

Investigating Oil and Gas CSEM Application on Vector Architectures

Félix D. P. Michels(✉), Lucas Mello Schnorr, and Philippe O. A. Navaux

Institute of Informatics (INF), Federal University of Rio Grande do Sul (UFRGS), Caixa Postal 15.064 – 91.501-970, Porto Alegre, RS, Brazil
{felix.junior,schnorr,navaux}@inf.ufrgs.br

Abstract. The importance of Controlled Source Electromagnetics (CSEM) has increased in the past decade. Along with this interest, its efficiency increased, data acquisition became easier and costs went down. For the Oil and Gas industry, modeling this data is necessary for exploration. The Modeling with Adaptively Refined Elements for 2D Electromagnetics (MARE2DEM), developed at Columbia University, is one of the tools used to model CSEM data. This paper will evaluate the performance observed during the investigation and implementation of the MARE2DEM's software using the vector architecture NEC SX-Aurora. MARE2DEM is a model for 2D electromagnetic geophysics, making it possible to model the presence of natural gas and petroleum on the depths of the ocean floor. Notably, we will show how the vector machine affects MARE2DEM's performance. It will be explained how the instrumentation of MARE2DEM works and the elaboration of the experiments to perform the investigation. Furthermore, it was necessary to elucidate the modifications done to the code source. The expected result is the runtime in seconds using different parallel decomposition settings. Lastly, we show results of this novel implementation with two workloads, a synthetic input and an input provided by Petrobras (oil & gas company), for which SX-Aurora provides performance improvements up to 27%.

Keywords: MARE2DEM · SX-Aurora · Vector architecture · Geophysics · Performance

1 Introduction

Simulations are necessary and powerful tools used in various fields, from research to commercial and industrial enterprises. The investments of the petrochemical industry in controlled-source electromagnetic (CSEM) has only increased in the

This work has been partially supported by Petrobras (2016/00133-9, 2018/00263-5), CNPq under the project (406182/2021-3) and Green Cloud project (2016/2551-0000 488-9), from FAPERGS and CNPq Brazil, program PRONEX 12/2014. Experiments presented in this paper were carried out using the PCAD infrastructure, http://gppd-hpc.inf.ufrgs.br, at INF/UFRGS.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13380, pp. 650–667, 2022.
https://doi.org/10.1007/978-3-031-10542-5_45

last 15 years, and the academic interest goes along with it [2,3,5]. The main reason for these investments is to reduce risk. CSEM provides more data and, therefore, more information to give new insights on the seafloor bed, consequently reducing risk. The CSEM method works as follows: A electrical-field transmitter is rested close to the seafloor but not at the bottom. Then, electromagnetic receivers are spaced at specific ranges on the seafloor, resulting in the CSEM data in these range intervals. Figure 1 represents the process mentioned above.

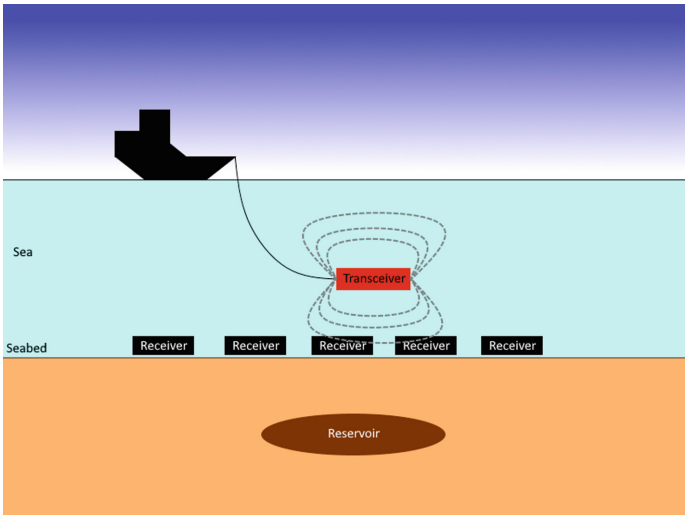

Fig. 1. Schematic view representing the CSEM method.

CSEM data acquisition produces a large quantity of data, leading to a tremendous computational problem for solving inverse modeling. Some regions with complicated geometry may require additional data, mainly to produce full 3D inversion. However, 2D inversion is much quicker and provides a more straightforward interpretation of actual data in a shorter runtime, making it a more robust, sensible, and feasible approach [21].

One necessary implementation to manage this kind of CSEM data is the application "Modeling with Adaptively Refined Elements for 2D Electromagnetics", referred to in this project as MARE2DEM [10]. MARE2DEM is an open-source code for 2D inversion of CSEM data, magnetotelluric (MT) data, and surface-borehole EM data, by parallel adaptive finite elements for onshore, offshore, and downhole environments. Due to the large quantity of data provided by CSEM, high computational power is necessary to execute such data set with efficiency. Therefore, efficient code and a powerful computer are preferred.

High-performance computing plays an essential role in various research fields, commercial sectors, and industries, especially Oil and Gas. It presents many advantages, such as facilitating evaluation to enabling new simulations and modeling [4]. Although the common strategy of increasing the density of chips is

reaching its physical limits, new ways to improve performance arise [1]. One way is by Heterogeneous computing, which mainly consists in using different architectures in unison, like CPUs and GPUs. Inside this particular path, vector engines, or vector architectures, emerge once again.

