Rapid Processing of Synthetic Seismograms Using Windows Azure Cloud*

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

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

Abstract

Currently, numerically simulated synthetic seismograms are widely used by seismologists for seismological inferences. The generation of these synthetic seismograms requires large amount of computing resources, and the maintenance of these observed seismograms requires massive storage. Traditional high-performance computing platforms is inefficient to handle these applications because rapid computations are needed and large-scale datasets should be maintained. The emerging cloud computing platform provides an efficient substitute. In this paper, we introduce our experience on implementing a computational platform for rapidly computing and delivering synthetic seismograms on Windows Azure. Our experiment shows that cloud computing is an ideal platform for such kind of applications.

1 Introduction

Seismic waves generated by natural and/or manmade seismic sources propagate through the interior of the Earth and cause motions of the ground. A seismogram is a record of the ground shaking recorded by a seismometer. Proper interpretation of seismograms allows seismologists to map the Earth's internal structures, and locate and measure the size of different seismic sources. Since the emergence of modern instruments using electronic sensors, amplifiers and recording devices in the early 1900s, seismograms have been systematically collected and archived at seismic data centers distributed around the world. An important example of seismic data centers is the Incorporated Research Institutions for Seismology (IRIS) [1] in the United States. Founded in 1984 with support from NSF, it now collects and archives seismograms from a network of hundreds of seismometers around the world and plays an essential role in scientific investigations of seismic sources and Earth properties worldwide.

Recent advances in computational technology and numerical methods have opened up the possibility to simulate the generation and propagation of seismic waves in three-dimensional complex geological media by explicitly solving the seismic wave-equation using numerical techniques such as finite-difference, finite-element, and spectral-element methods, thereby allowing seismologists to incorporate numerically simulated seismograms (*i.e.*, synthetic seismograms) into their research. The capability of conducting simulation-based predictions has initiated an inference cycle in which the simulation-based predictions (*i.e.*, synthetic seismograms) are validated against actual observations (*i.e.*, real observed seismograms collected and archived at seismic data centers). Where the seismological models (*i.e.*, the geological models of the Earth's interior and/or the rupture models of the seismic sources) are deficient, data assimilation techniques commonly used in atmospheric sciences and oceanography are adopted to improve the seismological model and reinitiate the inference cycle at a higher level.

The purpose of this study is to establish a cloud-based computational platform for rapidly computing and delivering synthetic seismograms to seismologists worldwide to facilitate simulation-based seismological inferences. This platform allows seismologists to download synthetic seismograms through their web browsers from our cloud-based data collection for any types of seismic source models in a manner similar to downloading observed seismograms from various seismic data centers such as IRIS. Such data centers are usually implemented using large-scale clusters. Although the dedicated clusters are desirable, they are cost-ineffective. In addition, large-scale high-performance computing systems shared by multiple users, such as TeraGrid, are inappropriate, where the applications often suffer from long delay on acquiring computation resources. Cloud computing is an ideal emerging platform for rapidly computing and delivering synthetic seismograms, as it allows users to acquire and release resources on-demand with very low scheduling overhead.

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

The underlying theoretical foundation for our platform is the "reciprocity principle" in seismology, which basically states that the seismogram generated by a seismic source and recorded at a seismometer is the same if we exchange the locations of the seismometer and the source. By using the seismometers as virtual sources and conducting wave-propagation simulations in realistic three-dimensional Earth models, we have established a database for Southern California, an earthquake-prone region with high seismic risk. Synthetic seismograms for any types of earthquakes in Southern California can now be generated on the fly based on user queries and delivered in real-time. We expect that our system will be useful for seismological research in general and seismic hazard assessment in particular.

Our system for analyzing synthetic seismograms is based on Microsoft Windows Azure [3]. This paper makes the following contributions: (1) To the best of our knowledge, it is the first system that utilizes cloud computing for such kind of applications. (2) Several innovative optimization techniques for performance and storage are designed and implemented on Windows Azure. We use the concept of multi-threading and .NET Task Parallel Library to decrease the total computation time. We develop an innovative way to store the data corresponding to the latitude and longitude. A data query algorithm is designed to speed up the searching.

