Earthq Sci (2013) 26(5):321–329
DOI 10.1007/s11589-013-0038-8

RESEARCH PAPER

A cloud-based synthetic seismogram generator implemented using Windows Azure

Po Chen · En-Jui Lee · Liqiang Wang

Received: 25 July 2013 / Accepted: 30 October 2013 / Published online: 13 December 2013

Abstract Synthetic seismograms generated by solving the seismic wave equation using numerical methods are being widely used in seismology. For fully three-dimensional seismic structure models, the generation of these synthetic seismograms may require large amount of computing resources. Conventional high-performance computer clusters may not provide a cost-effective solution to this type of applications. The newly emerging cloud-computing platform provides an alternative solution. In this paper, we describe our implementation of a synthetic seismogram generator based on the reciprocity principle using the Windows Azure cloud application framework. Our preliminary experiment shows that our cloud-based synthetic seismogram generator provides a cost-effective and numerically efficient approach for computing synthetic seismograms based on the reciprocity principle.

Keywords Reciprocity · Synthetic seismogram · Cloud computing · Windows Azure

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) 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 now 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 (Olsen 1994; Graves 1996; Akcelik et al. 2003; Olsen et al. 2003; Komatitsch et al. 2004), 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), and where the seismologic 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 are adopted to improve the seismologic models and reinitiate the inference cycle at a higher level (Chen 2011; Liu and Gu 2012).

The purpose of this study is to establish a cloud-based computational platform for rapidly delivering synthetic seismograms computed from well-calibrated three-

P. Chen (✉) · E.-J. Lee
Department of Geology and Geophysics, University of Wyoming, Laramie, WY 82071, USA
e-mail: pchen@uwyo.edu

L. Wang
Department of Computer Science, University of Wyoming, Laramie, WY 82071, USA

dimensional seismic structure models to seismologists worldwide to facilitate simulation-based seismologic 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. The underlying theoretic 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 means of 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 this system will be useful for seismologic research in general and seismic hazard assessment in Southern California in particular.

2 Receiver green tensor (RGT) and reciprocity

Following Zhao et al. (2006), the displacement synthetic seismogram from a point source located at $\boldsymbol{r}'$ with moment tensor M_{ij} can be expressed as (e.g., Aki and Richards 2002, Eq. 3.23)

$$u_k(\boldsymbol{r}, t; \boldsymbol{r}') = M_{ij}\partial'_j G_{ki}(\boldsymbol{r}, t; \boldsymbol{r}'), \tag{1}$$

where ∂'_j denotes the source-coordinate gradient with respect to $\boldsymbol{r}'$, and the Green's tensor $G_{ki}(\boldsymbol{r}, t; \boldsymbol{r}')$ relates a unit impulsive force acting at location $\boldsymbol{r}'$ in direction $\hat{\mathbf{e}}_i$ to the displacement response at location $\boldsymbol{r}$ in direction $\hat{\mathbf{e}}_k$. Taking into account the symmetry of the moment tensor, we also have

$$u_k(\boldsymbol{r}, t; \boldsymbol{r}') = \frac{1}{2}\left[\partial'_j G_{ki}(\boldsymbol{r}, t; \boldsymbol{r}') + \partial'_j G_{kj}(\boldsymbol{r}, t; \boldsymbol{r}')\right] M_{ij}. \tag{2}$$

Synthetic seismograms due to a finite seismic source can be obtained by superposing synthetics seismograms of point sources following a kinematic rupture propagation model.

Applying reciprocity of the Green's tensor

$$G_{ki}(\boldsymbol{r}, t; \boldsymbol{r}') = G_{ik}(\boldsymbol{r}', t; \boldsymbol{r}), \tag{3}$$

Eq. (1) can be written as

$$u_k(\boldsymbol{r}, t; \boldsymbol{r}') = \frac{1}{2}\left[\partial'_j G_{ik}(\boldsymbol{r}', t; \boldsymbol{r}) + \partial'_j G_{jk}(\boldsymbol{r}', t; \boldsymbol{r})\right] M_{ij}. \tag{4}$$

For a given receiver location $\boldsymbol{r} = \boldsymbol{r}_{\mathrm{R}}$, the receiver Green tensor (RGT) is a third-order tensor defined as the spatial–temporal strain field

$$H_{jik}(\boldsymbol{r}', t; \boldsymbol{r}_{\mathrm{R}}) = \frac{1}{2}\left[\partial'_j G_{ik}(\boldsymbol{r}', t; \boldsymbol{r}_{\mathrm{R}}) + \partial'_j G_{jk}(\boldsymbol{r}', t; \boldsymbol{r}_{\mathrm{R}})\right]. \tag{5}$$

