CSCI315 – Operating Systems Design

Department of Computer Science

Bucknell University

Multi-Processor Scheduling and Real-Time Scheduling

Ch 5.5, 5.6

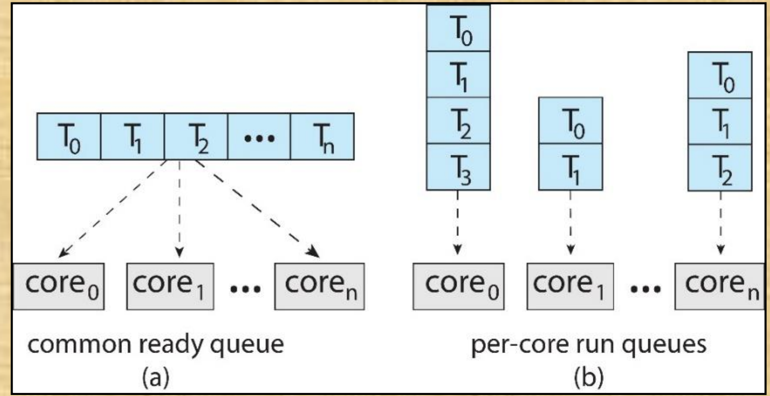
This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.
Xiannong Meng, Fall 2021.

Multiple-Processor Scheduling

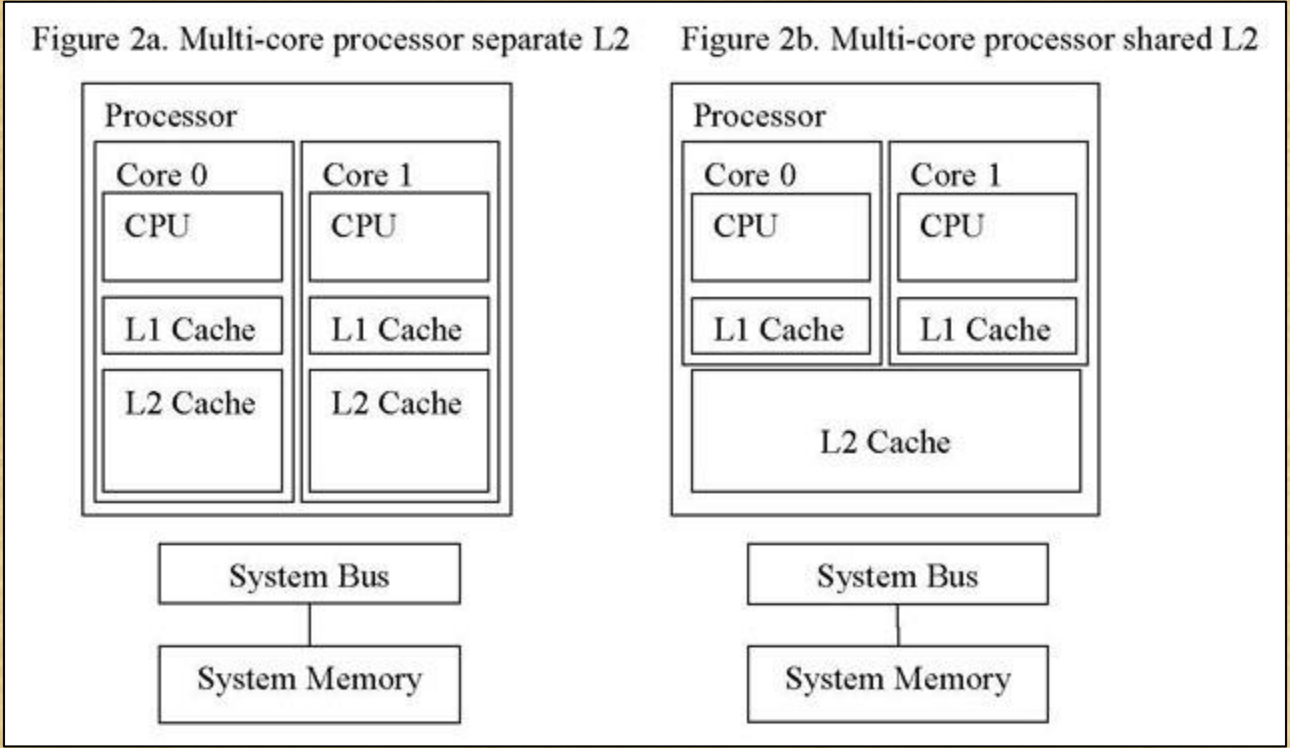
- CPU scheduling is more complex when multiple CPUs are available
- Multiprocessor may be any one of the following architectures:
  - Multicore CPUs
  - Multithreaded cores
  - NUMA (Non-uniform memory access) systems
  - Heterogeneous multiprocessing

