# Harnessing Cloud Computing for Geophysical Exploration

Rodrigo C. Machado*, Cristiano A. Kunas*, Arthur F. Lorenzon*, Alexandre Carissimi*, and Philippe O. A. Navaux*
*Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
{*rcmachado, cakunas, aflorenzon, asc, navaux*}*@inf.ufrgs.br*

***Abstract*—Geophysical exploration methods are important in discovering essential resources like oil and gas. However, traditional exploration often involves environmentally detrimental practices. To address this, software solutions simulate seismic imaging techniques for oil detection. In this scenario, the industry is now transitioning these applications to cloud-based Software as a Service (SaaS) models, offering benefits like resource optimization, eco-friendliness, and advanced data analytics. This shift, however, presents challenges in performance, scalability, and cost management. This paper presents a case study on the Fletcher modeling application as a SaaS in geophysical exploration, exploiting cloud hardware heterogeneity. Through an extensive set of experiments on the Google Cloud instances with different multicore processors, we show that the Fletcher SaaS model scales with increased hardware resources, with AMD instances offering a better performance-cost trade-off than Intel.**

***Index Terms*—Cloud Computing, Performance, Fletcher, Geophysical Exploration**

## I. Introduction

Geophysical exploration methods play an important role in advancing human civilization by facilitating the discovery of vital resources for the economic development of nations, including commodities like oil and gas. Nevertheless, investigating new oil reservoirs frequently requires the employment of intrusive practices, including the excavation of drilling sites in ecologically sensitive areas and the improper disposal of waste materials. In response to these environmental challenges and in pursuit of more sustainable and precision-driven exploration practices, software solutions are designed to replicate and simulate seismic imaging techniques, primarily focused on oil detection. By harnessing the capabilities of such applications, the geophysical exploration industry aims not only to reduce the negative ecological impacts associated with traditional methods but also to significantly enhance the accuracy and efficiency of drilling operations. Therefore, as the geophysical exploration industry strives to address such challenges, a compelling imperative emerges regarding the migration of these applications to the cloud [1].

The integration of geophysical exploration methods with cloud-based Software as a Service (SaaS) not only yields efficiency optimization but also signifies a substantial stride toward environmentally conscientious, agile, and forward-looking exploration methodologies. By harnessing cloud computing, these applications stand to benefit from inherent advantages, including on-demand resource allocation, parallel processing capabilities, and robust data analytics. Consequently, this transition fosters a more eco-friendly approach, as it minimizes the environmental footprint associated with traditional on-premises data centers and computational infrastructure. Furthermore, it empowers practitioners to achieve heightened operational flexibility, real-time collaboration, and rapid scalability, thereby advancing the state of the art in geophysical exploration while aligning with contemporary sustainability objectives and best practices.

However, the transition to a SaaS model in geophysical exploration presents challenges. The secure management of large, sensitive datasets is paramount, requiring advanced data security measures, encryption, and compliance with industry regulations [2]. Ensuring reliable, high-performance network connectivity in remote exploration sites is crucial for seamless data transfer and real-time collaboration. Maintaining consistent performance and availability of cloud services is essential to prevent disruptions [3], [4]. Cost management is also a concern, as scalable cloud resources can lead to unpredictable costs. In this scenario, striking the right balance between resource provisioning and cost containment is vital to realize the benefits of SaaS in geophysical exploration and to uphold performance standards.

Considering the aforementioned scenario, in this paper, we present a case study on the provision of the Fletcher modeling application as a SaaS in the geophysical exploration domain. For that, we provide different versions of the SaaS to exploit the heterogeneity of hardware resources available in the cloud. By doing so, we seek to demonstrate the practicality and benefits of this transition in a real-world context. (*ii*) delve into the critical discussion surrounding the trade-off between maintaining optimal performance and achieving cost-efficiency in a cloud-based environment. (*iii*) Outline and analyze the prospective challenges and emerging trends in the field of geophysical exploration SaaS. Anticipating and understanding these challenges is crucial for charting a path forward toward more sustainable and high-performance exploration practices in the cloud era.

Through the execution of the Fletcher SaaS model in different instances with distinct architectures from the *Google Cloud*, we show that it is able to scale as the number of hardware resources assigned to the instance increases. In the most significant case, it shows a speedup of $29.5\times$ over the sequential implementation on an instance with AMD Epyc processors. We also show that AMD instances are more capable of delivering a better trade-off between performance and costs than Intel ones to execute the Fletcher SaaS model. However, when comparing Intel to AMD instances, the effective utilization of hardware resources with increasing thread counts is better on Intel instances.

