MIRROR: Symmetric Timing Analysis for Real-Time Tasks on Multicore Platforms with Shared Resources

Wen-Hung Kevin Huang, Jian-Jia Chen, and Jan Reineke

TU Dortmund and University of Saarlands

09,06,2016 at DAC, Austin, USA

technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

Periodic Control System (Liu and Layland 1973)

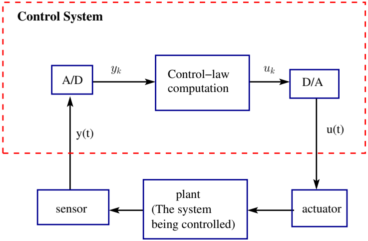
Pseudo-code for this system

set timer to interrupt periodically with period T;

at each timer interrupt
do

- perform analog-to-digital conversion to get y;
- compute control output u;
- output u and do digital-to-analog conversion;

od



Periodic Control System (Liu and Layland 1973)

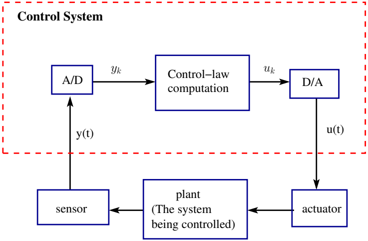
Pseudo-code for this system

set timer to interrupt periodically with period T;

at each timer interrupt
do

- perform analog-to-digital conversion to get y;
- compute control output u;
- output u and do digital-to-analog conversion;

od

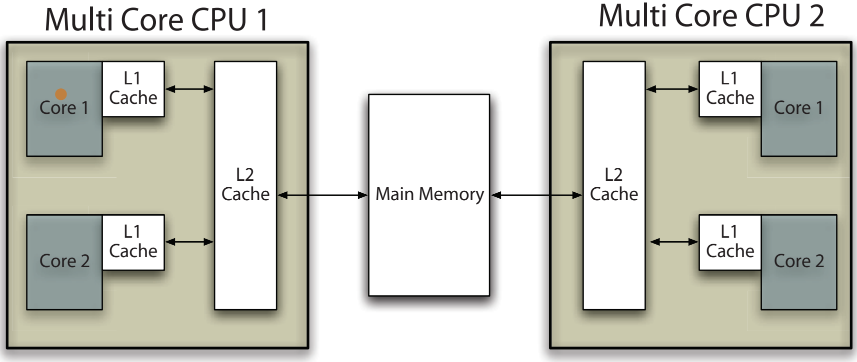


Liu and Layland Model:

- T_i: period of task τ_i
- D_i: relative deadline of task τ_i
- C_i: worst-case execution time of task τ_i
- U_i: utilization C_i/T_i