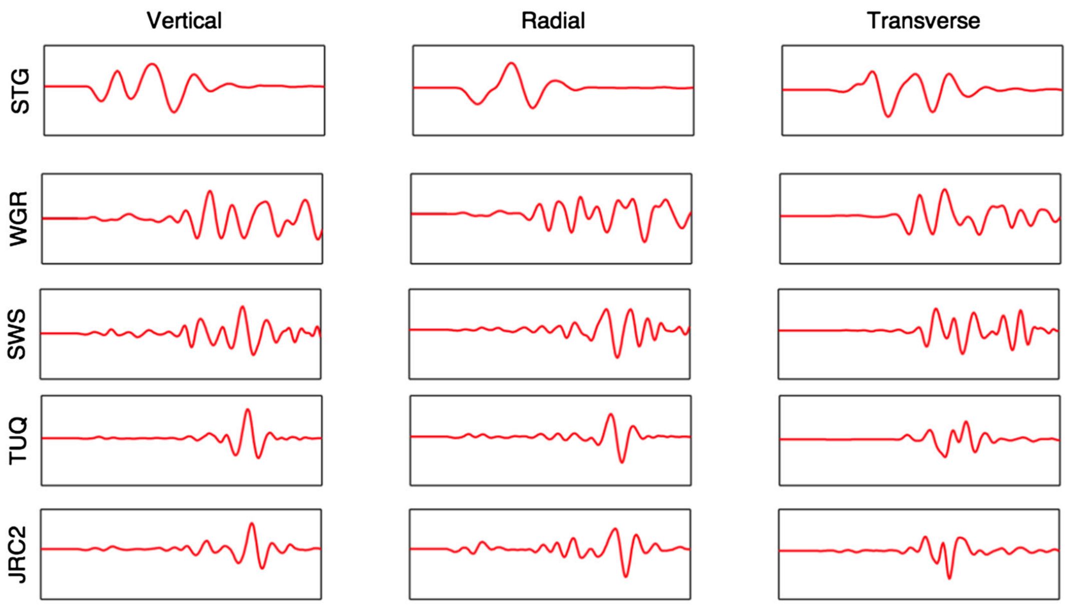
Using this definition, the displacement recorded at receiver location $\boldsymbol{r}_{\mathrm{R}}$ due to a source at $\boldsymbol{r}_{\mathrm{S}}$ with moment tensor $\boldsymbol{M}$ can be expressed as

$$\begin{aligned} u_k(\boldsymbol{r}_{\mathrm{R}}, t; \boldsymbol{r}_{\mathrm{S}}) &= M_{ij} H_{jik}(\boldsymbol{r}_{\mathrm{S}}, t; \boldsymbol{r}_{\mathrm{R}}) \text{ or } \mathbf{u}(\boldsymbol{r}_{\mathrm{R}}, t; \boldsymbol{r}_{\mathrm{S}}) \\ &= \boldsymbol{M} : \boldsymbol{H}(\boldsymbol{r}_{\mathrm{S}}, t; \boldsymbol{r}_{\mathrm{R}}). \end{aligned} \tag{6}$$

Most of the numerical algorithms for solving the seismic wave equation, such as finite difference, finite element, and spectral element methods, explicitly use the spatial gradients of the displacement (or velocity) and the stress (or stress rate) in their calculations. For a given receiver, the RGT can therefore be computed through three wave propagation simulations with a unit impulsive force acting at the receiver location $\mathbf{r}_{\mathrm{R}}$ and pointing in the direction $\hat{\mathbf{e}}_k$ ($k = 1, 2, 3$) in each simulation and store the strain fields at all spatial grid points $\boldsymbol{r}'$ and all time sample t. The synthetic seismogram at the receiver due to any point source located within the modeling domain can be obtained by retrieving the strain Green's tensor at the source location from the RGT volume and then applying Eq. (6).

Examples of synthetic seismograms generated using this approach for a magnitude-4.9 earthquake around the Los Angeles basin area are shown in Fig. 1. The seismic structure model used in this calculation was the three-dimensional Southern California Earthquake Center (SCEC) Community velocity model version 4.0 (CVM4) (Magistrale et al. 2000). The RGTs were computed by solving the 3D elastic wave equation using the 4th-order staggered-grid finite difference method. We also computed synthetic seismograms through the normal approach by propagating the wave-field from the earthquake source to the seismic stations, and the synthetic seismograms are identical to those obtained using Eq. (6).

For a region with well-calibrated three-dimensional seismic structure models, such as Southern California, we can compute and archive the RGTs for all seismic stations in that region in a cloud storage facility in advance. A user, who is interested in the synthetic seismograms of a particular earthquake, can enter and submit the earthquake source parameters (e.g., the hypocenter location, the focal mechanism) through a web page. The synthetic seismograms of that earthquake at all stations are then computed by combining those source parameters and the RGTs using Eq. (6) and returned to the user through the web browser. The details about the RGTs for 219 stations in Southern California computed using our latest full-3D, full-waveform tomography model can be found in Lee et al. (2011).



