Rapid 3D Seismic Source Inversion Using Windows Azure and Amazon EC2 *

Vedaprakash Subramanian, Hongyi Ma,
and Liqiang Wang
Department of Computer Science
University of Wyoming
{vsubrama, hma3, wang}@cs.uwyo.edu

En-Jui Lee and Po Chen
Department of Geology and Geophysics
University of Wyoming
{elee8, pchen}@uwyo.edu

Abstract

With its rapid development, cloud computing has been increasingly adopted by scientists for large-scale scientific computation. Compared to the traditional computing platforms such as cluster and supercomputer, cloud computing is more elastic in the support of real-time computation and more powerful in the management of large-scale datasets. This paper presents our experience on designing and implementing seismic source inversion on both cluster (specifically, MPI-based) and cloud computing (specifically, Amazon EC2 and Microsoft Windows Azure). Our experiment shows that applying cloud computing to seismic source inversion is feasible and has its advantages. In addition, we notice that both cluster and Amazon EC2 have obviously better performance than Windows Azure. Cloud computing is suited for real-time processing scientific applications but it (especially Azure) does not work well for tightly-coupled applications.

1 Introduction

Traditionally, High Performance Computing (HPC) has been widely used for scientific computing. However HPC is usually based on batch processing, which makes on-demand computing difficult to undertake. In general, cloud computing provides an attractive new platform compared to HPC, where virtual machines can be targeted to specific scenarios and are more elastic for on-demand computation. In addition, cloud computing provides convenient access to reliable, high-speed, and extreme scale storage on demand. Cloud computing is increasingly used as an alternative to the traditional computational platforms. This paper presents our experience on designing and implementing seismic source inversion on both cluster (specifically, MPI-based) and cloud computing (specifically, Amazon EC2 and Microsoft Windows Azure).

Seismic source inversion is a central problem in seismology. In the earthquake-prone areas such as Southern California, the accurate representation of earthquake sources is important for both reliable seismic hazard analysis and interpolation of geologic and tectonic structures. Recent advances in computer science allow seismologists to simulate wave propagation in complex three dimensional (3D) velocity models. From the inversion point of view, simulation of wave propagation within 3D velocity models will reduce errors from the true velocity structures and resolve source parameter estimations more accurately. The synthetic seismograms generated based on 3D velocity structure have much better representations than those generated based on 1D multiple layer model (*e.g.*[9]). By using 3D synthetic seismograms in seismic source inversion, the results usually have more accurate source parameters, location and magnitude estimations than the results based on 1D synthetic seismogram [6]. Figure 1 shows the workflow for our seismic source inversion. First, we generate synthetic seismograms based on 3D velocity model of our study areas around the initial source location. We start with coarse intervals of source parameters and then refine the ranges with higher waveform similarity iteratively to find the optimal source solution.

Recently, Subramanian et al. [9] implemented an algorithm on Microsoft Windows Azure to generate synthetic seismograms in 3D velocity model. This paper is the continuation of our previous work [9] implementing the algorithm to perform 3D seismic source inversion. The major differences between this work and our previous work [9] are: (1) The previous work takes the earthquake source parameters, its magnitude and location as input and generates synthetic seismogram based on 3D velocity model, whereas this work takes the 1D model seismograms as input and estimates the earthquake source parameters, its magnitude, and location on 3D model. (2) The previous work involves large amount of datasets, say around 100GB - 1TB for Southern California region, whereas this work involves small amount

*This work was supported in part by NSF under Grant 0941735

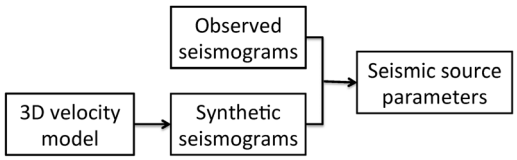

Figure 1. The workflow of our seismic source inversion.

of datasets, say around 300MB - 500MB. (3) The computation of the previous work is loosely-coupled *i.e.* it requires few communication between the worker roles, whereas the computation of this work is tightly-coupled. Hence it requires lots of communication between the worker roles.

