CS533 Concepts of Operating Systems

Class 2

Thread vs Event-Based Programming
Questions

- Why are threads “lightweight”?
- Why block instead of spin waiting for a mutex?
- If a mutex is a resource scheduling mechanism
  - What is the resource being scheduled?
  - What is the scheduling policy and where is it defined?
- What is coarse-grain locking?
  - What effect does it have on program complexity?
  - What effect does it have on performance?
Questions

- What is “lock contention”? 
  - Why is it worse on multiprocessors than uniprocessors?
  - What is the solution? ... and its cost?

- What else might cause performance to degrade when you use multiple threads?

- Why is multithreaded programming hard?
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”?