ASEG-PESA 2015
Geophysics and Geology together for Discovery
24th International Geophysical Conference and Exhibition
15–18 February 2015 Perth, Western Australia

Utilizing massively parallel co-processors in the AarhusInv 1D forward and inverse AEM modelling code

Casper Kirkegaard Kristoffer Andersen Tue Boesen Anders V. Christiansen Esben Auken Gianluca Fiandaca
Aarhus University
C.F. Moellers Allé 4
8000 Aarhus C - DK
casperk@geo.au.dk

SUMMARY

While the forefront of AEM research is focusing on the challenges of 3D modelling, the wide AEM community still rely on less sophisticated computational techniques for their calculations. Inversion of large time domain AEM surveys still prove a computational challenge within a 1D formulation, and require much more computational resources than can be delivered by an office workstation. Emerging Monte-Carlo based 1D Bayesian inversion schemes provide another example of applications that are currently limited by the 1D forward modelling rate.

In this abstract we describe our research in modifying the AarhusInv AEM inversion code to utilize next generation massively parallel co-processors. While our results are early and based on very little optimization, we still achieve comparable levels of performance (>80%) from a single co-processor and a 48 cpu core server. We estimate that performance on the co-processor can be speeded up by approximately another 4x with a limited amount of code restructuring/rewriting.

Key words: High performance computing, AEM, forward modelling, inversion

INTRODUCTION

From an academic point of view most aspects of 1D forward and inverse modelling of AEM data are already well understood. However, some applications are still limited by the time required for the computations. Bayesian inversion schemes based on Monte Carlo methods (Minsley, 2011) require the evaluation of an overwhelming amount of forward models to be applicable on a survey scale, and large time domain AEM surveys are also often inverted using approximate methods for the same reason. Brodie and Sambridge (2006) have shown how very large AEM surveys can be rapidly inverted on cluster computers and Kirkegaard and Auken (2014) demonstrated how large airborne TEM surveys can be inverted over night on a single 48 core computer using the AarhusInv code (Auken et al, 2014).

While high performance codes are available, it is also clear that there can be practical barriers preventing parts of the wide AEM community from running these codes due to practical issues related to the required computer hardware.

Since the computer clock frequency race effectively stopped with the introduction of the first commodity multi-core CPU in 2005, the theoretical performance of processors has continued to increase at the same exponential rate as before but in a fundamentally different way. Performance innovations in modern hardware no longer provide automatic translation into performance for existing codes, since the source of performance is now solely in the form of increased parallelism. The modern scientist has to carefully design his code to not only make use of multiple cores, but also to extract additional parallelism in the form of loop vectorization.

A processor supporting vector instructions has the ability to perform arithmetic operations on multiple elements of a vector at a time, effectively introducing another level of parallelism inside each core. In the simple example of calculating the sum of elements in a vector inside a do loop, a processor with N-width vector units is capable of processing N iterations of the loop at a time. Given recent increases in vector capabilities, vectorization has become an important source of performance that should no longer be ignored.

Until a few years ago all commodity class processors had relatively low core count (2-12) and vector units of width 2. Those characteristics changed dramatically with the introduction of the Intel Xeon Phi co-processor available in 2013. The launch of this product (MIC) marked the beginning of the many-core era with its massively parallel architecture featuring up to 61 cores/244 threads and 8 wide vector units. In terms of raw performance the MIC has a compute capability of more than 1 TFLOPS in double precision, or the equivalent of approximately 5x 8-core processors of high clock frequency. The MIC device is best described as a stand alone computer mounted on a PCI express expansion board with up to 16 GB of local memory. It runs its own Linux micro operating system, is priced from 1500 USD upwards and simply plugs into an expansion slot of a host computer. Programs can execute either directly on the MIC itself, or the host can offload certain demanding parts of a program to the MIC. Host machines with expansion slots for 4+ MICs are readily available, allowing for levels of performance that has previously been reserved for cluster computing.

The MIC differs significantly from general purpose GPUs in the sense that it is based on a standard x86 design and is capable of running any existing Fortran or C++ program with a simple recompile. Where the use of GPUs typically require a complete code rewrite, an existing code can be modified to efficiently utilize the resources of a MIC by exposing parallelism in the data structure and adding parallelizing OpenMP4.0 compiler directives by the most time consuming loops.

In this abstract we describe our experiences with extending the AarhusInv (Auken et al, 2014) inversion code base, to efficiently offload all forward computations to MIC hardware. AarhusInv is severely bottlenecked by forward modelling rate, implying that any performance increase will translate into a virtually identical reduction in inversion time. We start by outlining the numerical problem to be solved before discussing its solution, presenting early results and finally giving our conclusions.

