Implementation of Task Scheduling Algorithms of Multiprocessor and Mixed Critical Systems

***Suhas G. K.*[1]**, Charan K.V.*[2]**

[1,2]*Associate Professor, Shridevi Institute of Engineering & Technology, Tumakuru-572106*

****Corresponding Author***
E-Mail Id: Suhask300@gmail.com

ABSTRACT

The advent of multi-core architecture rises many challenges, issues and opportunities. Multicores have significantly increased performance of embedded real-time systems and high performance systems. It has also created a greater impact in the way of software development from application software's to operating system kernels. An increasing number of high performance systems are programmed using modern programming languages due to parallel programming revolution created by the multicores. The cost reduction trends in modern embedded systems induces functional consolidation which results in development of mixed critical systems (MCS). Two critical challenges are addressed in this research work are one is overcoming the limitations of modern programming languages to support for developing time sensitive applications in multicore systems. Another one is developing task scheduling strategy in multicore systems to decrease the runtime overhead as well as improving support for mixed critical systems.

***Keywords*: ** Task Scheduling , RTOS, GPOS

INTRODUCTION

The scheduling concept is integral part of embedded real-time system design and analysis. Real-time system quality also depends on the type of scheduling mechanism used in the system design. We identify two types of candidate algorithms for multicore systems one is responsible for ensuring the deterministic real-time response of the time sensitive tasks in mixed critical systems [2], another one helps for developing application with time-sensitive tasks using modern programming languages [5,6].

It is also considered by researchers and system designers from many years for solving open problems in these algorithms. Researchers in the Real-time systems field are contributing from several decades for empowering embedded operating systems using new technologies and adding extensions to the kernel source. The problem of efficient resource scheduling for optimal resource allocation to ensure better quality of service for real-time applications in multiprocessor architectures have been unresolved in the real time system field. Real-time embedded systems as showninFig1.1are key technological components in many application domains like (i) Avionics (ii) Medical embedded devices, (iii) Automotive (iv) Defense and (v) Telecommunication systems etc. When building a real-time system with hardware and software to obtain deterministic response, every part of the system has to ensure predictable timing including the embedded operating system and identifiable segments of it. Evaluating modern programming languages for it's suitability to be used for real-time embedded application development on widely adopted multiprocessor embedded systems, the literature provides many challenges and shortcomings with respect to real-time system development support. The inherent problems drives research towards enhancing real-time computing by using real-time scheduling algorithms.

- In Automotive, the embedded real-time system are used for controlling the breaks and engines of the vehicle for smooth traveling experience.
- In Avionics, the embedded real-time systems are used for the flight navigation and avoiding accidents.
- It is also used in health-care system for monitoring patient condition by regulating heart beats and blood pressure of the patient. They can also be used for entertaining patient with Television, electronic games and music.
- Embedded real-time system can also be found in air-traffic control system (ATC) to provide advisory services to flights for ensuring its safe landing at the destination airport. This is achieved by assigning unique arrival time for each aircraft.
- The real time systems are integral components in industries. Consider an example of industrial robots which are capable of automating assembly and repair operations in a unmanned sector of the factory.
- The military aircraft contain real-time embedded systems for performing many kind of mixed critical functionalities including navigation, communication, display screens, and weapon control system.
- Supervisory Control and Data Acquisition (SCADA) also uses real time systems. In SCADA systems raw data is collected from the sensors that are placed at diff erent geographical points, this data is processed and stored in a separate database. • Nuclear power stations are the unmanned areas where robots used to handle the radioactive other dangerous materials.

RTOS vs General Purpose OS

The real-time operating systems are intentionally used to support operations of the real-time applications by guaranteeing a certain capability within a specified time constraint. There are basically three main diff erences compared to general purpose operating systems (GPOS) Determinism, Task scheduling and Pre-Emptiveness.

Determinism

Goal of the RTOS is supporting real-time applications to obtain deterministic response time. Deterministic timing means we can predict task response time during designing of system. Tasks worst case execution time in RTOS can be pre-defined. Unlike RTOS in GPOS Latency is not a concern but it support for equally sharing the computational power among all task in a system to guarantee the system throughput instead of deterministic response time.

