Computers & Geosciences 62 (2014) 80–87

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Massively parallel regularized 3D inversion of potential fields on CPUs and GPUs

Martin Čuma [a,b,c,*], Michael S. Zhdanov [b,c]

[a] Center for High Performance Computing, University of Utah, 155 S 1452 E Rm 405, Salt Lake City, UT 84112, USA
[b] Department of Geology and Geophysics, University of Utah, 155 S 1452 E Rm 405, Salt Lake City, UT 84112, USA
[c] TechnoImaging, 4001 South, 700 East, Suite 500, Salt Lake City, UT 84107, USA

ARTICLE INFO

Article history:
Received 4 May 2013
Received in revised form
7 October 2013
Accepted 8 October 2013
Available online 16 October 2013

Keywords:
3D gravity inversion
3D magnetics inversion
Parallel computing
OpenACC

ABSTRACT

We have recently introduced a massively parallel regularized 3D inversion of potential fields data. This program takes as an input gravity or magnetic vector, tensor and Total Magnetic Intensity (TMI) measurements and produces 3D volume of density, susceptibility, or three dimensional magnetization vector, the latest also including magnetic remanence information. The code uses combined MPI and OpenMP approach that maps well onto current multiprocessor multicore clusters and exhibits nearly linear strong and weak parallel scaling. It has been used to invert regional to continental size data sets with up to billion cells of the 3D Earth's volume on large clusters for interpretation of large airborne gravity and magnetics surveys. In this paper we explain the features that made this massive parallelization feasible and extend the code to add GPU support in the form of the OpenACC directives. This implementation resulted in up to a 22x speedup as compared to the scalar multithreaded implementation on a 12 core Intel CPU based computer node. Furthermore, we also introduce a mixed single–double precision approach, which allows us to perform most of the calculation at a single floating point number precision while keeping the result as precise as if the double precision had been used. This approach provides an additional 40% speedup on the GPUs, as compared to the pure double precision implementation. It also has about half of the memory footprint of the fully double precision version.

1. Introduction

Potential fields methods have become an important tool for exploration on the large scale thanks to the emergence of reliable airborne data collection platforms. These instruments are capable of covering thousands of square kilometers with thousands of line kilometers and data collected as densely as every few meters (Lee, 2001; Dransfield and Zeng, 2009). Interpretation of the data acquired by these surveys has been limited by the huge amount of processing required.

For large surveys, structural interpretations are usually based on a choice or combination of Euler deconvolution (e.g., Thompson, 1982), eigenvector analysis (e.g., Beiki and Pedersen, 2010), wavelet analysis (e.g., Hornby et al., 2002), analytic signal (e.g., Salem and Ravat, 2003; Beiki, 2010), or depth-from-extreme-points (DEXP) methods (e.g., Fedi, 2007). While such methods may provide information about the sources of the potential field, it is not immediately obvious how that information can be quantified in terms of the physical properties in a 3D earth model. For this reason, inversion marks an important step in quantitative interpretation—particularly at deposit scales.

Generalized inversion methods first discretize the 3D earth models into cells of constant density, susceptibility or a variant of the magnetization vector. As gravity and magnetic inversion is non-unique, regularization must be introduced to recover the most geologically plausible solutions from the infinite number of mathematically equivalent solutions. Regularization effectively selects the class of models from which a unique solution is sought. Over the years, a variety of methods have been developed for 3D inversion of potential field data with both smooth (e.g., Li and Oldenburg, 1996, 1998; Li, 2001) and focusing (e.g., Portniaguine and Zhdanov, 1999; Zhdanov, 2002, 2009; Kirkendall et al., 2007) regularization.

The computational resources needed for deposit-scale discretization (e.g., cells smaller than 25 m^3) for entire airborne surveys easily exceed the capacity of high-end desktop computers. Although considerable improvements have been made with the information reduction approaches (Portniaguine and Zhdanov, 2002; Li and Oldenburg, 2003; Davis and Li, 2011), the practical limit of inverting large scale regional surveys with sufficient resolution remains. In practice, large airborne surveys are usually

* Corresponding author at: Center for High Performance Computing, University of Utah, 155 S 1452 E Rm 405, Salt Lake City, UT 84112, USA. Tel.: +1 8016523836.
E-mail address: m.cuma@utah.edu (M. Čuma).

