

Scalable seismic modeling with Azure Batch

George Iordanescu¹, Wee Hyong Tok¹, Philipp A. Witte²,
Mathias Louboutin² and Felix J. Herrmann²

¹ Microsoft Corporation

² Georgia Institute of Technology, School of Computational Science and Engineering

1 Contents

- Overview
- Seismic modeling with Devito
- Prerequisites
- Set up:
 - Docker container
 - Upload user files
 - Batch Shipyard configuration
- Submit and monitor a job
- Performance analysis and results
- Clean up
- Next steps

2 Overview

Numerical seismic modeling lies at the core of many geoscience applications, such as subsurface imaging or CO₂ monitoring, and involves modeling acoustic or elastic wave propagation in the subsurface. This physical process is modeled numerically by solving partial differential equations on large-scale two- and three-dimensional domains using finite-difference time-stepping. Typical seismic surveys, as used for resource exploration in the oil and gas industry, involve modeling seismic data for thousands of individual experiments. Solving wave equations for real-world problem sizes is computationally expensive, but as individual experiments are independent of each other, this process can be executed as an embarrassingly parallel workload.

This guide provides a walk through of how to deploy a parallel seismic modeling workload to Azure using [Azure Batch](#) and [Batch Shipyard](#). For discretizing and solving the underlying wave equations the example uses [Devito](#), a domain-specific language compiler for finite-difference modeling. Devito allows implementing wave equations as high-level symbolic Python expressions and automatically generates and compiles optimized C code during runtime. The Python script provided in the accompanying software models seismic data for a given seismic source location and stores the modeled data in [Blob storage](#). To model seismic data for a large number of individual source experiments, the workload is deployed as a parallel job to a pool of workers using Batch Shipyard, a tool for provisioning and monitoring workloads with Azure Batch. Each task of the batch job corresponds to modeling seismic data for a given seismic source location and can be executed on

a single compute node or on a cluster of multiple nodes using message passing (Figure 1). By leveraging the scalability of Azure Batch, the modeling workflow can be scaled to thousands of tasks, while automatic scaling enables cost efficient provisioning of computational resources.

The following sections provide a brief overview of seismic modeling with Devito and step-by-step instructions how the bundle the software into a docker container and deploy it as a parallel workload to Azure. This process involves uploading the necessary user data, such as the seismic model and acquisition geometry to Blob storage and setting up the batch environment. This guide demonstrates how to submit and monitor your job and discusses some possible extensions and applications of this software.

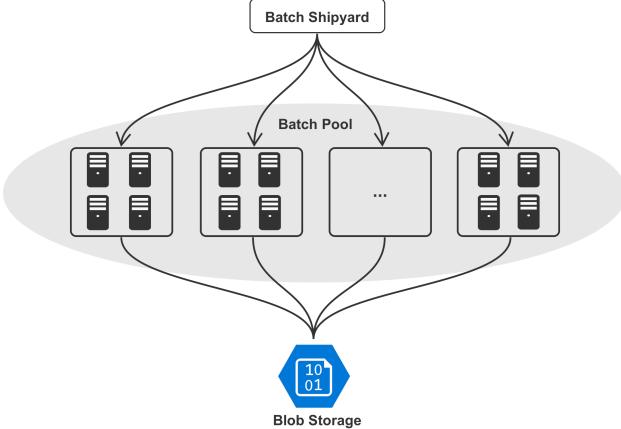


Figure 1: Parallel batch job for modeling seismic data. Jobs are submitted to a pool using Batch Shipyard and individual tasks either run on separate nodes or as distributed workloads on small clusters. Each task is responsible for modeling the data of a given source location and stores the generated data in Blob storage upon completion.

3 Seismic modeling with Devito

Seismic modeling is used in geophysical exploration and monitoring to numerically predict data that is recorded in seismic surveys. Such surveys involve a seismic source being repeatedly fired within the target area, which causes waves to propagate through the subsurface. Where physical properties of the subsurface such as wave speed or density change, waves are reflected and travel back to the surface, where they are recorded by an array of seismic receivers (Figure 2a). The receivers record relative pressure changes in the water as a function of time, sensor number and the source location (Figure 2b). Industry-scale surveys involve thousands of seismic source locations and data has to be modeled for each location individually by numerically solving the corresponding wave equation.