The rest of the paper is organized as follows. Section 2 introduces cloud computing and Windows Azure. Section 3 discusses the general workflow of the application. Section 4 presents the design and implementation details of our system. The experiments are presented in Section 5. Sections 6 and 7 give related work and conclusions, respectively.

2 Cloud Computing and Windows Azure

2.1 Cloud Computing

Recently, cloud computing has increasingly gained attention, as it provides a flexible, on-demand computing infrastructure. According to the definition given by the National Institute of Standards and Technology (NIST) [14], cloud computing has the following essential characteristics: (1) *On-demand self-service*: the computing resources are requested on demand without interaction with the service provider. (2) *Broad network access*: resources are available over Internet and can accessed by any platforms. (3) *Resource pooling*: the computing resources are pooled by the provider through multi-tenant model to various consumers on demand. (4) *Rapid elasticity*: capability can be quickly scaled in and out. (5) *Measured Service*: Resource usage can be monitored, then transparently controlled.

Based on the aforementioned features of cloud, it will be an ideal platform to deploy the system to compute and deliver synthetic seismograms with rapid and scalable performance. It is a cost-effective substitute for the traditional dedicated computing cluster or grid.

2.2 Windows Azure

The Windows Azure [3] is a Platform as a Service (PaaS) running in Microsoft data centers. The platform consists of a highly scalable (elastic) cloud operating system and a data storage system. The services are supported by physical or logical (virtualized) Windows Server instances. On Windows Azure, users can deploy their applications (created using Visual studio IDE and Windows Azure SDK) onto the cloud infrastructure but do not have to manage the underlying cloud infrastructures such as network, servers, operating system, and storage. Users have control over the application and its environment configurations. This helps us focus on the application rather than manage the cloud infrastructure. Windows Azure platform comes with a geo-locate feature which enables selecting the data center on which the service is provided. Since our application focuses on maintaining datasets for Southern California, we choose the geo-location on Azure to be the Central America which has better performance and bandwidth towards all the user queries. Moreover, Context delivery Network (CDN) feature enables caching the blobs across the 20 Azure CDN locations in order to provide better bandwidth and performance.

Azure service [10] consists of two major roles, namely web role and worker role, as virtual machines (VM). A *web role* is customized for web application and acts as a user interface which responds to the user input. A *worker role* is for generalized development, and may perform background processing for a web role. These roles execute on Microsoft .NET framework.

Azure provides four kinds of data structures for designing cloud applications: *blob*, *table*, *queue*, and *drive*.

Azure Blob provides a simple interface for storing files along with metadata. A blob container groups a set of blobs. A storage account can have multiple containers. Sharing is based on the container level, *i.e.*, a container can be set to private or public. When a container is `public`, all its contents can be read by anyone without requiring authentication. When a container is `private`, authentication is required to access the blobs in that container. Containers can have metadata associated with them. Metadata is in the form of <name, value> pair and can be up to 8KB per container. The blob metadata can be set and retrieved separately from the blob data. Azure supports two types of blobs, namely block blobs and page blobs. Each block blob consists of a sequence/list of blocks, which can size up to 4MB and the block blob can size up to 200GB. The block blob is mainly targeted at streaming workloads. Page

blob also consists of a sequence/list of pages, where each page is in fixed size (512 bytes), so all writes must be 512 byte aligned. The page blob is mainly targeted at random write workloads. Since our system's datasets remain constant throughout the service and can be accessed in stream based on user queries, hence we use block blobs to store datasets.

Azure Table contains a set of entities. An application may create multiple tables within a storage account. Each entity can hold up to 255 properties, within which there are two basic properties, *partition key* and *row key*, as the unique key for the entity. We use the partition key to automatically distribute and balance the load of the table's entities over many servers. The row key is the unique identifier of the entity within the partition it belongs to. Every entity has a version maintained by the system for optimistic concurrency. A single index system is used for Azure tables, where all entities in a table are sorted by the partition key and then the row key. Azure table does not follow any naming schema, so all of the properties are stored as <name, typed value> pairs. Thus entities in the same table can have very different properties. In our application, the table storage is used for maintaining a list of geographic regions for querying seismograms.

