GLOBAL SCHEDULING BASED REAL-TIME MULTICORE SCHEDULING

Pisula Mayan Guruge
Real Time Systems
Summer Semester 2024
Hamm-Lippstadt University of Applied Sciences
deundara-palliya-pisula-mayan.guruge@stud.hshl.de

***Abstract*—Advances in semiconductor technology have resulted in the evolution of processors from single-core to multi-core processors, enhancing their computational capability. Real-time systems require efficient scheduling algorithms to ensure that tasks get executed within the strict deadlines. This seminar paper presents global scheduling-based real-time multi-core scheduling and a performance evaluation of partitioned scheduling. The paper deals with the mathematical background, practical applications, and challenges faced in implementing these algorithms. This paper also discusses the advantages and disadvantages of global scheduling and its applications in many aspects.**

I. Introduction

With the evolution of semiconductor technology, multi-core processors are replacing their single-core predecessors. This need is apparent in the demand for increased performance and efficiency while dealing with complex computational processes. Multicore processors are important components of modern computer systems because of their ability to execute numerous threads simultaneously, particularly in real-time applications where completing tasks within their specified deadlines is crucial. Real-time systems are used in a wide variety of fields such as automotive, aerospace, telecommunications, and industrial control. They can be distinguished by their strict time limits. Because these systems need to complete tasks within specific deadlines, sophisticated scheduling algorithms have been established. These algorithms ensure that activities are completed on schedule, hence retaining the system's reliability and efficiency. This seminar paper provides an in-depth review of global scheduling-based real-time scheduling in multicore systems. It provides information about the development of a global scheduling model, its scientific formalization, and its practical utilization. We compare global scheduling to partitioned scheduling, then analyze their performance, and demonstrate the benefits and limitations of each strategy. The goal is to have a better knowledge of the current state and future prospects for global scheduling. [1] [7]

II. BACKGROUND

Real-time systems can be classified as hard or soft real-time systems based on the influence of their timing deadlines. A hard real-time system is when missing a deadline may result in a catastrophic failure, for example, flight control systems and medical treatment equipment. Soft real-time systems are where occasional delays in meeting a deadline are tolerated, like multimedia streaming and online gaming. Scheduling algorithms are critical for ensuring that real-time systems meet all timing requirements. They control the execution of tasks in a way that provides optimal use of computing resources while making sure all tasks are completed before the deadline. For multi-core processors, scheduling certainly becomes much more complex since multiple cores must be managed properly. There are two primary types of scheduling algorithms employed by real-time systems: partitioned scheduling and global scheduling. Partitioned scheduling assigns tasks to individual cores, while global scheduling enables tasks to migrate between cores for maximum resource utilization, achieving high flexibility in the system. [6]

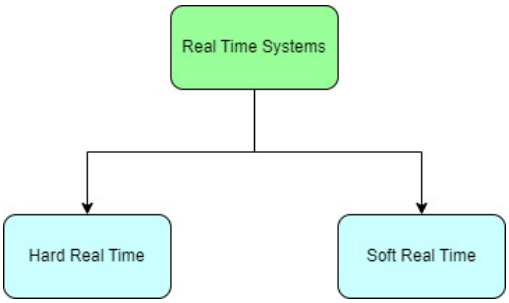


Fig. 1. Real Time Systems

III. Global Scheduling Based Real-Time Multicore Scheduling

The internal structure of multicore processors is an important part of global scheduling-based real-time multicore scheduling. Multicore processors can be classified as one of two types: homogeneous or heterogeneous. Homogeneous multicore processors are when the cores in question are identical, while the cores in heterogeneous multicore processors have different performance characteristics. [5]

A. Introduction to Global Scheduling

A global scheduling strategy would be to put all the tasks in only one queue, from where the scheduler assigns the tasks to the available cores Global scheduling is different from partitioning in allocating the assignment of tasks statically to the specific cores.

