International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Worst-Case Execution Time Analysis of Mixed Critical Applications on Multicore Systems

Preeti Godabole[1], G. P. Bhole[2]

Submitted: 06/06/2022 **Accepted**: 10/09/2022

Abstract: Multicore technology, which has previously been effective in general-purpose computers, is now making inroads into embedded systems. While this improves performance, it also raises the question of how to distribute software activities to the hardware platform's cores, because different allocations have varied added functional features. With the use of multicores, the tasks' execution time varies in an unpredictable fashion. Many scheduling problems in autonomous cars and avionics systems are mixed critical as they comprise tasks at different critical levels. The primary purpose of this research is worst-case execution time analysis and system makespan evaluation of fault-tolerant mixed critical applications. The study considers priority-based task scheduling algorithms for multicore critical systems. The work has considered distinct parameters of evaluation namely deviation in actual execution time, system makespan, and fault- tolerance. The experimentation on a real-time kernel indicates, that GEDF has minimal deviation of only 4.6% in the execution times as compared to PEDF which is 15.8% when active backups of high critical tasks are added to the system. The global approach outperforms the partitioned approach for the considered parameters in a mixed-critical application.

Keywords: *Multicore Critical Systems (MCSs); Real–Time Scheduling; Partitioned Scheduling (PS); Global Scheduling (GS); Multiprocessor (MP).*

1. Introduction and Motivation

Modern multicore (MC) critical systems are getting more performance-intensive as a result of the fact that, on the one hand, they feature more complicated functionality than previously, and, on the other hand, functionality that was traditionally implemented with hardware is progressively being shifted to software. In the design of systems that are set in real-time, integrating components with varied degrees of criticality into a single computing platform is becoming an increasingly relevant topic. Concurrently, these platforms are undergoing a transition of hardware (HW) from single-core to multi-core to many-core architectures in the near future. If a component in a system requires a higher level of assurance against failure, it is characterised as essential. "Mixed-Criticality Systems (MCS)" are structures that have components that fall under two or more unique crucially levels, such as "safety-critical, mission-critical, and non-critical," amongst others. The timing analysis of mixed critical systems is crucial and is essential for the system design.

The resource allocation process denotes the job and collation of the activities and their communications onto the resources of a MC system. This is done with the intention of optimising certain parameters, like "makespan of the system, reliability, memory requirement and energy usage." The makespan can be defined as the overall length of time required to complete the tasks on processors. Let t_f be the final completion time of tasks executing on processor P_i, the makespan of the whole system is then the maximum of schedule length of processors, that is-

$$t_f = \max\{t_f(P_i) | \, i \in [1, 2, \cdots, m]\} \quad (1)$$

Where m is the number of processor cores in the system. Majority of the research in scheduling MC system is done on single processor systems. Few of the researchers have considered multi-processor systems for scheduling MC systems for periodic and sporadic task models with deadline as the constraint. Some of the work is also carried

[1]*Research Scholar,*
Dept. of Computer Engineering and Information Technology,
Veermata Jijabai Technological Institute, Mumbai, India.
pgodabole_p16@ce.vjti.ac.in
[2]*Dept. of Computer Engineering and Information Technology,*
Veermata Jijabai Technological Institute, Mumbai, India.
gpbhole@ce.vjti.ac.in

on achieving fault tolerance in MC system (Hongxia Chai et.al, 2018). This work focuses on combining two parameters, makespan and fault tolerance as in (H. Youness et.al, 2021). The system model and mixed critical task model is expressed in section 2. Section 3 covers the evaluation of the makespan and WCET analysis of the major priority-based algorithms. The conclusion and future work is covered in section 4. The main contributions of the paper are –

- Computation of the worst-case execution time of mixed critical applications on a real-time Linux-based kernel.
- Evaluation of the priority-based both static and dynamic scheduling algorithms are covered based on the makespan of the system.
- Fault tolerance and Makespan used as parameters for evaluation. The transient errors are solved using re-execution of the active backups to make the system fault- tolerant.