Real-time processing is the critical feature of seismic source inversion. When executing such an application on the traditional supercomputer, it is often that the submitted jobs wait for a few minutes or even hours before being scheduled. Although a dedicated computing cluster might be able to make a nearly real-time response, it is not elastic; which means that the response time for different requests may vary significantly. Since earthquakes often happen in a burst within a short time, both traditional supercomputer and dedicated cluster are incapable of supporting real-time processing. Due to the elastic nature of the cloud computing platform, it is an ideal platform for our application, which provides much faster response time and means to scale up and down according to the requests.

We implemented two versions of seismic source inversion: one in C language and MPI for executing on regular cluster and Amazon EC2, the other in C# for executing on Windows Azure. We evaluated our implementation on cluster computing platform (specifically, Argonne National Labs Intrepid, *i.e.*, IBM Blue Gene/P, which is actually a supercomputer), Amazon EC2, and Microsoft Windows Azure. The experiment shows that applying cloud computing to seismic source inversion is feasible and has its advantages. The performance on EC2 and cluster/supercomputer are comparable, whereas Azure has a significant slowdown on performance probably because of its .NET platform and inter-node communication.

This paper makes the following contributions: (1) To the best of our knowledge, it is the first application on seismic source inversion using cloud computing. (2) We evaluated our current application on cluster, Amazon EC2, and Windows Azure and compared the performance.

The rest of the paper is organized as follows. Section 2 compares cluster computing and cloud computing. Section 3 introduces our implementations and the optimization techniques. Section 4 shows the experimental results on cluster computing and cloud computing platforms (Amazon EC2 and Windows Azure). Section 5 presents the related work. Finally, we give the conclusions and our future work in Section 6.

2 Cluster and Cloud computing

Clusters are the group of computers connected by high speed LAN such as Infiniband, and Gigabit Ethernet. These computers usually work together to solve a single computational problem. MPI (Message Passing Interface) is the most widely used library that enables the communication of parallel programs among compute nodes. Cluster system is designed for collaborative sharing of resources. Compared to cluster computing, cloud computing provides more elastic virtual machines and data storage on-demand. The cluster and cloud have the following differences in the aspect of scientific computing: (1) *Virtualization*: As programs run directly on cluster computers, cloud utilizes virtualized machines, which makes computing more portable, fault-tolerant, and supports automatic dynamic load balancing. Certainly virtualization may also affect the performance. (2) *Elastic on-demand resource:* the cloud computing resources are pooled through multi-tenant model to various consumers on-demand; the resources in cluster computing are relatively fixed. (3) *Cost-effective:* Instead of buying machines to solve a computational problem, it is cheaper to rent the machine. (4) *Network communication:* cluster usually uses the dedicated high-speed local area network technology (*e.g.*Gigabit Ethernet or Infiniband), whereas nodes on the cloud share network computation, which may slow down the inter-node communication. Hence, many research have shown that cloud computing platforms (*e.g.*Amazon EC2) have a slightly slower performance than the cluster. Nevertheless, cloud computing has its own advantages and disadvantages over the cluster.

3 Implementation

3.1 Implementation on Traditional Cluster Computing and Amazon EC2

The seismic source inversion for cluster computing is implemented in C language using MPI. The algorithm is shown in Figure 2. In our implementation, we partition the matrix of seismic source inversion by the index, where each column corresponds to a seismogram window and each row corresponds to one of the three angle(dip, rake and strike), and scatter it among all the processors. Thus each processor works on its column. At the end of the each iteration, the results from all processes in the current iteration are gathered on the root node and sorted. Finally, the seed is chosen for the next iteration and broadcasted. The entire computation contains 6 iterations (*i.e.*, `Iter_times`). So, at the end of the 6th iteration, we get the final solution matrix.

```
for (curr_Iter = 0; curr_Iter < Iter_times; curr_Iter++) {
   if (curr_Iter == 0)
      Initialize the values for computation;
   else
      Initialize the values for computation through the seeds
         from the previous iteration;
   Broadcast these values using MPI_Bcast;
   Partition the matrix by columns;
   Distribute among the processors;
   Perform the computation;
   Gather data on the root node using MPI_Gather;
   Sort the result;
   Determine the seed;
} // the end of for loop
```

