

Serverless HPC in the cloud: A seismic imaging case study

Philipp Witte



Georgia Institute of Technology



Felix J. Herrmann[†]



Charles Jones^{†‡}



Henryk Modzelewski*



Mathias Louboutin[†]



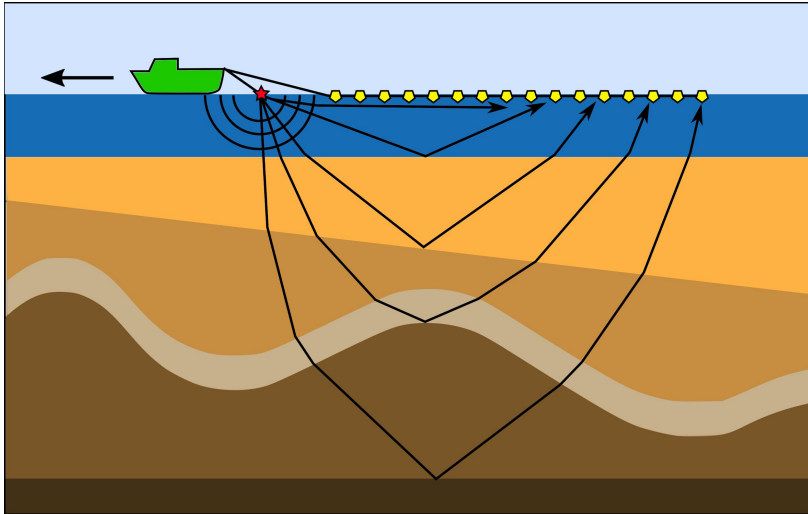
James Selvage^{†‡}

SLIM 

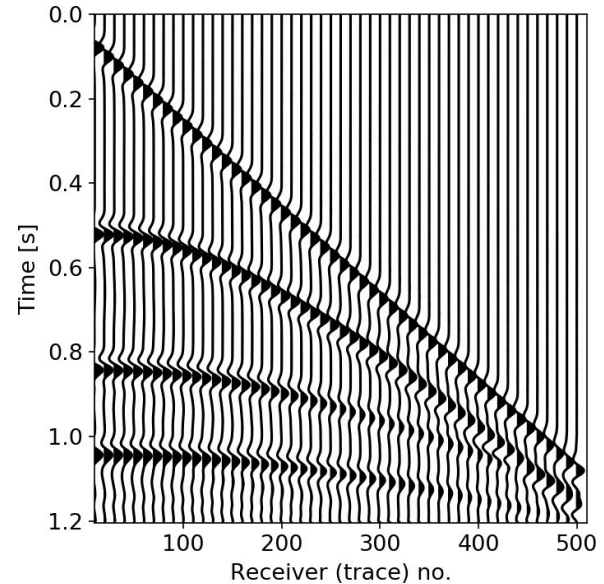


Introduction

Seismic exploration for subsurface imaging and parameter estimation:

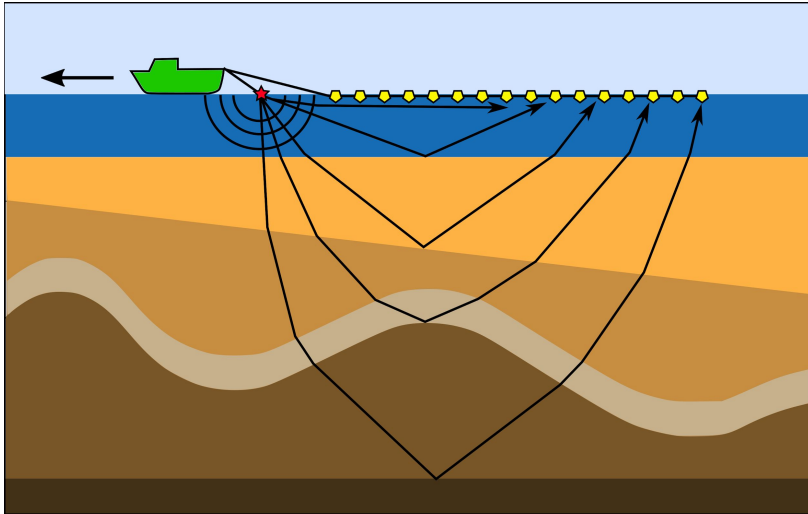


Seismic shot record
(wiggly plot)

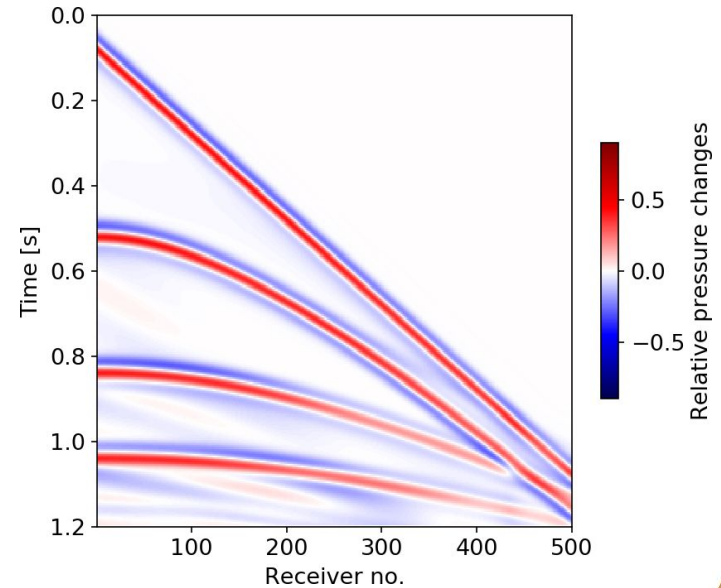


Introduction

Seismic exploration for subsurface imaging and parameter estimation:

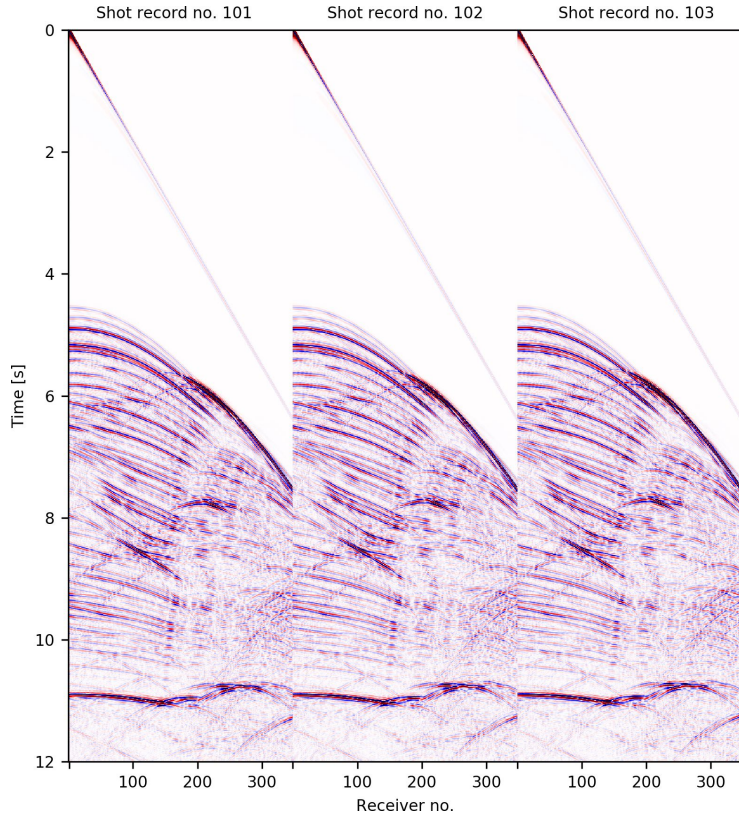


Seismic shot record
(image plot)



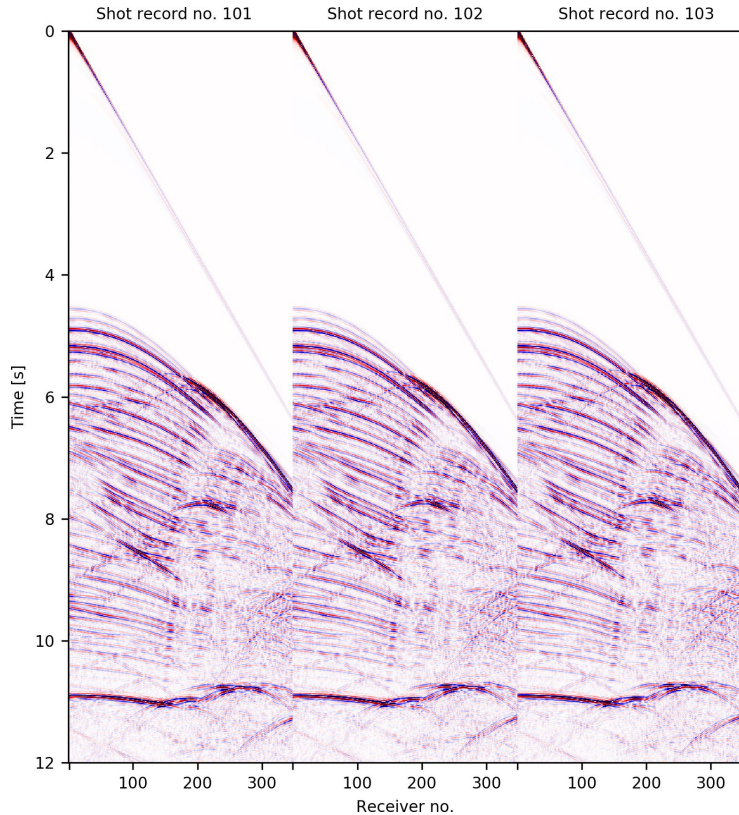
Introduction

Repeat for many source locations:

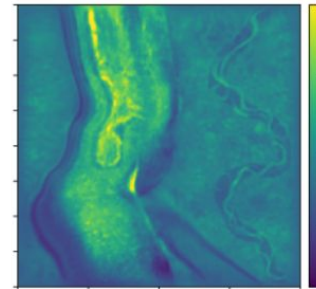
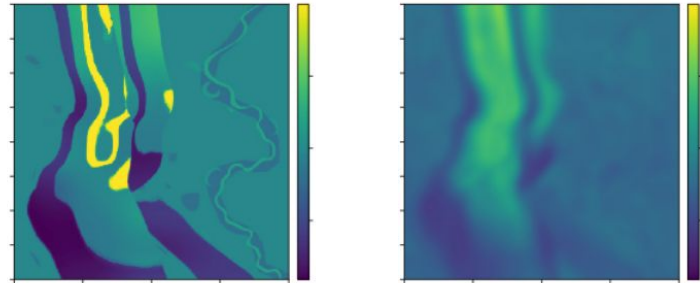


Introduction

Nonlinear seismic parameter estimation:

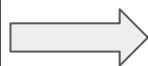
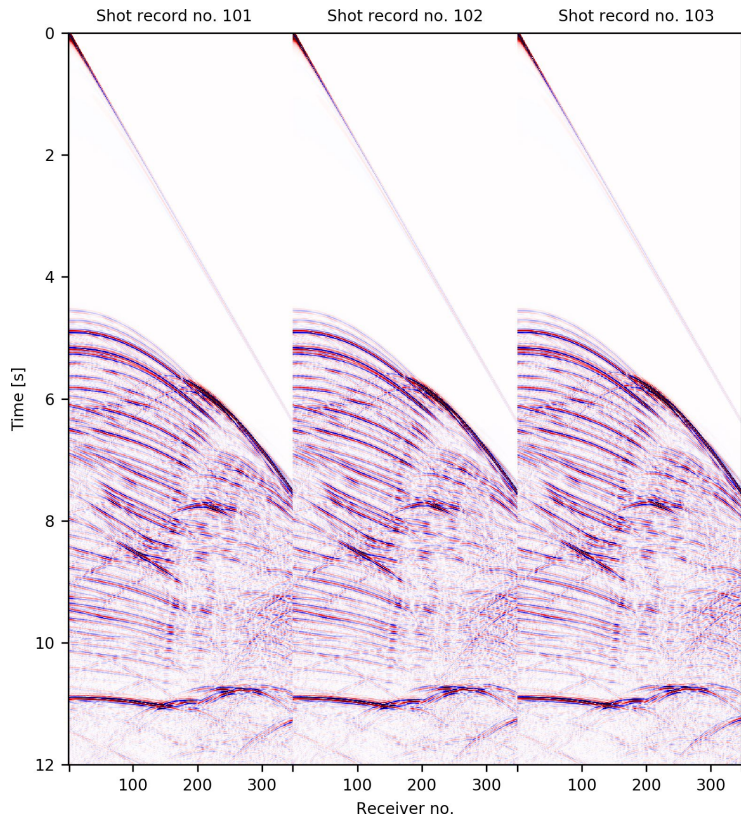


$$\underset{\mathbf{m}}{\text{minimize}} \quad \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$

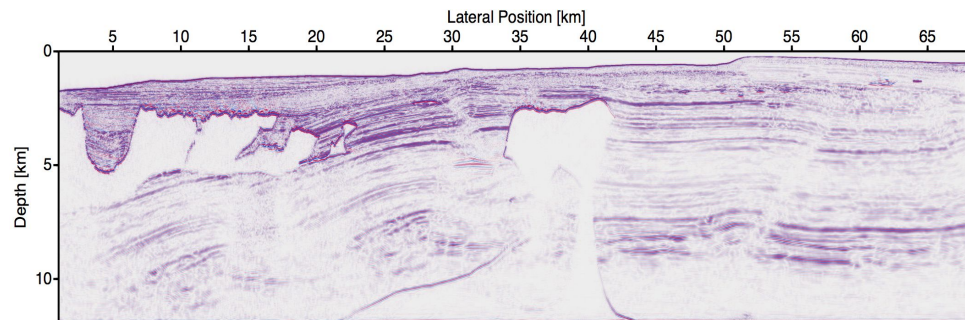


Introduction

Linearized inversion: Seismic imaging



$$\underset{\delta \mathbf{m}}{\text{minimize}} \quad \Phi(\delta \mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathbf{J}(\mathbf{m}, \mathbf{q}_i) \delta \mathbf{m} - \mathbf{d}_i\|_2^2$$



**Least squares reverse-time migration
(LS-RTM)**

Introduction

Challenges of seismic inverse problems:

- Large number of individual experiments

$$\underset{\mathbf{m}}{\text{minimize}} \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$

Introduction

Challenges of seismic inverse problems:

- Large number of individual experiments
- Need to solve expensive wave equations

$$\underset{\mathbf{m}}{\text{minimize}} \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$

Introduction

Challenges of seismic inverse problems:

- Large number of individual experiments
- Need to solve expensive wave equations
- Very high-dimensional problem (millions to billions of unknown parameters)

$$\underset{\mathbf{m}}{\text{minimize}} \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$

Introduction

Challenges of seismic inverse problems:

- Large number of individual experiments
- Need to solve expensive wave equations
- Very high-dimensional problem (millions to billions of unknown parameters)
- Large data sets (order of TB)

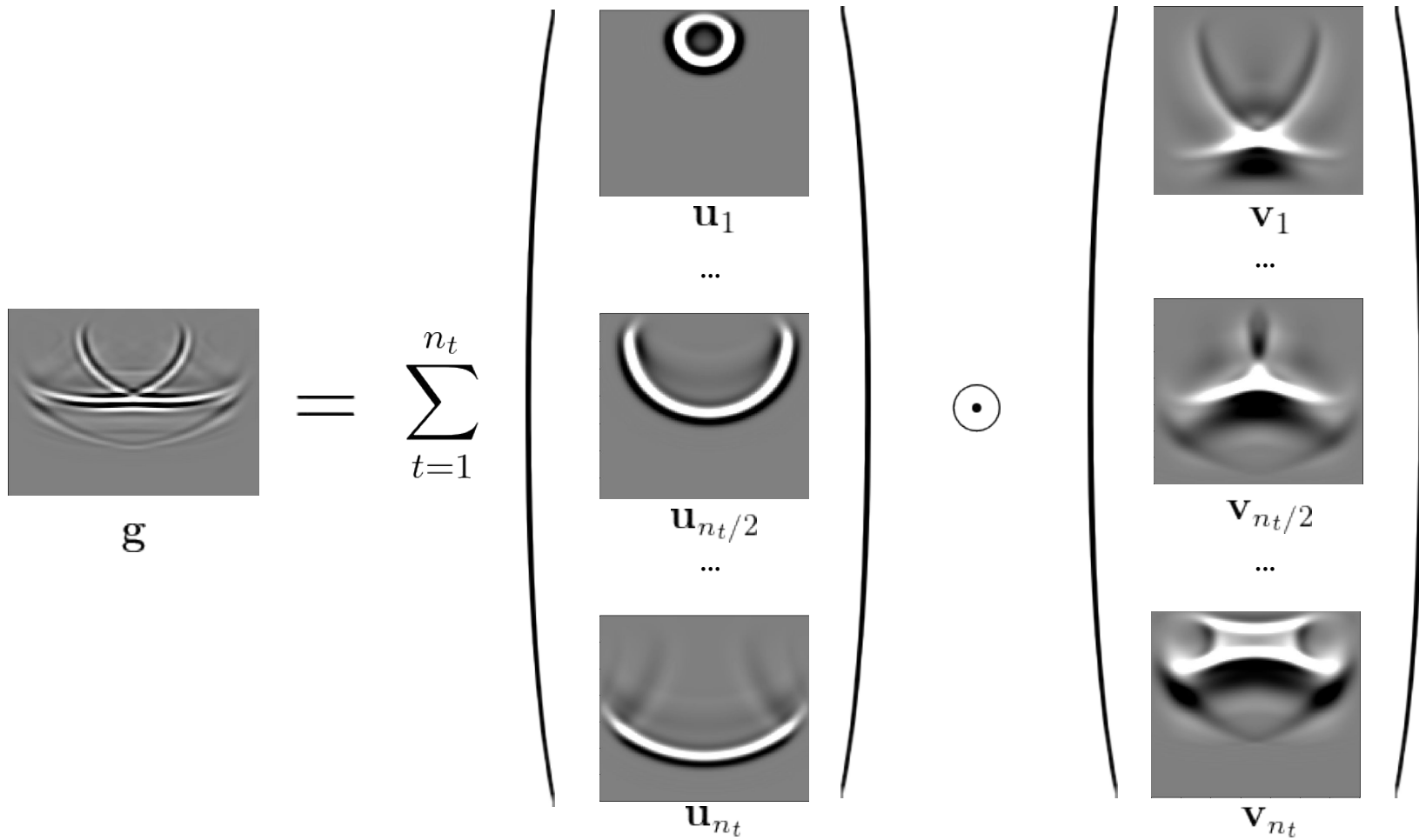
$$\underset{\mathbf{m}}{\text{minimize}} \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$

Introduction

Challenges of seismic inverse problems:

- Large number of individual experiments
- Need to solve expensive wave equations
- Very high-dimensional problem (millions to billions of unknown parameters)
- Large data sets (order of TB)
- Compute gradients via backpropagation

$$\underset{\mathbf{m}}{\text{minimize}} \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$



Introduction

Challenges of seismic inverse problems:

- Large number of individual experiments
 - Need to solve expensive wave equations
 - Very high-dimensional problem (millions to billions of unknown parameters)
 - Large data sets (order of TB)
 - Compute gradients via backpropagation
- > need ~TB of memory or checkpointing

$$\underset{\mathbf{m}}{\text{minimize}} \Phi(\mathbf{m}) = \sum_{i=1}^{n_s} \frac{1}{2} \|\mathcal{F}(\mathbf{m}, \mathbf{q}_i) - \mathbf{d}_i\|_2^2$$

Bottom line: Need access to HPC resources

BP upgrades Houston HPC, world's most powerful corporate supercomputer

Nine petaflop machine built by HPE and Intel

December 18, 2017 By: Sebastian Moss

PGS POWERS THEIR INFRASTRUCTURE WITH CRAY SUPERCOMPUTING AND STORAGE

Cray will provide Petroleum Geo Services (PGS) with a five-petaflop Cray® XC40™ supercomputer and Cray® Sonexion® Lustre® file system.

