Application Recovery in Parallel Programming Environment*

G. T. Nguyen[1], V. D. Tran[1], and M. Kotocova[2]

[1] Institute of Informatics, SAS, Dubravska cesta 9, 84237 Bratislava, Slovakia
giang.ui@savba.sk
[2] Department of Computer Science, FEI STU, Ilkovicova 3, 81219 Bratislava,Slovakia

Abstract. In this paper, fault-tolerant feature of TOPAS parallel programming environment for distributed systems is presented. TOPAS automatically analyzes data dependence among tasks and synchronizes data, which reduces the time needed for parallel program developments. TOPAS also provides supports for scheduling, load balancing and fault tolerance. The main topics of this paper is to present the solution for transparent recovery of asynchronous distributed computation on clusters of workstations without hardware spare when a fault occurs on a node. Experiments show simplicity and efficiency of parallel programming in TOPAS environment with fault-tolerant integration, which provides graceful performance degradation and quick reconfiguration time for application recovery.

1. Introduction

Advances in information technologies have led to increased interest and use of clusters of workstations for computation-intensive applications. The main advantages of cluster systems are scalability and good price/performance ratio, but one of the largest problems in cluster computing is software [5]. PVM [14] and MPI [15] are standard libraries used for parallel programming for clusters and although they allow programmers to write portable high-performance applications, parallel programming is still difficult. Problem decomposition, data dependence analysis, communication, synchronization, race condition, deadlock, fault tolerance and many other problems make parallel programming much harder.

TOPAS (Task-Oriented PArallel programming System, formerly knows as Data Driven Graph - DDG [9][10][11]) is a new parallel programming environment for solving the problem. The objectives of TOPAS are: making parallel programming in TOPAS as easy as by parallel compilers, with the performance comparable with parallel programs written in PVM/MPI; making parallel programs structured, easy to understand and debug, and to allow error checking at compilation time for removing frequent errors; providing support for optimization techniques (scheduling and load balancing); providing facilities for Grid

* This work is supported by the Slovak Scientific Grant Agency within Research Project No. 2/7186/20