EFFICIENT PARALLEL NUMERICAL SOLUTION OF THE TEM FORWARD PROBLEM

For airborne TEM methods we are typically interested in calculating the z-component of the secondary magnetic field, leading to the type of characteristic integral for the frequency domain H-field seen in equation 1.

$$(1)\quad H_z = \frac{m}{4\pi}\int_0^\infty \left(e^{-u_0(z+h)} + r_{TE}e^{u_0(z-h)}\right)\lambda^3 u_0^{-1} J_0(\lambda\rho)d\lambda$$

$$\lambda = \sqrt{k_x^2 + k_y^2}\,, \qquad \rho = \sqrt{x^2+y^2}\,, u_i = \sqrt{\lambda^2 - \hat{z}_i\hat{y}_i}$$

This equation is valid for the simple case of a vertical magnetic dipole source, but its functional form is representative of the solution for more complicated source types (Ward and Hohmann, 1988) The parameters entering the equation are as follows: k_x and k_y are the horizontal wave numbers, m is the transmitter magnetic moment of a dipole source at height h, $\hat{z}_i$ and $\hat{y}_i$the impedivity and admittivity of the i'th model layer and finally r_{TE} is the reflection coefficient. Equation (2) shows the recursion formula for calculating r_{TE}, iterating over N layers of the model from the bottom upwards, and we note that this entity is by far the most time consuming part of the integrand to compute.

$$(2)\quad r_{TE} = \frac{Y_0 - \hat{Y}_1}{Y_0 + \hat{Y}_1}, Y_i = \frac{u_i}{\hat{z}_i}, \hat{Y}_i = Y_i\frac{\hat{Y}_{i+1} + Y_i\tanh(u_ih_i)}{\hat{Y}_i + Y_{i+1}\tanh(u_ih_i)}, \quad \hat{Y}_N = Y_N$$

The main loops of a numerical solution to the full problem are outlined in Figure 1. Loop 1 iterates over all 1D models of the inversion problem, where NModels is typically significantly larger than the number of cores available. This is where we choose to parallelize the implementation, by distributing the workload of the iterations amongst the available cores. At this outer loop each individual iteration is relatively time consuming, effectively rendering the overhead associated with thread creation and load balancing negligible. Loop 2 iterates over solutions of equation 1 for a sweep of frequencies, which can later be rapidly transformed into a time domain solution for each model using a special case of the fast Hankel transform (Johansen and Sørensen, 1979). Loop 3 then solves equation 1 for each of the frequencies in loop 2, by discretely iterating over the integral limits of equation 1 using a fast Hankel transform. The final loop 4 performs recursive calculation of the reflection coefficient for each discrete evaluation of the integrand, for each frequency and each model.

```
1 DO I=1,NModels
2  DO J=Frequency1, FrequencyN
3   DO K=LowerIntegrationLimit,UpperIntegrationLimit
4    DO L=NModelLayers,1,-1
```

Figure 1. Pseudo-code for the loop hierarchy in a numerical solution of the 1D forward problem.

Loop 4 is where the implementation spends virtually all its time, making vectorization of this particular loop key to high performance. The problem is that the iterations of this loop are recursive, ie. each iteration depends on the result of the previous iteration, making vectorization impossible without code restructuring. In order to obtain early performance results for this abstract we chose the simplest way to vectorize the inner loop, namely by calculating the very time consuming evaluations of the tanh function in equation 2 in a separate vectorizable loop. This loop has no recursive dependency and the simple rewrite increased the performance of the full implementation by almost an order of magnitude.

In Figure 2 we compare the parallel scaling of the early implementation with the CPU performance results found in Kirkegaard and Auken (2014). The first thing to note from this figure is how a single MIC co-processor (224 threads) delivers more than 80% of the performance of the significantly more expensive 48 core server. We also note that stacking two MICs in a single host machine effectively doubles performance, thus significantly outperforming the cpu based solution.

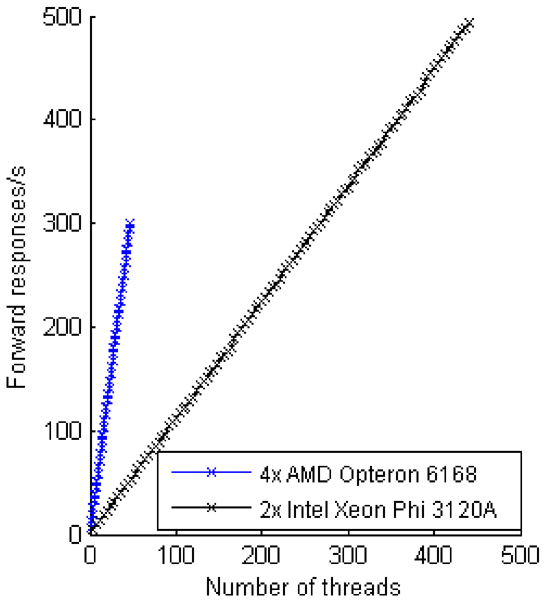

Figure 2. Parallel forward modelling rate for models of 19 layers. Comparison includes a 48 CPU core server system and a workstation like system with dual Intel Xeon Phi co-processors.

