Semi-Partitioned Fixed-Priority Scheduling on Multiprocessors

Shriya Dullur 2020102006

Abhinav Siddharth 2020112007

Deadline Monotonic with Priority Migration (DM-PM) is based on the concept of semi-partitioned scheduling. In consideration of the migration and pre-emption costs, a task is qualified to migrate, only if it cannot be assigned to any individual processors, in such a way that it is never returned to the same processor within the same period, once it is migrated from one processor to another processor.

Here, the most important condition is that no subtasks split from one task run in parallel. In such a way that a task never returns to the same processor within the same period once it is migrated from one processor to another processor.

During optimization of DM-PM algorithm, we have to sort all the tasks in the task set which is not applicable for an online scheduler.

Algorithm Description

- DM-PM assigns each task to a particular processor (m identical processors), according to kinds of bin-packing heuristics, upon which the schedulable condition for DM is satisfied.
- If none of the m processors has spare capacity enough to accept full share of a task Ti(n sporadic tasks), the task is shared among multiple processors.
- Ti's execution time is "split" into different portions. 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.
- 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
- A shared task Ti can be thus regarded as an independent task with an execution time $c'_{(i,k)}$ and a minimum inter-arrival time pi to which the highest priority is given, on every processor Pk.

Taskset Generation

Task sets are generated such that the total utilization of taskset is equal to the (system utilization) *(number of processors).

A task set T is generated as follows. A new periodic task is appended to T as long as U(T) ≤ U_{tot} is satisfied. For each task T_i, its utilization u_i is computed based on a uniform distribution within the range of $[u_{min}, u_{max}]$. Only the utilization of the task generated at the very end is adjusted so that U(T) becomes equal to U_{tot} (Thus the u_{min} constraint might not be satisfied for this task).

Evaluation

We show the results of simulations conducted to evaluate the effectiveness of DM-PM with other algorithms such as RMDP and Partitioned DM. Success ratio is used to compare different scheduling algorithms.

Simulation Setup

Every simulation has a set of parameters: u_{sys}, m, u_{min} and u_{max}. u_{sys} denotes system utilization, m is the number of processors, u_{min} and u_{max} are the minimum and maximum utilization of each individual task.

For every set of parameters, we generate 10,000 task sets. A task set is said to be successfully scheduled, if all tasks in the task set are successfully assigned to processors.

Since we know the utilization bound for the DM-PM algorithm is 0.5, the system utilization u_{sys} is set to values 0.5, 0.55 … 0.95 and 1.

We have three sets of m: 4, 8 and 16.

Each taskset is generated such that total utilization becomes equal to $u = m \times u_{sys}$.

The utilization of every individual task is uniformly distributed within the range of [u_{min},u_{max}].

We have five sets of [u_{min},u_{max}]: [0.1, 1], [0.25, 0.75], [0.4, 0.6], [0.5, 1] and [0.1, 0.5]

The minimum inter-arrival time of each task is also uniformly distributed within the range of [100, 1,000].

For each task, once u_i and p_i are determined, we compute the execution time of the task by $c_i = u_i \times p_i$.

Since RMDP is designed for implicit-deadline systems, for fairness we presume that all tasks have relative deadlines equal to periods. However, DM-PM is also effective to explicit-deadline systems where relative deadlines are different from periods.

Simulation Results

The performance of DM-PM is better as the number of processors is greater, because tasks are more likely to be successfully shared among processors, if there are more processors, when they cannot be assigned to any individual processors.

Since RMDP is also able to share tasks among processors, it outperforms P-DM that is based on classic partitioned scheduling. However, DM-PM performs better than RMDP.

While the performance of DM-PM is better as the number of processors increases, P-DM generally degrades in performance. RM-DP depends on the task utilizations, but usually it has a marginal increase in schedulability as the number of processors increases.

Algorithms Implemented

- DM-PM: Deadline Monotonic with Priority Migration Scheduling Algorithm
- RMDP: Rate Monotonic Deferrable Portion Scheduling Algorithm (Kato et al. 2008) Highest priority assigned to migratory tasks, fixed tasks have RM priority. RMDP uses Liu and

Layland's bound to schedule fixed tasks and then the remaining tasks are split and allowed to migrate based on the Liu and Layland bound.