Vector architectures are Single Instruction Multiple Data (SIMD), which has great potential for highly parallelizable scientific applications. Among these are numerical applications, time prediction, multimedia processing [15], collision simulation [9], data compression, and others. One prominent feature of these vector processors is the possibility of using an instruction to reproduce hundreds of operations. Moreover, all the results of the elements of a vector are independent, and therefore, checking the resulting data is not necessary. Memory access is done only once for each vector, inferring a small memory access latency.

The massive variety of architectures, while desirable and flexible, poses new challenges for the programmer [17]. Increasing the complexity of the structure will additionally extend the problem of imposing the application. Also, there are numerous frequent bottlenecks, such as these located in the memory subsystem, which encompasses cache pollution, *thrashing*, amongst others.

Vectorization can be an excellent tool for inverse modeling, especially for CSEM, in which the enormous quantity of data produces a challenging computational project. Therefore a vector architecture is a promising way to tackle this challenge and improve the performance of MARE2DEM's code, especially architectures with high memory and high memory bandwidth. NEC Corporation launched a new architecture in 2018, a vector processor known as SX-Aurora. This processor has eight processing cores at 1.6 GHz and three stages of cache memory. One of the novelties of this structure over the present ones is the dimension of its vector units, with 256 elements, eight bytes each. In addition, the NEC compiler makes choices automatically, identifying vectorizable areas and generating code for it. Nevertheless, the compiler needs the programmer's assistance to interpret the code and enhance automated vectorization, following specific guidelines. The main object of study is SX-Aurora, an excellent architecture for investigating the parallelization and vectorization of the MARE2DEM's code.

This work will investigate and analyze MARE2DEM's execution utilizing a vector architecture, the SX-Aurora, and the traditional x86 architecture. Two data sets will be used, a synthetic one and a real-world case provided by Petrobras (Brazilian Petroleum Corporation), Brazil's oil & gas corporation. Therefore, the main goals of this paper are:

- Describe the implementation of MARE2DEM on NEC's SX-Aurora. The standard Intel math library was substituted for the MARED2DEM in the SX-Aurora because NEC's architecture does not support it. All entries that required Intel's math library were rewritten, supporting NEC's mathematical libraries.
- To investigate the performance analysis of this implementation by comparing it with x86 architecture CPUs.

The article was organized as follows. Section 2 presents and discusses related works regarding the instrumentation, NEC's architecture, and MARE2DEM.

Section 3 is the methodology, workflow, and execution environment. Section 4 depicts the experimental results of three experiments. Finally, Sect. 5 presents the conclusion and possible future work.

2 Related Work

This section elucidates different scientific articles and manuals from the areas explored in this project. Thus, some studies related to the areas of performance analysis involving the SX-Aurora architecture, in which optimization techniques and vector machines or GPUs, High-Performance Computing (HPC), Score-P [13], and primarily MARE2DEM and CSEM related scientific papers and manual.

Myer, through the use of MARE2DEM, explores 2D inversion of marine CSEM and MT data [18]. The primary purpose is to make the inversion aware that the subsurface comprises similar geologic domains. The presented workflow is subjected to CSEM limitations, making fine-scale structural details hard to be resolved. However, in confounding settings, it is still possible to be used to map the rough qualities and general extent.

It is presented by Grayver a new 3-D parallel inversion scheme for CSEM data in the frequency domain based on a direct forward solver [8]. Gauss-Newton minimization provides data of model inversion, proving the approach's applicability to real-world problems, showing real-world data sets being possible to manage with only medium-sized clusters.

The application MARE2DEM [10] implements 2D forward and inversion modeling algorithms for MT and CSEM data. The inversion modeling uses CSEM information provided by a grid of electromagnetic receivers resting on the seafloor bed and its response to a transmitter below the sea's surface while being towed by a ship. This response happens in known intervals and later is converted into electrical resistivity, making it possible to study the components below the seabed.

Furthermore, MARE2DEM employs a finite element method (FEM) [11] to find the resistivity model. To accurately calculate CSEM and MT models' solutions, Kerry Key presents a parallel goal-oriented adaptive finite element method in his work. He also introduces a reliable goal-oriented error to guide the iterative mesh refinement. The overall performance is assessed utilizing clusters of 800 processors with real-world data sets, achieving execution times of only a few seconds.

Yavich and Zhdanov published their work on improving the finite element modeling through finite difference [23]. This novel implementation used a finite difference solver based on implicit factorization of the matrix. A comparison is made between their implementation and three other public available programs. One of them is MARE2DEM, and it was possible to achieve improved performance with similar accuracy.

Score-P is a unified performance-measurement infrastructure used for profiling, event tracing, and evaluating HPC applications [13]. It provides a variety of different performance evaluation tools, such as Tau [22], Periscope [7],

Scalasca [6], and Vampir [12]. Each one addresses Score-P behavior, utilizing certain factors. Score-P is a complete tool that focuses on bringing the aforementioned tools and being the central piece that binds them together, accommodating all components.

Similar to this work, the publication by Komatsu et al. shows the potential of the SX-Aurora architecture. A comparison with other architectures, NVIDIA Tesla V100 and SX-Ace, among others, shows results where the SX-Aurora can run efficiently, with up to 3.5× performance, in addition to getting a higher speedup of up to 2, 8 × [14].

Yokokawa et al. demonstrate the capability of the SX-Aurora architecture for I/O applications, comparing the I/O system of the new architecture with standard I/O systems. The distinct accelerated I/O function present in the architecture, for some instances, triples the performance in MB per second [24].