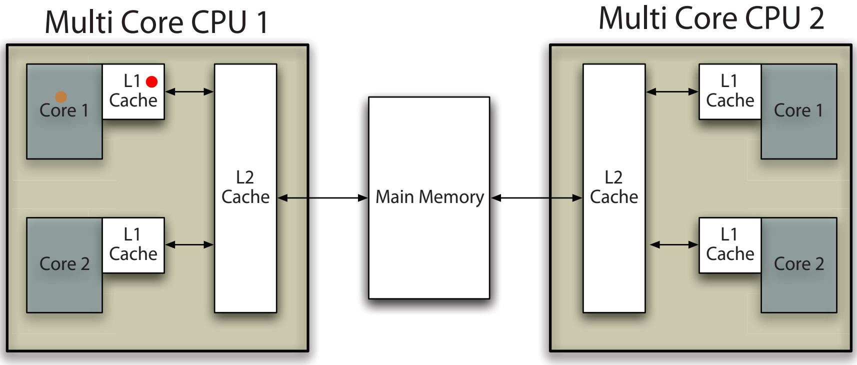
Predictability Due to Resource Sharing

Task is executed on Core 1



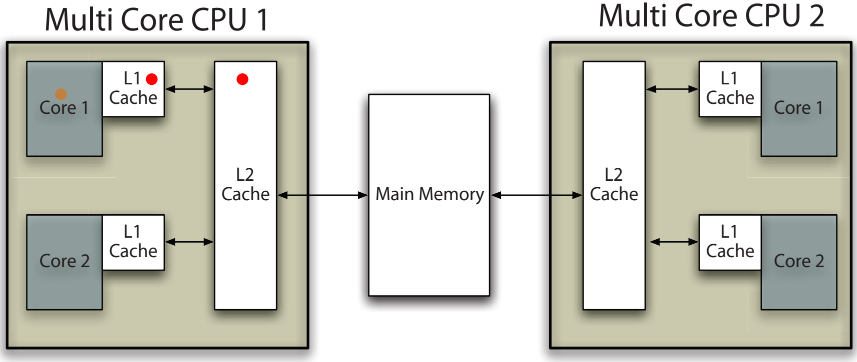
Predictability Due to Resource Sharing

L1 cache misses



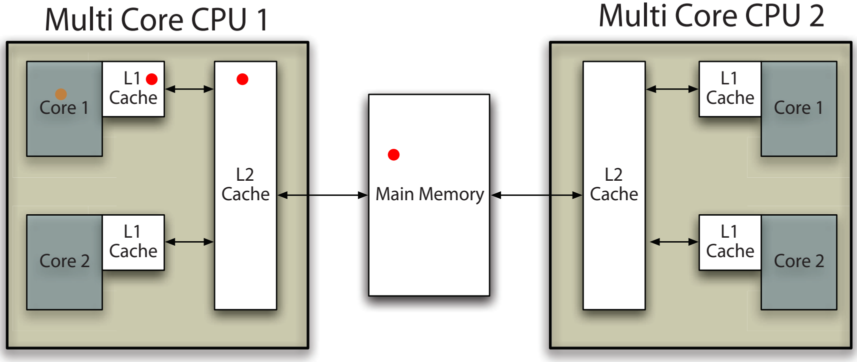
Predictability Due to Resource Sharing

L2 cache misses



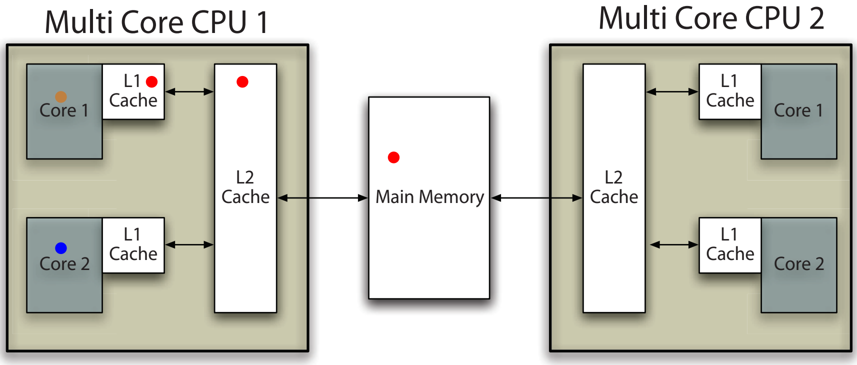
Predictability Due to Resource Sharing

Access memory



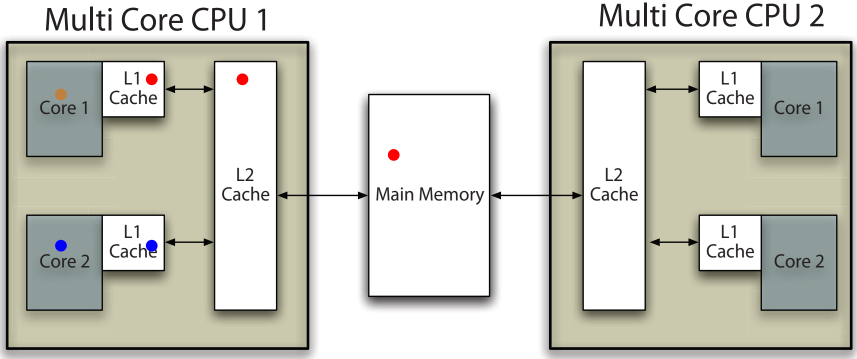
Predictability Due to Resource Sharing

Task is executed on Core 2

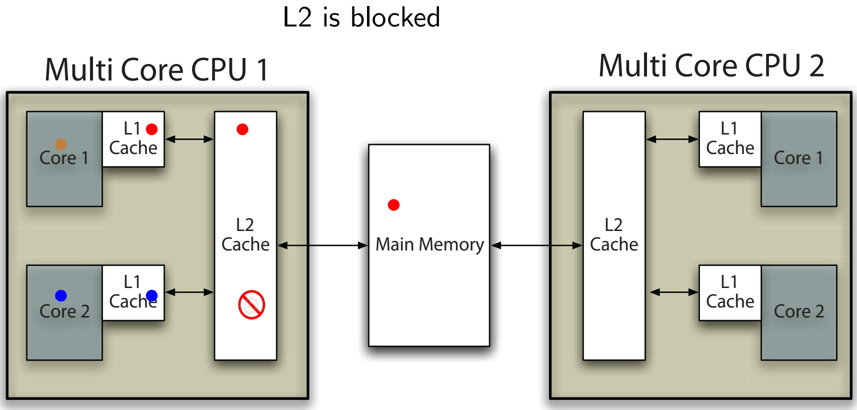


Predictability Due to Resource Sharing

L1 cache misses

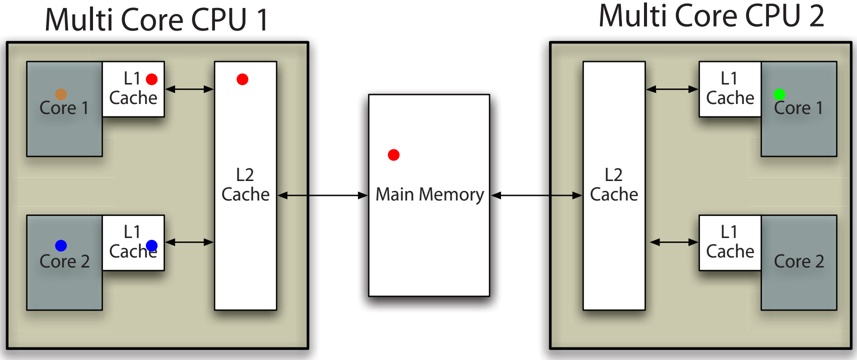


Predictability Due to Resource Sharing



