ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 10 N. 2 (2021), 137-156
eISSN: 2255-2863
DOI: https://doi.org/10.14201/ADCAIJ2021102137156

Cluster Based Real Time Scheduling for Distributed System

Girish Talmale[a], Urmila Shrawankar[b]

[a] Research Scholar, Computer Science and Engineering, G H Raisoni College of Engineering, Nagpur

[b] Associate Professor, Computer Science and Engineering, G H Raisoni College of Engineering, Nagpur

girishtalmale@gmail.com, urmila@ieee.org

KEYWORDS

Distributed System; Real Time System; Cluster Scheduling; Global Scheduling; Partition Scheduling

ABSTRACT

Real-time applications nowadays are implemented using a distributed architecture. Real-time task scheduling on such distributed system is a complex problem. The existing real-time task scheduling techniques for the distributed system are primarily based on partitioned and global scheduling. In this paper, we proposed cluster-based real-time tasks scheduling on a distributed system which is a hybrid scheduling approach where processing nodes group into clusters. The real-time tasks are not allowed to migrate among the processing nodes of the different cluster which reduce the migration and preemption overheads problem of global scheduling and improve system utilization which is the problem in partitioned scheduling approach. The performance of the proposed scheduler analyzes with different configurations of the cluster. The analysis of the result shows that task acceptance rate and system utilization increase as the number of clusters tuned to a smaller value. The migration, preemption, and scheduling overheads reduce with increasing the size cluster. The simulation result shows that the proposed scheduler increases the system utilization by approximately 20%. The migration and preemption overheads reduce by approximately 42% and 32 % as compared to benchmark global schedulers.

1. Introduction

Real-time systems comprise systems which correctness depends upon logical as well as temporal correctness. Logical correctness means to produce correct output. Temporal correctness means the

output must produce at the correct time. Let consider an example of an antilock brake real-time system in the car where the brake must be applied at the correct time (Gupta, 2019). Real-time computer system acts as a part of a larger system in most the complex applications such as air traffic control system and such systems are called real-time system (Coulouris et al., 2011). Real-time system changes its state as system physical time e.g. in a chemical plant the chemical reaction continues to affect its state no matter whether the controlling computer system is stopped. Hence it is recommended that the real-time system must decompose into several subsystems called clusters (Erickson and Anderson, 2019). Figure 1 shows a real-time system that consists of a controller cluster which is the collection of objects which operations are controlled by the real-time computer system. Computations cluster which represents the collection of processing units used to perform computations and operator cluster which represents the group of operators interact with control objects through the real-time computer system. A real-time computer system interacts with the control objects through an instrumental interface (Dellabani et al., 2019).

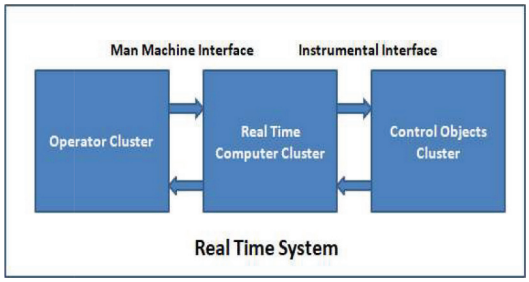

Figure 1: Real Time System Cluster

The distributed system is the system in which the set nodes communicate with each other through a communication network. When the real-time computer system is distributed then all the computational nodes get connected using real time communication networks as shown in Figure 2 (Hammou et al., 2019).

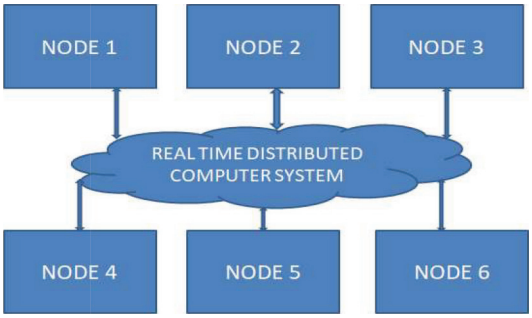

Figure 2: Real Time Distributed System

The reasons why real-time system implemented as the distributed system are as follows (Bastoni et al., 2010):

1. The nature of the application is distributed for example air traffic control system
2. To achieve high performance using efficient scheduling of tasks on distributed nodes

3. Fault-tolerant system as there is no single point failure in the distributed system
4. System architecture where the number of processing nodes communicates with each other
5. Composability: Large and complex systems are composed of several smaller subsystems. In a distributed real-time system the efficient integration of smaller subsystems is possible by the interaction of these distributed nodes.
6. Scalability: The distributed real-time system is scalable to accommodate the need for a large and complex application.

The importance of the problem:

The complex real-time system application like air traffic control systems, autonomous cars, avionics require state of art scheduling techniques to schedule various tasks efficiently and within the deadline for the massive number of distributed nodes to communicate through the network. The size and complexity of these distributed real-time applications increasing day by day and it is extremely difficult to schedule these distributed applications and achieve real-time performance. The run time overhead, scalability, response time, system utilization, changing communication and network architecture, tasks allocation, tasks dependencies, and allocations are few problems that need to addresses for a real-time distributed application. To extremely these problems efficiently, scalable, modular, and flexible real-time distribution scheduling is required.

The existing available solutions:

The recent publication provides the solution to these problems using time-triggered real-time scheduling of tasks on the distributed system (Zhang et al., 2014). The grouped tasks and network-level scheduling for a time-triggered real-time distributed system are developed (Craciunas and Oliver, 2016). The solution proposed by these publications are suitable for limited distributed application but the performance of this scheduler get decrease for large-scale distributed applications. The performance of these schedulers decreases due to high computational requirements, runtime overhead, and scalability. The problem of high computation requirement is overcome with today's modern processing platform but the problem of run time overhead and scalability must be addressed. The different run time overheads include migration overheads produce when tasks migrate from one processing node to another processing node on a distributed system. The task allocation and scheduling problem for multi-rate task sets are developed for multicore parallel systems (Puffitsch et al., 2015). These scheduling techniques addressed only specific task constraints and not suitable for real-time distributed systems.

Limitations in existing solutions:

The existing real-time scheduler for the distributed system, not address problems that occurred in a large-scale distributed system. The different run time overhead like migrations, preemption, and scheduling overheads are not considered. Real-time task allocation is a NP-Hard problem in the distributed application which requires some heuristics solution. The existing real-time scheduling techniques not flexible in terms of system utilization and response time. The existing solutions are not capable to handle large scalable real-time distributed systems.

