Towards Parameter-Based Profiling for MARE2DEM Performance Modeling

Bruno da Silva Alves(✉), Luciano Paschoal Gaspary, and Lucas Mello Schnorr

Graduate Program in Computer Science (PPGC/UFRGS), Porto Alegre, Brazil
{bsalves,paschoal,schnorr}@inf.ufrgs.br

Abstract. The Controlled Source Electromagnetic (CSEM) combined with seismic surveys has been used to explore new oil and gas reservoirs. The MARE2DEM application generates as mesh that represents a resistivity model of the seafloor underground. From a set of electromagnetic readings, the application runs a data inversion (using Maxwell's equations) along many steps to converge to a resistivity model that more closely matches the measured data. This data inversion procedure is very compute-bound because of the large amount of arithmetic operations involved. As consequence, the MARE2DEM application divides the workload into smaller work grains, called refinement groups due to the usage of Adaptive Mesh Refinement (AMR). These groups are processed independently in a parallel fashion by a set of workers. It is known that parallel processing suffers from delays and resource underutilization if the load remains imbalanced. In this article, we propose an analysis of the performance and imbalance of the MARE2DEM through source code inspection and trace analysis. The novelty of our investigation consists in the usage of runtime parameters to more profoundly understand and characterize the refinement groups' execution time and variability. Our results show that the execution time of the refinement groups is strongly impacted by both the number of processed nodes present on the input mesh and the measured data associated to each refinement group.

1 Introduction

The oil and gas exploration industry has been using marine Controlled Source Electromagnetic (CSEM) methods for the past years. CSEM data provides complementary information to seismic surveys. Besides, CSEM data acquisition is a better cost-effective alternative to new seismic surveys in areas where legacy seismic data is already present [2]. Figure 1 depicts the CSEM data acquisition. Firstly, stationary receivers are positioned on the seafloor above the region of interest. Then, a particular boat towing an electromagnetic wave transmitter (`tx`) navigates over the area. The receivers (`rx`) capture the electromagnetic fields influenced by the materials under the seabed. The boat navigates by following a vertical (from south to north, crossline) or horizontal (from west to east, inline) path that drives the data acquisition. In Fig. 1.A, we depict a crossline (Line-01) and an inline (Line-07) captured from the receivers at different times.

P. Navaux et al. (Eds.): CARLA 2022, CCIS 1660, pp. 63–77, 2022.
https://doi.org/10.1007/978-3-031-23821-5_5

The set of measurement lines completes the final dataset that represents the region of interest. In Fig. 1.B, we represent the Line-01 with a 2D representation as a slice from the 3D space. Obtaining the data is only the first step in the marine CSEM workflow. The key step relies on data inversion, where one can obtain a resistivity model by calculating the data inversion through Maxwell's equations [12]. Those equations describe how an electromagnetic field diffuses through a material depending on its resistivity [12]. And the output model identifies potential reservoirs in the areas where resistivity values are similar to the known oil and gas resistivity values. However, the electromagnetic fields' inversion involves iterative and expensive math operations. In this context, the MARE2DEM [7] application emerged as an open-source project for CSEM data inversion. MARE2DEM is an iterative program that searches for 2.5D resistivity models using adaptive refinement meshes.

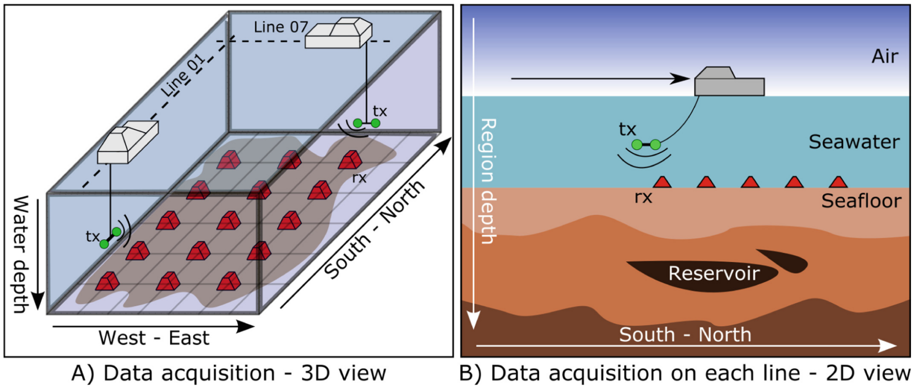

Fig. 1. CSEM data acquisition in 3D and 2D spaces. A boat tows the electric dipole transmitter (antenna in green). Receivers (in red) positioned on the seafloor capture the electromagnetic fields influenced by the materials under the seafloor. Distances and sizes were simplified for visualization. (Color figure online)