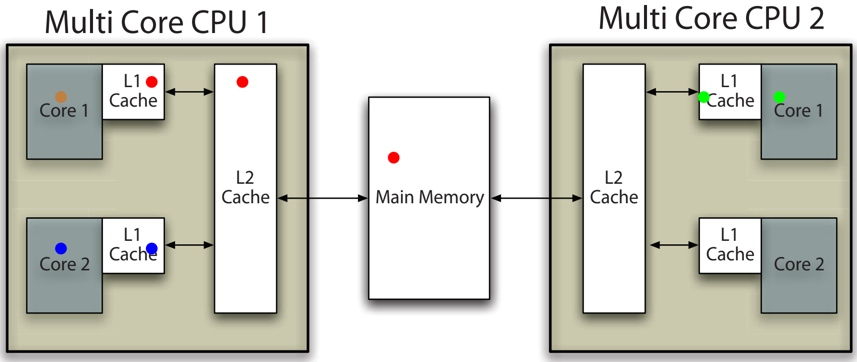
Predictability Due to Resource Sharing

Task is executed on Core 3



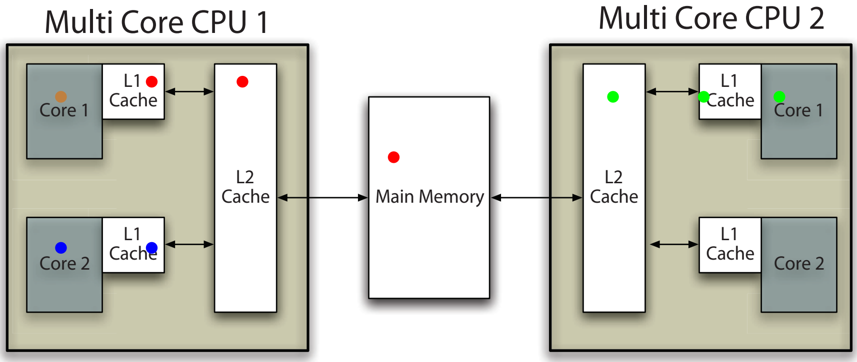
Predictability Due to Resource Sharing

L1 cache misses



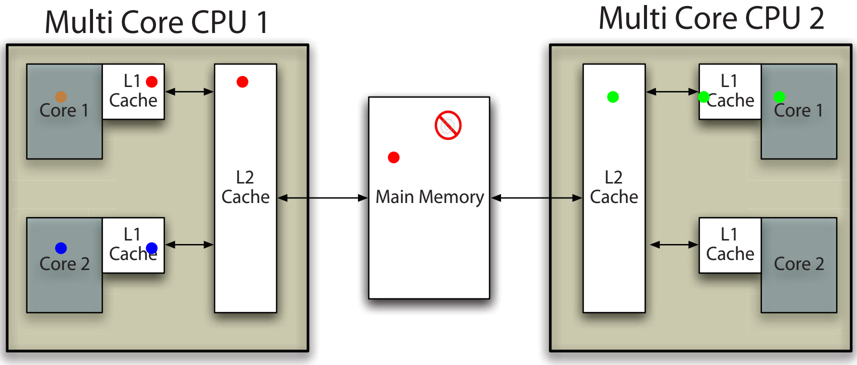
Predictability Due to Resource Sharing

L2 cache misses



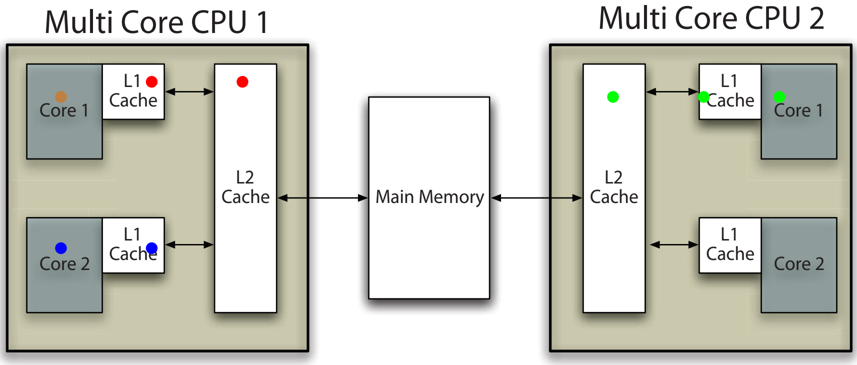
Predictability Due to Resource Sharing

Memory access is blocked



Predictability Due to Resource Sharing

Memory access for core 1 finishes



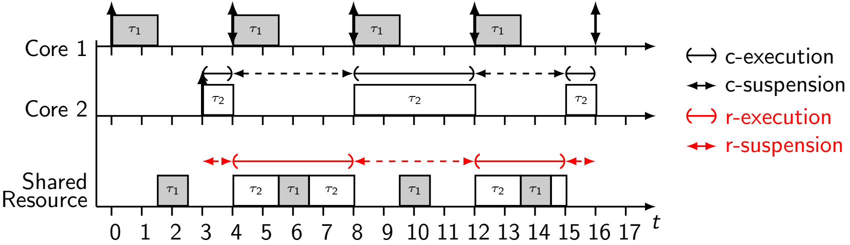
Typical Two-Phase Analysis Approaches

- Phase 1: Worst-case execution time (WCET) of a stand-alone program
  - WCET analyzers such as aiT or Chronous.