2. Literature Review

Deploying applications in real-time on MC is difficult as concurrent operations can intervene with collective resources, confounding worst-case timing analyses (Giannopoulou, et al. 2017). Researchers suggest Isolation Scheduling (IS) to solve this problem. It provides an outline to arrange jobs for multicore applications in real-time. IS enforces mutually exclusive task class execution, preventing inter-class interference. This cited paper proposes and analyses two unique IS approaches: a universal method based on a partitioned approach as well as fluid scheduling and on hierarchical server scheduling. (Sneha Chattopadhyay et.al; 2011) measurement-based analysis to estimate worst-case execution time (WCET). The periodic tasks were emulated to estimate the WCET of the program. The technique is independent of the program semantics. (Saifullah, et al. 2013) points out that MC processors outperform single-core processors. They can enable real-time applications that require a lot of processing power and must adhere to rigorous deadlines that single-core processors can't meet. Old-style MP real-time scheduling uses models of sequential programming and ignores parallelism within tasks. The authors extend their study to a "directed acyclic graph (DAG)" task model with unit implementation requirements. These can be changed into synchronic tasks with the similar decomposition and extension bounds. Synthetic workload recreations show safe and sufficient resource augmentation bounds. (Sanjoy Baruah et.al; 2016) analysed the workload on homogeneous multiprocessor system for a simple mixed critical application. The pre-emptive and non-pre-emptive scheduling algorithms are used to minimize the makespan of the system. (AntoninNovak et.al; 2019) proposed a linear integer programming-based approach to reduce the uncertainty of the processing times in mixed critical applications. The approximation algorithm minimizes the makespan of the system. The work carried out by (Sanjoy Baruah et.al, 2016 and AntoninNovak et.al; 2019) forms the basis of this study to evaluate the global and partitioned scheduling algorithms based on the makespan of the system and the execution time of the tasks. (Hongxia Chai, 2018) explored the mixed critical systems and has identified that fault tolerance is ignored by many of the researchers as a design requirement. So, this study considers transient faults occurring at the end of execution of the tasks while finding the makespan of the system. Cluster-based Scheduling has become increasingly important for using real-time MC systems on multicore processor platforms. In these approaches, the cores are divided into clusters, and each cluster of the global scheduler schedules the partitioned tasks among different clusters (N.KIM et.al, 2016). Cluster-based scheduling can be used when there are a large number of cores. (H. Youness et.al; 2021) proposed a new parameter, "weighted average makespan" to evaluate the mixed critical applications. The optimization of the schedule length and system reliability was done using the newly introduced measuring parameter.

Summary of the Review and Gaps identified

- Worst-case Execution Time analysis becomes unpredictable when multicores are used in MC systems.
- Fault-Tolerance as a design parameter is ignored and needs attention on multicore MC systems
- Makespan of the system is an important parameter that can be combined with fault tolerance to meet the timing constraints of MC system.

3. System Model

This section describes the semantics of the MC system and the problem that the work is trying to solve. Also, highlights the priority based scheduling algorithms used in the study.

1. A collection of mixed critical jobs $J = \{J_1, J_2 \ldots J_n\}$ is characterised by the following parameters {CL, Ci, Ti}, with $CL \in \{LOW, High\}$ representing the critical levels of the jobs, Ci is the worst-case execution time of the task and Ti is the time period.
2. A homogeneous quad-core system with real-time Linux-based kernel is used for recording the actual execution times (ACET) of the task, which eventually is used to calculate the makespan of the tasks and the system makespan as in equation 1.
3. The transient faults are temporarily assumed to be occurring at the end of the execution of the job.

To make the system fault-tolerant many mechanisms exist in the review like the use of specialised hardware, backups, re-execution and rollback ((E. A. Rambo and R. Ernst, 2017); (S. Barauah, 2017)). The strategy of active backups of high critical tasks is used to achieve fault tolerance in the MC system. Two copies of the backups are assumed to be running along with the primary copy at any instance of time. Two copies are maintained as per the triple redundancy model to achieve fault tolerance.

Problem Statement

Given a task set J with n mixed critical jobs along with the active backup of high critical jobs, identify a schedule that minimizes the makespan of the schedule on a homogeneous quad-core system. On an m processor system, the lower bound of the makespan of the system is given by (S. K. Baruah et.al, 2016) where Ci is the actual execution time of the job J_i -

$$Makespan = MAX \frac{\{\sum_{Ji \in J} Ci\ (LOW) \sum Ci\ (High)\}}{m} \quad (2)$$

The upper bound is –

$$Makespan = Max \sum_{Ji \in J} Ci \quad (3)$$

Scheduling Algorithms

The scheduling algorithms used in MC systems are majorly classified into two categories-

Global Scheduling (GS) - Tasks are distributed evenly across all processors in a system using global scheduling. Every task is stored in a single priority queue, from which it may migrate to any of the available processors. One of the most significant benefits of using global scheduling is that it eliminates the challenge presented by the issue of task assignment in PS. This is because all of the tasks are distributed evenly across all of the system's processors.

