SEMI-PARTITIONED FIXED-PRIORITY SCHEDULING ON MULTIPROCESSORS

Abhinav Siddharth 2020112007
Shriya Dullur 2020102006

MOTIVATION

In Semi-partitioned scheduling, most tasks are fixed to particular processors to reduce runtime overhead, while a few tasks migrate across processors to improve schedulability.

According to the prior work, a class of semi-partitioned scheduling offers a significant improvement on schedulability, as compared to a class of partitioned scheduling, with less preemptions and migrations than a class of global scheduling

The aim of this paper is **to improve schedulability** with succeeding the **advantage of partitioned scheduling** as much as possible.

This paper presents a new algorithm for **fixed priority scheduling of sporadic task** systems on multiprocessors.

System Model

- System is composed of m homogenous processors(P_1, P_2, ..., P_m) and n sporadic tasks (T_1, T_2, ..., T_n)
- Task T_i is characterized by (c_i,d_i,p_i)
- ci : worst case execution time
- di : relative deadline
- pi : minimum inter-arrival time (period)
- For each task Ti ,we assume ci ≤ di ≤ pi
- Each Task generates infinite sequence of jobs. A job released at time t has a deadline at time t+di .
- Jobs of Ti are executed sequentially and successive jobs of Ti are separated by period pi
- Each task is independent and preemptive.

1

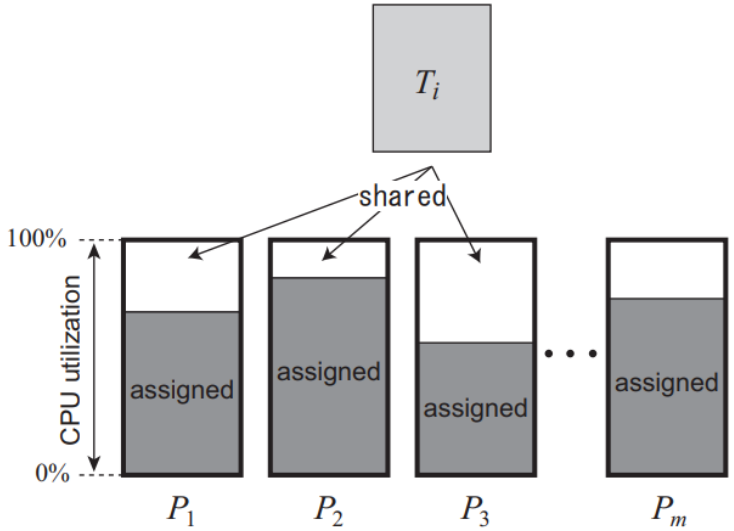
Deadline Monotonic with Prioirty Migration

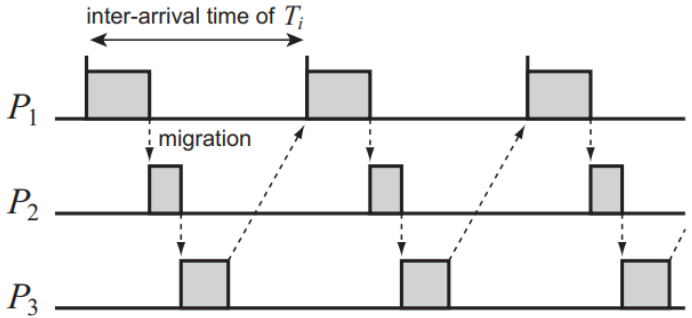
(DM-PM)

ALGORITHM DESCRIPTION

- Task Assignment to processor **: Bin-packing Heuristics + Schedulability conditions for DM**
- **Shared Task** : task becomes shared if none of the m processors has spare capacity enough to accept full share of a task T_i.
- Task is shared among multiple processors. The share is always assigned to processors with lower indexes. The execution capacity is then given to each share so that the corresponding processors are filled to capacity.
- The processors have **no spare capacity to receive other tasks**, once a shared task is assigned to them.
- A shared task T_i can be thus regarded as an independent task with an execution time $c'_{i,k}$ and a minimum inter-arrival time p_i , to which the highest priority is given, on every processor P_k.
- **Migration** : the shared task is qualified to migrate across the processors among which the task is shared

EXAMPLE OF TASK SHARED BETWEEN 3 PROCESSORS





DMPM Scheduling Policy

- A shared task is scheduled by the **highest priority** within the execution capacity on each processor
- Every job of the shared task is released on the processor with the **lowest index,** and it is sequentially migrated to the next processor when the execution capacity is consumed on one processor.
- Partitioned tasks are then scheduled according to **DM**

