Optimizing Resources in m-Redundant Avionic Systems - Comparative Evaluation

Swetha K*, Archana Sreekumar, Nithish N Nath, Radhamani Pillay V

Abstract

Redundancy is incorporated in safety critical systems for dependability and in traditional redundant systems, computing resources are inefficiently utilized under fault free condition. In this paper, an algorithm Resource Reclaimed Scheme (RR scheme) has been developed for scheduling of periodic tasks with critical aperiodic task arrivals in m-redundancy framework with resource augmentation. A novel task allocation and scheduling ensures effective resource utilization and simultaneous fault tolerant capability with graceful degradation. The proposed algorithm guarantees enhanced performance under fault free condition and less than full functionality up to m-1 faults. An earlier work has dealt with a similar dual system and this paper aims at a generalized m-processor scheme and in both cases, a comparison with the traditional redundant scheme has been studied for the significant impact of this novel idea. As a case study, an Avionic Mission System has being considered and simulation and performance evaluation validating the framework. An effective use of computing resources and improved process speedup which are critical measures for avionics systems, specially the fuel/weight ratio can have wide ramifications and contribute significantly in enhanced performance capability.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords: Safety critical systems; Fault Tolerant Scheduling; Redundancy; Resource Augmentation; Process Speedup

1. INTRODUCTION

Complex design of present safety critical like automobiles, nuclear power plants etc requires increased computational resources which led to the use of multi processors in the system. Among the static and dynamic scheduling, Safety critical system use static scheduling, so that the performance of the system is predictable.[1] Province can be allotted to handle the aperiodic tasks, such that it can handle both aperiodic and periodic tasks.

* Swetha K Tel.: +91-7736289265;
E-mail address: swethakumar18592@gmail.com.

1877-0509 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.453
thereby improving the performance of real time systems. Redundancy helps to achieve proper and safe functionality in case of processor failures. Underutilization of the resources are common in different redundancy techniques like 1:N modular redundancy, parallel redundancy, cold standby, hot standby etc. Proper Utilization of resources have enormous advantages especially in the field of avionics systems, these accounts for Size, Weight and Power (SWaP) constraints. 

As a case study, an Avionic Mission System has been considered. The main purpose of the mission system is to provide timely attack and also to maintain the adequate fuel/weight ratio. In this paper, a hardware redundancy method has been developed for recovery from physical faults. The scheme uses a m-processor environment that can tolerate faults, manages the computational resources efficiently and achieve functionality of the system. The proposed system can tolerate m-1 processor failures, and system will operate with graceful degradation. Proposed algorithm allocates and schedules both aperiodic and periodic tasks with computational resources being properly utilized. A novel task allocation and scheduling based on task criticality has shown promising results leading to resource augmentation.

The rest of the paper is organized as follows. Section 2 presents the literature survey and background study of scheduling of fault tolerant real time systems. The system model with the proposed approach and case study has been explained in Section 3. Sections 4 and 5 deal with analysis, simulation and performance evaluation. The conclusion and future scope are given in Section 6.

2. BACKGROUND STUDY

Dependability is the ability of a system to produce justifiable and trusted service[2] encompassing availability, reliability, safety and maintainability. Impairments to dependability, means of dependability and measures of dependability are given by Laprie[3]. Fault avoidance, fault tolerance, error removal and error forecasting are methods being used for attaining dependability. Fault tolerance achieves dependability by avoiding service failures in presence of faults. Fault tolerance employs both error detection and recovery. Fault tolerance in a system depends on the fault model and effect of faults in the system[4]. Failure can occur sporadically or instantaneously. Fault tolerant system has built in capability to preserve continued execution of programs in presence of faults[5]. The system can be considered fault tolerant if external assistance is not required to achieve fault tolerance. Partially fault tolerant system can be considered to be gracefully degradable, i.e the system reduces computational capability and shrink into a smaller system by discarding some previously used programs. This reduction is due to the decrease in the hardware used because of the faults. Permanent faults can occur due to the failure of components and recovery from such faults requires duplication of the components. Hardware redundancy is commonly used permanent fault tolerance technique, which is also employed in most of the present safety critical systems. Multiprocessor hardware designs includes parallel redundancy, hot standby, cold standby, 1:N redundancy. Timing checks, replication checks, reversal checks, coding checks, can be used for the identification faults[6]. Timing checks are employed using watch dog timers.[7]. Manimaran proposes algorithm for incorporating three fault tolerant approaches, primary back up, triple modular redundancy and imprecise computation.[8]. 