The remainder of this paper is organized as follows. In

979-8-3503-8160-3/23/$31.00 ©2023 IEEE
DOI 10.1109/SBAC-PADW60351.2023.00022

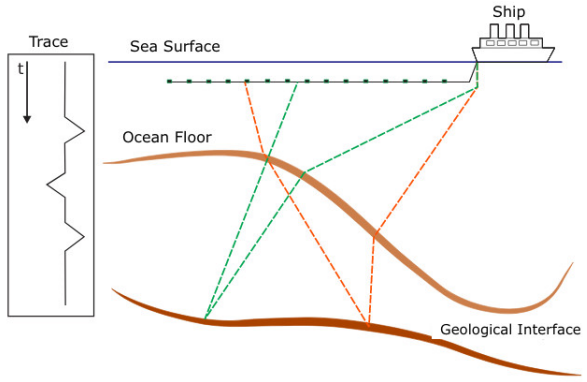



Fig. 1: Data collection in a marine seismic survey

Section II, we describe the SaaS and Fletcher model and list the Related Work. In Section III, we discuss the steps when providing the SaaS Fletcher. The evaluation is discussed in Section IV, while final considerations are drawn in Section V.

## II. Background

### A. Cloud Computing - SaaS

Cloud computing has emerged as a transformative paradigm in the field of information technology, reshaping the way computing resources are provisioned, managed, and utilized. This paradigm shift has implications for various industries and domains, revolutionizing traditional IT infrastructure and service delivery models. At the forefront of this transformation is the concept of Software as a Service (SaaS), a cloud computing model that has gained widespread adoption.

Unlike traditional software deployment, where applications are installed locally on individual devices or servers, SaaS offers a compelling alternative. In the SaaS model, software applications are hosted and maintained by cloud service providers, making them accessible to users via web browsers or APIs. This fundamental shift from ownership to subscription-based access heralds numerous advantages, such as eliminating the need for local installation and maintenance, reducing upfront capital expenditures, and enhancing scalability and flexibility.

### B. Fletcher Modeling

Fletcher modeling works as a technique for simulating wave propagation over time. This propagation is expressed through the acoustic equation **(Equation 1)**, where the velocity varies according to the specific geological layers **(Equation 2)**. Referring to the equations, $p(x, y, z, t)$ indicates the pressure at each location in the domain with respect to time, $V(x, y, z)$ is a representation of the propagation velocity, and $\rho(x, y, z)$ reflects the density [5].

$$\frac{1}{V^2}\frac{\partial^2 p}{\partial t^2} = \nabla^2 p \tag{1}$$

$$\frac{1}{V^2}\frac{\partial^2 p}{\partial t^2} = \nabla^2 p - \frac{\nabla \rho}{\rho} \cdot \nabla p \tag{2}$$

Seismic modeling initializes by collecting data in a seismic survey, as illustrated in **(Fig 1)**. The procedure begins with equipment attached to a ship, which at regular intervals emits seismic waves that reflect and refract in interactions with different environmental undergrounds, working as a sonar to map geological structures. When these waves return to the ocean's surface, specific sensors installed on cables towed by the ship capture and record seismic variations. These variations, a.k.a. seismic traces, correspond to the set of signals obtained by each sensor during the wave emission. Therefore, with each emission of waves, the seismic traces of all the microphones on the cable are recorded, providing an understandable overview of the subsoil. During this operation, the ship continues to move and emit signals periodically, thus producing a detailed image of the seabed and underground [6].

The algorithm Fletcher implements is based on the numerical solution of the wave equation. It is a partial differential equation that considers the environment's elastic properties (e.g., the propagation velocity of the wave) and is represented in a three-dimensional grid. The wave propagation process is iterative, where in each iteration, the algorithm calculates the approximate solution of the wave equation at each grid point, considering the information from previous iterations. During propagation, the wave energy spreads and changes as it interacts with the heterogeneities of the environment, updating the values at each grid point and allowing the algorithm to model seismic waves' reflection, refraction, and diffraction as they propagate underground.

