An Efficient Checkpointing Approach for Fault Tolerance in Time Critical Systems with Energy Minimization

Arvind Kumar
Department of Computer Engineering
FET, Jamia Millia Islamia
New Delhi-110025, India
arvinddagur@gmail.com

Bashir Alam
Department of Computer Engineering
FET, Jamia Millia Islamia
New Delhi-110025, India
babashiralam@gmail.com

***Abstract* – Aim of this paper is to find efficient number of checkpoints for transient fault tolerance in time critical systems. Energy is also the constraint in these battery operated systems due to small and fixed battery. A fault in time critical systems may cause damage if not tolerated in time. Time is an important aspect in these systems to perform an operation. A transient fault can be removed by check pointing policy but the goal is to apply efficient number of checkpoints to enhance battery utilization and schedulability. A system must be fault tolerant to work in fault prone environment. The proposed approach is more suitable to tolerate transient fault and enhancing schedulability with minimum energy consumption. The results and simulation shows that our proposed approach is better with respect to energy consumption and efficient checkpoints.**

Keywords— Checkpoint; Fault Tolerance; Scheduling

I. INTRODUCTION

Time critical systems are most commonly used systems and playing an important role in today's scenario. These systems such as tiny wireless node working in remote areas are highly dependable on their correct functioning with time and energy constraint. The correct functioning means to perform logical as well as physical function efficiently. A time critical systems can be considered efficient if it is performing correctly in the presence of fault. A time critical system can be categorized in hard time critical system and soft time critical system. Hard time critical systems are highly dependent on its deadline. Once the deadline is missing the system is said to be a failure. This failure can be avoided in the system by apply an appropriate approach of fault tolerance. Fault tolerance is the approach to produce desired results in the presence of fault [3][4]. Fault can be categorized on the basis of various reasons such as a fault occurring in a network due to Packet Loss, Packet corruption, destination failure or link failure, a fault occurring due to media crashes known as media fault, a fault occurring due to operating system crashes known as processor fault or a fault occurring due to shortage of resources is known as process fault or a fault occurring due to expire of application while using it is known as service expire fault. A fault can be due to hardware failure or software crashes [7][8]. A hardware type of fault can be tolerated by applying hardware redundancy and software fault can be tolerated by applying software redundancy [10]. Checkpointing and scheduling is well known and commonly used approaches for software fault tolerance [1][2].

In this paper we will investigate transient fault and calculate the efficient number of checkpoint by which we can enhance system utilization and energy consumption. The nature of task can be identified by some inherent error. If there are inherent errors in the system then the fault is transient and can be tolerated by checkpointing.
The figure1 shows the occurrence of transient fault with checkpoints applied for tolerating that fault.

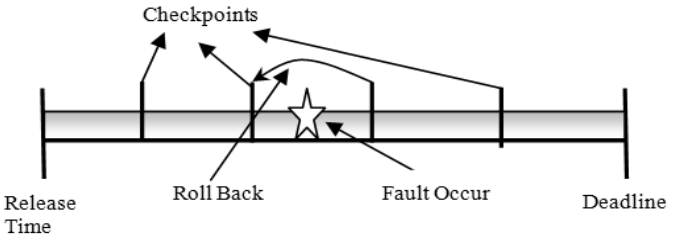

Figure 1: Single transient fault with checkpoints

A transient fault can be tolerated by rollback the system to previous safe state where a scheduler has been checked the correctness. The transient types of faults are very common in time critical system s. The system starts re-execution of task from beginning if no checkpoint is applied. Checkpoint is the approach to start the execution from previous checkpoint where the correctness of results has been checked. The scheduler checks the correctness of functioning on each checkpoint and decides to rollback or not[6].

In this paper we will find the efficient number of checkpoints to tolerate single transient fault. The checkpoints are efficient, if the total execution time (worst case execution time with re-execution time) to finish the task with fault tolerance is minimum.

ISBN:978-1-4799-8890-7/15/$31.00 ©2015 IEEE

Rest of the paper is organized as follows: Related work is presented in section II in which a brief review of real time scheduling algorithms and fault tolerance techniques are discussed. Some issues related to fault tolerance in time critical system are also discussed in this section. In section III, task model and assumptions are presented. The algorithm for fault tolerance is presented and explained in section IV. Section V shows the results of our proposed algorithm and finally last section concludes the paper.