Figure 2. The algorithm of our implementation on cluster.

```
for ( curr_Iter=0; curr_Iter < Iter_times; curr_Iter++) {
   if (Job Manager) {
      if (curr_Iter == 0)
         Initialize the values for computation;
      else
         Initialize the values for computation through the seeds
            from the previous iteration;
      Partition the matrix by columns;
      Distribute among worker roles using queue message;
   } else {
      wait till the message the recieved from the Job Manager;
   }
   Retrieve the computation from the queue message;
   Use TPL and performance the computation;
   Store the results in either as queue message or blob;
   Signal the Job Manager using queue message;
   if (Job Manager){
      Retrieve the queue message;
      if (queue message contains the blob name)
         Retrieve the data from the given blob;
      Update the result matrix;
      Sort the matrix;
      Determine the seed;
   }
} // end of for loop
```

Figure 3. The algorithm implemented on Windows Azure.

We ported the C code with MPI onto Amazon EC2 cloud. We used virtualized linux machines that had the same computing environment as that of clusters. The porting of the code from cluster to Amazon EC2 was straightforward.

3.2 Implementation on Windows Azure

The architecture of our implementation on Windows Azure is similar to the architecture used in our previous work [9]. Our implementation utilizes the following components: (1) Web role, which acts as a user interface. (2) Worker role, which does the computation. (3) Job manager, which coordinates the work among the instances of the computation worker role, gathers and sorts the results, determines the seed for the next iteration, and sends it to all worker roles. Moreover, Job Manager does a part of computation. (4) Azure queue and blobs, which act as communication interfaces between the web role and the worker role. The algorithm is shown in Figure 3.

The computation of our application is tightly coupled. The input for each iteration depends on the output of the previous iteration. Currently, unlike MPI, there is no easy way to broadcast and gather datasets on the Azure. We utilize queues and blobs for communication. Since the size of queue message is limited, we used blob whenever the size of the dataset exceeds the limit. An alternative is to use Windows Azure Appfabric service bus; we did not implement it in our application.

Our program is implemented based on two level parallelism, (1) *Multiple worker roles:* This is achieved by using multiple worker roles for the computation, and Job manager to cordinate the computation. (2) *Multiple threads in the same worker role:* This is achieved by using Task Parallel Library (TPL) to parallelize the computation among multiple cores. It contributes to significant reduction in the execution time[9]. Moreover, the program uses the local storage provided by each worker role. Since the datasets for the application is relatively small. We allocated local storage on each worker role of size 2GB and while initializing the worker role, the program pulls all the data from the cloud storage and makes the copy of the data on its local storage. By this method, we exploited the local storage as well as prevented the latency delay due to accessing the files in the cloud storage through the network.

4 Experiment

We evaluated the performance by measuring the execution time of our implementation for seismic source inversion on Argonne National Labs Intrepid (IBM Blue Gene/P), Amazon EC2, and Windows Azure. All experiments are based on the same dataset.

Figure 4 shows our experiement on Intrepid. The experiment shows that with the increase of number of processors, the execution time is decreased rapidly at the beginning, but then the communication time dominates, thus the performances at 500 processes and 200 processes do not have much difference.

Figure 5 shows our experiment on Amazon EC2, where we utilize the Amazon Machine Image(AMI) “ami-c17f9ca8” where openMPI is installed. Three instance types are used in our experiment: Large (“m1.large”, where

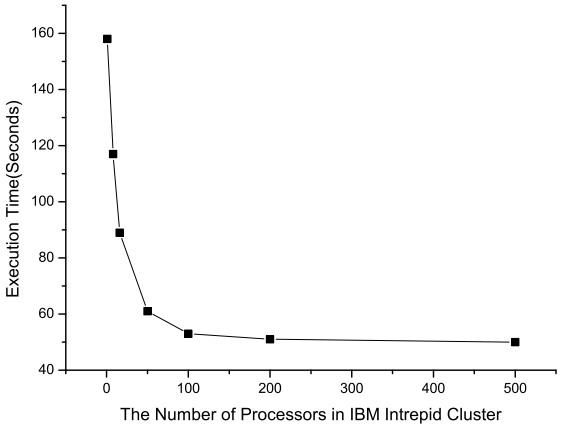

Figure 4. The performance measurement on Argonne National Labs Intrepid (IBM Blue Gene/P).