Task Scheduling

In computer operating system, task scheduling is one of the basic functionality in process management. It is a method of assigning some specified work to resources to finish the work. This kind of scheduling activity is carried out by a scheduler. General purpose operating systems are used to run diff erent applications and multiple processes simultaneously in the same platform.

It also ensures all running tasks will receive fair amount of processing time. While executing the tasks sometimes priority of the lower priority tasks will be boosted more than the priority of the several higher priority tasks. The RTOS mainly uses preemptive scheduling based on task priority, which is essential for consistent execution of high priority tasks.

Ensuring time bounded operations all system calls are deterministic.

- All kernel operations can be pre-empted in RTOS.
- Priority inversion problem is prevented in RTOS using certain mechanisms.

•

Pre-Emptiveness
Operating system schedules the higher priority task by preempting the many lower priority tasks, all operations of lower priority tasks are blocked due to preemption. It is used to solve problem of more latency for higher priority task which designer may not want. The worst case latency or time delay in execution is diff erent for every task in a system. Higher priority task latency is much smaller compared to latency of the lower priority tasks. An infinite loop is used for preemptive scheduling of tasks in RTOS. Each task acts as an independent program placed in a loop between ready and finish place. According to certain events or tokens or flags action will takes place but task will not return to the scheduling point unlike function does?

The embedded system means a software is embedded on hardware as one of the important component. They are widely used in small automation to larger avionic embedded systems. The complexity of embedded system software is increasing as the complexity of embedded system hardware is also improving. Earlier 8 bit embedded systems are not using operating systems inside but modern 8 bit devices are using small kernels as shown in Figure 2.

CAN
Controller Area Network
Collision Detection System
FlexRay
Brake-by-Wire System
LIN
Local Interconnect Network
Multifunction Keyless System

***Fig. 1:** Real time embedded systems examples.*

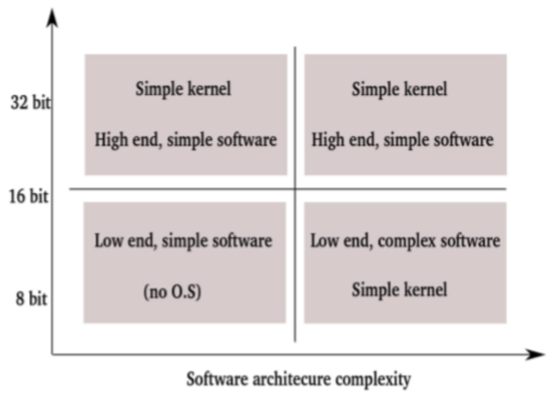
32 bit
16 bit
8 bit
Simple kernel
High end, simple software
Simple kernel
High end, simple software
Low end, simple software
(no O.S)
Low end, complex software
Simple kernel
Software architecure complexity

***Fig. 2:** Spectrum of embedded devices.*

The embedded system means a software is embedded on hardware as one of the important component. They are widely used in small automation to larger avionic embedded systems. The complexity of embedded system software is increasing as the complexity of embedded system hardware is also improving. Earlier 8 bit embedded systems are not using operating systems inside but modern 8 bit devices are using small kernals as shown in Figure 2.

Multicore architecture has become the trend of high performance processors. Multicore or Multi-processors introduces parallelization revolution in software or hardware. Figure 3 shows exponential growth of multi core processors and initial point of its evolution. This trend is continuing, moving from multi cores to many cores but according to Ambdhal's law speed up factor is a limiting element for chip manufactures for increasing number of cores in the processors. According to power usage and problems related to thermal will hiders [7] a further rise in the adoption of single core processors. Multicore architectures have got a momentum due to limitations in using high speed single core processor. This shift from single core to multicore processors [4] is a very big challenge in computing field for embedded real-time systems developers and designers for example consider existing old legacy systems using uniprocessor assumptions for its development.

There are systems developed many years ago and maintained by developers, and they cannot be replaced very easily due to huge investments for developing such systems. Another important aspect while shifting to multicore is task allocation among diff erent cores to improve performance of such multicore platforms. Here partitioning algorithm is presented to efficiently allocating tasks of the systems to diff erent cores. The partitioning technique also used to increase performance of the system by ensuring its correctness.