We demonstrate the implementation of the Fletcher method using Algorithm 1. It takes the following input parameters: the number of iterations for wave propagation (*endTime*), the interval at which the wave state is saved to disk (*threshToWriteWave*), and the grid dimensions (*sx, sy,* and *sz*). The algorithm starts by initializing the grid with the physical properties of the environment using the *initializeGrid()* function. Then, a pressure point representing the amplitude of the seismic wave at a specific moment is inserted at the central position of the three-dimensional pressure vector. Prior to the kernel's execution, this three-dimensional array is mapped to a one-dimensional array, following the conventional order of (*x, y, and z*). In this mapping scheme, points along the *x*-axis of the grid are placed contiguously within the resulting one-dimensional vector.

The loop from line *3* to *10* is responsible for iterating until the simulation is performed. Then, for each iteration, a modulated Gaussian pulse representing the amplitude of the seismic wave at a given time instant is inserted in the center of the three-dimensional grid (*insertSourcePointToDevice()*). Then, the propagation of the seismic wave is based on the computation of a 5-point stencil during the Kernel execution. Once the point associated with the acoustic wave is computed, the wave state is propagated to the previous state to proceed with the next iteration. Furthermore, when the number of iterations reaches a defined threshold, the wave is written to the disk (*writeWave()*). To date, we have *Fletcher* available through parallel implementations with OpenMP and OpenACC for multicore architectures; and with OpenACC and CUDA for heterogeneous architectures.

### C. Related Work

Cloud computing has become a popular alternative to dedicated infrastructure for HPC applications [7]–[9]. There has been a noticeable shift from traditional software delivery models to SaaS in recent years, gaining popularity due to its

**Algorithm 1** Fletcher Implementation

**Input:** $endTime$: number of iterations the wave will propagate.
$threshToWriteWave$: number of iterations where the wave will be stored in disk.
$sx$: size of dimension x.
$sy$: size of dimension y.
$sz$: size of dimension z.

```
initializeGrid(grid, sx, sy, sz)
initPropagatePointers(grid, initPoint)
for each dt in endTime do
    insertSourcePointToDevice()
    propagateWave(...)
    updatePointers()
    if dt == threshToWriteWave then
        writeWave()
    end if
end for
```

benefits, including better scalability and accessibility, easier administration, and operational efficiency. It also eliminates the need for upfront investment and long-term commitment, making it an attractive option.

Previous works explore running HPC applications in the cloud, evaluate the current performance of HPC applications on existing cloud infrastructures, and discuss various techniques to mitigate interference, virtualization overhead, and issues due to shared resources in the cloud [9], [10]. However, there are few works that address the use of SaaS for such applications. Church et al. [11] developed a technology that exposes HPC applications as a service through a SaaS cloud to allow researchers easy access to cloud computing resources. Stavrinides and Karatza [12] evaluate the performance of a SaaS cloud under various delay limits and different levels of variability of the workload's computational demand.

In the Oil & Gas area, there is an increasing, but still small, transfer of the processing of geophysical exploration models to the cloud [13], [14]. Some studies have investigated the factors that influence cloud adoption in this sector [15]–[17].

## III. Fletcher SaaS Model

As already discussed throughout the previous sections, the modern oil and gas industry increasingly harnesses computational simulations and data analytics to optimize exploration and extraction processes. With the ever-evolving complexity of these tasks and the vast computational demands they present, the traditional model of localized software deployment is proving restrictive. Recognizing this shift, in this Section we describe the SaaS model tailored specifically for the Fletcher modeling, designed to operate seamlessly across diverse cloud environments and instances.

Our Fletcher SaaS model capitalizes on Docker, a leading containerization technology. Docker facilitates the encapsulation of applications and their intricate dependencies into isolated, portable containers. This ensures that the Fletcher modeling runs consistently, irrespective of the underlying cloud infrastructure. Moreover, Docker's lightweight nature, in comparison to traditional virtual machines, allows for rapid deployment and efficient use of system resources. Therefore, integrating Docker into the SaaS delivery model offers providers the flexibility to update, scale, and maintain their services seamlessly, ensuring a reliable and consistent user experience. By abstracting the complexities of the application

