Comprehensive Analysis of Multiprocessor Scheduling Algorithms

Students Charels Hugo
Course INFO-F404
Instructors Joël Goossens, Yannick Molinghen

Abstract

This report explores parallelization strategies and schedulability analysis methods in real-time scheduling. We examine theoretical and practical aspects of parallelization, presenting an experimental evaluation of task sets under partitioned EDF algorithms with varying worker counts. The report visualizes and explains decision-making processes for schedulability analysis, with a focus on feasibility intervals. Empirical results are analyzed to draw insights into the performance and efficiency of these approaches.

1 Introduction

Scheduling tasks in real-time systems is critical for ensuring deadlines are met. This report focuses on two key aspects:

1. **Parallelization:** Identifying components suitable for parallelization and analyzing the impact of worker counts on execution time.
2. **Schedulability Analysis:** Developing methodologies to determine the feasibility of task sets, including the choice of appropriate feasibility intervals.

The report includes experimental results and theoretical discussions to offer a comprehensive perspective on efficient scheduling.

2 Materials and Methods

The implementation of this project involved the coding of partitioned EDF, EDF$^{(k)}$, and global EDF algorithms, as well as a scheduler simulator and the parsing of task sets provided in CSV format. Each task in the set is characterized by four parameters: offset O_i, computation time C_i, deadline D_i, and period T_i. The task set input follows the format:

```
0, 20, 40, 50
10, 80, 200, 200
```

2.1 Scheduling Algorithms

Partitioned EDF, EDF$^{(k)}$ and global EDF are common scheduling algorithms used in multiprocessor task scheduling, each with distinct characteristics:

2.1.1 Partitioned EDF

Partitioned EDF assigns tasks to processors based on heuristics:

- **First-fit:** Assign to the first eligible processor.
- **Best-fit:** Assign to the processor with maximum load that can accommodate the task.
- **Worst-fit:** Assign to the processor with minimum load.
- **Next-fit:** Assign to the current processor; move to the next if it cannot fit.

Tasks can be sorted by utilization:
We define the utilization of the task i by $U(\tau_i) = \frac{C_i}{T_i}$.

- **Decreasing Utilization:** Higher utilization tasks first.
- **Increasing Utilization:** Lower utilization tasks first.

2.1.2 EDF$^{(k)}$

Tasks are sorted by decreasing utilization, and the highest priority is assigned to the k first tasks.

2.1.3 Global EDF

Global EDF imposes no partitioning restrictions, enabling full task migration across processors.

3 Schedulability Conditions

Schedulability analysis involves determining whether a task set meets all deadlines under a given scheduling policy. Key conditions and algorithms for determining schedulability are discussed below.

We use utilization-based criteria for task schedulability:

Necessary Condition

Any task set τ is not **schedulable** if:

$$U(\tau) > m \quad \text{or} \quad U_{\max} > 1,$$

where $U(\tau)$ is the total utilization and $U_{\max}$ is the maximum utilization of a single task.

3.1 Partitioned EDF

Sufficient Condition

Using the FFDU heuristic, any sporadic-implicit deadline task set τ is **schedulable** if:

$$U(\tau) \leq \frac{m+1}{2} \quad \text{and} \quad U_{\max} \leq 1.$$

3.2 EDF$^{(k)}$

Sufficient Condition

Any task set τ is **schedulable** if:

$$|\tau| \geq m \quad \text{and} \quad U_{\max} \leq 1.$$

where $|\tau|$ is the number of tasks in the task set.

Sufficient Condition

Any sporadic implicit-deadline task set τ is **schedulable** if:

$$m \geq (k-1) + \left\lceil \frac{U(\tau^{(k+1)})}{1 - U(\tau_k)} \right\rceil.$$

3.3 Global EDF

Sufficient Condition

Any task set τ is **schedulable** if:

$$|\tau| \geq m \quad \text{and} \quad U_{\max} \leq 1.$$

where $|\tau|$ is the number of tasks in the task set.