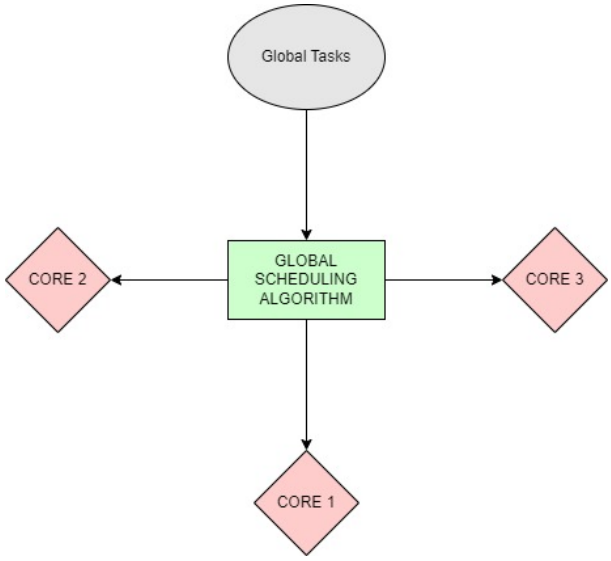


Fig. 2. Global Scheduling

The purpose of using global scheduling is to maximize the system's overall performance by balancing the load on the cores and using task migration to make use of any idle cores. (5) G-FP (Global Fixed Priority) and G-EDF (Global Earliest Deadline First) are global scheduling algorithms. G-FP schedules tasks with respect to their priority, for example, tasks with higher priorities are scheduled first, and thereafter the tasks with lesser priority are assigned sequentially. This is a very simple and predictable strategy, hence ideal for systems with well-defined priority levels. G-FP scheduling works well in cases where tasks have distinct priority levels and where the system load remains fairly constant. In industrial control systems, for example, emergency shutdown procedures can take precedence over ordinary monitoring duties. Ensuring that fixed priorities are set up for such vital tasks ensures due priority to them and hence assures system reliability. While G-EDF priority is dynamic, the changing priority is determined based on its deadlines. So the task with the earliest deadlines gets scheduled first. The dynamic characteristic of G-EDF makes it very suitable for systems with unpredictable workloads. For instance, in telecommunication, packets of data come at variable time intervals; the system has to process them within stringent deadlines to ensure good service quality. G-EDF may adjust the schedule dynamically according to the time of arrival of the data packets and deadline, securing real-time processing with minimal latency. [2]

B. Introduction to Multicore Processing

Multicore processors improve computing performance with the help of parallel processing technologies that enable the execution of multiple tasks concurrently on different cores. It's beneficial in real-time systems since they increase resource utilization and responsiveness through the employment of a large degree of parallelism. One important aspect of global scheduling in multicore processing is task migration. It involves the reallocation of tasks from one core to another for balancing the load and meeting deadlines. Under a multicore processing environment, while task migration can be beneficial to system performance, it increases overhead due to the need for inter-core communication and synchronization [6] [7]

C. Challenges and Considerations

However, implementing global scheduling in multicore systems can present several challenges.

- **Task Migration Overhead:** Several sources contribute to the overhead with respect to task migration, such as time and resources for inter-core communication; this has an impact on the system's performance. Strategies to minimize this overhead are essential for efficient scheduling.
- **Load Balancing:** Even distribution of work over all cores is a significant characteristic of a configuration so that no core has more tasks to perform while the other cores are idle.
- **Scalability:** In such an increasingly complex mixture of task migration and synchronization, the greater the number of cores, the greater the scalability of global scheduling algorithms will be, though possibly limited.
- **Predictability:** The predictability of the behaviour in task executions is one key design objective for real-time systems. The migration of tasks and dynamic changes of priority shall be controlled well enough to not produce unpredictable delays. [3] [6] [8]

D. Performance Evaluation

The performance of global scheduling algorithms is assessed using several measures: schedulability, load balance, and system overhead. Schedulability is the ability of the scheduling algorithms to guarantee the completion of a task within its deadlines, while load balance measures how busy the allocation of the tasks is on the cores. System overhead comprises the additional computational and communication costs enforced by the scheduling algorithm. Much higher schedulability and better load balance have been demonstrated in these studies using global scheduling algorithms such as G-EDF and LLF (Least Laxity First) compared to partitioned scheduling. However, these benefits come at the cost of higher system overhead that is a result of task migration and inter-core communication. On the other hand, partitioned scheduling typically introduces lower overhead and in general lower schedulability and imbalance of load, established in systems with dynamic workloads. Besides, the complexity of a scheduling algorithm and the associated computational overhead may impact the overall system performance. Also, the variabilities of the multicore processors and the task execution times may affect the scheduling algorithm's effectiveness. It is required, therefore, to execute a performance evaluation as comprehensive as it can be while taking all these factors and metrics into account. [2] [5]

IV. Global Multicore Scheduling vs Partitioned-Based Multicore Scheduling

A. Introduction to Partitioned Scheduling

Partitioned scheduling involves the static assignment of tasks to individual cores, with separate execution scenarios being provided for each core. This way, a lot of complexity from the scheduling process is removed since task migration between cores and inter-core communications is eliminated, reducing system overhead. However, partitioned scheduling naturally leads to load imbalance unless workloads are fairly distributed between the cores [8].