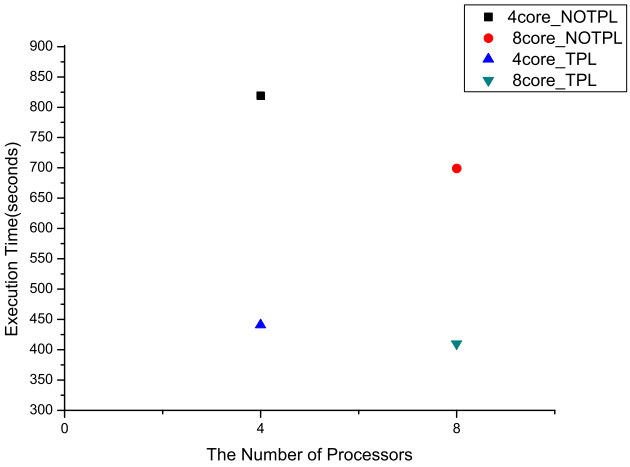

Figure 6. The performance measurement on Windows Azure.

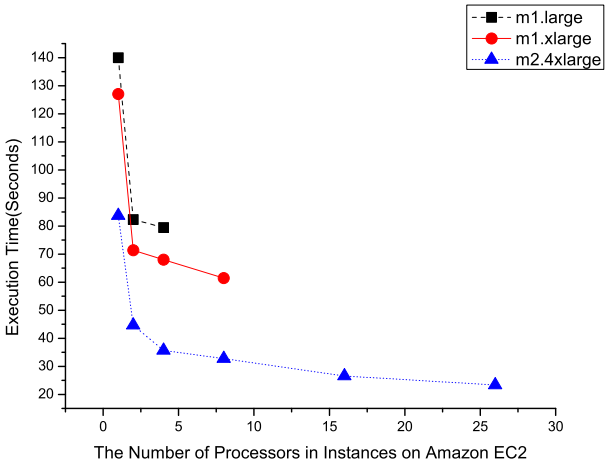

Figure 5. The performance measurement on Amazon EC2 using different instance types.

CPU cores are 2 and memory is 7.5 GB), Extra Large ("m1.xlarge", where CPU cores are 4, and memory 15 GB), High Memory Quadruple Extra Large ("m2.4xlarge", CPU cores are 8, and memory is 68.4 GB).

The experiment shows that the execution time decreases with the increase in the number of processors. Also, different types of instances have different performance. The instance with more CPU cores and larger memory has the better performance. Our application runs faster on EC2 because its CPU is 2.27 GHz, compared to the CPU on Intrepid is 850 MHz.

Figure 6 shows our experiment on Windows Azure. We compared the performance of Azure based on the number of worker roles and the number of cores on each worker role. The CPU speed of each worker role is 1.6 GHz. The experiment shows that the performance improves with the increase of the number of worker roles as well as with usage of all the CPU cores using TPL. But when comparing with the performance of Amazon EC2 and Intrepid (IBM Blue Gene/P), the performance of Azure is not good because of the .NET platform and inter-node communication on Azure.

5 Related Work

Traditionally, seismic wave processing utilizes cluster and grid computing. CyberShake [7] is a scientific workflow on grid computing which is used for Probabilistic Seismic Hazard Analysis (PSHA). CyberShake has been executed on grid-based computing environment at the Southern California Earthquake Center (SCEC) . Their analysis shows that the grid-based environment is an ideal option for CyberShake workflows and its data management. However, the grid-based environment will have its limit when the computational demands increase.

