Assignment 1: Scheduling

Author: Sebastian Daag
Email: sdg20001@student.mdu.se

Question 1 (Related to the Multi-core lecture)

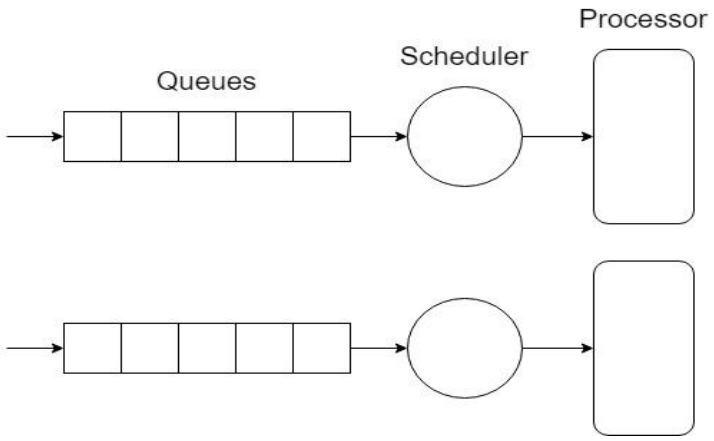
a) What does cache coherency refer to?
In a multi-core system, cache coherency refers to the consistency of data stored in different cashes. In other words ensuring that within the system, all processors have the same and a consistent view of each shared memory location at any point in time. So when one core modifies the data, those changes must be reflected in the caches of other cores to avoid inconsistencies.

b) Is shared memory always the best method for communicating among cores? When does it become problematic?
Shared memory is a common method for communication due to its efficacy and simplicity, but it may not always be the best, especially when the number of cores start to increase. This is because shared memory has limited scalability and a set performance bottleneck. So as the number of cores increases, the amount of contention for access to the shared memory increases and managing shared memory efficiently can become challenging. All leading to a slower performance and the communication overhead starts outweighing the benefits.

https://www.quora.com/What-are-the-advantages-and-disadvantages-of-a-shared-memory-architecture-compared-to-a-distributed-memory-architecture-for-parallel-computing-and-in-what-types-of-applications-would-each-architecture-be-most

c) Describe two main scheduling approaches in multiprocessor systems, their strengths and weak points?

Partitioned scheduling:
In a partitioned scheduling each core is assigned its own scheduler and ready queue, which means each partitions operate independently of each other. The tasks are pre-assigned to cores during design time and only execute within their pre-assigned cores.

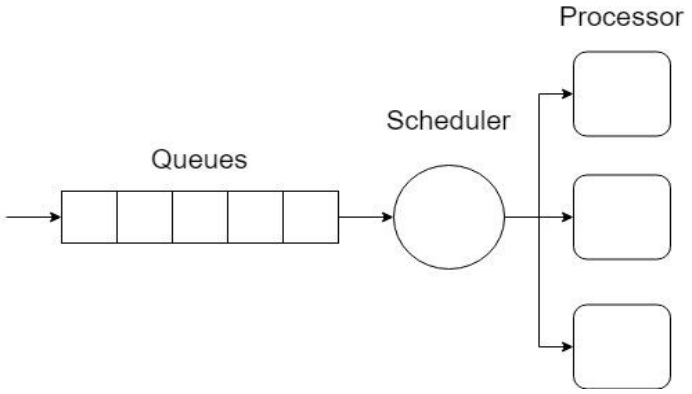


- Advantages:
    - As each partition can operate independently, it is easy to increase the scale of operations, reducing the need for global coordination.
    - Less complex than global scheduling.
    - Improved predictability and reduced interference between tasks due to the isolated partitions

- Disadvantages:
    - If communication between partitions is required for coordination between certain tasks, introducing additional overhead becomes required.
    - Each partition has a limited view of the system, which may lead to suboptimal decisions for resource utilization and task allocation.

Global scheduling:

In global scheduling multiple cores are assigned to one global schedule and unique queue. Which means the scheduler has a global view of the system, allowing it to make decisions based on the overall state of the system. Allowing tasks to be dynamically assigned to available cores during run-time based on system load.



- Advantages:
    - Can dynamically adapt to changes in task requirements during run-time.

    - The scheduler can efficiently coordinate tasks, avoiding conflicts and ensuring efficient use of resources.
    - The scheduler can make intelligent decisions about task placement, allowing tasks to be added or removed from the system dynamically.
- Disadvantages:
    - Some algorithms become obsolete, as RM and EDF are not optimal anymore for multiprocessors.
    - Scalability can become more challenging as the system grows because scheduling decisions for a large number of processors will eventually reach a performance bottleneck.

