IMPLEMENTATION OF THE BAROTROPIC VORTICITY EQUATION ON THE M P P

Max J. Suarez

NASA/GSFC/611
Greenbelt, Maryland 20771

and

Jim Abeles

Science Applications Research
4400 Forbes Blvd.
Lanham, Maryland 20706

ABSTRACT

A finite difference version of the equations governing two-dimensional, non-divergent flow on a sphere is implemented and integrated on the MPP. The MPP's performance is then compared with the CYBER's.

Keywords: Numerical weather prediction, computational fluid dynamics.

INTRODUCTION

The purpose of the work described here was to demonstrate the feasibility of using a massively parallel architecture to solve the hydrodynamic equations as they are used in numerical weather prediction (NWP).

Models used in NWP are commonly divided in two parts: the "dynamics" and the "physics". The dynamics performs the time integration of the equations of motion. The physics computes the heating, friction, and sources and sinks of water vapor. These two parts present very different problems to a highly parallel machine.

Many of the calculations in the dynamics involve the parallel updating of the many degrees of freedom allowed in the discretization and are thus very suitable to a machine like the MPP. Occasionally, however, it is necessary to obtain a spectral transform or solve an elliptic equation. These problems, although parallel, are non-local and thus difficult to implement efficiently on the MPP's nearest neighbor network. Fortunately, the non-local calculations can be minimized by a suitable choice of numerical scheme. For example, grid-point models, in which the equations are finite differenced in a latitude-longitude lattice, are much preferable to spectral models, which require frequent transformations between physical and spectral space. Still, non-

local calculations are not completely avoidable. In particular they appear in the solution of elliptic equations that occur when implicit time differencing schemes are used. Although these too could be avoided by using an explicit method (which is in fact done in many models, even on serial computers), we feel the architecture should not be so specialized as to completely forbid such choices.

Problems in the physics part of the codes are probably even more serious. In these, it is their non-parallel, rather than non-local, nature that makes for difficulties. As an example consider condensation. In most models this is done level by level, testing for super-saturation and passing the excess water to the next level below. That level in turn may become super-saturated, or may have been so already. The condensation calculation is then repeated and so on until "rainfall" reaches the surface. If parallelism is exploited by mapping each latitude-longitude point onto a different processor (this is really the only practical alternative in a machine with as many processors as the MPP), each one will in general encounter different condensation conditions. Processors at all grid points where there is no condensation, for example, will be idle in this segment of the code, and parallelism will be lost.

THIS STUDY

To start looking at the problems one faces with a parallel architecture, we decided to use the barotropic vorticity equation as a model of the "dynamics" part of NWP models. In this way we can test both the parallel grid-point updating segments and the more challenging problem of solving an elliptic equation.

At each step of the calculation we update the following equation for a new value of the vorticity:

$$\frac{\partial \zeta}{\partial t} = -\frac{u}{a\cos\varphi}\frac{\partial}{\partial \lambda}(\zeta + f) - \frac{v}{a}\frac{\partial}{\partial \varphi}(\zeta + f) \qquad (1)$$

where ζ is the vorticity, and u and v are the zonal and meridional velocity components of the non-divergent flow, φ and λ are the latitude and longitude, and f is the Coriolis parameter. As mentioned already, (1) is solved by finite-differencing on a latitude longitude grid. A leap-frog differencing scheme is used in time. Once a new value of the vorticity is obtained from the discrete version of (1), the Poisson equation:

$$\nabla^2 \psi = \zeta \qquad (2)$$

is solved for the stream-function. To solve (2) we use a "fast" method in which the equations are first Fourier transformed in the zonal direction, then finite differenced in the meridional direction and solved as a set of tri-diagonal systems. The velocity components u and v are then obtained from

$$u = -\frac{1}{a}\frac{\partial \psi}{\partial \varphi}$$

$$v = \frac{1}{a\cos\varphi}\,\frac{\partial \psi}{\partial \lambda}$$

Having u and v, (1) can be updated again and the cycle completed.

To test the model, (1) was forced with sources of angular momentum and eddy vorticity, and damped by a linear drag.

Tests were conducted in parallel on the MPP and the CYBER 205 at Goddard Space Flight Center. The CYBER calculations were done with HALF-PRECISION (32-bit) arithmetic. Both MPP and CYBER codes were optimized for their machines to the best of our abilities; but both used exactly the same algorithm. In particular, the "fast" solver used for (2), which is very efficient on the CYBER, was retained on the MPP. On the other hand, a 128x128 square grid was used in both cases. This is optimal for the MPP. Higher resolution would require either doing a prohibitive amount of I/O, or keeping more than one grid-point per processor, which is not possible with the MPP's limited memory. The CYBER efficiency,in contrast, is independent of resolution for all practical choices.

RESULTS

The timing results are shown in Table I. We have separated these in two parts: the time spent solving the Poisson equation (2), and all the rest, which is mostly computing the right-hand-side of (1) and a little housekeeping. Units are msec./timestep. At the resolution used, we were taking 200 time steps per day. As may be seen, the code is approximately four times slower on the MPP than on the CYBER. This poor performance, however, is due entirely to the Poisson solver, which runs some ten times slower on the MPP. The updating of the vorticity equation is twice as fast on the MPP. This is a very encouraging result.

CODE	MPP	CYBER
UPDATING VORTICITY EQUATION	5.7	11.6
SOLVING POISSON EQUATION	65.0	6.3
TOTAL	70.7	17.9

TABLE I

If the NWP model is grid-point and uses explicit time differencing, the elliptic solver is not needed, and the MPP (or an MPP-like machine) should do very well in the dynamics. However, even if the model is implicit, and one or several elliptic equations have to be solved, the situation is not as bad as Table I would indicate. In a typical situation we would be solving some 40 equations like (1) (4 variables [u,v,T,q] at 10 levels), but at most 10 equations like (2). Using these figures, we can extrapolate our results to a full, grid-point, semi-implicit NWP model. This is shown in Table II. As may be seen, the situation is much improved; the MPP is now at near CYBER performance, even doing all ten vertical modes implicitly.

Obviously, much work remains to be done before massively parallel machines can be used efficiently for numerical weather prediction. In particular, it is imperative that much more parallel formulations and/or algorithms be developed for the physics codes, a problem we have not even begun to address here. Nevertheless, we feel that the results presented indicate a very real possibility of using MPP-like machines in NWP.

CODE	MPP	CYBER
UPDATING VORTICITY EQUATION	228	464
SOLVING POISSON EQUATION	650	63
TOTAL	878	527

TABLE II