0098-3004/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cageo.2013.10.004

divided into subsets inverted separately and resulting 3D earth models stitched or tiled together post-inversion (e.g., Phillips et al., 2010; Yang and Oldenburg, 2012).

Over the last decade, a large part of the computer speedup has been achieved by hardware parallelization, incorporating more and more processor cores into the CPU. Reports of parallel potential field inversion, however, have been relatively limited. Zhang et al. (2004) used genetic algorithm for 3D inversion of gravity data, dividing the 3D volume into 20 parallel subregions, each of which inverted gravity data up to 10 km outside of the subregion. Tondi et al. (2012) report on parallel inversion of gravity and seismic data using the ScaLAPACK library (Choi et al., 1996). While using a large number of processors, their model size is still limited by the amount of memory required to store the dense gravity response matrix.

We have recently introduced a general method of solving truly large-scale potential field inverse problems with massive parallelization where the modeling domain is discretized into hundreds of millions (even billions) of cells (Čuma et al., 2012). For such models, computation of the predicted data and direction of steepest descent at every iteration of inversion is not trivial. We tackle this problem on several fronts detailed in this paper. Our inversion method is designed to invert jointly any component of the gravity or magnetic fields, including the total, vector, and gradient components. The inversion output is a density distribution for gravity and susceptibility or magnetization vector components for magnetics.

Within the last decade, graphical processors (GPUs) have become attractive tools in high performance computing (HPC), due to their massively parallel design which allows for high computation throughput. However, their hardware design is considerably different from traditional CPUs and consequently programming on GPUs has several challenges. Firstly, since they are very high throughput, they demand algorithms with a lot of concurrent data processing. Second, since the GPU is a separate device that does not share the data with the main memory, data needs to be copied to the GPU before processing which can be a bottleneck if the data is large. Finally and importantly for non-experts, programming approaches for the GPUs (such as CUDA and OpenCL) are relatively of low level and require significant efforts both in learning and in coding. In the potential fields arena, Moorkamp et al. (2010) and Chen et al. (2012) use CUDA to implement gravity and gravity gradiometry forward modeling on a single GPU.

OpenACC is a new standard, the goal of which is to simplify programming on the accelerators and improve code portability across different systems and accelerators. It is a directive based approach, similar to OpenMP for multithreading, in which the programmer adds directives to the code (most often to loops) that tell the compiler to offload the computation within this directive to the accelerator (not necessarily a GPU, any other accelerators, such as the Intel Xeon Phi, can be supported). The compiler takes care of generating the code that will run on the accelerator. Thanks to the simplicity of the OpenACC and the portability promise, it has a potential to become a widely accepted approach for programming accelerators, similarly to what OpenMP has become for multicore processors. For these reasons, we chose OpenACC to port our massively parallel potential fields inversion code to the GPU accelerators. We should also note that there are efforts to extend OpenMP to support accelerators, or merge OpenACC into OpenMP, although at the time of this writing there was only limited accelerator support in the last OpenMP standard and no compiler implementation.

This paper is organized as follows. In the next section we briefly describe the theory behind the potential fields inversion. We then detail the approach we took in our massively parallel inversion algorithm that we published in Čuma et al. (2012), followed by a description of the OpenACC implementation and the mixed single–double precision approach that provides further speedup without compromising the quality of the result. We then validate the results obtained with the GPUs and with the mixed single–double precision approach on a real survey dataset. Finally we present some new results of the inversion of total magnetic intensity (TMI) data for magnetization vector over a very large area on only a handful of GPU equipped cluster nodes, a calculation which would require hundreds of nodes if done on CPUs only.

2. 3D potential fields modeling and inversion

Below we present a brief summary of our modeling and inversion methodology and refer the interested reader to Čuma et al. (2012) for details. Gravity (U_g) and magnetic (U_H) potentials are linear with respect to the 3D density (ρ) and magnetic susceptibility (χ). When calculating the gravity or magnetic response, the modeled domain is usually divided into rectangular prisms with constant ρ or χ. Volume integrals over the potentials' Greens functions that express this response can be evaluated analytically or numerically. In our modeling, we use numerical solution using single-point Gaussian integration with pulse basis functions. The potentials and their first and second spatial derivatives are then expressed through kernels that are numerically evaluated for each cell–receiver pair.

