Abstract: This paper describes a distributed algorithm for scheduling parallel programs represented by (macro-) dataflow graphs on multicomputer systems such that they are executed in a fault-tolerant way. Fault tolerance is based on dynamic redundancy comprising checkpointing, self-diagnosis and rollback recovery. The schedule is computed dynamically during the runtime of the process system. It works in a completely distributed way by making nodes which have finished a task responsible for allocating their ready task successors. The basic idea for achieving fault tolerance is to keep all input data sets of a task as checkpoints on different nodes in such a way that after a node failure the lost task can automatically be restarted on a remaining intact node. So, fail-soft behavior is realized in a fully distributed and user-transparent way. The algorithm is described in detail for the 1-fault case and some performance measurements on a multi-transputer system are given. Furthermore a graphical programming environment is presented which supports the programmer in all phases of program design by applying the abstract dataflow model of parallel computation.

Introduction

During the last few years parallel computers and workstation clusters have attracted more and more attention for high end applications which demand very
high computational performance like aerodynamics, plasma physics, meteorology, image processing or simulations of dynamical systems. The main advantages of these systems over more traditional approaches like vector-supercomputers are the good price/performance ratio based on the usage of (mostly) standard components. Another advantage is the scalability from small up to very large systems. As such systems grow, a single component failure gets more and more probable and means for fault-tolerance have to be applied. Even in small scale workstation-based environments the probability for a failure cannot be neglected because these systems are normally used by multiple users at the same time.

For utilizing the computational power offered by parallel computing systems it is necessary to find a suitable parallel solution to a given problem. This means that the following problems have to be solved:

1) dividing a job into (parallel) tasks,
2) assigning the tasks to the processing nodes of the parallel computer
3) (task scheduling problem),
4) providing fault tolerance.

In this paper we will concentrate on task allocation algorithms, where each task is executed completely on the same processor it has been started on. Task migration approaches which allow the preemption and reallocation of running tasks on other processors are more difficult to implement and are therefore not considered here. Descriptions of task migration algorithms can be found in [12].

For task allocation static and dynamic algorithms can be used. With the static approach the schedule is already computed at compile-time. As scheduling is an NP-hard problem, exact solutions are only practical for very small problem sizes [7], otherwise heuristic algorithms are usually employed. In both cases the task execution times have to be known in advance which is unrealistic for most practical applications. Static approaches are described in [14]. Dynamic scheduling algorithms compute the schedule during the runtime of the parallel program. The assignment of tasks to processors can either be made by a central master or in a distributed way by the individual processing nodes. Such dynamic solutions have e.g. been developed by [13] or [10]. Surveys of various scheduling problems and algorithms are given in [6] and [8].

The primary goal for nearly all scheduling algorithms developed so far has been high performance by exploiting as much parallelism as possible. Only