Previous work utilizes the same vector architecture, SX-Aurora [16]. Félix discusses in his work the performance analysis of the SX-Aurora architecture utilizing an artificial benchmark and a natural wave propagation application, essential to the oil and prospecting industry. Through simple optimization techniques, such as loop unrolling and inlining, it was possible to achieve performance improvements with the SX-Aurora up to $7,8\times$ with the NAS benchmark and up to $1,9\times$ with the actual application.

3 Methodology

The experimentation workflow follows four significant steps: preparing the environment, running the experiments, collecting data, analyzing. Figure 2 shows the steps, with additional information. The blue-green rectangles in the middle represent the significant steps. The shapes with red color are the three experiments performed. Moreover, the green circles show additional work related to that step.

The execution environment preparation means creating the routines for compilation/execution and data visualization. In this work, three experiments are presented, two of them utilizing an artificial demonstration data set and another one using a real-world data set, provided by Petrobras. The artificial demonstration data set is relatively small compared to the real-world data set, with only 1028 and 17054 data points, respectively. It serves to test the application, providing validation to any modification implemented. Petrobras provided the second data set. The second data set is real-world data collected on the ocean floor utilizing the proper set of equipment. Since it has large amounts of data, it is perfect for testing the SX-Aurora memory bandwidth advantage.

The compiler flags for MPI Fortran (mpif90 v.7.5.0), MPI C (mpicc v.7.5.0), NEC MPI Fortran (mpincc v.2.13) and NEC MPI C (mpinfort v.2.13) utilized were `-O2`, `-fPIC`, `-fpp` and `-cxxlib`. These last two are exclusive to MPI Fortran.

Secondly, the experiments selected in the previous step are carried out in the work environment. As mentioned, there are three experiments.

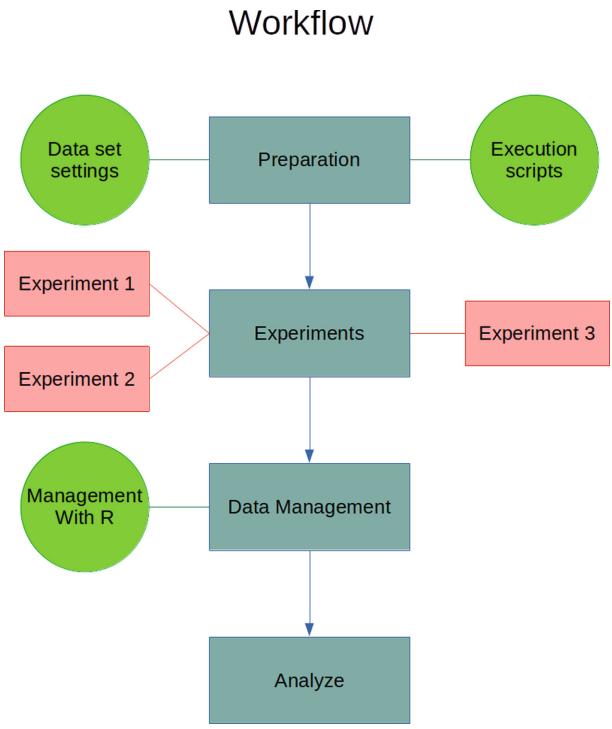

Fig. 2. Workflow of this project major steps. (Color figure online)

The first one uses the artificial demonstration data set. Several workers and data groups need to be chosen to run the application. For the x86 architecture, it was utilized 23 workers and one master corresponding to the 24 cores of the CPU and the default group distribution. NEC's architecture utilized seven workers plus one master and the same default distribution. The second experiment also utilizes the artificial demonstration data set. However, instead of 23 workers for x86 architecture and seven workers for SX-Aurora, only two will be used, which will be better explained in Sect. 4.3. Last, the third experiment uses Petrobra's data set. It utilizes 23 and 7 workers for the x86 architecture and the SX-Aurora architecture, respectively, plus the master for each. Ten executions were performed for all experiments to improve statistical rigor, achieving the mean execution time and error.

The third workflow step is data collection and management, using Score-P and R. The instrumentation of the MARE2DEM application uses Score-P. Our first step was to identify which sections of the source code implement the forward response computation, which contains the Finite Element Modeling (FEM), by the worker processes, considering a refinement group. Since we knew that the forward response parallel computation is implemented using the MPI interface,

we started by searching for MPI calls inside MARE2DEM's source code, mainly implemented in FORTRAN.

Inside the source code, a subroutine is implemented, which contains the worker's logic of which task to perform, according to the message received by the manager process. Inside is the call path that leads to the forward response computation. With this in mind, it was utilized the manual region instrumentation [13] functionality of Score-P to track the timestamps of the call sequence of all workers. This sequence was grouped into a single manual Score-P region and labeled as *compute*. *Compute* is the main MARE2DEM computing phase that will be analyzed since it encapsulates the majority of the parallel MPI computation performed by the workers.

The experiments utilized the resources of the *Parque Computacional de Alto Desempenho* (PCAD) infrastructure, http://gppd-hpc.inf.ufrgs.br, at INF/UFRGS. Figure 3 shows a detailed schematic of the SX-Aurora architecture. The environment has eight-core, global memory, and cache L3, each core with memory cache L1 and L2, one unit of scalar processing (SPU), and a vector processing unit (VPU), with each VPU containing load buffer, store buffer, and 32 vector parallel pipeline (VPP) [19]. The Table 1 shows the detailed specs of the architecture, containing the specs of the processors, cache, and global memories.

