

An Efficient Scalable Finite Element Model for Simulating Large Scale Hydrodynamic Flows

Prasada Rao

Department of Civil and Environmental Engineering, California State University,
Fullerton, CA 92831, E-mail: mprasadarao@fullerton.edu

Abstract

In this work, a parallel finite element method (FEM) for solving two dimensional shallow water flow equations is developed on a distributed computing environment. The parallel program is based on domain decomposition principles and inter processor communication was achieved through Message Passing Interface (MPI) protocols. The parallel implementation is arrived at by using the existing parallel software libraries (i.e., PETSc). The end discretized equations in the serial code are solved using the Galerkin finite element method. The results indicate that by using the parallel software tools that are currently available, one can arrive at an efficient scalable code which can be used across different parallel machines.

Key words Two dimensional, open channel, transients, speedup, solver, preconditioner

Introduction

Modeling large-scale hydrodynamic flows using the serial hydrodynamic models is constrained by the required computational time. A satisfactory simulation would require a strong coupling of these serial models with various high performance computing tools, so as to increase the applicability of these models for modeling real life flows. Arriving at parallel models can be done either by (a) writing an entirely new parallel code (b) converting an existing serial code onto a parallel one. While (a) does look appealing, it doesn't take into account the time that the serial code developers have invested to arrive at it. Also since all serial codes have an audience, writing a new parallel code may not retain all the serial code audience. The approach (b) addresses these two limitations. The existing parallel software libraries, facilitate in converting an existing serial code to a parallel one, with out changing the overall structure of the code. The focus of this work is not in writing a new parallel model for modeling hydrodynamic flows, but in improving the predictive capabilities of an existing serial model by imbedding into it different High Performance Computing (HPC) tools. A tight coupling of these two will address the limitations of the serial model for modeling large scale flows.

Over the past three decades many serial numerical models (largely focusing on the physics of the problem) for modeling flows in open channels/estuaries have been developed. Like any other serial code, the drawback in using these models for modeling large scale flows in real time is the required computational time. For explicit finite element formulations, this limitation is largely contributed to the time spent for solving the end linearized system of equations [1], $\mathbf{AX}=\mathbf{B}$. Using an efficient parallel solver

coupled with a preconditioner, can significantly reduce the time spent in the solver module thus translating to computational savings. The advantages of using solvers from the Krylov family, has not been fully investigated in computational hydraulics. In this work the solvers that are available in the Portable Extensible Toolkit for Scientific Computation (PETSc) library [2] have been used. PETSc library, built at the Argonne National Laboratory, is a powerful set of freely available multi-platform compatible tools for the solution of large-scale problems modeled by partial differential equations. The software uses MPI [3] for all interprocessor communication. An overview of PETSc can be found in Gropp and Smith [4].

Governing Equations and Numerical Scheme

The two dimensional depth averaged flow equations in conservation form can be expressed as [5]

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(hu) + \frac{\partial}{\partial y}(hv) = 0 \quad (1)$$

$$\frac{\partial}{\partial t}(hu) + \frac{\partial}{\partial x}\left(hu^2 + \frac{gh^2}{2}\right) + \frac{\partial}{\partial y}(huv) = gh(S_{ox} - S_{fx}) \quad (2)$$

$$\frac{\partial}{\partial t}(hv) + \frac{\partial}{\partial x}(huv) + \frac{\partial}{\partial y}\left(hv^2 + \frac{gh^2}{2}\right) = gh(S_{oy} - S_{fy}) \quad (3)$$

where h is the flow depth, u and v are the depth-averaged velocities in the longitudinal and normal directions, g is the acceleration due to gravity, S_{ox} and S_{oy} are the bed slopes of the channel along the x and y directions, and S_{fx} and S_{fy} are the friction slopes along the x and y axes.

The above equations, can be rewritten in matrix notation as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S} \quad (4)$$

In Eq. (4), \mathbf{F} and \mathbf{G} are the flux vectors that consists the nonlinear flow terms. The above set of equations can be discretized using the traditional finite element approach. Using the linear shape functions N_i , the finite element procedure approximation for \mathbf{U} , can be expressed as

$$\mathbf{U} \approx \hat{\mathbf{U}} = \sum_{j=1}^n N_j \mathbf{U}_j \quad (5)$$

with $\hat{\mathbf{U}}$ representing the approximate solution to the true solution, \mathbf{U}_j is the nodal value of the dependent variables and n is the total number of nodes in the domain of interest [6].

