# Section 10: Multiprocessor Scheduling

## Question 1

1. Describe the difference between Static Assignment vs. Dynamic Assignment of threads to Processors.
2. As described in the slides, how are both strategies implemented?

### Answer

1. With static assignment, newly created threads are attached to, and execute on, the same processor throughout is lifetime (unless moved by a load balancer).   
   With dynamic assignment, ready threads are assigned by the dispatcher to any available processor i.e. threads migrate between processors.
2. Static assignment is implemented with per-processor ready queues.   
   Dynamic assignment is implemented with a single, global ready queue.

## Question 2

What are the three disadvantages of dispatching processes to multiple processors from a single, shared ready queue (as opposed to having a ready queue per processor)?

### Answer

1. The shared global queue must be thread safe so mutually exclusive access to the shared queue (i.e. locking, etc.) may become a bottleneck.
2. Threads are unlikely to be re-scheduled on the same processor which makes each processor’s local memory cache less effective when threads move from processor to processor. (important)
3. If threads from all processes are dispatched for a single queue, it is unlikely all of the threads belonging to a single process will scheduled at the same time with the other threads belonging to the same process. (important)

## Question 3

In terms of processor cache utilization, what is the advantage of repeatedly dispatching threads to the same processor?

### Answer

As a thread executes, its current locality causes certain text and data pages to be loaded into its processor’s cache (L1 & L2). If the thread is re-scheduled to the same processor, the processor’s cache remains ‘hot’ i.e. all of the thread’s pages will remain in cache. Contrast with the situation where a thread executes on a different processor; each thread starts its execution with a cold cache that must be reloaded with the thread’s text and data pages.

## Question 4

In terms of throughput, what is the advantage of Gang Scheduling?

### Answer

Gang scheduling attempts to keep all of a process’s threads executing simultaneously. This is important when a process’s threads are tightly coupled, as they often are. Tightly coupled threads will block when synchronizing with a non-running peer thread. The blocking results in a type of ‘processor thrashing’ where a thread executes for a short period and blocks waiting to synchronize with a peer. When all of the threads are running simultaneously, there is no synchronization blocking, the process completes sooner.

## Question 5

1. Describe the meaning of Load Balancing in the context of Processor Assignment.
2. Is a Load Balancer needed with dynamic or static processor assignment?

### Answer

1. Load balancing is the reassignment of threads from assigned processors to other less heavily loaded processors. The ‘Load Balancer’ is a task that periodically runs and examines the load each processor is under. A processor’s load may be measured in terms of the number of threads assigned to the processor’s queue (static assignment). The goal is to spread the system’s processing load evenly across all of the system’s processors.
2. Load balancing is only needed with static assignment. Dynamic assignment naturally spreads the load as threads are assigned from a single global queue to the next available (idle) processor.

## Question 6

1. Describe the meaning of ‘Resource Aware Thread Placement’.
2. Describe the example discussed in class.

### Answer

1. Resource Aware Thread Placement has the scheduler assigning threads to processes which optimize system performance because of resources available to specific processors.
2. The example given in class was the placement of threads from the same process on processors that share L1 or L2 cache. The reason is that because threads from the same process share the same text and data memory blocks, their placement on adjacent processors will make the best use of the cache memory shared between those processors.

## Question 7

Describe the meaning of, and relationships between, events, tasks, and deadlines in the context of real-time operating systems.

### Answer

An Event is a signal (usually an interrupt) that triggers the execution of a task.

A Task is code that is executed (triggered) in response to the event. A task can also be regularly scheduled for execution by the RT OS.

A Deadline is a measurement of time between when an event occurs and the completion of the execution of a task that responds to the event and generates an output. The system must respond to, and produce the output within a specific amount of time (its deadline) or suffer some type of failure.

From the example of automotive airbag controllers given in class, a collision (**event**) is detected by an accelerometer and signals the automobile’s on-board computer. The system executes a specific **task** that examines the input and decides whether the air bags should be deployed. The system must respond to the ‘crash’ event and cause the airbags to deploy within a certain amount of time (**deadline**) otherwise the airbag effectiveness will be reduced or even become a hazard to the passengers.

## Question 8

What are the special characteristics of real-time tasks executed in response to incoming events?

### Answer

1. Execution of the task is triggered by an interrupt, not the OS scheduler.
2. Memory hosting the task’s code and data segments cannot be swapped out to disk i.e. no virtual memory.
3. Tasks cannot make any blocking system calls i.e. calls to IO or network communication.
4. The task must be designed and verified to execute well within the deadline imposed by the application and the processing capacity of the processor. Worse-case path analysis must be employed to ensure that no possible combination of state and inputs can cause the task complete after its deadline.

## Question 9

What are two types of tasks dispatched by real-time operating systems?

### Answer

1. Tasks that are executed in response to events from external sources.
2. Tasks that are periodically executed (scheduled) by the operating system.