Azure Queue is used to store messages or small datasets. The queue name is scoped inside the storage account. There is no limit on the number of messages stored in a queue (the only limit is the 100TB size limit on a storage account). A message is stored for at most a week. Queues can also have metadata associated with them, which is in the form of <name, value> pair. Messages in queues are relatively small in size (up to 8KB). To store larger datasets, one can store the data in Azure Blob or Azure Table, and then store the blob/entity name in a message. In our application, the queues are used for communication between the roles.

Azure Drive acts as a local NTFS volume that is mounted on the server's file system and is accessible to code running in any role (web or worker). The data written to a Azure drive is stored in a page blob defined within the Windows Azure Blob service, and cached on the local file system. The size of the drive ranges from 16MB to 1TB. Drives can be uploaded or downloaded via the Windows Azure Blob interface. Since the data written to the drive are stored in a page blob, the data are kept even if the role instance is recycled (*i.e.*, deleted and restarted). For this reason, Azure drive can be used to run an application requiring non-volatile state, such as a third-party database application. Our system does not use Azure drives.

3 Introduction to the Application

The application starts with a web site which acts as a user interface where users can request the generation of synthetic seismograms based on 3D velocity model by entering locations of earthquakes, stations, and earthquake source parameters, such as longitude, latitude, and depth for earthquake location and strike, dip and rake for source parameters. To increase the efficiency of generating synthetic seismograms, we store the receiver green tensors (RGTs), the strain fields generated by three orthogonal unit impulsive point forces acting at the receiver locations, for the 3D velocity model [17]. By applying the reciprocity principle [2], it can be shown that the RGTs provide exact Frchet derivatives of the seismograms at the receiver locations with respect to the moment tensor at any point in the modeling volume. The synthetic waveforms based on 3D velocity model are usually more accurate than synthetic waveforms based on 1D multiple layers velocity model. Figure 1 shows waveform comparisons of the Chino Hills earthquake on 29 July 2008 with M_w 5.4 among observed seismograms, synthetic seismograms based on updated 3D velocity model,

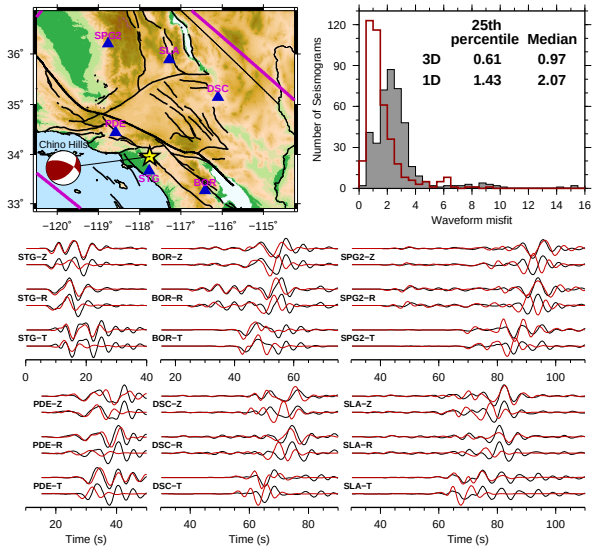

Figure 1. Waveform comparison for the 2008 Chino Hills earthquake at 153 stations with good signal-to-noise ratios. In the box, the histograms show the distributions of waveform misfits of SoCaL model (gray area) and CVM4SI2 (area included by red line). Black solid lines: observed seismograms; red solid lines: synthetic seismograms calculated using the finite-difference code. In each pair, the red line above is synthetic seismogram computed using CVM4SI2 and the red line below is the one computed using SoCaL. Star: epicenter of the earthquake; blue triangles: station locations of waveform comparison examples. The beachball shows the focal mechanism used for computing the synthetics.