Choice of the up-winding matrix and associated numerical parameter can be found in the works of [7,8]. Equation 4 can be written in matrix notation as

$$\mathbf{M}_{ij} \left(\frac{\partial \mathbf{U}}{\partial t} \right)_j + \mathbf{K}_{ij} \mathbf{U}_j = \mathbf{R} \quad (6)$$

where \mathbf{M}_{ij} is the mass matrix, \mathbf{K}_{ij} is the stiffness matrix and \mathbf{R} is the residual vector from the finite element integration (\mathbf{R} is zero at the exact solution). Since the system is non-linear, the well known Newton-Raphson routine is used to solve. The general algorithm is

$$\begin{aligned} \Delta \mathbf{r}_i &= -\mathbf{J}^{-1} \mathbf{F} \\ \mathbf{r}_{i+1} &= \mathbf{r}_i + \Delta \mathbf{r}_i \end{aligned} \quad (7)$$

where \mathbf{J} is the Jacobian matrix (the partial derivative of Eqs. 1,2 and 3 with respect to the dependent variables) and \mathbf{r} is the solution vector. The end non linear system of equations can be solved using an iterative solver and the solution vector updated before marching the solution with time, until the desired time period is reached for transient simulations.

The CFL stability criteria, which governs the choice of time step can be written as [5]

$$\Delta t = C_n \frac{\Delta x}{\max(|u| + \sqrt{gh})} \quad (8)$$

where C_n is the Courant number (≤ 1), and Δx is the grid spacing. With selected Δx and knowing flow conditions, the time step can be evaluated using Eq. 7.

Parallel Formulation

The present parallel formulation is based on domain decomposition (DD) principles. In the early stages of computing, DD techniques were applied to perform large computations on machines with insufficient memory. To address the memory problem, the domain was divided into sub-domains. Computations were carried out over the smaller domains one at a time, during which time the other sub-domains resided on tape. With the growth in computational hardware and low memory cost, the DD methods were pushed to the background. However, with the advent of parallel computing these methods are again blossoming [9]. The idea in these methods is to divide the original computational domain into a series of domains, with their number equal to the number of processors in the communicator. A processor, which is identified by its Rank, is assigned to each of the domains to perform the computations (i.e., solving the equations) over the nodes in that local domain. An ideal DD would require that the computational load among all the processors is fairly uniformly distributed. This would require that the initial spatial domain be carefully partitioned. Graph partitioning software, like ParMETIS can be used to this end. For the given domain geometry (i.e., nodes and their connectivities), ParMETIS splits the spatial domain equally among the processors in the communicator.

The inter-processor communication is required along the domain interfaces at the end of every time step was achieved through the use of Message Passing Interface, MPI [3]. MPI is a portable message passing tool. The codes written using MPI are portable to all

platforms supporting the message-passing model, which includes the distributed-memory parallel supercomputers (MPP's), shared-memory multiprocessors (SMP's) and networks of workstations (NOW's).

PARALLEL PERFORMANCE MEASUREMENTS

The performance tools used to measure the parallel performance aspects are (a) speedup (b) efficiency and (c) scalability. Speedup is the ratio of the time required for the serial to complete the execution to the time required by the parallel code. The efficiency indicates how well the processors are made use of in the execution of the parallel code. The values of E ranges from 0 to 1. The scalability of a parallel code is a measure of its capacity in yielding increased speedup in proportion to the number of processors [10].

Application

The performance details of the end code has been tested by applying it to a sample problem that consisted of 128,056 unknowns. The initial flow conditions in the rectangular flow domain were assumed to be uniform ($h = 0.2m$, $u = 0.1m/s$, $v = 0$). Over a portion of the upstream boundary, transient flow conditions are induced ($h = 0.5m$, $u = 0.3m/s$, $v = 0$). With time, this boundary condition finds wider area to propagate in the flow domain, thus causing two dimensional flow. The size of the global matrix was equal to 128,056. The solution obtained from the serial code has been used as the bench mark solution to measure the accuracy of the parallel solution. The serial code had a direct solver for solving the end system of equations.

The serial code was modified to facilitate running it in PETSc environment. The global matrix was stored in compressed sparse row format where only the non-zero values along with their matrix location indicies are stored. While PETSc library has different formats available to store the end global matrix, in the present work, the MatMPIBAIJ routine has been used. The parallel code was run on a cluster of HP superdomes. This 224 processor HP supercomputer consists of four HP Superdome servers interconnected by a high speed, low latency "hyperfabric" network.