Fault tolerant algorithm for multiprocessors were put forward by Mosse et al[9]. Algorithms suitable for single processor are not optimal for multiprocessor systems[10]. Davis and Burns discuss hard real time scheduling in homogeneous multi processor environment[11]. Multi processor scheduling methods like static table driven scheduling, static priority pre-emptive scheduling, dynamic planning-based scheduling, and dynamic best effort scheduling are discussed in [1]. Global scheduling and partitioned scheduling can be present in multiprocessors. Based on criticality of the system and characteristics of the tasks scheduling strategy differs. Mission critical systems employs static table driven scheduling, where the start time and end time are pre calculated and placed in a table. The tasks are dispatched in accordance to the table during run time. Dynamic online best planning approach effectively schedules the aperiodic tasks. The above mentioned mixed scheduling scheme checks for the feasibility of schedule during run time. The schedule thus created will contain scheduling for both guaranteed task and arriving aperiodic tasks[12]. Dynamic flexibility of
scheduling algorithm to schedule arriving aperiodic tasks in different processors [13,14] while statistically allocating the periodic tasks allows better utilization of the m - processor available in a system where n - arriving aperiodic tasks are to be handled. Scheduling paradigms for tackling underutilization of computational resources and adaptive fault tolerance methods are discussed in[12,16,18]. Performance metrics such as Average Response Time $A$ [15] deals with the time taken for soft aperiodic task to execute so that the average execution time gets minimised and overall execution time can given by Speedup $(S)$. Slack Margin $(M)$ gives the time available for scheduling of aperiodic tasks[16]. Effective Utilization $(U)$ gives a measure of the processor utilization under faults and the percentage of tasks guaranteed to be scheduled has been given by Guarantee Ratio $(G)$[16].

The Avionic Mission System has been considered for the case study. The functionality of the system is given by a task set with dependencies[17]. Design vis-a-vis the fuel/weight ratio is a matter of considerable importance. Any means by which the resources available can be effectively optimized to reduce these concerns can maintain fuel/weight ratio at optimum levels.

3. APPROACH

A framework has been created where the task allocation and scheduling of periodic critical tasks meets hard deadlines even under fault. The number of processors will be decided based on workload of the system. Workload has been assumed to be 60% with 40% slack time. With 'm' processors, the system can handle up to 'm-1' faults. The proposed scheme creates a static schedule for each of the processor; tasks being scheduled according to the precedences. The synchronization for execution of tasks and in built health check of the processors has been done by the module called Health Check and Executive (HCE). The scheme proceeds by allocating and scheduling critical tasks on all processors and sharing of non critical tasks among the processors. Aperiodic tasks has been scheduled by meeting all the constraints and deadlines. Aperiodic critical tasks have been scheduled immediately on any one of the processors where as aperiodic non critical tasks scheduled based on the availability of slack time.

3.1 Objective

- To maintain safe functionality for fault and enhanced performance for fault free conditions
- Develop an algorithm for fault asst fault free conditions Aperiodic arrivals Simulation
- Performance Evaluation

3.2 Assumptions

- The processor failures are detected by watch dog timer with a bounded latency
- Critical aperiodic tasks have a higher priority than critical periodic task.
- The interarrival time between faults is greater than the lower bound of Mean Time Between Failure.
- The hinder to redundancy relies upon the parallelizable tasks present in the system.
- The Worst case execution time is taken which incorporates overheads needed for all communication and pre-emption.
- Each fault is assumed to be a single event rather than burst of faults.
- A minimum time interval between arrival of aperiodic tasks is assumed in the system.
- A slack margin of 20% is allocated for a safety critical system.

3.2 Avionic Mission System

The below Table.1. gives the task set of AMS with its task attributes. Criticalities of tasks are focused by the impacts a task will produce in the performance of a system on missing of its deadline. The tasks can be mainly classified as basic, monitoring, actuating and control tasks.
While implementing for aperiodic arrivals this work has been extended with changes in execution time. Precedence constraints are ensured by the static order scheduler, by giving its marching orders to each processors before the commencement of the system.

The above Fig.1. represents the fault tolerant model of the system with 'm' processors and Health Check and Executive. The processors continuously sends signals to the Health Check and Executive (HCE). When the HCE fails to get a signal back from one of the processor, it enters into fault mode and resumes the operation thereafter. Synchronisation between critical tasks in the processors are achieved by signalling of HCE. When a fault occurs, dynamic online reconfiguration has been employed by HCE by considering the best fit and first fit task allocation approaches. The local table maintains the status of the tasks to be executed by the processor and HCE maintains the global task table.