ExxonMobil sets record in high-performance oil and gas reservoir computing

Combining software, engineering and geoscience skills, our advanced technical and computational capabilities offer a competitive advantage in resource analysis and decision-making.

Sep. 13, 2018

Introduction

Seismic imaging and parameter estimation:

- Need access to HPC resources
- Only available to few corporations, academic institutions
- Cloud as possible alternative?
- How does the cloud compare to HPC clusters?
(performance, cost, resilience, etc.)
- How can software be deployed to the cloud?

Outline

1. HPC clusters vs. the cloud
2. An event-driven approach to serverless seismic imaging
 - a. Problem formulation
 - b. Workflow components
3. Performance analysis:
 - a. Weak scaling
 - b. Strong scaling
 - c. Cost
 - d. Resilience
4. Case study:
 - a. 3D Seismic imaging on Azure
5. Discussion: HPC in the cloud - feasible and worth it?

Outline

- 1. HPC clusters vs. the cloud**
- An event-driven approach to serverless seismic imaging
 - a. Problem formulation
 - b. Workflow components
- Performance analysis:
 - a. Weak scaling
 - b. Strong scaling
 - c. Cost
 - d. Resilience
- Case study:
 - a. 3D Seismic imaging on Azure
- Discussion: HPC in the cloud - feasible and worth it?

Seismic inversion on HPC clusters

Conventional compute environment: **HPC clusters**



✓ Pros

- Best achievable performance
- 40+ years of experience and existing software
- Low mean-time-between failures (MTBF)
- Very fast inter-node connections possible (Infiniband)



✗ Cons

- Very high upfront + maintenance costs
- Only available to few companies + academic institutions
- Compromises regarding hardware (architecture/CPU/GPU/RAM)

Seismic inversion in the cloud

Cloud computing



Google Cloud Platform

✓ Pros

- Theoretically unlimited scalability
- High flexibility (hardware, jobs)
- No upfront + maintenance costs: pay-as-you-go
- Available to anyone
- Latest hardware and architectures available (GPUs, ARM)

✗ Cons

- Slower inter-node connections (depending on platform)
- Oftentimes larger MTBF
- High costs if not used properly
- Need to transition software
- Steep learning curve

Moving to the cloud

Cloud performance analysis:

- Many cloud benchmarking papers (but oftentimes outdated) (e.g. Garfinkel 2007; Jackson et al., 2009; Iosup et al., 2011; Benedict 2013; Mehrotra et al., 2016)
- Large latency and generally lower bandwidth
- Higher mean time between failures and high cost
- Good performance for embarrassingly parallel applications (Gupta et al., 2011; Sadooghi et al.; 2017; Kotas et al., 2018)
- Good performance on single cloud nodes/bare metal instances (Dongarra et al. 2003; Rad et al., 2015; Mohammadi et al., 2018)
- None of the benchmarks use cloud specific software

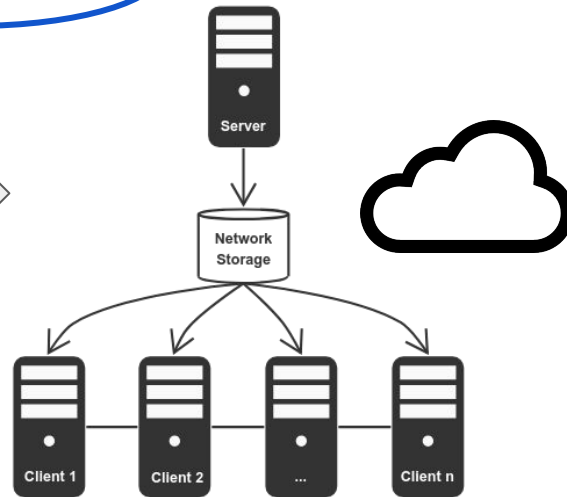
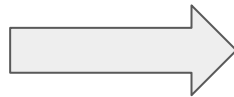
Moving to the cloud

```
#include <inttypes.h>
#include <sys/time.h>
#include <math.h>
struct profiler
{
    double loop_stencils_a;
    double loop_body;
    double kernel;
};
struct flops
{
    long long loop_stencils_a;
    long long loop_body;
    long long kernel;
};
extern "C" int ForwardOperator(double *m_vec, double *u_vec, double *damp_vec, double *src_vec, float
*src_coords_vec, double *rec_vec, float *rec_coords_vec, long lblock, struct profiler *timings, struct flops *flops)
{
    double (*m)[280] = (double (*)[280]) m_vec;
    double (*u)[280][280] = (double (*)[280][280]) u_vec;
    double (*damp)[280] = (double (*)[280]) damp_vec;
    double (*src)[2] = (double (*)[2]) src_vec;
    float (*src_coords)[2] = (float (*)[2]) src_coords_vec;
    double (*rec)[101] = (double (*)[101]) rec_vec;
    float (*rec_coords)[2] = (float (*)[2]) rec_coords_vec;
    {
        struct timeval start_kernel, end_kernel;
        gettimeofday(&start_kernel, NULL);
        int t0;
        int t1;
        int t2;
        ;
        for (int i3 = 0; i3<3; i3+=1)
        {
            flops->kernel += 2.000000;
            {
                t0 = (i3)%3;
                t1 = (t0 + 1)%3;
                t2 = (t1 + 1)%3;
            }
        }
    }
}
```

Legacy Fortran
or C code



Lift and shift



Moving to the cloud

```

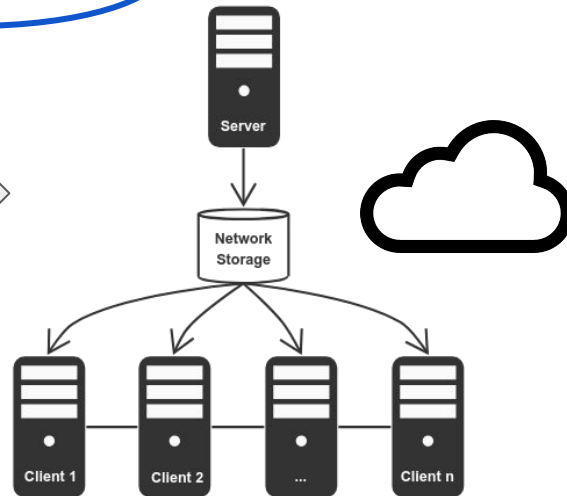
#include <inttypes.h>
#include <sys/time.h>
#include <math.h>
struct profiler
{
    double loop_stencils_a;
    double loop_body;
    double kernel;
};
struct flops
{
    long long loop_stencils_a;
    long long loop_body;
    long long kernel;
};
extern "C" int ForwardOperator(double *u_vec, double *u_vec, double *damp_vec, double *src_vec, float
*src_coords_vec, double *rec_vec, float *rec_coords_vec, long lblock, struct profiler *timings, struct flops *flops)
{
    double (*m)[280] = (double (*)[280]) m_vec;
    double (*u)[280][280] = (double (*)[280][280]) u_vec;
    double (*damp)[280] = (double (*)[280]) damp_vec;
    double (*src)[2] = (double (*)[2]) src_vec;
    float (*src_coords)[2] = (float (*)[2]) src_coords_vec;
    double (*rec)[101] = (double (*)[101]) rec_vec;
    float (*rec_coords)[2] = (float (*)[2]) rec_coords_vec;
    {
        struct timeval start_kernel, end_kernel;
        gettimeofday(&start_kernel, NULL);
        int t0;
        int t1;
        int t2;
        ;
        for (int i3 = 0; i3<3; i3+=1)
        {
            flops->kernel += 2.000000;
            {
                t0 = i3*(3);
                t1 = (t0 + 1)*(3);
                t2 = (t1 + 1)*(3);
            }
        }
    }
}

```

Legacy Fortran
or C code



Lift and shift



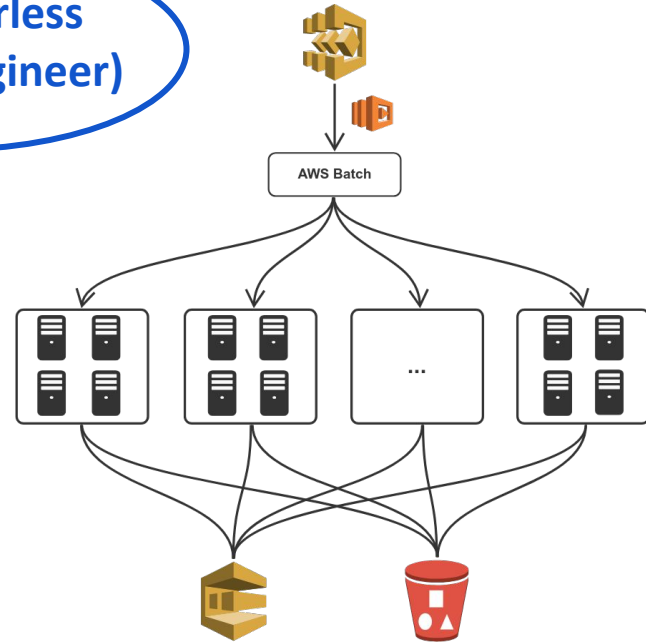
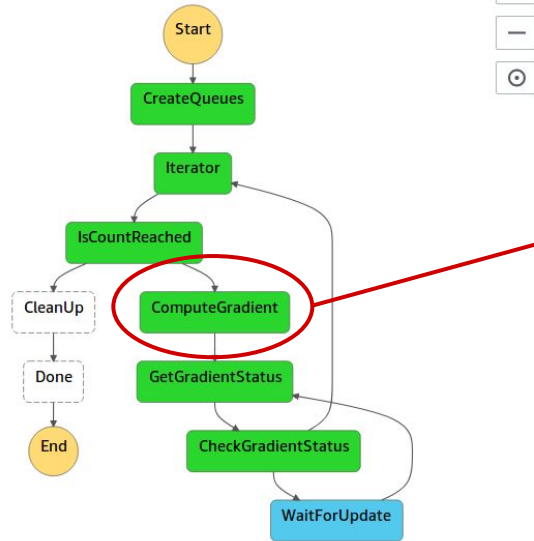
- Requires little to no work
- Long cluster start-up time and cost
- Idle instances/resilience/bandwidth/etc.
- Technically infeasible for industry scale

Moving to the cloud

Go serverless
(and re-engineer)

Visual workflow

■ Success ■ Failed ■ Cancelled ■ In Progress

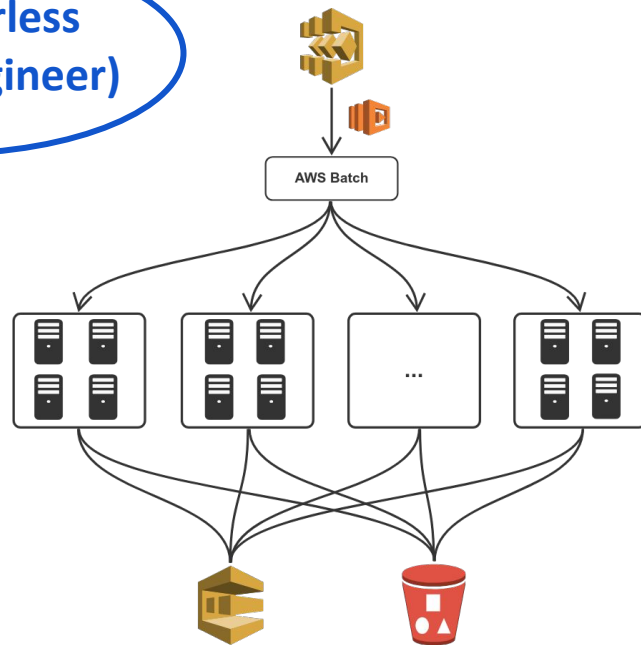
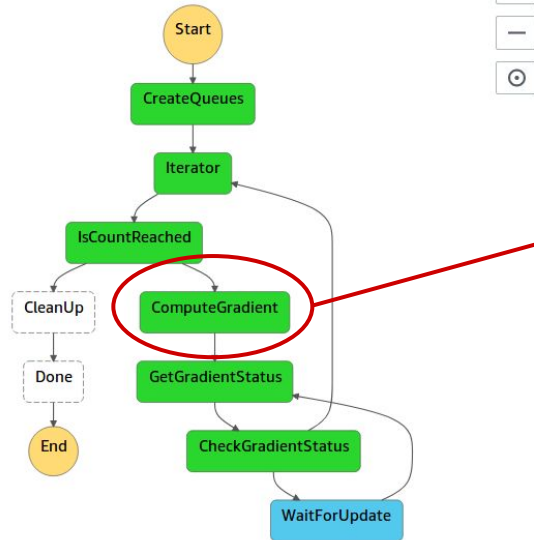


Moving to the cloud

Go serverless
(and re-engineer)

Visual workflow

■ Success ■ Failed ■ Cancelled ■ In Progress



- Save cost (up to **10x**): no idle instances, lower start-up time
- Resilience managed by cloud platform
- Requires re-engineering of software

Moving to the cloud

Performance in the cloud:

- Worse bandwidth, latency + resilience
- Problematic for long running applications (e.g. seismic imaging)

Cloud specific tools/technologies:

- Cloud object storage, containerized batch computing, event-driven computations, etc.
- How can we design software that takes advantage of novel cloud technologies and help control cost?



Need to redesign software (application dependent)

Outline

1. HPC clusters vs. the cloud
- 2. An event-driven approach to serverless seismic imaging**
 - a. Problem formulation
 - b. Workflow components
3. Performance analysis:
 - a. Weak scaling
 - b. Strong scaling
 - c. Cost
 - d. Resilience
4. Case study:
 - a. 3D Seismic imaging on Azure
5. Discussion: HPC in the cloud - feasible and worth it?

Serverless LS-RTM in the cloud

Typical components of LS-RTM*:
$$\underset{\delta \mathbf{m}}{\text{minimize}} \sum_{i=1}^{n_s} \frac{1}{2} \left\| \mathbf{J}(\mathbf{m}, \mathbf{q}_i) \delta \mathbf{m} - \mathbf{d}_i^{\text{obs}} \right\|_2^2$$

Serverless LS-RTM in the cloud

Typical components of LS-RTM*:
$$\underset{\delta \mathbf{m}}{\text{minimize}} \sum_{i=1}^{n_s} \frac{1}{2} \left\| \mathbf{J}(\mathbf{m}, \mathbf{q}_i) \delta \mathbf{m} - \mathbf{d}_i^{\text{obs}} \right\|_2^2$$

for $\mathbf{j} = 1, \dots, \mathbf{n}$

1. Compute gradient for all/subset of source locations: $\mathbf{g}_i = \mathbf{J}^\top \left(\mathbf{J} \delta \mathbf{m} - \mathbf{d}_i^{\text{obs}} \right)$

2. Sum gradients: $\mathbf{g} = \sum_{i=1}^{n_b} \mathbf{g}_i$

3. Update image based on optimization algorithm (SGD, CG, Adam, etc.):

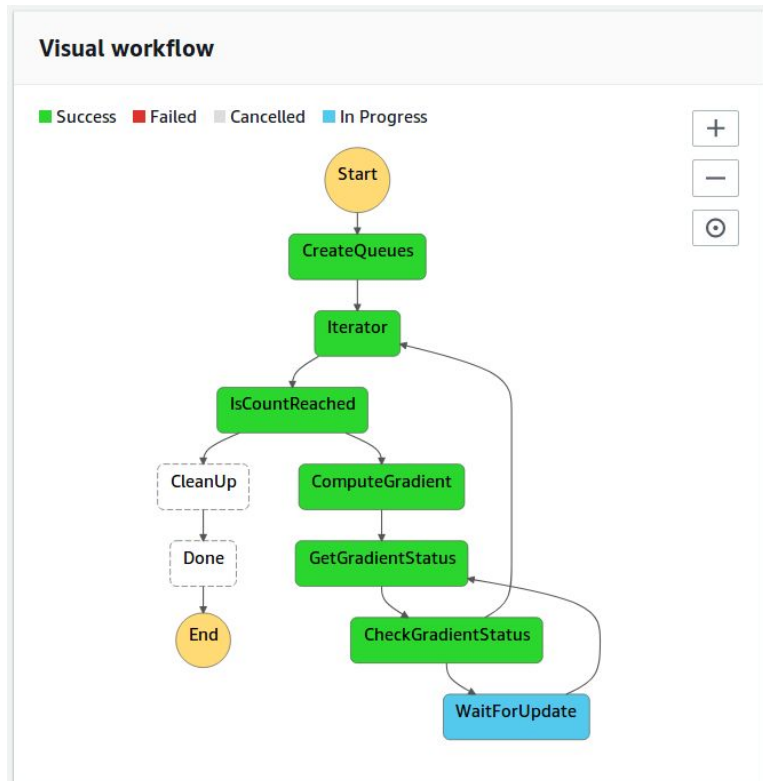
$$\delta \mathbf{m} = \delta \mathbf{m} - \alpha \mathbf{g}$$

end

Serverless LS-RTM in the cloud

Serverless workflow with Step Functions:

- Algorithm as collection of *states**
- No compute instances required to execute workflow (i.e. *serverless*)
- States invoke AWS Lambda functions to run Python code
- Lambda functions: upload + run code w/o resource allocation