Since the potentials are linear, the response can be written in discrete forms as

$$\mathbf{d}_g = \mathbf{A}_g \rho \tag{1}$$

$$\mathbf{d}_H = \mathbf{A}_H \chi \tag{2}$$

where $\mathbf{d}_g$ and $\mathbf{d}_H$ are Nd length vectors of the observed gravity and magnetic data, $\mathbf{A}_g$ and $\mathbf{A}_H$ are the $Nd \times Nm$ gravity and magnetic operators, numerically expressed as the sensitivity matrix, and ρ and χ are the Nm length vectors of gravity or magnetic susceptibility.

In the inversion, we use the measured response data $\mathbf{d}_g$, $\mathbf{d}_H$ to recover the unknown density ρ or susceptibility χ. The inversion process is non-unique so regularization has to be introduced, with the goal to recover the most geologically plausible solution from the infinite set of mathematically equivalent solutions. We minimize the Tikhonov parametric functional $\mathbf{P}(\mathbf{m})$

$$\mathbf{P}^{\alpha}(\mathbf{m}) = \varphi(\mathbf{m}) + \alpha s(\mathbf{m}) - > \min \tag{3}$$

where $\varphi(\mathbf{m})$ is the misfit functional of the measured and inversion predicted data, $s(\mathbf{m})$ is the stabilizing functional and α is the regularization parameter that balances the misfit and the stabilizing functional. We use adaptive regularization (Zhdanov, 2002), which decreases the stabilizer contribution as the inversion gets closer to the converged result. Model weights are also introduced based on the integrated sensitivity which ensures equal sensitivity of the domain cells located at different depths and horizontal positions from the measured data points.

The minimization of the parametric functional $\mathbf{P}^{\alpha}(\mathbf{m})$ is based on the re-weighted regularized conjugate gradient method (Zhdanov, 2002) by iteratively updating the model parameters $\mathbf{m}$ to minimize the vector of data residual errors $\mathbf{r}$, such as

$$\mathbf{m}_{i+1} = \mathbf{m}_i + k_i \mathbf{A}^T{}_{g,H} \mathbf{r}_i \tag{4}$$

where k_i is the step length and $\mathbf{A}^T_{g,H}$ is the conjugate transpose of the $Nd \times Nm$ matrix of the gravity or magnetic linear operators and i is the inversion iteration index. The inversion proceeds until the residual $\mathbf{r}$ reaches a given threshold or a maximum number of iterations is reached.

The use of a stabilizing functional is the core of the regularization approach. There are several choices for the stabilizing functional. Smooth stabilizers, which tend to produce smooth anomalies, seek to minimize a function of difference between the current model and the a priori model. Focusing stabilizers aim to produce sharper boundaries and contrasts, and are often based on minimization of volume of nonzero departures or nonzero gradients from the a priori model (Zhdanov, 2002). We implement a wide choice of smooth and focusing stabilizers to allow the interpreter the flexibility of choosing one over another for a given interpretation scenario.

3. Implementation

The most important difference from other 3D potential fields inversion implementations is that we compute the sensitivity matrices $\mathbf{A_{g,H}}$ on demand rather than saving and reusing them in each inversion iteration. For the size of problems we aim to solve, this matrix, which is of size of number of data *Nd* times number of 3D volume cells *Nm*, is too large to keep in memory even if various information reduction techniques are used. This is especially true for the accelerators which have limited memory capacity and high cost of data transfer between the host and the accelerator. Storing the sensitivity matrix in disk files is also time prohibitive especially when hundreds of processors need to read the sensitivity matrix at the same time.

In these situations, it is faster to recalculate the sensitivities whenever they are needed. In our approach, each sensitivity matrix point has to be calculated three times during every inversion step, at calculation of the steepest descent vector **lw**, the minimization step length *k* using the line search method, and the predicted data **dp**. Product of each sensitivity matrix value with a given property, for example, the residual **r** in the case of steepest descent vector calculation, is then accumulated into the variable of interest, e.g. cell *i* of **lw**, as

$$\mathbf{lw}_i = \sum_{j=1}^{Nd} (\mathbf{A}_{ij}\mathbf{r}_j) \tag{5}$$

That way we only store arrays of the size of *Nm* (steepest ascent vector, model parameters, etc.) or of *Nd* (predicted and observed data), which results in many orders of magnitude reduction in memory needs.