Multiple-Processor Scheduling

- Symmetric multiprocessing (SMP) is where each processor is self scheduling.
- All threads may be in a common ready queue (a)
- Each processor may have its own private queue of threads (b)
- Queuing theory implications?



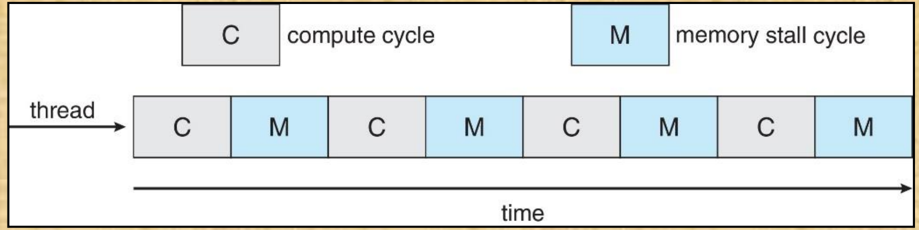
Multi-core Processor Architecture



https://www.drdobbs.com/development-and-optimization-techniques/212600040

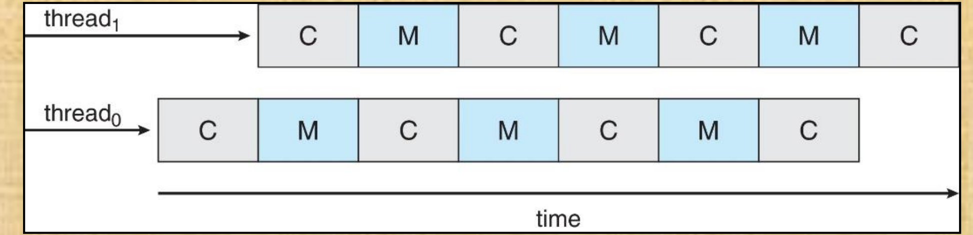
Multicore Processors

- Recent trend to place multiple processor cores on same physical chip
- Faster and consumes less power
- Multiple threads per core also growing
  - Takes advantage of memory stall to make progress on another thread while memory retrieve happens



Multithreaded Multicore System

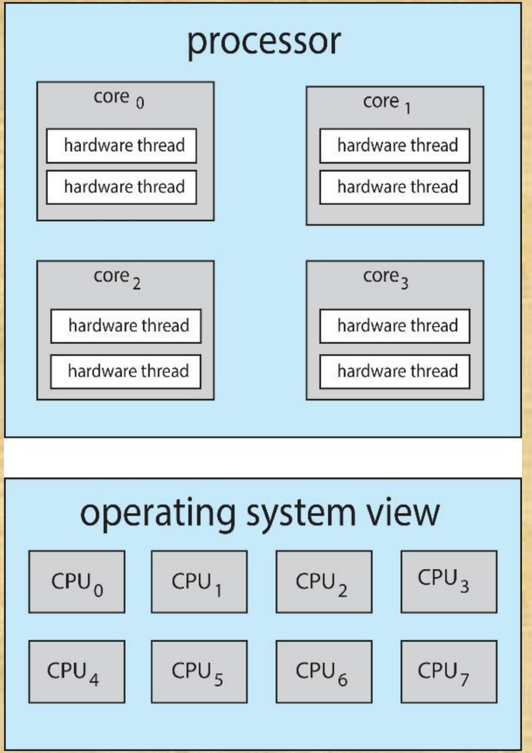
- Each core has > 1 hardware threads.
- If one thread has a memory stall, switch to another thread!



