Accelerating Multi-attribute Unsupervised Seismic Facies Analysis With RAPIDS

Otvio O. Napoli
Institute of Computing
UNICAMP
Campinas, Brazil
onapoli@lmcad.ic.unicamp.br

Vanderson Martins do Rosario
Institute of Computing
UNICAMP
Campinas, Brazil
vrosario@lmcad.ic.unicamp.br

Joo Paulo Navarro
NVIDIA
Campinas, Brazil
jpanavarro@nvidia.com

Pedro Mrio Cruz e Silva
NVIDIA
Campinas, Brazil
pcruzesilva@nvidia.com

Edson Borin
Institute of Computing
UNICAMP
Campinas, Brazil
borin@unicamp.br

Abstract—Classification of seismic facies is done by clustering seismic data samples based on their attributes. Year after year, the 3D datasets used in exploration geophysics constantly increase in size, complexity, and number of attributes, requiring a continuous rise in the classification efficiency. In this work, we explore the use of Graphics Processing Units (GPUs) to perform the classification of seismic surveys using the well-established machine learning method k-means. We show that the high-performance distributed implementation of the k-means algorithm available at the NVIDIA RAPIDS library can be used to classify facies of large seismic datasets much faster than a classical parallel CPU implementation (up to 258-fold faster in NVIDIA TESLAs GPUs), especially for large seismic blocks. We tested the algorithm with different real seismic volumes, including F3 Netherlands, Parihaka, and Kahu (from 12GB to 66GB).

I. INTRODUCTION

Lithofacies classification is a crucial step in seismic interpretation. The accurate classification of samples in terms of similarities leads to a better understanding of the areas of interest. Subsurface events contain valuable spatio-temporal information that has scientific and commercial importance, making accurate and fast interpretations a competitive advantage in exploration geophysics. Formally in seismic analysis, this process consisted of assigning lithofacies manually by human interpreters, following the amplitude responses. This labor-intensive task is being consistently improved by the use of automatic or semi-automatic interpretation tools. However, with the increasing quality of acquisition sensors, the size of 3D seismic surveys is facing a significant improvement in terms of definition, which ultimately leads to longer processing times. Furthermore, the interpretation quality may expressively be improved by adding new perspectives to the original amplitude volume, using, for example, derived attributes such as phase, frequency, and envelope. However, in the context of multi-attribute analysis, the computational complexity and requirements proportionally increase.

In the modern era of computing, massively parallel architectures are central in high-performance computing. The use of Graphics Processing Units (GPUs) at scalable infra-structures is enabling the processing of these increasing datasets. GPUs are well-known in the geophysics domain and are commonly used to solve inversion and migration problems such as reverse time migration [1], full-waveform inversion [2], Kirchhoff migration [3], and least-squares migration [4], for instance. We also find in literature efforts on porting attributes computations to GPUs as the work proposed by [5]. In their paper, an interactive-time curvature estimate was achieved maximizing the memory access pattern and loading data to GPU shared memory using a circular buffer.

Data-drive seismic attributes are well explored by the geophysics community. Unsupervised learning approaches may be used to categorize waveforms, or volume samples, in classes (clusters). The intuition behind is to use established algorithms such as Principal Component Analysis, or PCA [6], k-means [7] or Self-organizing Maps (SOM) [8] to automatically find the data distribution and properly classify the samples. Thus, similar feature vectors tend to be at the same cluster, indicating they have similar expressions [9]. For each volume of interest, we first train the algorithm and then perform predictions to classify the samples. The volume size has a direct impact on the attribute training time, meaning that very large datasets may become unfeasible to compute if we do not apply scalable parallelization strategies. For this reason, we present in this work a study exploring the large-scale implementation of k-means clustering available at the open GPU data science library named RAPIDS[9]. This package provides efficient machine learning algorithms running on multiple NVIDIA GPUs. With RAPIDS, we are able to train k-means over 3D seismic volumes up to 258x faster than conventional CPU versions.

The contributions of this work are summarized as follows:

- We tested and compared two open-source implementations of multi-attribute unsupervised seismic facies classification using k-means. One with a DASK-based CPU

https://rapids.ai/
machine learning implementation and other with a NVIDIA RAPIDS GPU-based implementation.

- We tested and analyzed the efficiency of the CPU k-means implementation approach in a 40-threads computer node, comparing it with the NVIDIA RAPIDS implementation with 2, 4, 8 and 16 NVIDIA TESLA GPUs. The tests were performed using three seismic datasets (F3 Netherlands, Parihaka, and Kaha) and we were able to show when using RAPIDS we can achieve up to 258x speedup over the CPU baseline approach.

