A Review on Multi-core Partitioned Scheduling

Shahzaib Waseem
B.Eng. Electronics Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
shahzaib.waseem@stud.hshl.de

***Abstract*—This document is a model and instructions for LaTeX.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Computers contained just a single processor up until the late 1900's, around when computers were first introduced [1]. While the invention of computers was a remarkable achievement, these computers were far from perfect. Since they relied on a single CPU, only a single program instruction could be executed at a time [1]. The concept of parallelization was, of course, then unknown [1]. Such single-core computers were naturally more vulnerable to bottlenecks when interfacing with faster computer peripherals that may have needed to wait for the computer to complete the task at hand [1]. Additionally, the weak multitasking nature of these computers especially in the context of hard-real time systems in combination with their constrained memory motivated computer engineers to research and develop multi-core systems [1].

As the name suggests, multi-core systems contain more than one processing unit and are able to handle execution of multiple program instructions in parallel [2]. This addresses the shortcoming of single-core processors first in the area of computer peripherals waiting for the CPU to become free by having another processor to handle their interfacing tasks [2]. Next, multi-core processors relatively increase a computer's ability to do multitasking by dividing execution of tasks among their multiple processors [2]. Interestingly, this leads to another benefit of better power consumption, as the workload on a single core is reduced [2]. However, even multi-core systems have their flaws in certain scenarios.

Having multiple cores or processors in a system implies that these cores must cooperate by sharing resources. However, mutually exclusive ownership/usage of a resource by a process at any given point in time may cause blocking and starvation for other cores and tasks. This delay faced by cores may lead to missing task deadlines, and that too in hard real-time systems in particular can cause catastrophic outcomes. Another factor in missing deadlines in multi-core systems is allowing tasks to switch between cores by disturbing the continuity of their usage of a shared resource, and having them potentially wait for their turn in doing so again [5]. Therefore, scheduling algorithms tailored and specialized for multi-core systems are needed.

Scheduling algorithms for multi-core systems are not and can not be directly mapped from existing scheduling algorithms for single-core systems [3]. Two of the most important reasons for this is that there is no concept of shared resources in single-core systems, and that the compatibility and affinity between a task and its executing core is not considered in single-core systems [4]. Below are some of the factors that a scheduling algorithm running on a multi-core system must consider:

- Predictability [7]
- Resource Utilization [7]
- Synchronization [9]

Predictability is the ability to determine beforehand whether a set of tasks can be scheduled in a way such that all their deadlines are met [8]. This can naturally allow for computational power to be directed towards set of tasks for which success is foreseen, and readjustment of the set of tasks or the algorithm in case failure is foreseen [8]. Resource Utilization refers to maximizing throughput and minimizing idle times for every core in the system [7]. Doing so can increase productivity as tasks can be executed faster in larger quantities since the core responsible for them performs all the work it can instead of being occasionally idle [7]. This is important especially for real-time systems where achieving the least amount of delay in completing a set of tasks is crucial, as it is important for the system to be free for any incoming tasks in the future. Synchronization is important to prevent race conditions in real-time systems [9]. This can happen when multiple tasks attempt to concurrently read and/or modify a shared resource, and end up overwriting data that another task was originally using, thereby causing outdated information to be used in computations within the real-time multi-core system [9]. Synchronization is also important in avoiding deadlocks in a system. Deadlocks can turn out to be extremely troublesome and tricky to handle, considering that their very nature contributes to increasing the likelihood of a task missing its deadline [9].

Considering the factors described above, quite a few scheduling algorithms for multi-core systems have been developed. These task scheduling algorithms are of three categories:

- Global scheduling
- Semi-partitioned scheduling
- Partitioned scheduling

Partitioned scheduling algorithms, which is the focus of this

paper, includes the following [10]:

- Partitioned Earliest Deadline First (P-EDF)
- Partitioned Rate Monotonic Scheduling (P-RMS)
- Partitioned Deadline Monotonic Scheduling (P-DMS)

Partitioned EDF, RMS, and DMS, all are extensions of these original single-core algorithms that have been adjusted and modified in a way that they can be applied to multi-core algorithms [11]. Furthermore, due to the fact that these algorithms must run on multi-core systems, the original single-core EDF, RMS, or DMS run on a each core independently, depending on whether partitioned EDF, RMS, DMS is selected to run on the multi-core system [11]. What this means is that each core has its own ready-queue for storing tasks to schedule, and its own scheduling algorithm [11]. In addition, one of the key differences between a partitioned scheduling algorithm and a global scheduling algorithm is that a task that is initially assigned to a core cannot migrate to another core [10]. Therefore, tasks assigned to cores by partitioned EDF, RMS, and DMS will loop back and forth between being in the ready-queue and being executed in the core they were assigned in until they are executed. We will now go into the details of each these partitioned scheduing algorithms.