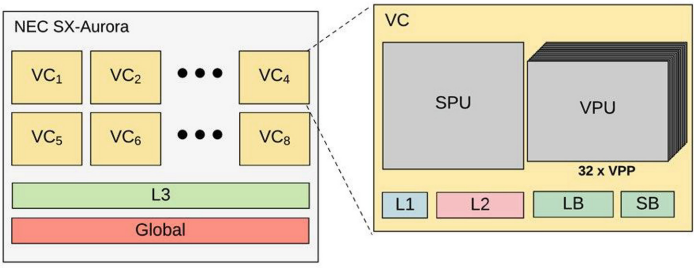

Fig. 3. Detailed scheme of each SX-Aurora core.

The Intel Cascade Lake microarchitecture represents the x86 architecture. In Table 2 you have the specifications of the Intel Xeon Gold 6226 processor, which has 12 cores operating at a frequency between 2.7 GHz and 3.7 GHz. Each core has 32 KB of the L1 cache of data and instructions and a private 1 MB L2 cache. The L3 shared across all cores has a capacity of 16.5 MB, and the machine also features 192 GB of DRAM memory [20].

Table 1. SX-Aurora Architecture.

Vector Engine	Type 10BE
Processor	8 cores @ 1408 MHz
Microarchitecture	SX-Aurora
Cache	8×32 KB L1I 8×32 KB L1D; 8×256 KB L2; 8×2 MB L3
Memory	HBM2 48 GB, 900 MHz

Table 2. Microarchitecture Cascade Lake (x86).

Processor	2×12 cores @ 2700–3700 MHz
Microarchitecture	Cascade Lake
Cache	12×32 KB L1I; 12×32 KB L1D; 12×1 MB L2; 16,5 MB L3
Memory	DDR4 192 GB, 2933 MHz

MARE2DEM implements a math library exclusive to Intel's processors. To execute MARE2DEM in the vector architecture SX-Aurora, this math library needs to be substituted by a general math library or implement NEC's math library, which works specifically to the needs of its vector engine. It was concluded that the latter is more desirable for the present work. It provides better support for parallelization and vectorization, and the nature of the library origin, being the same manufacturer of the vector engine, facilitated the porting process. Besides the math library, there were several other bugs and code rearrangements that needed to be performed.

Curiously, there were some minor discrepancies in the porting process due to different FORTRAN versions. Some syntaxes that are utilized in FORTRAN nowadays are not supported. Also, some code structures are very unreliable to work with. One of them is the bind(C) interface, which makes FORTRAN's subroutines and functions compatible with C code. This section did not work at all, luckly it was possible to rearrange the code to not need it.

One example of a substituted function is for sparse linear algebra equations. The original implementation relied heavily on intel's libraries, which were changed to use mostly NEC's functions. Using NEC's math library, it was possible to use its Sparse Basic Linear Algebra Subprograms (SBLAS) and HeteroSolver routines. The basic structure of this function follows: matrix storage, handle initialization, solution, finalization. The following paragraphs explain these stages.

In this implementation, we utilized Compressed Sparse Row (CSR) format. It consists of representing a matrix through three vectors. So the first step consists in storing the matrix in these three arrays. Nonzero entries are stored in the first vector. The column indices of these nonzero elements are kept in the second array. The third array stores the cumulative sum of the row counts.

Secondly, we set up the handles, which are all provided by the library. We need to call for the function that creates the handles while passing all the necessary parameters, for example, the CSR arrays and column and row counts.

Next, the math library provides the solution through the sblas_execute_mv_rd function. We need to provide the input we already created in the previous steps. Lastly, we need to finalize our code by returning our solution vector and destroying the handle.

In terms of mathematical functions, six needed to be replaced or recreated. There are two functions for factoring a sparse matrix, one for complex numbers and another for real numbers. Both needed to be recreated entirely, as the workflow of both libraries is very different. The same goes for the sparse linear algebra solver, one function each for complex and real numbers. Both were recreated, as described above. Lastly, two functions to free memory for both were replaced.

4 Investigation and Results

The results section will present the main findings obtained by this work. The following result to be shown is regarding the first experiment. It uses the artificial demonstrative data set provided along with MARE2DEM's application. The following result also utilizes the same data set but different worker and refinement group configurations. The last result comes from the third experiment utilizing the real-world data set provided by Petrobras.

4.1 Outcome Validation

After the modifications were done to the math libraries exclusive to Intel, a first experiment was performed. We needed first to be sure that all the modifications had no repercussions on the final result regarding the output of MARE2DEM's execution, which is shown by Fig. 4.

Figure 4 shows the visualization of MARE2DEM final iteration. The graph on the top shows the execution of an x86 architecture. On the bottom graph, the SX-Aurora execution is shown. These graphs provide the resistivity regarding the depth of the ocean floor.

Figure 5 also shows the MARE2DEM's outcome, now related to the final iteration of the real-world data set. The top graph shows the x86 and the bottom one the SX-Aurora implementation.

We can visualize that all executions are visually identical. Therefore we can assume that the modifications implemented, the NEC's math library and all minor changes had no impact on the final result.

4.2 Initial Artificial Demonstration Runtime

After assessing the impact on the final results, we can move forward with the execution's investigation. Next, Fig. 6 shows the execution of the artificial dataset on the x86 architecture.