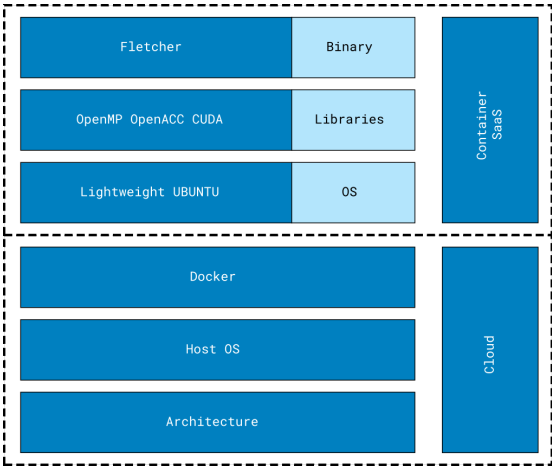



Fig. 2: Fletcher SaaS Stack

environment, Docker effectively eliminates compatibility issues, enabling Fletcher SaaS to offer a truly "*write once, run anywhere*" capability.

One of the primary advantages of this Docker-based SaaS deployment is its innate scalability. Given the unpredictable computational demands of oil and gas simulations, the ability to scale resources up or down based on real-time requirements is invaluable. Docker containers can be rapidly replicated and deployed across multiple cloud nodes, ensuring that computational bottlenecks are minimized and that simulations run efficiently. Furthermore, security, a paramount concern in the oil and gas sector due to the sensitive nature of exploration data, is bolstered in our Fletcher SaaS model. Docker's isolated environment means that each application instance runs in its own protected space, safeguarding against potential vulnerabilities and breaches. Cost-effectiveness is another compelling advantage. By allowing oil and gas firms to run simulations in a SaaS model on any cloud platform, they can leverage competitive pricing, avoiding vendor lock-in and optimizing operational expenses.

Figure 2 illustrates the stack of the Fletcher Saas, where the container has the Fletcher application binary and the parallel programming libraries employed to parallelize it (OpenMP, OpenACC, and CUDA, as discussed in Section II). It also contains a lightweight Ubuntu OS version. The image of this container can be obtained through the official repository[1]. In order to execute the Fletcher SaaS, we illustrate in Figure 3 the execution flow. The first step is to access the Cloud service, which can be any environment with Docker support (e.g.,

[1]https://hub.docker.com/r/rdrigomachado/fletcher-cuda12-min

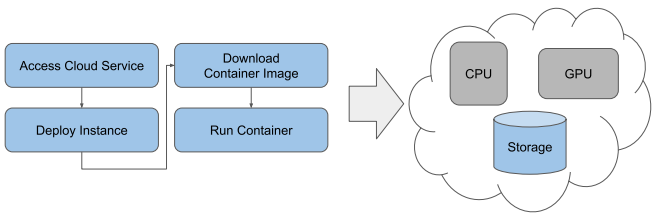



Fig. 3: Workflow execution of Fletcher SaaS

TABLE I: Main characteristics of each instance

| Instance | *n2d-std-16* | *n2d-std-32* | *n2d-std-64* | *n2d-std-128* | *n2d-custom-16* | *n2d-custom-24* | *n2d-custom-48* | *c2-std-30* | *c2-std-60* | *c2-custom-15* | *c2-custom-30* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Processor** | AMD Epyc | | | | | | | Intel Xeon | | | |
| **Microarchitecture** | Zen 2 | | | | | | | Cascade Lake | | | |
| **#Virtual CPUs** | 16 | 32 | 64 | 128 | 16 | 24 | 48 | 30 | 60 | 15 | 30 |
| **#Phys. CPUs** | 8 | 16 | 32 | 64 | 16 | 24 | 48 | 15 | 30 | 15 | 30 |
| **Main memory** | 64GB | 128GB | 256GB | 512GB | 128GB | 128GB | 128GB | 120GB | 240GB | 120GB | 120GB |
| **Price per hour** | 0.27$ | 0.54$ | 1.08$ | 2.16$ | 1.02$ | 1.46$ | 2.93$ | 1.25$ | 2.51$ | 1.25$ | 2.51$ |

Microsoft Azure, Google Cloud, and Amazon AWS). Then, the second step consists of deploying a given instance. At this point, the user can choose to (*a*) deploy the instance with the Fletcher SaaS container during the initialization of the instance or (*b*) deploy the instance alone and then, when accessing the instance, download the Fletcher SaaS container image through *docker* commands. Once the container is accessible in the instance, the user can run it, selecting the appropriate hardware: *CPU* in the case of OpenMP and OpenACC implementations, or *GPU* with the OpenACC and CUDA implementations.