MARE2DEM receives as input a) the CSEM data described above, b) the region's geometry, and c) an initial resistivity model. The region's geometry is basically a polygon mesh that identifies the air, seawater, and seafloor areas where the mesh granularity can vary depending on the investigation interests. The initial resistivity model stores a resistivity value for each mesh section. A specialist can help define the initial resistivity values by evaluating the region. However, when resistivities are unknown, one can simply assign the same value for all mesh sections. The application does the CSEM data inversion on each iteration and searches for a better resistivity model than the current one. On this search, MAREDEM calculates an error for each mesh section representing a difference between the calculated electromagnetic fields and the measured data. Then, it identifies the areas with the highest errors, refines them using the adaptive refinement strategy, and calculates new resistivity values for each refined region and surroundings. Furthermore, the application uses parallelization to speed up the processing time. In this way, the code splits the domain problem into small sections called refinement groups, and those are assigned to workers through the coordinator-worker scheme. Each refinement group contains a copy of the whole mesh, and the data collected by the transmitters and receivers present in the group. However, the processing time for each refinement group varies depending on its configuration. The significant difference in processing time among the refinement groups leads to performance issues that make load balancing a challenging task.

Identifying the factors leading to a load imbalance is very challenging since MARE2DEM is a complex parallel application that relies on the Adaptive Mesh Refinement (AMR) technique. Performance visualization has proven its value for load imbalance characterization on the CSEM data inversion context [5] and also in other scenarios [1,6,11]. As consequence, we apply the same performance visualization methodology and tools to characterize the performance of the MARE2DEM application. As far as we are aware, there are no other reported investigation that tackles a performance analysis of the MARE2DEM application that considers the internal operations and their runtime parameters. Thus, an analysis of the MARE2DEM performance is proposed through the evaluation of the execution traces and the source code of the MARE2DEM application. The results show that the processing times of the refinement groups are guided by the number of mesh nodes processed in the refinement mesh and by the amount of input data mapped to each refinement group.

The article is organized as follows. Section 2 presents the dataset and application background. Section 3 details our experimental methodology and context. Section 4 presents our results, including the performance characterization of MARE2DEM's microkernels and of the refinement groups. Finally, Sect. 5 brings a summary of our observations and draw future work.

2 Dataset and Application Background

We present an overview of the MARE2DEM execution workflow, including its CSEM input dataset, the application's steps for data inversion, and the refinement groups used in the domain division.

2.1 CSEM Data

We use the CSEM data from the open-source MR3D (Marlim R3D) dataset available on [3]. The MR3D contains the geoelectric model of the Marlim field present in the Campos Basin, it occupies an area of 257.6 Km^2 in a region off the northern Rio de Janeiro coast in Brazil [10]. This model emerged as a standard for CSEM studies of the particular turbiditic reservoirs on the Brazilian coast [4]. The dataset contains the electromagnetic components of 25 inlines (Weast-East) and 20 crosslines (North-South) captured at six different frequencies that range from 0.125 Hz to 1.25 Hz. We selected one arbitrary line for the following evaluations since the application process one line at a time and we expect the lines to have similar characteristics. Figure 2 depicts the chosen inline referenced as Line `04Tx013a`. We show the model height and length on the Y and X axis. Figure 2.A shows the initial mesh refinement with a heavily refined rectangular area with 64 Km of length and 6 Km of depth. Each inline has a length of 42 Km from the first transmitter to the last one and an additional 11 Km area to the left and right of the first and last transmitters. Figure 2.B shows a smaller area closer to the transmitters and receivers. The blue points identify the 206 transmitters and the red triangles identify the 20 receivers placed in the irregular seafloor. Each transmitter is within 100 m of the next.

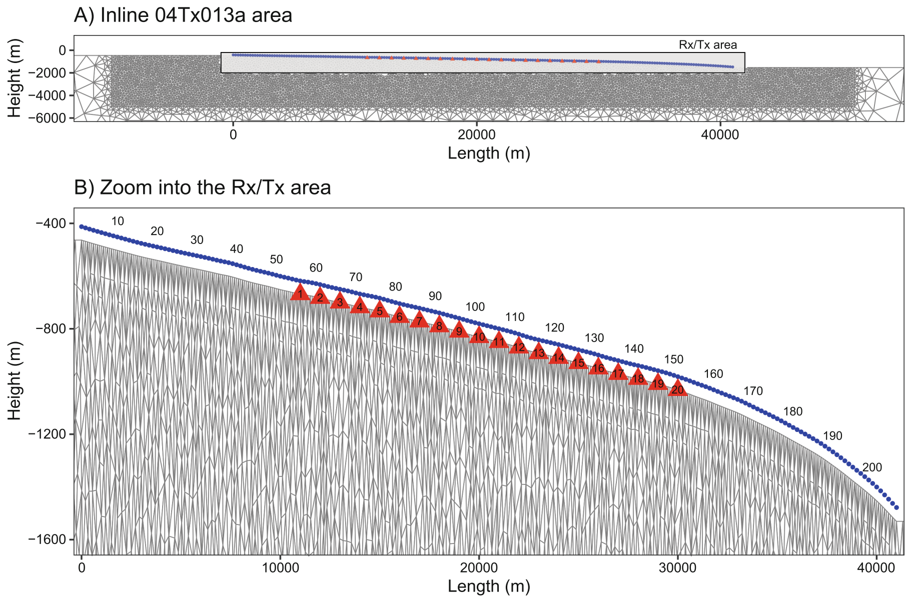

Fig. 2. Line `04Tx013a` of the MR3D dataset: the initial refinement mesh (A), where the polygon sizes reflects the refinement degree; and the rectangular Rx/Tx area (B), where the numbered red triangles represents the 20 receivers, and the blue points represents the 206 transmitters. (Color figure online)

