1878

Western

UNIVERSITY · CANADA

Chapter 5c – Multiprocessor Scheduling

Spring 2023

Overview

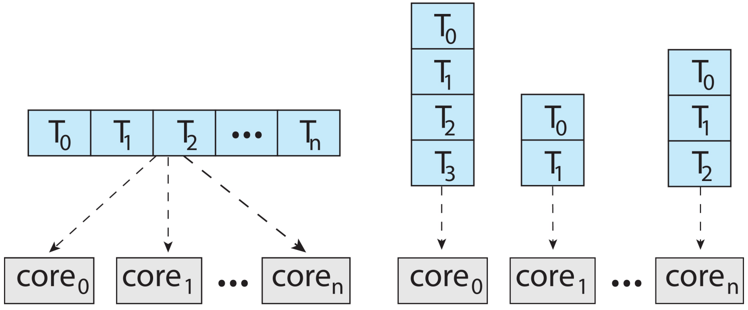
- CPU scheduling more complex when multiple CPUs are available
- "Multiprocess" may be any one of the following architectures:
  - Multicore CPUs (Asymmetric and Symmetric)
  - Multithreaded cores
  - (NUMA systems – Purposely reduce load-balancing and increase processor affinity to avoid long memory access times)
  - (Heterogeneous multiprocessing – Different cores run at different speeds to save battery power on mobile devices)

Asymmetric Multiprocessing

- Primary processor manages all kernel level scheduling decisions
- Other processors just execute user level code
- The primary processor may be a bottleneck. Not used by modern operating systems

Symmetric Multiprocessing

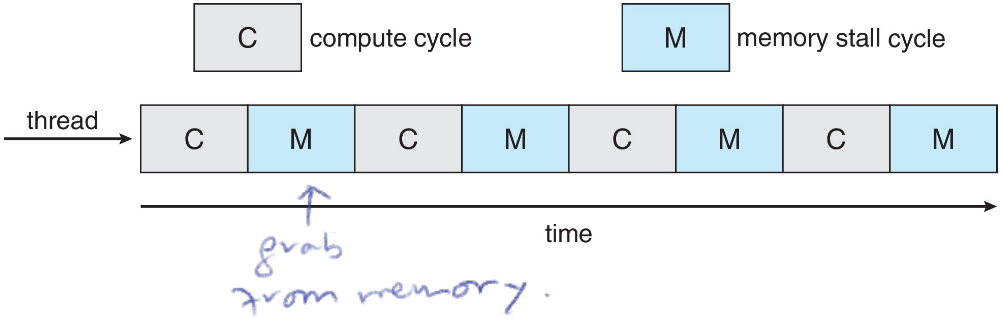
- Each processor is self-scheduling select what's to run next
- The schedulers may choose from
  - A common ready queue
    one ready queue per core
  - A private ready queue for that processor



- A common ready queue could lead to a race condition (two cores could select the same thread at the same time)
- A private queue could lead to an imbalance in the workload
- Most modern operating systems use Symmetric Multiprocessing (SMP) with a private queue per processor

Multicore Processors

- Each core appears to the operating system as a logical CPU
- Multicore CPUs are faster and consume less power
- A processor may need to access memory. This is not slow enough to warrant an I/O wait, but it is slow enough on modern processors to create a **memory stall**

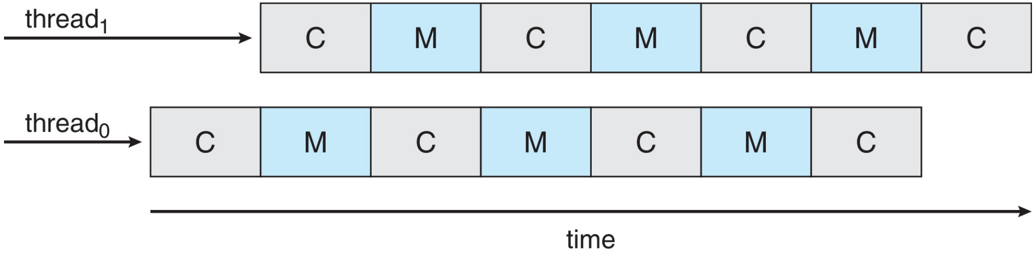


Multithreaded Multicore Processors

- Multiple threads can be handled by a core at once
- 2 or more hardware threads

when doing memory stall, switch to another thread.

