

Efficient Mapping of Real-Time Task Graphs on a Cluster with Processor Failures

Alaa Amin, Reda A. Ammar and Swapna S. Gokhale
Unit 1155, Computer Science and engineering Department
University of Connecticut
Storrs, CT 06269-3155

1. Introduction

Real time applications are composed of one or more tasks that are required to perform their functions under strict timing constraints. These applications have to meet their deadlines amidst contradicting goals, while maximizing resource utilization. A task missing its deadline may result in a domino effect, possibly causing other tasks to miss their deadlines resulting in a system failure. Scheduling real time applications on multiprocessor systems is a very complex problem because of the multiple conflicting objectives that must be simultaneously achieved.

Existing scheduling techniques for real-time applications on clusters do not: 1) Consider application task structure [2,4], 2) Handle fragmentation of the processing power appropriately, 3) Make an effort to minimize communication among the tasks while retaining the degree of parallelism as specified by the task structure of the application. 4) Use an appropriate performance criterion (they use average execution time which is not sufficiently accurate and 5) Consider reliability issues [3].

In this paper we identify several conflicting objectives that must be satisfied by any reliable real-time scheduling scheme. In order to consider these conflicting goals in an integrated fashion, we propose an objective function, which can then guide the scheduling algorithms. The main focus of our research is to achieve the required reliability by exploiting available resources and/or application conditions.

2. Application and cluster model

In this section we describe the application and the cluster model that will form the basis for the design of reliable real-time scheduling scheme. We assume that the real-time computer system

consists of a set of homogenous multiprocessors, and a set of interrelated real-time tasks. The multiprocessor cluster system is fully connected. Each real-time task is denoted by $T_i = (s_i, t_i, d_i)$ where s_i is the start time of task i , t_i is the execution time of task i and d_i is the deadline of task i . We denote $Rpp_i = t_i/d_i$ as the required processing power of task t_i . At any time each processor j has certain amount of available processing power, which is denoted by App_j . Both Rpp and App are fractions less than unity. Tasks are non-preemptable, i.e., when a task starts execution on a certain processor, it finishes to its completion. The reliability of a processor P_j in time interval t is $exp(-I_j t)$ where is the I_j the failure rate of processor P_j [1]. Each application consists of different control structures such as sequential, fork-join, conditional branching, loop, or a combination of the above structures.

3. Objective function

The design of a reliable real-time scheduling scheme requires simultaneous consideration of the multiple contradictory objectives. The proposed scheme must: 1) Satisfy the required deadlines of each submitted application, 2) Provide a high level of reliability while executing an application, 3) Maximize the utilization of the available processing power to increase the rate of applications admitted to the cluster without any delay and avoid the fragmentation of the procession power, 4) Minimize the inter-communication among sequential tasks, and 5) Keep the degree of parallelism as specified in the applications' task graphs. In order to consider the above parameters in an integrated fashion, we develop an objective function that can guide the proposed scheduling algorithms. The proposed objective function includes real time deadlines, reliability, a penalty due to context switching of

tasks assigned to the same processor, and quantitative measures of the communication and processing power fragmentation. The objective function will depend on the task structure of the application. In this paper we present the objective function for an application with a tandem (sequential) task structure (Equation (1)), and briefly describe how this objective function can be modified for the fork-join task structure.

$$F = \sum_{\forall group l} (R_{d_l})^a (R_{f_l})^b (m_l)^g (D)^h \quad (1)$$

The objective function given in Equation (1) consists of four terms. The first term is referred to as the “deadline term” and it is given by Equation (2). The deadline term captures the effect of missing the deadline. A processor may execute a task(s) if its available processing power is greater than or equal to the processing power required by the task(s).

$$R_{d_j} = \text{mis}(t_l / d_l - PP_j) \quad (2)$$

$$\text{where } \text{mis}(x) = \begin{cases} 1-x & \text{if } x > 0 \\ 1 & \text{otherwise} \end{cases}$$

The second term or the “reliability term” represents the reliability of a certain processor when it is executing a group of tasks.

$$R_j(k, l, t) = \prod_{i=k}^{k+l} R_{ij} \quad (3)$$

where

R_j = reliability of processor j when it executes a tandem group of l tasks starting at task k .

R_{ij} = reliability of processor j when it executes task i .

The third term or the “grouping term” represents the number of tasks that can be grouped together and executed on the same processor to minimize the communication overhead among the tasks. Finally, the fourth term or the “fragmentation term” measures the processing power fragmentation of the processing power of a processor when different tasks are assigned to it. The goal is to decrease the fragmentation of the processing power of the processor. This term may be represented by the covariance between the processing power required by the tasks and

the available processing power of the processor, and is given by Equation (5).

$$D(Rpp, App) = \sum (Rpp - \text{mean}_{Rpp}) (App - \text{mean}_{App}) \quad (5)$$

Given the distributions of the required (Rpp) and the available (App) processing power, Equation (5) measures deviation between the two distributions. As the value of D increases, the degree of matching between the required processing power of the tasks and the available processing power of the processors improves, and the fragmentation of the processing power is reduced. The best case match up occurs for the highest possible value of D .

In case of the fork-join structure, the deadline and the reliability term will be the same as in the case of the tandem structure. The grouping term is not needed. Instead, another term, referred to as the “parallelization term” is added. This term captures the degree of parallelization resulting from assigning tasks to different processors. Inclusion of the parallelization term eliminates the need for the fragmentation term. The value of the parallelization term depends on the number of processors (n) and the number of branches in the fork-join structure (m). In the case of $m > n$ the upper bound of this term is m otherwise it is n .

References

- [1] X. Qin, H. Jiang, and D. R. Swanson, “An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems,” in *Proc. of ICCP*, 2002.
- [2] D. Mosse, R. Melhem, and S Ghosh, “Analysis of a fault-tolerant multiprocessor scheduling algorithm,” in *Proc. of FTCS-24*, 1994.
- [3] A. Dogan and F. Özgüner, “Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing,” in *Proc. of ICCP*, 2000.
- [4] G. Manimaran and C. Siva Ram Murthy, “A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis,” in *Proc. of FTCS-29*, 1999.