Most importantly we note that we have obtained these results by doing very little to exploit the wide vector units of the MIC. Vectorization is most efficient when the number of loop iterations is large and micro benchmarking reveal the potential for another ~4x speedup, provided the length of the inner loop can be increased substantially. The number of model layers determine the length of loop 4, typically in the order of 5-30, making loop 4 by far the shortest of the loop hierarchy for any large scale time domain survey. We thus propose to extract significantly more performance from the implementation by interchanging loop 3 and 4 of Figure 1. This amounts to calculating the reflection coefficients for all integrand arguments before performing the actual Hankel transform. Such an approach brings the added benefit of resolving the recursive dependence in the inner most loop, rendering all parts of the reflection coefficient calculation vectorizable. In our particular code an interchange of loop 3 and 4 requires little more than approximately 300-400 lines of rewrite, however, the data logistics and indexing is somewhat involved and we were unable to complete it in time for abstract deadline. This will be demonstrated at the presention.

CONCLUSIONS

We have outlined the state of the art in 1D forward based high performance AEM inversion schemes and introduced the most important properties of the next generation massively parallel x86 compute device: Intel Xeon Phi.

The AarhusInv AEM inversion code is an excellent candidate for offloading its computations into the new technology, since it is bottlenecked by forward modelling rate and is already fully OpenMP parallelized. Apart from having to work around certain programmatic limitations and a number of early compiler issues, implementing the offload was mostly a matter of distributing data to/from the available devices. Our initial run of the offload code on a Xeon Phi had very poor performance, since the codebase had never been optimized to exploit vectorization. Improving on this was relatively easy, however. By performing a simple 30 line rewrite of the most important loop we had the calculation of the reflection co-efficient partially vectorized, thus increasing overall performance by approximately an order of magnitude. With this early implementation a single Intel Xeon Phi co-processor delivers more than 80% of the forward modelling rate of a 48 cpu core server. Performance scales linearly with the number of available devices, thus allowing for either unprecedented performance in a single device workstation or cluster class performance in a 4+ configuration.

While our early performance numbers might sound impressive, there is significantly more performance to be gained by pursuing further optimization in terms of vectorization. We describe a way of increasing the length of the inner-most vectorizable loop that we estimate will bring an approximate performance increase of another 4x, and we expect to be able to show such results for the conference. Operating at such significantly elevated level of performance opens up new possibilities in several applications: rapid inversion of AEM data in the field, inversion of very large AEM datasets on office workstations and Bayesian inversion of significantly larger datasets than currently possible.

ACKNOWLEDGMENTS

We thank the The Danish National Advanced Technology Foundation for funding the Airtech4Water project and the Willum Kahn Rasmussen Foundation for funding the HOBE project in which this research was conducted.

REFERENCES

Auken E., Christiansen A.V., Kirkegaard C., Fiandaca G., Schamper C., Behroozmand A.A., Binley A., Nielsen E., Effersø F., Christensen N.B., Sørensen K.I., Foged N., and Vignoli G. 2014. An overview of a highly versatile forward and stable inverse algorithm for airborne, ground-based and borehole electromagnetic and electric data. Exploration Geophysics 1-13, DOI: 10.1071/EG13097.

Brodie R. and Sambridge M. 2006. A holistic approach to inversion of frequency-domain airborne EM data. Geophysics 71, G301-G312.

Johansen H.K. and Sørensen K.I. 1979. Fast Hankel transforms. Geophysical Prospecting 27, 876-901.

Kirkegaard C. and Auken E. 2014. A parallel, scalable and memory efficient inversion code for very large scale airborne EM surveys. Geophysical Prospecting, Accepted.

Minsley B.J. 2011. A trans-dimensional Bayesian Markov chain Monte Carlo algorithm for model assessment using frequency-domain electromagnetic data. Geophysical Journal International 187,252-272

Ward S.H. and Hohmann G.W. 1988. Electromagnetic theory for geophysical applications. In: Electromagnetic methods in applied geophysics, Vol. 1 (ed. M.N. Nabighian), pp. 131-311. Society of exploration geophysicists (SEG).