Owing to the small size of the problem, the parallel code was run on up to 8 processors. The affect of the Krylov solver on the total time spent for solving the system of equations at one time step is shown in Figure 1. The affect of coupling the BiCGSTAB [11] solver with a preconditioner is shown in Figure 2. The solvers and preconditioners used in this investigation are available in the SLES interface of the PETSc libraries. The end used, can directly use these solvers, after storing the elements of the global matrix and vector in PETSc enabled format. Both these plots indicate that choosing the right solver combination does help in reducing the over all required time. Figure 3 plots the required CPU time, for the solution to reach $t=5s$, across 8 processors.

Conclusions

In this work, a two dimensional finite element was imbedded with different high performance computing tools, that are available in the PETSc library, and its performance tested for a sample two dimensional open channel flow problem. The parallel code is based on domain decomposition techniques. The results indicate that by using the currently available parallel software libraries, existing serial codes can be converted to fairly efficient parallel codes. Such a formulation would also retain all the serial modeling audience, as the changes required in the serial code are minimal when compared to writing a new parallel code.

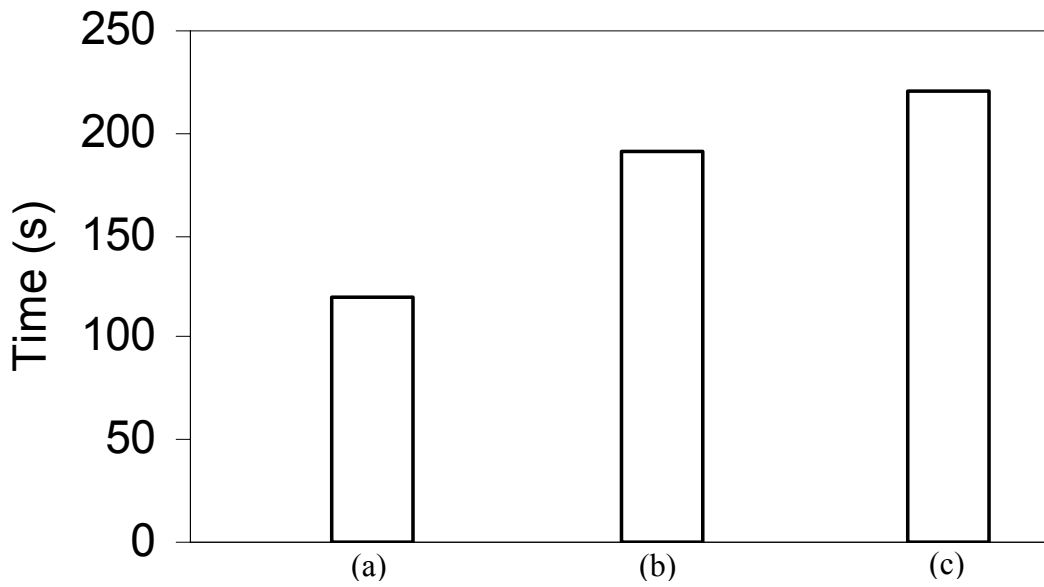


Figure 1. Effect of Krylov solver on solver module
 (a) BiCGSTAB (b) Conjugate gradient (c) Conjugate residual

