CPU Scheduling - Multicore
Reading

Silberschatz et al: Chapter 5.5
Multiple-Processor Scheduling

- So far, we’ve only dealt with a single processor

- CPU scheduling more complex when multiple CPUs are available due to load sharing.

- No single best solutions - no great surprise.
Multiple-Processor Scheduling

- Asymmetric multiprocessing (master)
- Here there is one processor that makes the decisions for
  - Scheduling, I/O processing, system activities
  - Other processor(s) execute only user code.
- This is a simple approach because only one processor accesses the system data structures, so sharing is not an issue with other processors.
Multiple-Processor Scheduling

- Symmetric Multiprocessing (SMP)

- Here, each processor is self-scheduling.

- Share a common ready queue or each processor may have its own private queue of ready processes.
Multiple-Processor Scheduling

- Whether there is a common ready queue or private ready queues, we have a scheduler for each processor that examines the ready queue and dispatches the CPU to a specific process for execution.

- Clearly, there are sharing issues, here, since each processor may update this common data structure or try to access a specific PCB in a queue ...

- Most all modern operating systems support SMP including Windows XP, Solaris, Linux, and Mac OS X.
MultiProcessor Scheduling

- **Challenges**
  - Processor affinity
  - Load balancing
Processor Affinity

- Processors have a cache
- What happens to the cache when a process leaves the running state
  - The contents of cache memory must be invalidated
  - Why? The next time the process is scheduled it may be scheduled on another processor
- Try to avoid migration (processor affinity)
Load Balancing

- Keep the workload balanced among all processors
- This is especially a problem in SMP systems
Summary

- Reviewed multicore