CHAPTER 8

Algorithms for Scheduling Imprecise Computations

J. W. S. Liu, K. J. Lin, W. K. Shih and A. C. Yu
Department of Computer Science
University of Illinois at Urbana–Champaign
Urbana, IL 61801

J. Y. Chung
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

W. Zhao
Department of Computer Science
Texas A&M University
College Station, TX 77843

Abstract

The imprecise computation technique ensures that all time-critical tasks produce their results before their deadlines by trading off the quality of the results for the computation time requirements of the tasks. This paper provides an overview of the problems in scheduling imprecise computations. It describes several workload models of imprecise computation that explicitly characterize the costs and benefits of the tradeoff and the scheduling algorithms that have been developed to achieve the desired tradeoff.

INTRODUCTION

In a (hard) real-time system, every time-critical task must meet its timing constraint, which is typically specified in terms of its deadline. Here, by a task, we mean a granule of computation that is treated by the scheduler as a unit of work to be allocated processor time, that is, scheduled. It is essential for every time-critical task to complete its execution and produce its result by its deadline. Otherwise, a timing fault is said to occur, and the result produced by the task is of little or no use. Unfortunately, many factors, such as variations in processing times of dynamic algorithms and congestion on
the communication network, make meeting all timing constraints at all times difficult. An approach to minimize this difficulty is to trade off the quality of the results produced by the tasks with the amounts of processing time required to produce the results. Such a tradeoff can be realized by using the imprecise computation technique [1–5]. This technique prevents timing faults and achieves graceful degradation by making sure that an approximate result of an acceptable quality is available to the user whenever the exact result of the desired quality cannot be obtained in time. An example of real–time applications where one may prefer timely, approximate results, to late, exact results is image processing. It is often better to have frames of fuzzy images produced in time than perfect images produced too late. Another example is tracking. When a tracking system is overloaded and cannot compute all the accurate location data in time, one may, instead, choose to have their rough estimates that can be computed in less time.

Since the advent of the imprecise computation technique, several workload models have been developed to characterize imprecise computations. These models explicitly quantify the costs and benefits in the tradeoff between the overall result quality and the computation time requirements. Several scheduling algorithms have been developed to exploit this tradeoff. This paper gives an overview of the recent results and discusses future directions in imprecise computation scheduling.

The remainder of the paper is organized as follows. The next section briefly describes the imprecise computation technique along with ways to implement imprecise computations. We then describe a basic deterministic workload model that characterizes imprecise computations and defines the performance metrics that provide the criteria for comparison of different scheduling algorithms. The problems in scheduling imprecise computations are at least as complex as the corresponding classical real–time scheduling problems. Almost all of the problems beyond that of scheduling unit–length, dependent tasks on two processors are NP–hard [6–8], and most simple heuristic algorithms for multiprocessor scheduling of dependent tasks have unacceptably poor worst–case performance [9–11]. For this reason, the approach that we have taken in scheduling dependent tasks is to first assign the tasks statically to processors and then schedule the tasks