Integrating Fault-Tolerant Feature into TOPAS Parallel Programming Environment for Distributed Systems

G. T. NGUYEN, V. D. TRAN
Institute of Informatics
Slovak Academy of Sciences
Dubravska cesta 9, 842 37 Bratislava, Slovakia
giang.ui@savba.sk

M. KOTOCOVA
Department of
Computer Science and Engineering
FEI STU
Ilkovicova 3, 812 19 Bratislava, Slovakia

Abstract

In this paper, TOPAS[1] - a new 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, dynamic load balancing and fault tolerance. 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

Nowadays, 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. 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. Although these libraries 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.

As the number of processing elements and the completion application times are increased, the probability of fault occurrence during application execution is also increased. Most of applications are not designed for handling such situations; they would crash when faults occurs. That may lead not only to performance degradations but also to serious damages. Fault tolerance problem can be solved by hardware or software. Fault tolerant software solutions are often cheaper because they do not require special additional spare hardware and provide better flexibility. Although several algorithms and environments are available, users often have to handle faults explicitly; that requires additional work and increases application development cost. Environments, in most cases, are specified to certain solving problems; or require very high additional cost due to the fault-tolerant feature; or do not support parallel applications at all. Therefore, this work targets to develop a fault tolerant parallel programming environment, where every (parallel) application written in it is automatically fault-tolerant without requiring any special supports from hardware, operating systems or programmers.

TOPAS (Task-Oriented PArallel programming System, formerly Data Driven Graph - DDG [9][10][11]) is a new parallel programming environment for solving the problem. The objectives of TOPAS are as follows: 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 computing (heterogeneous architectures, task migration, fault tolerance).

The objectives are rather ambitious, but not unachievable. Environment description, scheduling and load balancing in TOPAS are already presented in [9][10][11]. This paper is oriented mainly to the fault-tolerant aspect of the TOPAS environment, but for clear context, section 2 shows the TOPAS main features including scheduling and load balancing. Section 3 focuses on fault tolerance support and Section 4 demonstrates real examples of parallel programs written in fault-tolerant TOPAS.

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