II. Related Work

Time critical systems are more frequently used in today's scenario. Most commonly used real time applications such as aircraft control system are highly dependable of time as well as correct functioning. Many researchers have worked on correct functioning and time constraint [1][2]. The correct functioning in the time critical systems can't be achieved in the presence of fault. To work a system properly in the presence of fault, fault tolerance approach has been proposed. Fault tolerance in these systems can be achieved by checkpointing if the fault is transient [3][4]. Using checkpointing policy with minimized energy consumption & calculated sufficient or an optimal number of checkpoints must be uniformly distributed before task's deadline to adjust the voltage levels & to find a feasible schedule in the system [11]. In [5][11] offline scheduling algorithm is mentioned that combines checkpointing with DVFS policy but it could not handle non-preemption in the system. In [6] Scheduling algorithm has been developed for an aperiodic task, with DVFS. Authors present schedulability where transient faults occur in Poisson distribution where fault tolerance is achieved by checkpointing policy [3][9].

Real time systems need same source of energy to do any job [12]. Consider two techniques to minimize energy consumption in the system. First is DVFS & second is DPD. DVFS is dynamic voltage frequency scaling technique. DVFS is assigned a voltage level that is adjusted according to available storage energy in the system. A Real time system has storage energy depending upon the capacity of the system. Sometimes it may be chargeable and sometimes not. It totally depends upon the system needs. DVFS is a policy to manage energy in the system [13][14].

III. Task Model and Assumptions

The model and assumptions made for solving the problem of applying optimal number of checkpoints are presented in this section. Task model describe how the task will be scheduled.

A. Task Model

Consider a time critical system of a set of n aperiodic task $T = \{T_1, T_2, ..., T_n\}$. Each task is modeled as a tuples of three parameters $<a_i; e_i; d_i>$, where a_i is arrival time, e_i is worst case execution time and d_i is respective deadline. $a_i < d_i$ means the arrival time of the task will be greater than its deadline. Each task is running on full speed for execution. The worst case execution time of a task means maximum response time of a task in the lifetime of a system. Figure 2 shows a task model in which a task i arrive at a_i with its execution time e_i and deadline d_i. It starts execution after some interval of time and finishes its execution before deadline.

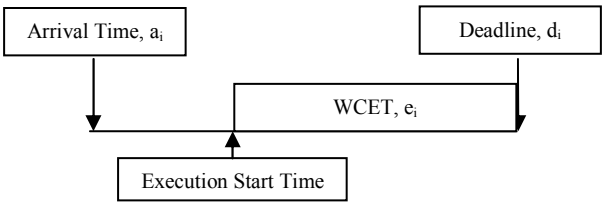

Figure 2: Task model

Above figure shows the arrival time a_i, execution start time, worst case execution time e_i, deadline d_i of a task T_i. The task not necessarily will start the execution from beginning but it will start to execute when the processor is free.

B. Assumptions

For proposing the approach we make the following assumptions related to tasks execution deadline and fault.

1. Single transient fault
2. Set of aperiodic task $T = \{T_1, T_2, ..., T_n\}$
3. Minimum one task arrived at t=0.
4. No fault occurs during rollback to previous checkpoint.
5. Time critical system is hard in nature
6. Task are non-preemptive in nature
7. Dynamic scheduling algorithm (EDF)
8. Every task is independent
9. Task execute on full speed.

IV. Proposed Approach

The objective of our algorithm is to find the maximum number of checkpoint for tolerating single transient fault. For achieving our goal we consider the task set as aperiodic and at least one task arrives at time t=0. Before explaining our approach let us explain the symbols used.

Symbol	Description
T	Task set
T_i	An aperiodic task with arrival time, execution time and deadline
d_i	Deadline of task T_i, with $d_i > a_i$
n_i	Number of checkpoint of current task
m	Number of checkpoint of higher priority task
r	Checkpoint overhead, consider r =1
a_i	Arrival time
e_i	Worst case execution time
F_i	Finish time

A. Scheduling Algorithm

Considering deadline and assumptions in mind, the checkpointing scheme for tolerating single transient fault is proposed in this algorithm. In this approach we consider a set

of task, T; initially we say the number of checkpoint is 1 with overhead also 1. Overhead is the time taken by scheduler for checking correct functioning of task. We also define the initial finishing time F_0 infinity because the finishing time will become double if we find it with n=1. The execution time will reduce with increasing the value of n but it will start increasing with some value of n. when this start to increase at that time we say these are the maximum number of checkpoints which are applicable for that task.

