Computers and Mathematics with Applications 70 (2015) 2762–2772

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

An application of partition method for solving 3D Stokes equation

Maria Ganzha[a], Krassimir Georgiev[b], Ivan Lirkov[b,*], Marcin Paprzycki[a]

[a] Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland

[b] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev, bl. 25A, 1113 Sofia, Bulgaria

ARTICLE INFO

Article history:
Available online 16 May 2015

Keywords:
Navier–Stokes
Time splitting
ADI
Incompressible flows
Parallel algorithm

ABSTRACT

In our previous work we have studied the performance of a parallel algorithm, based on a direction splitting approach, for solving of time dependent Stokes equation. We used a rectangular uniform mesh, combined with a central difference scheme for the second derivatives. Hence, the proposed algorithm required only solution of tridiagonal linear systems.

In our work, we are targeting massively parallel computers, as well as clusters of multi-core nodes. The somehow slower (experimentally-established) performance of the proposed approach was observed when using all cores on a single node of a cluster. To remedy this problem, we tried to use LAPACK subroutines from the multi-threaded layer library, but the parallel performance of the code (while improved) was still not satisfactory on a single (multi-core) node.

Our current work considers hybrid parallelization based on the MPI and OpenMP standards. It is motivated by the need to maximize the parallel efficiency of our implementation of the proposed algorithm. Essential improvements of the parallel algorithm are achieved by introducing two levels of parallelism: (i) between-node parallelism based on the MPI and (ii) inside-node parallelism based on the OpenMP. The implementation was tested on Linux clusters with Intel processors and on the IBM supercomputer.

1. Introduction

The solution of a tridiagonal system of linear equations lies at the heart of many programs developed for, so called, scientific computations. With the development and availability of multitude of parallel and vector computers, parallel algorithms (suitable for these machines) have appeared also for solving tridiagonal systems of equations.

Large tridiagonal systems of linear equations appear in many numerical applications. For instance, in [1], they arise in line relaxations needed by robust multigrid methods for structured grid problems. In [2] adaptive mesh refinement algorithm was used for a coupled system of nonlinear evolution equations of a hyperbolic type and a parallel algorithm was used to solve the tridiagonal systems of linear equations. The above papers used the classic parallel algorithm called the "partition method" introduced in [3].

* Corresponding author.
E-mail addresses: maria.ganzha@ibspan.waw.pl (M. Ganzha), georgiev@parallel.bas.bg (K. Georgiev), ivan@parallel.bas.bg (I. Lirkov), paprzyck@ibspan.waw.pl (M. Paprzycki).
URLs: http://inf.ug.edu.pl/~mganzha/ (M. Ganzha), http://parallel.bas.bg/~georgiev (K. Georgiev), http://parallel.bas.bg/~ivan/ (I. Lirkov), http://www.ibspan.waw.pl/~paprzyck/ (M. Paprzycki).

http://dx.doi.org/10.1016/j.camwa.2015.04.025
0898-1221/© 2015 Elsevier Ltd. All rights reserved.

On a serial computer, Gaussian elimination without pivoting can be used to solve a diagonally dominant tridiagonal system of linear equations. This algorithm, first described in [4], is commonly referred to as the Thomas algorithm. Unfortunately, this algorithm is not well suited for parallel computers. The first parallel algorithm for the solution of tridiagonal systems was described in [5]. It is now usually referred to as cyclic reduction. Stone introduced his recursive doubling algorithm in [6]. Both cyclic reduction and recursive doubling are designed for fine grained parallelism, where each processor owns exactly one row of the tridiagonal matrix. Wang proposed a partitioning algorithm that was aimed at more coarse-grained parallel computation, where the number of processors is many times smaller than the number of unknowns [3]. Diagonal dominance of the resulting reduced system in Wang's method was established in [7] and numerical stability of Wang's algorithm was analyzed in [8]. A unified approach for the derivation and analysis of partitioning methods applicable to solution of tridiagonal linear systems was given in [9,10]. There has also been attention directed towards a parallel partitioning of the standard LU algorithm. Sun et al. [11] introduced the parallel partitioning LU algorithm that is very similar to the Bondeli's divide and conquer algorithm [12]. For both the partitioning algorithms and the divide and conquer algorithms, a reduced tridiagonal system of interface equations must be solved. Here, each processor owns only a small number of rows in this reduced system. As an example, in Wang's partitioning algorithm, each processor owns one row of the reduced system. In [13], this reduced system is solved by recursive doubling. However, numerical experiments were performed only on a very small number of processors (typical for the times that this contribution was published).