1.1. Prior Work

Real-time task scheduling on a distributed system is a complex problem. Resource management is difficult as the knowledge about the workload is unknown. The synchronization of clocks among the nodes is a complex problem. The communication problem equally affects the schedulability due to packet loss and late or out-of-order delivery of packets (Atif and Hamidzadeh, 1998). Real-time scheduling on distributed systems is categorized into two main types partitioned and global scheduling (Ismail et al., 2017). Different scheduling algorithms are presented in these categories and all algorithms have some drawbacks. In Partitioned scheduling tasks are partitioned and assigned to dedicated nodes and tasks have to be executed on the same processing nodes. Partitioned scheduler tasks assignment problem is NP-hard (Zhang et al., 2018). Partitioned schedulers do not provide an optimal schedule. Partitioned schedulers do not support the rescheduling of tasks so dynamic task execution is not possible. In Global scheduling, all tasks are scheduled on a common queue and assigned to any free processing nodes, and allowed to migrate among the processing nodes (Zhang and Burns, 2009). The drawbacks of global scheduling are high migration, preemption, and scheduling overheads. To overcome the drawbacks of global and partitioned based scheduling this paper presents the cluster-based real-time scheduling on the distributed system. It is a hybrid scheduling approach in which a set of processing nodes represents a cluster and tasks assigned to the cluster can migrate and execute among the same cluster nodes (Sharma et al., 2020). Parallel and distributed computing is extensively used for applications with high computing and storage demands as mentioned in a recent publication (Blej and Azizi, 2019). Many technologies such as cloud computing, the Internet of Things, fog computing, Big data, machine learning used distributed and parallel computers for processing real-time data (Leng et al., 2020). The real-time distributed system implement using real-time tasks scheduling on the distributed system used partitioned and partitioned scheduling (Hluchy et al., 2001).

Partitioned Scheduler:

In partitioned scheduler tasks assigned to dedicated processing nodes and not allowed to migrate among the nodes of the cluster. Once the task assignment is done then the problem is converted to a uniprocessor (Günzel and Chen, 2020).In partitioned tasks mapped to dedicated processing nodes in many to one relationship. Set of tasks assigned to dedicated nodes as shown in Figure 3 (Gandhi et al., 2020)

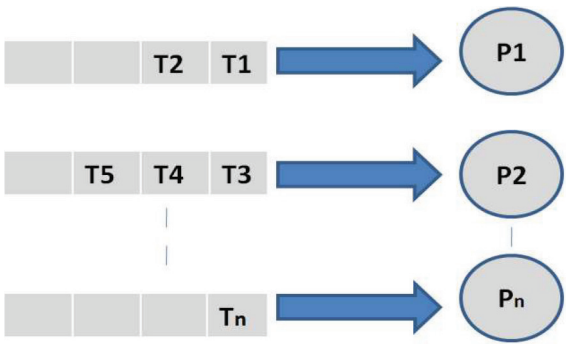

Figure 3: Partitioned Scheduling

Global Scheduling:

In global scheduling, tasks are maintained in the common tasks queue as shown in Figure 4 and assigned to distribute processing nodes and allowed to migrate among the distributed nodes to achieve load balancing (Nemitz et al., 2019).

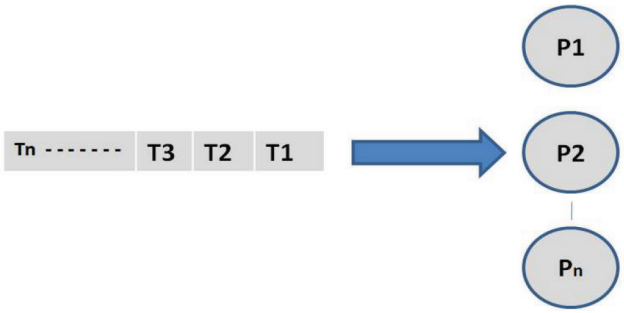

Figure 4: Global Scheduling

The migration of tasks from one processing node to another node is needed when the resource is not available, deadlock situation, lower priority tasks preemption, and efficient load balancing (Qamhieh et al., 2019). Table1 represents the advantages and drawbacks of partitioned and global scheduling in the distributed real-time system (Leoncini et al., 2019).

Cluster Scheduling:

Dallas effect shows that there are some tasks set that cannot be schedulable using partitioned scheduler but schedulable using the global scheduler. Similarly, some tasks set are not schedulable using a global scheduler but schedulable using the partitioned scheduler. So there is a necessity of the hybrid scheduler which gets the benefits of the partitioned and global scheduler and removes the drawbacks of its (Zhao et al., 2019). Cluster scheduling is a hybrid scheduling approach in which distributed processing nodes group together as a cluster as revealed in Figure 5 (Agrawal et al., 2020).

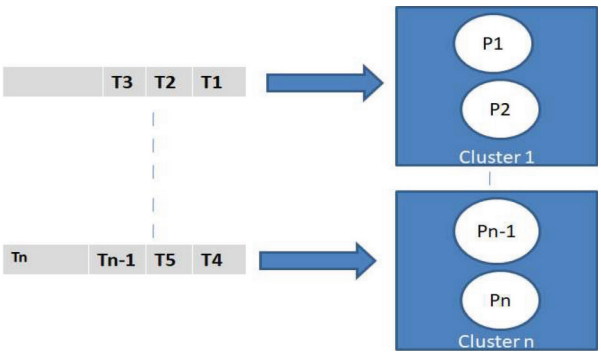

Figure 5: Cluster Scheduling

Real-time tasks allowed migrating and executing on the nodes of the same cluster only (Anderson et al., 2006). Once the task assignment is done using cluster scheduling the global scheduler is used to decide the order of execution of tasks. Different cluster formation heuristics are used to increase the performance of cluster schedulers (Saifullah et al., 2014). The number of clusters in a real-time distributed system must be less to schedule high task utilization efficiently. The size of the cluster must be large to reduce migration and scheduling overhead for low task utilization (Zhuravlev et al., 2011). Homogeneous and heterogeneous cluster formation can be possible (Brandenburg and Anderson, 2013). In a homogeneous cluster, there are identical numbers of processing nodes and in heterogeneous clusters the numbers of processing nodes are different (Bertogna et al., 2009).

1.2 Contributions

The main problem formation of this research manuscript is as follows.

"The cluster-based hybrid scheduler is developed for the real-time distributed system. It groups the distributed processing nodes into cluster and schedule tasks on this cluster to improve system performance and reduce different scheduling overheads"

The main contributions of this research are as follows.

