Real-Time Basics

I. Introduction

Real-Time Systems

correctness depends on temporal + functional aspects

must be predictable: worst case performance should be guaranteed

Real-Time Scheduling

assign computing resources to each tasks according to **scheduling policies**, guaranteeing that all constraints(deadlines) are met

Fundamental Problems of Real-Time Scheduling

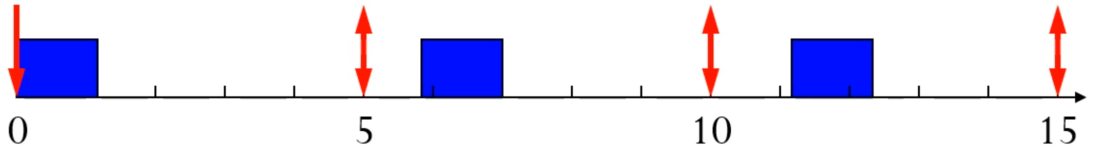
- Scheduling Algorithm Design: determine the order of execution
- Schedulability Analysis: validate that the algorithm meets every deadlines by picking worst case(even if it actually doesn't exists, pick conservatively) and conduct simulation for that case

Real-Time Task

Task = Series of Jobs

Real-Time Tasks can be modeled by below concept.

- Periodic task (p,e)
  - Its jobs repeat regularly
  - Period p = inter-release time ($0 < p$)
  - Execution time e = maximum execution time ($0 < e < p$)
  - Utilization U = e/p



II. Scheduling Algorithms in Uni-processor

Priority-based Real-Time Scheduling

- Task-level fixed-priority (TFP)

  RM(rate-monotonic): shorter period, higher priority → optimal fixed-priority

  DM(deadline-monotonic): shorter deadline, higher priority

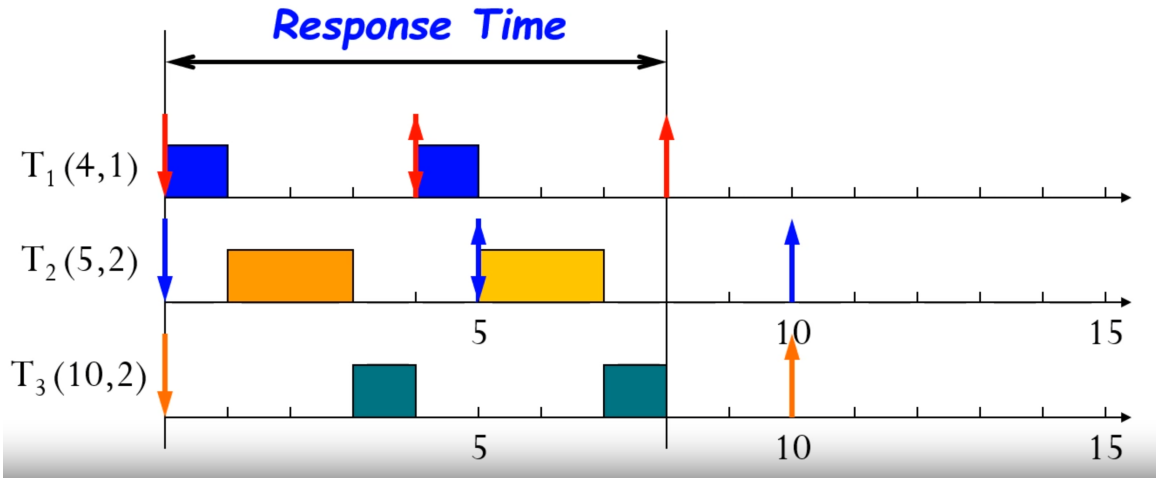
- Job-level fixed-priority (JFP)

  EDF(earlist deadline first) → optimal dynamic-priority

RM(Rate Monotonic)

- Response Time Analysis

Under RM scheduler, below is an example for checking response time for Task 3.



Deadline miss should not occur... → simulate until the LCM of periods? but takes too much time...

Therefore, Calculate Response Time: (Finish Time) - (Job Release Time)

= (Execution Time) + (Interference Time)

- Real-time system is schedulable under RM
  if and only if $r_i \le p_i$ for all task $T_i(p_i,e_i)$

Calculate Response Time with Iteration)

- Response Time (r_i) [Audsley et al., 1993]

$$r^{(k)}{}_i = e_i + \sum_{T_j \in HP(T_i)} \left\lceil \frac{r^{(k-1)}{}_i}{p_j} \right\rceil \cdot e_j$$

iter 1) interval = 2 → interference = 1×1 + 2×1 = 3 → increase interval into 2+3 = 5

iter 2) interval = 5 → interference = 1×2 + 2×1 = 4 → increase interval into 2+4 = 6

iter 3) interval = 6 → interference = 1×2 + 2×2 = 6 → increase interval into 2+6 = 8

iter 3) interval = 8 → interference = 1×2 + 2×2 = 6 → interval converges into 8

Just Calculating the correctness of first job guarantees the correctness of entire jobs.