Multithreaded Multicore System

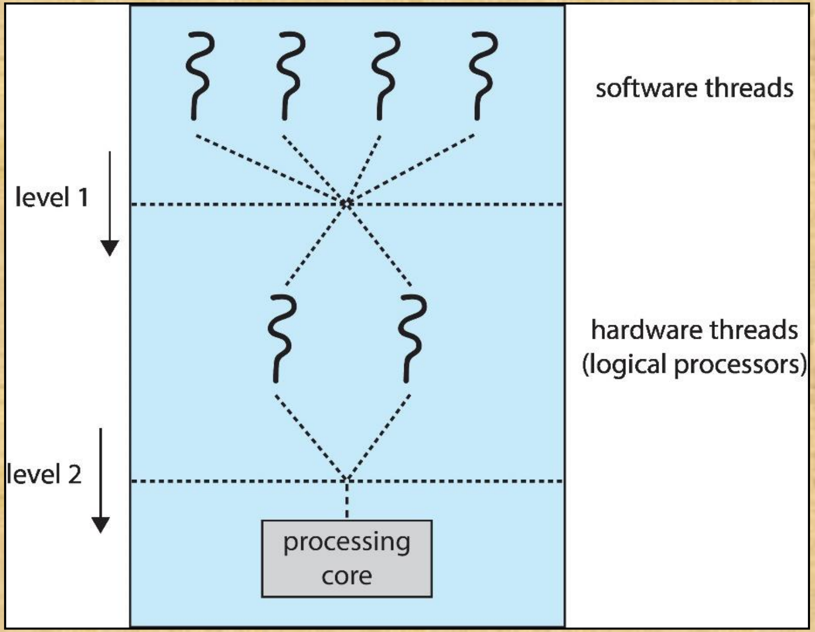
- **Chip-multithreading** (CMT) assigns each core multiple hardware threads. (Intel refers to this as **hyperthreading**.)

- On a quad-core system with 2 hardware threads per core, the operating system sees 8 logical processors.



Multithreaded Multicore System

- Two levels of scheduling:
    1. The operating system deciding which software thread to run on a logical CPU
    2. How each core decides which hardware thread to run on the physical core.



Multiple-Processor Scheduling – Load Balancing

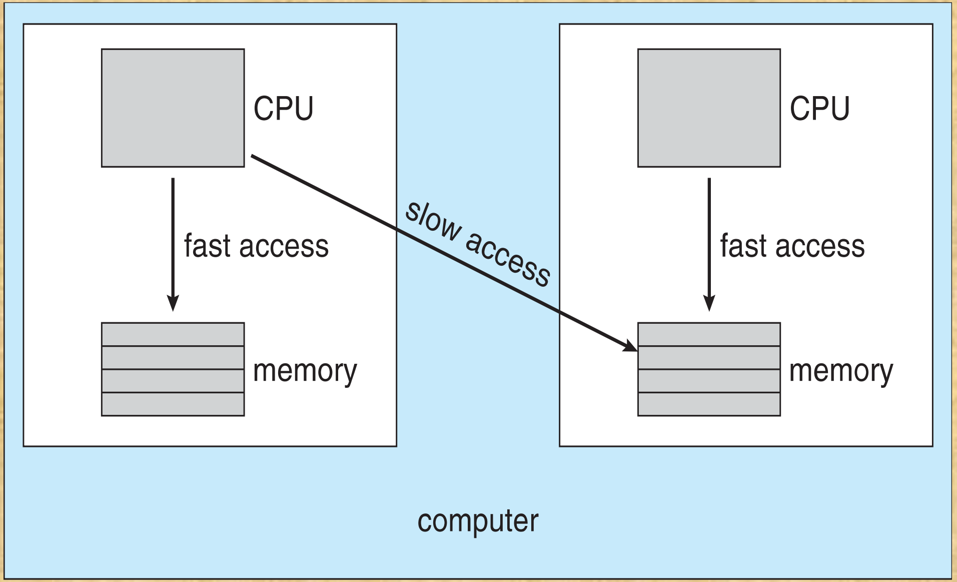
- If SMP, need to keep all CPUs loaded for efficiency
- **Load balancing** attempts to keep workload evenly distributed
- **Push migration** – periodic task checks load on each processor, and if found pushes task from overloaded CPU to other CPUs
- **Pull migration** – idle processors pulls waiting task from busy processor