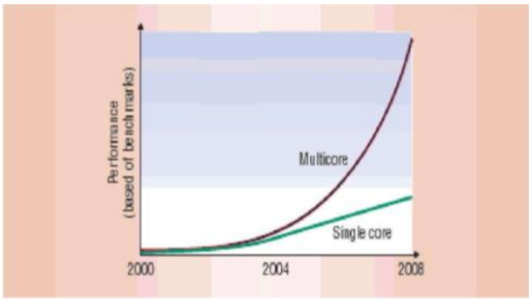

Fig. 3: *Multicore processors tren.*

LITERATURE SURVEY

Owen R. Kelly and Haken Aydlin identified that Multi core processors overcome many drawbacks of uniprocessor system. The emergence of multi core systems also received increased attention for developing multiprocessor scheduling algorithms of both uncritical and multi critical real-time applications. A majority of the research on task scheduling approaches in real-time systems have focused on only periodic tasks having same degree of importance or criticality. Research eff orts for scheduling mixed criticality tasks on Real-time multiprocessor system are relatively new. Experiments are performed to evaluate performance of the diff erent algorithms used for priority assignment and its schedulability tests. Simulator is developed using Java language to generate synthetic kind of mixed-critical task sets along with some of the varied parameters in a larger range for representing a various embedded hard real-time application using this eff orts. The results obtained by this experiments shows that by combing priority assignment algorithm developed by Audsley's with Bertogna *et al.* schedulability test surpass other approaches. According to Lukas Sigrist and Georgia. The mixed criticality scheduling problem and several policies to overcome the problem is rising due to use of multi-cores in embedded system deployments and multicritical applications are co-hosting in the same embedded platform. Scheduling policies to overcome mixed critical scheduling problem employs certain runtime monitoring mechanisms and handles certain exceptional events including task overruns. Special events will be handled in a timely fashion without compromising the safety of the system.

This paper introducing alternative implementations of the runtime mechanism in mixed criticality systems. This mechanisms are empirically evaluated on their runtime overheads. Two such policies are improved is flexible time triggered scheduling and Partitioned EDF-VD. There runtime overhead is measured on 60 core Xeon Phi and 4 core Xeon Phi processor. The usage of multicore systems is significantly increasing for embedded systems this includes safety critical systems like military, Automobiles, medical, avionics. Deploying applications having diff erent critical levels in the same embedded system requires proper isolation among those tasks. This kind of conditions creates mixed critical task scheduling problem. There are certain recently proposed techniques and policies exist they employ mechanisms in runtime to monitor execution of the tasks, identify exceptional conditions or events like overrunning of the tasks, reacting to this events by switching the scheduling modes.

Without aff ecting system's safety implementing this kind of runtime mechanism efficiently is very important for any scheduler to identify runtime events and respond in a timely fashion. This paper find out alternative implementations to these mechanisms and evaluates empirically how much it aff ects the runtime overhead on the scheduling of mixed-critical applications. Two scheduling schemes are implemented one is flexible time-triggered FTTS and another one is PEDF-VD (partitioned EDF-VD). The runtime overhead of both proposed scheduling technique is measured on an Intel @ Xeon Phi having 60 cores and Intel @ core i5 on 4 cores. Extensively executing user generated task sets on avionic applications of avionic industry, the observation is the runtime overheads also cannot be neglected on massively increased cores with multi core architectures due to which they loss schedulability upto 97%. Runtime mechanisms evaluation is the early phase of the design and incorporating it's overheads in schedulability analysis is becoming necessary step in the mixed-critical systems design. The need for small overheads bounded within a rang motivates the development of new improved architecture that are timing predictable and it's runtime environments are targeting for mixed critical applications. Response time analysis of the mixed critical system is carried out by S.K. Baruah, A. Burns and R.I. Davis in [7].