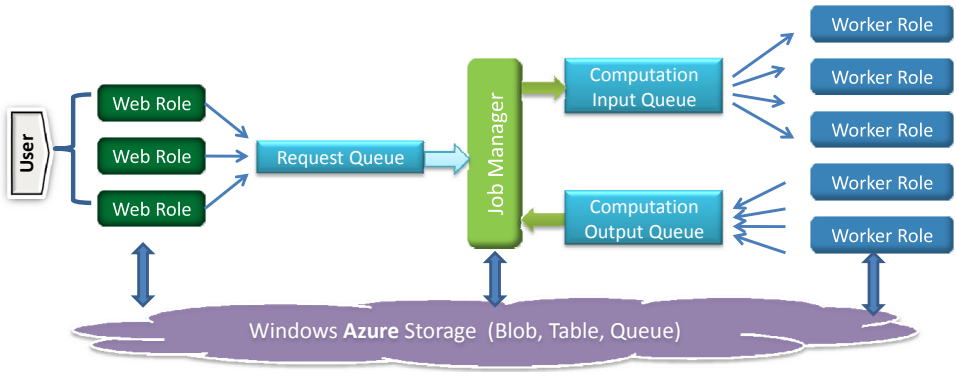

Figure 2. The architecture of our synthetic seismogram processing system implemented on Windows Azure.

CVM4SI2 [12], and 1D multiple layers velocity model SoCaL [6]. The synthetic seismograms can be used for different purposes such as seismic hazard analysis. Especially, the improvements in waveforms and arrival times of surface waves (*e.g.*SPG2 and DSC stations on Figure 1) may play essential roles for source inversion in low frequency band.

The web site runs as a service at virtual machines on Windows Azure cloud. User requests along with the parameters are then passed to the computation code running on another virtual machine on Azure cloud. Based on the location, the computation code requests the files from the datasets maintained on the Azure storage. We have implemented an efficient data localization algorithm to request files from the Azure storage according to the given location. Once the computation code gets its requested files, it does a computation on the data in parallel among all CPU cores of the virtual machine. At the end of the computation, the synthetic seismograms are generated and provided to users through the web interface.

4 Implementation

4.1 Overview of the System

Figure 2 shows the architecture of our synthetic seismogram processing system implemented on Windows Azure, which consists of four components: (1) Web role, which acts as an interface to users. (2) Job manager, which coordinates the work among the instances of the computation worker role and monitors the execution status of the system. (3) Computation worker role, which processes the work in parallel. (4) Three Azure queues, namely request queue, computation-input queue, and computation-output queue, where the request queue acts as a communication interface between the web role and the job manager, and the computation-input and computation-output queues act as the interfaces between the job manager and the computation worker role.

The web role and job manager utilize medium sized (as defined in Azure) virtual machines, and the computation worker role utilizes extra large sized (as defined in Azure) virtual machines. The medium sized virtual machine has 2 CPU cores, 3GB memory, and 500GB disk space storage, and the extra large sized virtual machine has 8 CPU cores, 15GB memory, and 2000GB disk space storage.

4.2 Job Manager

Job manager serves the following two purposes:

1. *Coordinate the computation*. The web role places user requests as messages into a request queue. When a message is retrieved, the jobs indicated by the message will be retrieved. Then these jobs are scheduled depending on the number of CPU cores on the corresponding VM. All jobs scheduled on the same CPU are sent to an instance of computation worker role as a message in the computation-input queue. For a CPU with 8 cores, each instance of the computation worker role processes the 8 sub-jobs in parallel using .NET 4.0 Task Parallel Library (TPL). The result is stored in a blob storage, and is accessible to all instances. Thus, a significant performance is gained by a factor of 8 * Num of VM instances using such a parallelization technique.

2. *Monitor system response*. The system response monitor is based on the message response time in the computation-input queue. The threshold for the response time is 2 ms. When the response time exceeds the threshold, a new computation VM instance is created. A linked list is used to preserve the creation times

of VMs. For a VM instance that had been allocated dynamically during the service, if there is no message in the computation-input queue to service and the VMs life is greater than one hour, then the VM instance will be removed.

Job manager performs these operations in multi-threading. The system response monitor runs as a child process and the coordinator of the work runs in the main process. Figure 3 shows the algorithm. The allocation and deallocation of a computation VM are asynchronous processes. The job manager has no control over the deallocation process, which is managed Azure system. In order to prevent Azure system from wrongly removing a VM that is still processing a job, a mutual exclusive lock is utilized between the job assignment and the deallocation of computation VM instances. Once the coordination thread obtains the lock, the deallocation thread cannot take place. Similarly, when the deallocation thread executes, the job manager cannot coordinate the work among the computation VMs.