The remaining of the text is organized as follows. We first describe the seismic facies classification problem, the algorithms that can be used, the pros and cons of k-means and some of its implementations in the **Seismic Facies Classification** Section. Then, in **Machine Learning Workflow** Section, we detail how we use the RAPIDS k-means implementation to perform seismic data classification. The **Experimental Setup** Section describes the computational and data resources that we used to perform our experiments. The **Experimental Results** Section presents and discusses the results we obtain followed by conclusions.

## II. SEISMIC FACIES CLASSIFICATION

The usage of unsupervised learning yielded good results in several approaches using volumetric data as input, such as XXX, YYY [1]. In general, these techniques benefit from the fact of being fast to train and predict and with the least use of computational resources compared to a very large models (e.g. deep learning models). Even so, the data size still growing either by new acquisition technologies, new formats and storage technologies or by new attributes that become relevant for analysis. TerraNubis [1], for instance, shows seismic datasets that at more than 200GB scale, which may be impossible to pre-process and extract features and train models to determined problems, with conventional tools (e.g. scikit-learn [1]).

Seismic facies classification is the problem of assigning specific classes to samples of seismic volumes based on their attributes. This classification allows the visualization of different geological settings, demanding complex analysis of enormous amounts of data. To handle the challenge of interpreting increasingly larger datasets, the use of machine learning has become an important tool. Unsupervised machine learning techniques are now widely used for this propose aiming to find natural clusters among different attributes that better highlight seismic patterns such as variances in amplitude and steeply dipping, low amplitude dipping areas, continuous dipping reflectors, among others. [10] compares a set of unsupervised learning techniques used to classify seismic facies including Self-Organizing Maps (SOM) and k-means and ANNs.

**SOM.** Also named Kohonen maps and developed by [8], is an Artificial Neural Network (ANN) used to produce a low-dimensional, discretized representation of the input space of the training samples. SOM applies competitive learning instead of using error-correction learning approaches (e.g. backpropagation), preserving the properties of the input space. This last property is very useful in the visualization task, especially because similar clusters end adjacent to each other in the latent space. SOM is widely used in seismic facies classification works [10]–[13].

**k-means.** Is an iterative clustering algorithm that minimizes the sum of distances from all points in a dataset to its clusters’ centroids. That is, k-means tries to minimize the following equation:

\[
E = \sum_{i=1}^{k} \sum_{j=1}^{n_i} d^2(x_{ij}, m_i),
\]

where \(E\) represents the sum of the euclidian distance from all samples in the dataset to the centroids, \(x_{ij}\) is the \(j\)th sample in the \(i\)th cluster, \(m_i\) is the center or mean of the \(i\)th cluster, \(n_i\) is the number of samples in the cluster, \(k\) is the number of clusters and \(d\) is the euclidian distance which is defined by:

\[
d^2(x_{ij}, m_i) = (x_{ij} - m_i) \ast (x_{ij} - m_i)^T.
\]

For each iteration of the algorithm, the centroids are moved towards the data center of mass and algorithm stops when reaching a given number of iterations or distances sum \(E\) reaches a defined convergence tolerance. In the end, the result is a set of clusters defined by the closest centroid that partitioning the data as well as possible.

In contrast to SOM, the number of clusters in k-means is a parameter that needs to be defined before its training phase. Moreover, k-means clustering has no structure (and consequently no relationship in the cluster numbering) usually generating a not ideal seismic facies visualization. In practice, k-means is normally applied to estimate the correct number of clusters in the data while SOM is used to generate the clusters for final visualization. However, despite its limitations, k-means is a very robust and widely available algorithm in several machine learning libraries and is much faster than SOM when handling large amounts of data. Thus, with the increasingly growth in the seismic data in the past years, dimensionality reduction techniques, as well as faster clustering algorithms, such as k-means, may be desirable, having in mind their limitations, in order to obtain an initial acknowledgment of visualization of different properties present in the seismic data. Thus, with the growth of seismic datasets resolution, dimensionality reduction and faster clustering algorithms are central in modern interpretation platforms.

In this work, we explore and present how k-means can be used to quickly obtain classification of seismic volumes in NVIDIA GPUs using the RAPIDS implementation with single and multiple GPUs. k-means shows not only to be fast in single GPUs, but also scales when using multiple GPUs.

