Parallel Performance of the Tectonic Loading Process Model at Transcurrent Plate Boundaries

Kengo Nakajima[(1)], Chihiro Hashimoto[(2)] and Mitsuhiro Matsu'ura[(2)]

(1) Department of Computational Earth Sciences, Research Organization for Information Science and Technology (RIST), Tokyo, Japan (e-mail: nakajima@tokyo.rist.or.jp, phone: +81-3-3436-5271, fax: +81-3436-5274). (2) Department of Earth and Planetary Physics, The University of Tokyo, Tokyo, Japan (phone: +81-3-5841-4318).

Abstract

In this study, existing code for the tectonic loading process simulation at transcurrent plate boundaries have been *parallelized* on workstation clusters using MPI. Matrix assembling and inversion processes have been mainly optimized for parallel computing. 2-way parallelization method has been developed for local operation both for parameter points and data/integral points. Good parallel performance results are obtained.

Introduction

In the tectonic loading process simulation at plate boundaries, boundary integral methods or boundary element methods are widely used. In these types of methods, assembling and inversion of large scale dense matrices are required. This matrix assembling and inversion parts are the most expensive process in the entire computational process from the viewpoint of both computation time and memory usage. Therefore, utilization of parallel computer is very important and critical issue in this research area in order to get higher resolution and to treat larger regions for computational model.

In this study, existing code[1] for the tectonic loading process simulation at transcurrent plate boundaries have been *parallelized* on workstation clusters. In the following sections (1) Outlined of the Original Code, (2) Parallelization of the Code and (3) Results and Conclusions are described.

In this work we adopted message passing style parallel programming model using MPI[2] mainly because of its efficiency, portability and robustness.

Physical and Numerical Modeling

Details of the physical modeling of the original code are found in [1]. Therefore just brief outline of the physical modeling is shown here.

Under the structure of the lithosphere-asthenosphere system and plate boundary, the physical, process of the stress accumulation on the plate boundaries are essentially governed by the coupled nonlinear system that consists of (1) fault slip equation, (2) shear stress equation (3) constitutive relation between shear stress and fault slip. In the method in [1], these coupled equations are linearlized by the Levenberg-Marquardt type least square method[3]. Large scale dense coefficient matrices are provided and linearlized equation systems are solved by direct method by Gaussian elimination[4]. Usually, this type method requires N^2 order of memory storage and N^3 order of operations for N unknowns. Therefore, parallel computing is important and critical both for CPU time and memory requirement.

Parallelization

Profiling

Before starting parallelization, the original code has been analyzed using profiling tool *pixie* on UNIX system. According to the profiling results, 96.5% of the entire process are devoted to the following two processes :

%time	seconds	procedure
77.2	375.3789	Matrix Assemble
19.3	93.7490	Gaussian Elimination

This measurement was executed on COMPAQ Alpha 21164 (500MHz) compatible single CPU workstation with Digital UNIX. The original code was written in FORTRAN90 and compiled by Digital FORTRAN compiler. In the parallelization process, main efforts will be done in these 2 parts. In this paper, parallelization on the matrix assembling part is mainly discussed.

Matrix Assembling

Original FORTRAN source code for matrix assembling part is shown in Fig.1 (i). There are 3 loops in the code. Inner 2 loops (both for *1-ma*) compute coefficient matrix *A(i,j)* and RHS vector *B(j)*. Unknowns *X* are coefficient of spline basis and vector length is *ma*. Each coefficient is defined on *parameter point*[1]. The outermost loop (for *1-nad*a) is for the effect of *data points* or *integral points*. The number of data/integral points are *ndata* and *nada* is usually several times as much as *ma*.

According to the FORTRAN source code in Fig.1 (i), the matrix formation process is perfectly independent for each parameter point and data/integral point. This means that this operation can be done in very localized manner and suitable for parallel computing. Therefore, the parallelization of this process is rather straightforward.

2-way Parallelization

Fig.1 (ii) shows the *parallelized* source code for the equivalent part written in FORTRAN and MPI. Original code has been parallelized with very small changes from the original code except some subsidiary parameters/arrays and MPI messages.

Each processor stores coefficient matrix in *distributed* manner as *A(ma,maP)* where *maP=ma/P*, *P= Processor Number*. Each parameter point is renumbered in cyclic manner in order to avoid load imbalancing during LU factorization during parallel processing[4][5]. Basically inner 2 loops are almost identical with the original code.

As is mentioned in the previous subsection, the outermost loop corresponds to the operation on each data/integral point and the number of data point *ndata* is larger than that of parameter point *ma*. Therefore, operation and storage for data/operation points also should be localized in order to save memory and computation time. In Fig.1 (ii), the outer most loop is from 1 to *ndataP* (= *ndata/P*) and the operation is as follows :

(1) Effect from each data/integral point is calculated locally
(2) At ***call mpi_allgather*** message (underlined), value of *sig2imat*, *dymat* and *dydamat* calculated at each integral point are delivered to each processor
(3) Parallel processes are synchronized at this MPI message and proceed to the inner 2 loops for local matrix

Before/after the MPI message, perfectly different *space* is referred for parallel processing. This type of parallization method is very different from typical finite element type local opera-

tions such as in [6]. In this study, we call this type of processing as *2-way Parallelization*. If there are *N* kinds of different types of parameters, *N-way Parallelization* is required.

