Dynamic Scheduling with real-time requirements

Md Akram
Electronic Engineering
Hochshule Hamm Lippstadt
Hamm, Germany
md.akram@stud.hshl.de

***Abstract*—Now a days many applications require dynamic scheduling with a predictable performance to full fill their requirements [1]. Although there are many scheduling processes available for real time system but among them dynamic scheduling is one of the most effective scheduling processes to satisfied real-time requirements of users. This paper investigates the overall mechanism of dynamic scheduling, algorithms, task modeling with case studies and application of dynamically scheduling in a real-time system. The motive of this paper is to enhance resource allocation and efficiency of CPU to ensure the timely execution of tasks in real-time systems by using dynamic scheduling.**

***Index Terms*—Real-Time, dynamic scheduling, requirements, simulations studies, multiprocessor**

I. Introduction

The purpose of real time computing system is performed certain tasks with exact accuracy. In such a system, "correctness depends not only on the logical result of the computation but also on the time at which the results are produced [2]. So, in real time system to finish tasks within specific frame of time is mandatory otherwise there has a possibility to occurs big problems. At the current time some import real time system is industrial automation, nuclear power plants, embedded system e.g. Such of those system all tasks much be finished by CPU within deadline for accurate results and avoiding system errors. In this case to full fill requirements of deadline CPU use scheduling process more specifically dynamic scheduling which basically use to organize tasks to run accordingly their deadline. In dynamic scheduling, many algorithms such as Earliest deadline first, Dynamic priority scheduling, Least laxity first e.g. applied for organize tasks and that's really effectively help to perform CPU to full fill their real time requirements [2]. The paper explores mechanism of dynamic scheduling, algorithms, task modeling with case studies and application of dynamically scheduling in a real-time system.

II. Dynamic Scheduling

In real-time systems, when tasks have to be finished within set time limits, dynamic scheduling is an essential concept. Dynamic scheduling allows real-time modifications to task priorities and resource allocations depending on the present condition of the system, in contrast to static scheduling, where schedules are fixed and predetermined. This flexibility is necessary in circumstances with dynamic and unpredictable applications [3]. In real-time systems, the main goal of dynamic scheduling is to maximise the efficiency of resources while ensuring that important tasks are completed by the deadline. This requires dealing with a number of difficulties, such as unpredictable task completion times, resource limitations, and reducing system overhead. Dynamic scheduling algorithms enable systems to make decisions in real-time depending on task priorities and system state. Examples of these algorithms are Earliest Deadline First (EDF) and Least Laxity First (LLF) [4].

III. Dynamic Scheduling algorithms

Dynamic Scheduling algorithms are very important for real time system for a better performance. These algorithms enable real-time modifications by continuously evaluating and prioritising jobs according to their deadlines and the status of the system. Least Laxity First (LLF), Rate Monotonic Scheduling (RMS), and Earliest Deadline First (EDF) are three important dynamic scheduling techniques.

A. Earliest Deadline First (EDF)

The EDF dynamic scheduling algorithm is a popular tool for prioritising jobs according to their deadlines. The task with the closest deadline is given the highest priority. This method is quite effective in reducing deadline misses in real-time systems because it guarantees that jobs are finished in the order of their urgency[3].

B. Least Laxity First (LLF)

The Least Laxity First (LLF) algorithm, which is often referred to as LLF, ranks jobs according to their laxity, or the gap between their execution time remaining and their deadline. Priorities are assigned based on how lax a task is. When jobs have different execution timeframes, LLF works especially well since it dynamically modifies priorities to make sure the tasks with the closest deadlines get done first [6].

C. Rate Monotonic Scheduling (RMS)

A common fixed-priority method for periodic jobs is RMS. The priorities of tasks in RMS are determined by their periodicity, the shorter the interval, the higher the importance. Although RMS is not a dynamic scheduling algorithm, it provides a baseline for comparison. RMS works well in fixed-priority scheduling scenarios, but it may also be expanded to dynamic situations with features like dynamic priority assignment [4].