According to the scheduling policy of DM-PM, the execution of a shared task Ti is repeated exactly at its interarrival time on every processor, because it is executed by the highest priority within a time interval of a constant length on each processor

MULTIPROCESSOR SCHEDULING

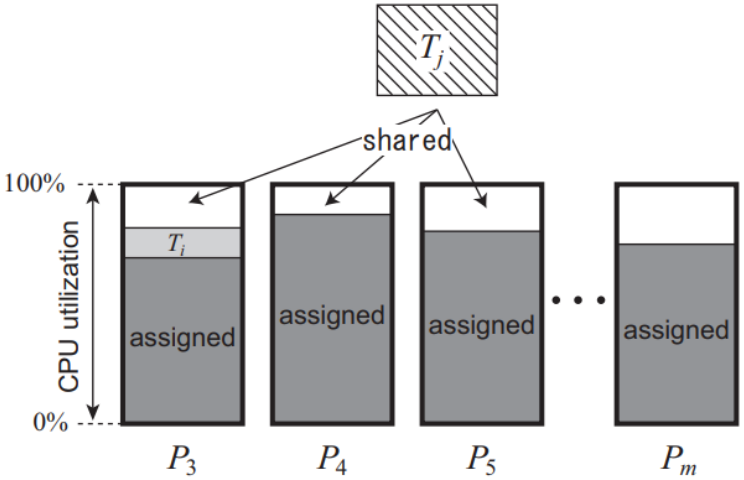
Allocation Problem: On which processor a task should execute?

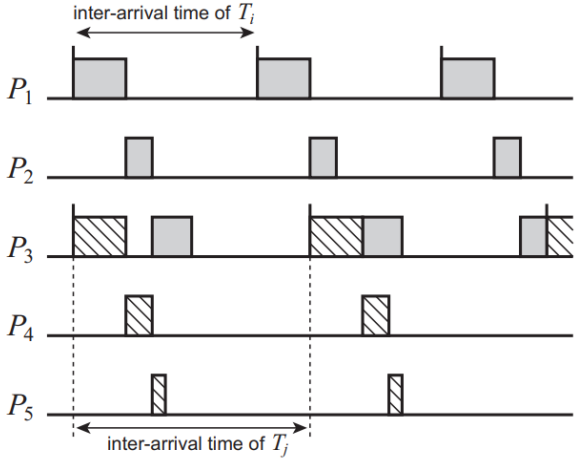
Job Level Migration: A single job can migrate to and execute on different processors; however, **parallel execution of a job is not permitted**.

Priority Problem: What order with respect to other jobs, each job should execute?

Fixed task priority: Each task has a single **fixed priority** applied to all of its jobs.

TWO SHARED TASKS





Execution capacity of shared tasks

- In Fixed priority scheduling, the worst-case response time of a task is obtained at critical instant.
- We assume all the tasks are released at critical instant.
- For two tasks T_i and T_j (fixed or shared), T_i is assigned with lower priority than T_j.
- The worst-case response time $R_{i,k}$ of each task T_i on P_k is given by

$$R_{i,k} = \sum_{T_j \in \mathcal{P}_k \cap \mathcal{H}_i} I_{i,j}(d_i) + c_i$$

- Where $\mathcal{P}_k$ is a set of tasks that have been assigned to processor P_k, and H_i is a set of tasks that have priorities higher than or equal to T_i . $I_{i,j}(d_i)$ is the maximum interference received by T_i by the higher priority task T_j

The maximum amount of time that a shared task Ts competes with a task T_i on processor P_k within a time interval of length d_i is given by,

$$W_{s,k}(d_i) = \left\lceil \frac{d_i}{p_s} \right\rceil c'_{s,k}$$

Where, p_s is the period of the shared task and $c'_{s,k}$ is the execution time of task Ts on processor P_k

For all tasks to meet deadlines, the following must hold for every task T_i on every processor P_k to which a shared task is assigned.

$$R_{i,k} + W_{s,k}(d_i) \leq d_i$$

$$c'_{s,k} = \min \left\{ \frac{d_i - R_{i,k}}{G} \;\middle|\; T_i \in \mathcal{P}_k \right\} \quad \text{where } T_i \in \mathcal{P}_k \text{, and } G = \lceil d_i/p_s \rceil$$

The schedulable condition of T_i for P_k here is defined by $R_{i,k} \leq d_i$. If T_i does not satisfy the schedulable condition, its utilization share is going to be split across processors.