Austin et al. [1] targeted parallel computers with thousands (to tens of thousands) of processors, such that for a 2D structured grid, line solves spanning hundreds of processors are realistic. They represent a memory efficient partitioning algorithm, for the solution of diagonally dominant tridiagonal linear systems of equations. This partitioning algorithm is well suited for current distributed memory parallel computers.

2. Alternating directions algorithm for Stokes equation

We consider the time-dependent Stokes equations written in terms of velocity $\mathbf{u}$ and pressure p:

$$\begin{cases} \mathbf{u}_t - \nu\Delta\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \times (0,T) \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega \times (0,T) \\ \mathbf{u}|_{\partial\Omega} = 0, \qquad \partial_n p|_{\partial\Omega} = 0 & \text{in } (0,T) \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, \qquad p|_{t=0} = p_0 & \text{in } \Omega, \end{cases} \tag{1}$$

where $\mathbf{f}$ is a smooth source term, ν is the kinematic viscosity, and $\mathbf{u}_0$ is a solenoidal initial velocity field, with a zero normal trace. The time interval $[0, T]$ is discretized on a uniform mesh and τ is the time step. We solve the problem (1) in the domain $\Omega = (0, 1)^3$, for $t \in [0, 2]$ with Dirichlet boundary conditions.

Guermond and Minev [14,15] introduced a novel fractional time stepping technique for solving the incompressible Navier–Stokes equations. This technique is based on a direction splitting strategy. They used a singular perturbation of the Stokes equation. In this way, the standard Poisson problem in the projection schemes was replaced by series of one-dimensional second-order boundary value problems.

Usage of central differences for the discretization in space, for the one-dimensional boundary value problems, leads to the solution of tridiagonal linear systems. In our original research we developed MPI code based on an application of the partition method for solving the tridiagonal system of linear equations, which arise in the alternating directions algorithm [16,17]. The analysis of experimental results showed that the algorithm is very well suited for distributed memory parallel computers but it has unsatisfactory performance on a single (multi-core) node of a cluster. To try to alleviate this deficiency, we used LAPACK subroutines from a multi-threaded layer library, for the solution of tridiagonal linear systems [18]. The experimental results showed that the code needs additional improvements. Here, one has to recall that to maximize performance of a cluster of multi-core nodes, one has to, first, optimize the per-node performance.

In the current work, we have developed a hybrid-parallel code based on combination of the MPI and the OpenMP standards. In our application of the partition method, each MPI process owns a small number of rows of the tridiagonal matrix, but the linear system has multiple right hand sides. In our hybrid implementation, each OpenMP thread solves the tridiagonal system with a small number of rows and a small number of right hand side (RHS) vectors. Specifically, let us consider a discretization, in space, with n_x, n_y, and n_z points in direction x, y, and z respectively. Then the one-dimensional problem in the x direction leads to a linear system with n_x rows and $3n_yn_z$ RHS vectors for the "velocity update" step and n_yn_z vectors for the "penalty" step (in the alternating directions algorithm [15]). We use $m = m_xm_ym_z$ MPI processes and k OpenMP threads. In the "penalty" step, each MPI process computes the coefficients in $\frac{n_x}{m_x}$ rows and $\frac{n_y}{m_y}\frac{n_z}{m_z}$ RHS vectors. Let us denote by M the number of rows and by K the number of RHS vectors owned by a single MPI process. In our current implementation each OpenMP thread solves a linear system with M rows and $\frac{K}{k}$ RHS vectors.

3. Experimental results

Let us now report on the experiments we have performed with the current implementation of the solver. In the experiments, we consider the time-dependent Stokes equation (1). The discretization in time was done with time step 10^{-2}.