- P-DM: Partitioned Deadline Monotonic Scheduling Algorithm P-DM assigns tasks based on first-fit heuristic for simplicity without sorting a task set. P-DM uses a response time analysis in the partitioning phase.
- FBB-FDD: Fisher Baruah Baker- First Fit Decreasing Scheduler (Fisher et al., 2005) FBB-FDD sorts a task set in non-decreasing order of relative deadline and assigns tasks to processors based on a first-fit heuristic FBB-FDD uses a polynomial time acceptance test in the partitioning phase. (acceptance test not implemented)

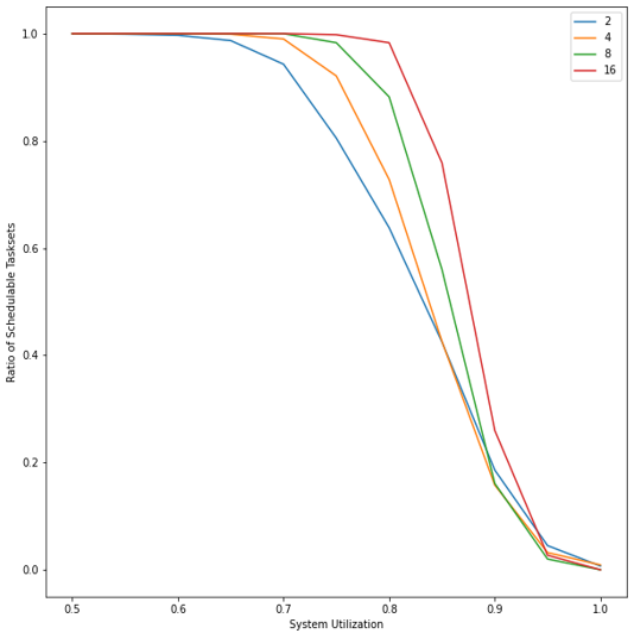

Figure 1: General trend of DM-PM vs number of processors

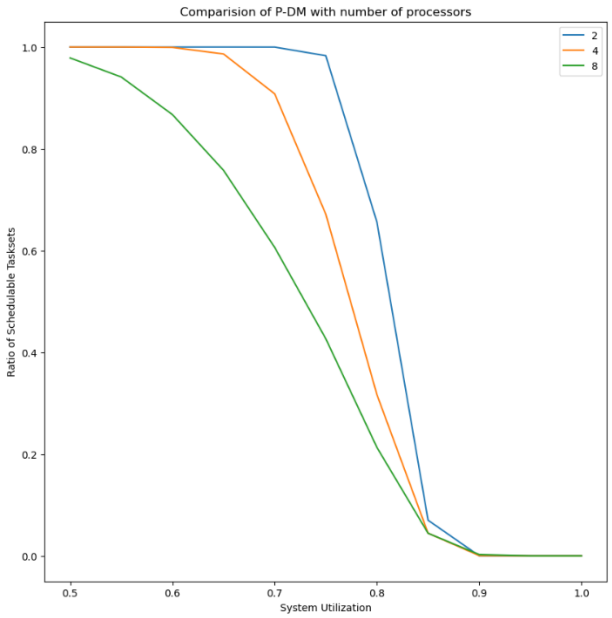

Figure 2: General trend of P-DM vs number of processors

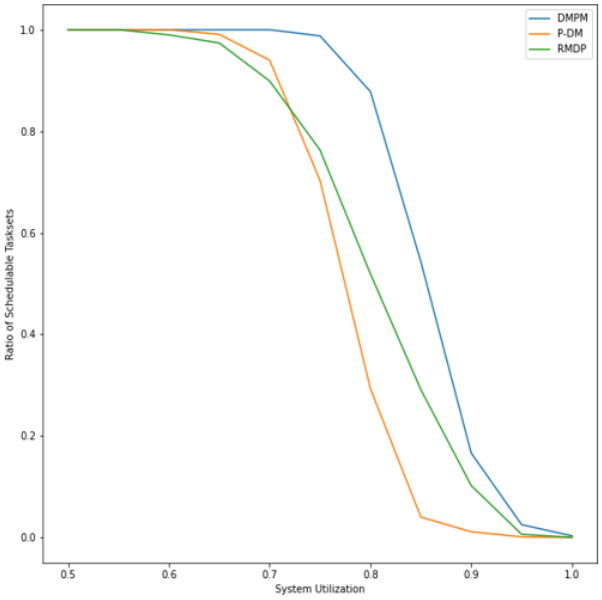

Figure 3: Comparison of algorithms for [0.1, 1] and m=4

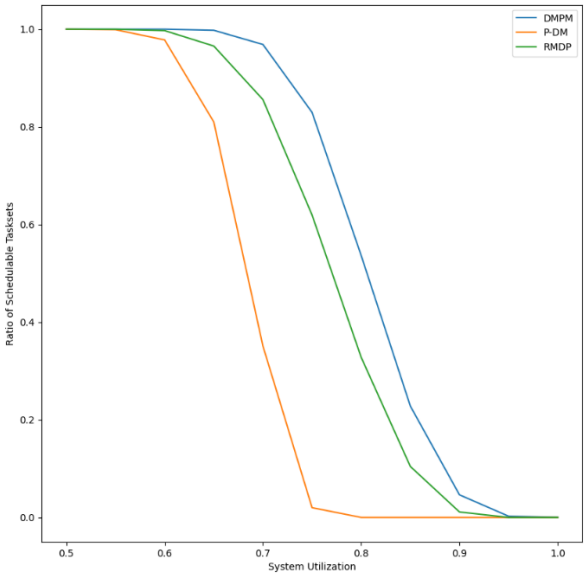

Figure 4: Comparison of algorithms for [0.25, 0.75] and m=4

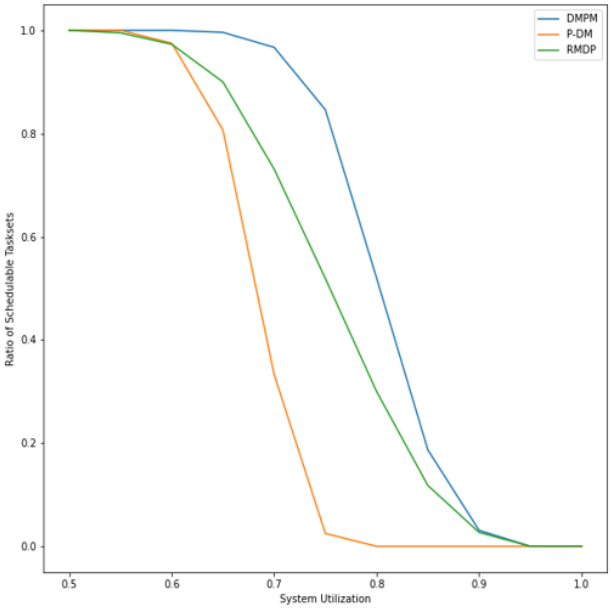

Figure 5: Comparison of algorithms for [0.4, 0.6] and m=4

From this result, we see that DM-PM prefers the case in which the utilization of every individual task is widely distributed.

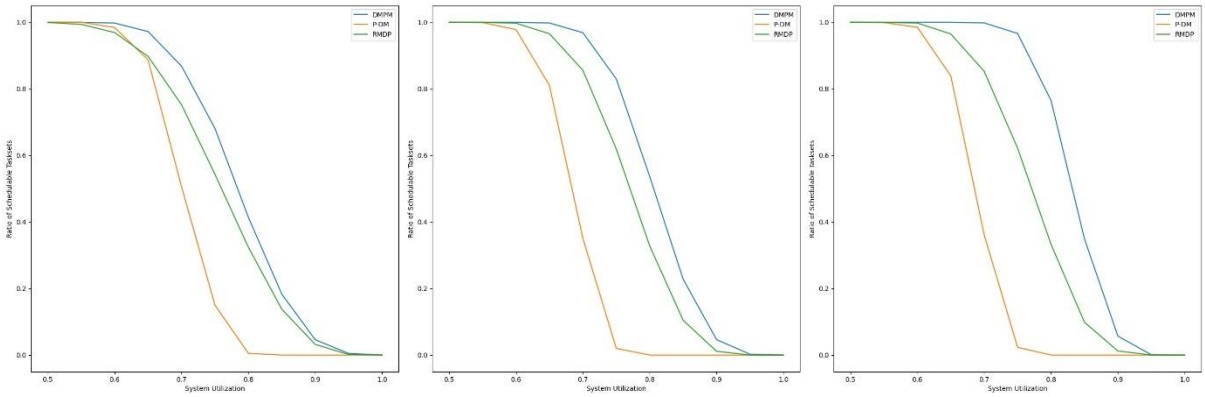

Figure 6: Comparison of algorithms for [0.25, 0.75] for m=2, 4 and 8

Properties of DM-PM

- Thus, we can see that DM-PM dominates RMDP and P-DM as all tasksets which aren't schedulable by the latter are schedulable by DM-PM as well as those which are schedulable by RMDP and P-DM are schedulable.
- DM-PM has utilization bound of 50% for tasksets with implicit deadlines.
- DM-PM is predictable. i.e. response times of jobs cannot be increased by decreases in their execution times. (by design and also proven by Ha and Liu 1994.)
- Since P-DM uses First-Fit bin packing, it is subject to timing anomalies. Increase in periods might lead to a different allocation which could be unschedulable. Since DM-PM also uses bin packing initially, it may lead to a different processor allocation which could lead to timing anomalies.
- DM-PM can exploit unused processor time of partitioned algorithms. However, this has a task migration overhead, which is not considered in the system model.

- Exact response time analysis is known to have pseudo-polynomial complexity; however, our implementation is much more efficient. Thus, our algorithm can be used to determine schedulability of large real time systems in a reasonable time scale.