Optimization

- Sorts a task set in non-increasing order of relative deadline before the tasks are assigned to processors.
- According to DM-PM, Ts is assigned to each P_k so that P_k is filled to capacity, except that Pl is a last processor to which T_s is assigned.
- There is no need to forcefully give the highest priority to Ts on Pl , because the next job of Ts will be released at its next release time, regardless of its completion time, whereas it is necessary to give the highest priority to Ts on the preceding processors.
- $$U_k \leq n_k(2^{1/n_k} - 1)$$
- Worst-case, relative deadline=period, utilization=69%.
- Assume shared task Ts on Pl. We then assume that another task T_i is later assigned to P_l .

$$d_s - c_s = d_s(1 - u_s), \text{ due to } d_i \leq d_s$$

Optimization

$$u_i = \frac{d_s(1-u_s)}{d_i} \geq (1-u_s)$$

- Worst case execution share of T_i on P_l: $u_i \geq 1-u_s$. (if u_s is high, u_i is 0 regardless of P_l)
- Hence, tasks with individual utilization greater than or equal to 50% are preferentially assigned to processors, before a task set is sorted in non-increasing order of relative deadline.
- Since no tasks have individual utilization greater than 50%, when T_s is shared among processors, the worst-case execution capacity of T_i is improved to $u_i = 1 - u_s \geq 50\%$.
- As a result, the optimized DM-PM guarantees that the processor utilization of every processor is at least 50%, which means that the entire multiprocessor utilization is also at least 50%. (worst-case)

2

Simulation setup and Results

PRIOR WORK

P-DM

P-DM algorithm is based on Partitioned Scheduling. PDM assigns tasks based on first-fit heuristic for simplicity without sorting a task set. The tasks are then scheduled according to DM.

PDM uses a response time analysis in partitioning phase.

FBB-FDD uses a polynomial-time acceptance test in a partitioning phase, while P-DM uses a RTA.

Rate-Monotonic Deferrable Portion

RMDP algorithm is based on Semi-Partitioned scheduling. It is designed for implicit deadline systems. The highest priority is given to migratory tasks, and other fixed tasks have the Rate Monotonic priorities. RMDP improves schedulability over the traditional fixed-priority algorithms.

Simulation Setup

Number of processors

Number of processors(m)

{ 2, 4, 8, 16, 32, 64}

System Utilization

Usys : values are taken in the range [0.5, 1]

Number of Tasks

1000 tasksets ; Each task set is generate such that the total utilization is equal to Usys *m

Uunifast Algorithm is used to generate tasksets

.

Minimum inter-arrival time

The minimum inter-arrival time of each task is uniformly distributed in the range of [100, 10000]