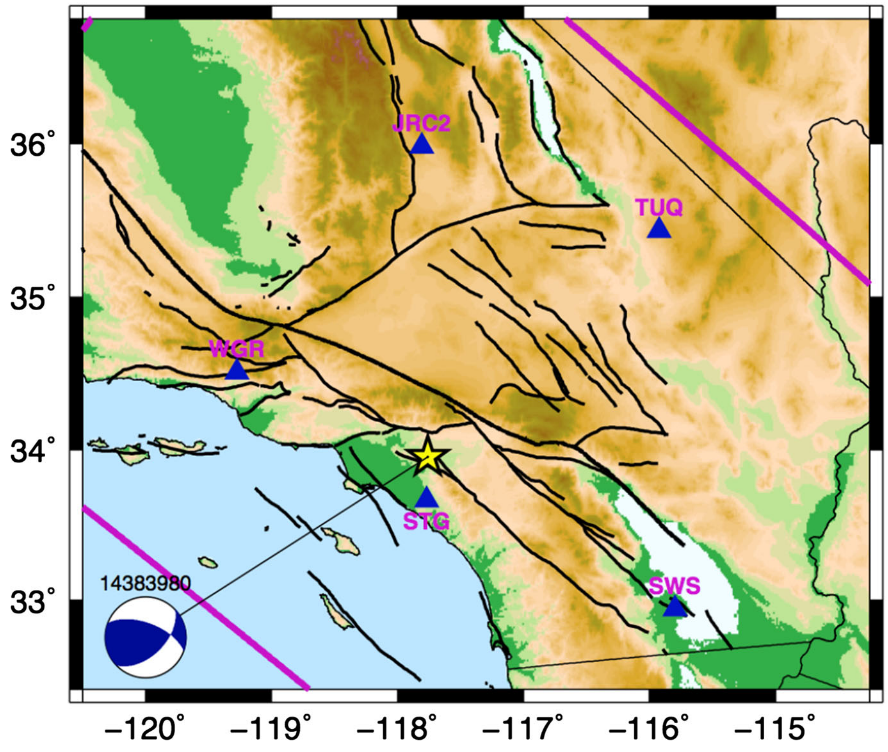


Fig. 1 Three-component (*vertical*, *radial*, and *transverse*) synthetic seismograms of the July 29, 2008, magnitude 5.5, Chino Hills earthquake at five different seismic stations. The synthetic seismograms were computed using both reciprocity (*red*) and the normal forward wave propagation from source to station (*blue*). Only the *red curves* are visible, because they are exactly on *top* of the *blue curves*. The lengths of the seismograms are 120 s. The amplitudes of the seismograms have been normalized. The focal mechanism of the earthquake is shown in the beach ball. The *yellow star* shows the location of the epicenter. Locations of the seismic stations are indicated using *blue triangles*. The names of the stations are marked above the station locations. Major faults in this area are shown as *black solid line*s. Background color shows regional topography with *green* indicating lower elevations and *yellow* to *brown* indicating higher elevations

3 Cloud computing and Windows Azure

High-performance computing (HPC) is traditionally implemented using computer clusters, which are set up using nearly identical computers (i.e., compute nodes) and special networking adapters for moving data among the nodes at very high speeds. But, networks have become faster, and a new clustering technology, known as the cloud, that leverages virtualization tools is becoming more appealing. Virtualization is a technology referring to the creation of a virtual (rather than actual) version of something, such as an operating system, a server, a storage device or network resources. A cloud is a virtual cluster composed of identical virtual machines (VMs) running on a collection of servers, which can be physically very different, connected by a virtual network, which can consist of different physical LANs and/or the Internet.

According to the definition given by the National Institute of Standards and Technology (NIST) (Mell and Grance 2011), in comparison with conventional-distributed computing based on traditional physical computer clusters, 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 the Internet and can be accessed by any platforms; (3) *Resource pooling*: the computing resources are pooled by the provider through the multi-tenant model to various consumers on demand; (4) *Rapid elasticity*: the computing capability can be scaled in and out very quickly; (5) *Measured Service*: Resource usage can be monitored and transparently controlled.

On a conventional physical cluster shared by many users, a user submits a job to the queue and waits for its completion. Depending upon the number of jobs in the queue and the computing capability of the cluster, the amount of time spent on waiting in the queue can be substantial. Compared with physical clusters, the cloud platform provides a flexible, on-demand computing infrastructure. Based on the aforementioned features, the cloud is actually an ideal platform to deploy the system for computing and delivering synthetic seismograms with rapid response and scalable performance. It is a cost-effective substitute for the traditional dedicated physical computing cluster.