1. The cluster-based hybrid real-time task scheduling technique is developed for the distributed system. First, the distributed processing nodes are first grouped into a set of clusters, and then the tasks assigned to the cluster allowed to execute on the processing nodes of each cluster only to improve performance and reduce scheduling overheads.
2. Analyze the effect of different clusters size on the performance of the distributed system. The number of clusters is equal to the number of processing nodes in the system to act cluster as partitioned based scheduling and the number of clusters identical to one to act cluster as global scheduling. The cluster can act as a hybrid scheduler by tuning the size of the cluster.
3. Evaluate the effect of different cluster configurations on task acceptance rate and system utilization using extensive simulation. The extensive simulation with large tasks and processing nodes is performed. The correlation between the task acceptance rate, system utilization, and cluster size is identified.
4. Compared and analyses the performance of proposed scheduling techniques with benchmarking on different system parameters like migration, preemption overhead, and response time.

1.3. Organization of Paper

The remaining part of the paper is organized as follows:

Section 2 explained the methodology of cluster-based in real-time scheduling on a distributed system with a system model and implementation of cluster scheduler for distributed system tasks scheduling. The experimental setup used for the simulation of the proposed scheduler is also discussed in this section. Sections 3 presents the simulation result of proposed scheduling with different performance parameters. The discussion of impact and analysis of result produce by proposed scheduling is done in Section 4. Section 5 presents the conclusions of the research and future work to be done as an extension of this research.

Table1: Partition Vs Global Scheduling (Ward and Anderson (2012))

Scheduling Techniques	**Advantages**	**Limitations**
	Existing uni-processor real Time Scheduler	Tasks assignment is NP-Hard
	used ones task assignment done	problem
	Used in automobile industry AUTOSTAR	Optimal scheduler not exist
Partitioned	No migration overheads	Not efficiently used the capacity of distributed nodes
Scheduling	Less scheduling overheads	Not efficient for high tasks utilization
	Efficient for low tasks utilization	Rescheduling not possible
	Isolation between distributed nodes	High average response time of tasks
	Support automatic load balancing	Existing uni-processor real time scheduling not used
	Optimal scheduler exist	High migration overheads
Global	Efficiently used the capacity of distributed processing nodes	High scheduling overheads
Scheduling	Tasks average response time is less	High preemption overheads
	Rescheduling is possible for dynamic tasks ex-	Less efficient for low tasks utili-
	ecution	zation
	Tasks assignment is not NP Hard	Synchronization of distributed nodes

2. Material and Methods

The scheduling methods for the distributed real-time system are categorized into time-triggered and event-triggered. In time-triggered real-time scheduling, the tasks are initializing at fixed time units. In distributed real-time systems the clocks at all the processing nodes are synchronized at the same global clocks. The non-preemptive static scheduling technique is used where each task and message activation is done as per the predefined schedule stored in the table. In the event trigger-based scheduling approach, the tasks are activated and schedule on distributed processing nodes when a particular event occurs. The event-triggered-based distributed real-time systems are implemented using the preemptive scheduling technique. The tasks get to activate and service on the occurrence of events. In the static event, scheduling can be fixed priority-based where particular tasks should get activate decide based on the priority assigned to tasks offline. In the dynamic earliest deadline first event scheduling the tasks having an earlier deadline will be selected and activated.

The efficient implementation of distributed real-time applications used both time-triggered and eventtriggered scheduling approaches on distributed processing architecture. The distributed real-time application like autonomous cars where the processing node executes the set of real-time tasks of a particular application. The real-time tasks will execute on the same processing nodes using a partitioned-based scheduling approach which faces several drawbacks. The large-scale distributed applications affect the cost and performance of the system. The partitioned-based solution is not optimal

as the tasks are not allowed to move to the processing node with enough processing power to increase the performance of the system. Some applications are distributed in nature such as brake systems in an automobile is dependent upon several other applications and the implementation of the fault-tolerant system is extremely difficult to implement in partitioned-based scheduling approach in distributed applications. The global scheduling approach allows tasks to migrate from one processing node to another which helps to improve system utilization but at the same time, it will increase the various run-time overheads in the distributed system like migration and preemption overheads.

The proposed cluster-based scheduling is hybrid scheduling approach which divides the processing nodes into the cluster and the real-time tasks allocate to these clusters for execution. The cluster-based scheduling for real-time distributed applications consists of two main steps.

Cluster scheduling on a real-time distributed system is divided into two steps as described in Figure 6.

1. **Tasks Assignment:** Where tasks are assigned to processing nodes of each cluster for execution. The tasks allocation techniques allocate tasks to the cluster as per the processing capacity of each cluster. The task workload is calculated. The sum of tasks workload should be less than or equal to the number of processing nodes in each cluster.
2. **Task Scheduling:** Ones the tasks assigned to the cluster nodes. Tasks scheduler is used to decide the order of execution of tasks on available processing nodes on the same cluster. The scheduler allows tasks to migrate on the other processing nodes of the same cluster which increases the overall system utilization.

2.1 System Model

The real-time tasks Ti (ei, di,pi) represents by parameter ei as worst-case execution period of tasks, di is the deadline of tasks and pi represent the period of tasks Ti (Van den Heuvel et al., 2012).

The real time distributed system consist of set of real time tasks represent as

$$T = \{T1,T2,T3,T\,Tn\} \tag{1}$$

Set of processing nodes is represented as

$$P = \{P1,P2,P3,P4\,Pn\}$$

The real-time task utilization is calculated by a ratio of the worst-case execution time of tasks to the period of the tasks for implicit deadline tasks where the tasks period is equal to the deadline of the tasks i.e for each task Ti as in equation 1(Block et al., 2007).

$$Task\ Utilization(U\text{Ti}) = ei/pi \tag{2}$$

Where *eí is Worst case execution time of tasks Ti pi is the period of tasks Ti*

For each processor tasks assigned must follow the following condition

$$\sum_{i=1}^{n} U_{Ti} \leq UPj \tag{3}$$

Where the *UPj represents the utilization of processor Pj and UPj ≤ 1*

Let the number of cluster in the real-time distributed system is represented as follows. The real-time distributed system with n processing nodes can be grouped into c cluster by using equation 3

$$n/c \tag{4}$$

So cluster set is represented as

$$C = \left\{C1, C2, C3 ---C\left(\frac{m}{n}\right)\right. \tag{5}$$

Cluster scheduling can be work as pure global scheduler when c=n and partitioned scheduler when c=1.

The set of tasks assigned to cluster Cj with x number of processing node is represented by equation 4 as follows

$$\sum_{i=1}^{n} U_{Ti} \leq UPj \tag{6}$$

Where UCj as the utilization of cluster UCj<=x.