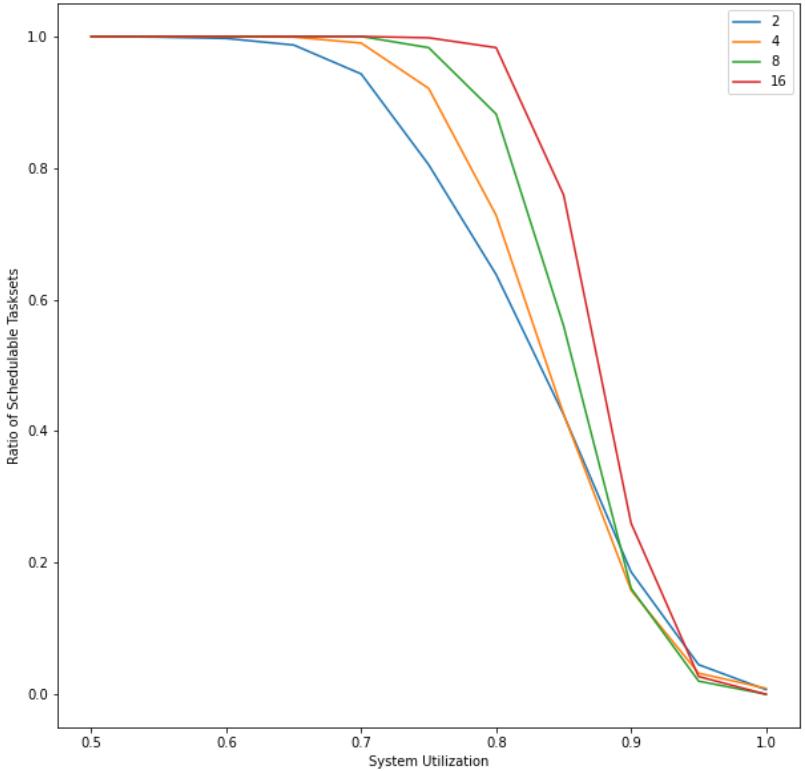
Evaluation metric

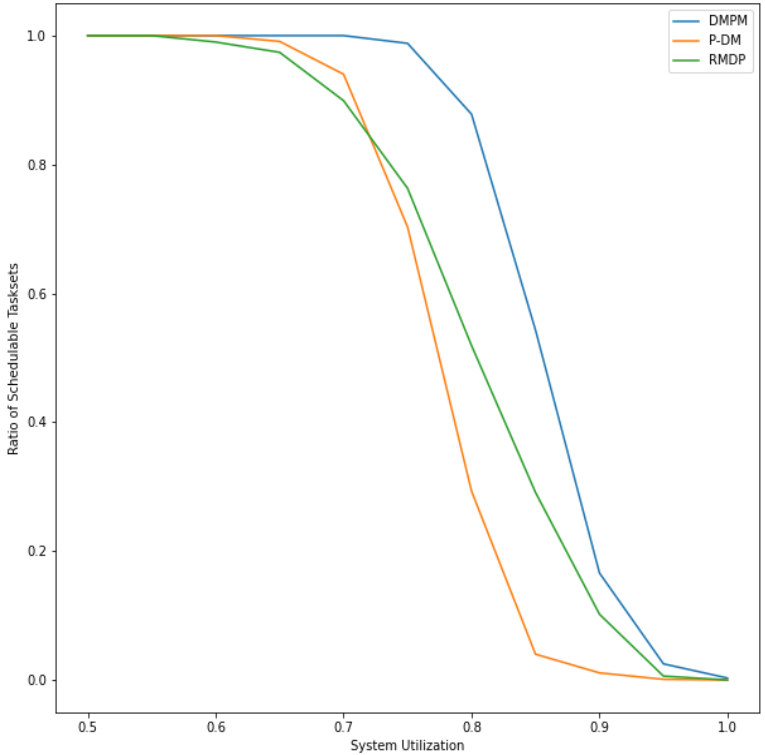
Success ratio = The number of successfully scheduled task sets / the number of submitted task sets