## IV. Evaluation

### A. Methodology

We have performed the experiments on eleven instances from the Google Cloud, as depicted in Table I. Each instance was configured to deploy the Fletcher SaaS as a container during its initialization. Hence, to execute the Fletcher application, the only step was to run the container through the *docker run* command and start the application with the input parameters. We considered two different grid input sets: *small* ($344x344x344$) and *large* ($512x512x512$). All the other parameters regarding the initialization were used as defined in the original implementation. For each instance, we have executed the Fletcher SaaS with a different number of threads, varying from 1 to the maximum number of cores available in the instance. This number was set through the OMP_NUM_THREADS environment variable. Each combination of instance, number of threads, and input set was executed 30 times and we collected the execution time, the performance – measured through the number of samples computed per second (*msamples/s*), and calculated the cost to execute the application. The results in the next subsections consider an average of all the executions.

### B. Performance Results

We start by discussing the performance results achieved when each combination of the number of threads and input set is executed on each instance. For that, Figure 4 illustrates the obtained numbers for the AMD and Intel instances, respectively. These outcomes are showcased as speedups compared to the sequential execution (using 1 thread) on the corresponding instance. Therefore, the taller the bar, the better the performance improvements. Each bar represents the values for the execution with a given number of threads.

The first observation is that the performance of the Fletcher SaaS model scales with an increase in the number of threads, regardless of the instance type (be it *standard* or *custom*). Consequently, the best performance on any given instance is consistently achieved when all available hardware resources, such as cores and cache memories, are fully utilized. Compared to its sequential counterpart, the most significant performance improvement was observed on the *n2d-std-128* instance running with *128* threads across 128 virtual CPUs, marking a $29.5\times$ improvement over sequential execution. In the realm of Intel instances, the best performance improvement was $20.1\times$ over the sequential on the *c2-std-60* instance. In a head-to-head comparison between Intel and AMD architectures, AMD consistently outperformed, primarily attributed to its greater core count. Specifically, the AMD instance execution surpassed its Intel counterpart by a notable *46%*.

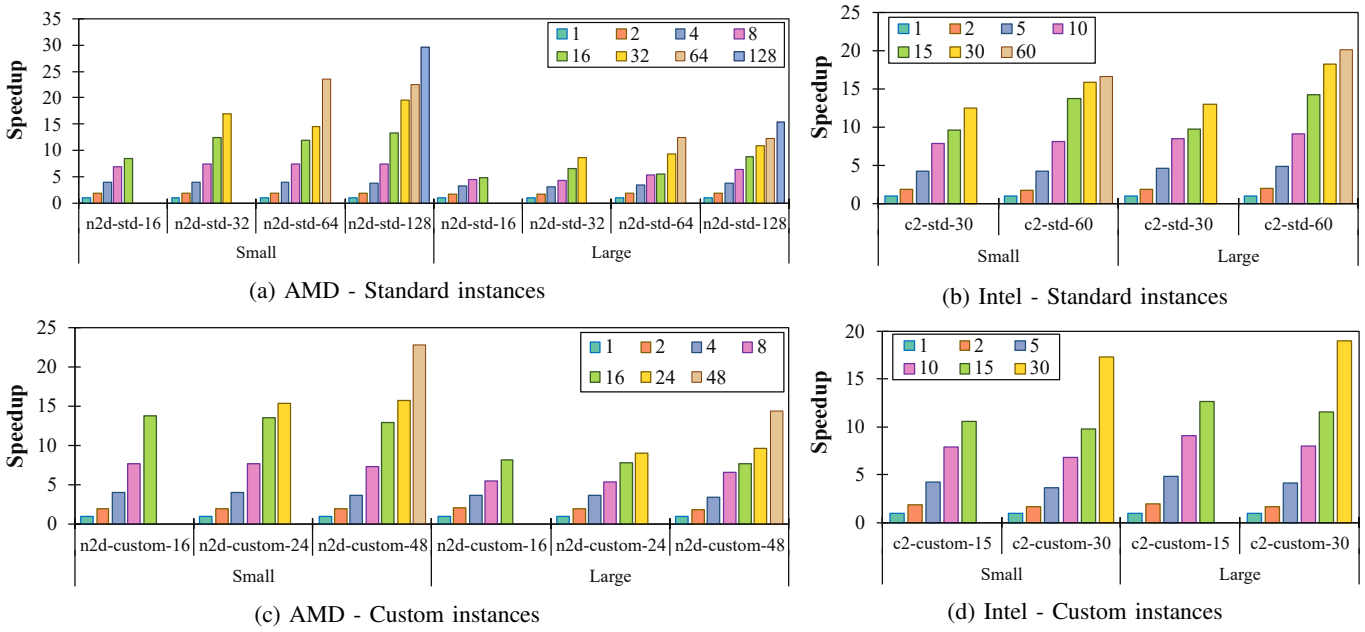
To examine the effective utilization of hardware resources (e.g., cores) with increasing thread counts, Figures 5 and 6 illustrate performance efficiency for AMD and Intel instances, respectively. Each graph showcases results for varying combinations of input sets, instance types (*standard* or *custom*), and thread numbers. Observing the standard instances (Figures 5.a, 5.b, 6.a, and 6.b), there is a noticeable decline in efficiency as the number of threads increases. However, this trend is less pronounced in custom instances, attributed to their *1 vCPU to 1 physical CPU* binding. Moreover, as the parallelism degree grows, Intel instances maintain better efficiency than AMD. This behavior may arise from AMD's memory hierarchy, where thread communication often takes place in regions that are more distant from the cores than in Intel's architecture.