2.2. Implementation of Cluster Scheduler for Real Time Distributed System

The cluster scheduling algorithm for a real-time distributed system is first to initiate the task sets with parameter worst-case execution time, deadline, and period of the task. Initialize the number of processing nodes and form the cluster it using different cluster heuristics (Mohaqeqi et al., 2018). The size of the cluster tune to a smaller size to increase the system performance which is suitable for tasks with high utilization. It solves the problem of the partitioned-based scheduling approach. The size of the cluster tune to a higher size to reduce the migration and preemption overheads which is suitable for low utilization tasks.

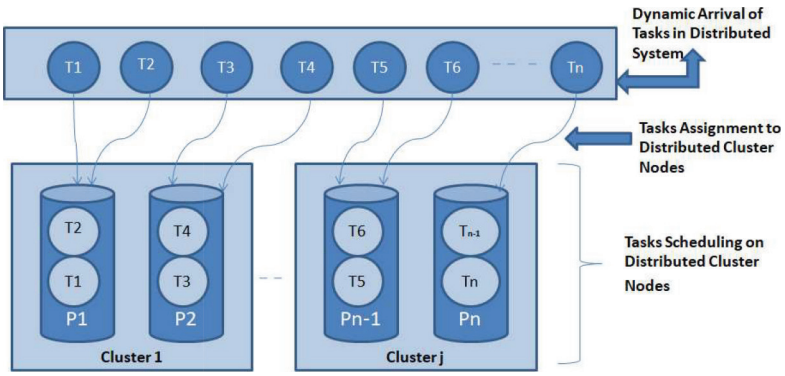

Figure 6: Cluster Scheduling Framework on Distributed Real Time System

Task assignment is the process where the set of generated tasks are assigned to a group of processors called a cluster. Assumptions of cluster scheduling algorithm for the distributed real-time system are as follows:

1. Tasks type is periodic.
2. Implicit tasks deadline where the deadline is equal to the period of tasks

3 Distributed multiprocessor system

4. Tasks deadline is hard.

5. Tasks are independent.

A cluster scheduling algorithm for a distributed real-time system is developed in which the task set is initialized with its worst-case execution time, deadline, and period of tasks. Distributed processing nodes are initialized and cluster tuning constant c is initialized based on different cluster formation heuristics. The number of clusters in a distributed real-time system is calculated as a ratio of the number of distributed nodes and cluster tuning constant c. Each real-time task utilization is calculated as a ratio of the execution period of tasks and their deadline. For each task, if its utilization is less than or equal to one then find the node of lowest utilization of the cluster and assign tasks to this cluster nodes and remove the tasks from task sets. If no processing node in the cluster can schedule the given tasks then select the nodes of the next cluster. If a task is not schedulable in any processing node of all clusters then the tasks will not be assigned to cluster nodes and the schedulability test fails.

Algorithm: Cluster Real Time scheduler on Distributed System

Input: Set of tasks T and processing node P
Output:Task scheduled on cluster
ditributed proceesing nodes

Initialize: Task Set

T = (T1.T2.T3– – –Tn}

Ti(ei,di,pi)

Processor Set P = (P1, P2, P3– – –Pm}

Number of processor ín each cluster as c

$$Cluster\ C = \left\{C1, C2, C3 - - - C\left(\frac{m}{c}\right)\right.$$

for each tasks Ti ϵ T do

Calculate U(Ti) = ei/pi

end for

while Tasks queue get empty

for each task Ti ϵ T do

if U(Ti) ≤ 1

index = 0

for each processor Pi in Cluster Cj where i = 1 to c and j = 1 to c/md

l = PUi

if l > PU(i + 1)

l = PU(i + 1)

index = i

end if

if U(Ti) ≤ PU (índex) then

PU(index) = PU(index) + U(Ti)

T = T – Ti

go to next task in T

else

j=j + 1

end if – else

end for

else

Ti not schedulable

end if – else

end for

end while

end

2.3 Experimental Setup

The performance evaluation of the proposed cluster-based real-time scheduler on a distributed system is done using Simso real-time scheduling simulator (Chéramy et al., 2014). Simso is an open-source pythonbased real-time scheduling simulator for multiprocessor systems extended for distributed multiprocessor systems. Tasks are generated randomly using the RandomFixedSum algorithm and schedule on the cluser-based real-time distributed scheduler. The proposed algorithm is evaluated based on parameters like utilization, migration, preemption, and scheduling overheads. The various system configurations used for experiment are as mentioned in Table 2

Table 2: System Configuration

Sr. No	**Parameters**	**Description**
01		I.Integrated Modular Avionics(IMA) (F-35,F-22,Dassault Ra-
	Workloads	fale)
02		2.NIST Hierarchical Real-time Control System (RCS) for Submarine Automation
03	Simulation Platform	Simso Real Time Simulator
04	Total Tasks	~10,000
05	Total Cores	02– – –512
06		Number of Cluster={01}(Pure Global Scheduler)
	Cluster	Number of Cluster={02,04,08,16,32,64,128,256,512}
		Number of Cluster ={512}(Pure Partitioned Scheduler)
07	Duration	~2000 simulations executed in ~ 2 hours.

The simulation result checks on a various set of tasks and processing nodes. The experiment setup and workload detail are given. The analysis of the proposed algorithm is tested on clusters with varying sizes as follows. The system utilization is varied from 0.60% to 0.95%. For different simulation parameters with a set of tasks set generated with utilization and period of the tasks is randomly generated using Random Fixed Sum which different built-in algorithms are provided in the simulator. RandomFixedSum algorithm is used to assign the period and utilization to different tasks generated randomly. Simso executed more than 1,000 simulations with various configurations.

3. Results

CPU utilization is calculated for the proposed scheduler for a randomly generated task set. Figure 7 shows the CPU utilization analysis of the proposed scheduler on a set of randomly generated tasks and compared with existing global and partitioned-based scheduling algorithms. The tasks scheduled under target utilization varied from 0 to 1 and as follows:

U(0.1,0.15,0.25,0.35,0.45,0.5,0.55,0.65,0.75,0.85,0.95}

The results of tasks utilization and acceptance ratio for different cluster variants. The result shows that for high tasks utilization the cluster with a smaller size gives a high acceptance rate. The Task acceptance ratio of tasks is measured on different task utilization for various sizes of the cluster as shown in Figure 7.