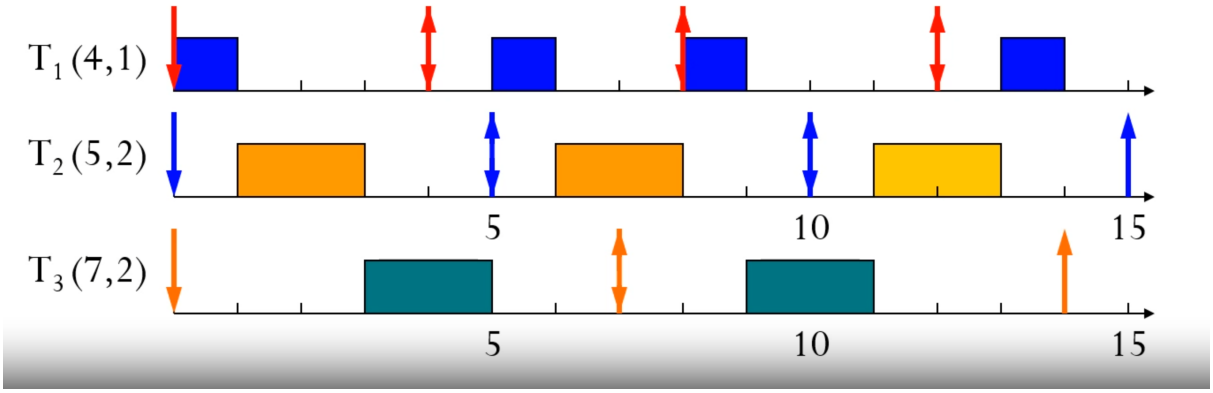
- Utilization Bound Analysis

  - Real-time system is schedulable under RM if
    $$\sum U_i \le n\,(2^{1/n}-1)$$

    Liu & Layland,
    "Scheduling algorithms for multi-programming in a hard-real-time environment", Journal of ACM, 1973.

When the number of jobs(n) increases, Utilization Sum decreases, and converges into ln(2)=0.69

EDF(Earlist Deadline First)



EDF is Optimal Scheduling Algorithm!!!

- Demand-Supply Comparison Analysis

- Real-time system is schedulable under EDF
  if and only if **dbf(t) ≤ t** for all interval t

  Baruah et al.
  "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor", Journal of Real-Time Systems, 1990.

- Demand Bound Function : dbf(t)
  - the maximum processor demand by workload over any interval of length t

- Utilization Bound Analysis

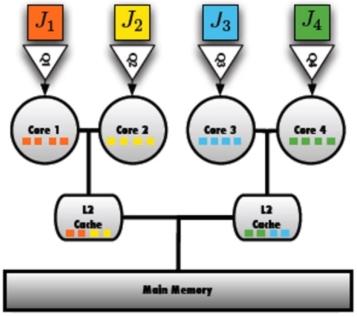
  - Real-time system is schedulable under EDF if and only if

    $$\sum U_i \leq 1$$

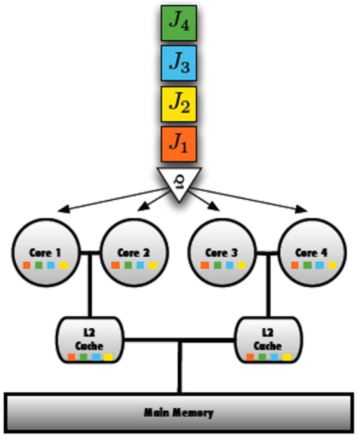
    Liu & Layland,
    "Scheduling algorithms for multi-programming in a hard-real-time environment", Journal of ACM, 1973.

III. Scheduling Algorithms in Multi-processor

Scheduling Approaches



Partitioned Scheduling
- task **statically** assigned to cores
- One ready queue **per core**
- uniprocessor scheduler on each core

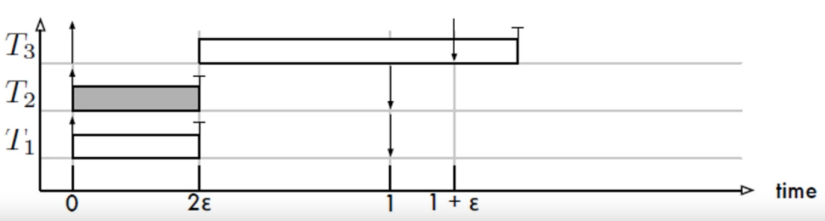


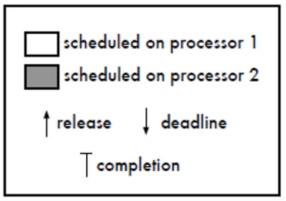
Global Scheduling
- jobs **migrate** freely
- All cores serve **shared** ready queue
- requires new schedulability analysis

Dhall Effect

A Difficult Task Set
- $m + 1$ tasks
- First m tasks — (T_i for $1 \leq i \leq m$):
  - Period $= 1$
  - WCET: 2ε
- Last task T_{m+1}
  - Period $= 1 + \varepsilon$
  - WCET $= 1$

Total utilization?