Partitioned Scheduling (PS) – In the partitioned approach, migration of tasks is not permitted method because each task is allocated to a dedicated processor. Every processor has its scheduler, each with their own distinct task run queue, and there is no room for relocation while the system is active. The primary advantage of utilising partitioned scheduling is that once a task has been assigned to a specific processor, the prevailing algorithms for uniprocessor scheduling can be used to schedule the task onto the processor. The second advantage of using PS is that there is no migration overhead associated with it. This makes queue management much simpler. Under the partitioned scheduling method, the utilisation of the system is low, and whether the overall utilisation of the task set ranges a little higher than fifty percent, then it is possible that the deadline will not be met (Akram, N., Zhang et.al, 2019).

Cluster scheduling (CS) – It is a mixture of PS and GS. Tasks are first partitioned into groups, then allocated to clusters, which are collections of processors, and finally scheduled universally within the cluster.

The scheduling mechanisms are also categorized based on the way priorities are assigned to the tasks. The priorities may be assigned statically which remains fixed as in rate monotonic or fixed priority scheduling. The priorities vary dynamically based on their deadlines as in the earliest deadline first (EDF). In this work, we cover three algorithms Partitioned EDF (PEDF), Global EDF(GEDF), and Partitioned fixed-priority(PFP) to evaluate on the basis of makespan and worst-case execution time.

4. Results and Discussions

The mixed critical applications are scheduled with three different policies PEDF, GEDF and PFP to identify the actual execution time of the jobs. The simulation is carried on a real-time Ubuntu kernel on a quad-core platform with all identical processors.

Experiment 1 – The main objective of this experiment is to identify the deviation in the ACET when active back-ups are to be maintained. The MC application consists of six tasks with three tasks at a high critical level and three tasks at a low critical level are all released at time 0; is simulated to find the actual execution time. First, the simulation is carried out with only primary tasks and ACET is measured as in figure 1a. Two backups of the high critical jobs are maintained to achieve fault tolerance, so in all there are 12 tasks (6 primary and 6 backups of high critical jobs with two backups for each high critical job). Now, the measurement of ACET is shown in Figure 1b.