Results

Fig.2 shows the parallel performance for (105km X 45km) case with 705 parameter points and 6946 data/integral points using 12X Alpha Cluster connected through 100BT Ethernet. Results show perfect parallel performance and superlinear effect in many processor cases.

Further Study

(1) Larger scale problem for real plate boundaries including subduction plate models
(2) Fast multipole method and its parallelization
(3) Implementation of *iterative* solvers with robust preconditioning
(4) Vectorization and optimization on the *Earth Simulator* (GS40)

```
      do i= 1, ndata
        call FUNCS
        sig2i= 1.d0/(sig(i)**2); dy= y(i) - ymod
        do j= 1, ma
          wt= dyda(j)*sig2i
          do k= 1, ma
           A(j,k) = A(j,k) + wt*dyda(k)
          enddo
          B(j)= B(j) + dy*wt
        enddo
      enddo
```

(i) Original Code

```
do i0= 1, ndataP
  sig2imat= 0.d0 ; dymat= 0.d0 ; dydamat= 0.d0
  i= gfMTBL(i0)
  if (i.ne.0) then
    call FUNCS
    sig2imat(1)= 1.d0/(sig(i)**2); dymat(1)= y(i) - ymod
    do j= 1, ma
      dydamat(j)= dyda(j)
    enddo
  endif
  call MPI_ALLGATHER (sig2imat, dymat, dydamat…)
  do ip= 1, PETOT
    is= (ip-1)*ma
    do j= 1, ma
      wt= dydamat(is+j)*sig2imat(ip)
      do k= 1, maP
        k1 = gMTBL(k); gA(j,k)= gA(j,k) + wt*dydamat(is+k1)
      enddo
      gB(j)= gB(j) + dymat(ip)*wt
    enddo
  enddo
enddo
```

(ii) Parallel Code using MPI

Fig.1 Parallelization of the Matrix Assembling Part

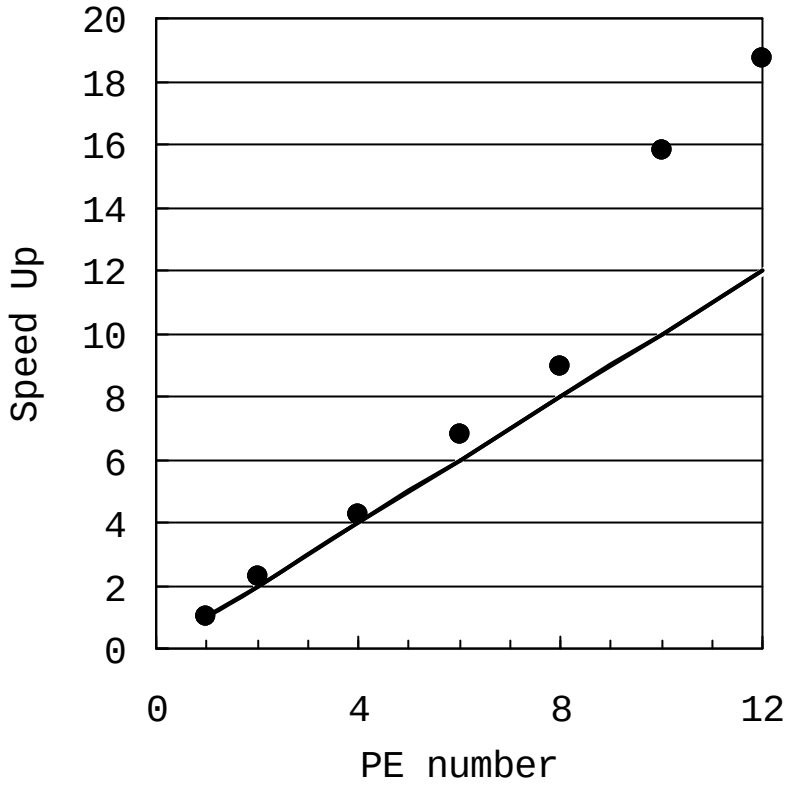

Fig.2 Parallel Performance for (105km X 45km) Region Case
705 parameter points and 6946 data/integral points using 12 Alpha Cluster connected through 100BT Ethernet(Line: Ideal, Circle: Computation Results)*

Acknowledgments

This work is a part of the "Solid Earth Platform for Large Scale Computation" project funded by "Special Promoting Funds of Science & Technology" of STA (Science and Technology Agency of Japan). Authors also thank to important suggestions on parallel computing by members of GeoFEM team.

References

[1] Hashimoto, C. and Matsu'ura, M., 1999, *Physical Modeling of Tectonic Loading Processes at Transcurrent Plate Boundaries*, 1st ACES Workshop Proceedings, 183-186.
[2] Gropp, W., Lusk, E. and Skjellum, A., 1994, *Using MPI : Portable Parallel Programming with the Message-Passage Interface*, MIT Press.
[3] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 1992, *Numerical Recipes in FORTRAN*, Cambridge University Press.
[4] Duff, I.S., Erisman, A.M. and Reid, J.K., 1986, *Direct Methods for Sparse Matrices*, Oxford Science Publications.
[5] LINPACK homepage http://www.netlib.org/linpack/index.html
[6] GeoFEM homepage http://geofem.tokyo.rist.or.jp/