Sufficient Condition

Any sporadic implicit-deadline task set τ is **schedulable** if:

$$U(\tau) \leq m - (m-1) \cdot U_{\max}.$$

4 Scheduler Simulator

The scheduler simulator determines if a task set is schedulable by applying the chosen algorithm. It computes the feasibility interval for each algorithm, which represents the time range during which all tasks must meet their deadlines. This interval is used to assess whether the task set can be scheduled over an infinite time horizon.

4.1 Partitioned EDF

Once tasks are assigned to individual processors, each processor independently simulates the scheduling of its assigned tasks. This independence arises because no task migration occurs between processors, meaning the state of one processor does not impact another.

Feasibility Interval: For each processor, the interval is defined as

$$[0, P)$$

where P is the hyperperiod of tasks assigned to that processor. This interval ensures that all tasks assigned to a processor are verified for deadline adherence.

4.2 EDF$^{(k)}$ and Global EDF

Simulating these algorithms requires a sequential computation to account for the dynamic interactions between tasks and processors. This interdependence arises because we cannot predict, or 'see' the future state of the schedule without sequentially progressing through the timeline.

Feasibility Interval: For EDF$^{(k)}$ and Global EDF, the feasibility interval is defined as

$$[0, \min(O_{\max} + 2P, 1\,000\,000))$$

where $O_{\max}$ is the maximum offset of the task set, and P represents the hyperperiod of the task set. This interval accounts for task interactions across all processors within a bounded simulation window. The interval is bounded by a maximum of $1,000,000$ time units to avoid excessive computation times. This limit ensures that the simulator remains efficient, as the LCM can grow very large for certain task sets.

5 Parallelization

Partitioned EDF can be parallelized once tasks are assigned to specific processors, as the scheduling simulation can be conducted independently for each processor. This independence arises because each processor manages its own tasks without needing to account for the state of other processors. In contrast, EDF$^{(k)}$ and Global EDF cannot be parallelized in the same way. These algorithms require a global view of the system's state at every moment t, as tasks can migrate between processors or be dynamically reassigned based on their deadlines. Without a "crystal ball" to predict the exact scheduling state across all processors, parallelizing the simulation for these algorithms is inherently infeasible.

Additionally, many operations within the simulation could be parallelized. For example, spawning jobs and checking whether deadlines are missed. However, this approach has not been employed here because the task sets under consideration involve only a few dozen tasks. In such cases, the overhead of parallelization would outweigh its potential benefits.

6 Experimental Plan

The experimental plan is structured as follows, with all executions performed using 8 cores:

1. Measure the average execution time of the simulation for partitioned EDF using the BFDU heuristic on a data set of task sets, varying the number of workers from 1 to 32. Each task set is simulated 100 times to compute a reliable mean execution time.

2. Plot the average execution time as a function of the number of workers and analyze the results to evaluate scalability.
3. Measure the success rates of all heuristics and task-ordering strategies for partitioned EDF.
4. Identify the best combination of heuristic and task ordering supported by plots.
5. Evaluate the success rate of Global EDF and EDF$^{(k)}$ for all possible values of k, comparing their performance between task sets.
6. Determine the best-performing algorithm between Global EDF and EDF$^{(k)}$, presenting the findings with comparative plots.
7. Compare the best Partitioned EDF configuration with the best-performing EDF-based algorithm.

7 Experimental Results

Figure 1 presents the mean duration required to determine the schedulability of each task set using simulation, as a function of the number of workers. The results indicate an approximately linear relationship between the duration and the number of workers. Notably, the computation is significantly faster with a single worker compared to 32 workers.