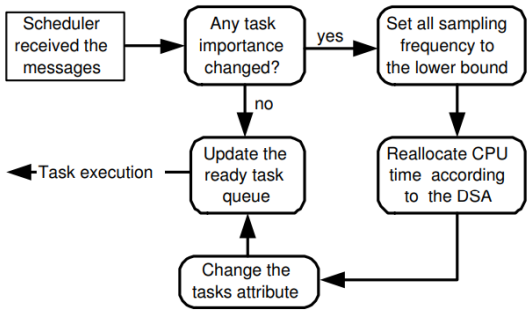

Fig. 1. Dynamical scheduling algorithm [11]

According the deadline of the task's CPU update the queue of tasks by the using of those algorithms. This process ensure that all important tasks meet their deadline and enhance the overall performance of CPU.

IV. Real-Time System and it's requirement

Real-time systems are developed to complete activities within predetermined time ranges, ensuring that crucial tasks are finished by the specified deadline [5]. According the tasks real-time system has many requirements and it need to full-fill for a better output. Real-time systems has several characteristics such as deterministic, reliability, concurrency and resource constraints [6]. Now a days, real-time system are widely used in many modern system for example Aerospace, Automotive, Health Care as well as telecommunication [7].

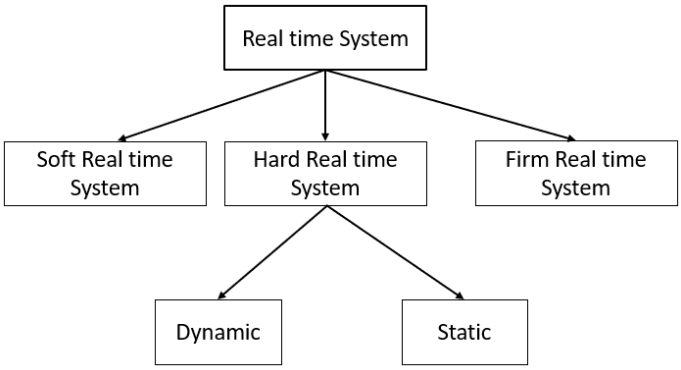

Fig. 2. Real-Time System

In general, there are three different types of real-time systems depending on their timing requirements such as hard, firm, and soft real-time systems.

A. Hard Real-Time Systems

These systems have strict deadline requirements. In a hard real-time system, missing a deadline might have adverse consequences like system failures or safety risks. Aircraft control systems and automobile airbag controllers are two examples [6].

B. Firm Real-Time Systems

While missing deadlines might have a negative impact on service quality, it is often acceptable in these systems. On the other hand, jobs that are not completed by the deadline consider as less important. A financial services transaction processing system is one example [8].

C. Soft Real-Time Systems

These systems have less strict deadline requirements; infrequent missed deadlines are accepted and do not have serious effects. Deadlines missed might cause a gradual decline in performance. Online gaming and multimedia streaming apps are two examples [5].

In real-time systems, scheduling is essential to ensuring that tasks are finished by the deadline. Static and dynamic scheduling are the two main methods of scheduling. Static scheduling follows predicted timetables while dynamic scheduling adjusts in real-time to shifting workloads and system conditions [3]. In situations with unpredictable workloads, dynamic scheduling is especially useful. Frequently Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF) algorithms use for scheduling process. EDF gives priority to jobs with the closest deadlines whereas RMS allocates fixed priorities based on work periodicity [6].

V. Task modeling

Dynamic scheduling in real-time systems requires efficient task modelling and prioritisation. These systems frequently manage a variety of jobs, each with its own special requirements and features, such as periodic, aperiodic and sporadic tasks. Defining priorities correctly optimises system performance and dependability by offering that important tasks are completed by the deadline.