The Windows Azure is a Platform as a Service (PaaS) running at Microsoft data centers. The platform consists of a highly scalable (i.e., elastic) cloud operating system and a data storage system. Users have control over their applications, as well as the environment configurations in which their applications run, such as databases, third-party libraries, the operating systems, etc., depending upon the level of manageability and control that the users prefer. The Windows Azure platform takes care of the management of the underlying physical infrastructure such as the networks, the servers, and the storage systems. For web applications, such as our synthetic seismogram generator, the users can choose among three pre-configured services: Web Sites, Cloud Services, and VM. The Web Sites service has a relatively simple architecture, which consists of a web frontend and an optional database. In this case, Windows Azure has the full control of the VMs hosting the web application. For more complicated applications, Cloud Services allow users to create highly scalable, multitier architectures and give users more freedom to customize the hosting VMs, such as installing third-party libraries. The VMs service gives users the most control over the hosting VMs including the operating systems, the configuration, and the installed software and services. In this case, the users can choose among a variety of pre-configured, published VM images or create and manage their own VM images. Considering the complexity of our synthetic seismogram generator application and the trade-off between manageability and control, we have chosen the Cloud Services in our current implementation.

A Cloud Service is actually a container that can host multiple Cloud Roles. A Cloud Role is a collection of codes (or “service bits” in Azure jargon) for realizing a particular purpose. A Cloud Role can have multiple Role Instances. A Role Instance is a single deployment of a Cloud Role to a dedicated VM. All the Role Instances of the same Cloud Role share the same service bits. Windows Azure has a set of pre-configured Cloud Roles that users can choose from and customize. Currently available Cloud Roles include the Web Role, which provides codes for generating a dynamic web frontend that handles the interactions with the customers of our application through browsers, and the Worker Role, which handles background data processing and computation and interacts with the Web Role. For instance, if we consider a Cloud Service with one Web Role with two Role Instances, and two different Worker Roles with one Role Instance for the first Worker Role and two Role Instances for the second Worker Role, Windows Azure will start five VMs when this Cloud Service is deployed. To serve more customers of our application, we can simply increase the number of Role Instances of each Cloud Role.

To balance the load on multiple Role Instances, Windows Azure provides an automatic Load Balancer. For multiple Role Instances of the same Web Role, each Role Instance will have a different IP address since each Role Instance runs on a different VM. The Load Balancer will provide all customers of our application with the same web address and will evenly distribute the requests from all customers among all the active Role Instances with different IP addresses automatically. If some of the Role

Instances become offline, the Load Balancer will automatically remove those Role Instances from its pool and evenly distribute requests among remaining active Role Instances.

Windows Azure provides a highly scalable cloud storage system, which can be accessed and managed through 3 types of data structures: Blobs, Tables, and Queues. The Blob is used for storing unstructured binary and text data. Blobs are grouped into Containers. A Container can store an unlimited number of Blobs, and every Blob must be inside a Container. The Table is used to store non-relational structured data. A Table is a collection of Entities, and an Entity is a set of Properties, which are simply name-value pairs. The Queue is used for storing messages that can be accessed by a client and providing reliable messaging between Role Instances. A user of the storage system will need to apply for a storage account, and each storage account has access to 100 TB storage space. A storage account can host any number of Containers, Tables, and Queues. Every data structure under a storage account can be addressed through a unique URL and accessed in a browser through that URL anytime, anywhere through an optional authentication process provided by Azure.

4 Implementation

An architectural overview of our synthetic seismogram generator application for Southern California is shown in Fig. 2. It starts with a web site, which acts as a user interface where users can request the generation of synthetic seismograms of a particular earthquake in Southern California by entering the longitude, latitude, and depth of the hypocenter location and the strike, dip, and rake of the focal mechanism. Users can also choose to enter the names of the seismic stations if they are interested in synthetic seismograms only at those stations, otherwise synthetic seismograms at all stations in Southern California will be computed and returned. The web site runs as one or more Role Instances of our Web Role at VMs in the Azure cloud. User requests along with the earthquake source parameters are then passed to the Role Instances of our Work Role running on other different VMs. Based on the user inputs, the instances of the Work Role request the files from the datasets maintained on the Azure storage. The RGTs of all seismic stations in Southern California have been computed using the finite difference code on our local 14-node computer cluster and then uploaded to the Azure cloud storage system in advance. We have implemented an efficient data localization algorithm to efficiently fetch files from Azure cloud storage according to the given hypocenter location and the user's station selection. Once the Worker Roles get the requested files, they calculate synthetic seismograms using Eq. (6). The generated synthetic seismograms are then sent to the instances of the Web Role and provided to the users through the web interface.