One might expect the data to have a measured electromagnetic field for each transmitter-receiver pairs at each of the used frequencies. However, the selected inline contains less data than the possible combinations of 206 tx, 20 rx, and the 6 frequencies due to the usage of a filter that eliminates non-relevant information. The transmitted fields can reach receivers in three ways: guided fields, reflected fields, and noisy fields. Figure 3 shows the field types captured by the receiver number 10 before applying the filter. We can see that the field classification depends on the distance between the transmitter and the receiver. When the elements are closer, the field directly reaches the receiver without capturing

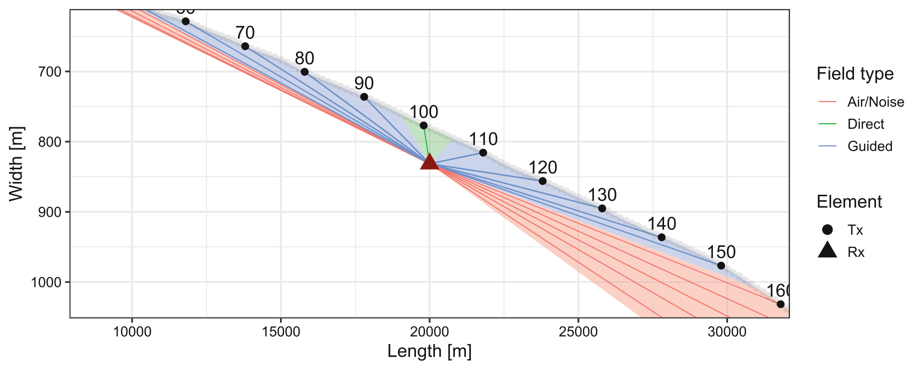

Fig. 3. The CSEM field classification using the data registered by the received number 10 of the MarlimR3D dataset: we depict that the field type (color) depends on the geographical distance between the transmitter and receiver (a Rx-Tx pair, the lines). (Color figure online)

the underground material signature (the green lines on Fig. 3). On the other hand, when the elements are far apart, the reaching field may capture undesired signatures like the ones present in the air (the red lines). The field that we are interested in are the ones influenced by the materials under the seabed (the blue lines). Those are considered guided fields. This last field type occurs when the transmitter and receiver are at a safe distance, not too close or far away. The MarlimR3D dataset contains only guided fields.

2.2 MARE2DEM

The MARE2DEM application is an open-source project[1] for 2D modeling in the context of electromagnetic geophysics. The application generate as output resistivity models for both CSEM (Controlled Source Electromagnetics) and MT (Magnetotelluric) data, but we focus our analysis on the CSEM inversion. The code is mainly written in Fortran, with some features written in MATLAB and C. The MATLAB's source code allows the users to create input models and visualize the results through routines that display a MATLAB graphical interface. The Fortran source code handles the bulk of compute-intensive operations, which mainly include the data inversion operations and the parallel job distribution policy. The use of Adaptive Mesh Refinement (AMR) is MARE2DEM's main feature. This feature removes the responsibility from the user of creating numerically accurate meshes, since it can refine an arbitrary input model's mesh on the regions with the highest possible errors, such as the ones closer to receivers and transmitters. We can also highlight that MARE2DEM can be run in parallel for large CSEM datasets. In what follows, we describe how the application calculates the resistivity models from the input data.

[1] MARE2DEM repository: https://mare2dem.bitbucket.io/.

The MARE2DEM application starts each iteration by calculating the misfit between the CSEM data and the current resistivity mesh. An initial resistivity model defined by the user is assumed as the current resistivity mesh in the first iteration. Then, the application searches for a model with a lower data misfit. The operation is carried out by refining the mesh on the regions with the highest misfit errors and adapting the resistivity according to the inversion process. The described process is repeated until a determined misfit is obtained or when a maximum number of iterations is reached. Further information about the behavior present on the MARE2DEM is available [7,8].

