An analysis of the impact of Virtual Machines allocation on the cloud: The case of Full Waveform Inversion on Amazon Web Services

Willian Massahiro Hayashida (CEPETRO/UNICAMP), Alexandre Camargo (CEPETRO/UNICAMP), Charles B. Rodamilans (CEPETRO/UNICAMP), Tiago A. Coimbra (CEPETRO/UNICAMP), José Ribeiro (CEPETRO/UNICAMP), Caian Benedicto (CEPETRO/UNICAMP), Martin Tygel (CCES/CEPID and CEPETRO/UNICAMP), and Edson Borin (CEPETRO/UNICAMP and IC/UNICAMP)

This paper was prepared for presentation at the 16th International Congress of the Brazilian Geophysical Society, held in Rio de Janeiro, Brazil, August 19-22, 2019.

Abstract

In order to optimize free cash flow, today, companies are reducing the capital expenditure (CAPEX) by outsourcing their computational processes. Additionally, cloud computing offers an attractive possibility of on-demand use of up-to-date virtual-machine clusters designed by and tailored to the user needs. In this way, it avoids the problem of acquiring, maintaining and updating in-house hardware. Cloud-computing clusters are realized by means of virtual machines that are allocated on physical servers, which are located on the cloud provider data centers. Their specific location may affect network communication performance and optimal virtual machine allocation depends on the specifics of the computational task, as well as the possibilities offered by the cloud provider. Some cloud providers allow users to choose among different allocation strategies when instantiating their set of virtual machines. In this work, we analyze the impact of different virtual machine allocation strategies on the performance and cost of executing a seismic imaging application, namely full waveform inversion (FWI), on a specific cloud provider, namely Amazon Web Services (AWS). Our results indicate that the virtual machine allocation strategy may significantly affect the performance and the cost of executing the FWI application on the cloud, which may cause the execution to be up to 3.1 times more expensive.

Introduction

Due to the high cost of acquisition and maintenance, high-performance computing clusters are usually limited to a few groups. Moreover, such machines suffer from the depreciation process and with that, it requires new investments, i.e., increase of the capital expenditure (CAPEX). Nonetheless, the cloud computing model (Netto et al., 2018) is allowing anyone to access high-performance computing systems as a service, paying only for what they use and with no need to acquire, install and maintain expensive computing hardware. In this sense, the cloud computing model has the power to turn high-performance computing into an accessible and popular service. When using the cloud, the user may instantiate a set of virtual machines (VMs) and connect them through a virtual network to implement its own high-performance computing cluster. Hence, adapting the system to the user's software is a matter of selecting the proper hardware for the virtual machines and configuring the network for the desired behavior, which is usually not possible when using in-house high-performance computing system. Moreover, there is no need to wait on processing queues, since each user may have instant access to its own cluster and pay only for what they use.

Virtual machines instantiated by users are allocated by the service provider on physical servers that are located on data centers. VMs that are allocated close to each other, i.e. on physical servers that are close to each other, usually have lower communication latency than VMs that are allocated far apart from each other. Hence, the allocation strategy affects the VM's communication performance, which may also affect the application's performance, specially if the performance of the application is highly dependent on the system communication's performance. In order to mitigate this problem, cloud computing providers are offering services that allow users to indicate the strategy that should be employed to allocate their virtual machines on the data centers. Therefore, selecting the proper strategy may be important to improve the application's performance.

Many seismic applications rely on high-performance computing (HPC) resources to process seismic data. The full waveform inversion (FWI) method is an example of such application and, due to its relevance (Peng et al., 2018), it is a case of interest when analyzing the impact of virtual machine allocation strategies on the performance of a high-performance application on the cloud.

In this work, we investigate how different virtual machine allocation strategies may affect the performance and cost of executing a seismic imaging application on the Amazon Web Services (AWS) and we show that the virtual machine allocation strategy may significantly affect the performance and the cost of executing the FWI application on the cloud.

FWI problem

Proposed by Tarantola (1984), the full waveform inversion (FWI) method is designed to invert subsurface elastic parameters (e.g, acoustic/elastic velocities and density) from seismic marine or land data acquired at the surface.

