

CS533 Concepts of Operating Systems

Class 2

The Duality of Threads and Events

Which style is used in modern OSs?

- ❑ Are modern operating systems (such as Linux) written in an event-based or multi-threaded style?
- ❑ How does event-based programming relate to interrupt handling?
- ❑ Where is the boundary between interrupt handling and thread execution?
 - How does this affect the approaches used for synchronization?

Concurrent Computation vs Concurrent Blocking

- ❑ How is the CPU scheduled:
 - In an event-based system?
 - In a thread-based system?
- ❑ How is live state managed across blocking I/O calls:
 - In an event-based system?
 - In a thread-based system?

Managing Highly Concurrent I/O

- ❑ What is the problem with making thread allocation decisions statically?
 - What is the Slashdot effect?
- ❑ Why is multi-threading not a good match for massive concurrency?
 - Is web service embarrassingly parallel?
 - What is the problem with the thread-per request model?
 - Why does the event handling model help?

Questions

- ❑ What is a thread pool?
- ❑ Why do the following techniques help during heavy load?
 - Thread pool resizing
 - Event batching
 - Adaptive load shedding
- ❑ Why does pipeline parallelism scale well?
 - Thread per stage vs thread per request
- ❑ What does it mean for a service to be “well conditioned”?

Duality

- ❑ What really happens underneath the abstraction?
 - For synchronous procedure calling and message passing
 - For asynchronous procedure calling and message passing
- ❑ Fundamental tasks:
 - *execute new instruction stream*
 - *exchange data*
- ❑ What is the costs of pushing state onto the stack and popping it off, vs the cost of allocating copying then freeing message memory?

Optimizations Discussed in Future Classes

- ❑ How could you optimize context switch costs for local message passing?
- ❑ How could you optimize data movement costs for local message passing?
 - What conventions would you need to follow to avoid needing synchronization?
 - How does this approach compare to local state management in procedure calling?
- ❑ How might you further optimize pointer passing?
 - ... and what does this remind you of? Hint: SP.

Duality

- Is threading just an automated “pattern” of context switch and data exchange that is done manually in event models?
- Do the two models have equal memory consumption?
 - Is over-allocation of memory inevitable in thread-based systems?
 - *If not, how can you avoid it ... and at what cost?*