### C. Trade-off between Performance and Cost-Efficiency

In the realm of cloud computing, the evaluation of parallel applications usually lies in striking a balance between performance and cost. While high-performing applications can improve user experience, they can also incur higher computational costs, especially when scaled across vast cloud resources. Conversely, cost-saving measures might compromise the quality of performance, potentially negating the benefits of parallel execution. Therefore, in this section, we assess the trade-off between these two metrics.

Figure 7 showcases the performance, denoted in MSamples/s on the *x-axis*, juxtaposed with the total cost, represented in $ on the *y-axis*, for the Fletcher SaaS model executed on both AMD and Intel instances. This evaluation is based on the small and large input sets using the number of threads that optimize performance for each instance. From the data, it is evident that no single instance excels in both top-tier performance and cost-efficiency simultaneously. The *n2d-std-128* configuration yielded the highest performance for both input sets, but it also incurred the highest cost for the large input set (similar to the Intel ones). Contrarily, the Intel instances, denoted by the *c2-* prefix, had comparable or even higher costs than their AMD counterparts, yet their performance was markedly inferior.

The insights drawn from Figure 7 emphasize the importance of holistic decision-making in cloud deployments. Stakeholders must weigh the dual imperatives of performance and cost, understanding that the highest-performing instance may not always represent the best value proposition. This underscores the need for comprehensive benchmarks tailored to specific




(a) AMD - Standard instances

(b) Intel - Standard instances

(c) AMD - Custom instances

(d) Intel - Custom instances

Fig. 4: Performance results for all instances, number of threads, and input set

applications, ensuring informed choices that balance both operational needs and budgetary constraints.

### *D. Future Directions and Challenges*

Considering the results presented in the previous subsections, one can envision future directions and challenges regarding the implementation of the Fletcher SaaS model.

**Adaptive Resource Allocation**. Develop a dynamic resource allocation strategy that can adjust based on the computational demands of the task at hand. By only using resources when they're needed, costs can be significantly reduced.

**Multi-Cloud Deployments**. Utilize multiple cloud providers to take advantage of the best features and pricing models of each. This approach also provides redundancy, ensuring high availability.

**Machine Learning-based Optimization**. Employ machine learning techniques to predict optimal configurations and resource needs based on historical data and specific task parameters. This predictive approach can lead to more efficient resource utilization, improving performance while curbing costs.

**Cost Analysis Tools**. Develop advanced tools that provide real-time insights into cost implications based on current resource usage. Such tools can guide decisions regarding when to scale resources up or down.

**Collaborative Cloud Models**. Investigate models where multiple stakeholders can collaboratively share cloud resources for their oil and gas simulations, effectively distributing costs.

**Open Standards and Interoperability**. Promote the use of open standards to ensure the SaaS model remains compatible across different cloud providers, enhancing portability and reducing vendor lock-in risks.