Typical Two-Phase Analysis Approaches

- Phase 1: Worst-case execution time (WCET) of a stand-alone program
  - WCET analyzers such as aiT or Chronous.
- Phase 2: Worst-case response time (WCRT) of a periodic/sporadic task by considering the competition with the other tasks
  - worst-case interference from the other tasks
  - utilization-based tests, response time analysis, busy-interval techniques, real-time calculus, max-plus algebra, etc.
- The notion of WCET is destroyed in multicore systems due to shared resources.
  - WCET depends on how the tasks on the other cores are co-executed
  - Assume the worst-case interference is too pessimistic

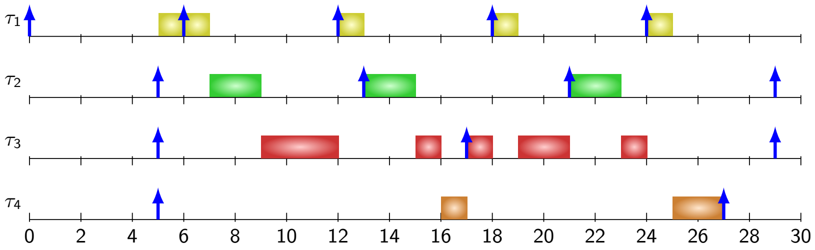
Self-Suspending Behavior



- Multiple cores may share a bus
- The contention on the bus can be considered as a suspension problem (with respect to the bus access)

Self-Suspension Tasks in Real-Time Systems

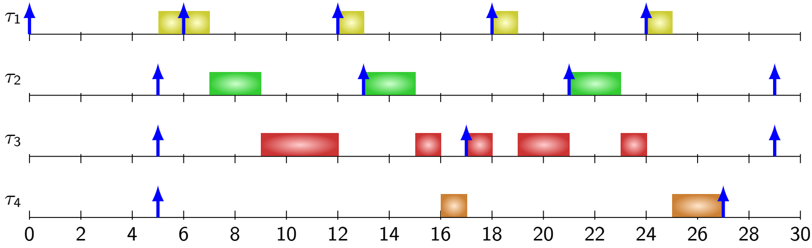
Suppose that we know the suspension time of each τ_i and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)



- A self-suspending task τ_i is a periodic task with jitter (PJD task)
  - Period: T_i
  - Self-suspension-time: S_i

Self-Suspension Tasks in Real-Time Systems

Suppose that we know the suspension time of each τ_i and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

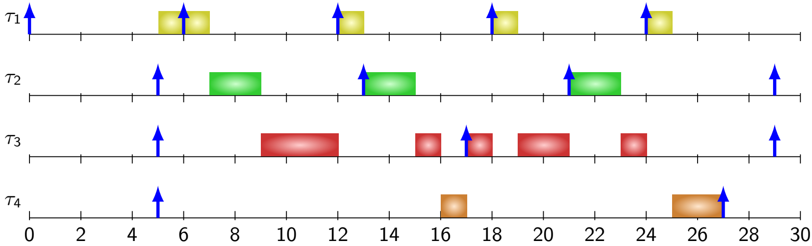


- A self-suspending task τ_i is a periodic task with jitter (PJD task)
  - Period: T_i
  - Self-suspension-time: S_i
- Schedulability test of task τ_k:

$$\exists t \text{ with } 0 < t \leq T_k \;\; and \;\; C_k + S_k + \sum_{j=1}^{k-1} \left\lceil \frac{t + S_j}{T_j} \right\rceil C_j \leq t.$$

Self-Suspension Tasks in Real-Time Systems

Suppose that we know the suspension time of each τ_i and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

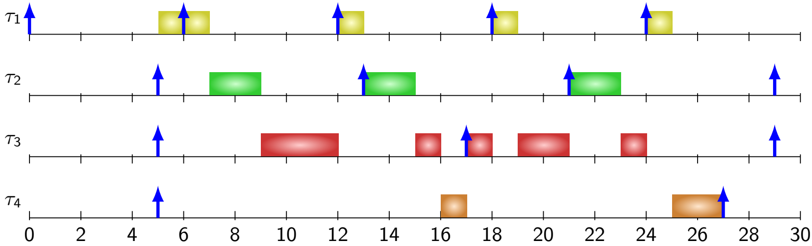


- A self-suspending task τ_i is a periodic task with jitter (PJD task)
  - Period: T_i
  - Self-suspension-time: S_i
- Schedulability test of task τ_k:

$$\exists t \text{ with } 0 < t \leq T_k \;\; and \;\; C_k + S_k + \sum_{j=1}^{k-1} \left\lceil \frac{t + D_j}{T_j} \right\rceil C_j \leq t.$$

Self-Suspension Tasks in Real-Time Systems

Suppose that we know the suspension time of each τ_i and would like to analyze the schedulability of the tasks on a core. (Constrained-deadline $D_i \leq T_i$)

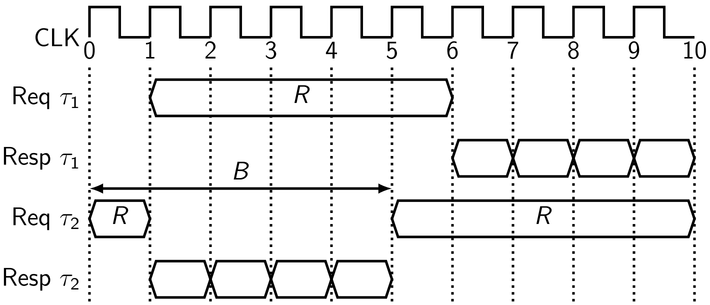


- A self-suspending task τ_i is a periodic task with jitter (PJD task)
  - Period: T_i
  - Self-suspension-time: S_i
- Schedulability test of task τ_k:

$$\exists t \text{ with } 0 < t \leq T_k \;\; and \;\; C_k + S_k + \sum_{j=1}^{k-1} \left\lceil \frac{t + T_j}{T_j} \right\rceil C_j \leq t.$$