4.3 Azure Partitioning and Load Balancing

Every data object, such as blob, table entity, and queue message, has a partition key. Specifically, we use the following partition key in our implementation.

Data Type	Partition Key
blob	container name + blob name
entity	table name + partition key
message	queue name

Azure has a master system that automatically maintains load balance across servers based upon these partitions. All objects with the same partition key are grouped into the same partition and are accessed from the same partition server. Grouping objects into partitions allows them to easily perform atomic operations across objects in the same partition. In addition, objects are also cached automatically based on the locality of data access.

4.4 Azure CDN

Azure Content Delivery Network (CDN) is used to deliver high-bandwidth blob content. Azure CDN currently has 20 locations globally and continues to expand. Azure CDN caches Azure blobs at strategically placed locations to provide maximum bandwidth. When CDN access is enabled for a storage account, the Azure portal provides two URLs, *i.e.*, Azure Blob URL and Azure CDN URL. Users

Thread-1: the thread that coordinate computation.

```
CloudQueueMessage msg = request_queue.GetMessage();
if (msg) {
  lock(); // for synchronization
  // read the work and split it
  while ( num_jobs_to_process > 0) {
    if ( num_jobs_to_process > num_CPU_cores) {
      num_jobs_to_schedule_currentVM = num_CPU_cores;
    } else {
      num_jobs_to_schedule_currentVM = num_jobs_to_process;
    }
    CloudQueueMessage message = new CloudQueueMessage();
    message.data = Job.getData(num_jobs_to_schedule_currentVM);
    computation_input_queue.AddMessage(message);
    num_jobs_to_process = num_jobs_to_process −
          num_jobs_to_schedule_currentVM;
  }
  unlock();
}
```

Thread-2: the thread that monitors system response time, then allocates and deallocates VMs.

```
// peek the last message in the queue
inputMsg = computation_input_queue.PeekMessage(last);
if (inputMsg) {
  inputMsg_ResponseTime =
       CurrentTime − inputMsg_InsertionTime;
  if (inputMsg_ResponseTime > 2 ms) {
    AllocateVM();
    create a new record in VM_LinkedList;
    record.VM_AllocatedTime = CurrentTime;
  }
}

if (No inputMsg) {
  lock(); // to provide synchronization
  record = VM_LinkedList.getFirst();
  while (record) {
    if (CurrentTime - record.VM_AllocatedTime > 1 hour) {
      while (the VM is processing some job) {
        thread_2.sleep(200 ms); // wait if any VM is running;
      }
      deleteVM();
      delete the record from the VM_LinkedList;
      wait for deallocation of VM to complete;
    }
  record = VM_LinkedList.getNext();
  }
  unlock();
}
```

Figure 3. The algorithm for Job Manager.

can use either of the URLs to access the blob in the container. When a request is made using the Azure Blob service URL, the blob is read directly from the Azure Blob service. When a request is made using the Azure CDN URL, the request is redirected to the CDN endpoint closest to the location from which the request was made. If the blob is not found at that endpoint, then it is retrieved from the Blob service and cached at the endpoint, where a time-to-live (TTL) setting is maintained for the cached blob. The TTL specifies how long the blob should be cached in the CDN till it is refreshed by the Blob service. The CDN attempts to refresh the blob from Azure Blob service once the TTL has elapsed. The default TTL is 72 hours. This provides performance improvement by caching the most frequently accessed blobs during their TTL periods.

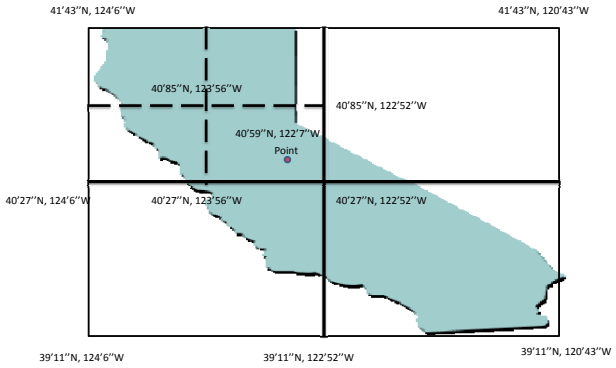

Figure 4. Bounding box based test to search the corresponding blob stored in Azure for a given point.