Multiple-Processor Scheduling – Processor Affinity

- When a thread has been running on one processor, the cache contents of that processor stores the memory accesses by that thread.
- We refer to this as a thread having affinity for a processor (i.e., “processor affinity”)
- Load balancing may affect processor affinity as a thread may be moved from one processor to another to balance loads, yet that thread loses the contents of what it had in the cache of the processor it was moved off of.
- **Soft affinity** – the operating system attempts to keep a thread running on the same processor, but no guarantees.
- **Hard affinity** – allows a process to specify a set of processors it may run on.

NUMA and CPU Scheduling

If the operating system is **NUMA-aware**, it will assign memory closes to the CPU the thread is running on.



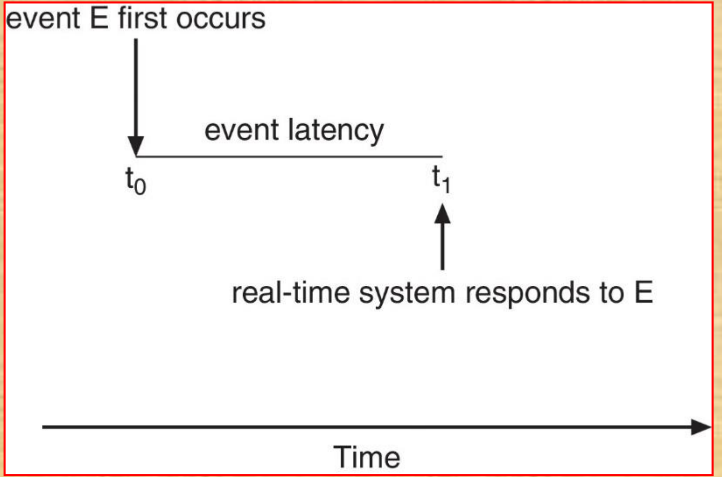
NUMA: Non-uniform memory access (time)

Real-Time CPU Scheduling

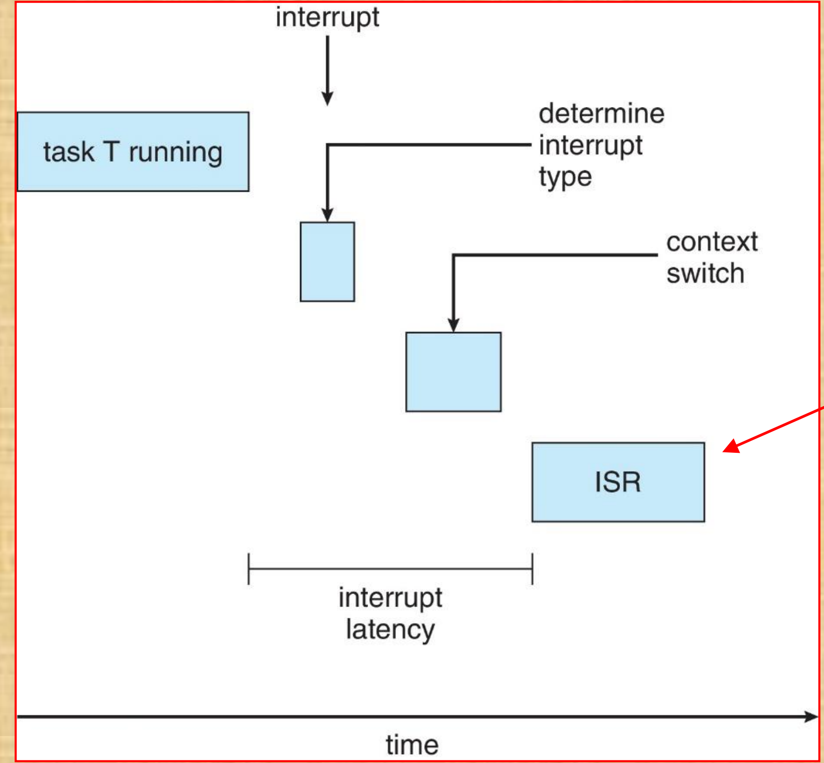
- Can present obvious challenges
- **Soft real-time systems** – Critical real-time tasks have the highest priority, but no guarantee as to when tasks will be scheduled
- **Hard real-time systems – task must be serviced by its deadline**