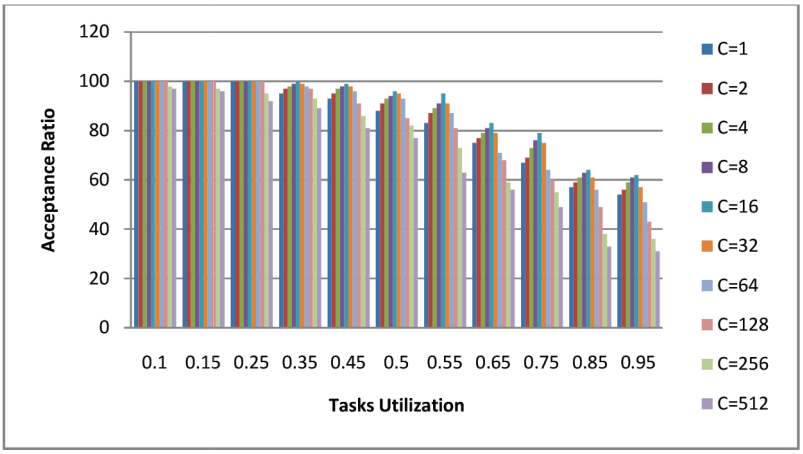

Figure 7: Task Utilization Vs Acceptance Ratio

The result shows the acceptance ratio is 100% utilization 0.35 for cluster scheduling. The task utilization to acceptance ratio analysis shows that as soon as the number of clusters increases the task acceptance rate is decreases. The task acceptance rate is reduced for fewer clusters.

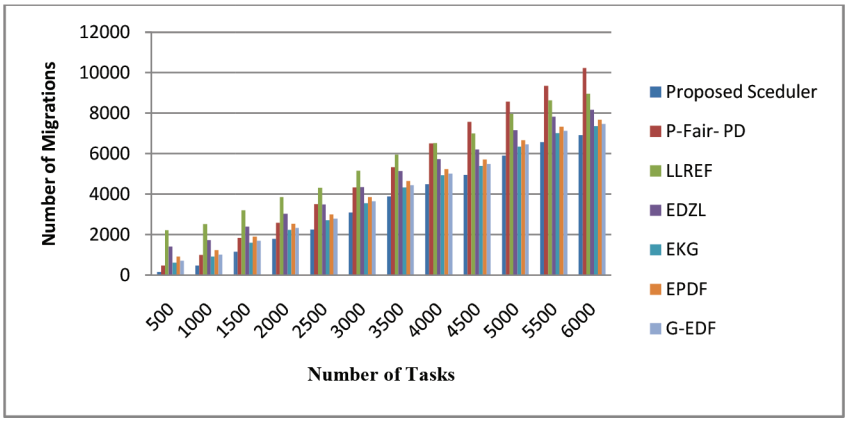

Figure 8: Migration Overhead Analysis

So the cluster scheduler acts as the partitioned scheduler for the cluster with the size equivalent to the number of processing nodes in a distributed real-time system. The key problem in the global scheduler is migration overheads which can be minimized using cluster scheduling for distributed systems by grouping the nodes into clusters and allowing tasks to migrate among the nodes of the same cluster only. The migration overheads analysis is performed with proposed cluster scheduling and a other variant of global scheduling methods. The comparative analysis for migration overhead of proposed scheduler with the existing variant of global schedulers is shown in Figure 8. The comparative analysis of the proposed dynamic cluster-based real-time scheduler for a distributed system with an existing global real-time scheduler is presented. The extensive simulation with task sets varied from 500 to 6000 is performed for this comparison.

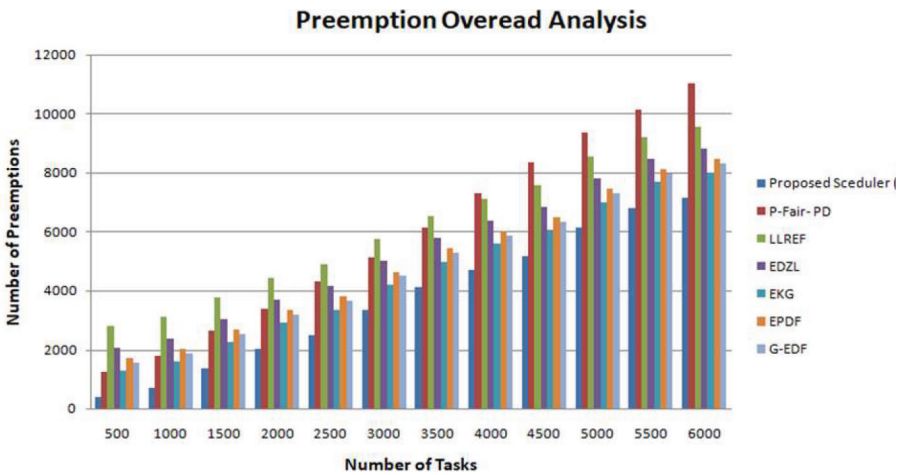

Figure 9: Preemption Overhead Analysis

The proposed real-time scheduler for distributed systems reduces migration overhead counts by 37% as compared to the optimal global scheduling algorithm. The preemption overhead analysis in Figure 9 shows that the number of preemption in the proposed scheduler reduces approximately by 30% as compared to global scheduling. The proposed scheduler results are checked with the sum of migration and preemption with the existing global scheduling algorithm as shown in Figure 10.

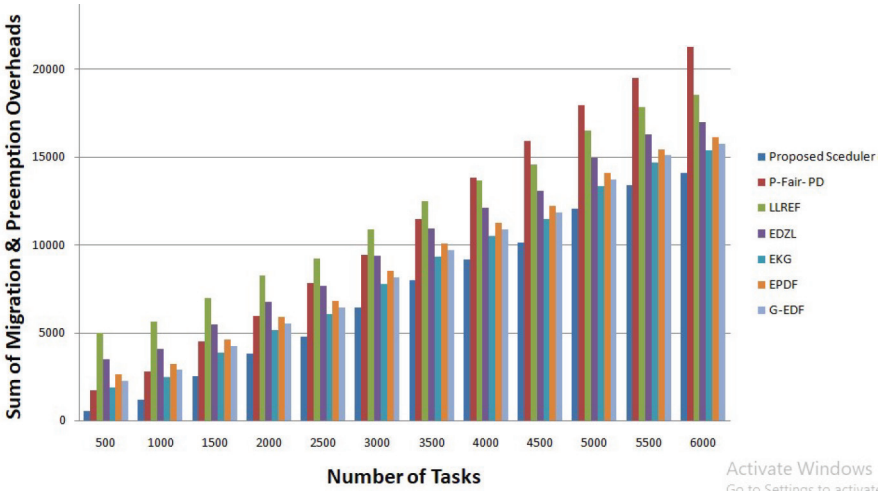

Figure 10: Sum of Migration and Preemption Counts Analysis

The result shows that the proposed scheduler migration and preemption count is reduced by 33% as compared to global scheduling variants. The proposed scheduling is checked with various variants of the cluster and performed the analysis of the result.

