6th International Symposium on Three-Dimensional Electromagnetics
Berkeley, California, USA, March 28-30, 2017

Development of a scalable and flexible computation strategy for large-scale Electromagnetic problems

Hao Dong[1*], Gary Egbert[2] and Naser Meqbel[3]
[1]China University of Geosciences, Beijing, China
[2]Oregon State University, Corvallis, United States
[3]Helmholtz Centre Potsdam GFZ German Research Centre for Geosciences, Potsdam, Germany
*donghao@cugb.edu.cn

SUMMARY

Thanks to the active development of numerical algorithms in the EM community, three dimensional (3D) electromagnetic modelling and inversion methods have been successfully applied to a wide range of geophysical problems at a range of scales, from the investigation of groundwater to the study of continental tectonics (e.g. [*Cox et al.*, 2010; *Schwarzbach et al.*, 2011; *Murphy and Egbert*, 2017]). However, widely available inversion programs such as ModEM or WSINV3DMT are not able to make effective use of advances in computer hardware and supercomputing technology over the last decade, imposing severe restrictions on grid resolution, and limiting our ability to explore model space. Here we discuss our recent efforts to implement scalable massively parallel computational approaches in the ModEM software.

Keywords: forward modelling, parallelization, scalable

INTRODUCTION

A common problem for many of the current 3D EM codes is the poor "strong-scaling" performance. The strong-scaling measurement indicates the efficiency of an algorithm for a given problem size when using an increasing numbers of parallel computing elements (processors). This is partly because most of these 3D forward/inversion algorithms still utilize relatively coarse-grained parallel techniques (e.g. [*Meqbel*, 2009; *Siripunvaraporn and Egbert*, 2009; *Kelbert et al.*, 2014]) A typical coarse-grained approach to parallelization in EM is parallelize over forward problems, e.g., solving for independent frequencies and source polarization simultaneously on different processors/CPUs. Apparently, the efficiency of this coarse-grained parallel scheme cannot be improved once the number of processors surpasses the number of independent forward problems. Hence a finer-grained parallel technique is needed in order to effectively utilize more processors.

On the other hand, while the increasing use of graphics processing unit (GPU) and many-core processor (MCP) provides a massive improvement of theoretical computing performance, it also challenges legacy codes with different instruction set architectures, making it impossible to make use of the new techniques before porting into a different platform (CUDA, OpenCL, etc.). Most current 3D forward/inversion parallel algorithms use a static load balance technique: all parallel tasks are evenly distributed to all processors regardless of the performance of each processor and the intensity of each task. As such the increasing heterogeneity of modern computing systems (e.g. CPUs are generally smaller in number but superior in performance comparing to GPUs) may actually limit the performance of such static schemes.

Here we describe development of a new general computing framework for ModEM, based on a two-layered MPI-PETSc parallel structure. A dynamic load balance scheme is designed to extend its performance. We also explore interactions of the parallelization strategy with forward problem formulation and numerical solver and preconditioner implementations. The scalability and versatility of the new strategy is demonstrated through the implementation of 3D magnetotelluric (MT) forward modelling.

PETSc libary

Portable Extensible Toolkit for Scientific Computation (PETSc) is a suite of libraries and routines developed by Argonne National Laboratory for the (parallel) solution of partial differential equation (PDE) problems and sparse matrix computations [*Balay et al.*, 2014]. PETSc offers access to a series of scalable linear/non-linear solvers and a simple interface to utilize heterogeneous systems with GPUs (support of MCPs is under development). Furthermore, as a third party library, PETSc is under active development and is continuously integrated with new advances of computational science, allowing users from different fields to focus on their own area of expertise.

As the core of EM forward/inversion is the PDE problem, which involves solving a number of large (sparse) systems of linear equations, the efforts of developing

numerical EM codes can be reduced by utilizing the various linear solvers (along with preconditioners) from PETSc. More importantly, PETSc provides distributed data type objects like Vector (Vec) and Matrix (Mat), which can be effectively implemented for finer grained partitioning parallelization methods as well as spatial domain decomposition.

Taking a basic sparse matrix-vector multiplication for example, generally PETSc distributes matrices by dividing them into diagonal and off-diagonal parts, which are stored like "row blocks" in each process (Figure. 1). Corresponding to the diagonal sub-matrix, a section of the input vector is also stored locally by the same process.