*Friedmann and Pizarro, AWS Compute Blog, 2017

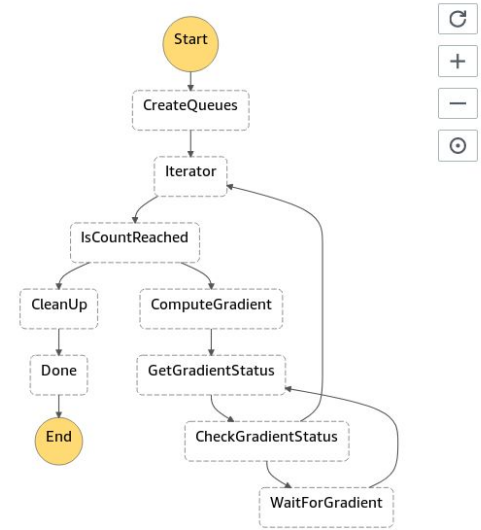
Serverless LS-RTM in the cloud

State machine defined as json file

Definition

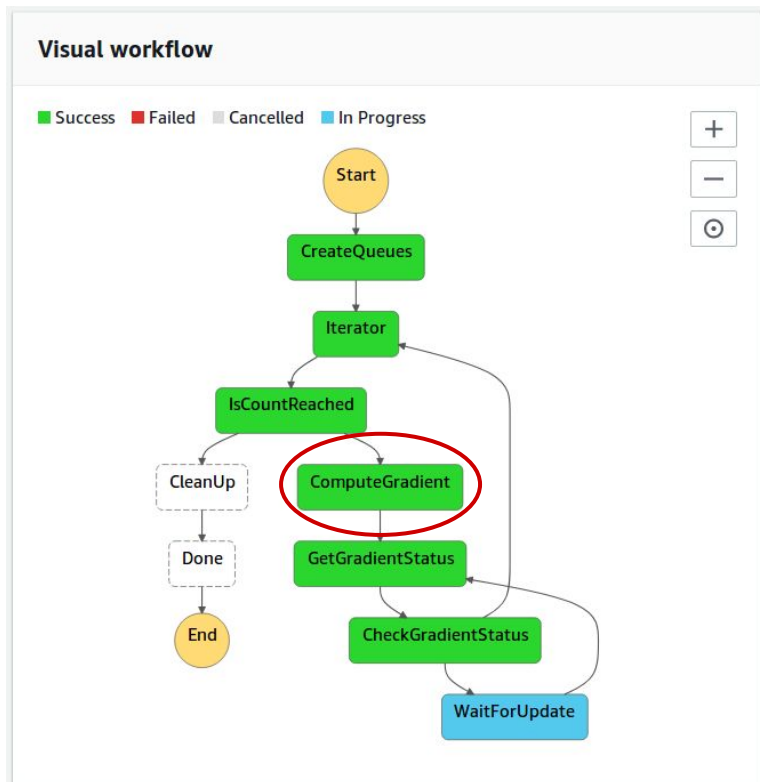
Generate code snippet [Learn more](#)

```
1 {
2   "Comment": "Iterator State Machine Example",
3   "StartAt": "CreateQueues",
4   "States": {
5     "CreateQueues": {
6       "Comment": "Create SQS queues and lambda triggers for the gradient reduction",
7       "Type": "Task",
8       "Resource": "arn:aws:lambda:us-east-1:851065145468:function:CreateQueues",
9       "ResultPath": "$",
10      "Next": "Iterator"
11    },
12    "Iterator": {
13      "Type": "Task",
14      "Resource": "arn:aws:lambda:us-east-1:851065145468:function:IteratorStochastic",
15      "ResultPath": "$",
16      "Next": "IsCountReached"
17    },
18    "IsCountReached": {
19      "Type": "Choice",
20      "Choices": [
21        {
22          "Variable": "$.iterator.continue",
23          "BooleanEquals": true,
24          "Next": "ComputeGradient"
25        }
26      ],
27      "Default": "CleanUp"
28    },
29    "ComputeGradient": {
30      "Comment": "Your application logic, to run a specific number of times",
31      "Type": "Task",
32      "Resource": "arn:aws:lambda:us-east-1:851065145468:function:ComputeGradient",
33      "ResultPath": "$",
34      "Next": "GetGradientStatus"
35    },
36    "GetGradientStatus": {
37      "Type": "Task",
38      "Resource": "arn:aws:lambda:us-east-1:851065145468:function:GetGradientStatus",
39      "ResultPath": "$",
40      "Next": "CheckGradientStatus"
41    },
42    "CheckGradientStatus": {
43      "Type": "Task",
44      "Resource": "arn:aws:lambda:us-east-1:851065145468:function:CheckGradientStatus",
45      "ResultPath": "$",
46      "Next": "WaitForGradient"
47    },
48    "WaitForGradient": {
49      "Type": "Task",
50      "Resource": "arn:aws:lambda:us-east-1:851065145468:function:WaitForGradient",
51      "ResultPath": "$",
52      "Next": "CheckGradientStatus"
53    },
54    "CleanUp": {
55      "Type": "Task",
56      "Resource": "arn:aws:lambda:us-east-1:851065145468:function:CleanUp",
57      "ResultPath": "$",
58      "Next": "Done"
59    },
60    "Done": {
61      "Type": "End"
62    }
63  }
64 }
```



```
graph TD
  Start((Start)) --> CreateQueues[CreateQueues]
  CreateQueues --> Iterator[Iterator]
  Iterator --> IsCountReached[IsCountReached]
  IsCountReached --> CleanUp[CleanUp]
  IsCountReached --> ComputeGradient[ComputeGradient]
  CleanUp --> Done((Done))
  Done --> End((End))
  ComputeGradient --> GetGradientStatus[GetGradientStatus]
  GetGradientStatus --> CheckGradientStatus[CheckGradientStatus]
  CheckGradientStatus --> WaitForGradient[WaitForGradient]
  WaitForGradient --> CheckGradientStatus
  CheckGradientStatus --> Iterator
```

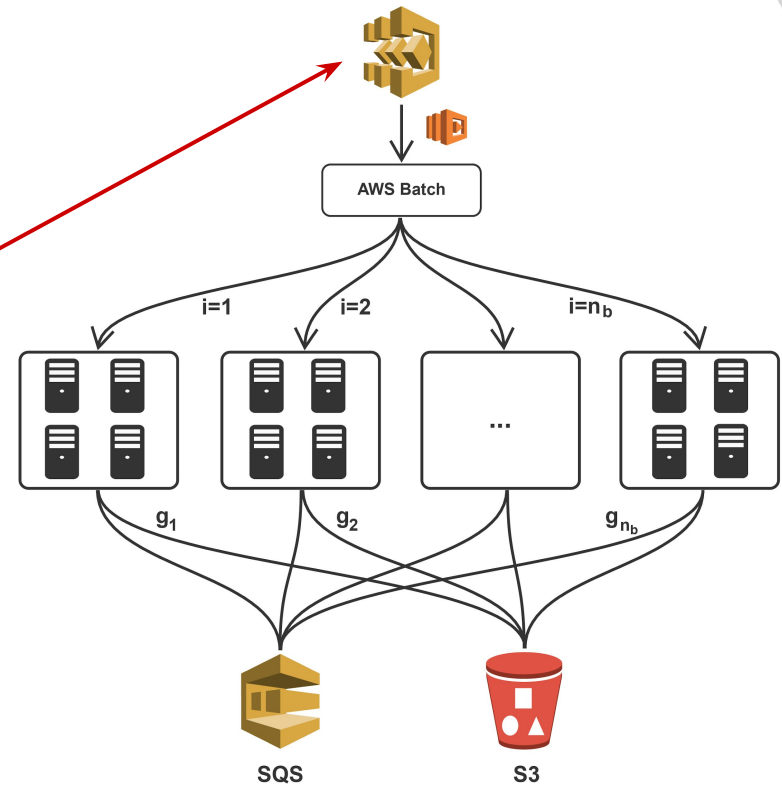
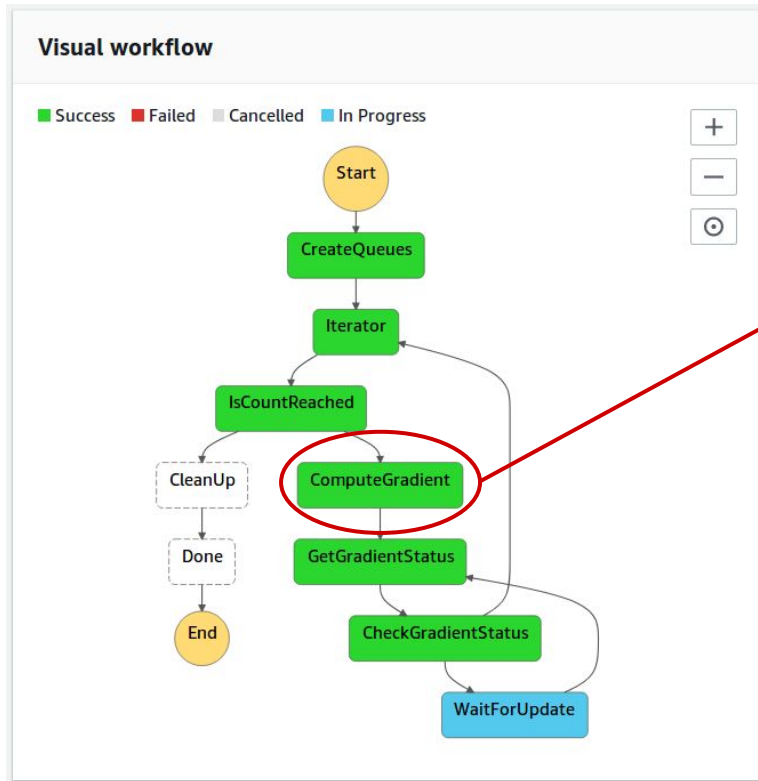
Gradient computations



Compute gradients of the LS-RTM objective function:

- embarrassingly parallel
- model predicted data + backpropagate residual + imaging condition
- compute/memory heavy process (store/recompute wavefields)

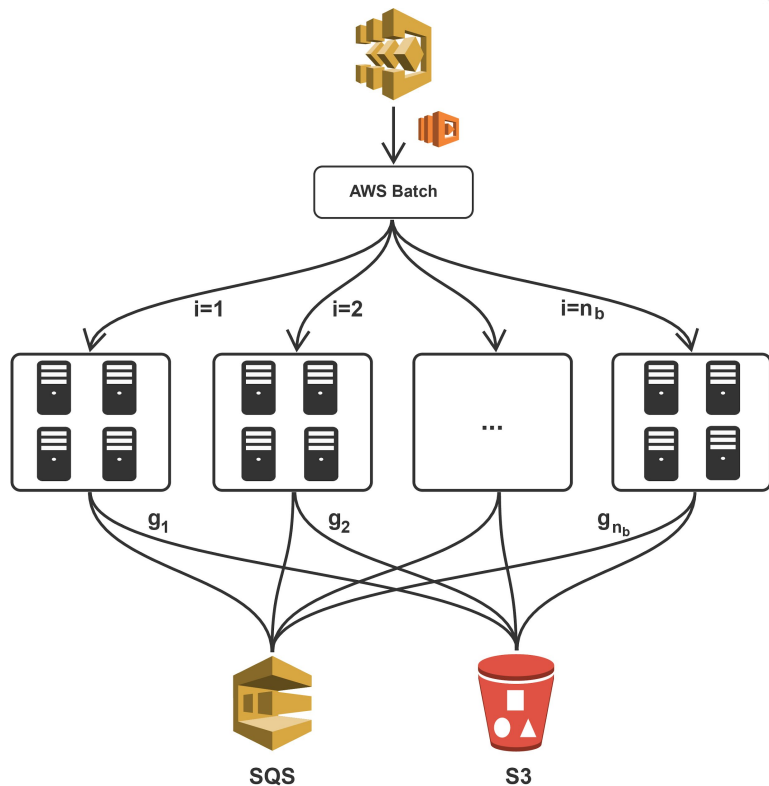
Gradient computations



Gradient computations

Nested levels of parallelization:

- Parallelize sum over sources (AWS Batch)
- Domain decomposition (MPI)
- Multithreading (OpenMP)
- Each gradient computed on individual instance **or** cluster of instances (cluster of clusters)



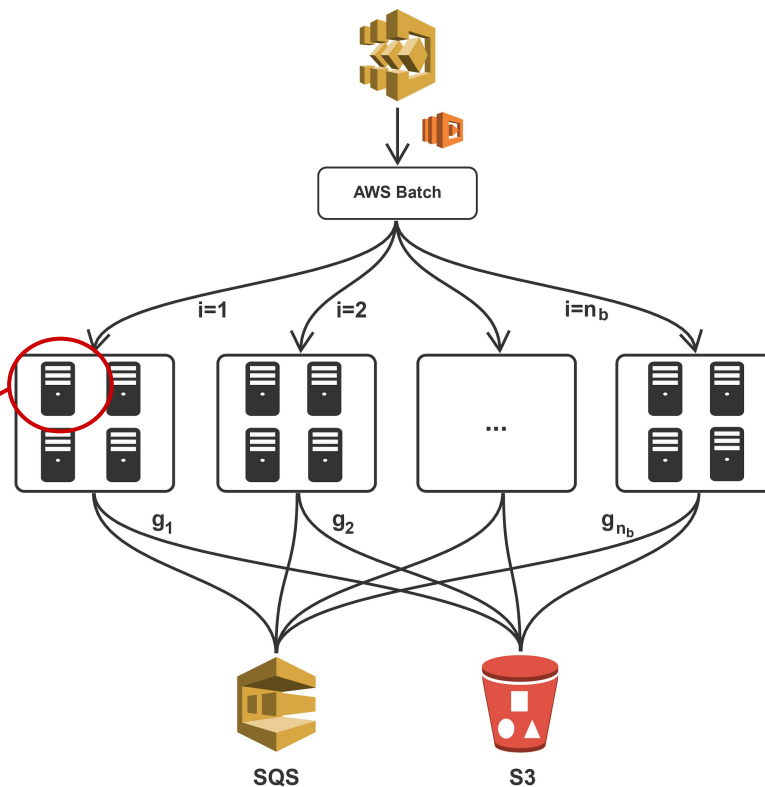
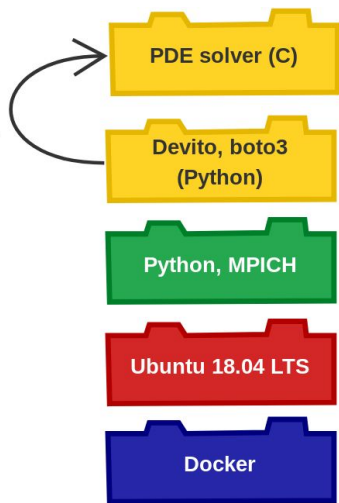
Gradient computations

* Luporini et al., 2018; Louboutin et al., 2019

Software to compute gradients:

- Batch runs docker containers
- Solve wave equations using Devito*
- Automated performance optimizations

performance
optimization +
JIT compilation



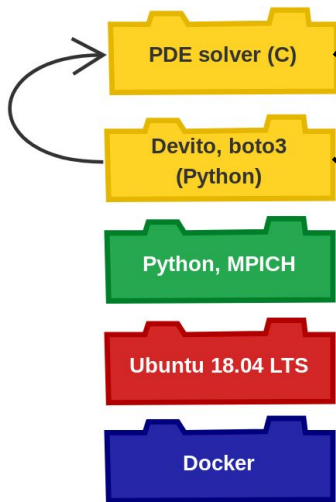
Gradient computations

* Luporini et al., 2018; Louboutin et al., 2019

Devito:

- Automatic C code generation
- Loop blocking, vectorization, refactoring, OMP, MPI, etc.

performance optimization + JIT compilation



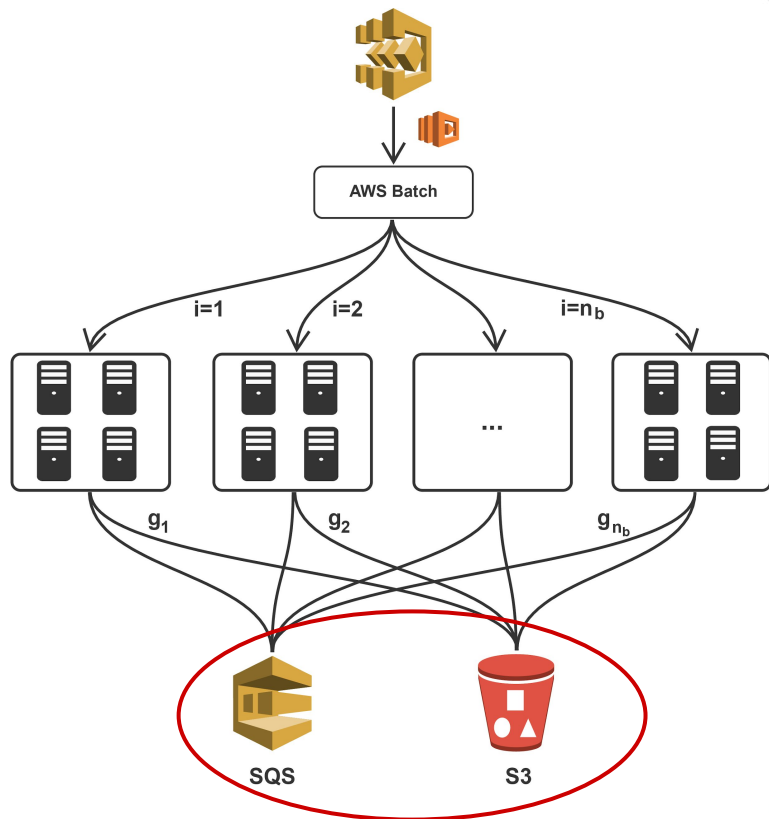
```
#include <inttypes.h>
#include <sys/time.h>
#include <math.h>
struct profiler
{
    double loop_stencils_a;
    double loop_body;
    double kernel;
};
struct flops
{
    long long loop_stencils_a;
    long long loop_body;
    long long kernel;
};
extern "C" int ForwardOperator(double *m_vec, double *u_vec, double *damp_vec, double *src_vec, float
*src_coords_vec, double *rec_vec, float *rec_coords_vec, long i1block, struct profiler *timings, struct flops *flops)
{
    double (*m)[280] = (double (*)[280]) m_vec;
    double (*u)[280][280] = (double (*)[280][280]) u_vec;
    double (*damp)[280] = (double (*)[280]) damp_vec;
    double (*src)[2] = (double (*)[2]) src_vec;
    float (*src_coords)[2] = (float (*)[2]) src_coords_vec;
    double (*rec)[101] = (double (*)[101]) rec_vec;
    float (*rec_coords)[2] = (float (*)[2]) rec_coords_vec;
    {
        struct timeval start_kernel, end_kernel;
        gettimeofday(&start_kernel, NULL);
        int t0;
        int t1;
        int t2;
        ;
        for (int i3 = 0; i3<3; i3+=1)
        {
            flops->kernel += 2.000000;
            {
                t0 = (i3)%3;
                t1 = (t0 + 1)%3;
                t2 = (t1 + 1)%3;
            }
        }
    }
}
```

$$\text{pde} = \text{model.m} * \text{u} . \text{dt}^2 - \text{u} . \text{laplace} + \text{model.damp} * \text{u} . \text{dt}$$

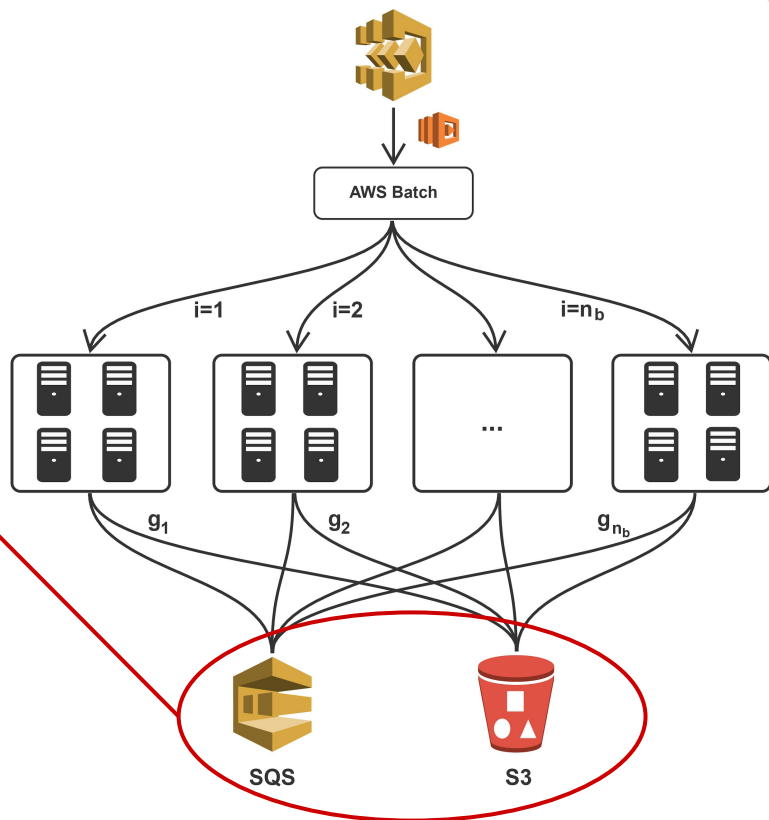
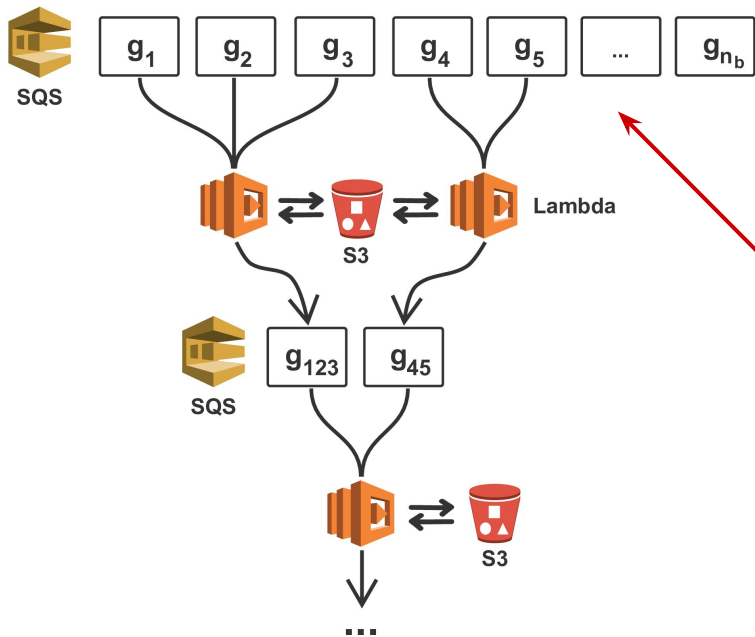
Gradient computations