[set of tasks T={T_1, T_2, T_3,…..T_n}, F_n is infinity, voltage level is maximum]

1 While (true) do
2 At t = 0, V = V_{max}
3 d ← min{d_i : T_i ∈ T}
4 Set task T_i having minimum deadline for execution
5 Find $F_n = e_i + n_i.r + \left\lceil \frac{e_i}{n_i} \right\rceil + \sum_{jT_{hi}} \left[e_j + m_j.r + \left\lceil \frac{e_j}{m_j} \right\rceil \right]$
6 If ($F_n \leq d_i$)
Then if ($F_n \leq F_{n-1}$)
Do n=n+1
Go to step 5
Else
Print n-1, number of checkpoints
7 Reduce voltage level upto minimum voltage
Go to step 5
8 Find finish time Fi of scheduled task Ti
9 At t=Fi, Apply step 3to 6 for each task
10 Exit

The algorithm finds the maximum number of checkpoints for tolerating single transient fault. At line 3, we find the minimum deadline of task available at t=0. At line 4 and 5, we calculate the worst case execution time with maximum number of checkpoints applicable to that task. In line 6, we check the deadline and the pervious execution time. If the deadline is still large and previous WCET (Worst Case Execution Time) is larger than increase the number of task and again calculates the same. The task will be schedulable if it is finishing its execution on or before its deadline. If the deadline of the task is small then we check the schedulability of task with checkpoints. The task can be schedulable without fault tolerance approach. At that time number of checkpoints will be zero, means no fault tolerance approach is applied here.

V. RESULTS

To check the results and simulation of our work we consider a set of five tasks T = {<0, 2, 10>, <1, 3, 20>, <5, 7, 25>, <6, 9, 40>, <10,3,50>} have their arrival time, execution time and deadline respectively. We start the task scheduling on maximum voltage considering no fault is in the system. The worst case execution time is calculated for each task. We apply the proposed approach on these tasks considering at time t=0, only one task arrived. Now calculate the F_n with n=1, and compare with deadline and if F_n is less than deadline then compare with F_0 which is initially infinity. Then again apply step 5 with increasing the n=n+1, and calculate the value of F_n with n = 2 and so on. We apply the same approach till the value of F_n is greater than deadline or F_n is greater than F_{n-1}. When any of the condition arrives, the loop terminates and prints n-1 as maximum number of checkpoint applicable on that task. After finding efficient checkpoint on maximum voltage we check available slack. If slack is available then reduce the voltage level and again apply same approach to find WCET and number of checkpoints. Applying the proposed approach we are able to minimize energy consumption with respect to maximum energy consumption.

Table 1: schedulability of task set

	a_i	e_i	d_i	n	ET	WCET	Schedulable check
T_1	0	2	10	2	5	5	Scheduled
T_2	1	3	20	3	7	12	Scheduled
T_3	5	7	25	4	13	25	Scheduled
T_4	6	9	40	3	15	40	Scheduled
T_5	10	3	50	3	7	47	Scheduled

The results shown in table 1 are the worst case execution time with number of checkpoints for each task. Once the number of checkpoints is calculated for first task, we check how many tasks arrive at the finish time of that task. At that time we apply the step 3 of proposed algorithm and again check the minimum deadline task. After finding the task with minimum deadline, the steps for finding maximum number of checkpoints is applied and so on. As table 1 shows the set of three tasks with maximum number of checkpoints applicable and the worst case execution time of these tasks.

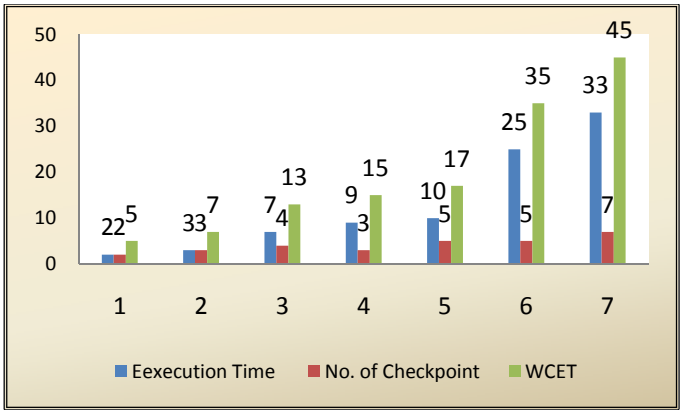

Figure 3: No. of checkpoint for various tasks (execution time) with WCET of each task

There are five tasks in the table 1 with its parameter values. ET is total execution time of individual task and WCET is worst case execution time of current task and its higher priority tasks. We find the number of checkpoints and also checks task schedulability on the basis of these parameters. As task T_4 has smaller deadline then task T_3, the task T_4 will be executed first and then task T_3 will be executed. Task T_4 has higher priority and preempt T_3 because it has smaller deadline

then T_3. Table shows the results in which task T_4 finishes its execution at 23 while T_3 finishes its execution at 34. When these tasks finish time compare with their deadline, both the tasks has been finished before their deadline then we can say the tasks are schedulable with fault tolerance.

