Chapter 38

AN EFFICIENT TRANSPOSITION ALGORITHM FOR DISTRIBUTED MEMORY COMPUTERS

Christina Christara, Xiaoliang Ding and Ken Jackson

Computer Science Dept.
University of Toronto
Toronto, Ontario M5S 3G4
Canada
{ccc, ding, krj} @ cs.toronto.edu

Abstract Data transposition is required in many numerical applications. When implemented on a distributed-memory computer, data transposition requires all-to-all communication, a time consuming operation. The Direct Exchange algorithm, commonly used for this task, is inefficient if the number of processors is large. We investigate a series of more sophisticated techniques: the Ring Exchange, Mesh Exchange and Cube Exchange algorithms. These data transposition schemes were incorporated into a parallel solver for the shallow-water equations. We compare the performance of these schemes with that of the Direct Exchange Algorithm and the MPI all-to-all communication routine, MPI_AllToAll. The numerical experiments were performed on a Cray T3E computer with 512 processors and on an ethernet-connected cluster of 36 Sun workstations. Both the analysis and the numerical results indicate that the more sophisticated Mesh and Cube Exchange algorithms perform better than either the simpler well-known Direct and Ring Exchange schemes or the MPI_AllToAll routine. We also generalize the Mesh and Cube Exchange algorithms to a $d$-dimensional mesh algorithm, which can be viewed as a generalization of the standard hypercube data transposition algorithm.

Keywords: Data transposition, shallow-water equations, Cray T3E, cluster of workstations, MPI, FFT, Helmholtz equation, $d$-dimensional mesh of processors.

*This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Information Technology Research Centre (ITRC) of Ontario, and Communications and Information Technology Ontario (CITO). We also thank Cray Research Inc. for allowing us to use one of their T3E computers to obtain our numerical results.
38.1 INTRODUCTION

Data transposition is required in many numerical applications, including both Fast Direct Solvers and the Alternating Direction (ADI) Method for linear algebraic systems as well as Spectral Methods for partial differential equations (PDEs). As a specific example, we investigate in this paper the data transposition required for the parallel solution of the shallow-water equations by a semi-implicit semi-Lagrangian scheme on a distributed memory computer. A non-linear two-dimensional (2D) Helmholtz equation must be solved at every time-step of this scheme. Applying a fixed-point iteration to the non-linear Helmholtz equation requires the repeated solution of a linear 2D Helmholtz equation. An elegant and effective means of solving the linear algebraic system arising from the spatial discretization of the linear Helmholtz equation is to use a series of independent Fast Fourier Transforms (FFTs) in one direction followed by a series of independent tridiagonal system solves in the other direction. This process is commonly referred to as a Fast Direct Solver (VanLoan92; Thomas94).

On a distributed-memory parallel computer, it is highly desirable to have each processor perform one or more FFTs using data stored locally on that processor and similarly to perform one or more tridiagonal system solves using data stored locally on that processor. However, to perform a FFT, a processor must have all the data from a row of the discretized 2D domain, while, to perform a tridiagonal system solve, a processor must have all the data from a column of the discretized 2D domain. That is, data transposition is required between the FFT and tridiagonal system solve phases of the Fast Direct Solver. This implies that each processor must exchange data with each other, a potentially time-consuming task. Several researchers have considered this data transposition problem; see, for example, (Choi95; Johnsson88; Nicol94) and the references therein.

In this paper, we propose the Mesh Exchange and Cube Exchange data transposition algorithms designed to exploit, respectively, a two- or three-dimensional processor interconnection network. The algorithms proposed do not require any restriction on the size of the matrix to be transposed, nor on the processor layout. Moreover, they can be extended to a $d$-dimensional processor interconnection network (mesh), for any $d$. Furthermore, the algorithm for a $d$-dimensional mesh is applicable on any machine on which a $d$-dimensional mesh can be embedded.

A parallel numerical method for the shallow-water equations incorporating these data transposition schemes was implemented using MPI (Gropp94). The method was run on both a Cray T3E computer with 512 processors and an ethernet-connected cluster of 36 Sun workstations. The numerical results indi-