In this workflow, wave equations are discretized and solved with Devito, a Python package for finite-difference modeling and inversion. Devito's application programming interface allows implementing wave equations as symbolic Python expressions that closely resemble the mathematical notation. For example, the acoustic isotropic wave equation with constant density is implemented as:

```
pde = model.m * u.dt2 - u.laplace + model.damp*u.dt
```

where `model.m` is a symbolic expression for the acoustic wave speed, `u` is the discretized acoustic wavefield and `u.dt2` and `u.laplace` are short-hand expressions for finite difference stencils of second temporal and spatial derivatives. The last expression implements absorbing boundaries to mimic wave propagation in an unbounded domain. Similarly, sources and receivers can be symbolically defined as well and added to this expression, making it possible to leverage Devito for real-world applications in exploration seismology.

During runtime, Devito automatically generates optimized C code for solving this equation from the symbolic expression, using its internal compiler. Devito's automated performance optimizations include equation clustering, FLOP-reduction optimization, SIMD-vectorization, loop-blocking, as well as the introduction of shared and/or distributed memory parallelism (i.e. multi-threading and domain decomposition).

This tutorial demonstrates how to use Azure Batch to model seismic data for a large number of source locations as an embarrassingly parallel workload, making it possible to easily scale to relevant problem sizes as encountered in real-world scenarios. The software is deployed to a pool of parallel workers as a Docker container, which contains Devito implementations of the tilted transverse-isotropic (TTI) wave equation. This tutorial covers how to build the required Docker containers from scratch and how to manage the parallel pool and job submissions with Batch Shipyard.

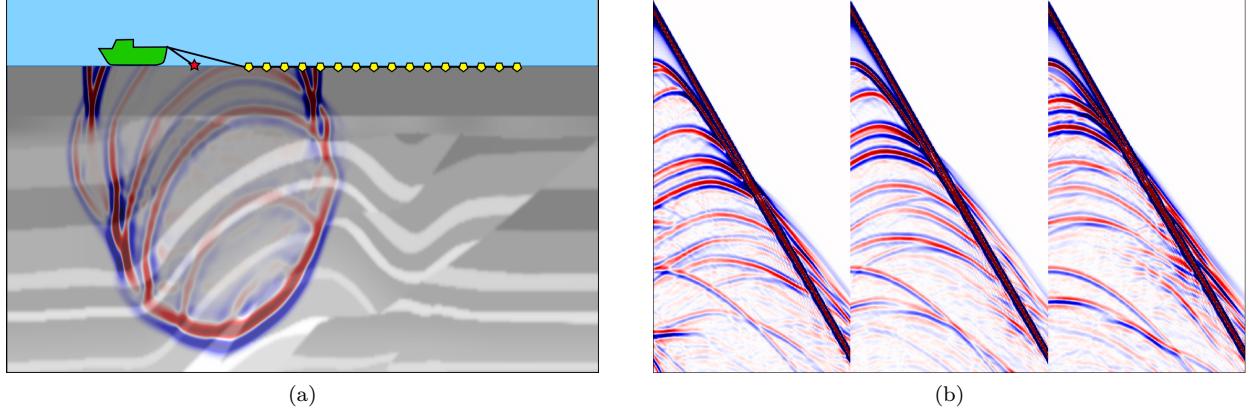


Figure 2: A marine seismic survey, in which a vessel tows a source and generates seismic waves that travel through the subsurface. At geological interfaces, waves are reflected back to the surface, where they are recorded by a set of seismic receivers (a). The receivers record pressure changes in the water as a function of time (vertical axis) and receiver number (horizontal axis) and the experiment is repeated for many source locations (b).