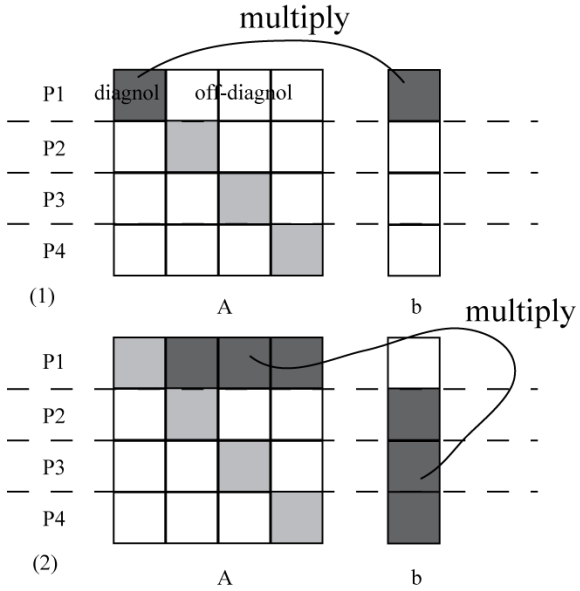

Figure 1. sparse matrix-vector multiplication using PETSc data structure, modified from [*Lange et al.*, 2013]

A two-step computation strategy of the matrix-vector multiplication can be effectively implemented for the storage structure, as shown in Figure 1:

1) Diagonal sub-matrix elements are multiplied with the local part of the vector, while remote vector elements are gathered from other processes and saved as a local copy.
2) Off-diagonal sub-matrix is then multiplied with the local copy of the gathered vector elements and added to the partial solution.

Similar optimizations are commonly used in PETSc library, providing a simple way to reduce the process granularity and to reduce the communication overhead.

Parallelization Strategy

With the fast iteration of multi-core processors and distributed memory clusters, more and more parallel systems are taking the strategy of two-level parallelism [*Rabenseifner*, 2003]:

1) inter-node level: Between the compute nodes, generally the exchange of parallel information is explicitly implemented through communications (by Ethernet, InfiniBand, etc.) between nodes.
2) intra-node level: inside a node, processors share the same memory space, allowing them to exchange information much more efficiently by directly manipulating the common memory space.

A popular way to build a two-level algorithm is the so-called "hybrid" MPI/OpenMP parallelization: Using message passing interface (MPI) for the inter-node level and shared memory interface of OpenMP for the intra-node level [*Drosinos and Koziris*, 2004]. However, the implementation of a hybrid MPI/OpenMP model is relatively complex and requires a high level of expertise and the understanding of both interfaces. Also, the speedup of a hybrid approach can be hindered by the increased communication overheads from the two different interfaces.

Therefore we design a two layered MPI-PETSc parallel structure with a common message passing interface. Figure 2 shows a chart that illustrates the workflow of the structure on a simple computational task, like MT forward modelling.

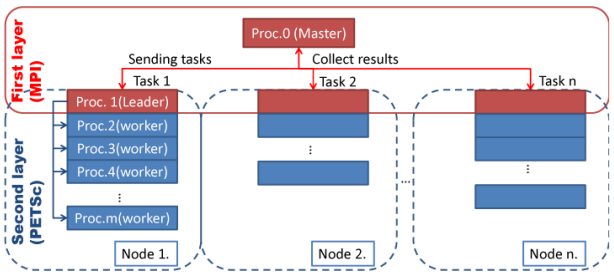

Figure 2. sparse matrix-vector multiplication using PETSc data structure, modified from

As is shown in Figure 2, the first layer follows the previous driving structure of MPI based one-layer parallelization in ModEM. Firstly, the "master" process divides all available "worker" processes into a number of groups that equals the number of "coarser" independent computational tasks. Preferably, one group only contains processes from a common shared memory node. The master then sends these tasks to the "leaders" of each group and then waits for the collective communication to gather the result.

The second layer feed the MPI communicator of the first layer to the PETSc API. The leader of each group then formulates the PETSc distributed data structure on the processes of the group and starts finer-grained

parallelized tasks using PETSc solvers. The leader then collects and packs the task result on the finish of the calculation and initializes the collective communication for the harvest of results.

In this case, the old ModEM parallel code can be best re-used without making fundamental changes to the existed utility routines like distributing model, receiver and transmitter information. The framework of the basic operations, like forward response and sensitivity calculation, can also be maintained by substituting the previous sequential solvers with parallel solvers from PETSc library.

Dynamic Load Balance

As with the two-layered structure, the load balance problem of the parallelization strategy described above also involves two levels: balance among independent (coarser) tasks and balance among the (finer) tasks regarding to different portions of distributed data.