### A. k-means Implementations

Sabeti and Javaherian [14] use k-means on multi-attribute 3D seismic data, showing that the classifier can be used to extract useful information about underground beddings and lateral changes in layers. Some works have presented
parallel and efficient implementations of k-means for CPUs and accelerators, even for geophysics purposes. For instance, [10] uses a k-means version implemented with MPI to accelerate the clustering computation with multiple-CPU nodes. [15] improves this approach by presenting their design of a scalable seismic analytics platform built upon Apache Spark and its managing features to process and visualize seismic data using GPU-accelerated nodes. [15] employ their framework to identify geologic faults, but using deep learning approaches.

In 2018, an open-source consortium named RAPIDS was announced. RAPIDS is a suite of data processing and machine learning libraries that enables efficient GPU acceleration. It has a great advantage of providing a user-friendly abstraction to well-known data-science libraries such as Numpy [16] and Pandas [17], as well as exposing high-level GPU parallelism and high-bandwidth memory speed through CUDA ([18]). With RAPIDS, we employed DASK to allow scalable processing with multiple GPUs. As far as we know, this is the first work to explore the performance and scalability of RAPIDS applied to seismic facies classification.

To evaluate the performance of k-means implementation for seismic facies classification, we use a machine learning workflow that is similar to related works [10], [13]. This workflow is detailed in the next section.

### III. MACHINE LEARNING WORKFLOW

In order to perform seismic facies classification, we use a machine learning workflow illustrated in Figure 1. All steps are performed using DASK [19], an open-source and flexible library for Python parallel computing. DASK allows easy creations and extensions of computing clusters.

![Machine learning workflow](image)

Given a SEGY cube as input, we perform the feature extraction (1) using d2geo, an open-source Python library for computing seismic attributes and DASK for parallelization [20]. Features are properties that describe different characteristics of the input objects used for training. For seismic data, we use the attributes designed to enhance the seismic reflection, highlighting different properties of the same dataset. Many attributes are well suited for facies classification since most of them highlight continuities, faults, impedance contrasts, among other important characteristics. [21] lists several attributes that allow accurate predictions of facies, mostly based on non-linear transforms, such as the Hilbert Transformation. We used the following set of attributes as features in our experiments: Amplitude, Cosine Instantaneous Phase, Dominant Frequency, Envelope, Instantaneous Bandwidth, Instantaneous Frequency, Instantaneous Phase, Reflection Intensity, and Second Derivative.

After extracting the features, the values are arranged in a DASK dataframe (2), i.e., a 2D matrix where each line corresponds to a specific point of the seismic cube and each column is the attribute value for that point. For classification purposes, we consider each line of the dataframe as a sample with their respective feature vector (nine column values). Following that, we normalized the dataframe column values using the z-score.

We convert and store (3) the dataframe in a Parquet file [22], an open-source Apache-designed format that stores nested data structures in a flat columnar way. Parquet is designed to handle large amounts of data and to support parallel access, which is desired when dealing with distributed computing.

In the fourth step (classification) we use for comparison two k-means implementations: one provided by DASK-ML library, which allows scalable training using multiple CPUs, and another one provided by the RAPIDS CUML library [23], which also uses DASK to support multi-GPUs training. Both offer similar interfaces to the well-known Python library for machine learning Sci-kit [24]. Further, both of them implements a k-means algorithm with an average time complexity of $O(knt)$, where $k$ is the number of clusters, $n$ is the number of samples and $t$ is the number of iterations.

During the classification step, the input data (in Parquet format) is not loaded into the memory beforehand. Instead, it is loaded on-demand, allowing the training process to overlap data ingestion with other operations. This process is orchestrated by DASK and is very useful when dealing with distributed memory. The results of the clusterization generates the classification (5), which can be used to plot the data for visualization.

The proposed workflow was used in our experiments to test both k-means implementations when classifying seismic data. The experiments’ scope, infrastructure and datasets are described in the next section.

### IV. EXPERIMENTAL SETUP

#### A. 3D Seismic Datasets

For our work, we use three seismic post-stacked open datasets, being them: (a) F3 Netherlands seismic survey, which is a small 3D marine data from offshore of Netherlands; (b) Parihaka, a marine data from New Zealand and; (c) Kahu, also a marine data from New Zealand. The original dataset sizes before and after feature extraction and conversion are shown in Table I.

| Data          | Original Data Size | Feature Extracted Dataframe Size |
|---------------|--------------------|---------------------------------|
| F3 Netherlands| 1.5 GB             | 12.5 GB                         |
| Parihaka      | 3.9 GB             | 30.2 GB                         |
| Kahu          | 6.1 GB             | 66.6 GB                         |

![Seismic surveys and dataframe sizes](image)
B. Hardware and Software Infrastructure