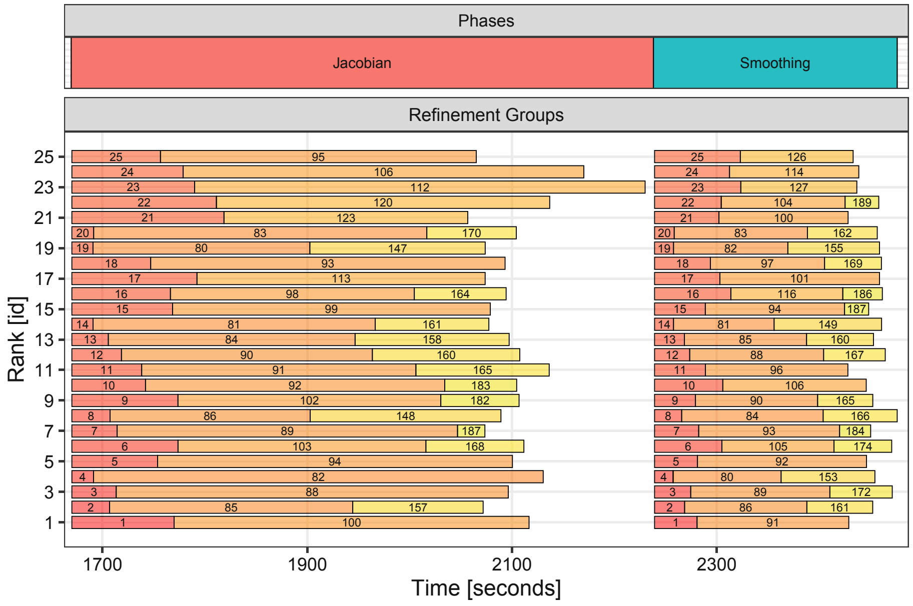

Fig. 4. The two MARE2DEM phases as gathered on its third iteration. The refinement groups processing is also shown for the first 25 workers (rank id) of a total of 80. The X-axis shows the elapsed time since application's start.

We can also explain how MARE2DEM works from the source code perspective by evaluating its function call stack. Figure 4 shows the main regions present on the application. MARE2DEM runs the Jacobian and Smoothing phases for each iteration (we show here the third iteration). The Jacobian phase (in red, left) calculates the model misfit and searches for a μ value that produces a model with a lower data misfit. Then, the Smoothing phase (in blue) seeks for the model with the smallest roughness given the calculated μ. This step stabilizes the inversion and prevents unexpected structures from appearing in the model. The application process the filtered CSEM data on each phase by grouping it into refinement groups. Each refinement group contains the data from a set of transmitters, receivers and frequencies. We show the refinement group's processing on the bottom facet. The Y axis shows parallel worker ranks, the X axis shows the time in seconds, and the labels on the orange rectangles identify each refinement group. Each phase processes all available refinement groups (from a total of 190 in this example), however, we only show the first 25 workers to facilitate the visualization reasons. The slowest rank determines the phase's processing time. The next section describes how the MARE2DEM creates the refinement groups.

2.3 Refinement Groups

The refinement groups composition is an essential step in the application's workflow, as it determines the size of the working grain that will be processed in parallel. The user defines a triplet, prior to execution, with the maximum number of transmitters, receivers, and frequencies that should have each refinement group. We set the application to use refinement groups with a maximum of 6 transmitters, 20 receivers, and 1 frequency. This means that each refinement group has a maximum of 120 transmitter-receiver pairs. However, the groups end up having fewer pairs due to the guided waves filtering of the CSEM data. A previous analysis showed that by setting the receivers and frequencies at 20 and 1, the chosen configuration (with 6 receivers) had the shortest execution time among the possible values for the receivers. Figure 5 shows the amount of Rx-Tx pairs (on the Y axis) for each refinement group (on the X axis). The colors identify the distinct frequencies used. We can see that all central refinement groups (identification ranging from 50 to 150) have a higher number of Rx-Tx pairs, and the peak appears around the refinement group 100. This behavior happens because the number of receivers is more significant around the central transmitters, leading to higher number of pairs in this central area. Another observation is that refinement groups with a higher frequency (1.250 Hz) have less data when compared to a lower frequency (0.125 Hz). The reason is that lower frequencies are capable to penetrate further in the medium, resulting in

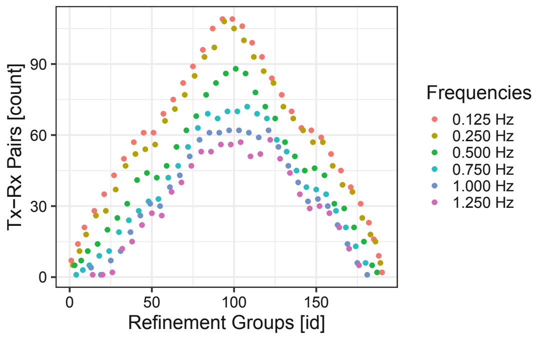

Fig. 5. Number of Rx-Tx pairs present on each of the refinement groups. The colors depict the CSEM frequency associated to each refinement group. (Color figure online)

more number of pairs in the dataset. While this example has been created with a MARE2DEM configuration setup that limits each refinement group to have only one frequency, one can change that parameter so one refinement group can have more frequencies. Such configurations would imply in a more elaborated mapping from data to refinement groups.