The on demand sensitivity calculation dramatically increases the computational needs. We employ numerous strategies to mitigate this.

First, for calculation of the sensitivity kernels, we use single-point Gaussian integration with pulse basis functions which is much faster than an exact analytic solution to the volume integrals and has been shown to be as accurate provided the vertical distance to the center of the cell is at least twice as large as the dimension of the cell (Zhdanov, 2009). With airborne surveys typically flying at 50–100 m above the surface, the vertical cell size for the uppermost inversion domain layer has to be less than 25–50 m.

Second, we employ the moving sensitivity domain approach (MSDA) which, due to the attenuation of the potential fields with distance, assumes zero contribution of 3D volume cells past certain radius (footprint) from the data receiver. This approach is similar to that developed for airborne electromagnetics (AEM) by Cox and Zhdanov (2007) and Cox et al. (2010) and for magnetotellurics (MT) by Zhdanov et al. (2011). Importantly, the sensitivity domain size is based not on the rate of attenuation of the fields due to the distance between the source and the receiver, but on the integrated sensitivity in the 3D volume. Within the sensitivity domain we include volume cells which constitute typically between 95% and 99% of the total sensitivity for each receiver which makes the inversion results virtually indistinguishable from those using the full 3D volume response. The size of the sensitivity domain is typically 200 km for gravity, 15 km for gravity tensor and TMI and 4 km for magnetic tensor (Čuma et al., 2012).

Third, the sensitivity matrix calculation loops have been restructured and optimized for scalar processors. We reorganized computations involving the kernels to reuse data as much as possible and ensure sequential access to the data in the memory (see Fig. 1a for a code sample). It is a well-known fact that memory performance in current computers is several orders of magnitude slower than the CPU processing speed so efficient data layout in the memory is critical. This resulted in a several-fold speedup as compared to the naive implementation.

Fourth, the whole 3D inversion program is implemented as a multilevel parallel application. The 3D earth model is divided in a distributed fashion over the Message Passing Interface (MPI). As the horizontal extent in the large surveys is larger than the vertical depth of interest, we distribute the 3D domain over the horizontal dimension in a round-robin fashion in order to achieve better load balance in case of non-rectangular surveys. On a fine-grained level, loops over the local domain cells, the observation points and a few other auxiliary loops within each MPI process are further parallelized with the shared memory OpenMP standard. Because current revision of the OpenMP standard only supports scalar reduction, if the accumulated value runs over data points, the OpenMP loop goes over the domain cells, and, vice versa. We have experimented with several scheduling schemes and determined that dynamic scheduling provides a modest advantage with a chunk size adjustable as an input parameter depending on the data and model size. The two-level parallelism approach has multiple advantages. It reduces the number of communicating processes, greatly reducing communication stress on the network. It also saves memory since there are data structures that do not need to be duplicated when parallelizing over the OpenMP threads. Locality of the processes and threads on the sockets and cores of the cluster nodes is controlled at runtime by placing the processes on CPU sockets and threads within each process on the cores of the process's socket. This way the CPU L3 cache is used by multiple threads and there is less stress on the slower main memory. The 3D inversion is relatively light in MPI communication, largely thanks to the linearity of the forward modeling operators, which makes all cells independent of each other. Most MPI communication is located in accumulation of the sensitivity products and in regularization as reduction operations. Thanks to this, the program exhibits excellent parallel scaling.

On the CPUs, we also found that it is advantageous to precompute a neighbor list of the cell–receiver pairs that are within the footprint distance. For typical TMI or gravity gradient survey, we achieve 40–60% speedup as compared to explicitly evaluating each cell–receiver pair horizontal distance. The footprint is assumed to be cylindrical; thus only neighbor list for one horizontal layer needs to be stored. Since the domain is MPI-distributed horizontally, in order to ensure sequential access into the neighbor list array, we bin receivers into horizontal cells (see array neigh_ind[Nxy][Nrfp] in Fig. 2a, Nxy is number of horizontal cells, Nrfp is number of receivers within footprint distance from a cell). Due to potentially large size of the neighbor list and limited GPU memory amount, we opted for not using it on the GPU.