For a dense distribution of sources and receivers, data acquisition consists of so-called shot records, namely the reflection response of the subsurface medium due a single source is recorded by an ensemble of receivers for a given recording time. Seismic data samples have the form $u(s,r,t)$, in which u is the recorded amplitude (or trace), s and r are the source and receiver locations at the surface and t is the time sample. As such, seismic data occupy a five-dimensional volume (two coordinates each for source and receiver position) and one coordinate for time. Hence, seismic data can be huge, of the order of terabytes.

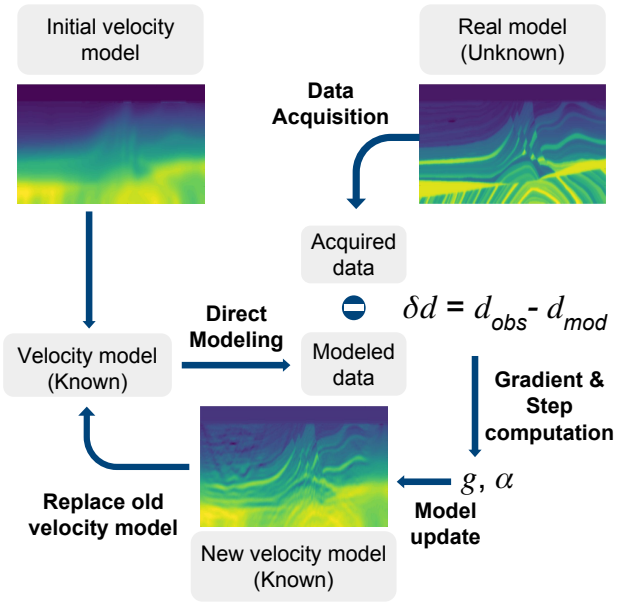


Figure 1: Full waveform inversion computing flowchart, where d_{obs} is the observed data, d_{mod} is the modeled data, g is the descent direction based on gradient of misfit function and α is the step length.

FWI follows the pattern of a classical inversion problem approach: an initial grid model of the subsurface is assumed with user-selected parameters assigned at a grid point. Under the use of the same source and receiver configuration as the original acquisition, shot records are simulated by solving the wave equation applied to that model. As a second step, the obtained shot records are compared with the corresponding observed ones. Using least squares optimization schemes, the misfits between the simulated and original shot records are applied to produce a new model with updated parameters. The process is the iterated until a pre-assigned stop criterion. A cartoon of the iterative scheme is provided in Figure 1. For the purposes of the present paper, the above brief description of FWI is an adequate choice. For a thorough and comprehensive explanation of the subject, the reader is referred to Virieux et al. (2014).

The whole process requires a huge amount of operations and a high computational power is usually necessary to enable its execution in a reasonable amount of time. To address this problem, this technique is often parallelized and executed on high-performance computing clusters.

For simplicity, but without loss of generality, we consider a FWI cloud computation applied to the Marmousi model, a public-domain, high-complexity, and 2D-acquisition acoustic data set (Versteeg, 1994). For the computation, we used the TOY2DAC 2D acoustic frequency-domain full waveform modeling and inversion software (Métivier et al., 2014). We investigate the impact of different virtual machine allocation strategies on the performance of TOY2DAY under the use of the *Amazon Web Services* (AWS) cloud.

AWS Cloud

The AWS is a cloud computing provider that have multiple data centers worldwide. Its data centers are organized in regions and each region may have one or more availability zones, which contain the data centers themselves. Each data center has multiple physical servers (computers). When instantiating a VM, the user is required to indicate the desired region. By default, AWS may select any arbitrary physical server into the data center to allocate this VM. However, the location of the virtual machines affects the performance of the communication when instantiating multiple VMs to create a high-performance cluster. As a consequence, the performance of the software may also be affected.

In order to guide the VM allocation process, the user may: a) indicate a placement group mode, or b) acquire a dedicated host or instance, as illustrated in Figure 2.