Certification requirement is a major challenge for many of the safety critical embedded systems using tasks having multiple critical levels. In many cases only few of the system functionality are safety critical and hence require certification. The rest of the functions are they do not require any certification or they are very less critical in nature. The runtime monitoring is very challenging and scheduling analysis of such systems of such systems is very complex in mixed criticality systems. The paper contains novel implementation technique for fixed priority scheduling of multi criticality systems on uniprocessors. The proposed scheme monitors the jobs during run-time. An optimal scheme for task priority assignment is developed and sufficient amount of response time analysis is also derived. The proposed novel idea is dominate all previously implemented schemes. Benefits of the proposed scheme is also evaluated. There are three kind of priority assignment techniques as shown below

- Partitioning the criticality (PC), it is most commonly used scheme also called as criticality monotonic priority assignment scheme. Here task having higher criticality will be assigned a higher priority.
- Static mixed criticality (SMC), all jobs of a task can execute only up to its bounded execution time but it cannot execute further any more time. Tn such cases they are aborted and scheduled again when it is safe to execute. Here priorities of the jobs cannot change once assigned.
- Adaptive mixed criticality (AMC), the proposed priority assignment technique is used in adaptive mixed criticality. Run time monitoring of the tasks and switching its priority levels enhances its performance compared to static critical systems. unlike SMC here job is not withdrawn from the execution if it exceeds representative execution time instead of that its priority will be enhanced allowing it to finish its execution

According to Ramirez and Arnoldo since from few couple of years Real-time systems have been dependent on multiprocessor architectures. However, the significant improvement in multi core architectures, which is having many units of processing on a unique chip, has raised the interest in solving many problems for developing such systems. As a result of this, task scheduling technique for multi core and many core architectures now a days have obtained a huge amount of attention. In past decade there are many scheduling techniques have been developed. Many cases have received only 50 % of least upper bound in the system utilization, this is very pessimistic. Investigation of schedulability of the real-time tasks on this multi core and many core systems, Theoretical schedulability tests can be used. However, task sets with good system utilization bound, one of the best alternative method is exhaustive simulation of the system. RealtssMP is also one of the simulation tool among few tools to conduct such simulations of scheduling algorithms and also perform schedulability analysis. Simulation tools is also proved to be beneficial in the evolution of the algorithms in terms of its performance and developing new scheduling techniques or algorithms. This kind of tools also used as an educational tools for academic purpose. Alejandro Alonso, Emilio Salazar, and Miguel A. de Miguel identified tool set for developing mixed critical partitioned system in. Virtualized mixed-criticality multicore systems exhibit new challenges for its development and it is becoming active research now a days. There is an added complexity exist: Now it is required to find exactly how to partition the tasks and assign those tasks to partitions. Here many issues are discussed including application criticality level, dependability and security requirement and timing requirement etc. The MultiPARATES tool depends on model driven engineering's (MDE's) that is more appropriate approach in this settings, it supports or helps in bridging the gap between partitioning and design issues. How systems are developed now a days can be changed by MDE.

Generally, modeling technique have showing their benefits when adopted to embedded systems. These kind of advantages can be achieved by reusing with intensively abstracting or generating all boiler plate codes automatically. Integrating so much functionality on a single hardware to exploits its power of processing hardware leads to many advantages and also presents possible problems. Toolset Architecture the major components of the toolset and flow of data is shown in Figure 4 System modeling: It is the main input tool of a system. It is comprised of 3 models for preparing the executing platform, the advantages and certain kind of restrictions to be used during partitioning.

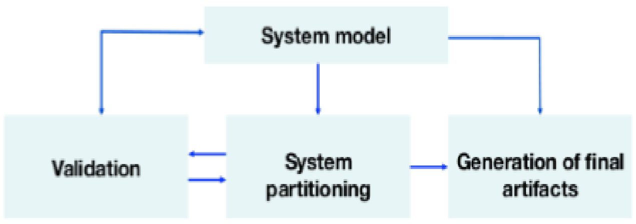

Fig. 4: *Overall tool set architecture.*

Maxime Cherony, Anne-Marie and Pierre-Emmanuel Hladik [3] have enlightened issues about need of simulators for numerous modern task scheduling algorithms. Modernhardwarearchitecturesaredesigned withdiff erentspecificationssuchascache and memory access platforms. We need to design tools for studying those hardware.

Using accurate simulators otherwise even real-time platform having multiprocessors can be used to execute tasks. Here, it requires scheduler to be developed in a C or Assembly language and incorporated it into an operating system. This is lot of time consuming, as well as various realistic tasks need to be generated for evaluation purpose in laboratory. It is also preferred to use Average simulator to accurately simulate the behavior of hardware and software components of the system that influences on its performance.

