Geophysical Research Abstracts,
Vol. 10, EGU2008-A-00553, 2008
SRef-ID: 1607-7962/gra/EGU2008-A-00553
EGU General Assembly 2008

Speeding up of the MOST Program

M. Lavrentiev Jr. (1,2), A. Romanenko (2)

(1) Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia, (2) Novosibirsk State University, Novosibirsk, Russia (mmlavr@nsu.ru)

Recently MOST (Method of Splitting Tsunami) software package has been accepted by the USA National Ocean and Atmosphere Administration as the basic tool to calculate tsunami wave propagation and evaluation of inundation parameters. Acceleration of codes should provide additional time for tsunami hazard mitigation. Part of MOST software, responsible for calculation of wave propagation over deep ocean is addressed. Perspectives of performance optimization through comparison of different architectures and parallel techniques have been studied.

Wave propagation is simulated by nonlinear shallow water system. System is presented in special form, which admits splitting along coordinates. Nonlinear system is solved by iterations. Each iteration consists of solution to linearized system by splitting. Numerical algorithm is described as follows:

1. read of input data and variables initialization;

2. for each time step deviation wave and velocity field are calculated along axis X and then along axis Y.

Calculations along coordinates could be performed independently. This suggests performance gain through program parallelization. Several technologies for algorithm parallelization have been considered: two for distributed memory architectures (these could be also applied for shared memory systems) and one for shared memory architectures. Both MPI and OpenMP technologies have been used.

For distributed memory systems the following parallelization strategy has been finally chosen. Computational domain was split along Y axis to be addressed to different

processors. First, all processors perform calculations along X axis, Then the first processor make calculation through the corresponding part of Y axis and pass result to second processor through common boundary and so on. After completion of calculations along Y axis, all processors write results and repeat the loop. This means that processors perform calculations with shift at one half of time step.

For shared memory systems iterations along coordinates were paralyzed. The best performance have been achieved using static schedule type (i.e., each computational thread has equal number of loop iterations).

For distributed memory architectures performance gains was 4 times for 8 processors. Better acceleration, 4 times on 6 processors was achieved in case of shared memory. Further increase in number of computational nodes does not result in additional performance gain. Analysis of stagnation reasons suggests better result for combination of distributed and shared memory parallelization.