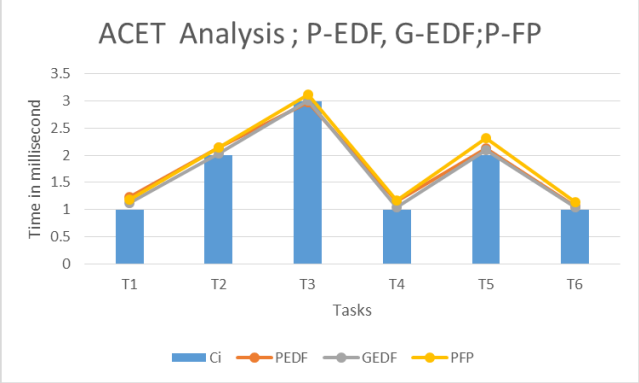

Figure 1a- ACET of tasks without backups

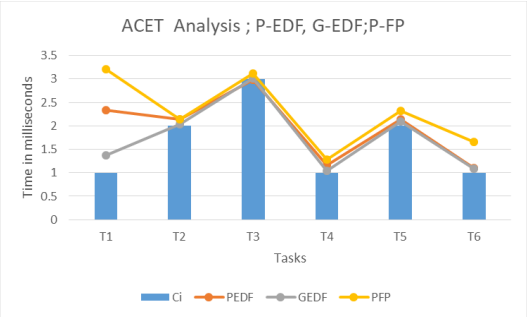

Figure 1b- ACET with active backups of high critical jobs

Though the worst-case execution time is considered while simulating the MC tasks, there is variation in the execution times when active backups of high critical tasks are considered to achieve fault tolerance. The actual execution time of the tasks is same as assumed WCET when the backups of the tasks are not considered as in figure 1a in all three algorithms. But, with the presence of the active backups of the high critical tasks, the actual execution time is changing drastically in PEDF and PFP. The deviation in the ACET of the jobs of the synthetic task set is shown in Table I. Column 3, 5, and 7 in the table gives the worst-case ACET of each of the task including the six active backups for three different algorithms. Column 4, 6 and column 8 gives the deviation of the actual execution time when active backups along with primary tasks are considered w.r.t to only the primary tasks. The average deviation of the tasks ACET when active backups are introduced with PEDF scheduling is around 15.8%, with GEDF is 4.6% and with PFP it is 37.66%. The allocation of tasks to processors in partitioned scheduling is based on the utilization factor of the tasks. Global approach is better as compared to partitioned approaches. If the allocation of the tasks in PEDF is based on other parameters along with utilization factor, the deviations can be reduced. Optimization of task allocation to minimize the deviations in ACET can be the future direction of work.

Table I: Deviation in ACET for different scheduling algorithms for synthetic task set with 6 tasks and around 600 jobs

Task	Ci	ACET-PEDF	Deviation	ACET-GEDF	Deviation	ACET-PFP	Deviation
T1	1	2.33	90.50 %	1.37	22.44 %	3.20	169.84%
T2	2	2.14	0.00 %	2.04	0.00 %	2.14	0.00 %
T3	3	2.98	0.00 %	3.01	0.00 %	3.12	0.00 %
T4	1	1.16	0.00 %	1.05	0.00 %	1.28	9.52 %
T5	2	2.13	0.00 %	2.09	0.00 %	2.31	0.00 %
T6	1	1.10	4.45 %	1.09	5.24 %	1.66	46.60 %

Experiment 2-

The main aim is to evaluate the scheduling algorithm based on the system makespan. The makespan of the job is the overall length of time required to complete the tasks on processors. The makespan of the whole system is the maximum of schedule length of processors. The ACET of each task is used to evaluate the makespan of the system when the schedule is made using both global and partitioned approaches. The system makespan is evaluated based on the upper bound as in equation 2. The aim is to identify the scheduling strategy that minimizes the makespan when the system has both primary and active backups of high critical tasks. The experimentation is carried out on a homogeneous quad-core platform and the results of the makespan for all three algorithms is shown in table II. All the values are in milliseconds and P0-P3 represent processor cores. The allocation of tasks to different processors is done based on the first-fit decreasing utilization algorithm.

	GEDF m=4	PEDF m=4	PFP m=4	GEDF m=2
P0 Core	4.21	8.55	8.74	9.09
P1 Core	6.53	6.62	8.34	7.29
P2 Core	5.22	3.40	5.46	-
P3 Core	1.04	0	0	-
System Makespan	6.53	8.55	8.74	9.09

Table II: System Makespan on a quad core platform

The results indicate the global approach outperforms the partitioned approach when the makespan of the system has to be minimized. This indicates the need to improve the task allocation mechanism in partitioned approaches. Also, the number of processor cores is minimized to check the system makespan in case of a global approach. This acts as a system design parameter. The partitioned approaches are unable to schedule the task set with only two processors as all the tasks are unable to meet the deadlines. Hence, the system makespan is calculated only for GEDF on a two-processor platform. The system makespan where GEDF schedules the task set with two cores is mentioned in the last column of Table II. There is a rise of around 39% in the system makespan when the number of processor cores are reduced to 2.

Conclusion

The study evaluated partitioned and global scheduling algorithms based on some distinctive parameters like deviation of actual execution times and system makespan in a fault-tolerant environment. Partitioned scheduling wastes resource capacity and causes work splitting problems. Global scheduling has a high overhead for task migration and queue management. Though, the global approach has a high overhead it gives a minimal deviation of only 4.6% in the execution times of the synthetic mixed-critical task set when the active backups of the high critical tasks are added to achieve fault tolerance. Also, the system makespan is minimal when global scheduling is used. The experimentation indicates that multi-core real-time scheduling is needed to minimize the system makespan. The PEDF has a higher deviation in worst-case execution times as compared to GEDF when the system has to be fault-tolerant. This may be due to the allocation strategy used in task partitioning. As GEDF has a high overhead cost, this study gives the future direction to optimize the task allocation mechanism in a partitioned approach which aims at minimizing the system makespan, reducing the deviation in the execution times and also make the system fault-tolerant.