This kind of simulators are useful in prototyping the system, it does not require real-time implementation of the operating system and its tasks. However, simulator allows for the extensively experimenting with diff erent use cases. Or Diff erent kind of matrix can also be used for the analysis purpose. It is one of the limitations is. It never shows real-time task scheduling behavior in detail on a system, but enough amount of good insights on possible consequences and tendencies can be identified. In two level hierarchical scheduling algorithm (2LHiSA) [8] author has improved theruntimeperformanceandtotalsystemutilizationusingEDFatbothtopleveland local levels. Lower runtime complexity in fixed priority algorithms for job scheduling on multi core processor architecture can be obtained using earliest deadline first scheduling scheme. However, EDF also not an optimal algorithm in multiprocessor architecture. The author introduced a novel migration based scheduling algorithm with certain restrictions for task migration on multiprocessor systems. The proposed system divides the multiprocessor scheduling problem into two levels of schedulers. It work in two phases.

Partitioning a task in which task is statically assigned to a particular processor, it cannot migrate to any other processor during execution. Those task that cannot be assigned to any processor due to bin-packing problem they are qualified for migration to any processor.

Underutilizationofprocessorsisavoidedbyusingadditionalprocessorgrouping phase, sum of the unused processor time from each group of processors is equal to processing ability of one processor. Figure 5 shows local level scheduler with no migrations. Figure 7 shows Global level scheduler with full migration. Figure 8 shows two level scheduling with restricted migration. Figure 6 shows proposed two level hierarchical scheduling it is based on restriction in task migration. The experimental evaluation is provided for theoretical scheduling analysis. THe STORM simulator is used for performing the experiments. Number of task preemptions and migrations have been reduced compared to Pfair and ASEDZL algorithms.

Table 2.4: *Various Multiprocessor task scheduling Algorithms and it's challenge*

Algorithm	**Mechanism used**	**Features**	**Challenges and limitations**
EDF	Uses dynamic Priority Assignment and deadline based scheduling criteria	CPU utilization is full	It is more difficult to implement
RM	Uses static Priority Assignment and period based scheduling criteria	Simple to implement	It does not work for multicore processors
LLF	Uses dynamic Priority Assignment and Laxity based scheduling criteria.	CPU utilization is full and efficient in all conditions	It is more difficult to implement
U-EDF	It uses vertical EDF (G-EDF) and horizontal EDF with slight variation in EDF	It is optimal for task preemptions, migrations have been reduced	It will not work for sporadic tasks because prior knowledge of the task arrival times is required
Pfair	Set of sub tasks have been created using a large task, Within the time interval they must execute all subtasks.	Optimal scheduling of periodic tasks on unicore and multicore processors.	It uses more assumptions, less practical.
HLFEL	First simple algorithm for statically scheduling task graph, Priorities of task is decided by certain static level attributes.	Scheduling technique used for parallel processing.	Communication time is neglected.
ISH	Same as HLFET but it uses idle times created by processors due to partial scheduling.	Scheduling technique used for parallel processing	It will not consider all kinds of static attributes.
MCP	Task priorities are determined based on as late as possible (ALAP'S) attribute.	It gives higher performance to tasks which is having minimum start time.	It will neglect communication cost when computing task priorities.
ETF	Earliest start times of the task is determined and particular task having smaller time to start is selected. It is also uses static level attributes.	Scheduling technique used for parallel processing	It will not minimize the duration of scheduling at each step
DLS	Priorities of tasks are determined based on dynamic level attributes instead of static level attributes.	Scheduling technique used for parallel processing.	Scheduling list is not maintained during scheduling process.
CNPT	Criticality node (CN's) is used as a task priorities, works in two phases Listing and processor assignment.	Performance is greater and complexity is less.	It will not consider other dynamic and static attributes
EDZL	Zero laxity jobs are given the top priority	More number of task sets meet their deadline	It will not guarantee that arbitrary deadline independent sporadic tasks will be scheduled successfully