Finally, we have implemented a mixed single–double precision approach, taking advantage of the fact that the values of the sensitivities in the potential fields are well represented by single precision floats. Moreover, thanks to the footprint approach, the small sensitivity values below the single precision resolution are not included in the calculation by default since they are outside of the footprint anyway. The key to correct single precision

a

```
for (ixx=0;ixx<Nx_local;ixx++) {  // loop over local MPI process X slices
  ix = ixx*nproc + ixstart; // global X index from round-robin distribution

  for (iy=0;iy<Ny;iy++) {

  #pragma omp parallel for
private(xdr,ydr,zdr,ir,irr,irt,ic,rr,rr2,rr3,rr5,lxdr,iz,c,id,z,irm)
    for (irr=0;irr<Num_neigh;irr++) {  // loop over receivers within this
horizontal cell's footprint, parallelized with OpenMP
      ir = Neigh_ind[ix+iy*Ny][irr];  // receiver index, irr sequential
      xdr=x[ix]-rx[ir].x; ydr=y[iy]-ry[ir].y;  // horizontal Rx-cell distance
      for (iz=0; iz<Nz; iz++) {
        zdr=z[iz]-rx[ir].z;  // vertical Rx-cell distance
        rr2 = 1/(xdr*xdr+ydr*ydr+zdr*zdr);
        rr = Dsqrt(rr2); rr3 = (rr*rr2); rr5 = (rr3*rr2);
        id = ix*Nx_local*Ny + iy*Nz + iz; // model cell index
        for( ic=0;ic<invdp->Nc; ic++)  {  // loop over data components
          irt = ir + Nrec;  // data point index
          preddatal[irt] += get_sens(xdr,ydr,zdr,rr2,rr3,rr5,ic)*m[id];
}}}}}
MPI_Allreduce(preddatal, preddata, Nrec*Nc, MPI_DOUBLE, MPI_SUM,
MPI_COMM_WORLD);
```

b

```
#pragma acc kernels loop independent present(Nrec,Nc,Nx,Ny,Nz,rx,…])
copyin(m[Nx_local*Ny*Nz]) copyout(preddatal[Nrec*Nc])
for (irt=0;irt<Nrec*Nc;irrt++) {  // loop over receivers and components,
offloaded to the GPU using the OpenACC kernels loop directive
ic = irt/Nrec;  ir = irt - ic* Nrec; // true component and receiver index
#pragma acc loop seq
  for (ixx=0;ixx<Nx_local;ixx++) {  // loop over local MPI process X slices
    ix = ixx*nproc + ixstart;   // global X index
    for (iy=0;iy<Ny;iy++) {
      xdr=x[ix]-rx[ir].x; ydr=y[iy]-ry[ir].y;
      r2xy = (xdr*xdr+ydr*ydr); // horizontal Rx-cell distance squared
      if (r2xy <= footDist2) {  // footDist2 is footprint squared
        for (iz=0; iz<Nz; iz++) {
          zdr=z[iz]-rx[ir].z;
          rr2 = 1/(r2xy+zdr*zdr); // 1 over Rx-cell distance squared
          rr = Dsqrt(rr2); rr3 = (rr*rr2); rr5 = (rr3*rr2);
          id = ix*Nx_local*Ny + iy*Nz + iz;  // model cell index
          preddatal[irt] += get_sens(xdr,ydr,zdr,rr2,rr3,rr5,ic)*m[id];
}}}}}}
MPI_Allreduce(preddatal, preddata, Nrect*Nc, MPI_DOUBLE, MPI_SUM,
MPI_COMM_WORLD);
```

Fig. 1. a. Simplified C code of the scalar CPU implementation of the predicted data calculation. Function get_sens contains a switch statement that selects and executes the appropriate data component kernel. Fig. 1b. Abbreviated C code of the OpenAcc GPU implementation of the predicted data calculation. In the actual implementation the get_sens function is manually inlined since the PGI compiler version 12.8 that we used does not support non-inlined subroutines in the accelerated kernel.