Real-Time CPU Scheduling

- Event latency – the amount of time that elapses from when an event occurs to when it is serviced.
- Two types of latencies affect performance
  1. **Interrupt latency** – time from arrival of interrupt to start of routine that services interrupt
  2. **Dispatch latency** – time for schedule to take current process off CPU and switch to another



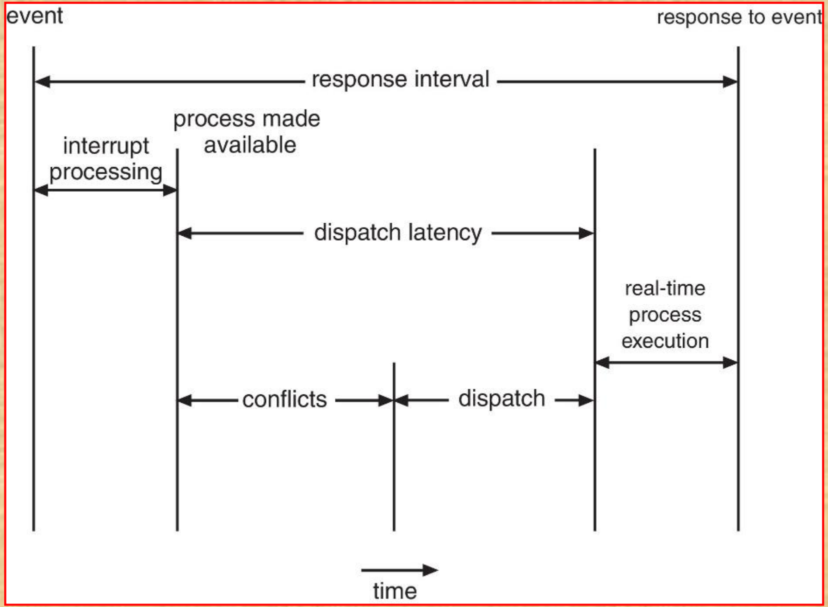
Interrupt Latency



ISR: Interrupt service routine

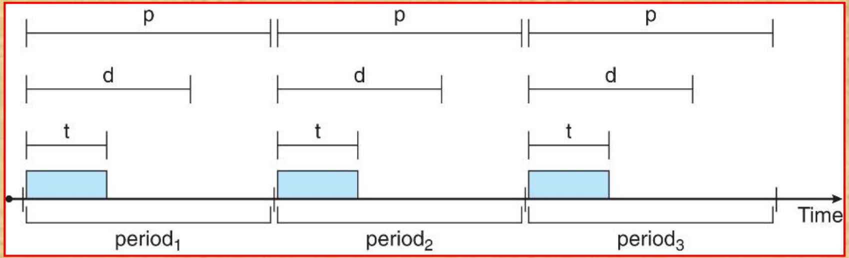
Dispatch Latency

- Conflict phase of dispatch latency:
  1. Preemption of any process running in kernel mode
  2. Release by low-priority process of resources needed by high-priority processes