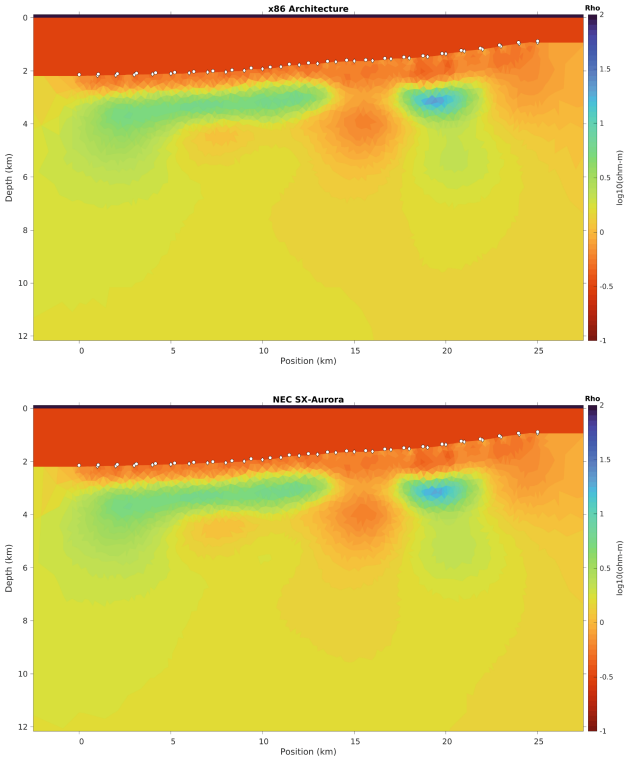

Fig. 4. Demonstration data set comparison for outcome validation between x86 and SX-Aurora.

The x86 architecture execution represented in Fig. 6 shows the runtime, in seconds, for every 23 workers. We can see that the final runtime was 206 s. The image shows the expected behavior of one iteration of MARE2DEM's execution. There is a clear division between left and right on these runtime graphs. They denote two different stages of one iteration, as the manager needs all workers to end their load to start the second stage.

Following the next result, we have Fig. 7. Here we execute the first data set, the artificial one, on NEC's SX-Aurora.

Identical to the previous graph, Fig. 7 gives the runtime for each of the seven workers. The final runtime was 237 s. So, the SX-Aurora performed slower than the x86 architecture, around 15% slower.

Both Fig. 7 and Fig. 6 show the traces collected via Score-P, which can be hard to understand at a glance. To facilitate the comparison between both runtimes, Fig. 8 is presented.

Figure 8 shows on the x-axis the experiment being run, while the y axis represents the runtime of the application in seconds. The light blue refers to the SX-Aurora architecture and the darker blue to the x86 architecture. It is easier now to see the difference in runtimes of both platforms.

Initially, it was expected for the SX-Aurora to improve the performance of MARE2DEM already. However, upon further investigation, especially if the data set was proper, it was attested that the amount of data being transferred was

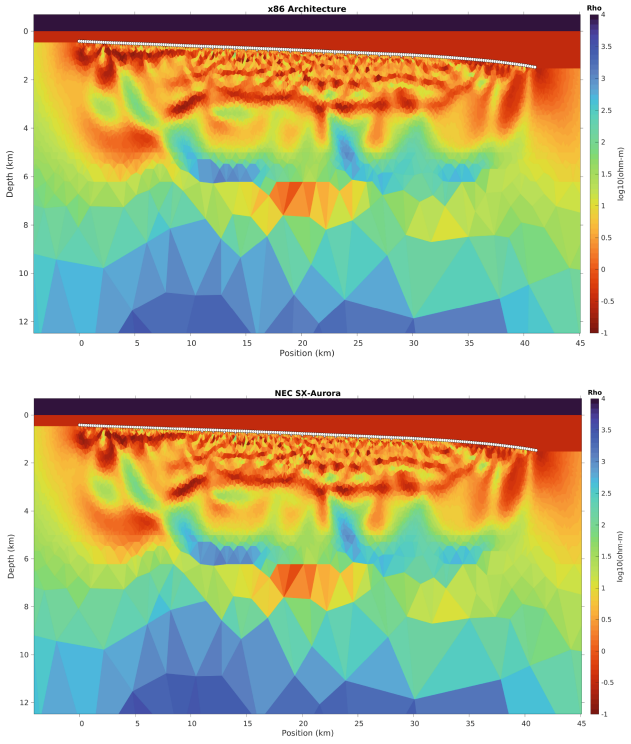

Fig. 5. Comparison for outcome validation between x86 and SX-Aurora, related to the real world data set.

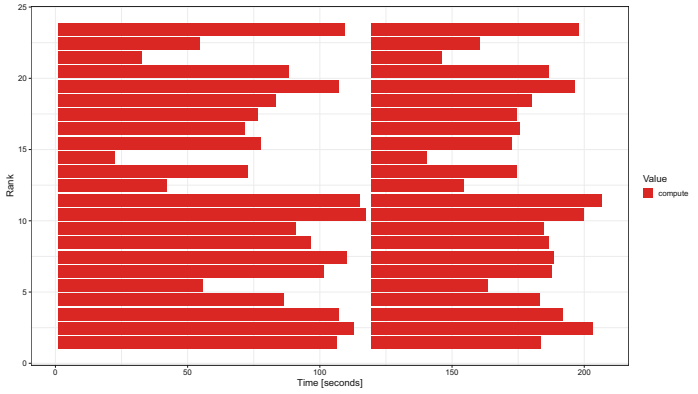

Fig. 6. Results in seconds of experiment 1 utilizing x86 architecture

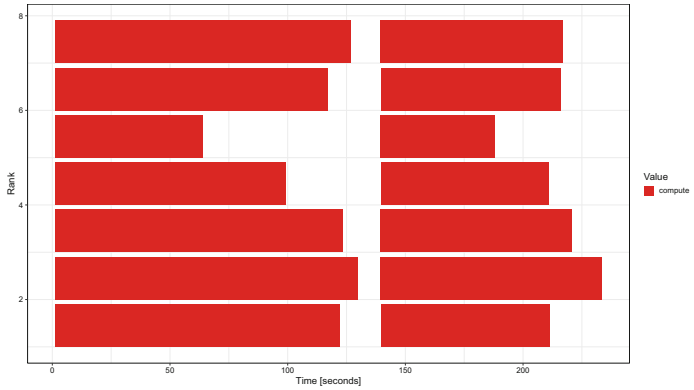

Fig. 7. Results in seconds of experiment 1 utilizing the SX-Aurora

small for each worker, which goes against the advertised advantages of this architecture. SX-Aurora has a very high memory transfer rate, and this power was not being harvested in this data set, resulting in a negative result.