When used EDF for Global Scheduling, this problem occurs.

Utilization Bounds

- For $\varepsilon \to 0$, the **utilization bound approaches 1**.
- Adding processors makes no difference!

Global vs. Partitioned Scheduling

- Partitioned scheduling is easier to implement.
- Dhall Effect shows limitation of global EDF and RM scheduling.
- Researchers lost interest in global scheduling for ~25 years.

Since late 1990ies…

- It's a limitation of EDF and RM, not global scheduling in general.
- Much recent work on global scheduling.

Partitioned Scheduling

Reduction to *m* uniprocessor problems

- Assign each task **statically** to one processor
- Use uniprocessor scheduler on each core
  - Either fixed-priority (**P-FP**) scheduling or EDF (**P-EDF**)

Find task mapping such that

- No processor is **over-utilized**
- Each partition is **schedulable**
  - trivial for implicit deadlines & EDF

The mapping problem is a NP-hard problem… Using Deep Learning, we can seek for nice solutions.

However, there are some limitations in partitioned scheduling,

Theorem: there exist task sets with utilizations arbitrarily close to $(m+1)/2$ that cannot be partitioned.

Andersson, B., Baruah, S., and Jonsson, J. (2001). Static-priority scheduling on multiprocessors. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 193–202.

A difficult-to-partition task set

- $m + 1$ tasks
- For each T_i for $1 \leq i \leq m + 1$:
  - Period = 2
  - WCET: $1 + \varepsilon$
  - Utilization: $(1 + \varepsilon) / 2$

Partitioning not possible

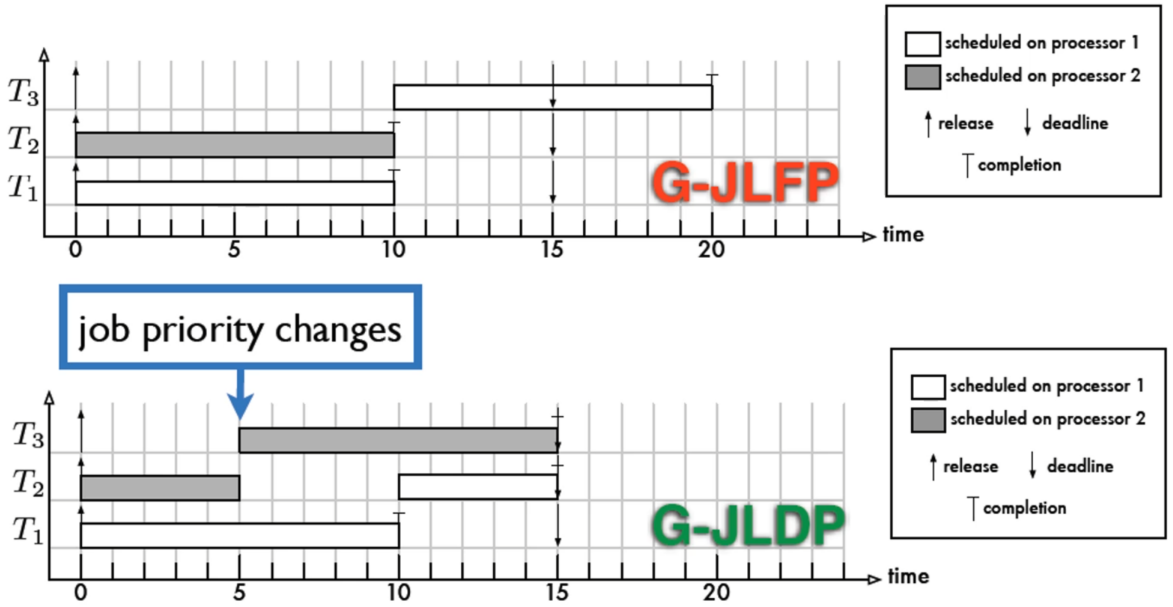
- Any two tasks together over-utilize a single processor by ε!
- Total utilization = $(m + 1) \cdot (1 + \varepsilon) / 2$

Therefore, global scheduling algorithm that can enhance the total utilization very close to the number of processors was proposed (other than just using EDF). But this algorithm has constraint that the deadline has to be same with the period of the task.

Job-Level Dynamic-Priority (JLDP) Scheduler

- No restrictions.
- The priority of each job changes over time.
- Priorities are a function of time, job identity, and system state.

But, no general critical instant is known! → the first job does not guarantee the schedulability of the whole tasks.



Pfair/PD2, ...etc: other many global-JLDP Algorithm has been proposed.