The Worst case scenario considers occurrence of m-1 faults and arrival of critical aperiodic tasks at the instant of execution of critical periodic tasks which affects the safe functionality of the system.
Algorithm for scheduling the system

Input: The tasks set
Output: Scheduling scheme normal mode

1. for i=1 to n do
2. check for the signal from HCE for the execution of first critical tasks in all processors
3. if (signal from HCE) then
4. start the first task in all the processors
5. continue the rest of tasks
6. if (critical periodic task)
7. add task to all processors
8. if (non critical periodic task)
9. add task to a specific processor
10. for arriving aperiodic task
11. check the criticality of task
12. if (critical aperiodic)
13. check the criticality of periodic task scheduled at that instant
14. if (critical periodic task scheduled)
15. replace the existing task by aperiodic task to any one processor
16. continue the periodic task scheduling in other processors
17. if (non critical periodic task scheduled)
18. add aperiodic task to any of the non allocated processor
19. if (non critical aperiodic task).
20. if arrival time of non critical task < the slack time
21. execute it in the first slack period after the release of the task
22. if time of arrival of the critical task > the slack time after the release of the task then
23. add the task to the queue
24. execute task at the end of the all tasks after in that processor
Algorithm for fault mode
Input: Any kind of failure to the processor
Output: Fault tolerant scheduling
1. for i=1 to hyper period do
2. if (alive signal from any processor is absent) then
3. The HCE signals the other processors to allocate the non critical tasks of failed processor without violating precedence constraints.

4. SIMULATION

The Time Optimization Resource and Scheduling (TORSCHÉ) toolbox in the MATLAB has been utilized for the task set scheduling and simulation for the proposed algorithm. Using the toolbox, one can easily research the application execution before the actual implementation.

Fig. 4. Scheduling trace with hard aperiodic arrival

The Fig. 4. shows the scheduling trace of periodic tasks in which critical periodic has been replaced by arriving hard aperiodic task in one processor whereas other processors continues the scheduling of periodic tasks. The Fig. 5. shows the scheduling trace of periodic tasks with an arrival of soft aperiodic task. The aperiodic task has been scheduled on the first available slack time in processor 3. The Fig. 6. and Fig. 7. shows the scheduling trace of tasks after 1st failure and 3rd failure with hard aperiodic arrivals.

Fig. 5. Scheduling trace with soft aperiodic arrival
5. PERFORMANCE EVALUATION

Process Speedup ($S$) of DRS and RR scheme are shown in Table 2 and it shows that the proposed RR scheme uses the computing resources more effectively. Slack Margin ($M$) of RR scheme under different fault conditions has been given in Table 3. Effective Utilization ($U$) given in Table 4 shows an increase in slack time in RR scheme. The Fig.8. shows the average response time ($A$) of the RR scheme with fault at different instants. Guarantee Ratio ($G$) for the TRS and RR schemes are given in Table 5. In fault mode, the guarantee with respect to the change in the execution time of the soft aperiodic task is given in the table.

Table 2. Process Speedup of TRS and RR schemes

<table>
<thead>
<tr>
<th>Approach Used</th>
<th>Normal Mode (time units)</th>
<th>Fault Mode (time units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRS</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>RR scheme</td>
<td>43</td>
<td>46</td>
</tr>
</tbody>
</table>

Table 3. Slack Margin under different conditions

<table>
<thead>
<tr>
<th>Approach Used</th>
<th>Normal Mode (time units)</th>
<th>Fault Mode (time units)</th>
<th>No. of Faults</th>
<th>Slack time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRS</td>
<td>5.33</td>
<td></td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>RR scheme</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 4. Utilization of TRS and RR schemes

<table>
<thead>
<tr>
<th>Approach Used</th>
<th>Normal Mode (time units)</th>
<th>Fault Mode (time units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRS</td>
<td>0.681</td>
<td>0.681</td>
</tr>
<tr>
<td>RR scheme</td>
<td>0.5861</td>
<td>0.6178</td>
</tr>
</tbody>
</table>

Table 5. Guarantee Ratio under different conditions

<table>
<thead>
<tr>
<th>Approach Used</th>
<th>Normal Mode (time units)</th>
<th>Fault Mode (time units)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No of Faults</td>
<td>Guarantee Ratio</td>
</tr>
<tr>
<td>RR scheme</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.889</td>
</tr>
</tbody>
</table>

Fig. 8. Average Response Time with fault at different time instants

6. CONCLUSION

This paper presents an improved m-redundant scheduling of an avionic mission system. The proposed RR scheme takes care of m-1 faults with m processors. The performance metrics elucidate the improved functional capabilities of the avionics mission system. By efficiently utilizing the extra slack margin available, response time of aperiodic tasks is reduced and thus the safety margins are improved. The scheme maintains safe functionality with graceful degradation under the worst case scenario. Future scope includes the consideration of burst aperiodic arrivals. An evaluation of mean time to recovery can be considered for future research.

References

7. Ellis F. Hitt and Dennis Mulecare, Fault Tolerance in Avionics, CRC Press LLC; 2001
16. A. Anand, Radhamani V. Pillay et.al Modelling, implementation and testing of an effective fault tolerant multiprocessor real-time systems, IEEE Conference on Parallel Distributed and Grid Computing, Dec 2012