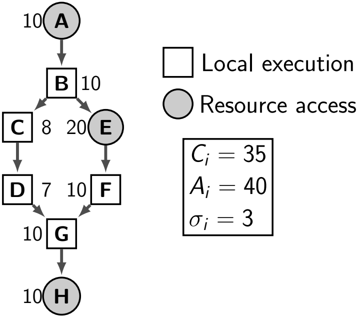
Platform Model

- Multicore with a share resource
- For example, atomic (non-split-transaction) bus
  - Bus sits idle while memory processes the request and sends the response
- Fixed-priority arbitration



Task and Scheduling Model

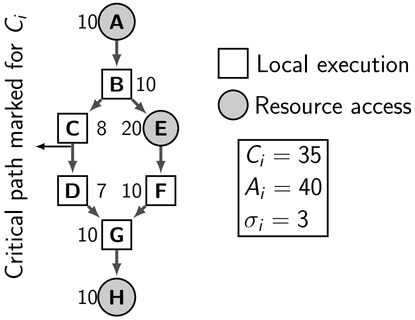
- Resource access task τ_i $(C_i, A_i, T_i, D_i, \sigma_i)$
  - C_i: upper bound on local computation
  - A_i: upper bound on resource accesses
  - T_i: period
  - D_i: relative deadline $(D_i \leq T_i)$
  - σ_i: the maximum number segments of consecutive resource accesses
- Path analysis
- Fixed-priority scheduling (we use deadline-monotinic scheduling)



$C_i = 35$
$A_i = 40$
$\sigma_i = 3$

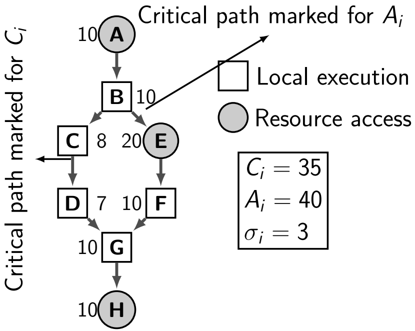
Task and Scheduling Model

- Resource access task τ_i $(C_i, A_i, T_i, D_i, \sigma_i)$
  - C_i: upper bound on local computation
  - A_i: upper bound on resource accesses
  - T_i: period
  - D_i: relative deadline $(D_i \leq T_i)$
  - σ_i: the maximum number segments of consecutive resource accesses
- Path analysis
- Fixed-priority scheduling (we use deadline-monotinic scheduling)



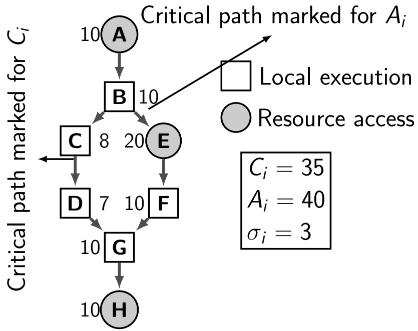
Task and Scheduling Model

- Resource access task τ_i $(C_i, A_i, T_i, D_i, \sigma_i)$
  - C_i: upper bound on local computation
  - A_i: upper bound on resource accesses
  - T_i: period
  - D_i: relative deadline $(D_i \leq T_i)$
  - σ_i: the maximum number segments of consecutive resource accesses
- Path analysis
- Fixed-priority scheduling (we use deadline-monotinic scheduling)



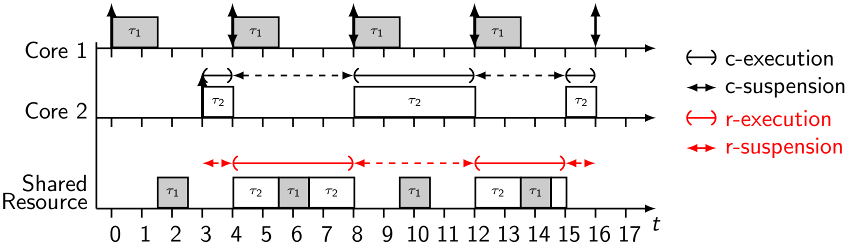
Task and Scheduling Model

- Resource access task τ_i $(C_i, A_i, T_i, D_i, \sigma_i)$
  - C_i: upper bound on local computation
  - A_i: upper bound on resource accesses
  - T_i: period
  - D_i: relative deadline $(D_i \leq T_i)$
  - σ_i: the maximum number segments of consecutive resource accesses
- Path analysis
- Fixed-priority scheduling (we use deadline-monotinic scheduling)



Assume compositional properties: 75 is a safe upper bound.