Question 2: (Related to the System-level Design lecture)

Describe various aspects and constraints that have a large impact on the result of an embedded system design.
s

Question 3: (Related to the Cloud, IoT, Resource Virtualization in ES lecture)

Consider a FOG node that has a number of virtual machines allocated to multicore. Core 1 has three different virtual machines VM1, VM2 and VM3. (Q1=2, P1=4), (Q2=3, P2=8) and (Q3=2, P3=16) where Q1, Q2, Q3 are budgets for VM1, VM2, VM3 respectively and P1, P2, P3 are periods of VM1, VM2, VM3.

VM1 has three periodic tasks τ1, τ2, τ3 and each task is characterized by C=execution time, T=period and D=deadline. C1=2, T1=D1=8, C2=3, D2=T2=16, C3=2, D3=T3=32

VM2 has two periodic tasks τ4, τ5 with parameters C4=3, T4=16, D4=10, C5=3, T5=16 and D5=15. VM3 has one periodic task τ6 with parameters C6=2, T6=16, D6=14

Assume that the fixed priority-scheduling algorithm is used for all schedulers (global and local) and priorities are selected based on the deadline monotonic approach.

Set up
VM1: Budget Q1 = 2, Period P1= 4, Priority = 1 (Highest)

Tasks	Period (T)	Deadline (D)	Exec. Time (C)
Task τ1	8	8	2
Task τ2	16	16	3
Task τ3	32	32	2

VM2: Budget Q2 = 3, Period P2= 8, Priority = 2

Tasks	Period (T)	Deadline (D)	Exec. Time (C)
Task τ4	16	10	3
Task τ5	16	15	3

VM3: Budget Q3 = 2, Period P3= 16, Priority = 3 (Lowest)

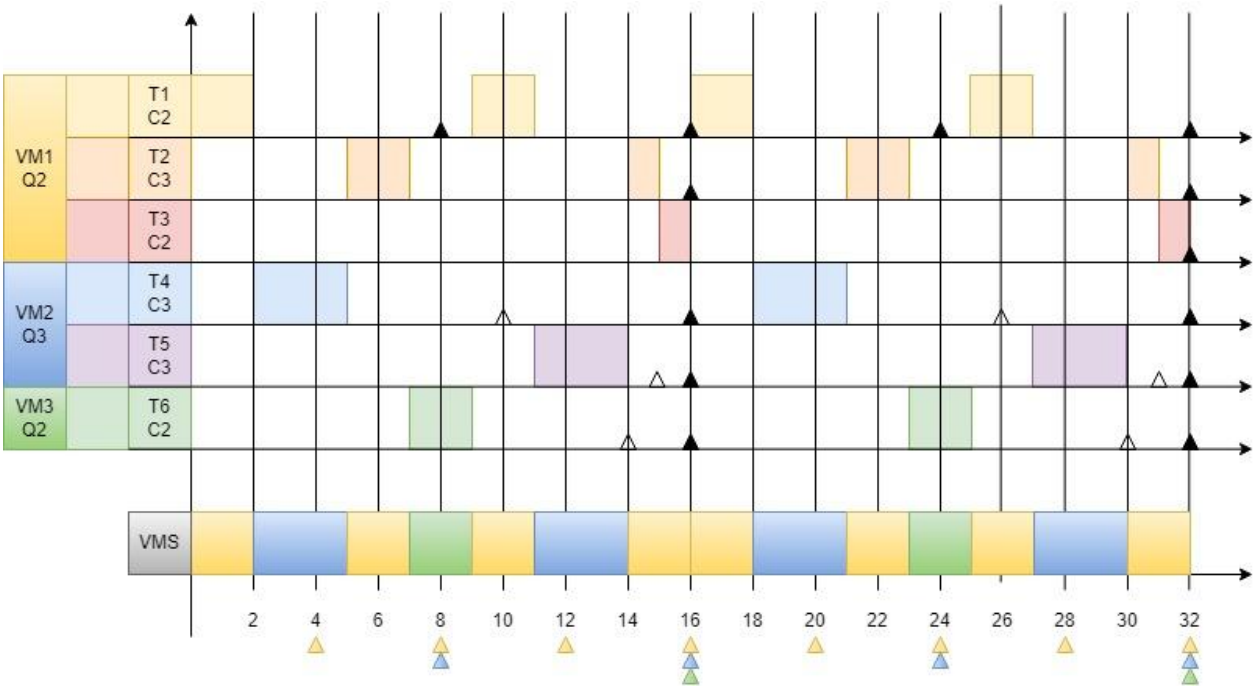
Tasks	Period (T)	Deadline (D)	Exec. Time (C)
Task τ6	16	14	2

a) Will all tasks meet their deadlines? Motivate your answer by drawing the execution trace of VMs and tasks.
Assume that the fixed priority-scheduling algorithm is used for all schedulers (global and local) and priorities are selected based on the deadline monotonic approach.