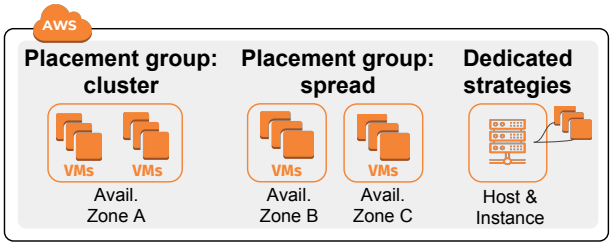


Figure 2: Virtual Machine allocation strategies.

The placement group mode allows the users to indicate whether they want their VMs placed as close as possible to each other (cluster) or in a distributed fashion (spread), for example. The cluster placement group mode is usually helpful to reduce the communication latency between VMs while the spread mode is useful to increase availability, since VMs are likely to be placed in physical servers located on different racks or data centers in the same region.

Microprocessor virtualization technologies allow AWS to instantiate multiple VMs in the same physical server without users knowing that they are sharing the same hardware. This technology allows AWS to improve utilization of its data centers, however, in some cases it may affect the performance of VMs (*e.g.* when one of the VMs is aggressively consuming the microprocessor data cache). In order to prevent these problems, users may acquire specialized services in which the physical server is fully dedicated to its VMs. AWS allows two modes: dedicated host and dedicated instance.

The dedicated host mode allocates a physical server that will be fully dedicated to a single user. Also, in this mode, the user may select which VMs will be allocated on the given physical server and may instantiate VMs to the same

physical server over time. The dedicated instance mode guarantees that only VMs from the same user will be allocated on the same physical server. The main difference from the dedicated host mode is that, with a dedicated host, the user has visibility and control over how instances are placed on the server.

Related Work

Binotto et al. (2015) presented a method to support users in deciding how to distribute computational jobs between on-premise and cloud resources using an FWI application as a case study. Such method and experimental results rely on the cost ratio between executing the application on the local and the cloud infrastructure. Nonetheless, they do not evaluate the impact of VM allocation strategies on the cost and performance of the cloud.

In 2008, Evangelinos and Hill (2008) investigated the feasibility of using cloud computing resources to execute HPC workloads. They executed Coupled Atmosphere-Ocean Climate Models and analysed the network performance under different MPI implementations. They concluded that the network performance was one to two orders of magnitude below the level encountered in dedicated systems or supercomputers. At that time, there was no API support to ensure VMs were placed near each other, even so, the authors concluded that the results were encouraging.

Zaspel and Griebel (2011) investigated the appropriated AWS cloud instances for the computational fluid dynamics (CFD) problem with the NaSt3DGPF software. They used a cluster placement group to make sure that the instances were at a single location and had fast interconnect. They observed that there was a limit of network performance when they tried to use more instance than AWS service default limits without a formal solicitation to use more instances (without pre-order solicitation). They argued that the cloud is appropriate "for moderately sized parallel CFD problems on up to 64 CPU cores or 8 GPUs" and the cloud computing is a viable alternative for mid-sized HPC applications. Different from our work, they did not evaluate the cluster and spread placement strategies.

Folgar et al. (2017) investigated the latency impact of allocation strategies on Bcast and Exchange MPI collectives in a physical cluster and a private cloud environment. By using a private cloud created with Apache CloudStack and Intel MPI Benchmarks (IMB), they observed that the physical cluster (cluster without virtualization) offered a lower communication latency (up to 78% of improvement) than the private Apache CloudStack based cloud system. They also compared the cluster and spreed placement strategies in the private cloud and concluded that the cluster strategy has a minor latency for message size smaller than 4 KiB (79% of improvement) and the spreed strategies has a minor latency for messages larger than 4 KiB (33% of improvement). The authors did not compare dedicated strategies (host and instance), nor investigated a public cloud. Also, they did not use a real application (like FWI or CFD) in their experiments.

Materials and Methods

We selected three different server types from AWS to conduct our experiments, as described in Table 2. The first row shows the name of each server type and the second one indicates the number of virtual CPUs (a.k.a. microprocessor cores); the third row states the amount of main memory and the remaining ones indicate the price per hour for each allocation mode. There are no extra costs when specifying a placement group and the user is basically charged by the regular On-Demand price. These prices are based on the North Virginia region on January 27th, 2019.