4.3 Effects of One Worker on the First Data Set

After the enlightenment of the last result's investigation, a second experiment was proposed, utilizing the same dataset. A new worker and refinement group configuration was chosen in this second one, so it is possible to appropriate the advantages of the SX-Aurora architecture.

Hence, the configuration to best take advantage of SX-Aurora high memory transfer was to use only one worker, besides the manager, and one refinement group. Theoretically, it would allow for a larger-sized data set and higher data transfers during the MARE2DEM execution. Figure 9 shows these results.

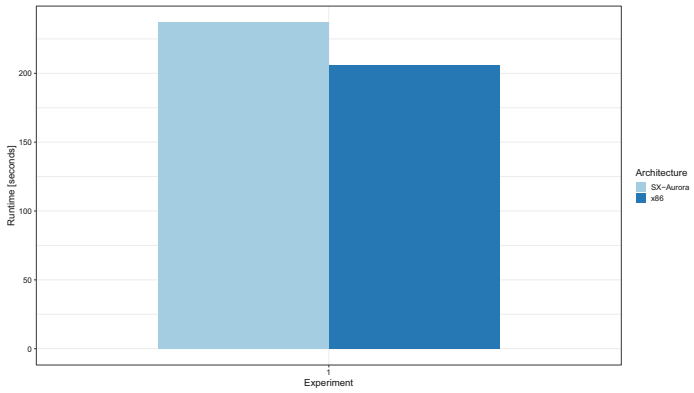

Fig. 8. Experiment 1 comparison between x86 and SX-Aurora runtimes (Color figure online)

In Fig. 9 we have a comparison of MARE2DEM's execution on the x86 and SX-Aurora architectures. The top blue line refers to the x86 execution, and the red bottom line refers to the SX-Aurora. Again, similar to the previous two graphs, on the x-axis is displayed the execution runtime. On the y axis are the two different architectures.

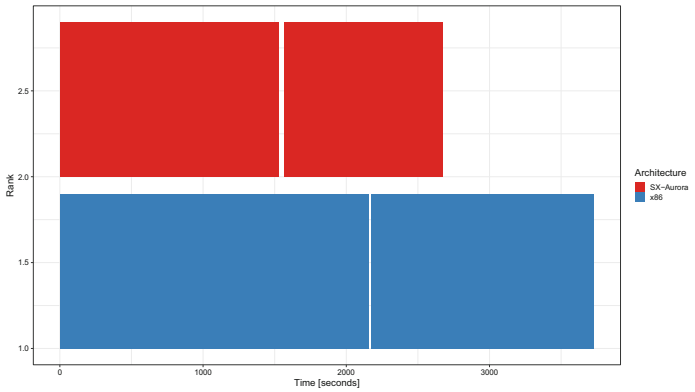

Fig. 9. Experiment 2 comparison of the runtimes between x86 and SX-Aurora. (Color figure online)

It is easier to compare both architectures' runtimes now. We can see that SX-Aurora can improve the runtime without supporting multiples cores and more significant data transfers. The x86 architecture execution runtime was of 3728 s approximately. SX-Aurora runtime reached around 2677 s. An increase in performance of approximately 39%.

This experiment shows that the SX-Aurora architecture has potential regarding performance gains. However, the experiment does not reflect any real-world use case, as both platforms are underutilized, which is evident in the total runtime of both experiments, boasting an increase in execution runtime of around 17 times more significant in the worst case.

4.4 Real World Data Set

As the last experiment was a test to investigate the potential of SX-Aurora architecture, the last experiment will put this potential to test in a real-world scenario. This data set, provided by Petrobras, is larger than the previous one, by roughly 50 times the amount of data.

Figure 10 shows a similar graph as before. We have on the x-axis the runtime in seconds. The y axis shows the workers. Particularly on this graph, we have the presence of the manager, illustrated by the first worker. This graph regards the execution of the real-world data set on the x86 architecture, achieving, for one iteration, an execution runtime of 2964 s approximately.

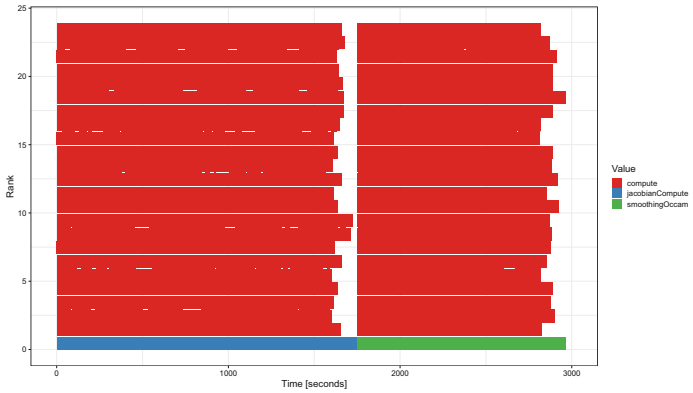

Fig. 10. Runtime for real world dataset - 1 Iteration - x86 architecture.