4.1 System overview

As shown in Fig. 2, our system consists of four components: (1) the Web Role, which acts as a browser interface to users; (2) the 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 carries out the actual computation; (4) three Azure Queues, namely the request queue, the computation-input queue, and the computation-output queue. The request queue acts as a communication interface between the web role and the job manager. 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

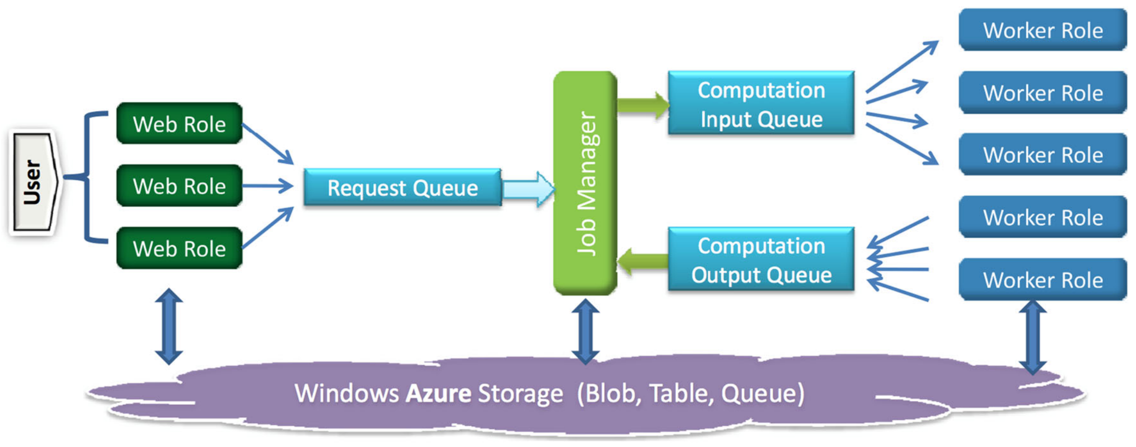


Fig. 2 A schematic overview of our synthetic seismogram generator cloud service application's architecture as implemented using Windows Azure

medium-sized VMs, each of which has 2 CPU cores, 3 GB memory, and 500 GB disk storage, and the computation worker role utilizes extra-large-sized VMs, each of which has 8 CPU cores, 15 GB memory, and 2 TB disk storage.

4.2 Job manager

An important advantage of using cloud computing for our application is its capability to increase or decrease the number of VMs used for computation very quickly and dynamically based on the number of user requests. This capability is realized through using our job manager, which is implemented using multi-threading. Our job manager has two threads; thread 1 is responsible for scheduling and coordinating computational tasks, and thread 2 is responsible for monitoring the response of our system and dynamically adjusting the number of VMs used for computation. Some segments of the codes used for implementing these two threads are shown in Fig. 3.

An instance of the web role places user requests as messages into the request queue. When a message is retrieved by thread 1 of the job manager, all the jobs corresponding to that message are retrieved and then scheduled. All the jobs scheduled on the same CPU are sent to the same instance of the computation worker role as a message in the computation-input queue. For a CPU with 8 cores, an instance of the computation worker role will process 8 jobs in parallel at a time using the .NET 4.0 Task Parallel Library (TPL), which is a collection of APIs for realizing multi-thread parallelism on multi-core CPUs. On those extra-large-sized VMs, TPL allows us to obtain a speedup factor of about 8 times the number of instances of the computation work role. The results of the calculations are stored into the cloud storage as blobs, and the URL addresses of those blobs are placed as messages into the computation-output queue.

Thread 2 of the job manager constantly monitors the message response time for messages waiting in the computation-input queue. If the response time exceeds a pre-set threshold value, a new instance of the computation work role is initiated on a new VM. A linked list is used to store the create time of the VMs hosting the work role instances. For a newly created computation work role instance, if there is no message in the computation-input and the life of the VM is greater than another pre-set threshold, the VM will be removed from the pool and deallocated. However, the job manager itself does not have control over the deallocation process, which is managed only by Azure. In order to prevent Azure from deallocating a VM that is still processing a job, we use a mutual exclusion lock between the job assignment and the deallocation of a computation VM. Once thread 1 of the job manager obtains the lock during job scheduling, the deallocation call from thread 2 cannot take place. On the other hand, when the deallocation process is initiated by thread 2, thread 1 of the job manager cannot assign jobs to the VM.