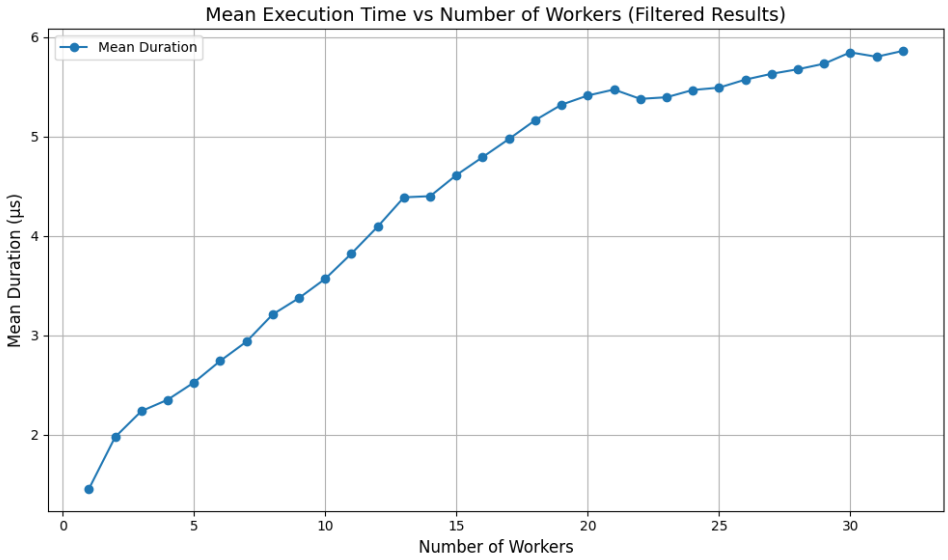

Figure 1: Mean duration to determine schedulability as a function of the number of workers.

Figure 2 compares the success rates of various heuristics and orderings for partitioned EDF scheduling. The success rate represents the percentage of task sets deemed schedulable under each heuristic and ordering combination. Among the tested configurations, the Worst-Fit heuristic combined with decreasing utilization ordering achieved the highest success rate.

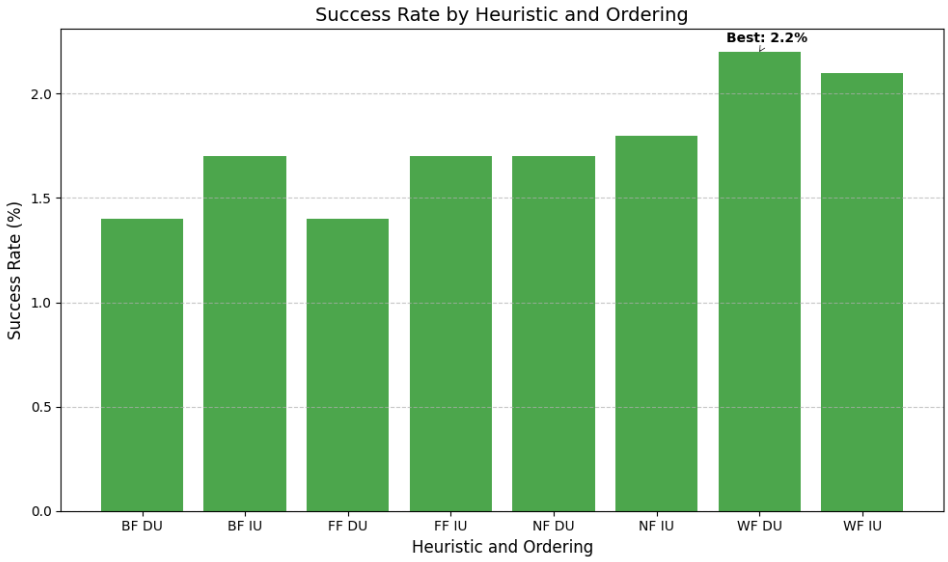

Figure 2: Success rate by heuristic and ordering for partitioned EDF scheduling.

Finally, Figure 3 illustrates the comparison of schedulable, unschedulable, and undetermined results for global EDF and various EDF$^{(k)}$ configurations, where $k \in \{0, \ldots, |\tau| - 1\}$. The proportion of schedulable task sets remains the same across all configurations. However, the results differ for unschedulable and undetermined task sets. Notably, EDF$^{(5)}$ exhibits the lowest percentage of unschedulable results and the highest percentage of undetermined outcomes, while global EDF shows the opposite trend.