Key Observations: Symmetric Property



- From the core perspectives for τ_2
  - accessing or waiting: [3,4), [8,12), [15, 16)
  - suspension: [4,8), [12, 15)
- From the shared resource perspectives for τ_2
  - executing or waiting: [4,8), [12, 15)
  - suspension:[3,4), [8,12), [15, 16)

Schedulability Test for Task τ_k

- WCRT is upper bounded by the minimum $t | 0 < t \leq D_k$

$$(C_k + exec\_core(t)) + (A_k + exec\_resource(t)) \leq t$$

Schedulability Test for Task τ_k

- WCRT is upper bounded by the minimum $t | 0 < t \leq D_k$

$$\left(C_k + \sum_{\tau_i \in hp(\tau_k, c)} \left\lceil \frac{t + T_i}{T_i} \right\rceil C_i \right) + \sigma_k B + \left(A_k + \sum_{\tau_i \in hp(\tau_k, r)} \left\lceil \frac{t + T_i}{T_i} \right\rceil A_i \right) \leq t$$

  - σ_k B: the maximum blocking time by the lower priority tasks on the shared resource
  - hp(τ_k, c): higher-priority tasks thanτ_k on the same core
  - hp(τ_k, r): higher-priority tasks than τ_k on shared resource

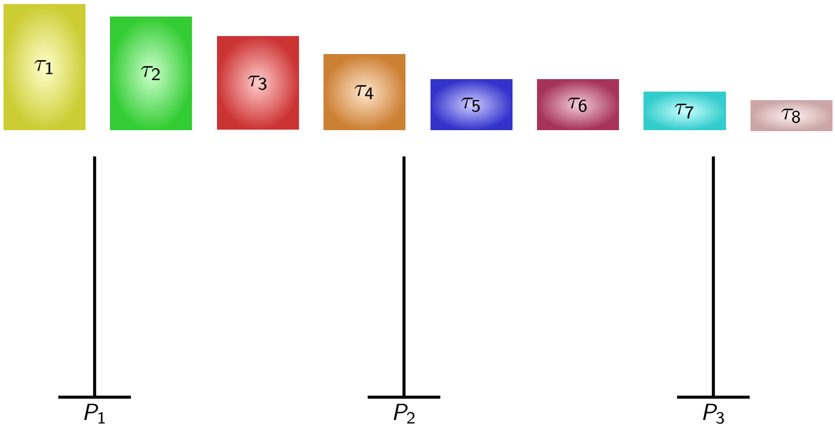
Schedulability Test for Task τ_k

- WCRT is upper bounded by the minimum $t | 0 < t \leq D_k$

$$\left(C_k + \sum_{\tau_i \in hp(\tau_k, c)} \left\lceil \frac{t + T_i}{T_i} \right\rceil C_i \right) + \sigma_k B + \left(A_k + \sum_{\tau_i \in hp(\tau_k, r)} \left\lceil \frac{t + T_i}{T_i} \right\rceil A_i \right) \leq t$$

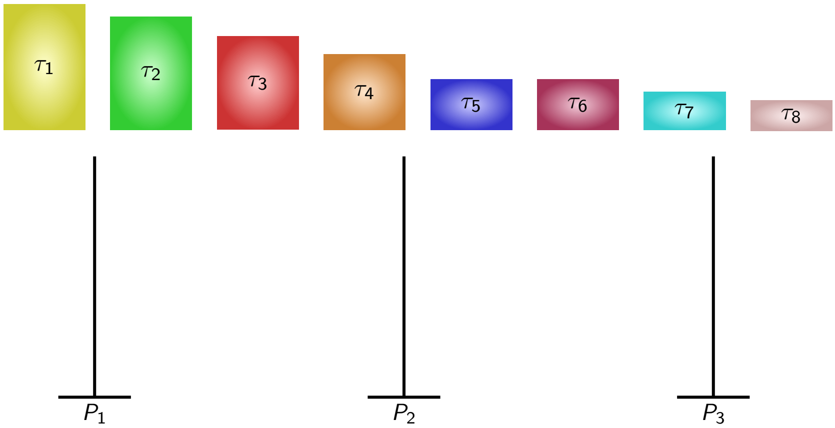
  - σ_k B: the maximum blocking time by the lower priority tasks on the shared resource
  - hp(τ_k, c): higher-priority tasks thanτ_k on the same core
  - hp(τ_k, r): higher-priority tasks than τ_k on shared resource
- Pessimism of the above response time analysis: number of resource access segments was not exploited
- In our paper, we explain how to calculate and utilize the information σ_k in a symmetric and more precise manner

Task Assignment (Partition)



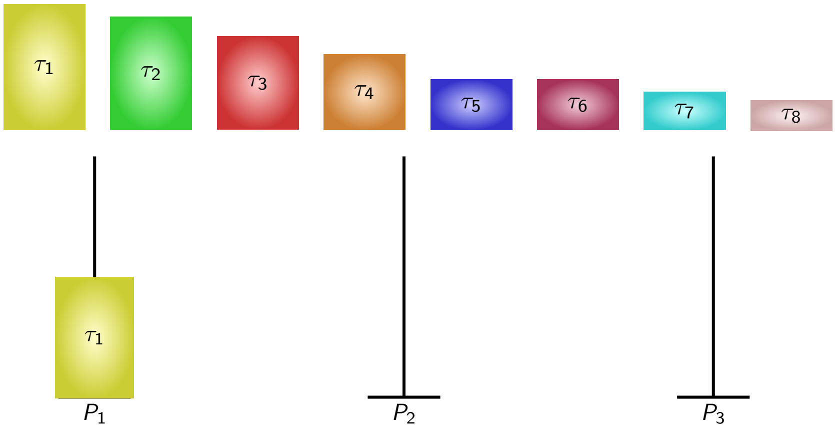
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