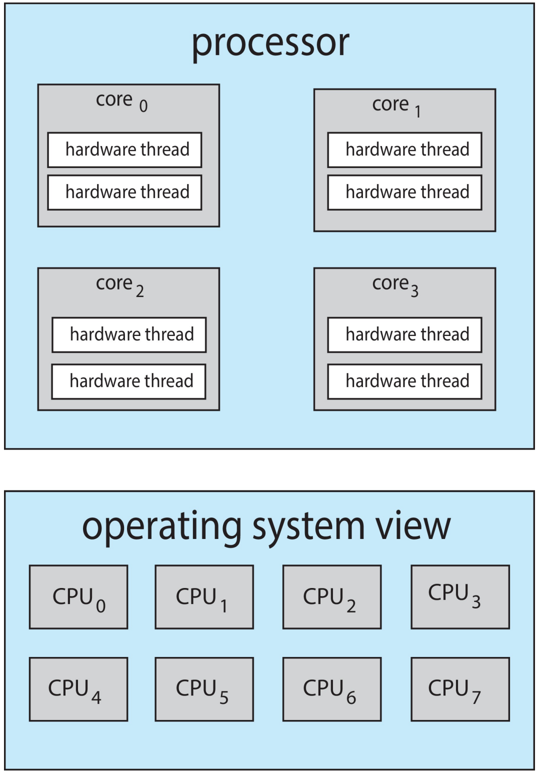
same core



Multithreaded Multicore Processors

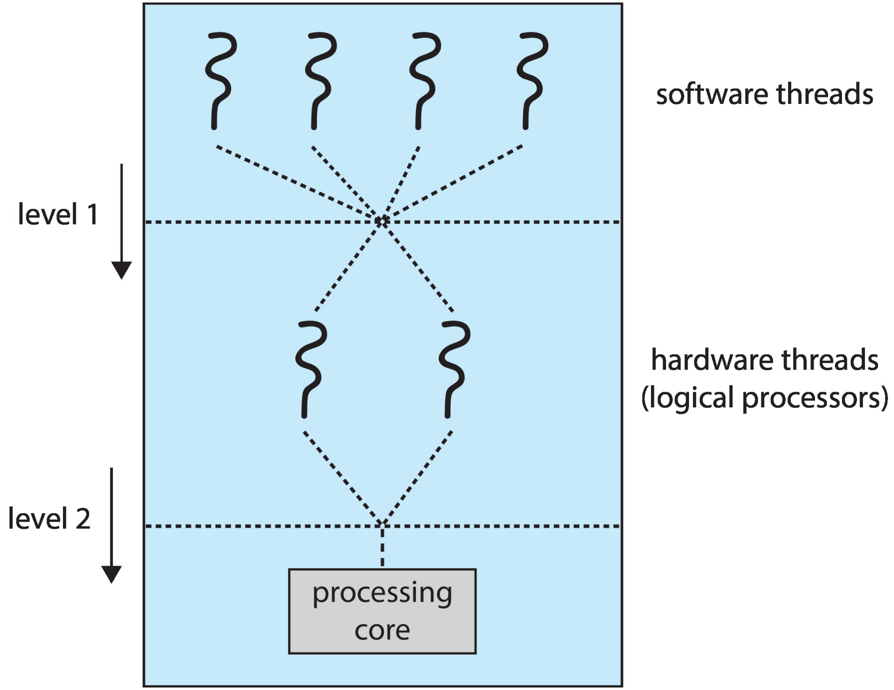
- E.g. 2 hardware threads x 4 cores = 8 logical processors
- Intel i7 supports two threads per core

e.g. i7-5700 .



Multithreaded Multicore Processors

- CPU scheduling algorithms
- Chip has built-in thread scheduling



Load Balancing

- With SMP, it is important to keep the workload balanced
- Only necessary with private ready queues
- Two approaches
    - **Push migration** – periodic task checks load on each processor, and if an issue is found, pushes task from overloaded CPU to other CPUs
    - **Pull migration** – idle processors pulls waiting task from busy processor
    - Both are possible and most operating systems use both

Processor Affinity

- Threads populate the cache of the processor they run on
- If the thread migrates to another processor, it needs to invalidate and repopulate the cache on that processor, which is costly
- Therefore, most operating systems will try to avoid migrating threads across processors
- This is another reason to use a private ready queue instead of a common one
- **Soft affinity** – the operating system attempts to keep a thread running on the same processor, but no guarantees. (Automatic)
- **Hard affinity** – allows a thread to specify a set of processors it may run on. (Manual)

Processor Affinity

- Load-balancing counteracts the benefits of processor affinity
  - Keeping threads on the same processor can improve speed for that thread
  - Moving threads to other processors can improve speed in aggregate
- A good scheduler algorithm needs to keep these concepts in balance

1878

Western

UNIVERSITY · CANADA