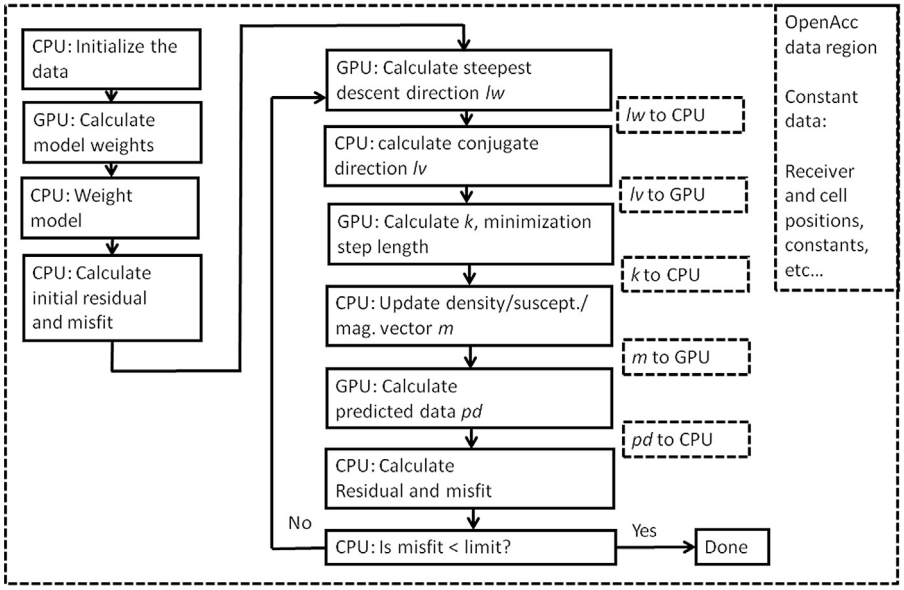

Fig. 2. Flowchart of the regularized reweighted inversion implementation using OpenACC. Program flow is shown in solid lines and data flow in dashed lines.

implementation is to store the sensitivity matrix accumulate products into double precision variables. For example for calculation of **lw**, we calculate each sensitivity kernel in single precision, but the product of $\mathbf{A}_{ij} \times \mathbf{r}_j$ is accumulated into a double precision variable. The final accumulated value is then copied into a single precision property value, e.g. $\mathbf{lw}_i$. This results in minimal difference between the double precision (DP) and mixed single–double precision (SPDP) inversion results. Since the scalar CPU unit achieves the same performance in single and double precisions, SPDP performance is the same as DP on the CPUs. The GPU's shared memory multithreading (SIMT) approach allows for efficient vectorization of the many component's potential fields kernels and thus also improves performance of the SPDP since DP vectors are replaced by twice as many SP vectors. An added benefit of the SPDP version is half the memory usage of the DP code, which is especially useful on memory limited accelerators. We utilize preprocessor directives to switch between the SPDP and DP so building one or the other is as simple as changing a compiler flag before the build.

The original MPI-OpenMP code is well suited for an accelerator port. Its main computation kernel is massively parallel, it runs over all domain cells times all data points within each cell's footprint,

guaranteeing a very high throughput on the accelerator. The OpenMP kernels have already been computationally optimized therefore the multi-nested kernel loops that were present in the OpenMP implementation were mostly kept intact, replacing the OpenMP parallelization directive with OpenACC. The interchange ability between the OpenMP and OpenACC also proved very useful in debugging. In general, a well optimized OpenMP threaded loop is an excellent candidate for OpenACC kernel. However, we did have to do a few loop rearrangements to improve the vectorization and also be careful about the data accumulation within these loops. There were several further problems which related to the limitations of the Portland Group (PGI) compiler's OpenACC implementation at the moment, such as mapping of the nested loops into thread blocks and threads, and support for multiple switch statements (which we use to differentiate between many gravity and magnetics components). Dealing with these problems, and figuring out the optimal accelerator loop mapping took a good part of the porting efforts. In Fig. 1, we show the simplified code for the predicted data calculation using CPU and GPU. In the scalar CPU code in Fig. 1 a, we loop over the model horizontal dimensions first, followed by a loop over the receivers within that cell's footprint, and then followed by the vertical model dimension and the data components. This minimizes the amount of arithmetic operations needed. In the GPU implementation (Fig. 1 b), the GPU kernel loops over the all the data (receivers times the components), eliminating the need for expensive accumulation of the predicted data accumulate across the GPU threads. Also, the receivers vs. components are ordered such that the receiver index is the fastest changing, minimizing the need to calculate multiple component branches in a single GPU warp and resulting in better vectorization potential.