3 Methodology and Experimental Context

We adopted a five-step methodology to characterize the MARE2DEM application's behavior. 1) Code inspection to select the code's relevant regions and parameters. 2) Code instrumentation via manual region instrumentation and parameter-based profiling. 3) Parallel MARE2DEM execution for traces acquisition with the MR3D dataset. 4) Conversion from the OTF2 (Open Trace Format Version 2) traces to CSV (Comma-Separated Values) files with the inspected parameters. 5) Trace analysis and behavior characterization. These steps represent a cyclic process where step 1 may follow step 5 in a new analysis cycle. We repeat the cycle until a sufficient number of regions and parameters allow us to explain application behavior. In the following, we detail the software and hardware

In the first step, we started our search for the code's crucial regions by first looking at the wider regions and then refining it by inspecting the functions inside those regions. From the manager's code, we select three relevant regions: the application's main function, the iteration processing and phases processing. The other regions comes from the worker's code inspection: the refinement group processing, the subset-group processing, and the microkernels. The microkernels are the most elementary functions during the refinement group processing, we detail the worker's stack call in the next section. Then, we fully characterize each region by selecting a set of parameters to be collected. The parameters are the values of the variables present at the scope of each region. We select the following parameters for each worker's region: the iteration number, the phase name, the worker identification, the refinement group identification, the subset-group identification, the number of mesh nodes processed by each microkernel, the number of Rx-Tx pairs on each refinement group.

For steps 2 and 3, we use Score-P 7.0 for code instrumentation and trace gathering. Score-P is a software with a set of tools to track performance-related events such as event duration, hardware counters, communication metrics, function stack level, among others [9]. We manually instrument the MARE2DEM source code to capture such performance events for the regions identified in the first step. One should enclose the code's region with Scorep special flags to manually identify the regions of interest. Although Score-P allows to track all functions on the source code automatically or even to filter for a set of functions listed on a particular file, we adopted manual region instrumentation since it can increase control over the region's coverage and reduce instrumentation overhead significantly. This strategy allows one to group functions together or to even select a part of a function to be instrumented. Also, the control over the output trace file size increases. We also set the Score-P to collect the regions' start and end timestamp and the previously mentioned parameters.

In the third step, we use four computing nodes from our local cluster[2] . Each of the nodes is equipped with 2 Intel Xeon E5-2650v3 processors (20 cores, 40 threads) running at 2.3 GHz with 128 GB DDR4 RAM memory. The nodes run the Debian 10 (buster) operating system with the Linux kernel 4.19.0-20-amd64, and they are interconnected with a 1 Gbit/s local network. We configure the MPI (OpenMPI 3.1.4) to use all the available cores on each node, leading to 20 processes per node that were mapped into 79 MARE2DEM workers and 1 manager. For step 4, we use the otf2csv[3] tool to convert the OTF2 format to CSV and improve it to associate the parameters values with each tracked region. For the last step, we build a mean trace as a result of four executions of the application. Our mean trace represents the mean duration for each captured region by using an unique key created from the captured parameters. At least 99% of the regions have a relative standard error less than 5%, therefore our mean trace is considered representative. Furthermore, we conducted the traces analysis using reproducible notebooks filled with experiments annotations and R codes blocks with the Tidyverse package. We made available a reproducible companion[4] of the analysis phase associated with this article.

4 Results

We present the performance characterization by evaluating the MARE2DEM's elementary functions, called microkernels, and then by extending our analysis to wider regions such as iterations and refinement groups.

4.1 Performance Characterization of the Microkernels

The refinement groups represent the workload that each worker receives to compute independently. To compute each refinement group, the application needs to calculate 30 Fourier transformations. Those transformations are grouped in subset-groups of size 5. Figure 6 depicts the processing of the 7th refinement group (red rectangle) for the Jacobian and Smoothing phases. We show the elapsed time since the application's start in the X-axis, the colored rectangles represent the operations, and the rectangle position on the Y-axis represents the operation's stack level (operations on the top call the ones on the bottom). In what follows, we identify the worker's mesh copy as base-mesh. On each subset-group (orange rectangles), the worker refines its base-mesh on the first subset (blue rectangle, `local_refinement`). The following subsets at the same subset-group reuse the mesh previously refined. When moving to the next subset-group, the worker starts again by using the base-mesh. For example, in the first subset-group that contains the subsets from 1 to 5, the worker refines the base-mesh

[2] UFRGS-PCAD cluster: http://gppd-hpc.inf.ufrgs.br/.

[3] otf2csv: https://github.com/schnorr/otf2utils.

[4] Companion: https://github.com/Alves-Bruno/CARLA2022-companion.