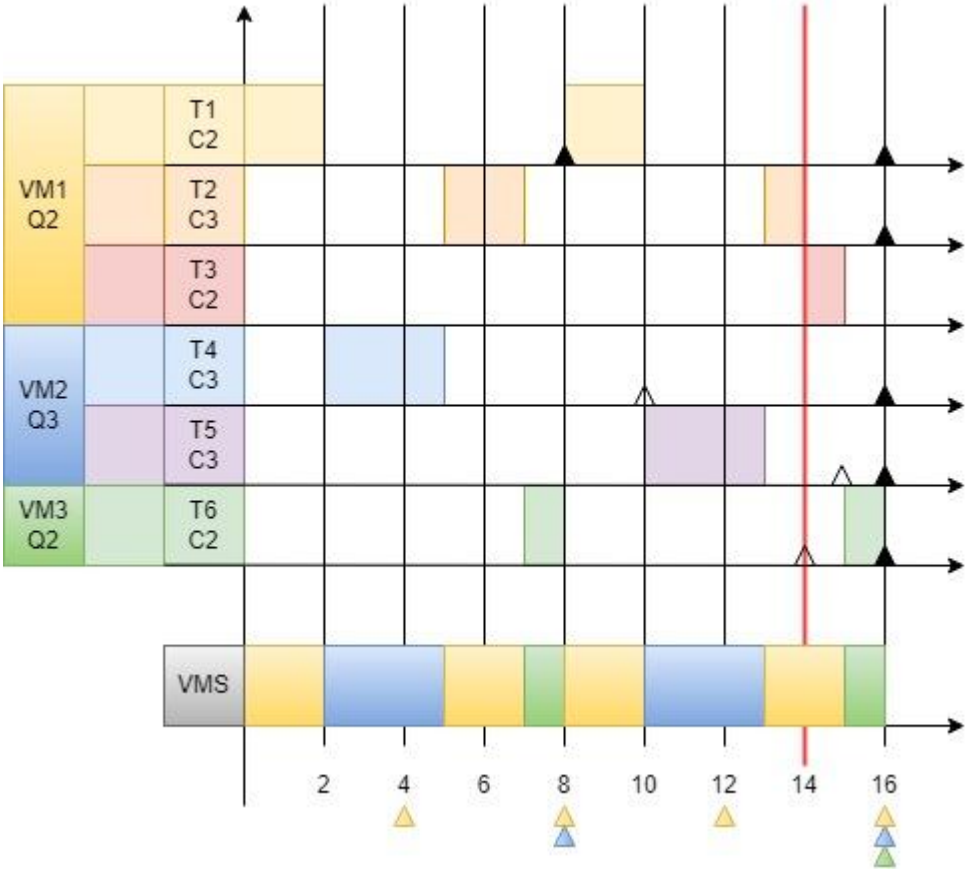
Starting at LCM(8,16,32) = 32

VM 1	VM2	VM3
Task 1: prio 1 (highest)	Task 4: prio 1 (highest)	Task 6: prio 1 (highest)
Task 1: prio 2	Task 5: prio 2	
Task 1: prio 3		