Type	c5.2xlarge	r5.2xlarge	z1d.2xlarge
vCPUs	8	8	8
RAM (Gib)	16	64	64
On-Demand Price (U$/h)	0.340	0.504	0.744
Ded. Instance Price (U$/h)	3.366	6.653	4.910
Ded. Host Price (U$/h)	0.360	0.534	0.789

Table 1: AWS server types

This present application of the TOY2DAC software uses the Metis (Karypis and Kumar, 1998), Intel MKL (Intel, 2018), Mumps (Amestoy et al., 2001) and a set of optimization tools called Seiscope Optimization Toolbox (Métivier and Brossier, 2016). We compiled the Metis library using the GCC 5.4.0 compiler and the remaining ones using the Intel compilers: mpiicc e mpiifort.

In the experiments, the Marmousi data set was sampled for a grid of 681 $\times$ 141 points. The result after 20 iterations for two frequencies ($\omega = 3$ Hz and 5 Hz) is illustrated in Figure 3.

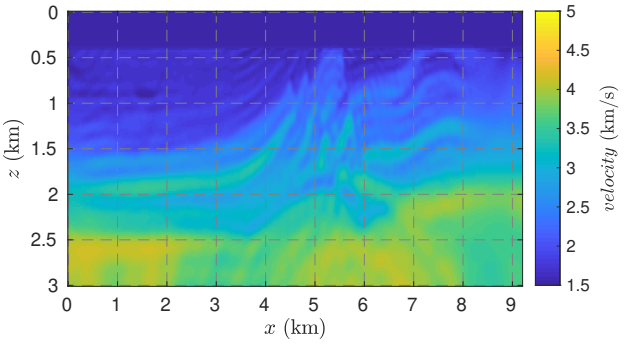


Figure 3: Velocity model estimated by TOY2DAC.

The application code was designed to run in parallel machines using both the Message Passing Interface (MPI) and the OpenMP programming APIs. Hence, it is capable of exploring parallelism on distributed and shared memory computers.

We first created a virtual machine disk image with Ubuntu 16.04 LTS and installed the TOY2DAC and the required libraries. Then, in order to investigate the impact of the

VM allocation strategies on the application performance, we executed 8 MPI processes on 4 VMs, being 2 processes per VM, and 4 OpenMP threads per process, a total of 32 threads (8 per VM). We execute the application three times and compute the median and the standard deviation.

Also, we used the ping Linux tool to measure the communication latency between each pair of VM and we use it as a metric of inter VM communication performance. We first compute the average time required to send 20 packages of 64 bytes each between two VMs. Finally, we compute the average latency between all 4 VMs and the standard deviation.

Experimental Results

Table 2 presents the average latency between VMs measured when instantiating the 4 VMs using different allocation strategies. It is possible to notice that there is a clear performance difference between allocation strategies and that the dedicated host and dedicated instance approaches provide the lowest latencies. Also, the cluster mode provides, in general, better latencies than the spread and the default modes.

Type	c5.2xlarge	r5.2xlarge	z1d.2xlarge
cluster	0.090 ± 0.007	0.100 ± 0.005	0.065 ± 0.010
spread	0.159 ± 0.025	0.099 ± 0.006	0.146 ± 0.024
ded. host	0.061 ± 0.011	0.056 ± 0.009	0.052 ± 0.006
ded. inst.	0.059 ± 0.007	0.054 ± 0.008	0.053 ± 0.006
default	0.168 ± 0.026	0.104 ± 0.006	0.150 ± 0.032

Table 2: Average ping latency between VMs (ms)

