Real-Time Programming

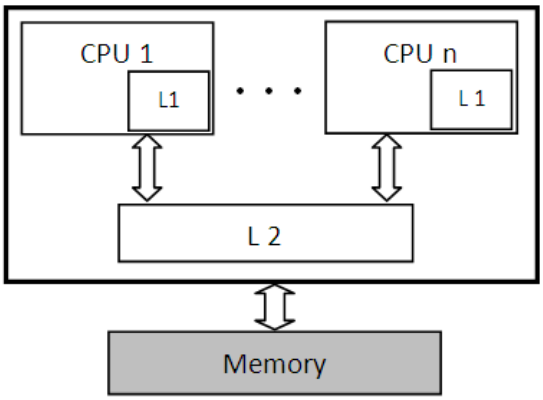
Farhang Nemati

Spring 2016

Introduction

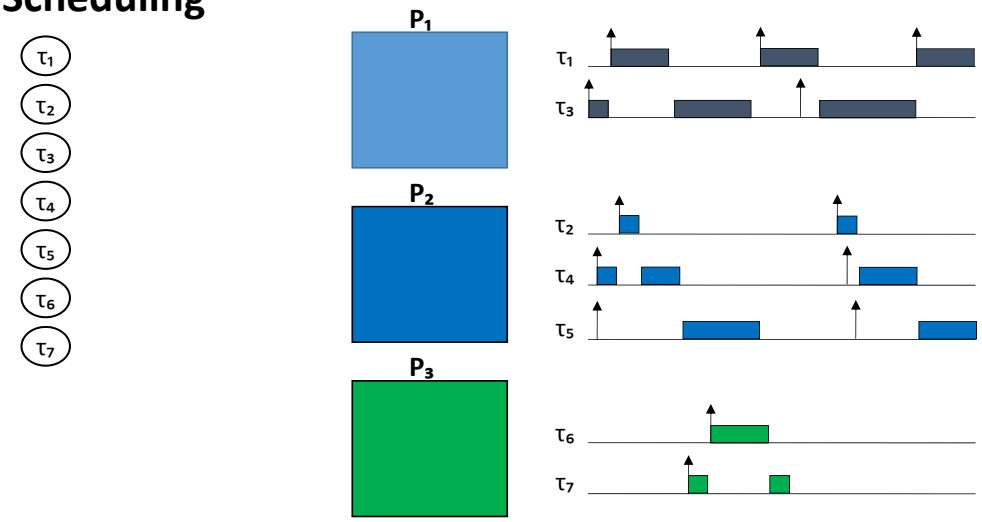
- **Multi-core Architectures**
  - Two or more independent processors (cores) on one chip (Single-chip multiprocessor).
  - Processors may have independent (L1) and/or on-chip shared (L2) cache.
  - Actual parallelism.
  - Very fast inter-core communication.
  - Shared Memory and BUS.
  - Cores may have identical or different performance.
  - Overcome thermal and power consumption problems.

Introduction



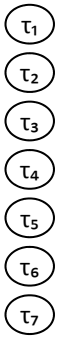
Scheduling on Multiprocessors

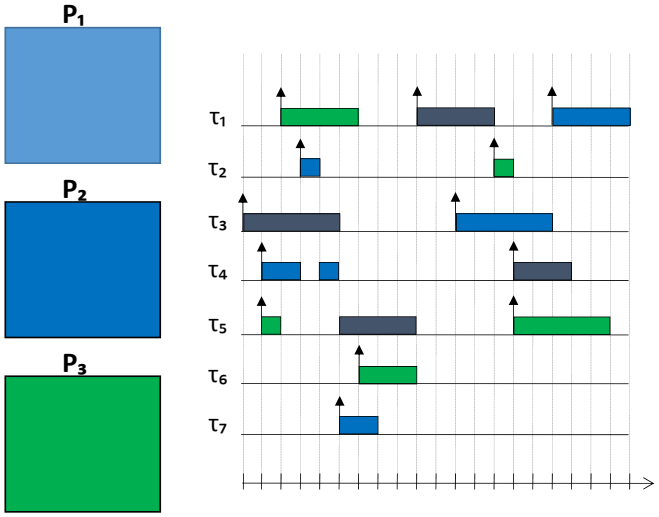
- **Partitioned Scheduling**



Scheduling on Multiprocessors

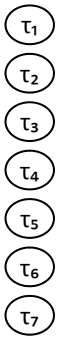
- **Global Scheduling**

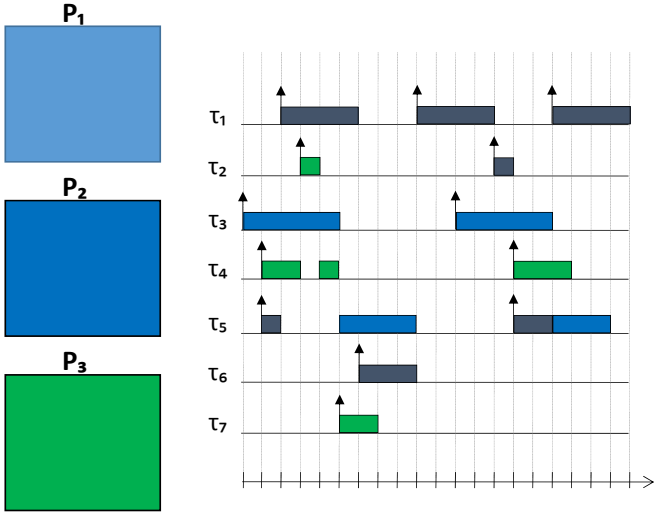




Scheduling on Multiprocessors

- **Hybrid Scheduling**





Assign Tasks to Cores (Linux)

```
#include <pthread.h>
int pthread_setaffinity_np(pthread_t thread, size_t cpusetsize, const cpu_set_t *cpuset);
int pthread_getaffinity_np(pthread_t thread, size_t cpusetsize,  cpu_set_t *cpuset);
```

Some macros for working with **cpu_set_t**:

```
void CPU_ZERO(cpu_set_t *set);
void CPU_SET(int cpu, cpu_set_t *set);
void CPU_CLR(int cpu, cpu_set_t *set);
int  CPU_ISSET(int cpu, cpu_set_t *set);
```

Assign Tasks to Cores (Linux)

Set the attribute object of thread to assign the thread to specific processors at the time the thread created creating:

```
int pthread_attr_setaffinity_np(pthread_attr_t *attr, size_t cpusetsize, const cpu_set_t *cpuset);
```

Assign Tasks to Cores; Example

- See the attached code (setAffinityExample.c)