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();
}
```

Fig. 3 Code segments of the job manager as implemented using the Windows .NET framework. The code segment for thread 1 is shown in the upper panel, and the code segment for thread 2 is shown in the lower panel

4.3 Data management

As shown in Eq. (5), the RGT for a station is a spatial–temporal 4D volume. This 4D volume is sampled using the same 3D uniform spatial mesh that is used in the finite difference simulations. We divide the spatial mesh evenly into 4,096 blocks, and each block corresponds to one Azure container in our storage account. The RGT on one spatial grid point, which is a 6-component time series, corresponds to one Azure blob. All the RGTs within one block are placed inside the container corresponding to that block. Each container is indexed using the longitudes and latitudes of the bounding box of its corresponding spatial block. For instance, for a spatial block whose range of latitude is from 35°25′N to 34°51′N and range of longitude is from 119°3′W to 116°47′W, its corresponding Azure container is given an index of 3525-3451-11903-11647. Inside of each container, the blobs are also indexed using the longitudes and latitudes of its corresponding spatial grid points.

In Azure, every data object, such as a blob, a table entity, and a queue message, has a partition key. Azure automatically maintains load balance across different data servers based upon partitions. All the data objects with the same partition key are grouped into the same partition server. Grouping objects into partitions allows them to easily perform atomic operations across all data objects in the same partition. In addition, data objects in the same partition are also cached automatically to improve bandwidth. In our application, the container index combined with the blob index forms the unique partition key for every blob. A direct consequence of this choice of the partition key is that the RGTs of different seismic stations at the same spatial grid point are all grouped into the same partition server. Considering Eq. (6), for a given earthquake hypocenter location, the mathematical operation applied on the RGTs of different seismic stations is actually identical. And, it is often the case that a user who is interested in a particular earthquake usually wants all the synthetic seismograms of that earthquake at multiple seismic stations. For different earthquakes, which usually have different hypocenter locations, data accesses are then directed to different partitions. This choice of the partition key actually allows us to maximize the benefit of the caching capability of the automatic Azure data partition system while helping Azure to balance the workload among its partition servers.

To improve the bandwidth of data movement, Azure uses a content delivery network (CDN), which currently has 20 locations (i.e., endpoints) globally and is rapidly expanding. Azure CDN can cache 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., the Azure blob URL and the Azure CDN URL, for accessing the same blob. Users can use either of them to access the data object. If the user requests data through the blob URL, the blob is then read directly from the Azure blob service. If a request is made through the CDN URL, the request is redirected to the CDN endpoint closest to the location from which the request is made. If the blob is not found on the endpoint, it is retrieved from the Azure blob service and cached at the endpoint, where a time-to-live (TTL) setting determines how long the blob will be cached. In our implementation, we enabled Azure CDN, and the TTL is set to 1 h. The reason for doing this in our implementation is that there might be situations in which multiple users may request for the synthetic seismograms of the same earthquake in a relatively short period of time, for instance, shortly after the actual earthquake occurred. Under such circumstances, caching the blobs of that earthquake will significantly reduce the data delivery time for the second and subsequent requests.