B. Comparison Between Global and Partitioned Scheduling

Global and partitioned scheduling have distinct advantages and disadvantages, making them suitable for different types of real-time systems. [6] [8]

- **Schedulability:** Global scheduling algorithms have better schedulability, compared to partitioned scheduling, for example, under G-EDF. Using global scheduling allows all algorithms to take advantage of all available resources fully, considering the task's completion within the deadline. However, challenges can occur in schedulability if Partitioned scheduling is used as it involves the static assignment of tasks to specific cores thereby leading to uneven distribution of tasks among the cores.
- **Load Balance:**Since global scheduling assigns tasks dynamically to all cores based on the current availability, better load balancing is guaranteed. Partition scheduling may sometimes lead to load imbalance if the tasks are spread unevenly from the start.
- **System Overhead:** Global scheduling raises additional overhead because of task migration and inter-core communication. Partitioned scheduling, as explained before, has less overhead because the tasks are statically assigned to the cores and there is no requirement for migration.
- **Complexity:** Global scheduling algorithms are extremely complex to implement as they require a complex system design and careful management of task migration and synchronization. As opposed to global scheduling, partitioned scheduling does not involve task migration and hence is simpler in terms of implementation.
- **Scalability:** As the number of cores increases, the complexity and overhead associated with global scheduling also increase, limiting its scalability. However, partitioned scheduling scales more easily as each core operates independently without the need for migration.

V. Illustration of a Simple Scheduling Scenario and Model Checking

A. Illustration of Global Scheduling Scenario

An example of how the scheduler schedules multiple tasks with varying numbers of execution times within a particular time frame is illustrated in Figure 3.

In this particular scenario, two Cores and three tasks are taken into consideration, we consider both cores to be homogeneous.

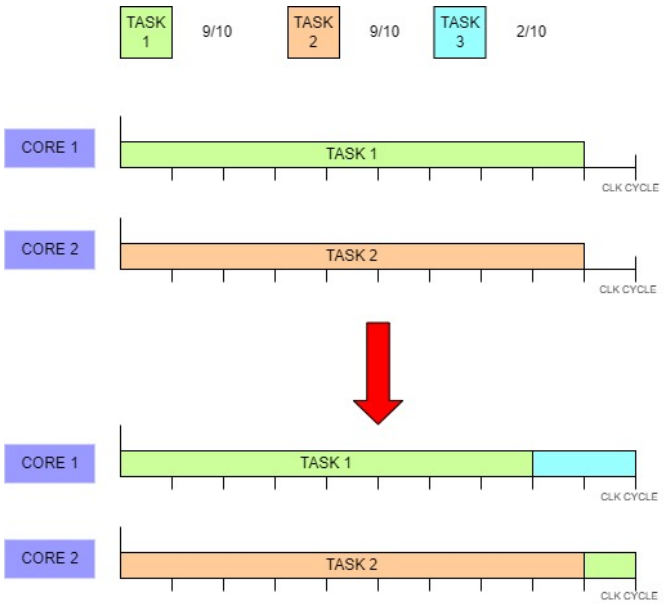


Fig. 3. Illustration of Global Scheduling

Task 1 has an execution time of 9 that must be completed within 10 clock cycles. Similarly, Task 2 has an execution time of 9 that must be completed within 10 clock cycles. But Task 3 has an execution time of 2 that can be completed at any time within the 10 clock cycles.

- First we address the task dependency constraints: Task 3 cannot be split between the two cores because all its operations are sequential. Therefore, this dependency requires that Task 3 be run on a single Core, as splitting it will violate its sequential execution requirement.
- Therefore to utilize all available clock cycles in the cores, we can consider splitting the last process of Task 1(which can be completed within an execution time of 1). Hence, the last operation of Task 1 is split off creating an independent operation that can be scheduled, this strategy helps optimize core utilization.
- Next Task 2 is then assigned to Core 2 which occupies 9/10 clock cycles. To ensure that Core 2 is not left idle, the split, independent operation of Task 1 is then assigned to Core 2 so that all clock cycles in the core are fully optimized.
- Simultaneously the first 8 operations of Task 1 are assigned to Core 1 occupying 8/10 clock cycles. Task 3 is then scheduled for the two remaining clock cycles of Core 1. Since Task 3 requires 2 clock cycles and assigning it to Core 1 completes the 10 clock cycles, Core 1 is also optimally utilized.
- Therefore, in summary, Core 1 runs the first 8 operations of Task 1 followed by Task 3, utilizing all 10 clock cycles efficiently. Core 2 runs Task 2 and the last operation of Task 1, ensuring it also utilizes all 10 clock cycles without any idle core time.