Summation of gradients

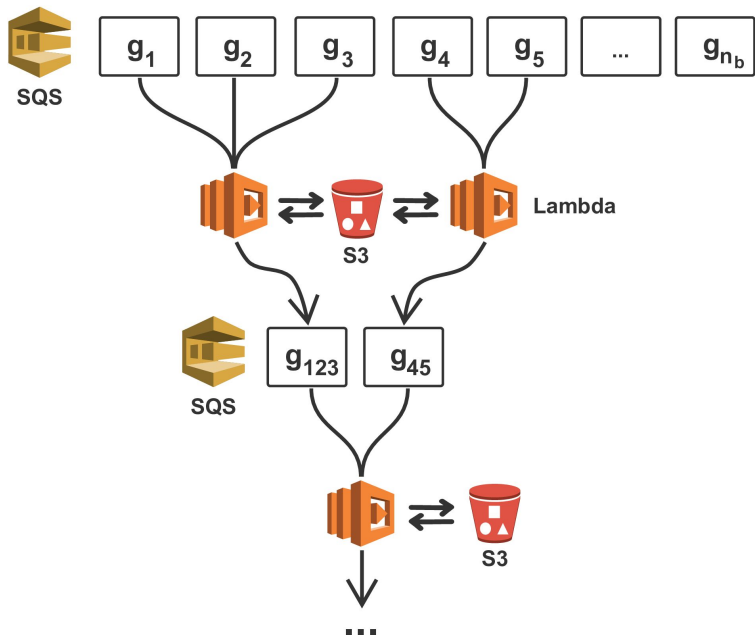
- Gradients stored in object storage (S3)
- Virtually unlimited I/O scalability
- Send object IDs to message queue
- Event-driven gradient summation using Lambda functions



Gradient computations



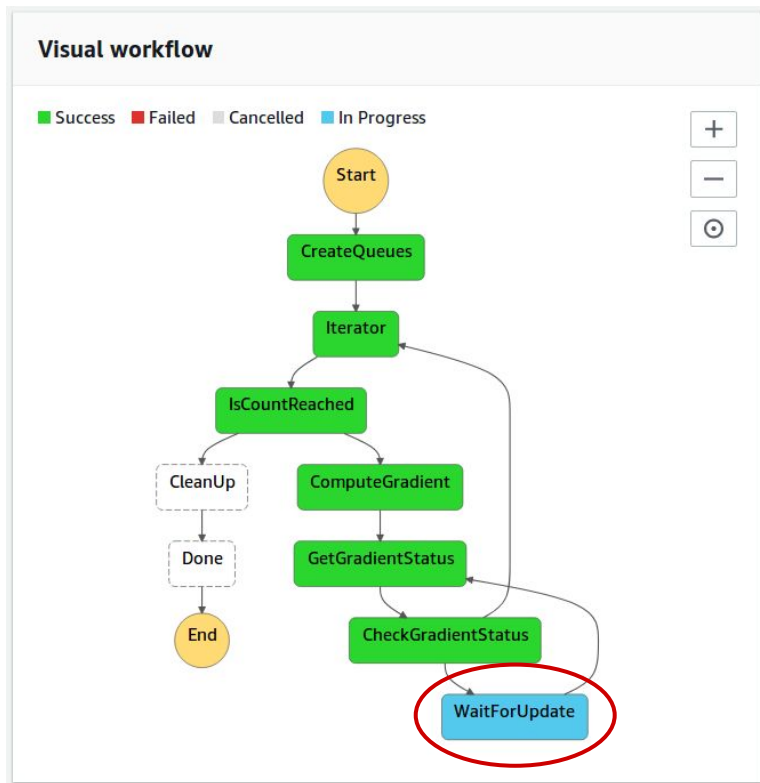
Gradient computations



Event-driven gradient reduction

- AWS Lambda functions
- Cheaper than compute nodes
- Asynchronous and parallel
- Invoked as soon as at least 2 gradients are available
- Stream gradients from S3 -> sum -> write back
- Update image after final summation

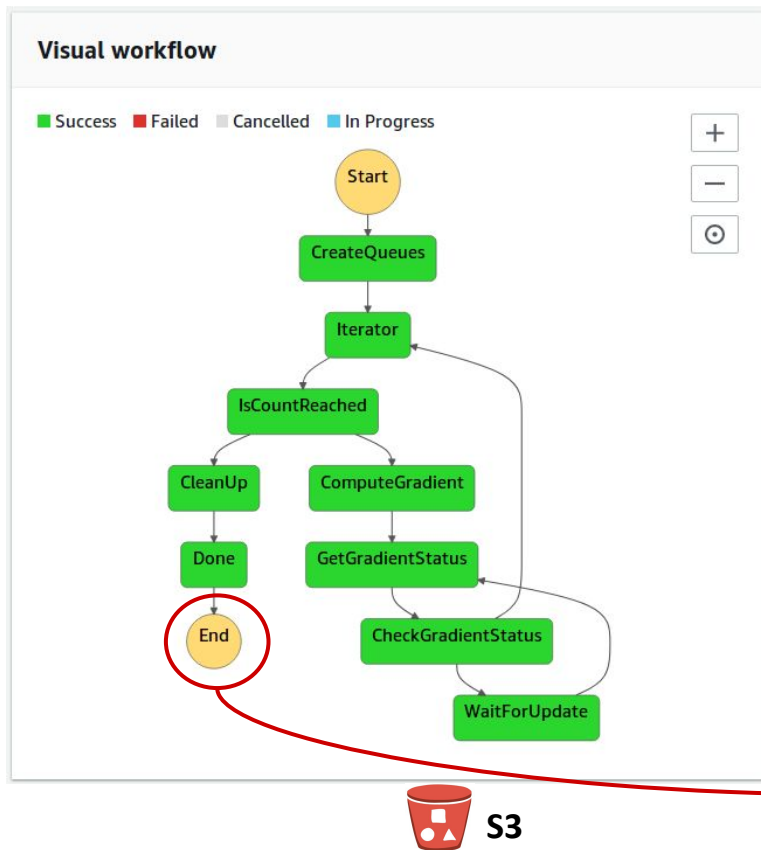
Gradient computations



Serverless workflow:

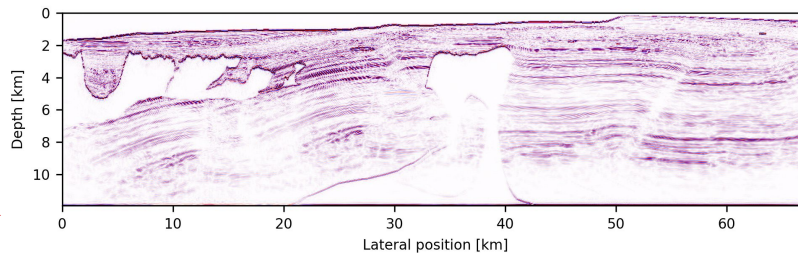
- No additional EC2 instances during gradient computation
- State machine waits for updated image
- Automatic progression to next iteration

Gradient computations



Serverless workflow:

- No additional EC2 instances during gradient computation
- State machine waits for updated image
- Automatic progression to next iteration
- Clean up resources after final iteration



Outline

1. HPC clusters vs. the cloud
2. An event-driven approach to serverless seismic imaging
 - a. Problem formulation
 - b. Workflow components
- 3. Performance analysis:**
 - a. Weak scaling
 - b. Strong scaling
 - c. Cost
 - d. Resilience
4. Case study:
 - a. 3D Seismic imaging on Azure
5. Discussion: HPC in the cloud - feasible and worth it?

Weak scaling

Single workflow iteration:

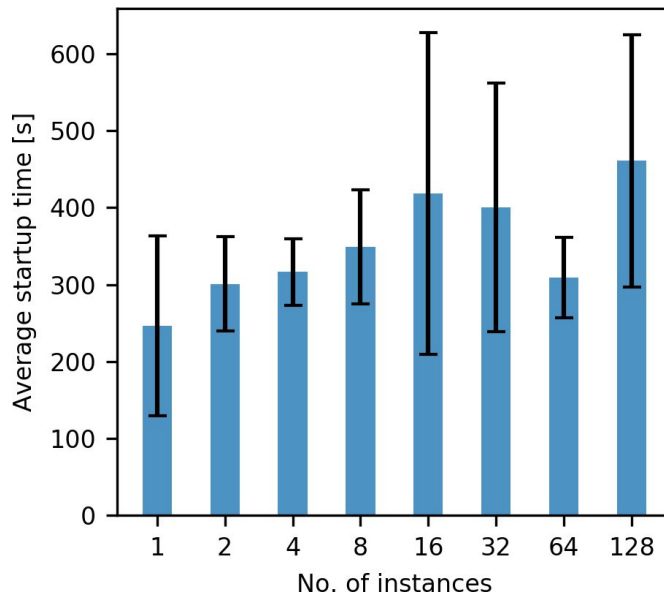
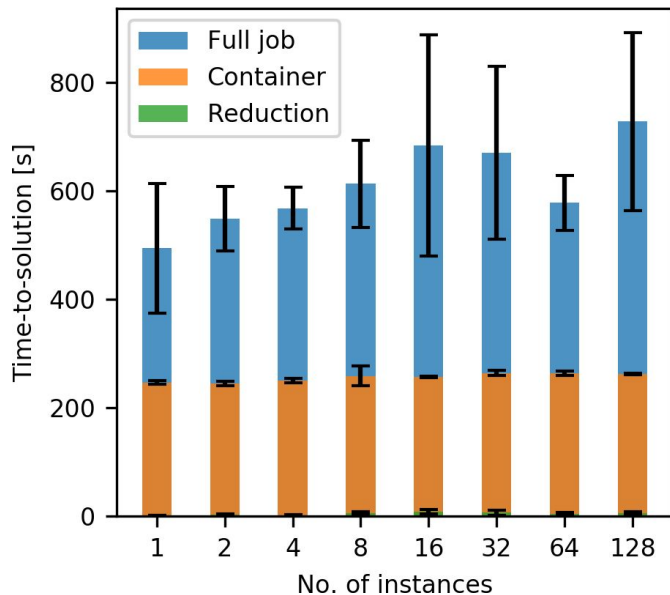
1. Submit AWS Batch job to compute gradient for given batch size
2. Sum gradients using Lambda functions
3. Lambda function performs image update

Time-to-solution:

1. Time it takes AWS Batch to launch EC2 instances + start docker container
2. Run time of the containers
3. Additional gradient summation time
(difference between timestamp of last gradient and updated image)

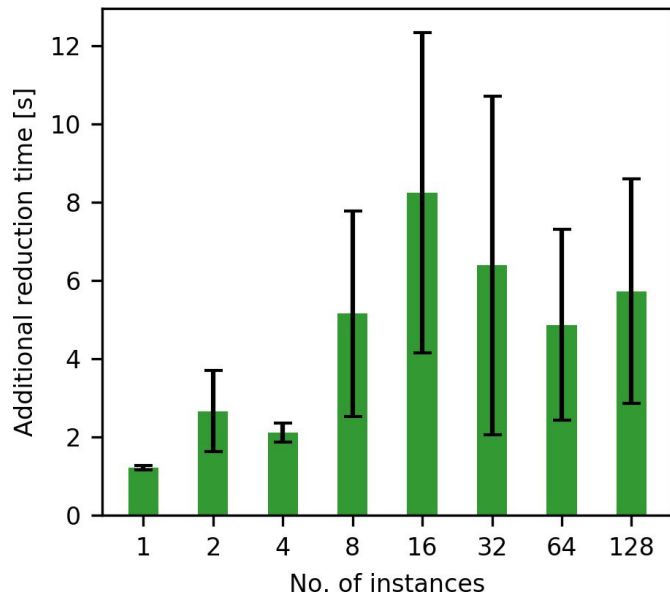
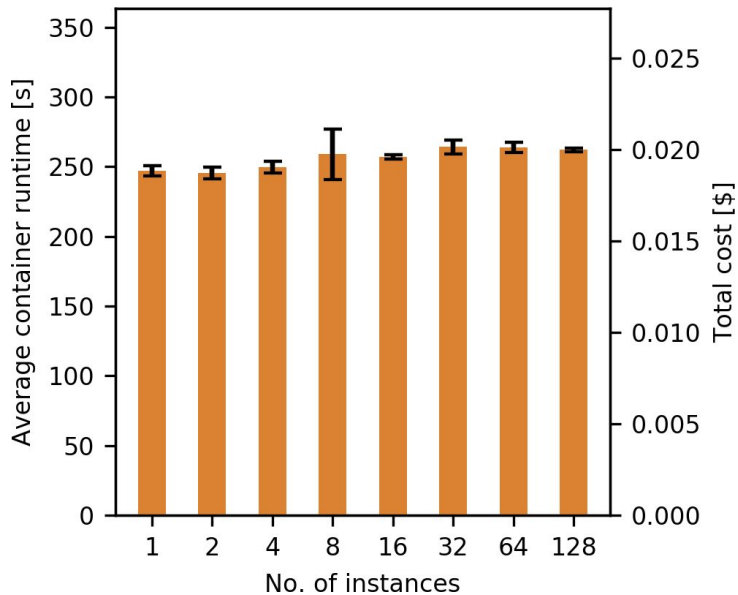
Weak scaling

- Time-to-solution of single SGD iteration
- Workload: 1 element of gradient per instance
- Runtime as function of no. of instances (i.e. for an increasing batch size)



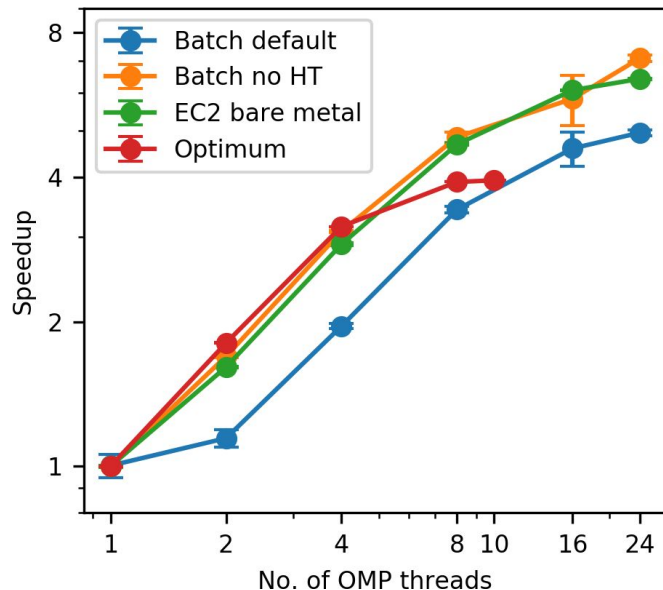
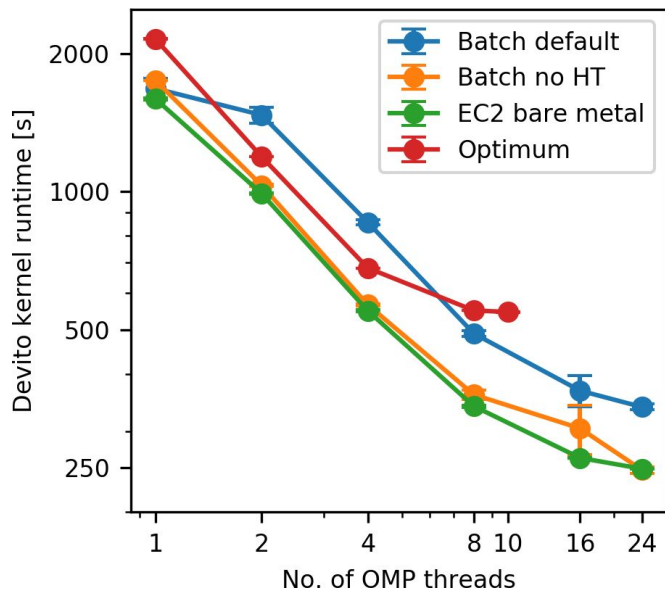
Weak scaling

- Average container runtime expectedly stable (run fully independent)
- Additional gradient reduction time varies, but overall small
- Cost only depends on container runtimes + Lambda runtime for reduction



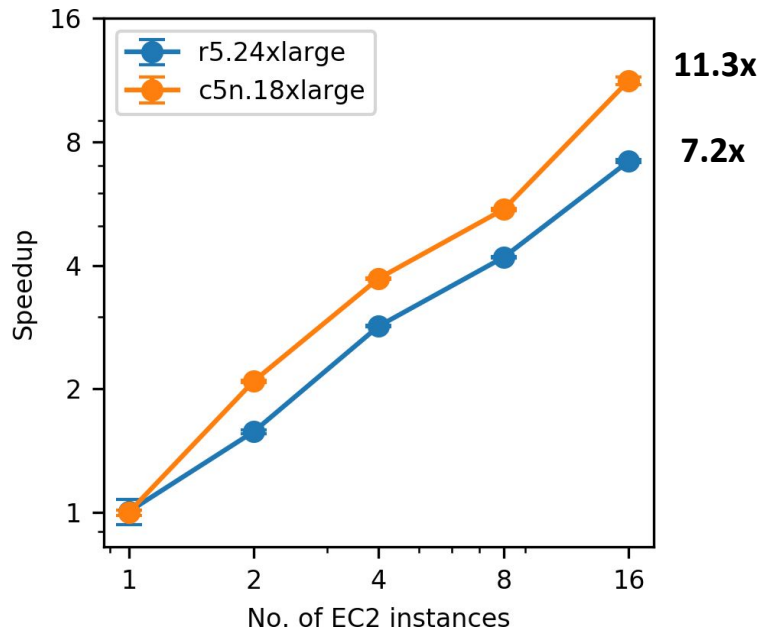
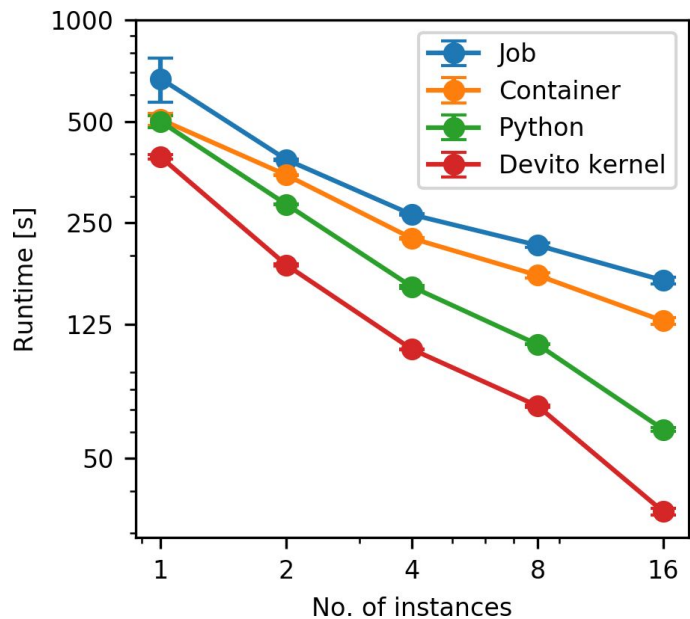
Strong scaling - OpenMP