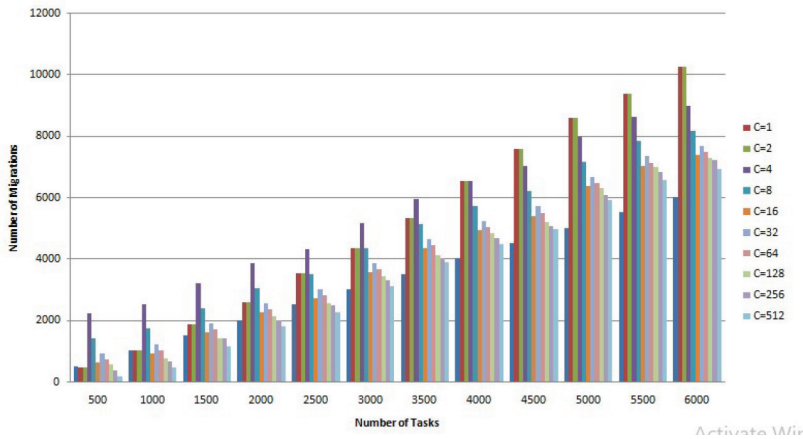

Figure 11: Migration Overhead Analysis with Different Cluster Size

Figure 11 shows the result of the proposed scheduler with different cluster sizes. In the cluster with a smaller size, the migration overhead is increased. Cluster with the higher size the migration overhead gets reduced. For the tasks with less utilization, we have to create a large number of the cluster so that the migration and preemption overhead can reduce to a small number as compared to the global scheduler. Task with higher utilization, the number of clusters in the distributed system should be tuned to a smaller value.

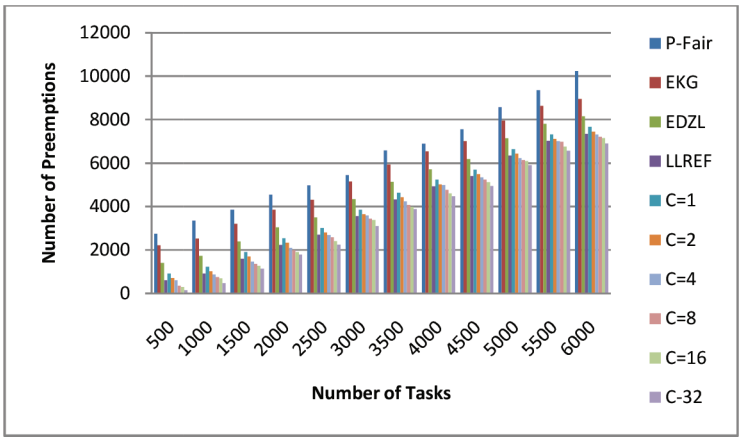

Figure 12: Preemption Overhead Analysis with Different Cluster Size

Figure 12 shows the preemption overhead analysis for the proposed cluster-based scheduler with various sizes of clusters. The preemption overhead is inversely proportional to the size of the cluster. The number of preemptions reduces as the cluster increases. The size of the cluster tune to a large value for tasks with low utilization to reduce the number of migrations.

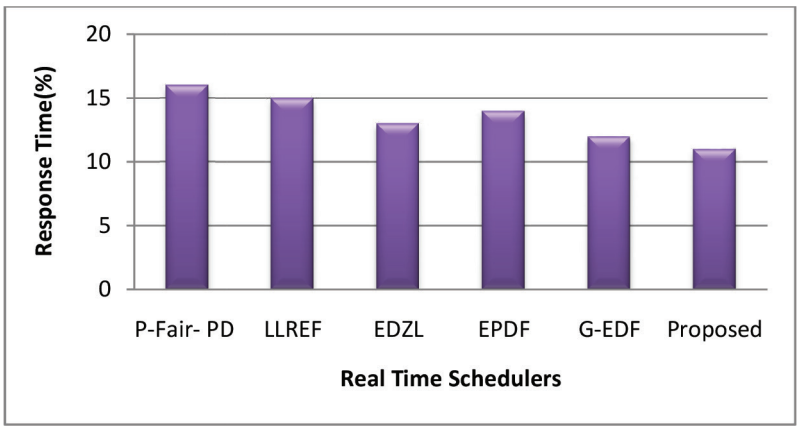

Figure 13: Response Time Analysis

Response time analysis performed with proposed scheduler and existing global real-time scheduling algorithm for multi-core platform. The response time of the proposed scheduler is compared with the existing global real-time scheduling algorithms. The figure presents the performance of the proposed realtime scheduling algorithm reduces the response time by approximately 38 % as compare to P-Fair PD. Figure 13 shows the comparative analysis reduces the response time of the proposed scheduler by approximately 30% as compare to LLREF, 16% as compare to EDZL, 24 % as compare to EPDF and 9% as compare to G-EDF.

4. Discussion

The results indicate that the proposed cluster-based hybrid real-time scheduler provides the flexible and scalable scheduling framework for the distributed system by improving the task acceptance rate and reduce the migration, preemption overheads, and response time. The research study gives the flexible scheduling approach for the distributed system by tuning the size of the cluster. The smaller size cluster performance is equitant to the global scheduling approach and solves the problems of partitioned-based scheduling. Larger cluster size is equivalent to partitioned scheduling and reduces migration and preemption overheads which are the drawback of global scheduling. The size of the cluster adjusts with some moderate value to get the advantages of both the existing scheduling approach and reduce limitations of these scheduling techniques. The analysis of the result shows that the task acceptance rate is inversely proportional to the size of the cluster.

$$Tasks\ Acceptance\ Rate \propto \frac{1}{Size\ of\ Cluster}$$

As soon as we decrease the size of the cluster, the task acceptance rate increases. We can tune the size of the cluster to a smaller size for higher system utilization. The average task response time is also inversely proportional to the size of the cluster**.**

$$Average\ Tasks\ Responce\ Time \propto \frac{1}{Size\ of\ Cluster}$$

The analysis of data collected from the extensive simulation shows that the migration overhead is directly proportional to the size of the cluster. As the size of the cluster that is the number of clusters in the distributed real-time system increases, the average tasks migration overhead decreases.

$$Average\ Tasks\ Migration\ Overheads \propto Size\ of\ Cluster$$

The analysis of simulation results shows that cluster size is directly proportional to average task preemption overheads. The preemption overheads can be decreases by tuning the size of the cluster to a smaller value.

$$Average\ Tasks\ Preemption\ Overheads \propto Size\ of\ Cluster$$

The proposed cluster-based real-time distributed system can purely act as partitioned-based scheduling when the size of the cluster is equal to the number of distributed processing nodes.

