Event-driven workflows for large-scale seismic imaging in the cloud
Philipp A. Witte↑, Mathias Louboutin↑, Henryk Modzelewski*, Charles Jones↑, James Selvage↑ and Felix J. Herrmann↑
Disclaimer

- The following work was developed using Amazon Web Services (AWS) and therefore contains terminology referencing AWS services and product names.

- Technology presented in this talk is not tied to one specific cloud provider and has been replicated on other platforms.
Seismic wave equation-based inversion

- Solve wave equations for many sources
- Propagate over many wavelengths
- Massive data I/O
- Curse of dimensionality
- Infeasible for very large models (e.g. SEAM)
- Requires HPC environments

FWI

LS-RTM (< 120$)

SEAM (?$)
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/CPUs/GPUs/RAM)
Seismic inversion in the cloud

Cloud computing

✅ 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

Lift and shift

Legacy Fortran or C code
Moving to the cloud

- **Lift and shift**
  - Legacy Fortran or C code
  - 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)
Moving to the cloud

Go serverless (and re-engineer)

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

Typical components of LS-RTM*:

\[
\text{minimize } \sum_{i=1}^{n_s} \frac{1}{2} \left\| \mathbf{J}(\mathbf{m}, \mathbf{q}_i) \delta \mathbf{m} - d_{i}^{\text{obs}} \right\|_2^2
\]

- 1. Compute gradient for all/subset of source locations: 
  \[ \mathbf{g}_i = \mathbf{J}^\top \left( \mathbf{J} \delta \mathbf{m} - d_{i}^{\text{obs}} \right) \]
- 2. Sum gradients: 
  \[ \mathbf{g} = \sum_{i=1}^{n_b} \mathbf{g}_i \]
- 3. Update image based on optimization algorithm (SGD, CG, etc.): 
  \[ \delta \mathbf{m} = \delta \mathbf{m} - \alpha \mathbf{g} \]

* e.g. Valenciano, 2008; Dong et al., 2012; Zeng et al. 2014
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

```
*Comment*: "Iterator State Machine Example",
"Start": "CreateQueues",
"States": {
  "CreateQueues": {
    "Comment": "Create SQS queues and lambda trigers for the gradient reduction",
    "Type": "Task",
    "ResultPath": "$p",
    "Next": "Iterator"
  },
  "Iterator": {
    "Type": "Task",
    "ResultPath": "$p",
    "Next": "IsCountReached"
  },
  "IsCountReached": {
    "Type": "Choice",
    "Choices": {
      "Variable": "$s.iterator.continue",
      "BooleanEquals": true,
      "Next": "ComputeGradient"
    }
  }
}
```

Diagram of the state machine.
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 shot records (AWS Batch)
- Domain decomposition (MPI)
- Multithreading (OpenMP)
- Each gradient computed on individual instance or cluster of instances (cluster of clusters)
Gradient computations

Software to compute gradients:

- Batch runs docker containers
- Solve wave equations using Devito*
- Automated performance optimizations (loop blocking, vectorization, refactoring, OMP, MPI, etc.)

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

Sparsity-promoting LS-RTM of the BP Synthetic 2004 model:

- 1348 shot records
- Velocity model: 67.4 x 11.9 km (10,789 x 1,911 grid points)
- 20 iterations of linearized Bregman method
- Batchsize of 200 shot records per iteration
- Curvelet-based sparsity promotion

BP Synthetic 2004
Numerical examples

Sparsity-promoting LS-RTM on the BP Synthetic 2004 model

Image after ~3 data passes (total cost of < 120 $)
Numerical examples

Reverse-time migration of the BP TTI model:

- 1641 shot records
- Velocity model: 78.7 x 11.3 km (12,596 x 1,801 grid points)
- Anisotropic modeling using pseudo-acoustic TTI equations*
- True adjoints of linearized Born scattering operator
- Domain-decomposition to compute gradients
- Each gradient computed on MPI cluster of 6 instances (no spot instances)

*Zhang et al., 2011
Numerical examples

Reverse-time migration of the BP TTI model

RTM image (total cost of approx. 420 $)
Numerical examples

RTM image (total cost of approx. 420 $)
## Numerical examples

<table>
<thead>
<tr>
<th></th>
<th>BP TTI 2007</th>
<th>BP 2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of shots</td>
<td>1641</td>
<td>1348</td>
</tr>
<tr>
<td>Instances/gradient</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Instance type</td>
<td>m5.xlarge</td>
<td>r5.large</td>
</tr>
<tr>
<td>Runtime/gradient</td>
<td>13.5 minutes</td>
<td>45 minutes</td>
</tr>
<tr>
<td>On-demand price/gradient</td>
<td>0.26 $</td>
<td>0.0945 $</td>
</tr>
<tr>
<td>Spot price/gradient</td>
<td>N/A</td>
<td>0.027 $</td>
</tr>
<tr>
<td>On-demand price/data pass</td>
<td>425.35 $</td>
<td>127.39 $</td>
</tr>
<tr>
<td>Spot price/data pass</td>
<td>N/A</td>
<td>35.99 $</td>
</tr>
</tbody>
</table>
### Numerical examples

<table>
<thead>
<tr>
<th></th>
<th>BP TTI 2007</th>
<th>BP 2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of shots</td>
<td>1641</td>
<td>1348</td>
</tr>
<tr>
<td>Instances/gradient</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Instance type</td>
<td>m5.xlarge</td>
<td>r5.large</td>
</tr>
<tr>
<td>Runtime/gradient</td>
<td>13.5 minutes</td>
<td>45 minutes</td>
</tr>
<tr>
<td>On-demand price/gradient</td>
<td>0.26 $</td>
<td>0.0945 $</td>
</tr>
<tr>
<td>Spot price/gradient</td>
<td>N/A</td>
<td>0.027 $</td>
</tr>
<tr>
<td>On-demand price/data pass</td>
<td>425.35 $</td>
<td>127.39 $</td>
</tr>
<tr>
<td>Spot price/data pass</td>
<td>N/A</td>
<td>35.99 $</td>
</tr>
</tbody>
</table>

#### Serverless approach:

- Full flexibility
- On-demand + spot instances or combination of both
- Large number of instance types (memory, compute, GPU, HPC nodes)
- Adjust resources according to priority: cost, turn-around time, importance, etc.
Cost comparison

Compute 100 gradients for BP model:

- 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
Strong scaling - MPI

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

Seismic imaging in the cloud:

- Need to rethink how to bring software to the cloud
- Lift and shift approach not ideal (complexity, resilience, cost)
- Instead: take advantage of new cloud technologies
- High-throughput batch computing, serverless/event-driven computations, object storage, spot instances
- Only pay what you use: up to 10x cost reduction
- **Software based on separation of concerns + abstractions is prerequisite to go serverless**
Future directions

Go large:

- Collaboration with cloud providers to run at industry-scale
- 3D TTI RTM and LS-RTM
- SEAM model: long offset data acquisition w/ 3D elastic modeling
- Keynote speech at 4th EAGE workshop on HPC for Upstream
  (Dubai, Oct. 8, presented by F. J. Herrmann)

Check for updates on our website and on Researchgate:

https://slim.gatech.edu/
https://www.researchgate.net/lab/SLIM-Felix-J-Herrmann
Acknowledgments

This research was funded by the Georgia Research Alliance and the Georgia Institute of Technology.