References

[1]. H. Youness, A. Omar and M. Moness, "An Optimized Weighted Average Makespan in Fault-Tolerant Heterogeneous MPSoCs," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 8, pp. 1933-1946, 1 Aug. 2021, doi: 10.1109/TPDS.2021.3053150.

[2]. N. Kim, B. C. Ward, M. Chisholm, C. Y. Fu, J. H. Anderson, and F. D. Smith, "Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning," IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 1-12, 2016.

[3]. Sai, M. P. ., V. A. . Rao, K. . Vani, and P. . Poul. "Prediction of Housing Price and Forest Cover Using Mosaics With Uncertain Satellite Imagery". International Journal on Recent and Innovation Trends in Computing and Communication, vol. 10, no. 8, Aug. 2022, pp. 36-46, doi:10.17762/ijritcc.v10i8.5666.

[4]. Robert I. Davis and Alan Burns. 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43, 4, Article 35 (October 2011), 44 pages. https://doi.org/10.1145/1978802.1978814.

[5]. M. J. Traum, J. Fiorentine. (2021). Rapid Evaluation On-Line Assessment of Student Learning Gains for Just-In-Time Course Modification. Journal of Online Engineering Education, 12(1), 06–13. Retrieved from http://onlineengineeringeducation.com/index.php/joee/article/view/45

[6]. Sneha Chattopadhyay, M.J. Tresina, Shankar Narayan, "Worst Case Execution Time Analysis of Automotive Software", Procedia Engineering,Volume 30,2012,Pages 983-988,ISSN 1877-7058,https://doi.org/10.1016/j.proeng.2012.01.954. (https://www.sciencedirect.com/science/article/pii/S1877705812009642)

[7]. S. Baruah, "Schedulability Analysis for a General Model of Mixed-Criticality Recurrent Real-Time Tasks," Proc. - Real-Time Syst. Symp., pp. 25–34, 2017.

[8]. Harsh, S. ., Singh , D., & Pathak , S. (2022). Efficient and Cost-effective Drone – NDVI system for Precision Farming. International Journal of New Practices in Management and Engineering, 10(04), 14–19. https://doi.org/10.17762/ijnpme.v10i04.126

[9]. E. A. Rambo and R. Ernst, "Replica-Aware Co-Scheduling for Mixed-Criticality Systems," pp. 1–20, ECRTS, 2017.

[10]. Antonin Novak, Premysl Sucha, Zdenek Hanzalek, "Scheduling with Uncertain Processing Times in Mixed-Criticality Systems", European Journal of Operational Research (2019), DOI: https://doi.org/10.1016/j.ejor.2019.05.038

[11]. S. K. Baruah et al., "Mixed-Criticality Scheduling to Minimize Makespan," Leibniz International Proceedings in Informatics, LIPIcs, vol. 65, pp. 7.1-7.13, Dagstuhl Research Online Publication Server, Dec 2016. The definitive version is available at https://doi.org/10.4230/LIPIcs.FSTTCS.2016.7

[12]. Gupta, D. J. . (2022). A Study on Various Cloud Computing Technologies, Implementation Process, Categories and Application Use in Organisation. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 8(1), 09–12. https://doi.org/10.17762/ijfrcsce.v8i1.2064

[13]. Akram, N., Zhang, Y., Ali, S., & Amjad, H. M. (2019, January). Efficient task allocation for real-time partitioned scheduling on multi-core systems. In 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST) (pp. 492-499). IEEE.

[14]. Kose, O., & Oktay, T. (2022). Hexarotor Yaw Flight Control with SPSA, PID Algorithm and Morphing. International Journal of Intelligent Systems and Applications in Engineering, 10(2), 216–221. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/1879

[15]. Guoqi Xie, Gang Zeng, Liangjiao Liu, Renfa Li, Keqin Li, "High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems", Journal of Systems Architecture, Volume 70, 2016, Pages 3-14, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2016.04.008.

[16]. Katuk, N., & Chiadighikaobi, I. R. (2022). An Enhanced Block Pre-processing of PRESENT Algorithm for Fingerprint Template Encryption in the Internet of Things Environment. International Journal of Communication Networks and Information Security (IJCNIS), 13(3). https://doi.org/10.17762/ijcnis.v13i3.5101