## V. Concluding Remarks

The evolution of geophysical exploration methodologies, driven by the imperative need for sustainable and precision-driven practices, has been significantly augmented by the integration of cloud-based SaaS models. In this paper, the Fletcher modeling application as a SaaS has underscored the potential of such a transition, especially when leveraging the heterogeneity of cloud hardware resources. We have shown the Fletcher SaaS model's scalability on Google Cloud, particularly with AMD instances, which showcases a promising trajectory for the industry. However, the challenges—ranging from data security to cost management—cannot be understated and require attention to ensure the seamless adoption of these technologies. Hence, in future work, we intend to perform a comparative analysis of other cloud platforms beyond Google Cloud could provide a more comprehensive understanding of SaaS performance in geophysical exploration. Additionally, as data security remains paramount, the development and integration of advanced encryption and compliance measures tailored for geophysical datasets will be crucial.

## Acknowledgements

This study was partially supported by Petrobras grant n.º 2020/00182-5, by CNPq/MCTI/FNDCT - Universal 18/2021 under grants 406182/2021-3 and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

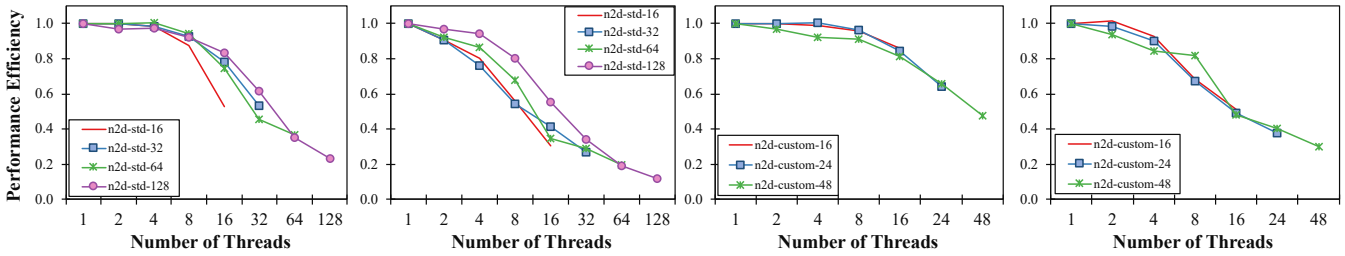
## References

[1] F. A. Portella and F. M. de Souza, "Reservoir simulation in the cloud," in *High Performance Computing in Clouds: Moving HPC Applications to a Scalable and Cost-Effective Environment*. Springer, 2023, pp. 265–282.

[2] C. Carpenter, "The cloud offers opportunity for oil and gas cybersecurity," *Journal of Petroleum Technology*, vol. 71, no. 08, pp. 72–73, 2019.

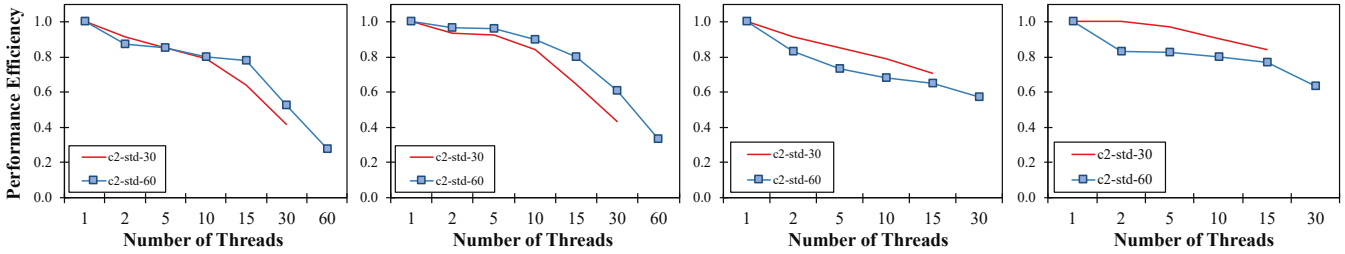
[3] P. O. A. Navaux, A. F. Lorenzon, and M. da Silva Serpa, "Challenges in high-performance computing," *Journal of the Brazilian Computer Society*, vol. 29, no. 1, pp. 51–62, 2023.

[4] A. F. Lorenzon and A. C. S. Beck Filho, *Parallel computing hits the power wall: principles, challenges, and a survey of solutions*. Springer Nature, 2019.