Version 1, no interrupts of tasks allowed, most likely wrong

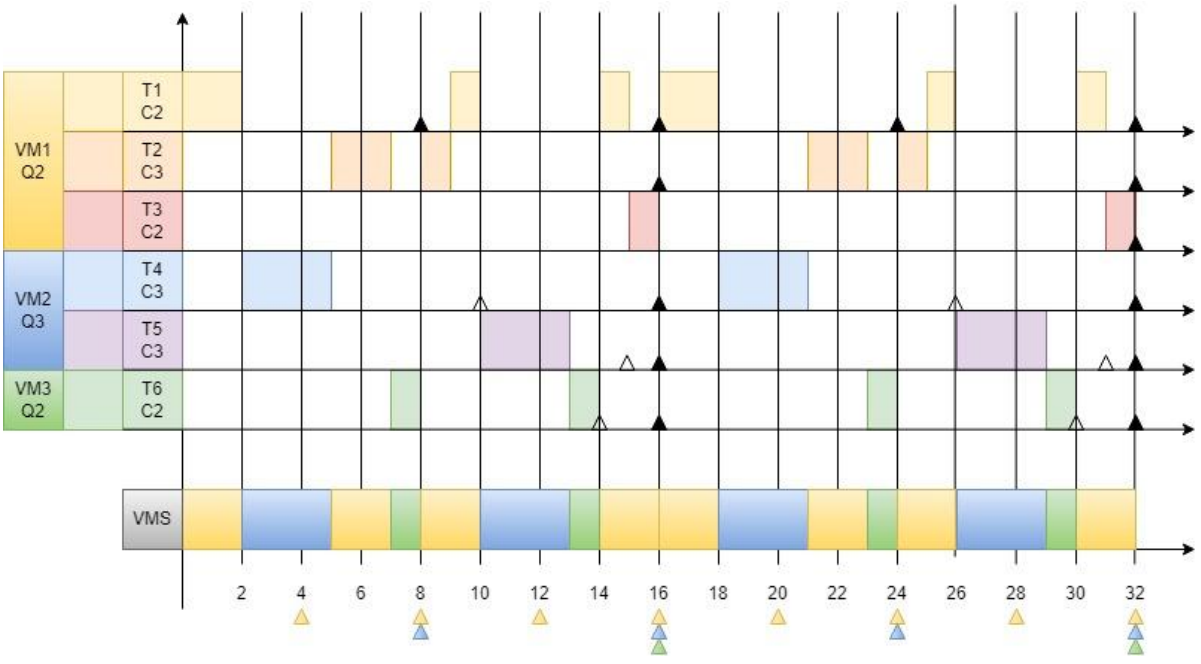


Version 2, interrupts of tasks allowed, most likely correct

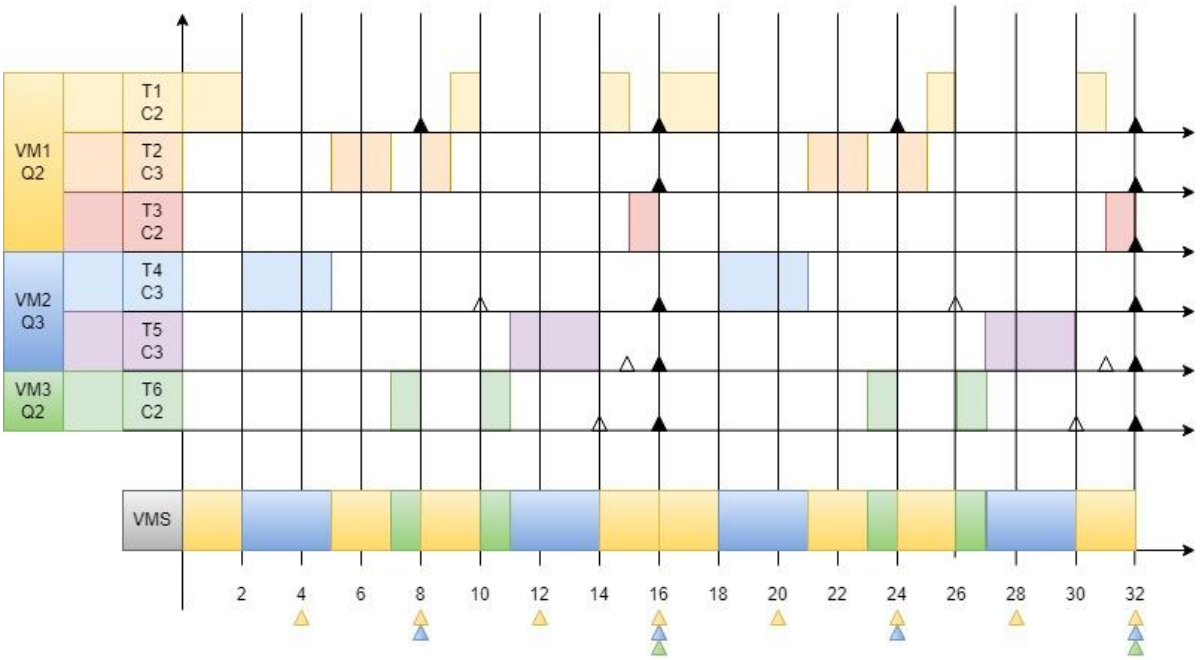


b) If we change the global level scheduler to EDF what will happen?
With EDF

Version 1



Version 2



c) If there are some tasks that miss their deadlines, what can be done to make them meet their deadlines without changing the given parameters of tasks and VMs? (Hint: see Lecture 4: Schedulability Analysis)

Using offsets:

- What if tasks cannot be released simultaneously?
    - Everything in the system may not happen at once.
- Response times of tasks (especially with lower priority) increase with the increase in the system load.
    - Response times of lower priority tasks can be reduced if the tasks are scheduled with offsets.
- In order to avoid deadlines violations due to high transient loads, embedded systems are often scheduled with offsets.

d) (Optional) Suppose that there is one more copy of VM2 and another one of VM3 and there are two cores available to execute the 5 VMs. How will you distribute the VMs so that all tasks meet their deadlines? Please motivate your design.