Figure 4 depicts the incorrect task management scenario where

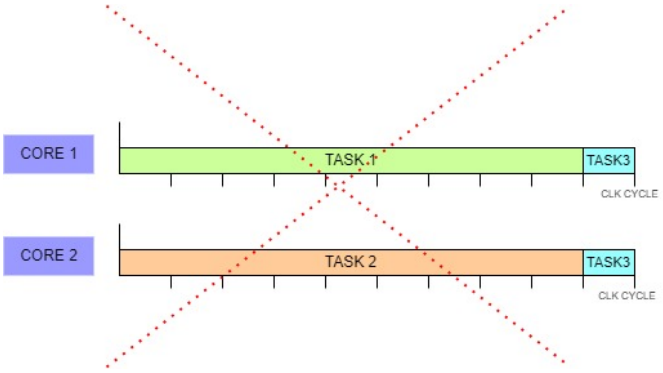


Fig. 4. Illustration of Incorrect Task Scheduling

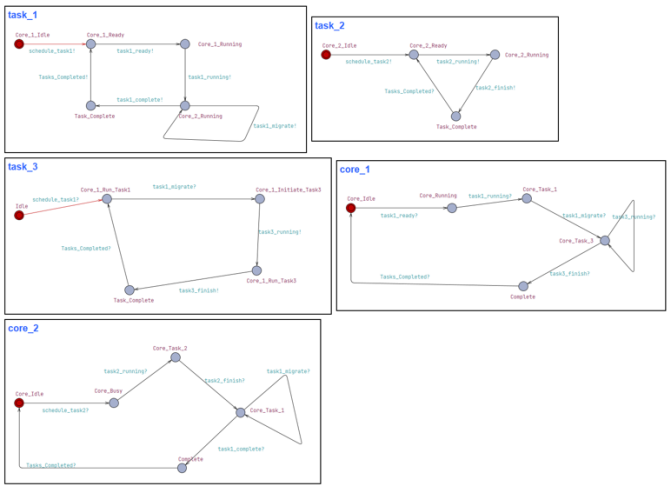


Fig. 5. Uppaal Model of Global Scheduling

task 3 is split and allocated to cores in a way that leads to inefficiencies and potential violations of task dependencies. Task 3 is sequential in execution and divided between Core 1 and Core 2. The allocation violates the task dependency constraint of Task 3, that all its operations need to be executed sequentially without any interruption. Parallelism or distribution of operations belonging to Task 3 between cores could certainly lead to inconsistency and loss of integrity and correctness in its execution.

B. Uppaal Model for Global Scheduling

The Uppaal model created depicts the scenario illustrated in Figure 3. Initially, task1 is assigned to core 1 and task2 is assigned to core 2. In core 1, after 8 clock cycles, the scheduler transfers the last process of task1 to core 2. Task2 continues executing until it reaches and completes its 9-clock cycles. As soon as Task2 finishes executing, task1 is assigned to core 2 and task3 is assigned to core 1. After 10 clock cycles, all tasks have been executed and completed.This ensures that all resources have been fully utilized without their being an idle core.

VI. Advantages and Disadvantages of Global Scheduling

A. Advantages

- **Flexibility:**The use of global scheduling in multi-core systems allows for dynamic task scheduling because it ensures the effective usage of resources, which adapts to the workload.
- **Load Balance:**It provides better load balance because tasks are allowed to migrate from core to core, thereby avoiding the possibility that some of the cores will be overutilized while others remain underutilized.
- **High Schedulability:**Using algorithms like G-EDF or LLF ensures that it achieves high schedulability, and also meets task deadlines when compared to partitioned scheduling. [5] [6]

B. Disadvantages

- **Overhead:**System performance is affected since task migration and inter-core communication produces overhead.
- **Complexity:**The implementation and management of global scheduling algorithms are complex and require intricate system designs.
- **Scalability Issues:**As the number of cores increases, the complexity and overhead corresponding with global scheduling also increase, limiting its scalability. [3] [8]

VII. Applications

Global scheduling-based real-time multicore scheduling is widely used in various applications where timing and performance are critical.