Thanks to the on-demand calculation of the sensitivity matrix, our inversion code also does not require massive data transfers between the main memory and the accelerator. Most of the data needed in the kernel computation are constant (receiver and cell positions, topography, etc.), so they can be loaded to the accelerator memory only once at the start of the computation. Then there are only a handful of arrays of the size of the number of inversion domain cells *Nm*, or the number of data points *Nd*, that need to be communicated between the accelerator and the host, and vice versa. This results in a negligible amount of time spent on data transfers. One aspect that required a larger coding rewrite is the OpenACC limitation that only contiguous arrays can be transferred to the GPU. Since we store some of our data arrays in structures, we had to create separate arrays for each property and transfer it to the GPU separately rather that transferring a single structure that holds all these arrays. Fig. 2 shows a flow chart of the OpenACC inversion algorithm indicating parts that are executed on the host and on the accelerator and data moves between the host and the accelerator.

The coding effort was fairly minimal; it took us less than two weeks to implement the OpenACC in the state of the art production grade inversion code, which included the time to learn the OpenACC basics and troubleshooting the compiler limitations. The latter part involved interaction with PGI engineers who were very responsive and helpful.

4. Results

For the validation and performance evaluation we chose a Total Magnetic Intensity (TMI) airborne dataset collected by Fugro around Broken Hill, NSW. Broken Hill is a historic mining district in New South Wales (NSW), Australia, and host of the world-class Broken Hill strataform sediment-hosted Ag–Pb–Zn deposit. The host geology consists of the Willyama Supergroup of metamorphosed clastic and volcanoclastic sediments, basic to acidic volcanics, and intrusions of 1715–1590 Ma age. Mineralization is sediment exhalative in origin, and subsequently modified by metamorphism, folding and shearing. Today, mining has virtually eliminated the magnetic response of the Broken Hill orebody; however, there are several other magnetic anomalies in the area. The survey covers approximately 55×60 km and includes a bit over 1 million data points.

Here we should emphasize that in this work we only use scalar CPU implementation. Results from a vectorized implementation will be reported in a future contribution. The vector floating point unit (FPU) model puts restrictions on the data locality and on the program flow. Our potential fields inversion was designed for fast throughput on a scalar FPU, having a branch to choose the given potential field component kernel (one out of ca. 50) in the innermost loop. In order to vectorize, the loop structure has to be reorganized, by moving the loop over data components outside, resulting in an extra calculation. Another problem with the vector SIMD implementation in the CPUs is the treatment of program branches. The GPUs also include vector FPUs, however, their vector execution model (SIMT) uses more efficient logic for treatment of program branching in the hardware by executing only those branches that are needed in the given vector warp (subset of the vector, commonly of size 32), resulting in a minimal overhead if only single branch statement is executed in each warp. The SIMD implementation on the CPU executes each branch for every vector element, even if that branch is not being used. This results in reduced performance when multiple data components are used jointly in the inversion. For vectorized CPU single component (TMI susceptibility) inversion, we achieve close to theoretical speed up as compared to the scalar CPU, however, in the multi-component case, e.g. the TMI magnetization vector inversion, the scalar code performs better than the vector code. Fixing this problem would require writing unique loops for each of the (ca. 50) data components, which we have not done yet as the GPU performance is satisfactory for our current needs.

For initial evaluation, we used coarser domain with $200 \times 200 \times 50$ m cells, resulting in about 3.9 million cells and just under 35,000 data points. The data were inverted for susceptibility only. Table 1 summarizes the timings on a single dual socket Intel Westmere EP six core 2.8 GHz CPU (Xeon X5660) node (12 cores total) with two NVidia Tesla M2090 accelerators. On this node we ran two MPI processes, each process using six OpenMP threads. For the GPU runs, we also ran two MPI processes each also using one Tesla card. We ran 10 iterations for each inversion.

Notice first that the mixed single–double precision CPU code runs almost as fast as the double precision. This is because the kernel loops are scalar and as such the single and double precisions have about the same instructional requirements when run on the scalar floating point CPU unit. The Tesla card shows about $13\times$ speedup for the DP code and $19\times$ speedup for the SPDP code, as compared to the 12 cores of the Westmere EP. The theoretical speedup of the GPU vs. CPU is $5\times$ for DP, so, we are achieving roughly twice that. This is due to the lack of vectorization on the CPU. The SSE4 unit on the Westmere EP processors is capable of processing DP vectors of size two (128 bits), thus vectorization doubles the CPU speed that we achieve with the scalar kernel. Vectorizability of codes on CPUs is becoming

Table 1
Single node CPU and GPU inversion performance on a small model with double precision (DP) and mixed single–double precision (SPDP).