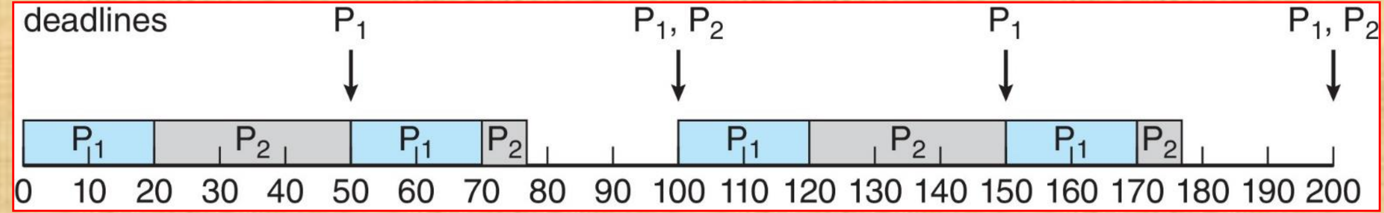
Priority-based Scheduling

- For real-time scheduling, scheduler must support preemptive, priority-based scheduling
  - But only guarantees soft real-time
- For hard real-time must also provide ability to meet deadlines
- Processes have new characteristics: **periodic** ones require CPU at constant intervals
  - processing time *t*, deadline *d*, period *p*
  - $0 \leq t \leq d \leq p$
  - **Rate** of periodic task is $1/p$



Rate Monotonic Scheduling

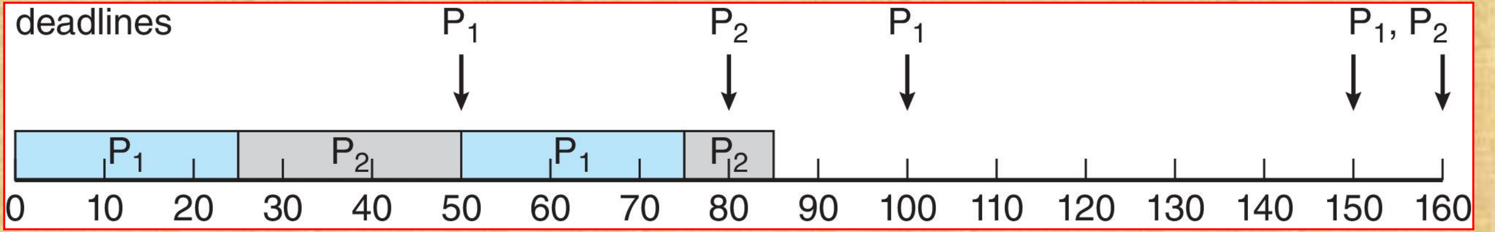
- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority
- Longer periods = lower priority
- P_1 is assigned a higher priority than P_2 when P_1 has a shorter period.



Period(P_1) = 50
Period(P_2) = 100
t_1 = 20, t_2 = 30
d_1 = 50, d_2 = 80
Both meet deadlines

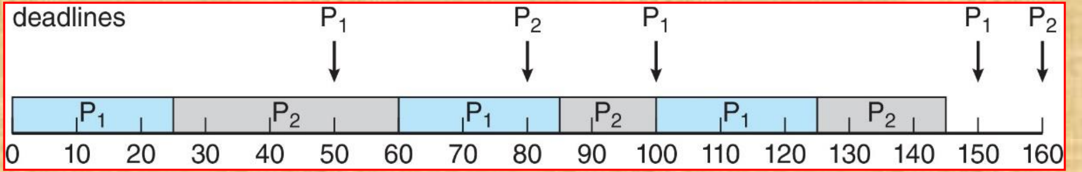
Missed Deadlines with Rate Monotonic Scheduling

- Process P_2 misses finishing its deadline at time 80, if we make t_1 = 25 and t_2 = 25.
- The figure illustrates the idea



Earliest Deadline First Scheduling (EDF)

- Priorities are assigned according to deadlines:
  - The earlier the deadline, the higher the priority
  - The later the deadline, the lower the priority
  - No preemption
- The figure illustrates the idea, if we have
  - $d_1 = 50$ and $d_2 = 80$, $t_1 = 25$ and $t_2 = 35$,
  - Both will miss deadlines.



Proportional Share Scheduling

- T shares are allocated among all processes in the system
- An application receives N shares where $N < T$
- This ensures each application will receive $\boldsymbol{N} / T$ of the total processor time