There are three major types of task in real time system. Those are:

A. Periodic

Periodic tasks are those tasks which repeat on a regular basis or the period. It has two major characteristic including regular interval and fixed deadline. One good example of periodic task is sensors data. Suppose temperature sensor

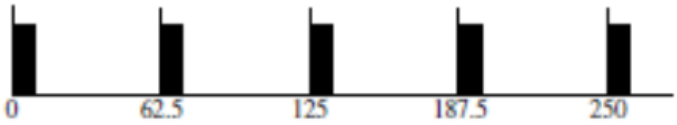

Fig. 3. Periodic Task [12]

sending temperature data to the microcontroller in every one minutes [6].

B. Aperiodic

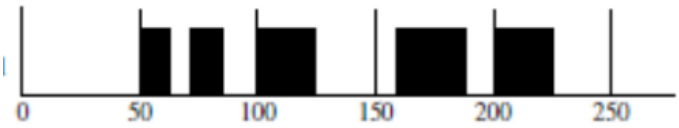

Fig. 4. Aperiodic Task [12]

Aperiodic tasks are those tasks which repeat on a irregular basis or period. It has some characteristic such as irregular interval and flexible deadline. One example of aperiodic task is keypad input. User can give input from keypad at any time [8].

C. Sporadic

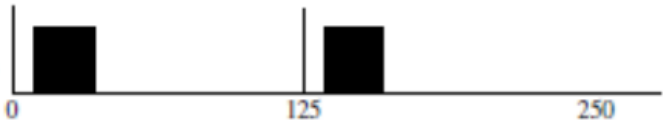

Fig. 5. Sporadic Task [12]

Similar to aperiodic tasks, sporadic tasks have a minimum inter-arrival time to avoid them from happening too frequently. These jobs are often used in some systems such as emergency signals in safety-critical systems, where certain events must be handled quickly but their precise time is unpredictable [7].

Determining task priorities in real-time systems is essential to ensuring that the most important jobs are completed before the deadline. According the types of task several algorithms use to priorities task which I discuss in dynamic scheduling algorithms section. Usually for dynamics scheduling for managing aperiodic task and sporadic task use EDF and LLF where as static scheduling use RMS process. Whenever any new task come in task queue this time scheduler update the queue according their deadline by using those algorithms.

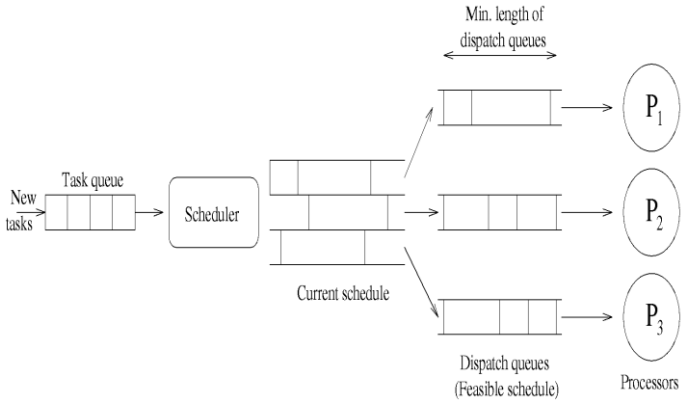

Fig. 6. Task modeling [1]

VI. Relationship between Dynamic scheduling and Real-Time System

Dynamic scheduling is essential in real-time systems for efficient operation. We know that in real time system many operation need to finish within specific time frame that's why scheduling process is a useful tools manage the situation. As I already discuss according the real time tasks such as hard real time system, firm real time system and soft real time system the processor need to be finish those tasks within deadlines [3]. Dynamic scheduling plays an important role for real-time systems when it need to deal with unpredictable workloads and variable tasks. Dynamic scheduling makes sure that significant tasks are completed by the deadline that increase the system performance and stability. Dynamic scheduling algorithms like Earliest Deadline First (EDF) and Least Laxity First (LLF) are frequently used to prioritise jobs with the earliest deadlines [6].Dynamic scheduling is integrated with Real-Time Operating Systems (RTOS) to further improve the system's resource management capabilities[9]. In many different applications, the connection between real-time systems and dynamic scheduling is essential. Dynamic scheduling in automobile systems guarantees prompt reactions to safety-critical features such as anti-lock braking and airbag deployment. It controls network traffic in telecommunications to ensure service quality. It adjusts production schedules in manufacturing to accommodate various workloads [7].