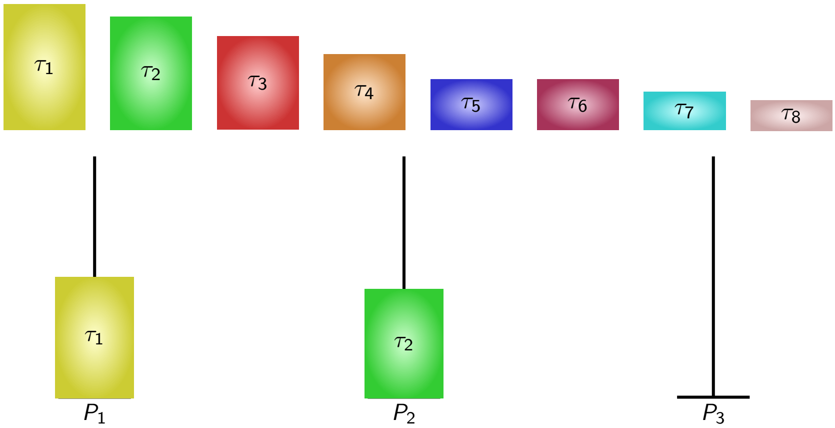
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



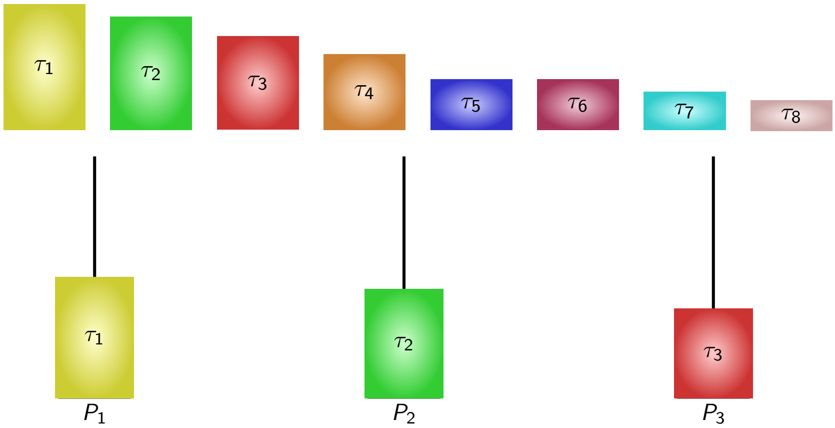
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



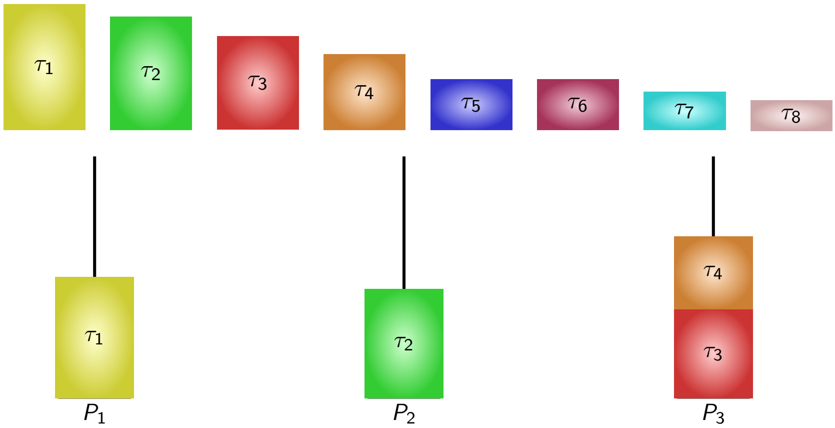
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



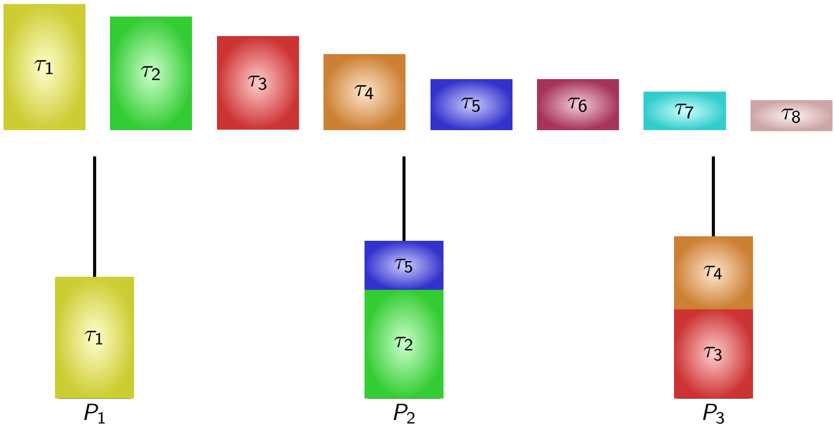
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



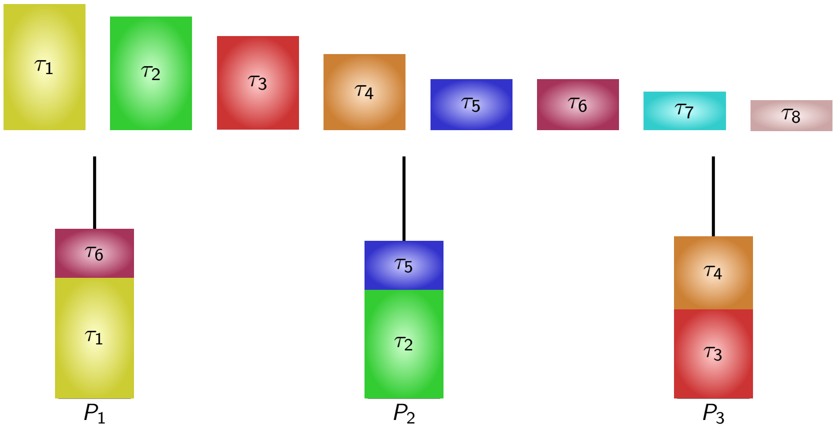
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