Traditional static load balance scheme would evenly distribute tasks to processes on either level. For the first layer, while the system matrices from the independent tasks are often identical, different right hand sides may result in rather contrasting solve time. For the second layer, individual rows from a matrix may also involve various amount of computational time. Both cases may result in potential load imbalance among or within individual process groups.

Apparently, a more dynamic load balance approach requires the estimation of both the computational performance of each processor/node and the intensity of each computational task. While the performance (or at least on-sheet performance) of processors can always be determined a-priori, the intensity of tasks seems less obvious to determine.

For lower level tasks like solving sparse linear systems, a simple (and less resource demanding) scale would be the number of non-zeros (density) in different row blocks [*Williams et al.*, 2007]. Each process is distributed with a block of continuous rows with a similar amount of non-zeros. This also help balances the memory bandwidth load. For upper level tasks, a post-priori method seems to be appropriate: computational power is initially evenly distributed to different tasks; the number of processes in each group is then adjusted according to the runtime of previous iteration/task (i.e. jobs took considerably more time in the last iteration will receive more processing power, and vice versa).

Implementation in ModEM

In ModEM the forward equations are formulated as a curl-curl equation in the electric field, using a matrix-free approach. A block D-ILU preconditioner is used, along with a divergence correction. To make better use of PETSc capabilities we are completely rewriting the forward code making use of more standard sparse matrix operations.

We are simultaneously experimenting with alternative formulations of the forward problem. One approach is to modify the curl-curl equation by adding $\nabla\nabla\cdot$ to the equations in the air, and $\sigma\nabla\nabla\cdot\sigma$ in the Earth. These modifications effectively enforce appropriate divergence-free conditions (electric fields in the air, current in the conductive Earth). Alternative solution strategies, such as multi-grid, are also under investigation. The choice of solution approach will certainly have implications for design of the parallelization, so we intend to explore these different aspects of the forward solver in tandem.

ACKNOWLEDGMENTS

HD acknowledges the support from Natural Science Funds of China for Young Scholars (41504062). GDE acknowledges support from NSF grant ER-125496.

REFERENCES

Balay, S., J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. Smith, and H. Zhang (2014), *PETSc Users Manual Revision 3.4*, Argonne, IL (United States).

Cox, L. H., G. A. Wilson, and M. S. Zhdanov (2010), 3D inversion of airborne electromagnetic data using a moving footprint, *Explor. Geophys.*, *41*(4), 250, doi:10.1071/EG10003.

Drosinos, N., and N. Koziris (2004), Performance comparison of pure MPI vs hybrid MPI-OpenMP parallelization models on SMP clusters, in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, pp. 15–24, IEEE.

Kelbert, A., N. Meqbel, G. D. Egbert, and K. Tandon (2014), ModEM: A modular system for inversion of electromagnetic geophysical data, *Comput. Geosci.*, *66*, 40–53, doi:10.1016/j.cageo.2014.01.010.

Lange, M., G. Gorman, M. Weiland, L. Mitchell, and J. Southern (2013), Achieving Efficient Strong Scaling with PETSc Using Hybrid MPI/OpenMP Optimisation, in *Supercomputing*, edited by J. M. Kunkel, T. Ludwig, and H. W. Meuer, pp. 97–108, Springer, Berlin Heidelberg.

Meqbel, N. (2009), The electrical conductivity structure of the Dead Sea Basin derived from 2D and 3D inversion of magnetotelluric data, Freie Universitat Berlin.

Murphy, B. S., and G. D. Egbert (2017), Electrical conductivity structure of southeastern North America: Implications for lithospheric architecture

and Appalachian topographic rejuvenation, *Earth Planet. Sci. Lett.*, *462*, 66–75, doi:10.1016/j.epsl.2017.01.009.

Rabenseifner, R. (2003), Hybrid Parallel Programming on HPC Platforms, in *5th European Workshop on OpenMP*, pp. 185–194.

Schwarzbach, C., R.-U. Börner, and K. Spitzer (2011), Three-dimensional adaptive higher order finite element simulation for geo-electromagnetics-a marine CSEM example, *Geophys. J. Int.*, *187*(1), 63–74, doi:10.1111/j.1365-246X.2011.05127.x.

Siripunvaraporn, W., and G. Egbert (2009), WSINV3DMT: Vertical magnetic field transfer function inversion and parallel implementation, *Phys. Earth Planet. Inter.*, *173*(3–4), 317–329, doi:10.1016/j.pepi.2009.01.013.

Williams, S., L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel (2007), Optimization of sparse matrix-vector multiplication on emerging multicore platforms, in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing - SC '07*, p. 1, ACM Press, New York, New York, USA.