Our experiments were performed on the hardware specified in Table II. Experiments with 2 and 4 GPUs were executed on a DGX Station, 8 GPUs on DGX-1 and 16 GPUs on DGX-2. We used the RAPIDS 0.13 docker container image, which includes all NVIDIA libraries.

| Node           | Processor       | Cores | RAM    | GPUs  |
|----------------|-----------------|-------|--------|-------|
| DGX Station    | Xeon E5-2698 v4 | 20    | 256GB  | 4x Tesla V100 |
| DGX-1          | Xeon E5-2698 v4 | 20    | 512GB  | 8x Tesla V100 |
| DGX-2          | Dual Xeon Platinum 8168 | 24 | 1.5TB  | 16x Tesla V100 |

V. EXPERIMENTAL RESULTS

A. F3 Netherlands Seismic Data

Figure 2 (top) shows a section of F3 Netherlands (inline 100) served as input for the described machine-learning workflow. We executed K-means varying the number of clusters from 5 to 12. The result of the facies classification using 8 clusters can be seen in Figure 2 (bottom), where the colors indicate the associated facies. Based on the features used, it is possible to note that the classification highlights continuous, horizontal and low amplitude reflectors with the green color (facies 4). Very high amplitude reflectors are highlighted with strong purple and yellow colors (facies 2 and 3) while the orange and gray ones (facies 1 and 7) denote continuous oblique areas.

Figure 3 shows training performance using multiple GPUs with RAPIDS implementation. It is worth noticing that the performance improves by 1.88x when we increase the number of GPUs from 2 to 4, but there is a marginal (3%) performance reduction when increasing the number of GPUs from 8 to 16. This occurs due to low GPU occupancy (memory and CUDA cores utilization) when processing small datasets. We also notice that the performance does not significantly change when using a different number of clusters, as the k-means time complexity grows linearly with the number of clusters. In fact, the time complexity of the implementation is \( O(nkt) \), where \( n \) is the number of samples, \( k \) is the number of clusters and \( t \) is the number of iterations.

A speedup comparison with CPU DASK implementation is shown in Figure 4 (blue). The speedup is the geometric mean of all speedups from GPU over CPU using 5 to 12 k-means clusters. The highest speedup was about 186-fold when using 8 GPUs compared to DASK k-means CPU implementation with 40 threads. The overall speedup does not increase when using 16 GPUs.

B. Parihaka Seismic Data

For Parihaka, we also performed training procedures ranging from 5 to 12 clusters. At the time of our experimentation, the default RAPIDS k-means memory footprint was up to 4 times the size of the input data. Thus, as shown in Table II (Parihaka dataframe), we cropped the dataset from inline 0 to 700 summing a total of 30.2GB, so it could fit the available GPU memory. The execution times for training is shown in Figure 3 (bottom). Similarly to F3 Netherlands, the execution time decreases as the number of GPUs increases from 4 to 8 (about 55%). However, the processing time is almost the same when increasing the number of GPUs from 8 to 16, for the...
same reason as the previous dataset. The speedup achieved was about 217-fold, as depicted in Figure 4 (red bars).

Fig. 4. RAPIDS speedup compared to CPU DASK implementation using 40 threads.

C. Kahu Seismic Data

For Kahu seismic data, we performed training with k-means using 16 GPUs, in order to fit the training data to the GPU memory. The speedup over the CPU implementation is presented in Figure 4 (yellow). With a larger dataset we were able to saturate all server resources, achieving a training speedup of 258-fold. Comparing to the speedup of Parihaka which is nearly half of the size of Kahu, we had a speedup of about 18%.

VI. CONCLUSIONS

Seismic facies classification is an important step in seismic interpretation, allowing the visualization of different geological settings. We presented a scalable machine learning workflow to support seismic facies classification over large 3D seismic volumes. We use the NVIDIA RAPIDS library and compared their k-means implementation with a classical multi-core CPU implementation. The results show a speedup up to 258-fold in a DGX-2 server, with 16x GPUs. RAPIDS k-means implementation offer all necessary tools to easily scale training, when GPU resources are available. However, we verified that training in small datasets regimes have a gain upper limit, meaning that adding more GPU devices may not proportionally reduce training time. This pipeline may be adapted to several seismic processing and interpretation tasks, aiming to provide efficient and scalable solutions to different applications.

VII. ACKNOWLEDGMENTS

We would like to thank NVIDIA for the computational resources used in our experiments and Petrobias for research support.