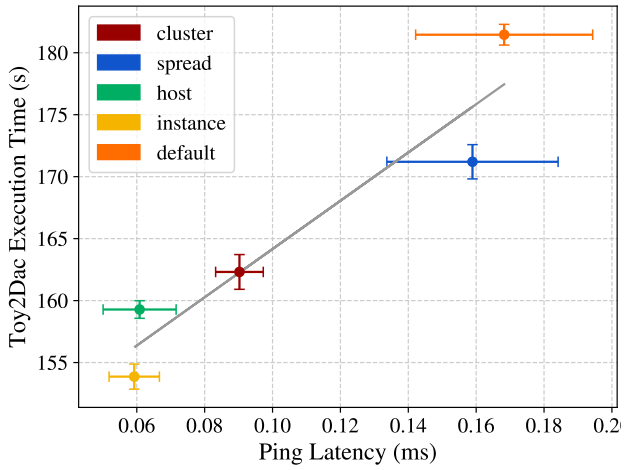
Table 3 shows the time it took to execute the TOY2DAC application for each allocation strategy and VM type.

Type	c5.2xlarge	r5.2xlarge	z1d.2xlarge
cluster	162.312	177.085	140.873
spread	171.201	176.336	154.856
ded. host	159.281	160.679	147.777
ded. inst.	153.865	160.734	146.003
default	181.456	177.734	158.949

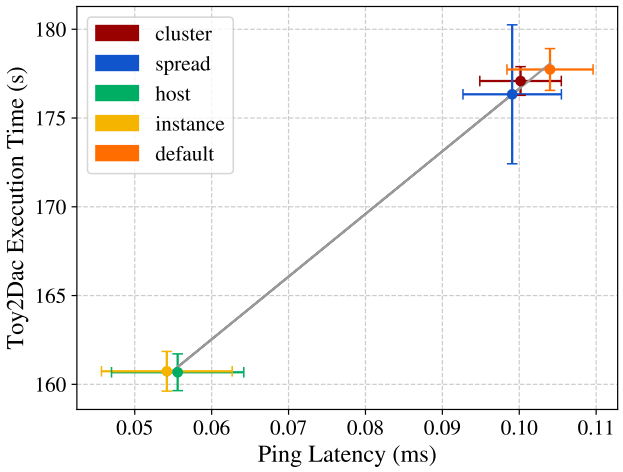
Table 3: TOY2DAC execution time (s)

By comparing Tables 2 and 3 it is possible to notice that there is a correlation between the average communication latency among VMs and the performance of the TOY2DAC application. Figure 4 plots the TOY2DAC execution time against the average latency between VMs and suggests that this relation is almost linear.

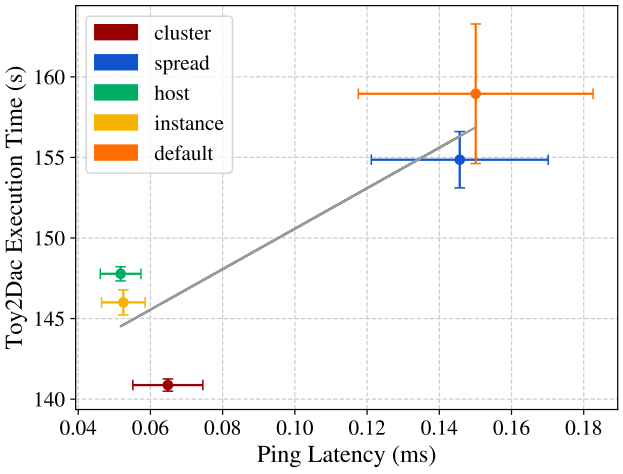
The result presented in Figure 4 corroborates our hypothesis that the allocation strategy may affect communication latency and, as a consequence, the performance of the application. Also, these results indicate that the dedicated host and dedicated instance modes provide lower latency and reduced execution times. Nonetheless, the increased prices associated with these two modes may cause the total execution cost (calculated by multiplying the execution time by the price per hour) to be higher than the other modes.



(a) c5.2xlarge server type



(b) r5.2xlarge server type



(c) z1d.2xlarge server type

Figure 4: Average ping latency between VMs versus performance of the FWI application.

Table 4 presents the total cost normalized by the best allocation mode for each VM type. For example, the dedicated host mode costs 2.44 times more than the cluster mode when executing the TOY2DAC on a

c5.2xlarge VM type.