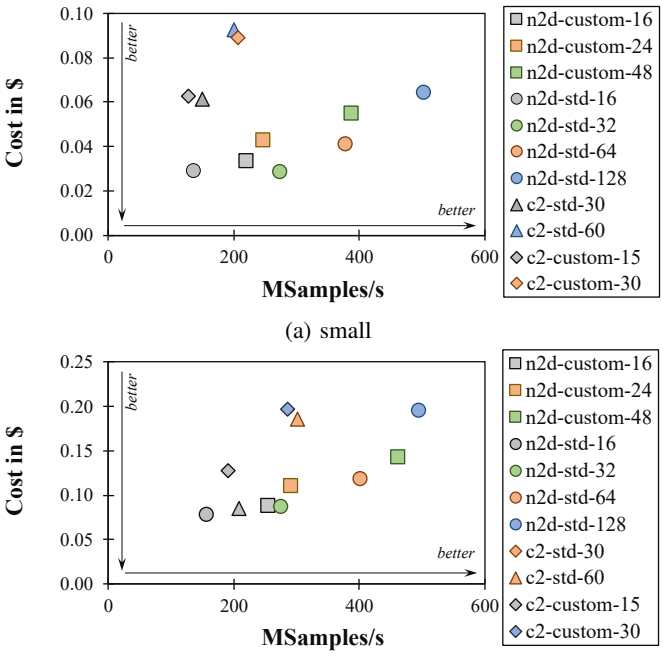

a) Small - standard b) Large - standard c) Small - custom d) Large - custom

Fig. 5: Performance efficiency of the AMD instances




a) Small - standard b) Large - standard c) Small - custom d) Large - custom

Fig. 6: Performance efficiency of the Intel instances




(a) small

(b) large

Fig. 7: Performance vs cost for each machine


[5] R. P. Fletcher, X. Du, and P. J. Fowler, "Reverse time migration in tilted transversely isotropic (TTI) media," *Geophysics*, vol. 74, no. 6, pp. WCA179–WCA187, 2009.

[6] C. Chu, B. K. Macy, and P. D. Anno, "Approximation of pure acoustic seismic wave propagation in tti media," *Geophysics*, vol. 76, no. 5, pp. WB97–WB107, 2011.

[7] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, "Performance evaluation of amazon ec2 for nasa hpc applications," in *Proceedings of the 3rd workshop on Scientific Cloud Computing*, 2012, pp. 41–50.

[8] E. Roloff, M. Diener, L. P. Gaspary, and P. O. Navaux, "Hpc application performance and cost efficiency in the cloud," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 2017, pp. 473–477.

[9] M. Kumaresan and G. P. Venkatesan, "Enabling high performance computing in cloud computing environments," in *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*. IEEE, 2017, pp. 1–6.

[10] A. M. Maliszewski, E. Roloff, E. D. Carreño, D. Griebler, L. P. Gaspary, and P. O. Navaux, "Performance and cost-aware hpc in clouds: A network interconnection assessment," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–6.

[11] P. Church, A. Wong, M. Brock, and A. Goscinski, "Toward exposing and accessing hpc applications in a saas cloud," in *2012 IEEE 19th International Conference on Web Services*. IEEE, 2012, pp. 692–699.

[12] G. L. Stavrinides and H. D. Karatza, "Performance evaluation of a saas cloud under different levels of workload computational demand variability and tardiness bounds," *Simulation Modelling Practice and Theory*, vol. 91, pp. 1–12, 2019.

[13] N. T. Okita, A. W. Camargo, J. Ribeiro, T. A. Coimbra, C. Benedicto, and J. H. Faccipieri, "High-performance computing strategies for seismic-imaging software on the cluster and cloud-computing environments," *Geophysical Prospecting*, vol. 70, no. 1, pp. 57–78, 2021.

[14] P. A. Witte, M. Louboutin, C. Jones, and F. J. Herrmann, "Serverless seismic imaging in the cloud," *arXiv preprint arXiv:1911.12447*, 2019.

[15] A. Guimaraes, L. Lacalle, C. B. Rodamilans, and E. Borin, "High-performance io for seismic processing on the cloud," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 18, p. e6250, 2021.

[16] S. N. Kayum and M. Rogowski, "High-performance computing applications' transition to the cloud in the oil & gas industry," in *the proceeding of IEEE High Performance Extreme Computing Conference (HPEC)*, 2019.

[17] C. Cheng, X. Sun, and Z. Shen, "Research on seismic wave reverse time migration data regularization and parallel computing in cloud environment," *Journal of Computational Methods in Sciences and Engineering*, vol. 19, no. 1, pp. 43–56, 2019.