- Fixed workload: single gradient (of batch size 1)
- Runtime as function of no. of threads
- Performance on bare metal vs. container similar (w/o hyperthreading)



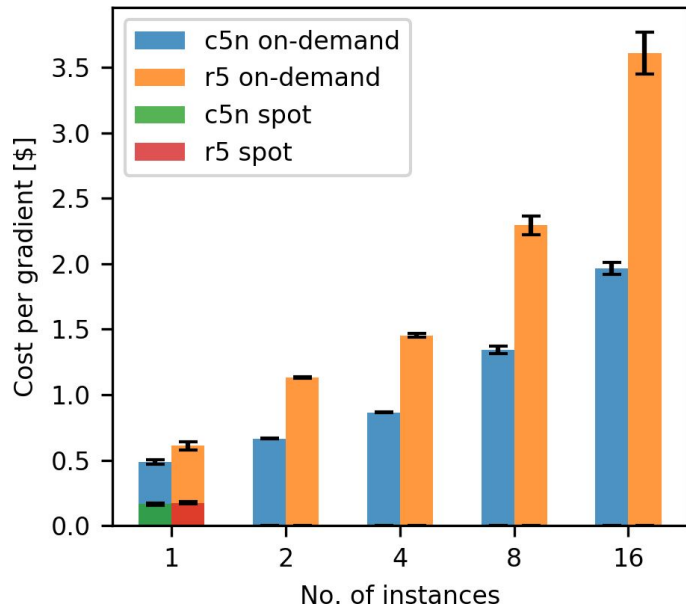
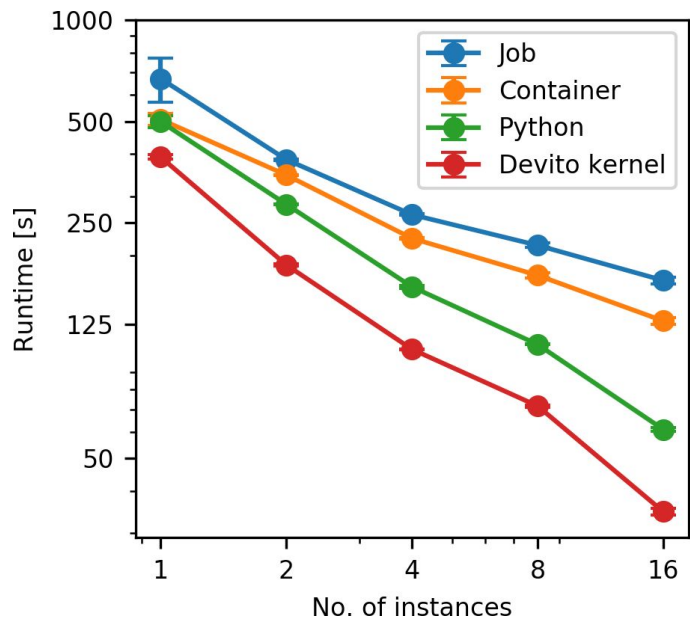
Strong scaling - MPI

- Fixed workload: single gradient (of batch size 1)
- Runtime as function of no. of instances (per gradient)
- Good speed-up **but** significant cost increase (workload is memory bound)



Strong scaling - MPI

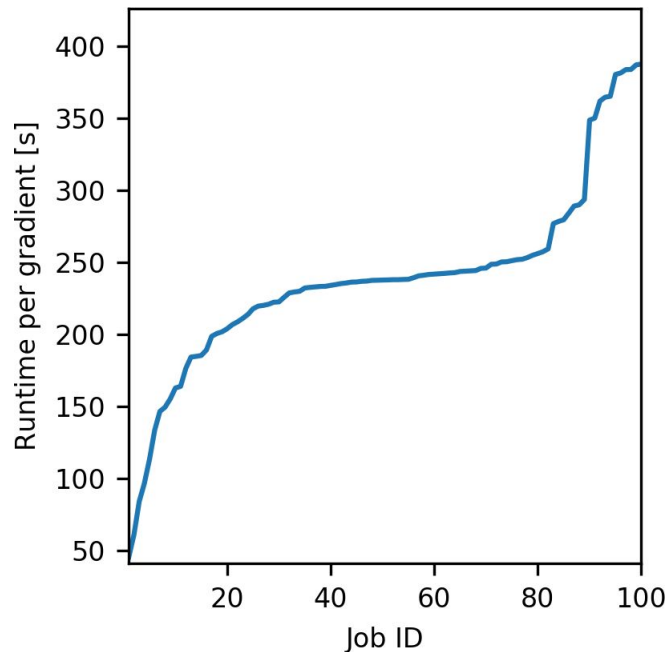
- Fixed workload: 1 gradient
- Runtime as function of no. of instances (per gradient)
- Good speed-up **but** significant cost increase (workload is memory bound)



Cost comparison

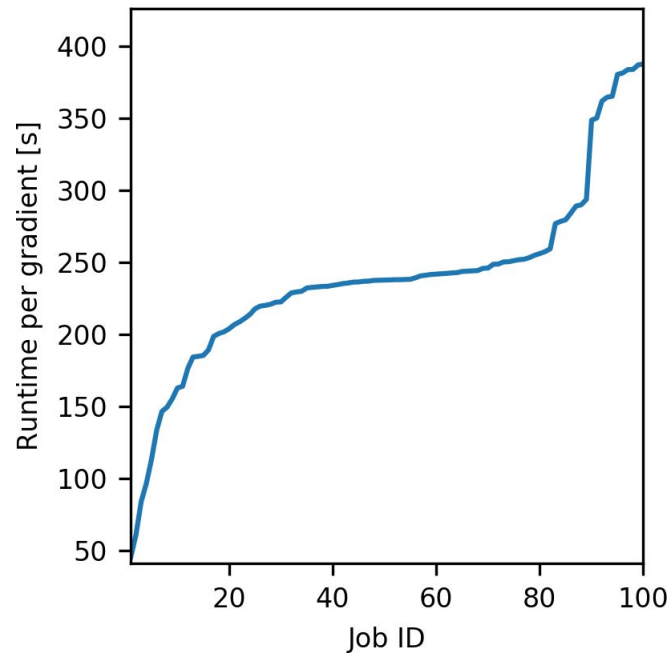
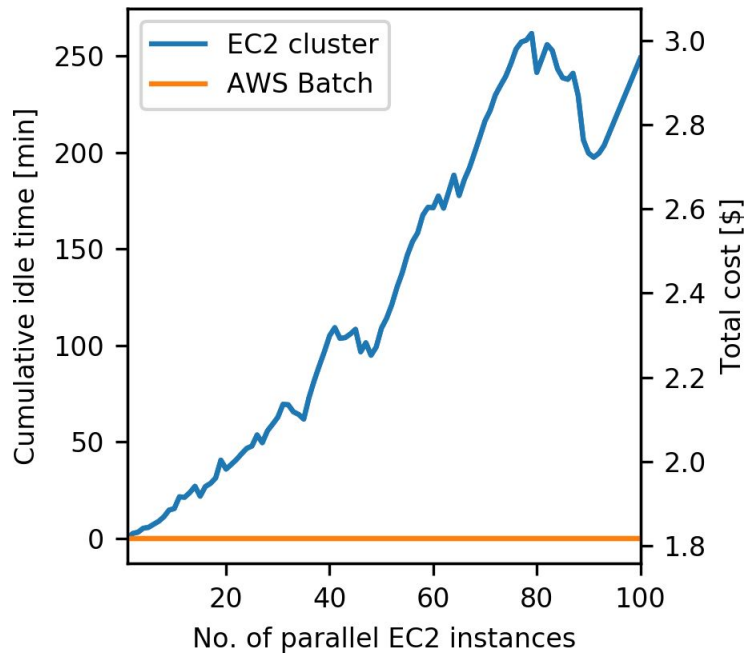
Compute gradient of batch size 100:

- Runtime varies for each gradient (EC2 related, varying max. offset, etc.)
- Fixed cluster: nodes have to wait until last gradient is computed
- Batch: each instance runs only as long computations last
- No cost during wait time for other gradients



Sorted runtimes of 100 gradients

Cost comparison

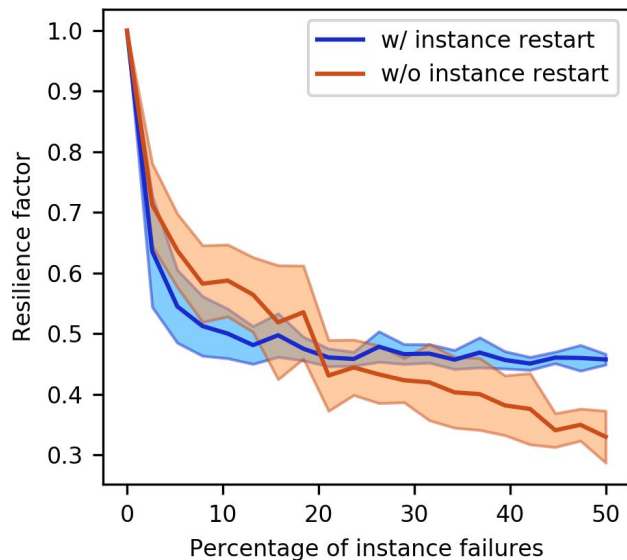


Sorted runtimes of 100 gradients

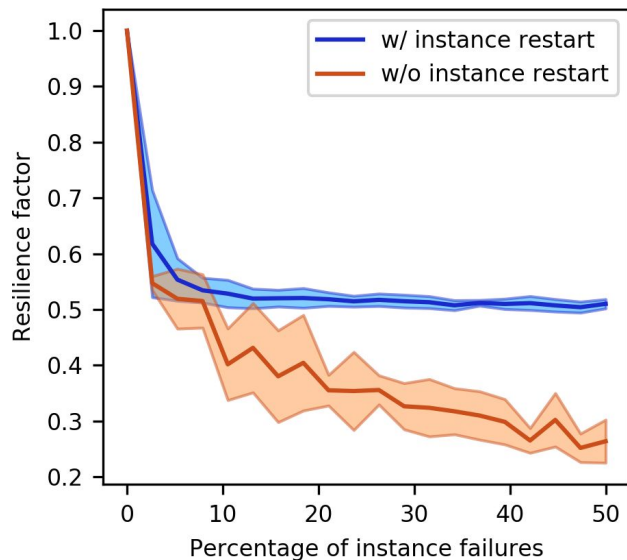
Resilience

- AWS Batch: instances restart automatically
- Resilience factor = original runtime / runtime with errors
- Compute gradient of batch size 100 and randomly kill instances
- Compare RF w/ and w/o instance restarts

Runtime per gradient: ~5 minutes



Runtime per gradient: ~45 minutes



Outline

1. HPC clusters vs. the cloud
2. An event-driven approach to serverless seismic imaging
 - a. Problem formulation
 - b. Workflow components
3. Performance analysis:
 - a. Weak scaling
 - b. Strong scaling
 - c. Cost
 - d. Resilience
- 4. Case study:**
 - a. 3D Seismic imaging on Azure
5. Discussion: HPC in the cloud - feasible and worth it?

Multi platform approach

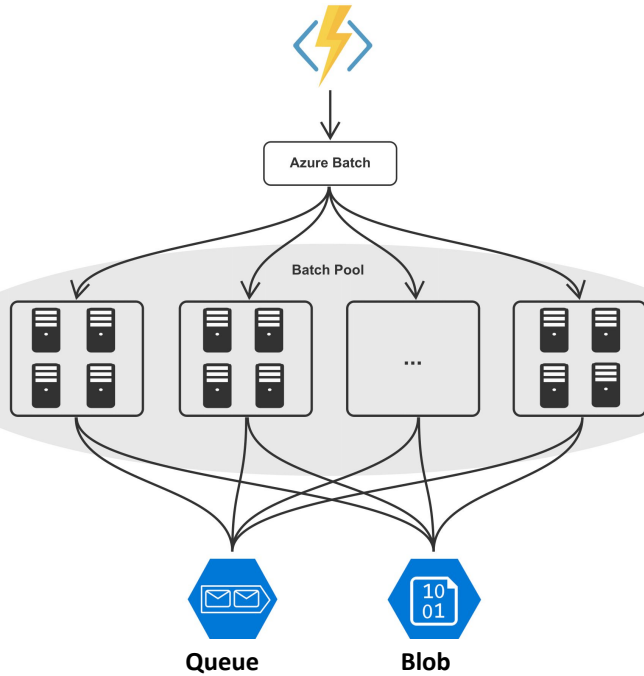
	Azure	AWS	GCP
Compute instances	Virtual machines	EC2	Compute engine
Object storage	Blob	S3	Cloud storage
Batch computing	Azure Batch	AWS Batch	Pipelines
Serverless functions	Azure functions	Lambda functions	Cloud functions
Message queues	Queue storage	SQS	Cloud Pub/Sub
Distributed file system	Azure files	EFS	Cloud filestore

<https://docs.microsoft.com/en-us/azure/architecture/aws-professional/services>

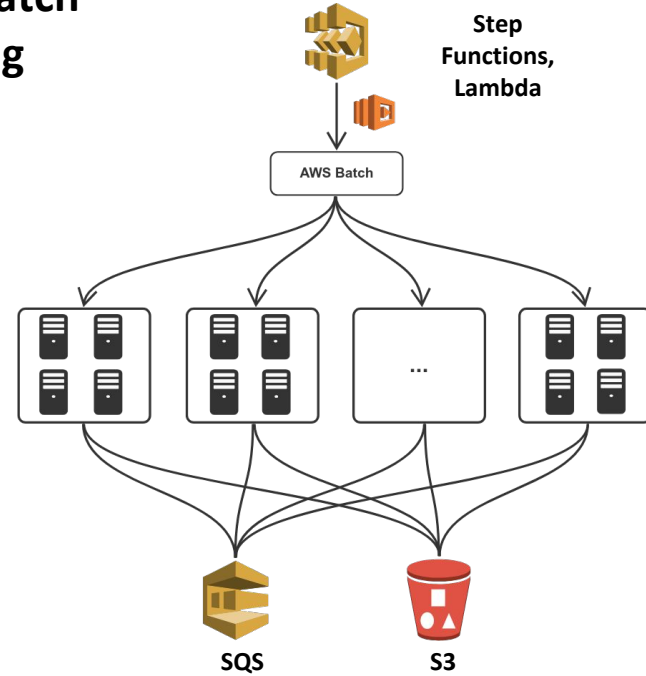
<https://cloud.google.com/docs/compare/aws/>