Size of Cluster = Number of Processing Nodes in Distributed System (for Partitioned Scheduling)

The proposed scheduler act as purely global scheduling when the size of cluster is set to 1.

Size of Cluster = 1 (for Global Scheduling)

The performance of the proposed cluster-based real-time scheduler checked with various challenges of the distributed software system as follows.

Scalability is one of the key challenges with a distributed real-time system. The existing partitioned- based approach is not suitable for an open system where the tasks dynamically get added to the existing schedule. In cluster-based scheduling support for an open system. The performance of the proposed scheduler does not affect the large distributed system. The extensive simulation performed with a large number of processing nodes varied from 1 to 512. The proposed scheduler performance is checked with large tasks set varied from 500 to 6000. Global scheduling produces extensive migration and preemption overheads for the large distributed system so global scheduling is not suited for scalable distributed systems. The challenge in a distributed real-time system is security. The tasks scheduled on one cluster are separate from another and are not allowed to schedule on another cluster processing nodes. One another issue with the distributed real-time system is the concurrency issue. The proposed scheduler handles concurrency efficiently as the tasks schedule on cluster share resources. The synchronization protocol for each cluster is implemented which is responsible to synchronize the task sets assigned to the cluster. The load balancing must be ensured in a distributed real-time system. The cluster-based approach is efficiently used to balance. The distributed system provides fault tolerance and achieving fault tolerance in the distributed system is the challenge. The proposed cluster-based approach is used to ensure fault tolerance by separating the tasks from cluster to cluster. If one of the processing nodes in the cluster fails then the system performance will degrade but the overall system will not crash like in partitioned-based scheduling.

The rationality of the proposed scheduler is checked with a schedulability test. The schedulability test is used to test whether all the tasks assigned to the schedule are schedulable within the deadline or not. The task utilization calculated for all tasks assigned to each cluster using the formula in equation 2 as follows.

$$\text{Task Utilization for Clsueter Ci} = \sum_{i=1}^{n} \sum_{j=1}^{m} ei/pi$$

The set of processor assigned to cluster are Ci={P1,P2,--------Pn}

If the utilization of the total task assigned to each cluster is less or equal to the number of processing nodes assigned to the cluster then the tasks are schedulable on the cluster processor else the tasks set are not schedulable not the cluster nodes.

If Tasks Utilization for Cluster $Ci \leq n$ then tasks is schedulable else tasks are not schedulable on the cluster. The distributed real-time system in which tasks schedule using cluster scheduling provides high availability and reliability as the cluster-based system does not have a single point of failure like partitioned scheduling. In partitioned scheduling, the task sets are assigned to dedicated processing nodes and if that node fails then the system gets down. The distributed real-time system scheduled using global scheduling is down due to high migration and preemption overheads. The distributed real-time system scheduling cluster scheduling is highly available and reliable as one of the processing nodes fails then also the system is running with degraded performance.

Communication latency is one of the critical parameters in the distributed system. The communication latency in proposed cluster-based scheduling is reducing as the tasks are not allowed to migrate outside the cluster. The communication latency can be further reduced by increasing the size of the cluster.

5. Conclusions and Future Work

The real-time scheduling for large distributed systems is a complex problem. The existing real-time scheduling techniques for distributed systems are not flexible in terms of system utilization, run-time overheads, and response time. In this research, we introduced a cluster-based real-time hybrid tasks scheduler is on a distributed system. The cluster-based tasks assignment technique allocates tasks to processing nodes and the global scheduler in each cluster is used to schedule the order of execution of tasks. The proposed scheduler improves resource utilization by grouping the distributed processing nodes into the cluster. The tasks set assigned to these clusters to improve system utilization and tasks acceptance ratio over partitioned-based scheduling. The proposed scheduler reduces the migration and preemption overhead as compared to the global scheduler as the tasks are allowed to migrate within the processing nodes of the same cluster. The performance of the proposed hybrid scheduler is approximately equivalent to the partitioned scheduler for cluster size equal to the number of processing nodes in the distributed system. When the cluster size is tune to one, then the proposed scheduler act as a global scheduler. The proposed scheduler performance was evaluated using extensive simulation for various sizes of clusters and the result shows that the proposed algorithm increases the percentage of the tasks acceptance ratio by 8% and 20% as compare to global and partitioned scheduling. The migration and preemption overhead is also significantly reduced using cluster schedules as compared to global scheduling. The migration overhead analysis shows that the proposed scheduler reduces the number of migration by 42% as compared to the global scheduling approach. The numbers of preemption reduce by 33% in the cluster scheduler as compared to the global scheduler. The average response time of the proposed scheduler was reduced by 34% as compare to partitioned-based scheduling.

Energy-efficient scheduling in the distributed system performs an important role so the proposed cluster-based real-time scheduling for a distributed system is extended to address energy-efficient distributed scheduling. The proposed scheduler can be extended for the aperiodic tasks model. The proposed scheduler was extended to support resource synchronization on a distributed system. The performance analysis of the proposed cluster-based scheduling approach for the distributed system extended for tasks with mixed-criticality.

6. References

Agrawal, K., Baruah, S., Ekberg, P., & Li, J. (2020). Optimal scheduling of measurement-based parallel real-time tasks. *Real-Time Systems,* 56(3), 247–253. Doi:10.1007/s 11241-020-09346-z

Anderson, H., Calandrino, J. M., & Devi, U. M. C. (2006). Realtime scheduling on multicore platforms. In Proceedings of the of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06) (pp. 179–190).

Atif, Y., & Hamidzadeh, B. (May 26–29 1998). A scalable scheduling algorithm for real-time distributed systems. *Proceedings of the 18th International Conference on Distributed Computing Systems* (pp. 352–359).

Bastoni, A., Brandenburg, B., & Anderson, J. (2010). An empirical comparison of global, partitioned, and clustered multiprocessor EDF schedulers. In 31st IEEE Real-Time Systems Symposium.

Bertogna, M., Fisher, N., & Baruah, S.(2009) Resource-sharing servers for open environments. *IEEE Transactions on Industrial Informatics,* 5(3, August), 202–219. doi:10.1109/TII.2009.2026051

Blej, M. M., & Azizi, M. (2019). Tasks parameter impacts in fuzzy real time scheduling. In. *Studies in Fuzziness and Soft Computing*. Cham, Germany: Springer, 69–78. doi:10.1007/978-3-030-02155-96

Block, H., Leontyev, B. B., Brandenburg, & Anderson, J. H. (Aug. 2007). A flexible real-time locking protocol for multiprocessors. In 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) (pp. 47–56).