4.5 Data Storage

The seismic data are stored in the form of blobs. Each blob represents a data file recorded by a seismometer, which is identified by its latitude and longitude. In order to group the blobs, the entire region of California is divided into several smaller groups called blocks based on the seismic wave observation stations. Currently, there are 4096 stations presenting in the entire region of California. Hence the whole region is divided into 4096 blocks. Even though these blocks are in irregular shapes, each block is characterized by its ranges of latitude and longitude. The ranges of latitude and longitude are used to form the identification number for the block. For example, the identification number for a block whose range of latitude is from $35\,^{\circ}25'N$ to $34\,^{\circ}51'N$ and range of longitude is from $119\,^{\circ}3'W$ to $116\,^{\circ}47'W$, is given by 3525-3451-1193-11647. This identification number is used as the container name for the block. All blobs whose data under the given block are stored in its corresponding container. This helps in grouping the blobs in a better way. The unique container name and the blob name form a unique partition key, which will help Azure balance the workload among their servers. Moreover, Azure CDN has been enabled for the data storage. The TTL for the blobs is set to 1 hour.

4.6 Data Query

Data query requires locating the blob corresponding to the given point (latitude and longitude). The straightforward approach is to maintain a table storage that contains all the points within the entire region of California and their corresponding blobs. This table storage uses a linear search to look for the given point and locates its corresponding blob. The time complexity of this linear search is $O(N)$, where N is the number of points. As the entire region of California has around 100 million points, thus such a linear search would incur long time of searching. Hence this approach is inefficient.

Given a point (latitude and longitude), to locate the corresponding blob, we design an efficient algorithm by testing bounding box to narrow down the search. For California region, there are 4096 stations, which means there are totally 4096 blocks. We have 16 Azure tables to preserve the identifiers for these blocks. According to the identifiers, we can reach a blob directly. Each of the 16 Azure tables corresponds to a bounding box areas, which is a specific region (1/16 of the whole area shown in Figure 4). Given a point $(40\,^{\circ}59'N, 122\,^{\circ}7'W)$, we test 4 one-level bounding boxes as shown in Figure 4. We find that the point is inside the upper-left subarea. Then we divide the area into four quadrants along its mid points. We continue such test on the second-level bounding box regions. Thus we narrow down the search to a specific region. Then we use a linear search in the corresponding Azure table to find the container name for the given point. The blob name is known by the latitude and longitude of the given point. According to the container name and the blob name, the corresponding blob is located for a given point.

5 Experiment

We evaluated the execution performance of our system on various configurations and numbers of virtual machine instances. The experiment has been conducted on the datasets for different number of seismic wave observation stations, *i.e.*, 10, 100, 500 and 1000 stations. Figure 5 shows the total execution time of the program under various configurations. Each computing job involves all available stations.

Firstly, the experiment shows that the .NET 4.0 Task Parallel Library (TPL) helps decrease the execution time.

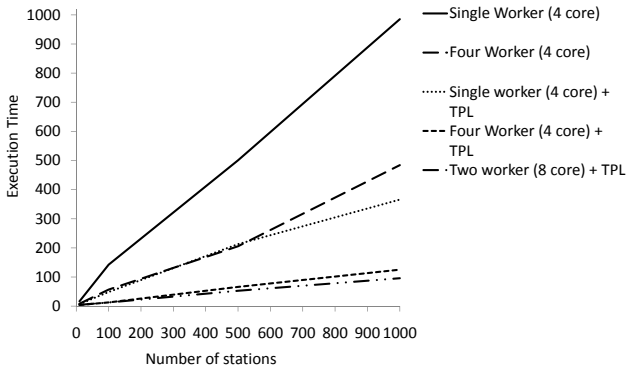

Figure 5. The execution time on performance measurement.