Given the longitude, latitude, and depth of a hypocenter location, we need an efficient query algorithm that can quickly locate the blob whose spatial location is closest to the hypocenter. In our current implementation, we have adopted a hierarchical quad-tree searching algorithm. Using the whole state of California as an example, Fig. 4 shows the search process for an arbitrary epicenter location at (40°59′N, 122°7′W). First, the whole region is divided into four level-1 bounding boxes, and by comparing the given epicenter location with the longitude and latitude ranges of each level-1 boxes, we find that this epicenter lies in the upper-left level-1 box. The upper-left quadrant is then divided into four level-2 boxes, and by comparing the location of the epicenter with the longitude and latitude ranges of the level-2 boxes, we find that the epicenter is inside the lower-right level-2 box. This process is iterated until the correct container is identified. Within the selected container, the correct blob is then identified through a linear search. Since the containers and the blobs are all indexed by their longitudes and latitudes, the search algorithm does not have to inspect the content of the data objects and can be carried out highly efficiently.

5 Performance analysis

We have carried out extensive experiments to evaluate the performance of our application. Figure 5 shows the average total execution time for generating different number of synthetic seismograms of a random selection of earthquakes. All experiments were carried out using the same set of RGTs that had already been computed and uploaded to Azure cloud storage.

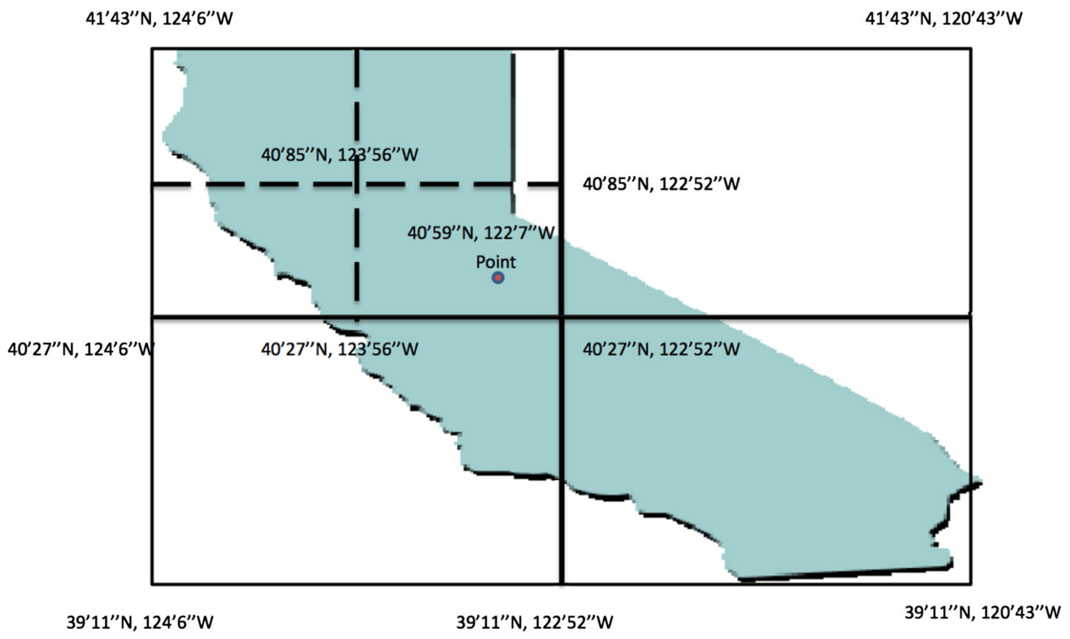


Fig. 4 An example of our quad-tree searching algorithm for locating the blob corresponding to a given geographic point. *Black solid lines* indicate the boundaries of the 4 level-1 bounding boxes. The two *black dash lines* indicate the inner boundaries of 4 level-2 bounding boxes

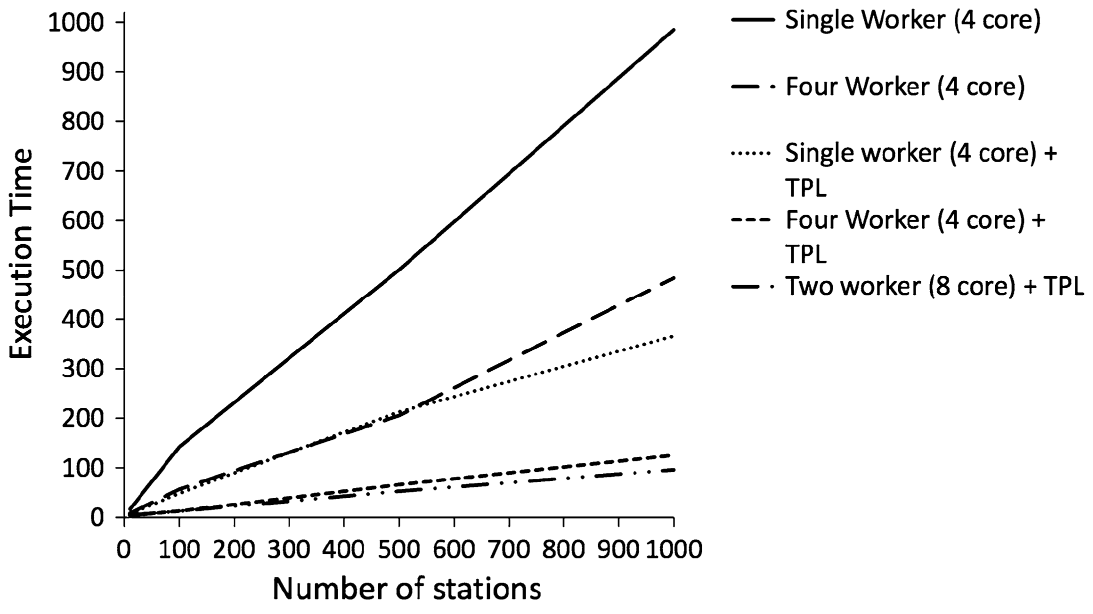


Fig. 5 The total execution time in seconds for our performance evaluation experiments. The unit on the *vertical axis* is seconds. The *different line* styles correspond to different number of worker role instances and thread numbers. *Solid line* single worker (4 core), *long dash line* 4 workers (4 core), *short dash line* 4 workers (4 core) with TPL, *dotted line* single work (4 core) with TPL, *dash double-dot* 2 workers (8 core) with TPL

Figure 5 shows that the .NET 4.0 TPL can help reduce the total execution time significantly. The total execution time for 1,000 seismograms performed by a single worker role instance without using TPL (i.e., using one CPU core and one thread) is about 985.06 s, while the execution time for one worker role instance with TPL (i.e., using four CPU cores and four threads) is about 365.91 s. If we repeat the same experiment using 4 worker role instances but without TPL (i.e., using 4 CPUs but only 1 core on each CPU), the total execution time is 484.04 s. If we turn on the TPL (i.e., using all 4 cores on each of the 4 CPUs), the total execution time is reduced to 125.42 s.

If we use 2 worker role instances with TPL on two 8-core CPU VMs (i.e., using all 16 cores on 2 CPUs, 8 threads on each CPU), the total execution time for generating 1,000 seismograms is about 95.97 s. If we use a single worker role instance with TPL on one 8-core VM (i.e., 8 threads on the same CPU), the execution time for generating 1,000 seismograms is about 134.47 s, which is similar to the performance of 4 worker role instances with each using a 4-core CPU. This comparison exposes the effects of the number of messages on the performance of our application. In the former case, the job manager sends only one message into the queue, while in the second case the job manager sends 4 messages. The total number of messages in the queue has a significant impact on the performance of our application. To improve the performance of our application, for a given number of worker role instances, it is preferable to utilize all cores on the same CPU and for a given number of total core counts, it is preferable to reduce the number of worker role instances.