Multi platform approach

Serverless batch computing



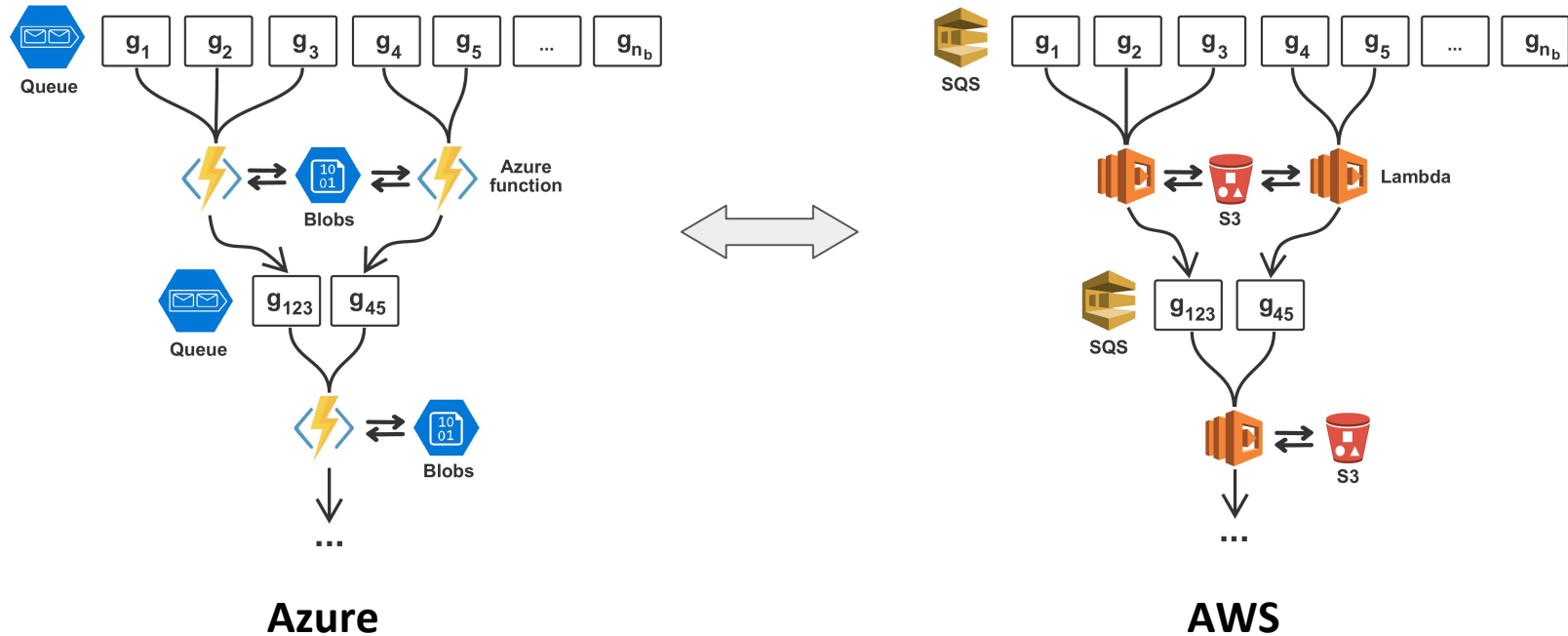
Azure



AWS

Multi platform approach

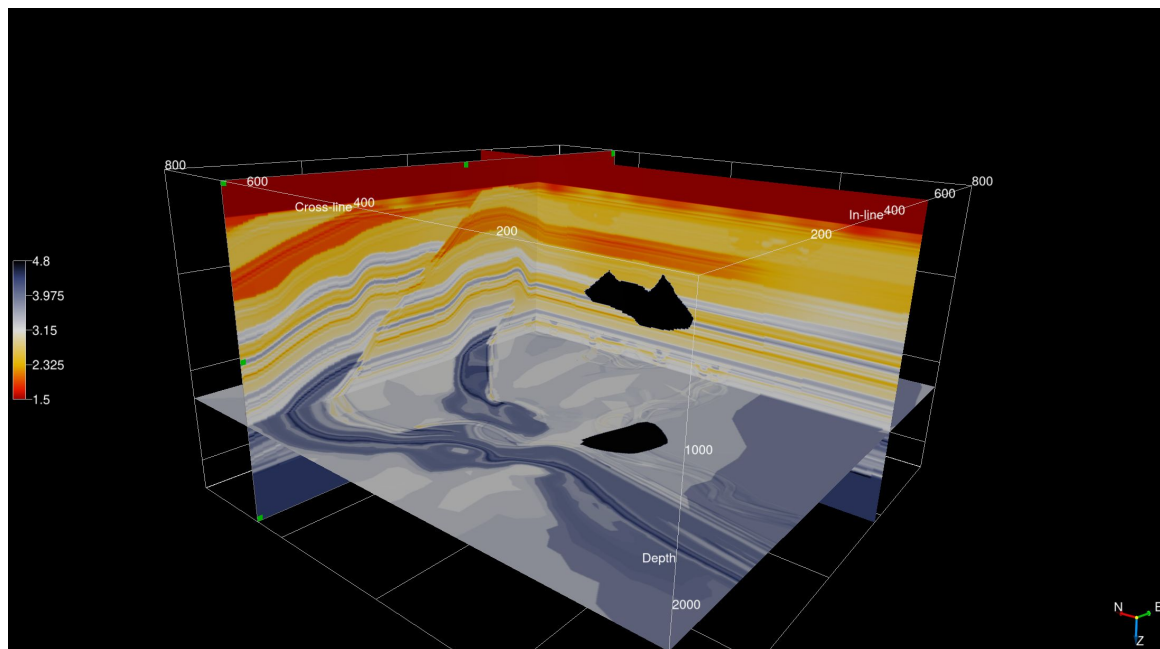
Event-driven gradient summation



3D imaging case study

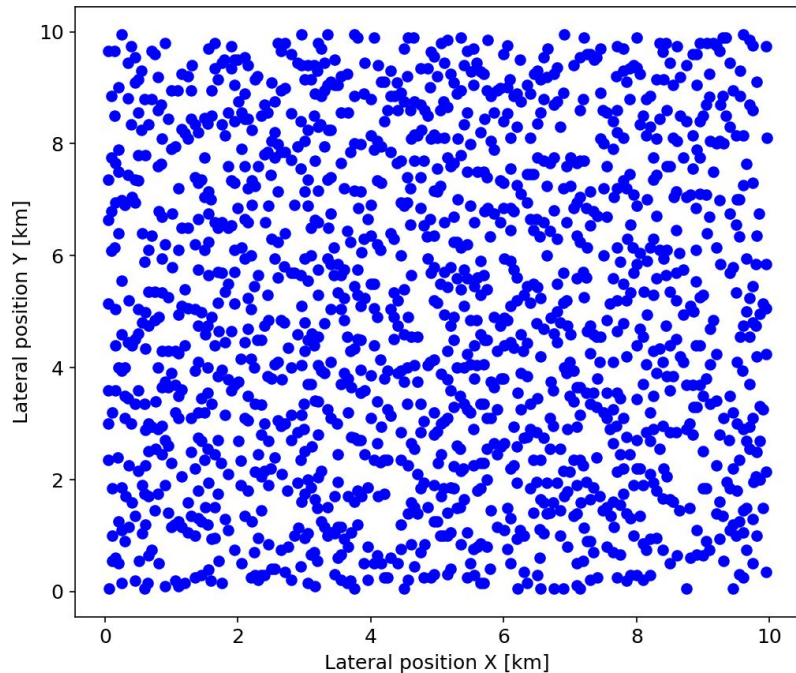
Synthetic 3D seismic velocity model:

- Domain: 10 x 10 x 3.325 km
- Grid: 881 x 881 x 347 (12.5 m grid + ABCs) \Rightarrow 2.7 million unknowns

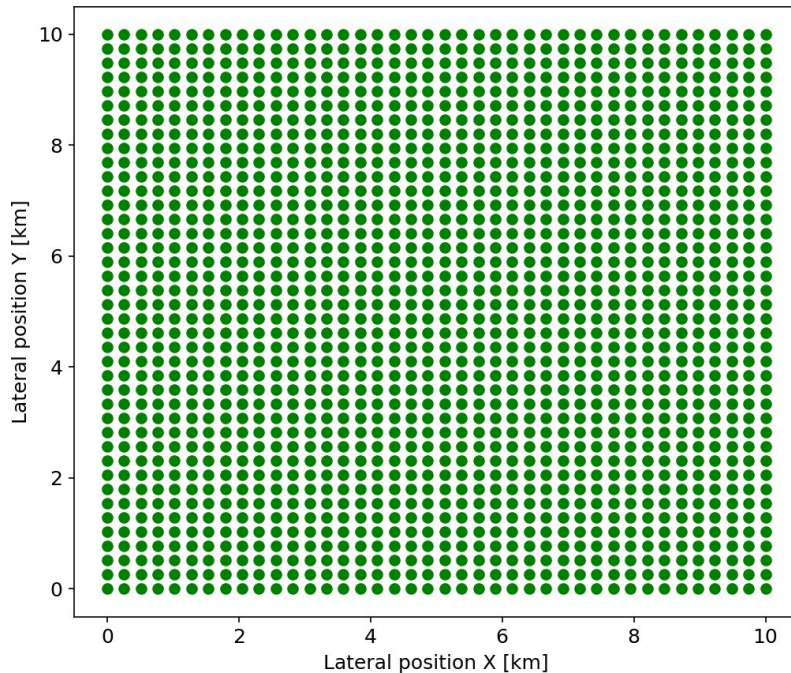


3D imaging case study

Acquisition geometry:



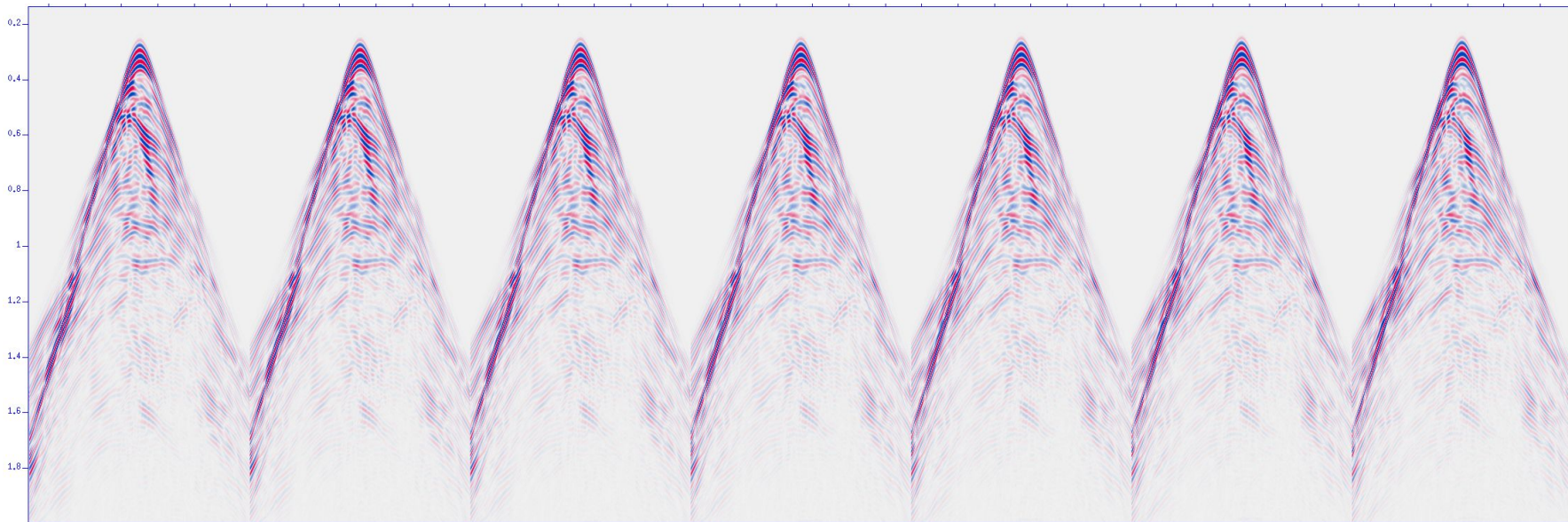
Source grid



Receiver grid

3D imaging case study

- Observed data: 1,500 shot records
- Modeled w/ anisotropic acoustic wave equation ([Zhang et al., 2011](#))



Shot records in xline

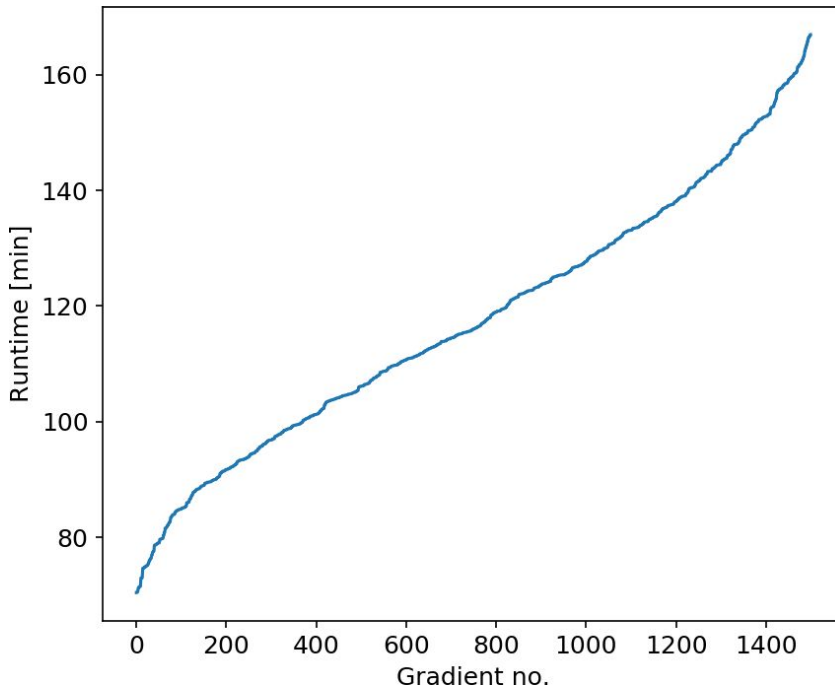
3D imaging case study

Experiment:

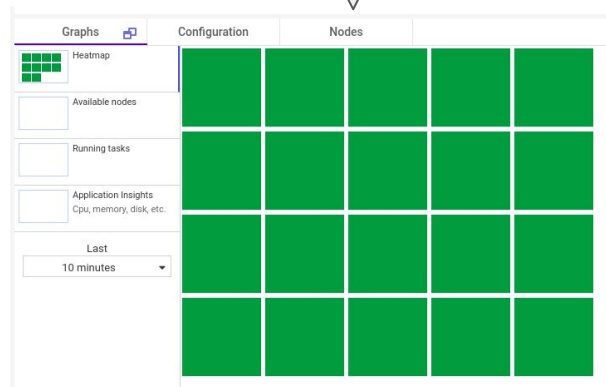
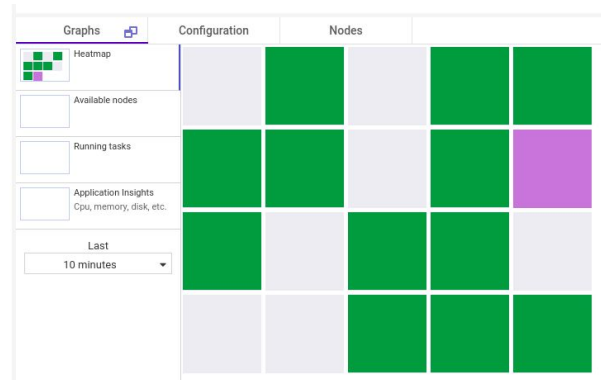
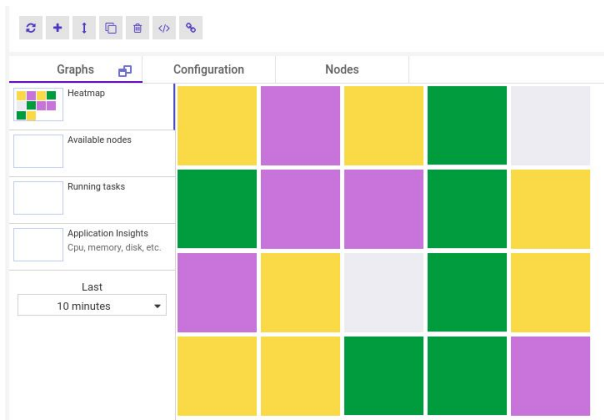
- Single gradient w/ batchsize 1,500
- 100 nodes (E64, ES64)
- 432 GB RAM, 64 vCPUs per node
- 2 nodes per gradient

Timings + cost:

- Average runtime: 110 minutes per gradient
- Average cost per gradient: 11\$ (dedicated)
- Peak performance: 140 GFLOPS per VM (14 TFLOPS total)
- Total cost: 17,000\$



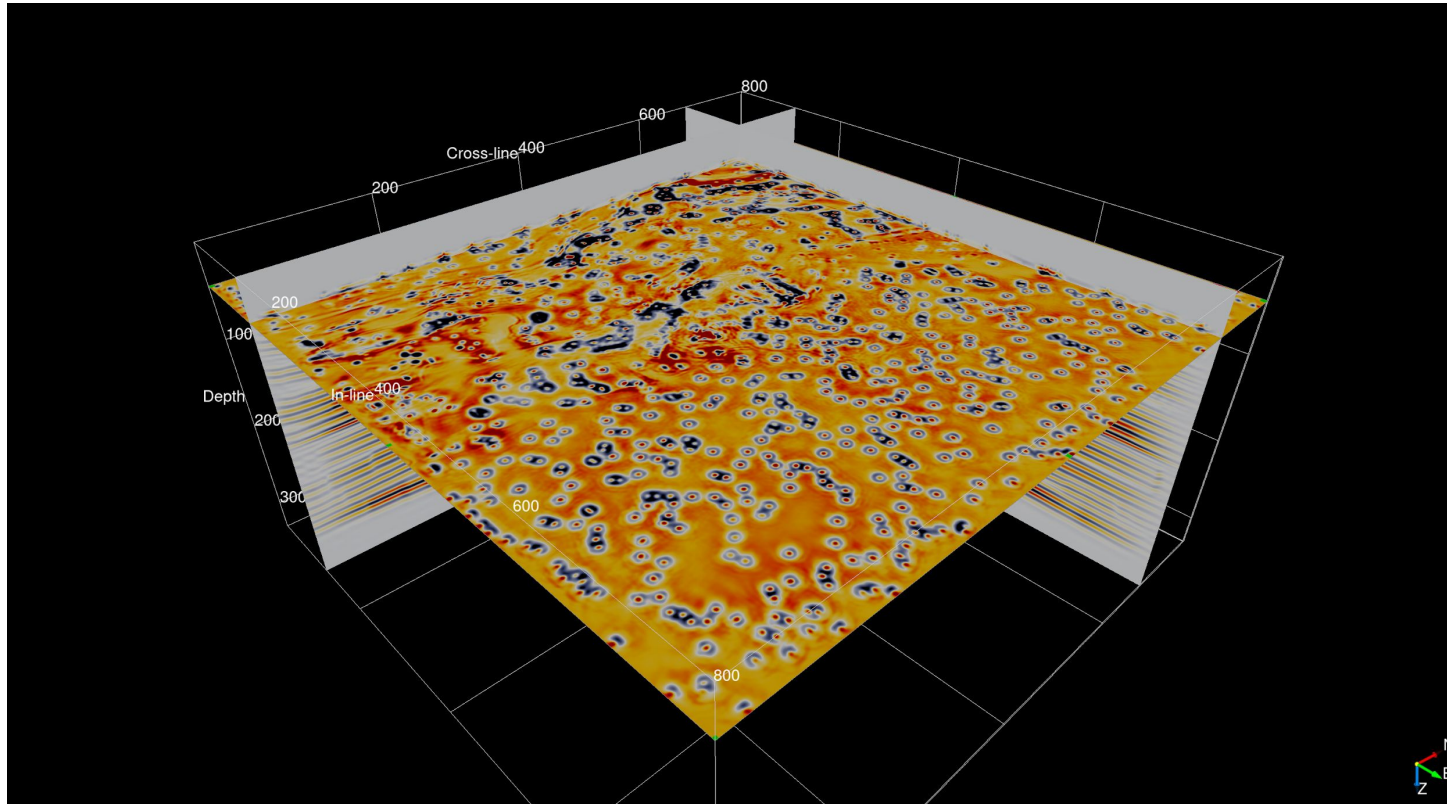
3D imaging case study



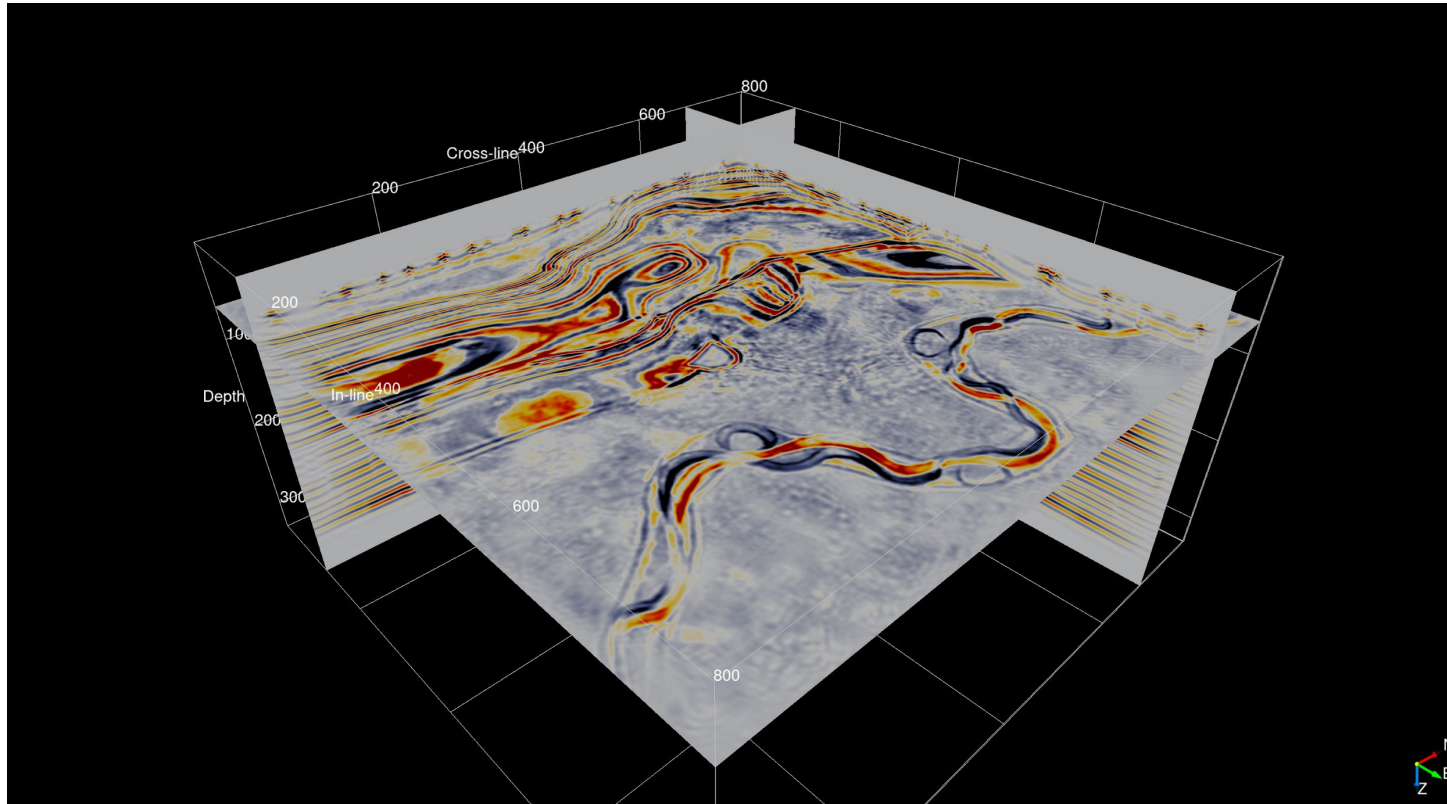
Azure Batch:

- Jobs start as VMs are added to pool
- Do not need to wait for full pool
- No long idle times

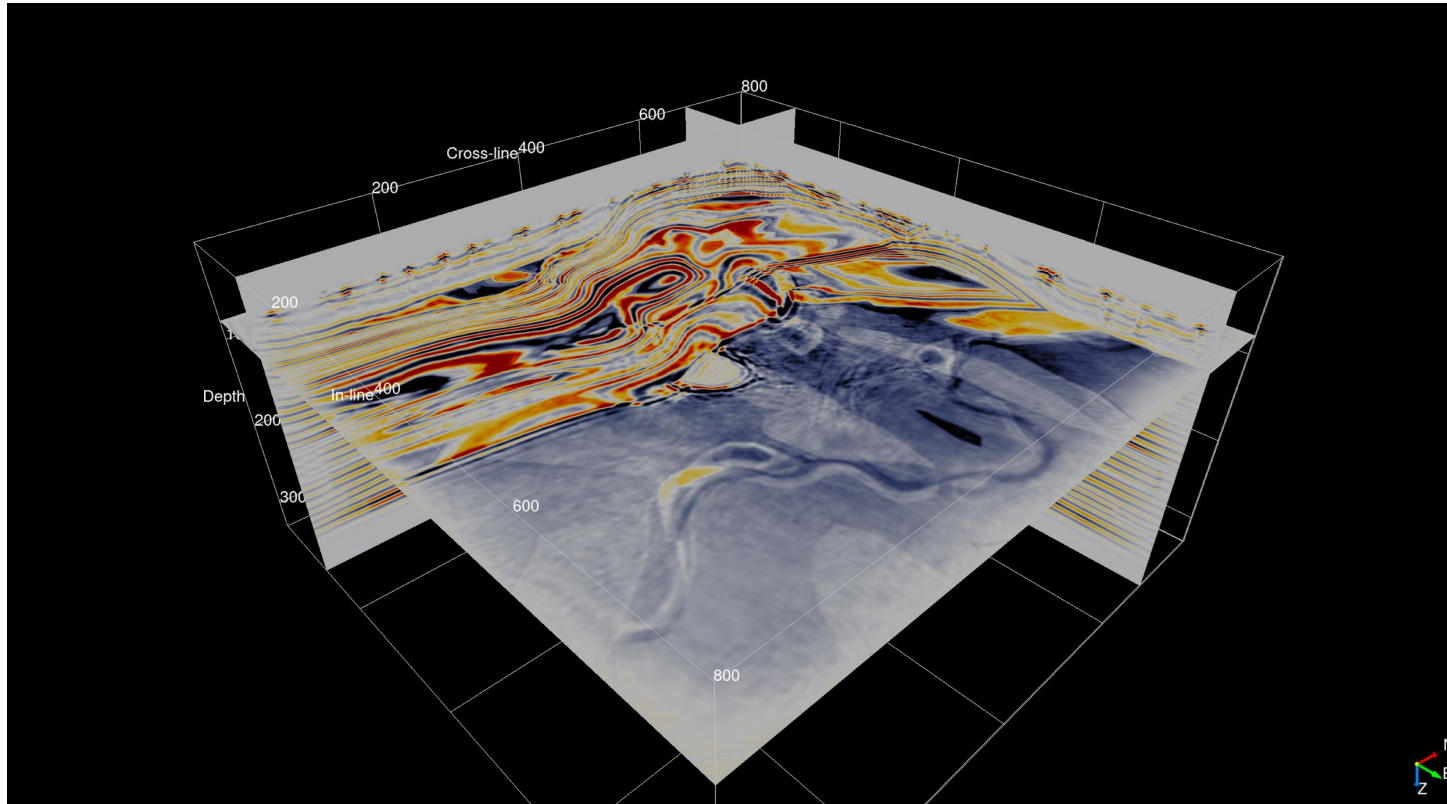
3D imaging case study



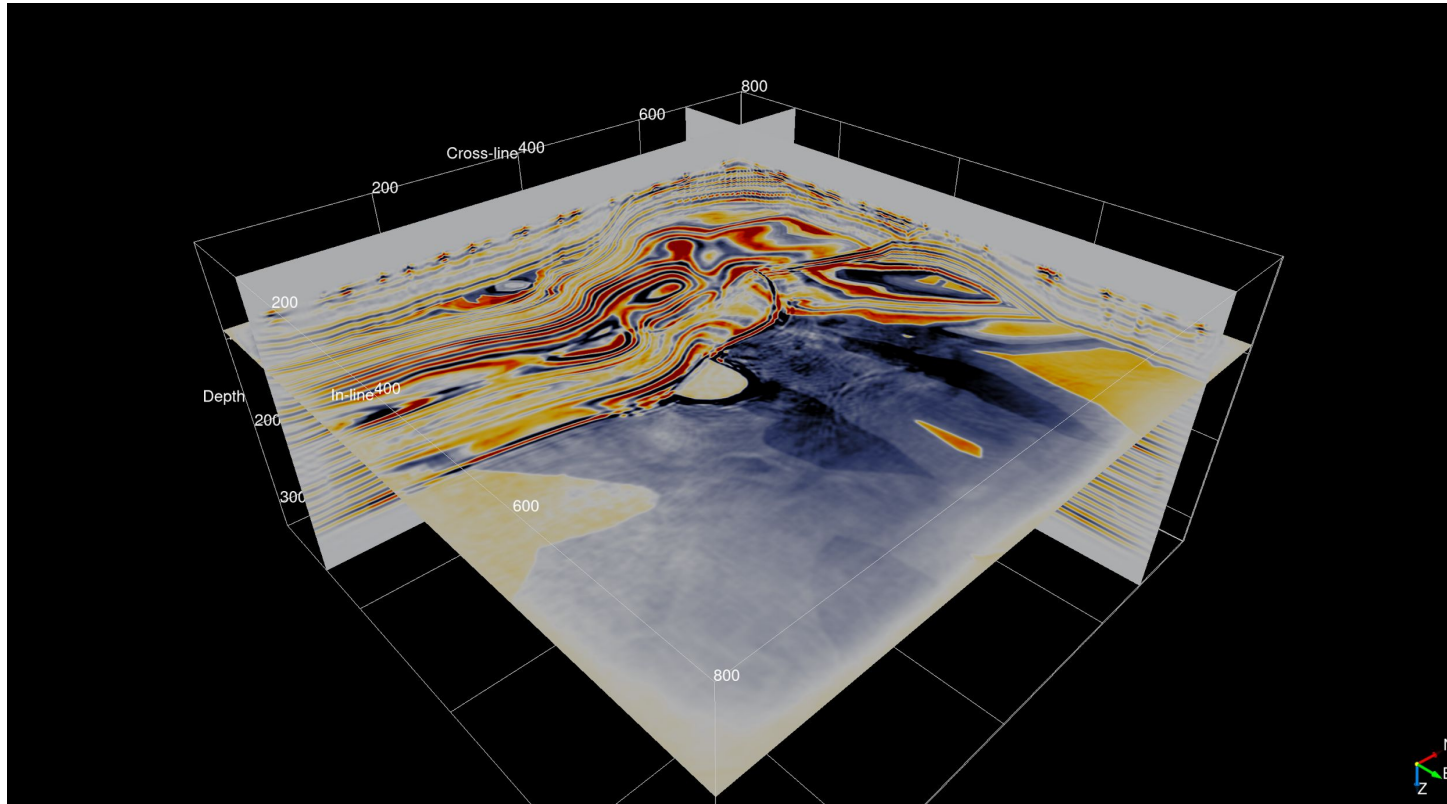
3D imaging case study



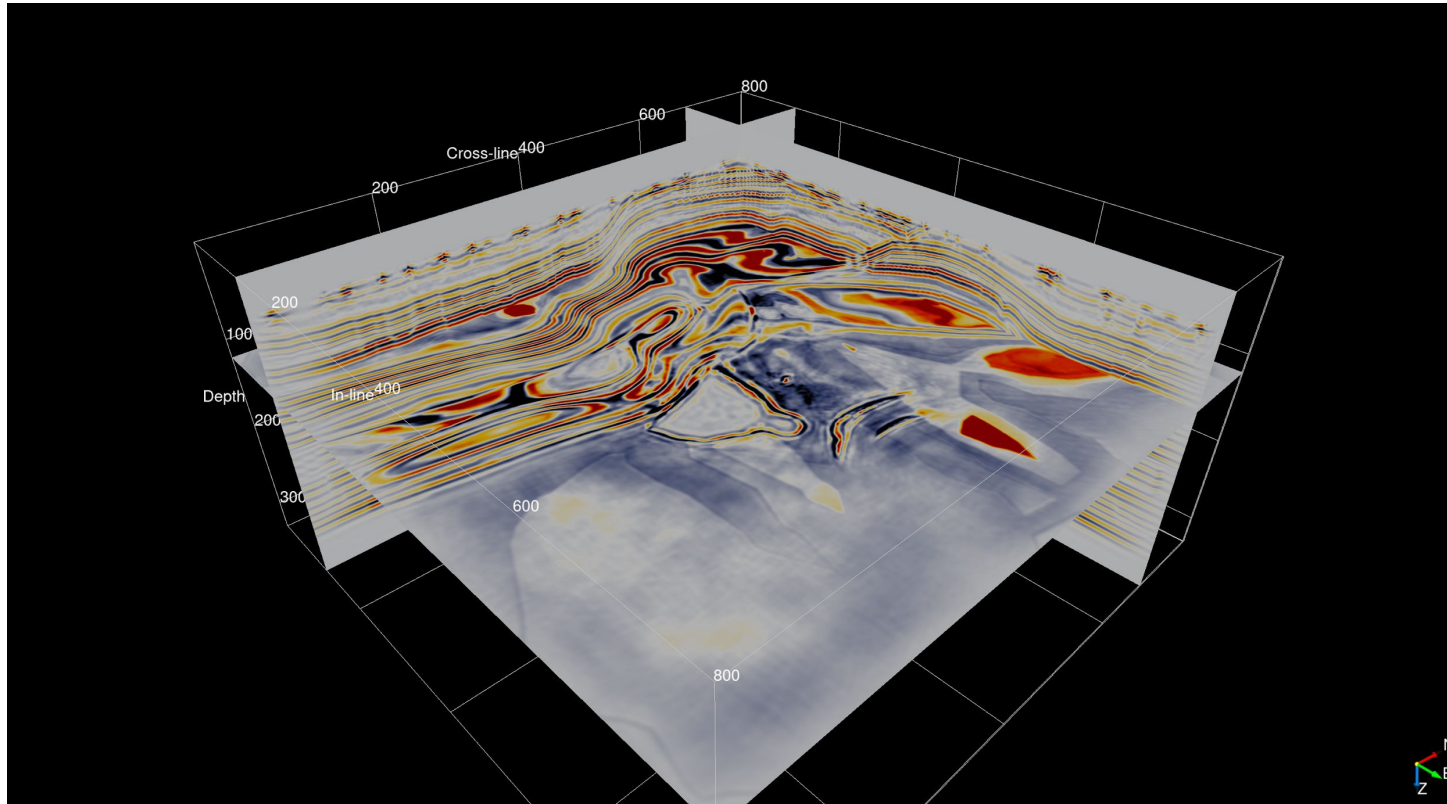
3D imaging case study



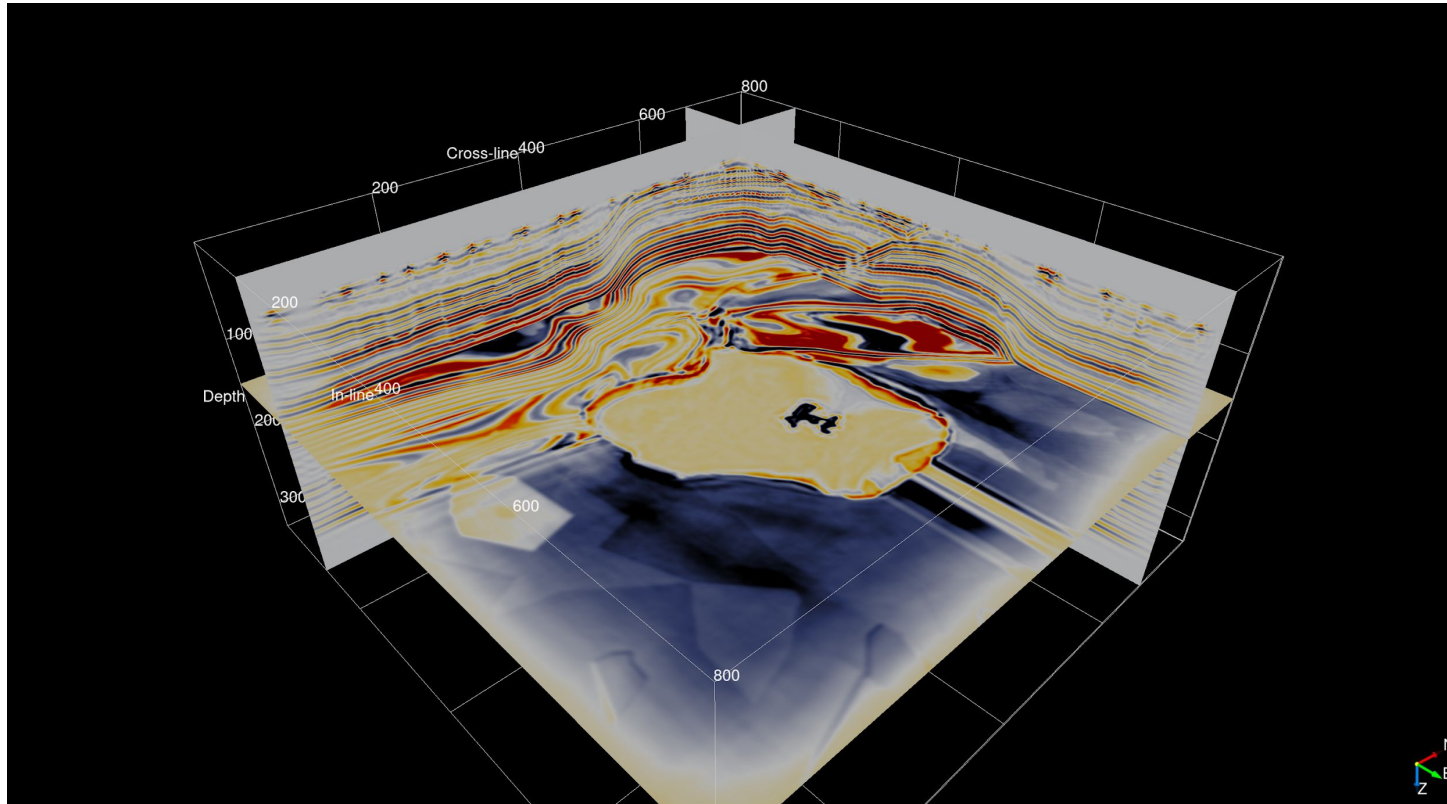
3D imaging case study



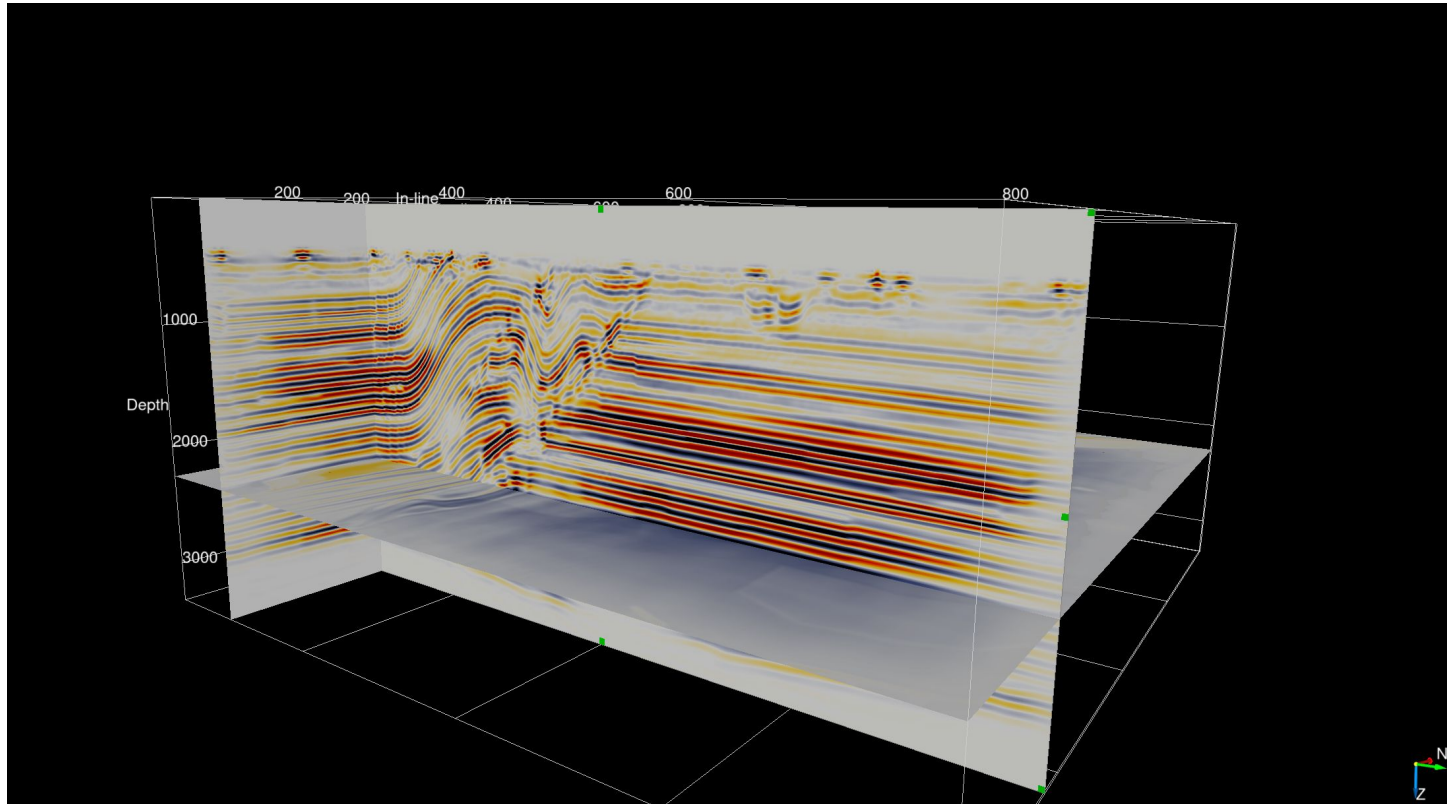
3D imaging case study



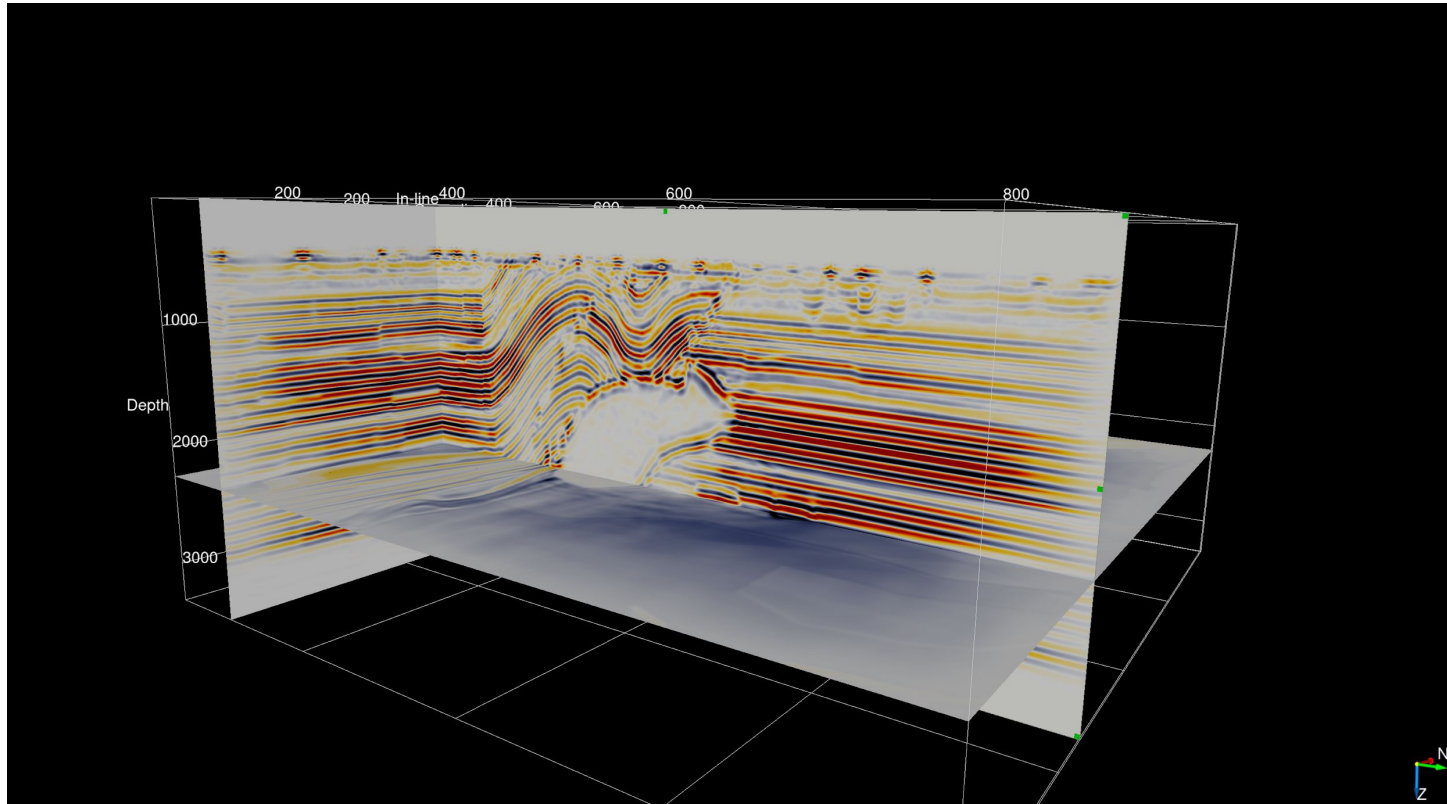
3D imaging case study



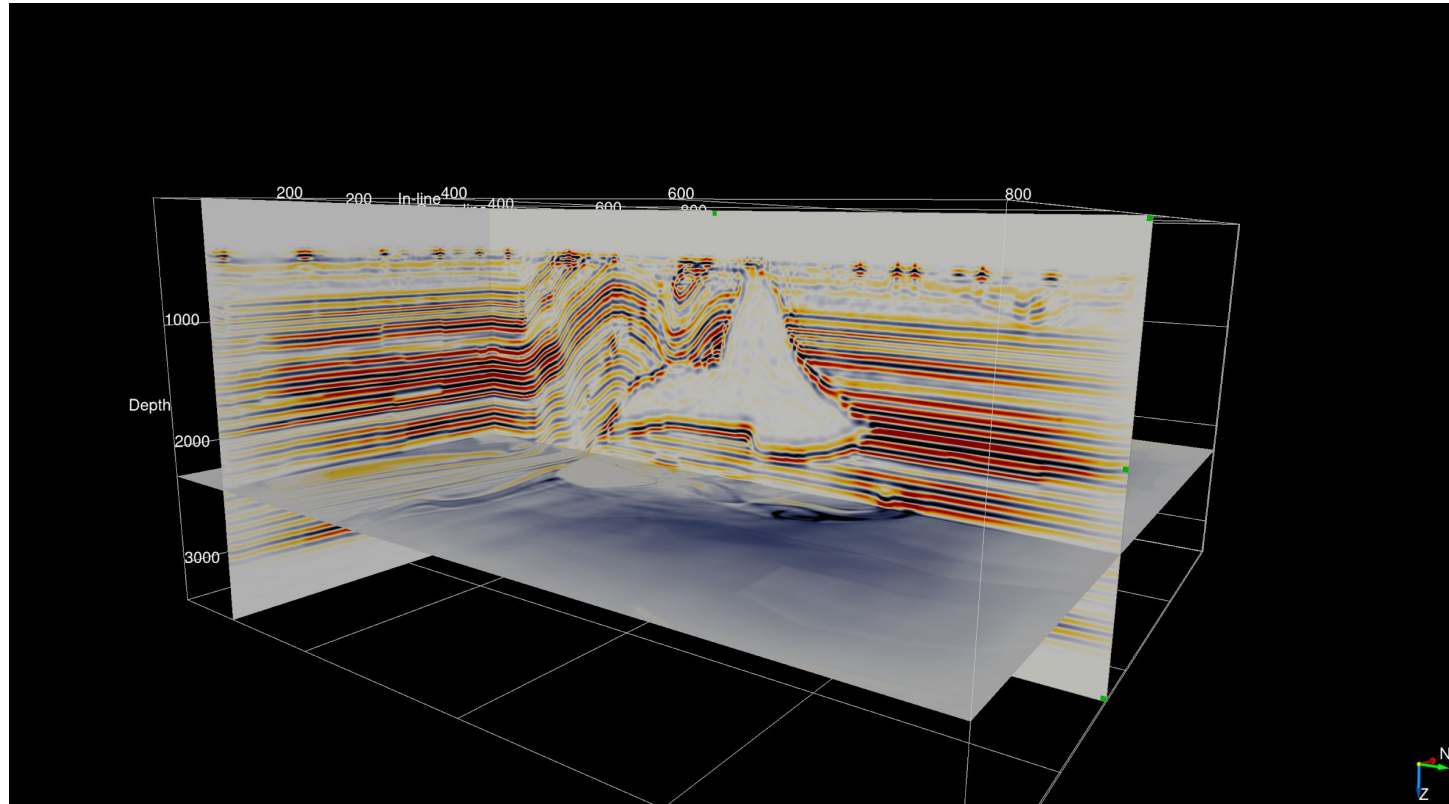
3D imaging case study



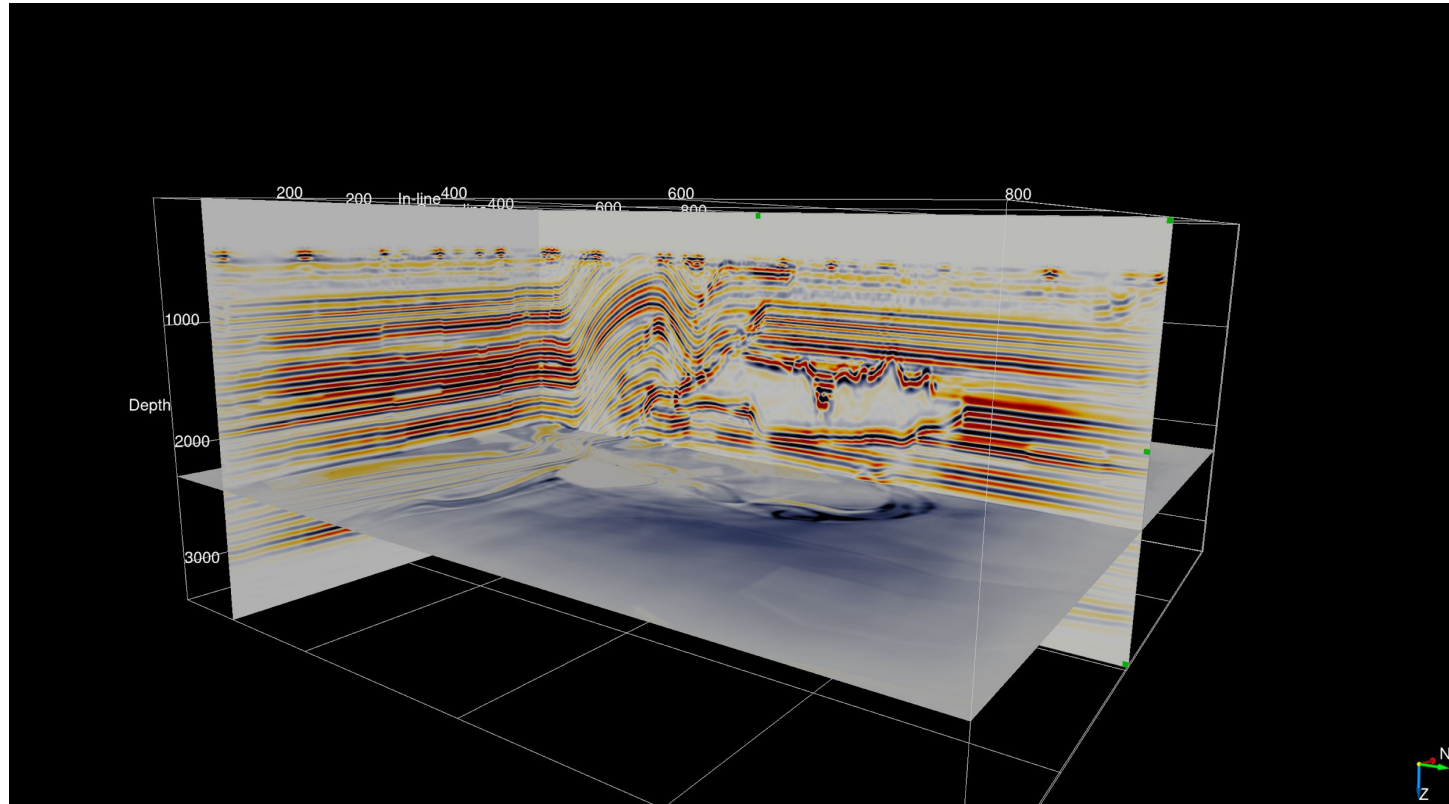
3D imaging case study



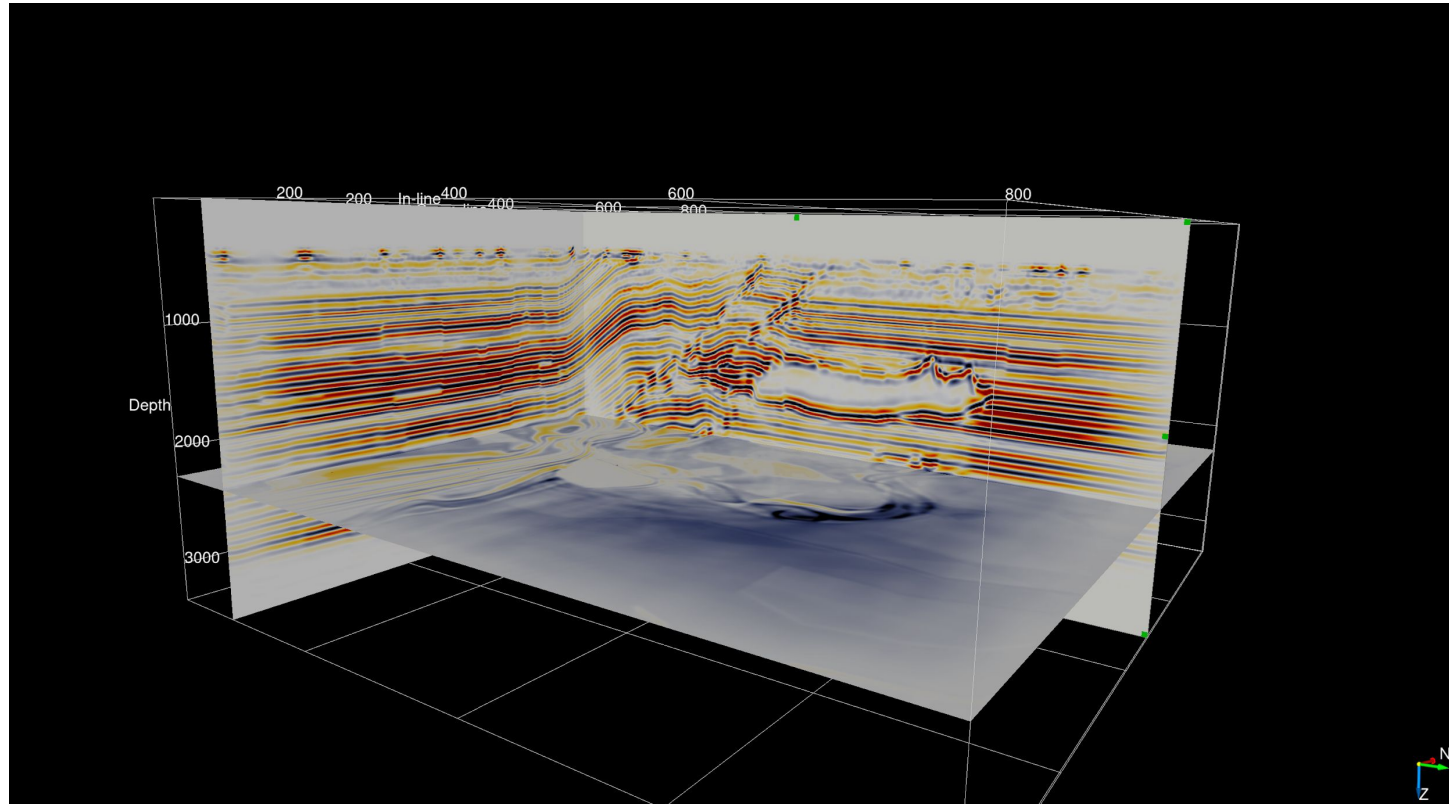
3D imaging case study



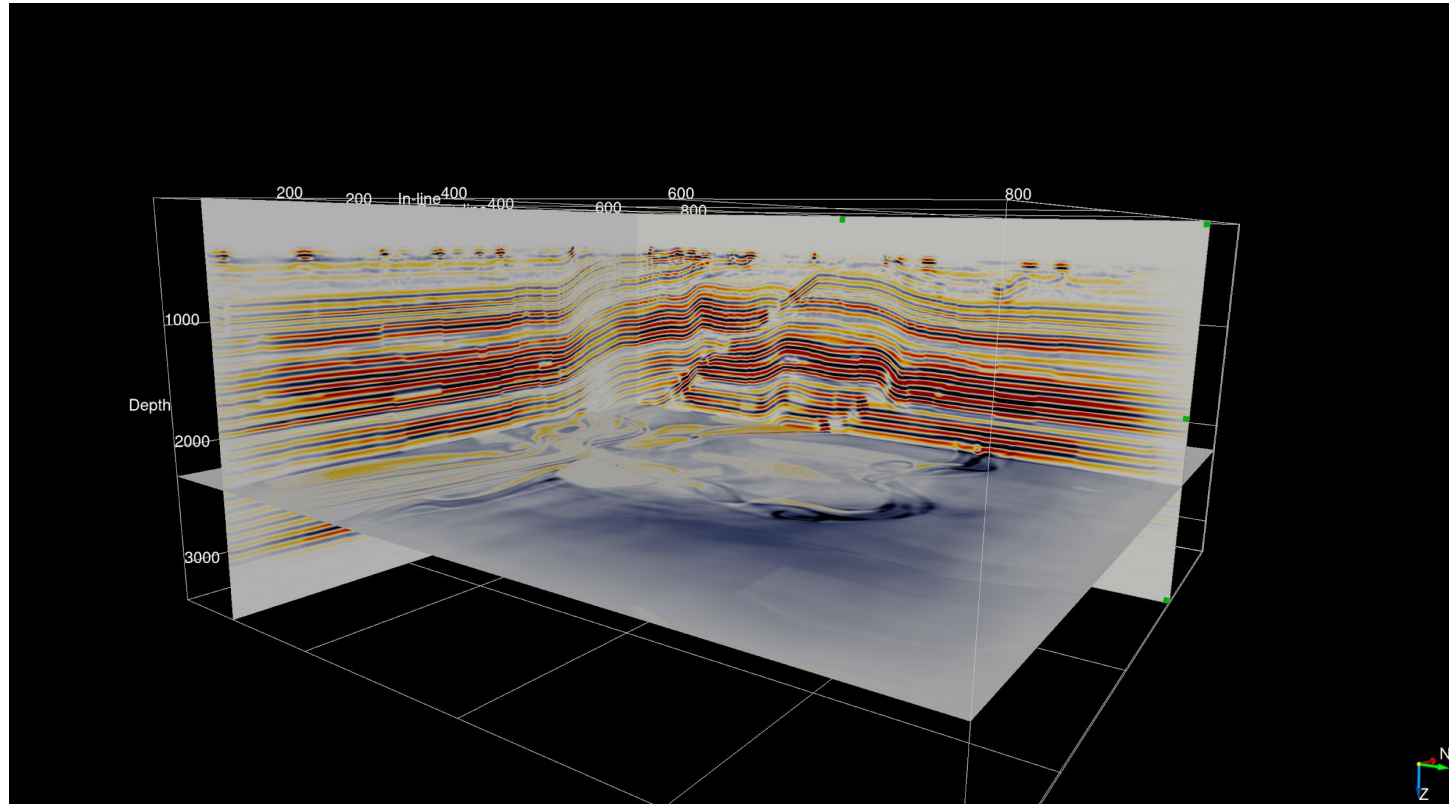
3D imaging case study



3D imaging case study



3D imaging case study



Outline

1. HPC clusters vs. the cloud
2. An event-driven approach to serverless seismic imaging
 - a. Problem formulation
 - b. Workflow components
3. Performance analysis:
 - a. Weak scaling
 - b. Strong scaling
 - c. Cost
 - d. Resilience
4. Case study:
 - a. 3D Seismic imaging on Azure
5. **Discussion: HPC in the cloud - feasible and worth it?**

Discussion

Is cloud computing competitive for HPC? For research?