VII. Case Studies and Applications

Dynamic scheduling is essential to many real-world systems because it makes task execution timely and effective. This section analyses case studies from the robotics, aerospace, and automotive sectors including the advantages and disadvantages that exist for these applications.

A. Automotive Industry

Dynamic scheduling is essential in the automotive industry for handling the intricate relationships between different control systems. For example, the Engine Control Unit (ECU)

uses dynamic scheduling to control emissions, ignition timing, and fuel injection. A case study by Di Natale et al. (2007) demonstrated that using Earliest Deadline First (EDF) scheduling greatly increased system responsiveness and decreased deadline misses as compared to static scheduling Although dynamic scheduling Improved responsiveness, reduced deadline misses but it increased computational overhead [10].

B. Aerospace Industry

Dynamic scheduling in the aerospace industry confirms the reliable functioning of flight control systems, which need to deal with a wide range of sensor inputs and control directives in real-time. A case study on the Airbus A380's flight control system highlighted the use of dynamic scheduling methods to set priorities for important operations like collision avoidance and autopilot adjustments. Although dynamic scheduling enhanced ability to handle unpredictable events, improved safety but at the same time it increased complexity in system verification and validation [9].

C. Robotics

Dynamic scheduling is necessary for real-time control and decision-making in robots. The efficiency of Least Laxity First (LLF) scheduling in handling tasks including obstacle recognition, path planning, and motion control was illustrated in a case study involving an autonomous mobile robot. It is true that dynamic scheduling good for adaptability to changing environments, improved real-time decision-making but on the other hand the performance degradation under high task loads [6].

VIII. Conclusion

To sum up, dynamic scheduling is essential to the effectiveness and accuracy of real-time systems. In order to ensure that tasks are performed within the strict deadlines of real-time systems, dynamic scheduling is an essential part of their design and operation. Dynamic scheduling methods such as Rate Monotonic Scheduling (RMS), Least Laxity First (LLF), and Earliest Deadline First (EDF) provide the flexibility and efficiency needed in a variety of applications by adjusting to changing workloads and system variables.

References

[1] G. Manimaran and C. Siva Ram Murthy, A Fault-Tolerant Dynamic Scheduling Algorithm for Multiprocessor Real-Time Systems and Its Analysis, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 9, NO. 11, NOVEMBER 1998.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] Liu, C. L., Layland, J. W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM (JACM), 20(1), 46-61, 1973).

[4] Stankovic, J. A., Spuri, M., Di Natale, M., Buttazzo, G. Implications of Classical Scheduling Results for Real-Time Systems. IEEE Computer, 28(6), 16-25, 1995.

[5] Stankovic, J. A. Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems. Computer, 21(10), 10-19, 1988.

[6] Buttazzo, G. C. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Springer, 2011.

[7] Kopetz, H. Real-Time Systems: Design Principles for Distributed Embedded Applications. Springer, 2011.

[8] Burns, A., Wellings, A. J. Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX. Addison-Wesley, 2009

[9] Rajkumar, R., Lee, C., Lehoczky, J., Sha, L. Practical Solutions for QoS-based Resource Allocation Problems. Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS), 296-306, 1998.

[10] Di Natale, M., Stankovic, J. A., Spuri, M. (2007). Dynamic Scheduling in Real-Time Systems: EDF and Related Algorithms. Proceedings of the IEEE, 95(5), 891-912, 1998.

[11] QIAN LIN, DYNAMIC SCHEDULING OF REAL-TIME CONTROL SYSTEMS, MECHANICAL ENGINEERING NATIONAL UNIVERSITY OF SINGAPORE, 2004

[12] K.TARAKESWAR, Priority-Driven Scheduling of Periodic Tasks, CSE DEPT, GPCET, 2005