6 Discussion and conclusions

In this study, we have implemented a cloud-based synthetic seismogram generator using Windows Azure. The theoretic foundation of our system is built on the receiver RGTs and the reciprocity principle. The RGTs for all seismic stations in Southern California were computed on a conventional computer cluster using a well-calibrated three-dimensional seismic structure model and then transferred to the Windows Azure cloud storage. We have built a highly efficient and highly scalable Windows Azure cloud service application that can quickly query the RGT dataset, compute the synthetic seismograms for all stations based on the earthquake source parameters entered by the user through our web interface, and deliver the results to the user. Preliminary performance analysis has shown that the cloud platform is a suitable, highly efficient, and cost-effective platform for implementing this type of applications.

One important advantage of the cloud platform is its capability to quickly and dynamically scale-out and scale-in the computational resources used for the application based on user requests. In our application, this capability is realized using our multi-thread job manager. The Azure cloud storage provides abundant storage capacity. However, to effectively take advantage of this storage capacity, we need to minimize data query and data transfer overheads. To maximize the bandwidth, data objects need to be carefully arranged, grouped, and placed in the cloud. One bottleneck of our application is the communication between different role instances, in our case the communication between web role instances and worker role instances. Our experiments have shown that the efficiency of our system can be significantly improved by minimizing the total number of such communications. Our result also suggests that at the current stage for tightly coupled applications that require a large number of communications, the conventional cluster with dedicated high-speed interconnect is still preferred over the cloud platform.

References

Akcelik V, Bielak J, Biros G, Epanomeritakis I, Fernandez A, Ghattas O, Kim EJ, Lopez J, O'Hallaron D, Tu T, Urbanic J (2003) High resolution forward and inverse earthquake modeling on terascale computers. Proceedings of the 2003 ACM/IEEE conference on supercomputing, pp 52–73

Aki K, Richards PG (2002) Quantitative seismology, 2nd edn. University Science Books, Sausalito, California, USA

Chen P (2011) Full-wave seismic data assimilation: theoretical background and recent advances. Pure Appl Geophys 168(10): 1527–1552

Graves R (1996) Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences. Bull Seismol Soc Am 86:1091–1106

Komatitsch D, Liu Q, Tromp J, Süss P, Stidham C, Shaw JH (2004) Simulations of ground motion in the Los Angeles basin based upon spectral element method. Bull Seismol Soc Am 94: 187–206

Lee E-J, Chen P, Jordan TH, Wang L (2011) Rapid full-wave centroid moment tensor (CMT) inversion in a three-dimensional earth structure model for earthquakes in Southern California. Geophys J Int. doi:10.1111/j.1365-246X.2011.05031.x

Liu Q, Gu YJ (2012) Seismic imaging: from classical to adjoint tomography. Tectonophysics 566–567:31–66

Magistrale H, Day S, Clayton RW, Graves R (2000) The SCEC Southern California reference three-dimensional seismic velocity model Version 2. Bull Seismol Soc Am 90:S65–S76

Mell P, Grance T (2011) The NIST definition of cloud computing: recommendations of the National Institute of Standards and Technology. US Department of Commerce, Special Publication 800-145. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, pp 20899-8930

Olsen KB (1994) Simulation of three-dimensional wave propagation in the Salt Lake Basin. Ph.D. Thesis, University of Utah, Salt Lake, pp 157

Olsen KB, Day SM, Bradley CR (2003) Estimation of Q for long-period (>2 s) waves in the Los Angeles Basin. Bull Seismol Soc Am 93:627–638

Zhao L, Chen P, Jordan TH (2006) Strain Green's tensors, reciprocity and their applications to seismic source and structure studies. Bull Seismol Soc Am 96(5):1753–1763