Type	c5.2xlarge	r5.2xlarge	z1d.2xlarge
cluster	1.00	1.04	1.00
spread	1.07	1.04	1.10
ded. host	2.44	3.13	1.74
ded. inst.	1.02	1.00	1.10
default	1.13	1.05	1.13

Table 4: TOY2DAC execution cost normalized by the best allocation mode for each VM type

Table 4 indicates that the dedicated host mode causes the execution cost to be 1.7 to 3.1 times more expensive than the best mode for each VM type. Nonetheless, the other allocation strategies provide very similar costs (less than 13%).

Even though the price per hour of the dedicated instance mode is higher than the cluster, spread and default modes, its reduced execution time cause it to offer a competitive cost in this experiment.

Figure 5 shows the total execution cost versus the execution time for each allocation strategy and VM type. These results indicate that the dedicated instance mode offers a good trade-off between performance and cost. The dedicated host mode offers, in general, better performance, however, its cost is usually high. The spread and the default modes do not provide good performance and the cost is often a little higher than the ones achieved by the dedicated instance and the cluster mode. Finally, the cluster mode provides a very good execution cost, but its performance may vary drastically, being the best one when using z1d.2xlarge server types and one of the worse when using r5.2xlarge server types.

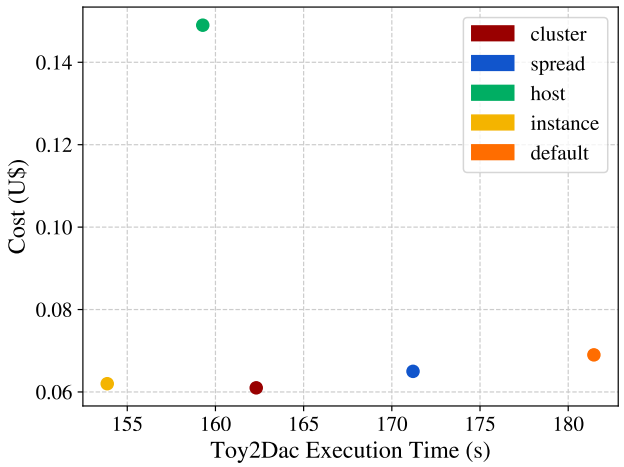
The experimental results indicate that the virtual machine allocation strategy may affect both: the execution time and the cost of executing high-performance parallel code on the cloud.

Conclusions

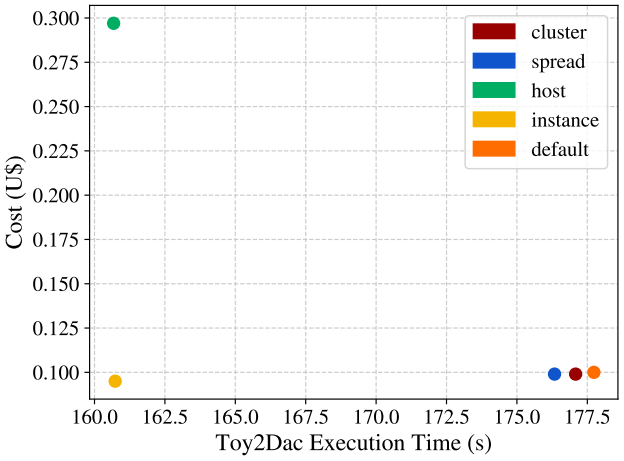
In this work, we explored the impact that different virtual machine allocation strategies have on the performance and cost of executing a high-performance geophysics software on the AWS cloud. We evaluated how 5 different allocation strategies (default, cluster, spread, dedicated host and dedicated instance) affect the cost and the performance of an FWI application on three different virtual machine types.

Our results indicate that the virtual machine allocation strategy may significantly affect the execution time and the cost of executing a high-performance parallel application on the cloud. For the experimental setup used in our experiments, the default allocation strategy was, on average, 1.10 to 1.17 times slower than the fastest strategy. Also, the dedicated host strategy was 1.7 to 3.1 times more expensive than the cluster strategy. The cluster strategy, on the other hand, offered the best costs but its performance is not always good. Finally, the dedicated instance mode offered a very good trade-off between cost and performance.