Next, Fig. 11 has the execution of the MARE2DEM application on the SX-Aurora architecture. As the last graph, the x-axis and y-axis are the runtime in seconds and the workers. The final runtime is around 2341 s for one iteration.

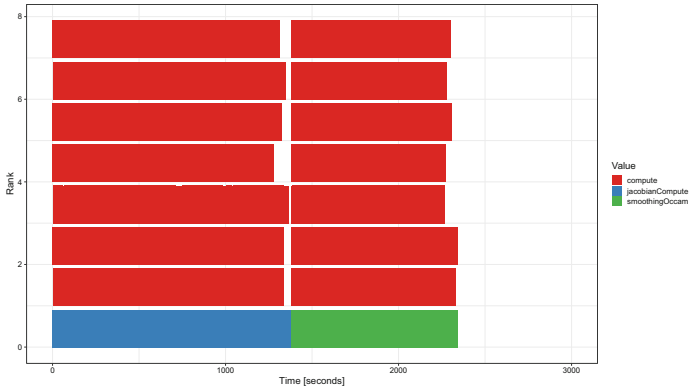

Fig. 11. Runtime for real world dataset - 1 Iteration - SX-Aurora.

To better represent the comparison between runtimes, Fig. 12 is shown. The x-axis represents the experiment, while the y axis shows the application runtime in seconds. The light blue refers to the SX-Aurora architecture and the darker blue to the x86 architecture. Here it is possible to visualize the last statistic, the SX-Aurora has an impact on performance, of around 27% increase.

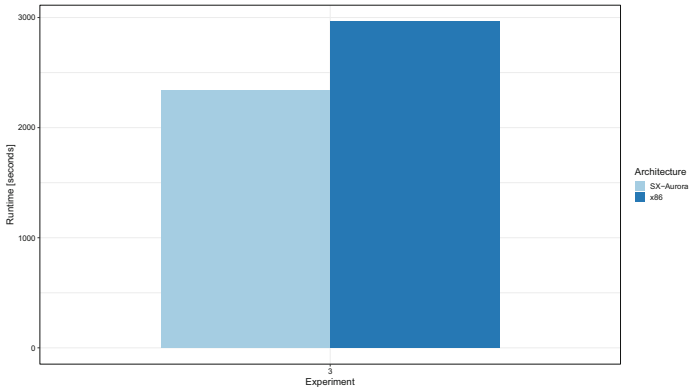

Fig. 12. Comparison of experiment 3 for x86 and SX-Aurora runtimes

Now, the last illustration, Fig. 13, elucidates and realizes some of the potentials that NEC's SX-Aurora has under the right circumstances, especially when we can take advantage of its inherent positive traits, like its vector architecture, which can perform multiple of the same calculation with only one instruction and, in this case specifically, the high memory bandwidth available.

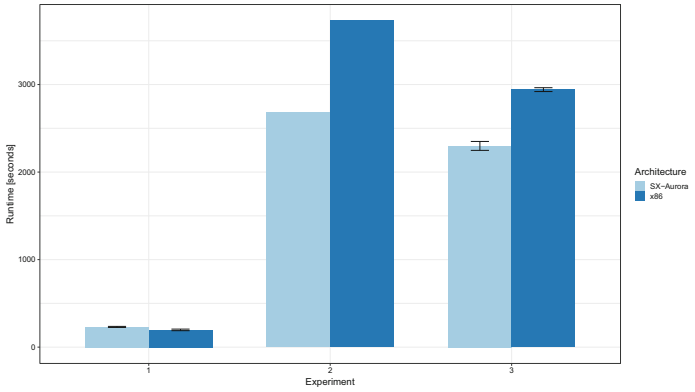

Fig. 13. Runtime comparison between x86 and SX-Aurora for all three experiments.

Figure 13 shows in the y axis the runtime of each experiment. The x-axis displays the number of all experiments that achieved the runtimes exhibited on the y axis. The light blue color gives the SX-Aurora architecture, and the darker blue represents the x86 architecture. To improve the statistical rigor, we employed ten executions to the first and last experiment to have the mean execution time and error. It is visible that the mean execution time is similar to the three experiments performed, approximately maintaining the differences in runtime between platforms.

This graph, Fig. 13 illustrates the path taken by this project. We began the runtime evaluation with worse performance for SX-Aurora. However, after analyzing and evaluating the advantages of the architecture, the two subsequent experiments utilized and realized this potential that SX-Aurora has when harvesting its potential.

5 Conclusion and Future Work

Modeling CSEM data is an arduous process. Large amounts of complex data need to be read, transferred, modified and iterated. It poses a significant challenge for developers and users. Therefore it is crucial to have the right tool for the right job. There are significant efforts in the development of vector processors to act also as performance accelerator devices. In addition to having a high SIMD processing capacity, vector accelerators stand out for allowing applications to benefit from performance gains, with little or no intervention in the application's source code, which is less frequent when accelerating with other heterogeneous architectures.

This work utilizes a SIMD architecture to try and harvest its potential on a CSEM modeling software called MARE2DEM. The chosen architecture is a vector engine titled SX-Aurora. We show that it is possible to utilize its strengths, improving general performance. However, it is necessary to consider the advantages of this kind of architecture and the specific positives the SX-Aurora has.