- **Automotive:**In automotive systems, real-time scheduling finds its application in engine control, braking systems, and ADAS (advanced driver assistance systems). The tasks need guaranteed timing for safety and reliability. Global scheduling algorithms can provide resource utilization optimality with stringent timing requirements in complex automotive systems.
- **Aerospace:**Real-time scheduling finds prominent applications in aerospace systems, for example, flight control and navigation systems. Global scheduling will benefit such systems as the load at different cores is balanced dynamically, and the mission-critical deadlines are met with guaranteed performance and reliability.
- **Telecommunications:**Real-time scheduling in telecommunications is very important in signal processing, data transmission, network management, and so forth. Global scheduling algorithms can, therefore, lend efficiency and responsiveness to such systems for handling dynamic workloads and timely delivery of data items.
- **Industrial Control Systems:**Industrial control systems, manufacturing, or automation systems that involve monitoring, control, and data acquisition also utilize real-time scheduling. Global scheduling will optimize resource usage and timely execution of important tasks, hence improving system performance and reliability.

- **Multimedia Systems:**Real-time scheduling in multimedia systems is necessary for audio and video processing to offer smooth, uninterrupted playback. Global scheduling algorithms can dynamically allocate resources, balancing loads across different cores, subject to the timing requirements of multimedia tasks indicated.
- **Healthcare:**Real-time scheduling is applied in healthcare systems in activities such as medical imaging, patient monitoring, and diagnostic systems. Global scheduling techniques can be implemented to enhance performance and reliability so that they can work on time as well as produce accurate results.
- **Finance:**Real-time scheduling in financial systems overlooks activities like transaction processing, risk assessment, and market analysis. On the same lines, resource utilization can be optimized by applying global scheduling algorithms to enhance the efficiency and responsiveness of financial systems. [5]

VIII. Conclusion

Global scheduling-based real-time multicore scheduling offers flexibility, load balance, and schedulability, but also suffers from large overhead, complexity, and low scalability. We can compare global scheduling against partitioned scheduling to better realize their respective strengths and limitations, therefore applying them in the most suitable real-time systems. This seminar paper presents global scheduling-based real-time multicore scheduling, regarding the mathematical foundations, practical applications, and performance evaluation. Further advances in semiconductor technology and rising real-time system complexities will be followed by more sophisticated scheduling algorithms. Based on insights and findings brought out by the present paper, it is possible to further global scheduling-based real-time multicore scheduling for enhancing performance and reliability in various domains of real-time systems by researchers and practitioners.

References

[1] Dongsong Zhang, Fangyuan Chen, Shiyao Jin. "Global EDF-based Online, Energy-efficient Real-time Scheduling in Multi-core Platform." National Laboratory of Parallel and Distributed Processing College of Computer, National University of Defense Technology Changsha, China.

[2] J.-J. Chen. "Partitioned Multiprocessor Fixed-Priority Scheduling of Sporadic Real-Time Tasks." 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), Toulouse, France, 2016, pp. 251-261. doi: 10.1109/ECRTS.2016.26.

[3] Schedulability Analysis for MultiCore Global Scheduling with Model Checking.

[4] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. "Thermal-Aware Global Real-Time Scheduling on Multicore Systems." 2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, USA, 2009, pp. 131-140. doi: 10.1109/RTAS.2009.34.

[5] S. Jadon and R. S. Yadav. "Multicore processor: Internal structure, architecture, issues, challenges, scheduling strategies and performance." 2016 11th International Conference on Industrial and Information Systems (ICIIS), Roorkee, India, 2016, pp. 381-386. doi: 10.1109/ICIINFS.2016.8262970.

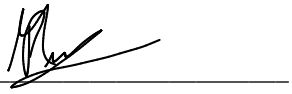
[6] H. Ismail, D. N. A. Jawawi, and M. Adham Isa. "A weakly hard real-time tasks on global scheduling of multiprocessor systems." 2015 9th Malaysian Software Engineering Conference (MySEC), Kuala Lumpur, Malaysia, 2015, pp. 123-128. doi: 10.1109/My-SEC.2015.7475207.

[7] Q. Zhou, G. Li, and J. Li. "Improved Carry-in Workload Estimation for Global Multiprocessor Scheduling." IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 9, pp. 2527-2538, 1 Sept. 2017. doi: 10.1109/TPDS.2017.2679195.

[8] H. Chishiro and N. Yamasaki. "Experimental Evaluation of Global and Partitioned Semi-Fixed-Priority Scheduling Algorithms on Multicore Systems." 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, Shenzhen, China, 2012, pp. 127-134. doi: 10.1109/ISORC.2012.25.

IX. Affidavit - Deundara Palliya Pisula Mayan Guruge

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.



Deundara Palliya Pisula Mayan Guruge
Lippstadt, 12.07.2024