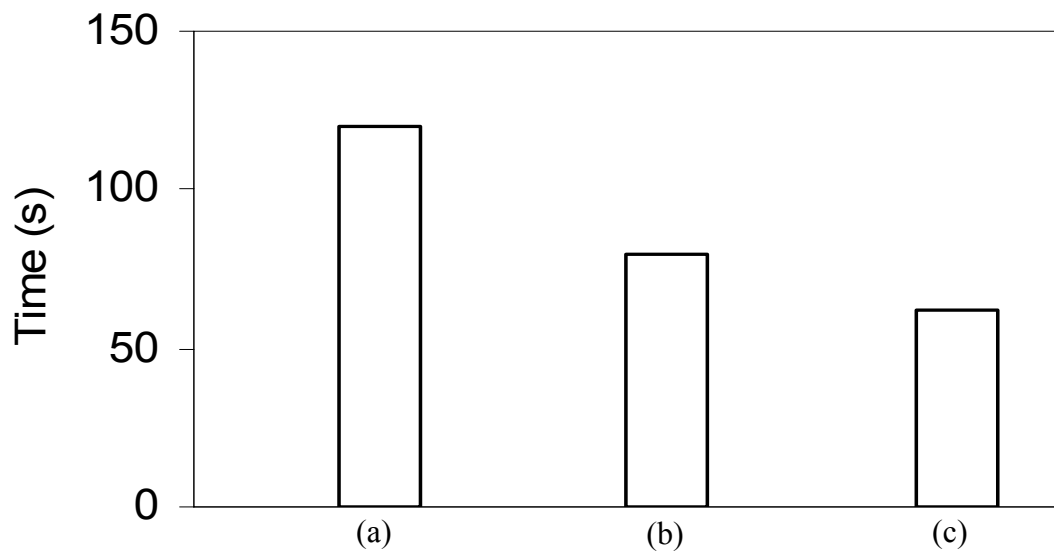


Figure 2. Effect of coupling Bi-CGSTAB solver with a preconditioner
 (a) no preconditioner (b) ILU (c) Additive Schwarz method

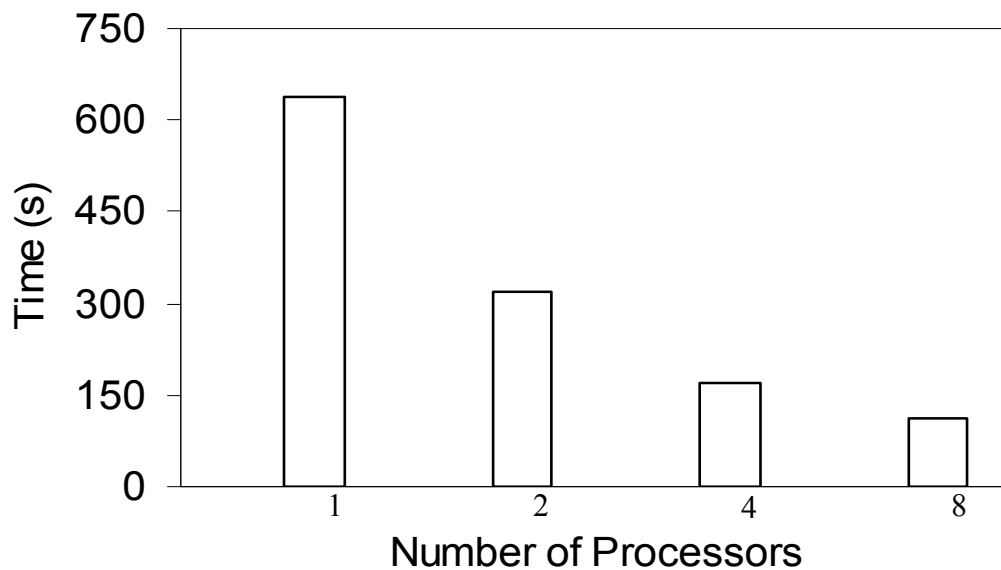


Figure 3. Observed CPU times across 8 processors (BiCGSTAB solver with Additive Schwarz method)

References

1. Eriksson, K., D. Estep, P. Hansbo and C. Johnson (1996), *Computational Differential Equations*, Cambridge Univ. Press.
2. Satish Balay, William Gropp, Lois Curfman McInnes, and Barry Smith. (2003), PETSc home page, <http://www.mcs.anl.gov/petsc>
3. Gropp, W., E. Lusk, and A. Skjellum (1994), *Using MPI: Portable Parallel Programming with the message passing Interface*, MIT Press.
4. Gropp, W. and Barry F. Smith (1994), Scalable, extensible, and portable numerical libraries, in Proceedings of Scalable Parallel Libraries Conference, 87-93, IEEE, Los Alamitos, CA.
5. Chaudhry, M.H. (1993), *Open Channel Flow*, Prentice Hall, NY.
6. Hughes, T.J. (1987), *The Finite Element Method*, Prentice-Hall Inc., Englewood Cliffs, NJ.
7. Berger, R.C., and Stockstill, R.L. (1995), Finite element model for high velocity channels, *J. Hydr. Engrg.*, ASCE,113,710-716.
8. Hicks, F.E. and Steffler, P.M. (1992), Characteristic dissipative galerkin finite element scheme for open channel flow, *J. Hydr. Engrg.*, ASCE,118,337-352.
9. Smith, B.F., Bjorstad, P. and William Gropp. (1996), *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press.

10. Kumar, V., Grama, A., Gupta, A. and Karypis, G. (1994), *Introduction to Parallel Computing: Design and analysis of algorithms*, Benjamin Cummings Inc.
11. Van der vorst, H.A. (1992), A fast and smoothly converging variant of BiCG for the solution of non symmetric linear systems, *SIAM J. Sci. Stat Comput.*, 13:631-644.