REFERENCES

[1] J. Cabezas, M. Araya-Polo, I. Gelado, N. Navarro, E. Moranco, and J. M. Cela, “High-performance reverse time migration on gpu,” in 2009 International Conference of the Chilean Computer Science Society, 2009, pp. 77–86.

[2] Z. Wang, J. Huang, P. Yong, and Z. Yang, “3d variable-grid full waveform inversion accelerated on gpu,” vol. 2019, no. 1, pp. 1–5, 2019. [Online]. Available: https://www.earthdoc.org/content/papers/10.3997/2214-4609.201901001

[3] G. Liu, Z. Yu, J. Wang, and B. Li, “Accelerating kirchhof pre-stack depth migration on a gpu by overlapping ray tracing and imaging,” Computers & Geosciences, vol. 128, pp. 80 – 86, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098300418307519

[4] H. Knibbe, C. Vluk, and K. Oosterlee, “Reduction of computing time for least-squares migration based on the helmholtz equation by graphics processing units,” Computational Geosciences, vol. 20, no. 2, pp. 297–315, Jan. 2016.

[5] L. Martins, M. A. Goncalves da Silva, M. Arruda, J. Duarte, P. M. Silva, R. Beaucor Seixas, M. Gottass, P. Souza, and J. Panetta, “Accelerating curvature estimate in 3d seismic data using gpgpu,” in 2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing, 2014, pp. 105–111.

[6] I. Jolliffe, Principal Component Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_455

[7] S. Lloyd, “Least squares quantization in PCM,” IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129–137, mar 1982. [Online]. Available: https://doi.org/10.1109/TIT.1982.1056489

[8] T. Kohonen, “Self-organized formation of topologically correct feature maps,” Biological cybernetics, vol. 43, no. 1, pp. 59–69, 1982.

[9] S. Chopra, K. Marfurt, and R. Sharma, “Unsupervised machine learning facies classification in the delaware basin and its comparison with supervised bayesian facies classification,” SEG Technical Program Expanded Abstracts, pp. 2619–2623, 2019.

[10] T. Zhao, V. Jayaram, A. Roy, and K. J. Marfurt, “A comparison of classification techniques for seismic facies recognition,” Interpretation, vol. 3, no. 4, pp. SAE29–SAE58, 2015.

[11] A. Roy, M. Matos, and K. J. Marfurt, “Automatic seismic facies classification with kohonen self organizing maps tutorial,” Geohorizons Journal of Society of Petroleum Geophysicists, vol. 15, pp. 6–14, 2010.

[12] C. Song, Z. Liu, Y. Wang, X. Li, and G. Hu, “Multi-waveform classification for seismic facies analysis,” Computers & Geosciences, vol. 101, pp. 1–9, 2017.

[13] T. Zhao, F. Li, and K. J. Marfurt, “Seismic attribute selection for unsupervised seismic facies analysis using user-guided data-adaptive weights,” Geophysics, vol. 83, no. 2, pp. O31–O44, 2018.

[14] H. Sabeti and A. Javaherian, “Seismic facies analysis based on k-means clustering algorithm using 3d seismic attributes,” in Shiraz 2009-1st EAGE International Petroleum Conference and Exhibition. European Association of Geoscientists & Engineers, 2009, pp. cp–125.

[15] L. Huang, X. Dong, and T. E. Clee, “A scalable deep learning platform for identifying geologic features from seismic attributes,” The Leading Edge, vol. 36, no. 3, pp. 249–256, 2017.

[16] E. Oliphant, A guide to NumPy. Trelgol Publishing USA, 2006, vol. 1.

[17] W. McKinney et al., “Data structures for statistical computing in python,” in Proceedings of the 9th Python in Science Conference, vol. 445. Austin, TX, 2010, pp. 51–56.

[18] S. Cook, CUDA Programming: A Developers Guide to Parallel Computing with GPUs, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[19] Dask Development Team, Dask: Library for dynamic task scheduling, 2016. [Online]. Available: https://dask.org

[20] B. Fitz-Gerald, “D2geo: A framework for computing seismic attributes with python,” 2018, online; available at https://github.com/bfitzgerald3; accessed 14-April-2020.

[21] T. Wrona, I. Pan, R. L. Gawthorpe, and H. Fossen, “Seismic facies analysis using machine learning,” Geophysics, vol. 83, no. 5, pp. O83–O95, 2018.

[22] D. Vohra, “Apache parquet,” in Practical Hadoop Ecosystem. Springer, 2016, pp. 325–335.

[23] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” arXiv preprint arXiv:2002.04803, 2020.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Edouard Duchesnay, “Scikit-learn: Machine learning in python,” Journal of Machine Learning Research, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html