Applying cloud computing to seismic processing is a relatively new research area. Juve et al. study the performance of Amazon EC2 cloud for a memory-intensive seismic application [5]. The experiment shows that the performance of the cloud is nearly the same to that of NCSA's Abe cluster [8], a typical high performance computing (HPC) system.

Cloud computing has been widely used to execute scientific workflows. Hoffa et al. [4] apply cloud computing to a widely used astronomy application-Montage. According to their experiment, the virtual environment can sustain good compute time but the whole execution time can suffer from resource scheduling delays and wide area communications. We [1] implement a high performance workflow system called MRGIS based on MapReduce cloud computing platform to execute GIS applications efficiently. The experiment demonstrates that MRGIS can significantly improve the performance of GIS workflow execution. Vecchiola et al. [10] study the role of cloud in scientific computing. As an example of scientific computing on cloud, a preliminary case study on using Aneka [2] is presented for the classification of gene expression data and the execution of fMRI brain imaging workflow. To compare the performance of cloud (*e.g.*Amazon EC2) with a dedicated HPC system, He et al. [3] show that the virtualization technology adds a little performance overhead and the poor network-capabilities of the public cloud decreases the performance of application. But clouds with better network-capabilities may improve the performance of HPC applications.

6 Conclusions and Future Work

This paper presents our implementation of 3D seismic source inversion algorithm on traditional cluster, Amazon EC2 and Windows Azure, and comparison of the performance. The experiment shows that applying cloud computing to seismic source inversion is feasible and has its advantages. Based on our experiments, we found that cloud computing is suited for real-time processing applications but it (especially Azure) does not work well for tightly-coupled applications. In the future, we will combine our previous application of synthetic seismograms generation with the 3D seismic source inversion. By using 3D synthetic seismograms in seismic source inversion, the results usually have more accurate source parameters, location and magnitude estimations than the results based on 1D synthetic seismograms.

References

[1] Q. Chen, L. Wang, and Z. Shang. MRGIS: A MapReduce-Enabled high performance workflow system for GIS. In *the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES)*. IEEE Press, december 2008.

[2] X. Chu, K. Nadiminti, C. Jin, S. Venugopal, and R. Buyya. Aneka: Next-generation enterprise grid platform for e-science and e-business applications. In *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 151–159, Washington, DC, USA, 2007. IEEE Computer Society.

[3] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn. Case study for running hpc applications in public clouds. In *the 1st Workshop on Scientific Cloud Computing (ScienceCloud 2010)*. ACM, 2010.

[4] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the use of cloud computing for scientific workflows. In *the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES)*, pages 640–645, Washington, DC, USA, 2008. IEEE Computer Society.

[5] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Scientific workflow applications on amazon EC2. In *Workshop on Cloud-based Services and Applications in conjunction with 5th IEEE Internation Conference on e-Science (e-Science 2009)*. IEEE Press, 2009.

[6] E.-J. L. Lee, P. Chen, T. H. Jordan, and L. Wang. Rapid centroid moment tensor (cmt) inversion in a three-dimensional earth structure model for earthquakes in southern california. In *Geophysical Journal International*. Geophysical Journal, 2011.

[7] P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, H. Francoeur, V. Gupta, Y. Cui, K. Vahi, T. Jordan, and E. Field. SCEC cybershake workflows automating probabilistic seismic hazard analysis calculations. In *Workflows for e-Science*, pages 143–163. Springer-Verlag, 2007.

[8] National Center for Supercomputing Applications. http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/Intel64Cluster/.

[9] V. Subramanian, L. Wang, E.-J. Lee, and P. Chen. Rapid processing of synthetic seismograms using Windows Azure cloud. In *In the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2010)*, Indianapolis, USA, 2010. IEEE Computer Society.

[10] C. Vecchiola, S. Pandey, and R. Buyya. High-performance cloud computing: A view of scientific applications. In *ISPAN '09: Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pages 4–16. IEEE Computer Society, 2009.