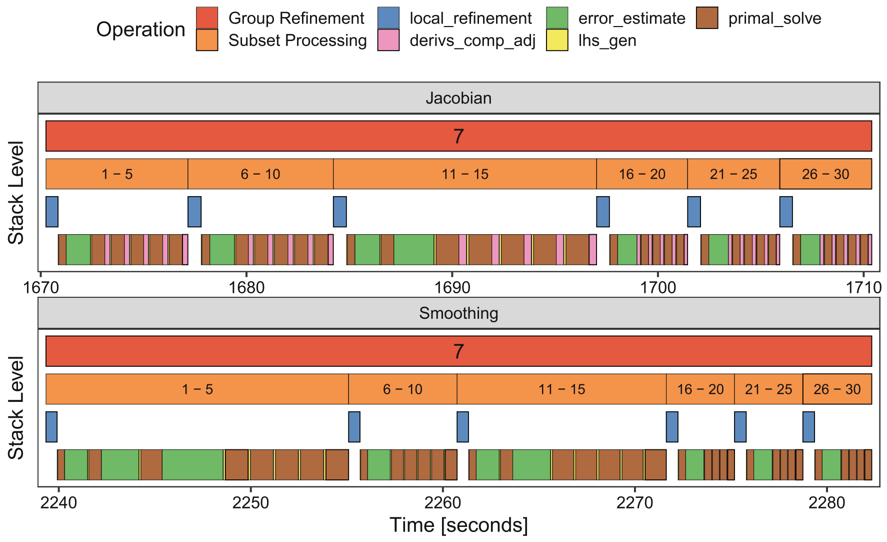

Fig. 6. Stack state of the refinement group processing (number 7) for both Jacobian and Smoothing phases: the orange rectangle shows the subset-group processing; the pink, green, brown, and yellow rectangles represent the microkernels. (Color figure online)

when processing subset 1 and then reuses the resulting refined mesh in the subsets 2 to 5. The operations at the bottom of each facet represent the microkernels (pink, green, yellow, and brown rectangles). They are the most basic operations carried out by MARE2DEM during the refinement group processing. The main functional difference between the phases is that the `derivs_comp_adj` operation is only present at the Jacobian phase.

We compare the relative time of each operation in order to determine the impact of the operations when compared to the total execution time taken by the application. The Table 1 resumes the operations' impact by showing the number of calls (instances), the relative time, and the variability (minimum, mean, and maximum durations) for each basic operation. The application spend more than 48% of its execution time in the `primal_solve`, and this operation has the highest number of instances. Thus the `primal_solve` is the most expensive operation. The `error_estimate` operation follows the `primal_solve` as the second most expensive operation by occupying a significant relative time and by having the higher mean duration. The `derivs_comp_adj` is a particular microkernel, since it is only present at the Jacobian phase. Thus, it leads to a low number of instances that consume a significant amount of (relative) time. Despite its number of instances, the `derivs_comp_adj` is considered expensive as the duration mean is the second highest when compared to the other microkernels. The `lhs_gen` and `local_refinement` represent less than 7% of the execution time, and we exclude the `local_refinement` operation from the following evaluations since this operation has the lowest instance number and its impact on the execution is low.

Table 1. Microkernel's impact on the MARE2DEM performance: the relative time, operation's variability (minimum, mean, and maximum durations), and number of instances called during execution for each microkernel.

Operation	Instances	Relative time	D. min	D. mean	D. max
`local_refinement`	43.3 K	3.14%	0.26 s	0.81 s	1.77 s
`lhs_gen`	304.1 K	3.86%	0.03 s	0.14 s	0.66 s
`derivs_comp_adj`	45.6 K	11.11%	0.03 s	2.72 s	19.01 s
`error_estimate`	105.3 K	33.72%	0.19 s	3.57 s	14.42 s
`primal_solve`	304.1 K	48.16%	0.20 s	1.77 s	8.41 s
Total →	802.4 K	100.00%	–	–	–

Figure 7 shows the duration (Y-axis) for each instance of the microkernels (facets) as a function of the number of processed nodes (X-axis). We can observe that `error_estimate`, `lhs_gen`, and `primal_solve` duration is strongly impacted by the number of processed mesh nodes in both phases. However, the `derivs_comp_adj` differs from the others microkernels because the number of processed mesh nodes cannot fully explain the duration. There are many instances with the same number of processed nodes, but with different durations. To fully understand this particular microkernel, we need to also consider the number of Rx-Tx pairs as an impacting factor. The `derivs_comp_adj` rightmost facet shows the number of Rx-Tx pairs with a binned color scale. Thus, we can conclude that the `derivs_comp_adj` duration is impacted by both the number of processed nodes and Rx-Tx pairs.

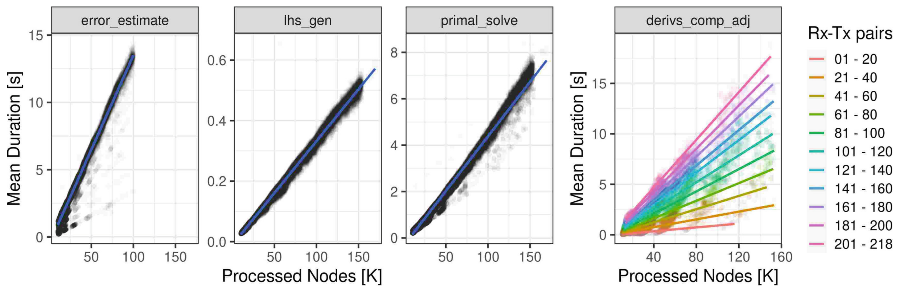

Fig. 7. Microkernels' duration of the Jacobian (all facets) and Smoothing (only the three facets from left-to-right) MARE2DEM phases as a function of the number of processed mesh nodes and Rx-Tx pairs (note the different Y scales).