II. Partitioned Earliest Deadline First (P-EDF)

A. Overview

Partitioned Earliest Deadline First is a scheduling algorithm derived from the single-core scheduling algorithm, 'Earliest Deadline First' to be used now on multi-core systems [11]. Earliest Deadline First is a preemptive scheduling algorithm, meaning that, while a task is being executed, it can be switched out and placed back in the ready-queue by the scheduler, depending on the algorithm, even if the task didn't finish executing [11]. This cycle repeats until the task is finally executed. For EDF, upon arrival of any task, if a given task in the ready-queue has a deadline that is earlier than that of the currently executed task, then the task with the earlier deadline will switch places with the originally executing task and have the CPU begin its execution instead. This can be best illustrated with an example.

Suppose we have a set of periodic tasks T that all arrive at $t = 0$ with deadlines equal to next periodic time values, as

$$T = \{t_1, t_2, t_3\}$$

a set of periods P for each task as

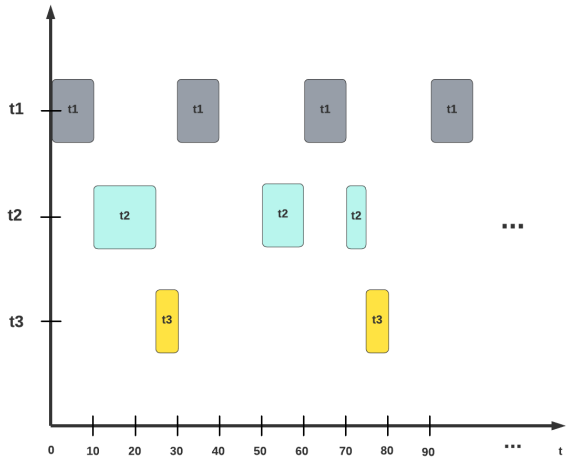
$$P = \{p_1, p_2, p_3\} \quad \Big| \quad p_1 = 30, p_2 = 50, p_3 = 60$$

a set of execution times E for each task as

$$E = \{e_1, e_2, e_3\} \quad \Big| \quad e_1 = 10, e_2 = 15, e_3 = 5$$

Below is a gantt chart of how T can scheduled with respect to P, and E:

Fig. 1. Gantt chart of tasks T scheduled according to Earliest Deadline First (EDF), with periods P and execution times E



B. Task assignment example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

III. Partitioned Rate Monotonic Scheduling (P-RMS)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

A. Overview

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

B. Task assignment example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

IV. Partitioned Deadline Monotonic Scheduling (P-DMS)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

A. Overview

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

B. Task assignment example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

V. Uppaal implementation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

VI. Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla non est porttitor, consequat ex sed, malesuada diam. In gravida laoreet sem rutrum suscipit. Aliquam congue consectetur erat id tristique. Fusce volutpat quis quam eu vulputate. Etiam diam dui, pretium id elit tempor, convallis molestie lacus. Duis interdum posuere faucibus. Duis iaculis hendrerit ipsum, in elementum sapien rhoncus sit amet. Nulla facilisi. Nunc vel tellus et justo tempus posuere et sit amet velit. Nam vestibulum augue erat, et vulputate risus maximus accumsan. Maecenas lectus lorem, pellentesque ut tempus condimentum, tristique vitae leo. Suspendisse sed nibh quam. Integer quam turpis, porta vel quam sit amet, dictum accumsan mi. Maecenas aliquet ligula sit amet leo tristique pulvinar. Nam tincidunt dui ut ligula gravida, mattis interdum dolor porttitor. Donec vel enim vitae urna pretium malesuada ut ac dolor.

References

[1] Edwin Robledo, What Was the First Computer?, 8th November 2022

[2] GeeksForGeeks, Advantages and Disadvantages of Multicore Processors, 15th April 2023

[3] Kumar, N., Vidyarthi, D.P., A novel energy-efficient scheduling model for multi-core systems, Cluster Comput 24, 643–666 (2021).

[4] Zhuo Chen, and Diana Marculescu, Fellow, IEEE, Task Scheduling for Heterogeneous Multicore System

[5] Behnaz Pourmohseni, Fedor Smirnov, Stefan Wildermann, Jürgen Teich, Real-Time Task Migration for Dynamic Resource Management in Many-Core Systems, Workshop on Next Generation Real-Time Embedded Systems, 20th January 2020

[6] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment Scheduling Algorithms for Multiprogramming," J. Assoc. Comput. Mach., vol. 20, no. 1, pp. 46–61, 1973

[7] Xu, J., Shi, H. & Chen, Y. Efficient tasks scheduling in multicore systems integrated with hardware accelerators. J Supercomput 79, 7244–7271 (2023)

[8] M. Lordwin Cecil Prabhaker, R. Saravana Ram Real-Time Task Schedulers for a High-Performance Multi-Core System. Aut. Control Comp. Sci. 54, 291–301 (2020).

[9] Herlihy, M., & Shavit, N. (2012). The Art of Multiprocessor Programming, Revised Reprint. Elsevier

[10] Artem Burmyakov, Borislav Nikolić, An exact comparison of global, partitioned, and semi-partitioned fixed-priority real-time multiprocessor schedulers, 15th October 2021

[11] Baruah, S. Partitioned EDF scheduling: a closer look. Real-Time Syst 49, 715–729 (2013)

[12] Subham Datta, Scheduling: Earliest Deadline First, 18th March 2024