Figure 3: Comparison of schedulable, unschedulable, and undetermined results for global EDF and EDF(k).

8 Discussion

The experimental results provide several key insights into the performance, feasibility, and practicality of the partitioned EDF, EDF$^{(k)}$, and global EDF scheduling algorithms.

First, the parallelization of partitioned EDF demonstrates significant efficiency gains. By dividing tasks across processors, each processor can independently schedule its assigned tasks, leading to highly efficient execution. However, as the simulation time for partitioned EDF takes only a few microseconds, it becomes challenging to demonstrate that threads are truly useful. The overhead of creating threads is comparable to the time required to find the solution using a single thread. Therefore, while parallelization offers theoretical benefits, it would be more meaningful to evaluate its performance with significantly larger task sets and a greater number of cores.

Second, the success rates of partitioned EDF, as shown in Figure 2, highlight the limitations of this approach. The percentage of task sets deemed schedulable with partitioned EDF is approximately 2%, which is a remarkably small amount. In comparison, global EDF and EDF$^{(k)}$ can also schedule around 2% of task sets by leveraging shortcuts in their global view of the system. This result implies that partitioned EDF performs worse than its global counterparts in terms of schedulability.

Additionally, the results for EDF$^{(k)}$ and global EDF introduce another key observation. For these algorithms, no feasible interval was identified that could conclusively determine schedulability. When there was no detectable error in the schedule, the result was marked as *undetermined*. This outcome suggests that the task set could be schedulable, or it may not be, but as the simulation runs until $t = 1{,}000{,}000$, the probability of unschedulability decreases as the simulation time increases. This extended time horizon provides a strong, albeit non-absolute, indication of the task set's schedulability.

Finally, Figure 3 compares EDF$^{(k)}$ and global EDF configurations. Notably, EDF$^{(5)}$ minimizes the percentage of unschedulable task sets at the cost of increasing undetermined outcomes. Global EDF, on the other hand, shows a higher percentage of unschedulable task sets but provides more deterministic results. This trade-off reflects the fundamental differences between EDF$^{(k)}$ and global EDF: increasing k improves flexibility and may increase the schedulability.

In conclusion, when comparing the best-performing partitioned EDF configuration (Worst-Fit with decreasing utilization) against EDF$^{(k)}$ and global EDF, partitioned EDF emerges as the least favorable approach in terms of schedulability success rates. While its parallelization capabilities are notable, the small percentage of schedulable task sets highlights its limitations in real-world scenarios.

9 Conclusion

This report analyzed and compared partitioned EDF, EDF$^{(k)}$, and global EDF scheduling algorithms, focusing on execution efficiency, feasibility intervals, and schedulability success rates. The experimental results demonstrate that partitioned EDF, while efficient in execution time, fails to provide competitive schedulability success rates compared to EDF$^{(k)}$ and global EDF. The small percentage of task sets deemed schedulable (approximately 2%) underscores the limitations of partitioned EDF.

EDF$^{(k)}$ and global EDF both exhibit better schedulability outcomes. However, the absence of a feasibility interval means that some results remain *undetermined*. Specifically, when no error was detected in the schedule, the task set may still be unschedulable, though the probability of unschedulability decreases as the simulation time extends to $t = 1{,}000{,}000$. This limitation highlights the need for further investigation into longer feasibility intervals and their computational costs.

In conclusion, while partitioned EDF offers fast execution times due to its parallelization potential, its low success rate in scheduling task sets makes it less practical for real-world systems. EDF$^{(k)}$ and global EDF, despite their sequential nature, provide more robust schedulability guarantees and should be preferred for systems requiring higher task set feasibility.

Disclaimer

This document was co-written with the assistance of generative AI, primarily for improving grammar, style, and clarity. The ideas, results, and analyses are the work of the authors.