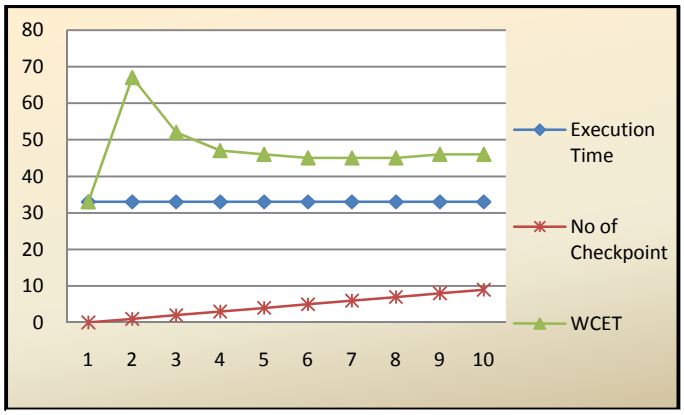

Figure 4: No. of checkpoint of task with execution time = 33

Figure 3 shows the execution time and number of checkpoint for various tasks. Each task has its execution time, number of checkpoint and WCET. Figure 4 shows A task with execution time e=33 and its optimal number of checkpoints which is 7. The WCET start increase after n=7. At n=7, WCET is 45, at n=8 WCET is 46. The WCET increases when number of checkpoints increases. So the optimal and efficient number of checkpoint will be 7.

VI. CONCLUSION

The proposed approach is based upon checkpointing to tolerate transient fault. Efficient number of checkpoints has been calculated with minimum energy consumption. The given algorithm is more suitable to finding efficient number of checkpoint for tolerating single transient fault. To achieve maximum utilization of processor and battery we proposed efficient checkpointing algorithm.

REFERENCES

[1]. C. M. Krishna, and Kang G. Shin, "On Scheduling Tasks with a Quick Recovery from Failure", IEEE TRANSACTIONS ON COMPUTERS, VOL. c-35, NO. 5, MAY 1986

[2]. Jun, Shen, Quiang, Wu, Xuwen, Li, Yanhua, Zang, "Research of The Real-time Performance of Operating System", 978-1-4244-3693-4/09/$25.00 ©2009 IEEE

[3]. Pankaj Kumar and R. K. Sharma, "A Novel Task Scheduling Algorithm for Time critical system s" International conference on Communication and Signal Processing, April 3-5, 2013, India ©2013 IEEE

[4]. D. K. Pradhan, Fault Tolerance Computing: Theory and Techniques. Prentice Hall, 1986.

[5]. Hagbae Kim, and Kang G. Shin,"Evaluation of Fault Tolerance Latency from Real-Time Application's Perspectives",IEEE TRANSACTIONS ON COMPUTERS, VOL. 49, NO. 1, JANUARY 2000

[6]. Francesco Quaglia, "A Cost Model for Selecting Checkpoint Positions in Time Warp Parallel Simulation",IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 12, NO. 4, APRIL 2001

[7]. Xiao Qin Hong Jiang David R. "An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems", Proceedings of the International Conference on Parallel Processing (ICPP'02) 0-7695-1677-7/02 $17.00 © 2002 IEEE

[8]. S. W. Kwak, B. J. Choi, and B. K. Kim, "An optimal checkpointing strategy for real-time control systems under transient faults," IEEE Trans. Reliab., vol. 50, no. 3, pp. 293–301, Sep. 2001.

[9]. C. M. Krishna, and Kang G. Shin, "On Scheduling Tasks with a Quick Recovery from Failure", IEEE TRANSACTIONS ON COMPUTERS, VOL. c-35, NO. 5, MAY 1986

[10]. Wei Luo, Xiao Qin, Member, IEEE, Xian-Chun Tan, Ke Qin, and Adam Manzanares "Exploiting Redundancies to enhance Schedulability in Fault-Tolerant and Real-Time Distributed Systems" IE4EE Transactions on system man and cybernetics – part A : systems and humans, VOL. 39, NO. 3, May 2009.

[11]. Izosimov, V., Pop, P., Eles, P., Peng, Z., Scheduling of fault-tolerant embedded systems with soft and hard timing constraints. In: Proceedings of the DATE 2008.

[12] Guohui Li. Fangxiao Hu& Ling Yuan." An Energy – Efficient Scheduling Scheme For Aperiodic Tasks in Embedded Systems". Third International Conference on Multimedia & Ubiquitous Engineering, 2009.

[13] Shabo Liu. Qinru Qiu & Qing Wn." Energy Aware Dynamic Voltage and Frequency Selection For Real Time Systems With Energy Harvesting."EDAA,2008

[14] K. H. Kim and J. Kim, An Adaptive DVS Checkpointing Scheme for Fixed-Priority Tasks with Reliability Constraints in Dependable Real-Time Embedded Systems, in Embedded Software and Systems, Vol. 4523, pp. 560-571, 2007.