Minimum and maximum Task utilizations

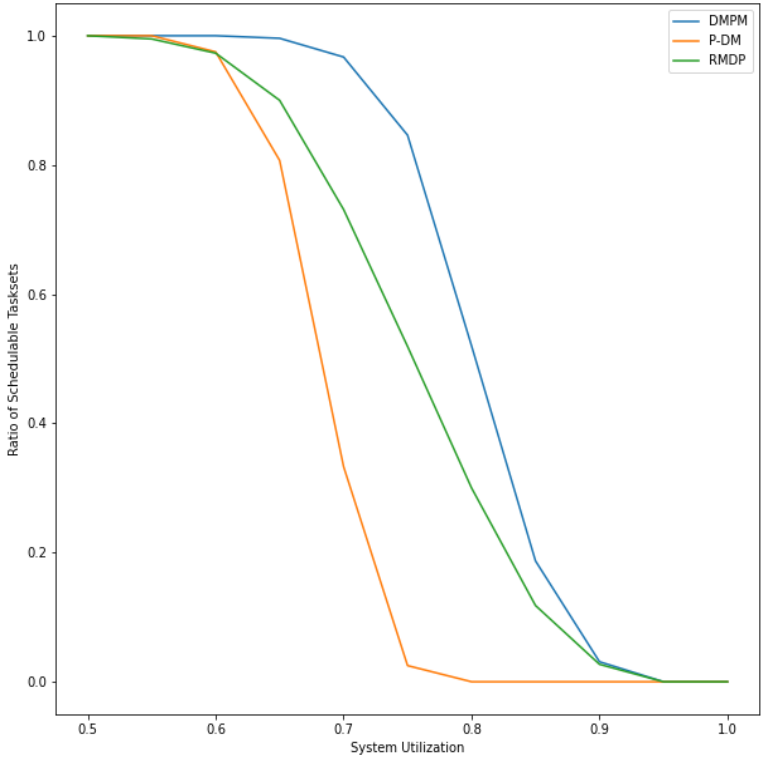
IMPLEMENTATION

Comparison of DM-PM algorithm with different number of processors

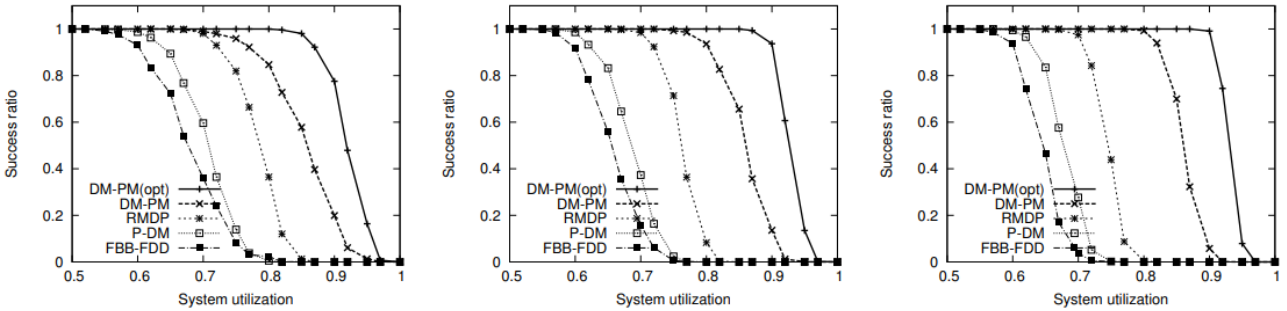




[umin,umax]=[0.1,1], m=4



[umin,umax]=[0.4,0.6], m=4



(a) $m = 4$ (b) $m = 8$ (c) $m = 16$

Figure 9. Results of simulations ($[u_{min}, u_{max}] = [0.25, 0.75]$).

Observations

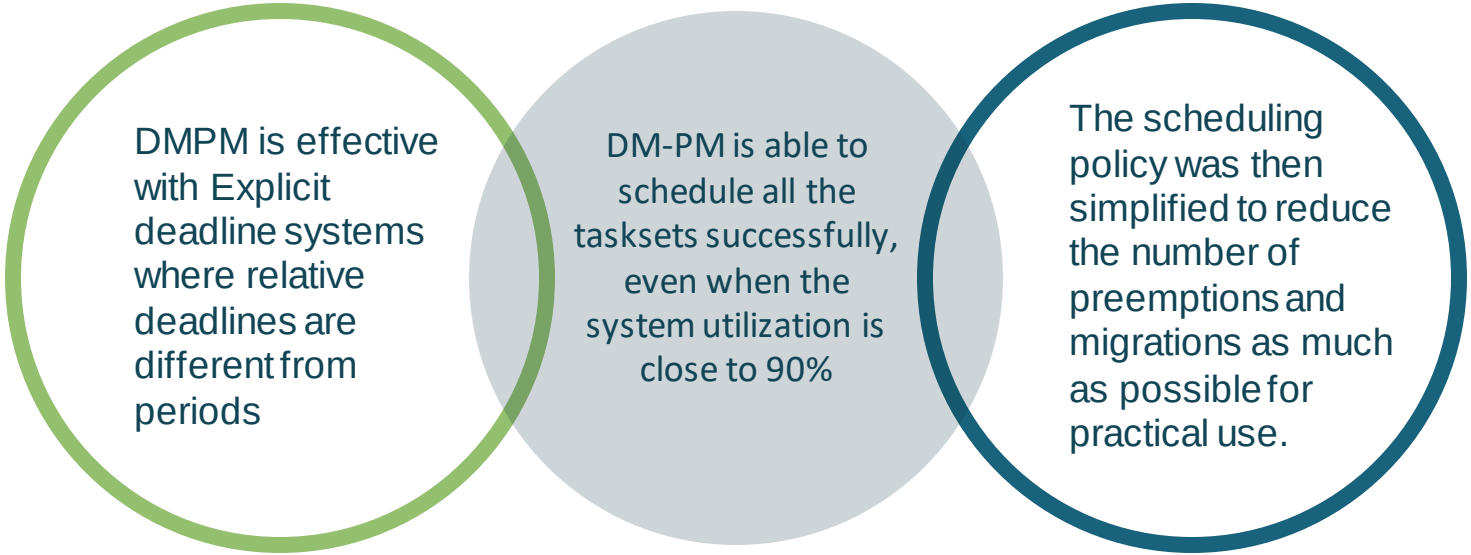
DM-PM is able to schedule all task sets when system utilization is smaller than 0.7 to 0.8

A task set is said to be successfully scheduled, if all tasks in the task set are successfully assigned to processors.

DM-PM is able to schedule all the tasksets successfully, even when the system utilization is close to 90%

The performance of DM-PM is better as the number of processor is greater, because tasks are more likely to be successfully shared among processors.

Advantages of DMPM



Flaws and improvements

The costs of scheduler invocations, preemptions, and migrations are not modeled.

Lakshmanan et al. (2009) developed a semipartitioning method based on fixed, priority scheduling of sporadic tasksets with implicit or constrained deadlines called PDMS_HPTS. It splits a single task on each processor. Utilization bound of at least 60% for tasksets with implicit deadlines.

PDMS_HPTS_DS (Decreasing density) has a utilization bound of 69.3% if the maximum utilization of any individual task is no greater than 0.41

Thank You !