4 Experimental set up

The model used in this tutorial is a 3D synthetic TTI model derived from the 3D SEG Salt and Overthrust models, with dimensions of $3.325 \times 10 \times 10$ km. The model consists of six physical parameters, namely P-wave velocity, density, Thomsen parameters epsilon and delta, as well as tilt and azimuth of the anisotropy symmetry axes. The model is discretized with a 12.5 m cartesian grid, which results in $347 \times 881 \times 881$ grid points, including 80 grid points for absorbing boundaries in each dimension. The seismic data is modeled for 2 seconds using a Ricker wavelet with 15 Hertz peak frequency, with data being recorded by 1,500 receivers that are randomly distributed along the ocean floor (Figure 3a). The source vessel fires the seismic source on a dense regular grid, consisting of 799×799 source locations (638,401 in total, Figure 3b). For modeling, source-receiver reciprocity is applied, which means that sources and receivers are interchangeable and data can be sorted into 1,500 shot records with 638,401 receivers each, which reduces the number of individual PDE solves to 1,500. For the numerical wave propagation, this tutorial provides Devito implementations of the pseudo-acoustic TTI wave equation with an 8th order finite-difference discretization in space and a 2nd order stencil in time.

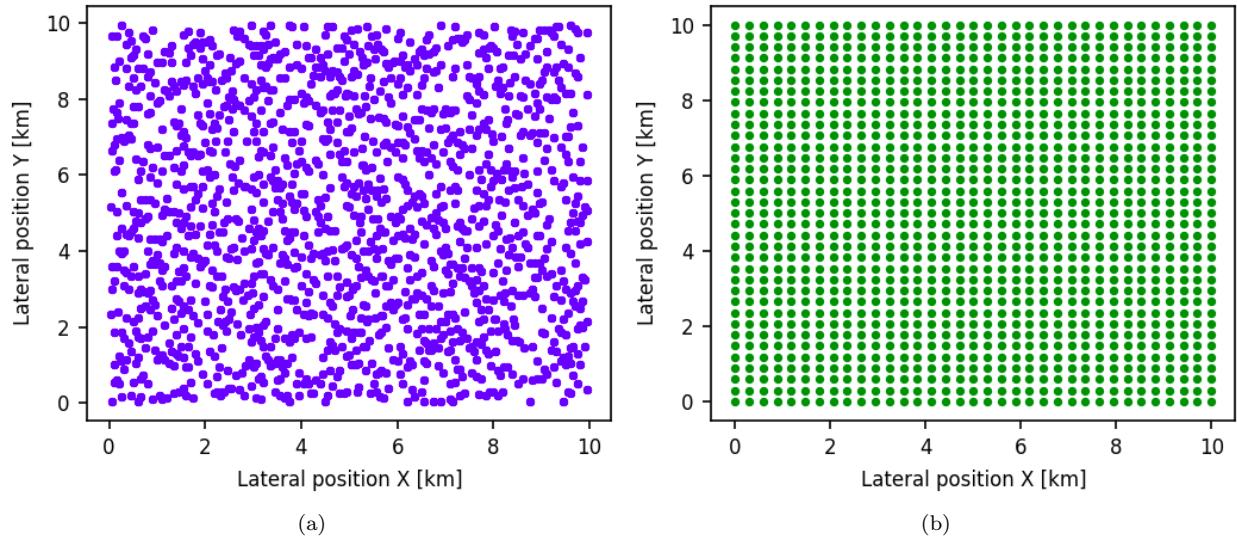


Figure 3: Randomized receiver grid with 1,500 ocean-bottom nodes (a) and dense source grid with 799 x 799 shot locations (b). For modeling, source-receiver reciprocity is applied, resulting in 1,500 shot location with a dense receiver grid.

5 Prerequisites

- Install Azure CLI, Batch Shipyard
- Optional: Batch Explorer
- (Or run everything from George’s docker container. Will it be made public?)

6 Azure Set up

6.1 Docker container

- Optional: Build Docker containers and upload to Dockerhub
- Alternatively: Use pre-built containers

6.2 Upload user files

- Upload model + geometry
- Upload Python script

6.3 Batch shipyard configuration

- Computational set up: HBv2 nodes, 120 CPU cores, 480 GB memory.
- Use one full node per source location. Each modeling task has 30 MPI ranks with 4 OMP threads each (i.e. we use all 120 cores per node).
- Pool size: 100 nodes.