The total execution time for 1000 stations performed by a single worker role without TPL (*i.e.*, using one CPU core, thus totally 1 thread) is 985.06 seconds, whereas the execution time with TPL (*i.e.*, a single worker role using 4 CPU cores, thus totally 4 threads using the same CPU) is 365.91 seconds. The same experiment conducted on four worker roles shows that the total execution time without TPL (*i.e.*, totally four threads using 4 different CPUs) is 484.04 seconds, whereas the execution time with TPL (*i.e.*, totally 16 threads, 4 threads on each CPU) is 125.42 seconds.

The experiment conducted on two worker roles with TPL on 8-core CPU machines (*i.e.*, totally 16 threads, 8 threads on each CPU) shows that the total execution time of computation of data from 1000 stations is 95.97 seconds. The execution time on a single worker role with TPL on 8-core CPU machine (*i.e.*, totally 8 threads on the same CPU) is 134.37 seconds. Thus, a single worker role using 8-core machine is almost equal to four cored worker roles where each uses a 4-core machines. The reason is due to the decrease on the number of queue messages to process. For the former case, the job manager sends two messages in the queue; whereas for the latter case, the job manager sends four separate messages. Hence the number of messages is twice.

We conclude that the number of messages significantly affects on the total execution time. To increase the performance, it is better to utilize all cores on the same CPU.

6 Related Work

Traditionally, seismic wave processing utilizes cluster and grid computing. CyberShake [13] 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 [11]. The experiment shows that the performance of the cloud is nearly the same to that of NCSA's Abe cluster [15], a typical high performance computing (HPC) system.

Cloud computing has been widely used to execute scientific workflows. Hoffa et al. [9] 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 [4] 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. [16] study the role of cloud in scientific computing. As an example of scientific computing on cloud, a preliminary case study on using Aneka [5] 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. [7] 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.

Specifically, Hill et al. [8] discuss the performance experiments conducted on Windows Azure. Through their experiments, it has concluded that Windows Azure mechanisms provides good performance. They have also recommended some experience while developing scientific applications to optimize the Azure storage services. They have also suggested that dynamically adding VMs to a deployment at runtime is a useful feature of Azure by enabling dynamic load matching.

7 Conclusions

In this paper we have implemented a system for processing synthetic seismograms on Windows Azure. We also propose and implement several optimization techniques such as job manager, data storage and data query algorithm. We evaluate the system on various configurations of virtual machines offered by Windows Azure and compare the execution time for our different optimization approaches. Different optimization techniques affects the system performance dramatically. The experiment shows that cloud

computing is an ideal platform for the rapid generation and delivery of synthetic seismograms.

References

[1] Incorporated Research Institutions for Seismology (IRIS), 2010. http://www.iris.edu.

[2] K. Aki and P. Richards. *Quantitative Seismology*. University Science Books Sausalito, California, 2002.

[3] Microsoft Azure. http://www.microsoft.com/windowsazure/.

[4] 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.

[5] 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.

[6] D. Hadley and H. Kanamori. Seismic structure of the transverse ranges, california. *Geological Society of America Bulletin*, 88:1469–1478, 1977.

[7] 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.

[8] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey. Early observations on the performance of windows azure. In *the 1st Workshop on Scientific Cloud Computing (ScienceCloud 2010)*. ACM, 2010.

[9] 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.

[10] R. Jennings. *Cloud Computing with the Windows Azure Platform*. Wiley Publishing, 2009.

[11] 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.

[12] E. Lee, P. Chen, T. H. Jordan, P. J. Maechling, M. Denolle, and G. Beroza. Full-3d waveform tomography for southern california. *Eos Trans. AGU*, 91(26), 2010.

[13] 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.

[14] P. Mell and T. Grance. The NIST definition of cloud computing, 2010. http://csrc.nist.gov/groups/SNS/cloud-computing/.

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

[16] 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.

[17] L. Zhao, P. Chen, and T. H. Jordan. Strain green's tensors, reciprocity and their applications to seismic source and structure studies. *Bulletin of the Seismological Society of America*, 96(5):1753–1763, 2006.