PACE cluster at Georgia Tech:

- 192 GB compute node w/ 24 cores: \$7,200 (5 year warranty)
- Intel Xeon Gold 6226 (Cascade Lake) @ 2.7Ghz
- **\$0.164** per hour

Equivalent node on AWS:

- m5.12xlarge instance w/ 192 GB RAM, 24 cores
- Intel Xeon Platinum 8000 (Skylake-SP) @ 3.1Ghz
- On-demand: **\$2.304** per hour -> **14x**
- Spot-price: **\$0.5694** per hour -> **3.5x**

Discussion

On-premise cluster favorable:

- Need permanent access and run at full capacity
- Run fixed type of workload (always need same memory, CPUs)
- Software that runs in production environment

Cloud favorable:

- Need irregular access
- Variable hardware requirements (varying memory, CPU/GPU)
- Size/number of workloads varies over time

Take-home message

Moving to the cloud:

- Possible to use for HPC workloads
- Important to redesign software
- Avoid *lift and shift*
- Understand how your application can take advantage of cloud services/technologies (event-driven, batch computing, etc.)
- Avoid idle resources + use spot instances

Acknowledgments

Many thanks to:

- Microsoft Azure
- Organizers of HotCSE Seminar



THE UNIVERSITY
OF BRITISH COLUMBIA

osokey

Arxiv preprint: <https://arxiv.org/abs/1909.01279>