- Job: Model xx of the 1,500 shots.
- How to fill out config files (`credentials.yaml`, `config.yaml`, `pool.yaml`, `jobs.yaml`)

7 Submit and monitor a job

- Start small pool: `./shipyard pool add -v` (then resize)
- Submit job: `./shipyard jobs add --tail stdout.txt -v`
- Check `stderr.txt` for Devito output
- Connect to compute nodes and run `top`
- Monitor CPU usage + memory in Batch Explorer

8 Performance analysis and results

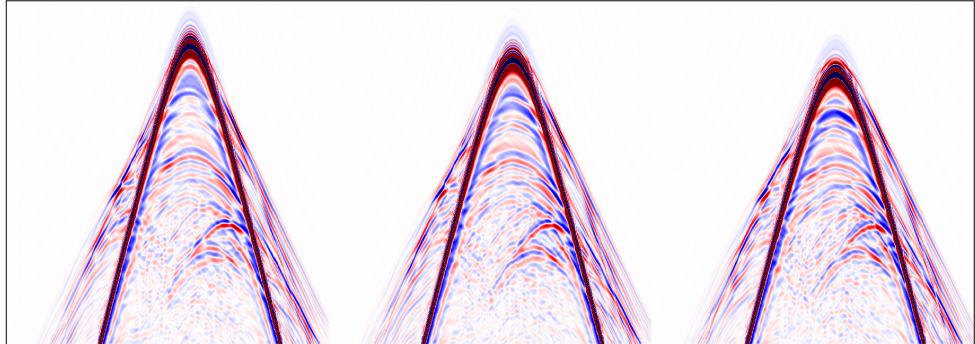
- Show scalability
- Plot performance, FLOPs
- Download + plot seismic data

9 Clean up

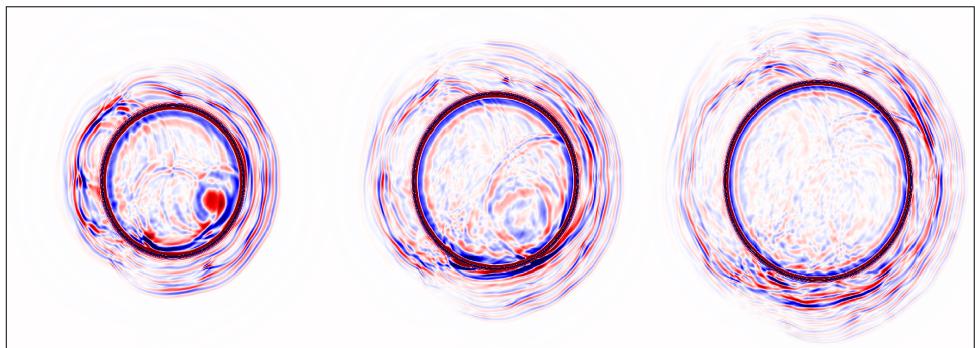
- Kill job: `./shipyard jobs del -v`
- Shut down pool: `./shipyard pool del -v`

10 Next steps

Numerical seismic modeling functions as a key ingredient to a broad variety of applications, including subsurface imaging, parameter estimation or monitoring of geohazards. As such, the seismic modeling workflow demonstrated in this tutorial can function as a building block for more sophisticated workflows, which rely on forward modeling as the underlying workhorse. For example, instead of forward propagating a seismic source to model seismic data, recorded data from a seismic survey can be backpropagated in time by solving an adjoint wave equation, which results in a seismic image of the subsurface. An example of seismic imaging using a combination of Azure Batch and Azure Functions can be found [here](#). Furthermore, the forward modeling module from this tutorial can function as a building block for iterative workflows, such as full-waveform inversion or least-squares imaging.



(a)



(b)

Figure 4: Crossline samples of a 3D modeled seismic shot record (a) and time slices at increasing recording times (b).