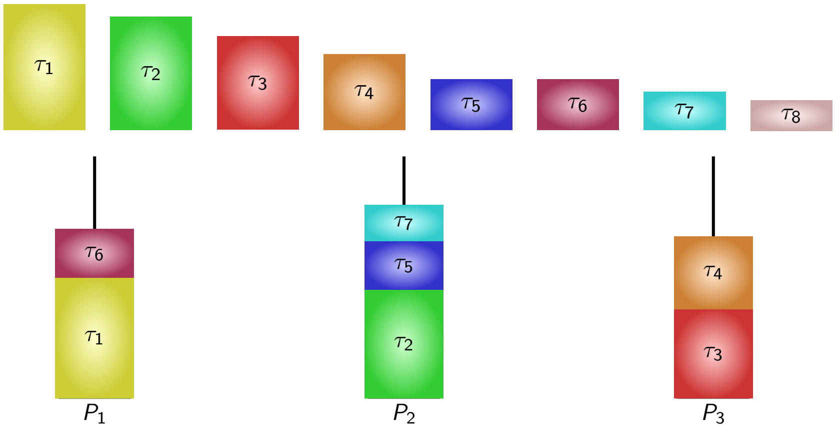
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



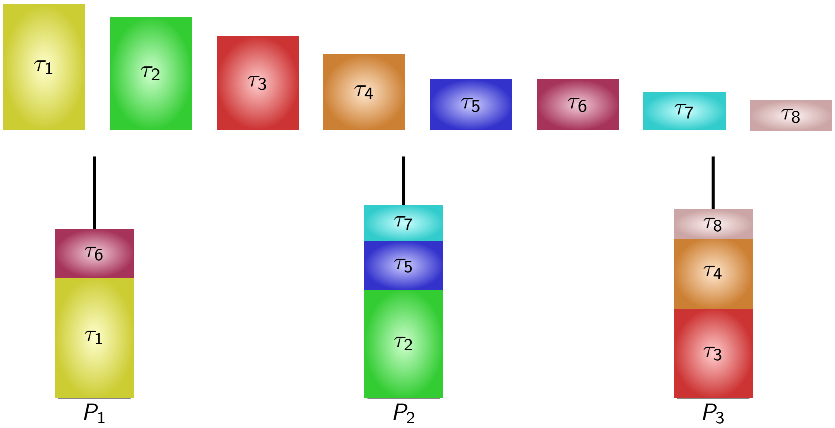
- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)



- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Task Assignment (Partition)

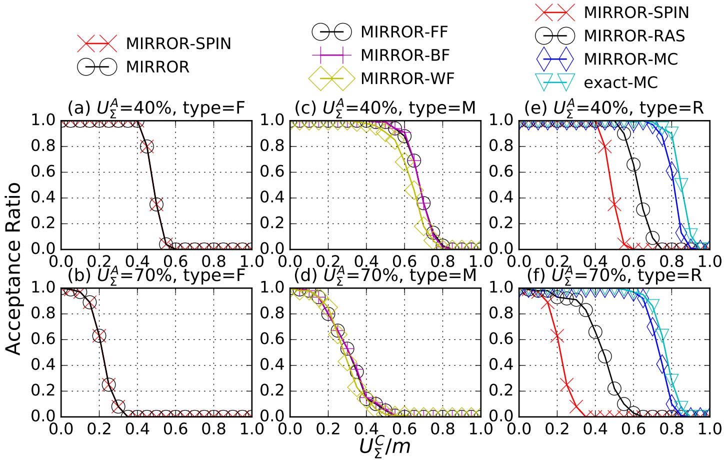


- Schedulability tests are based on the previous slide.
- Fitting can be First-Fit (FF), Worst-Fit (WF), Best-Fit (BF)

Experiments

- Configuration
  - 4-core platform (m=4)
  - 20 tasks
  - Periods [10-1000ms]
  - Each utilization level:100 task sets
- Existing results:
  - Exact-MC (Bonifaci et al. in RTNS 2015): do memory access and then do execution
  - MIRROR-SPIN (This resembles the test from Altmeyer et al. in RTNS 2015)
- Evaluation Metrics:
  - The acceptance ratio of a level: the number of task sets that are schedulable by the test divided by the number of task sets.

Experiments



The number of resource access segments σ_i: 1 (rare access, type=R), 2 (moderate access, type=M), and 10 (frequent access, type=F).

Conclusion and Extensions

- Fixed-priority, deadline-monotonic scheduling bus + bus-aware timing analysis + FFDM = high schedulability
  - A general treatment to handle multicore resource accesses
  - The treatment is compatible with existing task partitioning methods
  - The view points are symmetric
  - First result with worst-case resource augmentation guarantees (i.e., speedup factors) for this research line
- Extensions
  - Similar techniques can be applied for multiple shared resources
  - The pessimism can be further reduced by counting the interference more precisely