An extensive experimental load was carried out, which showed evidence of the strengths of vector architectures mentioned. Although there was a need to change the source code, it was mainly due to Intel-specific libraries and some FORTRAN version discrepancies. Intending to investigate the impacts that a vector engine might have, we improved performance, the runtime in seconds, of a real-world CSEM data set from around 2964 s to 2341 s, approximately 27%.

As future work, we will extend this analysis to more real-world data sets, if possible. Implementation of performance optimization techniques such as loop unrolling and inlining, similar to Félix's [16] implementation and expand it further with loop tiling and loop interchange optimizations. These techniques are advantageous in obtaining performance gains in the SX-Aurora TSUBASA vector accelerator [16]. Finally, a more in-depth study of the correlation between group refinement configurations and overall performance is desired.

References

1. Brodtkorb, A.R., Dyken, C., Hagen, T.R., Hjelmervik, J.M., Storaasli, O.O.: State-of-the-art in heterogeneous computing. Sci. Program. **18**(1), 1–33 (2010). https://doi.org/10.1155/2010/540159
2. Constable, S., Srnka, L.J.: An introduction to marine controlled-source electromagnetic methods for hydrocarbon exploration. Geophysics **72**(2), WA3–WA12 (2007). https://doi.org/10.1190/1.2432483
3. Cooper, R., MacGrego, L.: Renewed interest in CSEM in oil and gas exploration. GEOExPro **17**(5) (2020)

4. Ezell, S.J., Atkinson, R.D.: The vital importance of high-performance computing to us competitiveness. Inf. Technol. Innov. Found., 28 April 2016
5. Fanavoll, S., Gabrielsen, P.T., Ellingsrud, S.: CSEM as a tool for better exploration decisions: case studies from the Barents Sea, Norwegian Continental Shelf. Interpretation **2**(3), SH55–SH66 (2014). https://doi.org/10.1190/INT-2013-0171.1
6. Geimer, M., Wolf, F., Wylie, B.J.N., Ábrahám, E., Becker, D., Mohr, B.: The Scalasca performance toolset architecture. Concurr. Comput. Pract. Exper. **22**(6), 702–719 (2010)
7. Gerndt, M., Fürlinger, K., Kereku, E.: Periscope: advanced techniques for performance analysis. In: PARCO (2005)
8. Grayver, A.V., Streich, R., Ritter, O.: Three-dimensional parallel distributed inversion of CSEM data using a direct forward solver. Geophys. J. Int. **193**(3), 1432–1446 (2013). https://doi.org/10.1093/gji/ggt055
9. Hennessy, J.L., Patterson, D.A.: Computer Architecture. Horgan Kaufmann Publishers, Cambridge (2019)
10. Key, K.: Mare2dem: a 2-d inversion code for controlled-source electromagnetic and magnetotelluric data. Geophys. J. Int. **207**(1), 571–588 (2016)
11. 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
12. Knüpfer, A., et al.: The Vampir performance analysis tool-set. In: Parallel Tools Workshop (2008)
13. 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, Berlin (2012). https://doi.org/10.1007/978-3-642-31476-6_7
14. Komatsu, K., et al.: Performance evaluation of a vector supercomputer SX-aurora TSUBASA. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2018, pp. 685–696. IEEE (2018)
15. Kshemkalyani, P.A.: Vector processors (2012). https://www.cs.uic.edu/~ajayk/c566/VectorProcessors.pdf
16. Michels, F., Serpa, M., Carastan-Santos, D., Schnorr, L., Navaux, P.: Otimização de aplicações paralelas em aceleradores vetoriais nec sx-aurora. In: Anais do XXI Simpósio em Sistemas Computacionais de Alto Desempenho, pp. 311–322. SBC, Porto Alegre, RS, Brasil (2020). https://doi.org/10.5753/wscad.2020.14079. https://sol.sbc.org.br/index.php/wscad/article/view/14079
17. Mittal, S., Vetter, J.S.: A survey of CPU-GPU heterogeneous computing techniques. ACM Comput. Surv. (CSUR) **47**(4), 1–35 (2015)
18. Myer, D., Key, K., Constable, S.: Marine CSEM of the Scarborough gas field, part 2: 2D inversion. Geophysics **80**(3), E187–E196 (2015). https://doi.org/10.1190/geo2014-0438.1
19. NEC: SX-Aurora TSUBASA A100-1 series user's guide (2020). https://www.hpc.nec/documents/guide/pdfs/A100-1_series_users_guide.pdf. Accessed Sept 2021
20. Perez, A.F., Ziv, B., Fomenko, E., Meiri, E., Shen, H.: Lower numerical precision deep learning inference and training (2018). https://software.intel.com/content/www/us/en/develop/articles/lower-numerical-precision-deep-learning-inference-and-training.html. Accessed September 2021
21. Price, A., Turpin, P., Erbetta, M., Watts, D., Cairns, G.: 1D, 2D and 3D modeling and inversion of 3D CSEM data offshore West Africa, pp. 639–643 (2008). https://doi.org/10.1190/1.3063732. https://library.seg.org/doi/abs/10.1190/1.3063732

22. Shende, S.S., Malony, A.D.: The TAU parallel performance system. Int. J. High Perform. Comput. Appl. **20**(2), 287–311 (2006). https://doi.org/10.1177/1094342006064482
23. Yavich, N., Zhdanov, M.S.: Finite-element EM modelling on hexahedral grids with an FD solver as a pre-conditioner. Geophys. J. Int. **223**(2), 840–850 (2020)
24. Yokokawa, M., et al.: I/O performance of the SX-Aurora TSUBASA. In: 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 27–35. IEEE (2020)