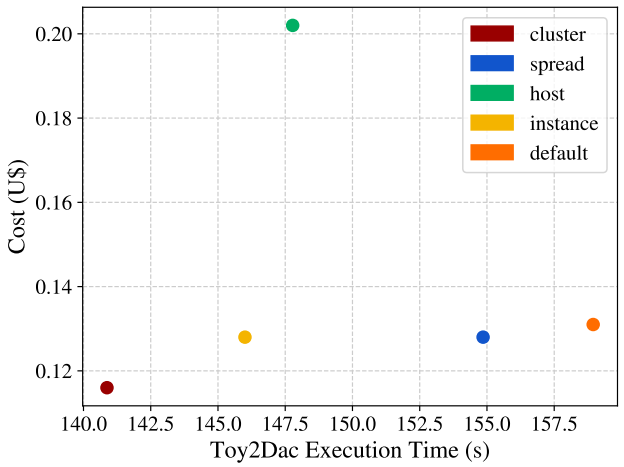
In future works, we plan to explore different applications and virtual machine types to make these conclusions more general.



(a) c5.2xlarge server type



(b) r5.2xlarge server type



(c) z1d.2xlarge server type

Figure 5: Execution cost versus execution time for each allocation strategy and VM type.

Acknowledgments

This work was possible thanks to the support of Petrobras, Fapesp (2013/08293-7), CAPES (2966/2014), and CNPq (313012/2017-2). The authors also thank the High-Performance Geophysics (HPG) team for technical support.

References

Amestoy, P. R., I. S. Duff, J.-Y. L'Excellent, and J. Koster, 2001, A fully asynchronous multifrontal solver using distributed dynamic scheduling: SIAM Journal on Matrix Analysis and Applications, **23**, 15–41.

Binotto, A., L. Tizzei, K. Mantripragada, and M. Netto, 2015, Supporting the scheduling over local HPC and cloud platforms: An FWI case study: Presented at the Second EAGE Workshop on High Performance Computing for Upstream.

Evangelinos, C., and C. Hill, 2008, Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2: ratio, **2**, 2–34.

Folgar, F. G., G. I. Fernandez, J. I. Z. Avila, N. S. Iglesias, A. J. G. Loureiro, and T. F. Pena, 2017, A study of the influence of vm allocation policies on mpi bcast and mpi exchange latency in cloud: IEEE Latin America Transactions, **15**, 1490–1496.

Intel, 2018, Intel math kernel library: `https://software.intel.com/en-us/mkl`.

Karypis, G., and V. Kumar, 1998, A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices: University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN.

Métivier, L., F. Bretaudeau, R. Brossier, S. Operto, and J. Virieux, 2014, Full waveform inversion and the truncated newton method: quantitative imaging of complex subsurface structures: Geophysical Prospecting, **62**, 1353–1375.

Métivier, L., and R. Brossier, 2016, The seiscope optimization toolbox: A large-scale nonlinear optimization library based on reverse communication: Geophysics, **81**, F1–F15.

Netto, M., R. Calheiros, E. Rodrigues, R. Cunha, and R. Buyya, 2018, Hpc cloud for scientific and business applications: Taxonomy, vision, and research challenges: ACM Computing Surveys (CSUR), **51**, 8.

Peng, C., M. Wang, and A. Gomes, 2018, Subsalt imaging improvement possibilities through a combination of fwi and reflection fwi: The Leading Edge, **37**, 52–57.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

Versteeg, R., 1994, The marmousi experience: Velocity model determination on a synthetic complex data set: The Leading Edge, **13**, 927–936.

Virieux, J., A. Asnaashari, R. Brossier, L. Métivier, A. Ribodetti, and W. Zhou, 2014, An introduction to full waveform inversion, *in* Encyclopedia of exploration geophysics: Society of Exploration Geophysicists, R1–1–R1–40.

Zaspel, P., and M. Griebel, 2011, Massively parallel fluid simulations on amazon's hpc cloud: 2011 First International Symposium on Network Cloud Computing and Applications, IEEE, 73–78.