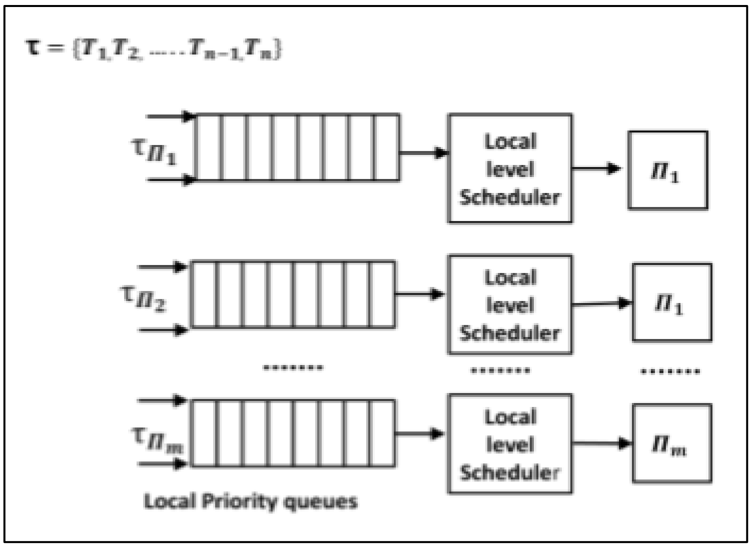

Fig. 5: Task scheduling with no migration.

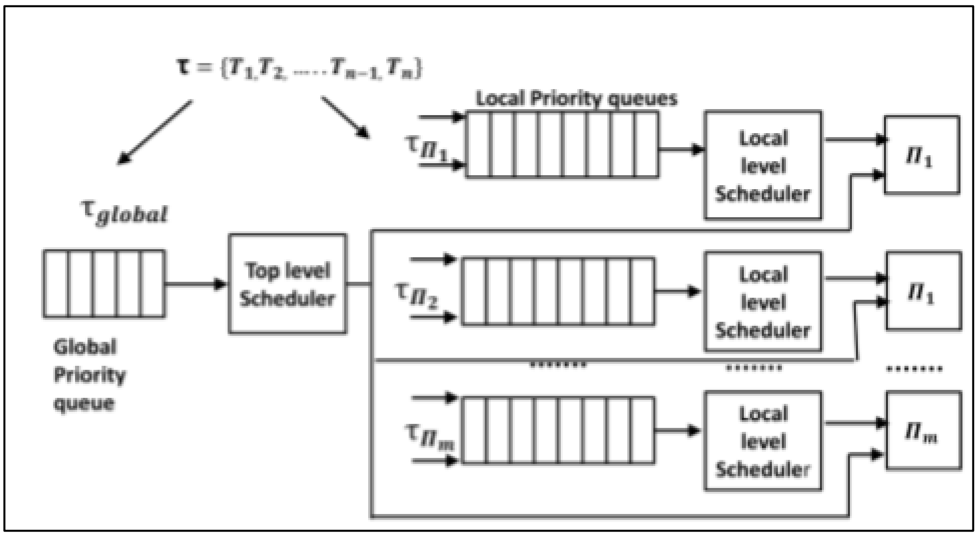

Fig. 5: Proposed 2lHiSA.

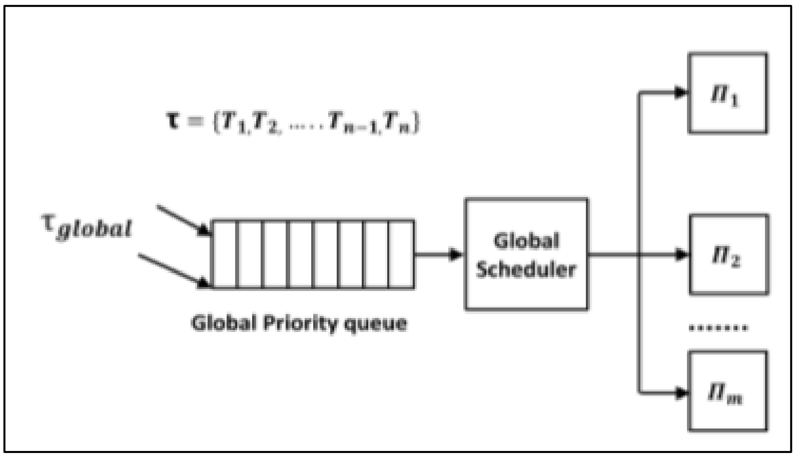

Fig. 7: Task scheduling with full migration.