4.2 Iterations and Refinement Groups

We now consider the previous parameters in evaluating the processing of the iterations and refinement groups. We start by taking a closer look into the iterations durations that is guided by the slowest parallel worker. Figure 8 shows (at the left) the mean duration (Y-axis) of each MARE2DEM's iteration (X-axis). Previous evaluations have revealed that the iterations' duration for both phases increases as the application progresses, and we confirm this for the present configuration. Besides that, we can observe that the Smoothing phase becomes more expensive than the Jacobian when considering the final iterations (iterations 5 to 8). The higher execution times observed at the Smoothing phase happens because the application may repeat the processing of all refinement groups due to convergence conditions controlling the search of the smallest roughness value. We show the number of repetitions needed for the Smoothing phase on each of the iterations (numeric labels on top of each point). Still in Fig. 8 (at the right), we show the duration as a function of the number of processed nodes. The number of processed nodes is a sum of the number of nodes present on the mesh at each microkernel function call on a given iteration. The iteration duration during Smoothing is completely explained by the number of processed nodes. However, this explanation remains insufficient when considering the Jacobian phase. For example, the iterations 3 and 5 (during Jacobian) decrease the duration from the previous iteration despite the increase of processed nodes.

Figuring out the size of the workloads is an important step towards distributing workloads efficiently among parallel workers. In this sense, we consider the duration of each refinement group as the size of the parallel tasks. The Fig. 9 shows the refinement groups' duration pattern (top-row) for the last four iterations, as they are the most expensive ones. We also show the processed

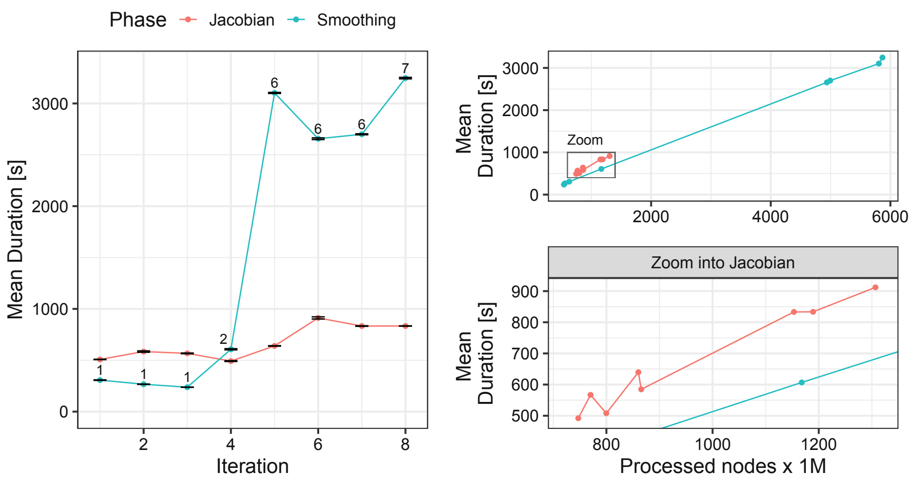

Fig. 8. The duration and number of processed nodes for each iteration and phase.

nodes pattern (bottom-row). The colors represents the 6 CSEM frequencies that reflects the Rx-Tx pairs showed on Fig. 5. The pairs count remains equal on all iterations as they represents the collected data. Despite the differences in scale, the pattern of processed nodes is similar for the two phases, where the central (near to 100) and central-right groups (100 to 160) have higher values than the side groups (0 to 25 and 175 to 206). However, the duration pattern is slightly different in the two phases. During Smoothing, the pattern of the groups follows the same pattern as in the bottom row. During the Jacobian, the pattern of refinement groups reflects a mix between what is shown in the bottom-row and the number of pairs on Fig. 5. Although the duration is strongly driven by the amount of processed nodes, there are differences between the two phases due to the presence of the `derivs_comp_adj` microkernel during the Jacobian.

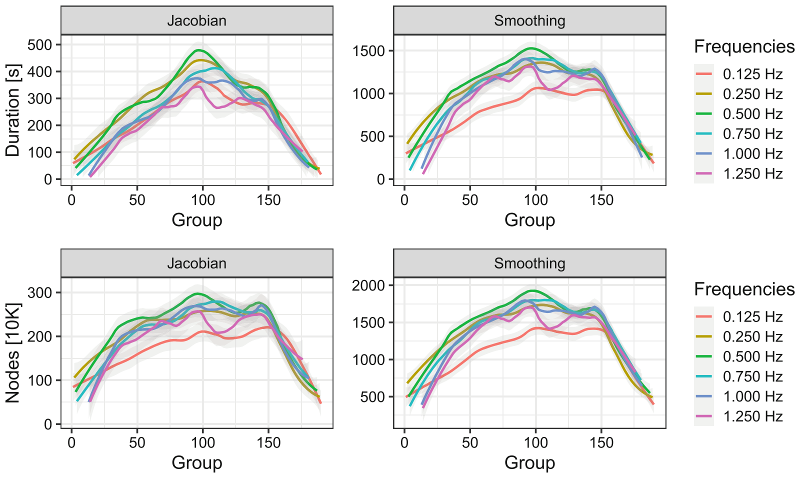

Fig. 9. The signature of each refinement group's duration and processed nodes at the last four iterations. The colours indicate the CSEM frequency of each group. Center and center-right groups are the ones that take longer to process. (Color figure online)

