**

Multithreading

- N:1 Multithreading (Pure ULT)
  - **Coroutine**
  - Implementations

Multithreading

- N:1 Multithreading (Pure ULT)
  - **Coroutine**
  - Implementations
    - `Boost.Coroutine` for C++

Multithreading

- N:1 Multithreading (Pure ULT)
  - **Coroutine**
    - Implementations
      - `Boost.Coroutine` for C++
      - Python's generator and `greenlet` for Python

Multithreading

- N:1 Multithreading (Pure ULT)
  - **Coroutine**
    - Implementations
      - `Boost.Coroutine` for C++
      - Python's generator and `greenlet` for Python
      - Ruby's `fiber`

Multithreading

- M:N Multithreading (Hybrid threading)

Multithreading

- M:N Multithreading (Hybrid threading)
  - Features

Multithreading

- M:N Multithreading (Hybrid threading)
  - Features
  - Implementations

Multithreading

- M:N Multithreading (Hybrid threading)
  - Features
  - Implementations
    - Erlang processes

Multithreading

- M:N Multithreading (Hybrid threading)
  - Features
  - Implementations
    - Erlang processes
    - Goroutines

Multithreading

- High level threading libraries/standards

Multithreading

- High level threading libraries/standards
  - OpenMP
  - Intel Threading Building Blocks
  - Apple Grand Central Dispatch

Some issues…

Some issues…

- Shared resources

Some issues…

- Shared resources
  - Race conditions

Some issues…

- Shared resources
  - Race conditions
  - Concurrency control

Some issues…

- Shared resources
  - Race conditions
  - Concurrency control
    - Atomic operations (XCHG, CAS, Fetch-and-X, …)

Some issues…

- Shared resources
  - Race conditions
  - Concurrency control
    - Atomic operations (XCHG, CAS, Fetch-and-X, …)
    - Locking (mutex, semaphore, condition variable, …)

Some issues…

- Shared resources
  - Race conditions
  - Concurrency control
    - Atomic operations (XCHG, CAS, Fetch-and-X, …)
    - Locking (mutex, semaphore, condition variable, …)
    - Multiversion Concurrency Control (MVCC)

Some issues…

- Communications between concurrent components

Some issues…

- Communications between concurrent components
- Shared memory

Some issues…

- Communications between concurrent components
  - Shared memory
  - Message passing

Summary

	Weight	Scheduling Policy	Utilize Multicores	Thread Safety	Other Features
Process	Most Heavy	Cooperative & preemptive	Yes	Less important	Dispatch across networks
KLT	Heavy	Cooperative & preemptive	Yes	Important	
ULT (Coroutine)	Light	Cooperative	No	Less important	Must be mapped onto KLT
Hybrid	Light	-	Yes	-	Separate logic and runtime
High level threads	-	-	Yes	Less important	Task-based

Q&A

Thanks!