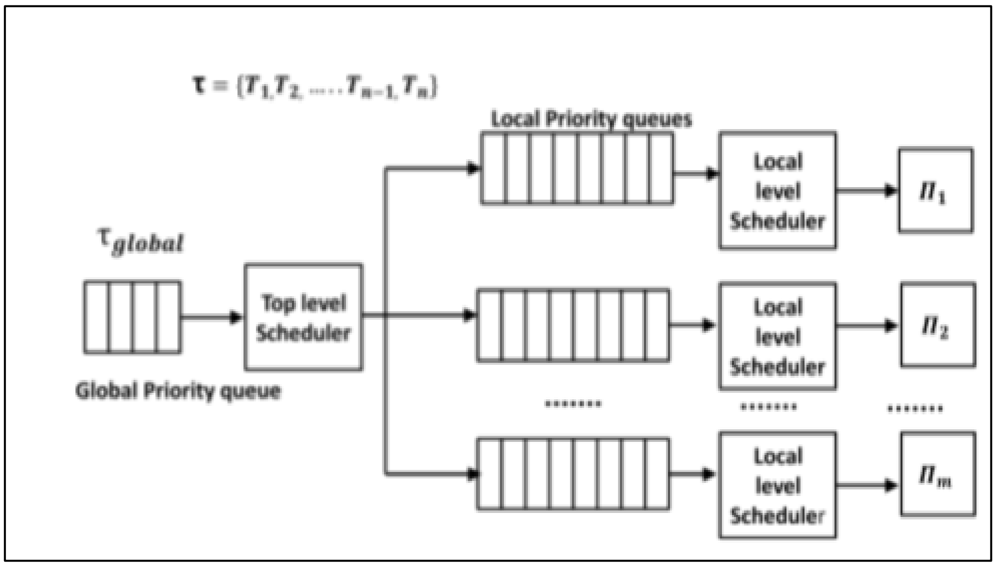

Fig. 8: *Task scheduling with restricted migration.*

CONCLUSION

The advent of multi-core architecture rises many challenges, issues and opportunities. Multicores have significantly increased performance of embedded real-time systems and high performance systems.

It has also created a greater impact in the way of software development from application software's to operating system kernels. An increasing number of high performance systems are programmed using modern programming languages due to parallel programming revolution created by the multicores.

The cost reduction trends in modern embedded systems induces functional consolidation which results in development of mixed critical systems (MCS). Two critical challenges are addressed in this research work are one is overcoming the limitations of modern programming languages to support for developing time sensitive applications in multicore systems.

Another one is developing task scheduling strategy in multicore systems to decrease the runtime overhead as well as improving support for mixed critical systems.

REFERENCES

1. Chen, S., Mulgrew, B., & Grant, P. M. (1993). A clustering technique for digital communications channel equalization using radial basis function networks. *IEEE Transactions on neural networks*, *4*(4), 570-590.
2. Duncombe, J. U. (1959). Infrared navigation—Part I: An assessment of feasibility. *IEEE Trans. Electron Devices*, *11*(1), 34-39.
3. Lin, C. Y., Wu, M., Bloom, J. A., Cox, I. J., Miller, M. L., & Lui, Y. M. (2000, May). Rotation-, scale-, and translation-resilient public watermarking for images. In *Security and watermarking of multimedia contents II* (Vol. 3971, pp. 90-98). SPIE.
4. Suhas, G. K., Devananda, S. N., Jagadeesh, R., Pareek, P. K., & Dixit, S. (2021). Recommendation-Based Interactivity Through Cross Platform Using Big Data. In *Emerging Technologies in Data Mining and Information Security* (pp. 651-659). Springer, Singapore.
5. GK, M. S., Verma, V. K., Devananda, S. N., BR, C. R., Manchale, P., & Pareek, P. K. An Exploration on Recommendation Based Interactivity

through Multiple Platforms in Big Data.

6. GK, D., SN, D., Pareek, P., & MS, N. M. (2021, April). A Altmetrics analysis in social media using Bigdata. In *Proceedings of the International Conference on Innovative Computing & Communication (ICICC)*.
7. NR, D., GK, S., & Kumar Pareek, D. (2022). A Framework for Food recognition and predicting its Nutritional value through Convolution neural network.
8. Hossain, M. D., Kabir, M. A., Anwar, A., & Islam, M. Z. (2021). Detecting autism spectrum disorder using machine learning techniques. *Health Information Science and Systems*, *9*(1), 1-13.