5 Conclusion

MARE2DEM is an open-source application for 2D resistivity modeling on the context of oil and gas exploration using CSEM data. In this work, we evaluated the performance of the MARE2DEM application when running with the Brazilian MR3D dataset. The code inspection, trace evaluation and performance visualization lead us to the following conclusions. The application spends more than 92% of the execution time on the `primal_solve`, `error_estimate` and `derivs_comp_adj` microkernels, hence those functions deserve some optimization efforts in a eventual code review. The imbalance on the parallel tasks, identified as refinement groups in the MARE2DEM terminology, is a consequence of the Adaptive Mesh Refinement algorithm that refines the groups differently depending on the configuration of the refinement group (number and placement of receivers, transmitters, and frequencies). The variability on the refinement groups' duration is mainly explained by the number of processed mesh nodes for both Jacobian and Smoothing phases. We show that the central and central-right refinement groups have longer execution times, while side groups need shorter execution times. In building a more efficient load distribution, one must consider the presented duration patterns for each of the CSEM frequencies used. As future work, we intend to conduct a wider performance analysis considering all possible configurations for the refinement groups definition during the MARE2DEM initialization phase. Our goal would be to estimate the best MARE2DEM's configuration for a given set of compute resources.

Acknowledgements. FAPERGS (16/354-8, 16/488-9), the CNPq (447311/2014-0), the CAPES (Brafitec 182/15, Cofecub 899/18), and Petrobras (2018/00263-5). This study was financed in part by the Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001. Experiments have been executed in INF/UFRGS's High-Performance Computational Resources, http://gppd-hpc.inf.ufrgs.br.

References

1. Cogo Miletto, M., Leandro Nesi, L., Mello Schnorr, L., Legrand, A.: Performance analysis of task-based multi-frontal sparse linear solvers: structure matters. Future Gener. Comput. Syst. **135**, 409–425 (2022). https://doi.org/10.1016/j.future.2022.05.013
2. Cooper, R., MacGregor, L.: CSEM: back from the brink (2020)
3. Correa, J.L., Menezes, P.T.L.: Marlim R3D (MR3D) - the full azimuth CSEM dataset (2018). https://doi.org/10.5281/zenodo.1256787
4. Correa, J.L., Menezes, P.T.L.: Marlim R3D: a realistic model for controlled-source electromagnetic simulations phase 2: the controlled-source electromagnetic data set. Geophysics **84**(5), E293–E299 (2019). https://doi.org/10.1190/geo2018-0452.1
5. Dagostini, J.I., da Silva, H.C.P., Pinto, V.G., Velho, R.M., Gastal, E.S.L., Schnorr, L.M.: Improving workload balance of a marine CSEM inversion application. In: 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 704–713 (2021). https://doi.org/10.1109/IPDPSW52791.2021.00107
6. Isaacs, K.E., et al.: State of the art of performance visualization. In: Borgo, R., Maciejewski, R., Viola, I. (eds.) EuroVis - STARs. The Eurographics Association (2014). https://doi.org/10.2312/eurovisstar.20141177
7. Key, K.: MARE2DEM: a 2-D inversion code for controlled-source electromagnetic and magnetotelluric data. Geophys. J. Int. **207**(1), 571–588 (2016). https://doi.org/10.1093/gji/ggw290
8. Key, K., Ovall, J.: A parallel goal-oriented adaptive finite element method for 2.5-d electromagnetic modelling. Geophys. J. Int. **186**(1), 137–154 (2011). https://doi.org/10.1111/j.1365-246X.2011.05025.x

9. Knüpfer, A., et al.: Score-P: a joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir. In: Brunst, H., Müller, M.S., Nagel, W.E., Resch, M.M. (eds.) Tools for High Performance Computing 2011, pp. 79–91. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31476-6_7
10. Nascimento, T.M., Menezes, P.T.L., Braga, I.L.: High-resolution acoustic impedance inversion to characterize turbidites at Marlim field, Campos basin, Brazil. Interpretation **2**(3), T143–T153 (2014). https://doi.org/10.1190/INT-2013-0137.1
11. Veroneze Solórzano, A.L., Leandro Nesi, L., Mello Schnorr, L.: Using visualization of performance data to investigate load imbalance of a geophysics parallel application. In: Practice and Experience in Advanced Research Computing, pp. 518–521. Association for Computing Machinery, New York (2020). https://doi.org/10.1145/3311790.3400844
12. Zhdanov, M.S.: Geophysical Electromagnetic Theory and Methods. Elsevier, Oxford (2009)