Brandenburg, B. B., & Anderson, J. H. (2013). The OMLP family of optimal multiprocessor real-time locking protocols. *Design Automation for Embedded Systems,* 17(2), 277–342. doi: 10.1007/s10617-012-9090-1

Chéramy, M., Hladik, P.-E., & Déplanche, A.-M. (2014). SimSo: A simulation tool to evaluate real-time multiprocessor scheduling algorithms.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed systems: Concepts and design* (5th ed). New York: Addison-Wesley.

Craciunas, S. S., & Oliver, R. S. (2016). Combined task- and network-level scheduling for distributed time-triggered systems. Real-Time Syst 52(2):161–200. https://doi.org/10.1007/s11241-015-9244-x

Dellabani, M., Combaz, J., Bensalem, S., & Bozga, M. (2019). Local planning semantics: A semantics for distributed real-time systems. Leibniz. *Trans. Embed Syst.*, *6*(1):10.

Erickson, J. P., & Anderson, J. H. (2019). *Soft real-time scheduling. Handbook of real-time computing*. Berlin: Springer.

Gandhi, N., Roth, E., Gifford, R., Phan, L. T. X., & Haeberlen, A. (2020). *Bounded-time recovery for distributed real-time systems*. Real, Philippines: IEEE Publications-*Time and Embedded Technology and Applications Symposium (RTAS)*, Sydney, & Australia. (2020), pp. 110–123. doi: 10.1109/RTAS48715.2020.00-13

Günzel, M., & Chen, J. (2020). Correspondence Article: Counterexample for suspension-aware schedulability analysis of EDF scheduling. *Real-Time Systems,* 56(4), 490–493. doi: 10.1007/s11241-020-09353-0

Gupta, K., Arora, V. K., Shukla, H., Beedu, B. K., & Nagpal, A. (2019). *Dynamic scheduling of distri buted storage management tasks using predicted system characteristics*, U.S Patent application 10/168, *953*, filed.

Hammou, B. A., Lahcen, A. A., & Mouline, S. (2019). A distributed group recommendation system based on extreme gradient boosting and big data technologies. *Applied Intelligence*, *59*, 1–22.

Hluchy, L., Dobrucky, M., Viet, T. D., & Astalos, J. (2001). *'The Mapping, Scheduling andLoadBalancing Tools of GRADE', parallel program development for cluster computing, advances in computation: Theory and practice*, *5* (pp. 265–279).: Nova Science Publishers, Inc.

Ismail, D., Mahbubur Rahman, V. P., Modekurthy, & Saifullah, A. (2017). Work-in-Progress: Utilization based schedulability analysis for wireless sensor-actuator networks. In (pp. 137–140). Real, Philippines: IEEE Publications-Time and Embedded Technology and Applications Symposium (RTAS). Institute of Electrical and Electronics Engineers. (2017).

Leng, C., Qiao, Y., Hu, X. S., & Wang, H. (2020). Co-scheduling aperiodic real-time tasks with end-to-end firm and soft deadlines in two-stage systems. *Real-Time Systems*, *56*(4), 391–451. doi:10.1007/s11241-020-09352-1

Leoncini, M., Montangero, M., & Valente, P. (2019). A parallel branch-and-bound algorithm to compute a tighter tardiness bound for preemptive global EDF. *Real-Time Systems*, *55*(2), 349–386. doi:10.1007/s11241-018-9319-6

Mohaqeqi, M., Nasri, M., Xu, Y., Cervin, A., & Arzén, K. (2018). Optimal harmonic period assignment: Complexity results and approximation algorithms. *Real-Time Systems*, *54*(4), 830–860. doi:10.1007/s11241-018-9304-0

Nemitz, C. E., Amert, T., & Anderson, J. H. (2019). Real-time multiprocessor locks with nesting: Optimizing the common case. *Real-Time Systems,* 55(2), 296–348. doi:10.1007/s11241-019-09328-w

Puffitsch, W., Noulard, E., Pagetti, C. (2015). Off-line mapping of multi-rate dependent task sets to manycore platforms. Real-Time Syst 51(5):526–565. https://doi.org/10.1007/s 11241-015-9232-1

Qamhieh, M., George, L., & Midonnet, S. (2019). Stretching algorithm for global scheduling of real-time DAG tasks. *Real-Time Systems,* 55(1), 32–62. doi:10.1007/s11241-018-9311-1

Saifullah, A., Ferry, D., Li, J., Agrawal, K., Lu, C., & Gill, C. D. (2014). Parallel real-time scheduling of DAGs. *IEEE Transactions on Parallel and Distributed Systems*, *25*(12, December), 3242–3252. doi:10.1109/TPDS.2013.2297919

Sharma, R., Nitin, N., Rahman, M. A., & Dahiya, D. (2020) Priority-based joint EDF-RM scheduling algorithm for individual real-time task on distributed systems. *Journal of Supercomputing*, Issue 1 2020.

Van den Heuvel, M. M. H. P., Bril, R. J., & Lukkien, J. J. (May 2012). Transparent synchronization protocols for compositional real-time systems. *IEEE Transactions on Industrial Informatics*, *8*(2), 322–336. doi:10.1109/TII.2011.2172448

Ward, C., & Anderson, J. H. (2012). Supporting nested locking in multiprocessor real-time systems. In *Proceedings of the of the 24th Euromicro conference on Real-Time Systems (ECRTS2012)* (pp. 223–232).

Zhang, F., & Burns, A. (2009). Improvement to quick processor demand analysis for edf-scheduled realtime systems. In *Real-Time Systems*. ECRTS'09. 21st Euromicro Conference on (pp. 76–86). IEEE.

Zhang, L., Goswami, D., Schneider, R., Chakraborty, S. (2014). Task- and network-level schedule co-synthesis of ethernet-based time-triggered systems. In: 19th Asia and South Pacific design automation conference (ASP-DAC), pp 119–124.

Zhang, T., Gong, T., Han, S., Deng, Q., & Hu, X. S. (2018). Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks. *IEEE Transactions on Mobile Computing,* 18(11), 2502–2517. doi:10.1109/TMC.2018.2877681

Zhao, Y., Zeng, H., & H. (2019). The concept of Maximal Unschedulable Deadline Assignment for optimization in fixed-priority scheduled real-time systems. *Real-Time Systems*, *55*(3), 667–707. doi:10.1007/s11241-019-09332-0

Zhuravlev, S., Saez, J. C., & Prieto, M. (2011). Survey of Scheduling Techniques for Addressing Shared Resources in multicore Processors. *ACM Computing Surveys*, *V*(N, September), 1–31.

7. Conflict of Interest

Authors declare no conflicts of interest.