	CPU DP	CPU SPDP	GPU DP	GPU SPDP
Runtime (s)	1760	1741	134	92
Speedup w.r.t. CPU DP	1.00x	1.01x	13.1x	19.2x

increasingly important since the current Intel Xeon CPUs have 4 DP wide (256 bytes) vector unit and the Intel Phi accelerator 8 DP wide (512 bytes). Also note that we compare parallel single CPU performance to parallel single GPU performance, since current CPUs have multiple processing units (cores).

The benchmarking model presented in Table 1 is relatively small, especially when accounting for a limited number of receivers in each cell's footprint, and as such does not provide nearly as much parallelism for the GPU as more realistic finer model discretizations. For a more detailed study we used 50 m^3 cell size in the inversion domain and inverted to depth of 4800 m, resulting in $1094 \times 1190 \times 96$ grid with just under 125 million cells. We ran 10 inversion iterations.

Table 2 shows the timing of mixed single–double and double precision OpenACC implementation on two cluster nodes with two Intel Westmere EP six core CPUs at 2.8 GHz per node equipped with two NVidia Tesla M2090 cards per node, as compared to 48 cluster nodes (576 cores) with the same CPUs. All inversions ran two MPI processes and six OpenMP threads per node. The GPU inversion utilized one Tesla card per process for the kernel calculations and six OpenMP cores for non-accelerated auxiliary calculations. The 10 inversion iterations using CPUs took 6 305 s on 48 CPU nodes, while the same calculation on 2 GPU nodes took 6 737 s. The SPDP implementation reduces the runtime by a further 40%. Since we have determined earlier (Čuma et al., 2012) that our CPU inversion code scaling is nearly linear, the GPU implementation improves the speed ca. $22\times$ for the double precision and ca. $32\times$ for the mixed single–double precision.

In Table 3, we present weak MPI scaling for the GPU implementation. In weak scaling, we increase the problem size proportionally to the number of processes, so, ideally, the runtimes stay the same for different number of nodes and the runtime ratios stay at 1. We kept the number of data points the same (131,072) and doubled the number of cells with each doubling of number of nodes, thus having 31,244,640 cells for one node calculation up to 124,978,560 cells for 4 nodes. The memory usage per process on the host (running two processes per node) in the double precision case was about 13 GB, mainly storing the large model parameters arrays. The GPU memory usage was just 500 MB, well below the 6 GB capacity of the Tesla M2090 card, since only limited number of arrays needs to be duplicated on the GPU. In the mixed single–double precision case, the memory usage was about half. The parallel scaling is nearly linear; although there seems to be about 1% performance deterioration with doubling the node count, the variation is too small not to rule out the computation noise. We note that CHPC only has six GPU nodes so four nodes (8 processes) was the maximum we could scale this test to.

Table 2
Runtime and speedup of CPU and GPU inversion on a large model. The speedup is scaled to provide value per one CPU or one GPU.

	Runtime (s)	Speedup per one process wrt. CPU implementation
48 CPU nodes DP	6305	1x
2 GPUs DP	6831	22.2x
2 GPUs SPDP	4653	32.5x

Table 3
Weak scaling of GPU accelerated inversion.

	DP runtime (sec)	Parallel scaling	SPDP runtime (sec)	Parallel scaling
1 node	6737	1.00	4617	1.00
2 nodes	6832	1.01	4654	1.01
4 nodes	6896	1.02	4703	1.02

However, since all we did as compared to the original MPI/OpenMP code is to replace the non-communicating sensitivity calculation kernels that are now offloaded to the GPU, we expect the code to scale well to hundreds of GPU nodes.

Next we compared the mixed single–double and double precision results of inversion for the magnetization vector inversion, a procedure that can be useful in detecting attributes of remanent magnetization Zhdanov et. al. (2012). Due to three times larger computational requirements, we evaluate the difference using the coarser domain and data grid. In Fig. 3 we look at the magnetization vector absolute value obtained with DP and the difference between DP and SP-DP values after 100 inversion iterations. The

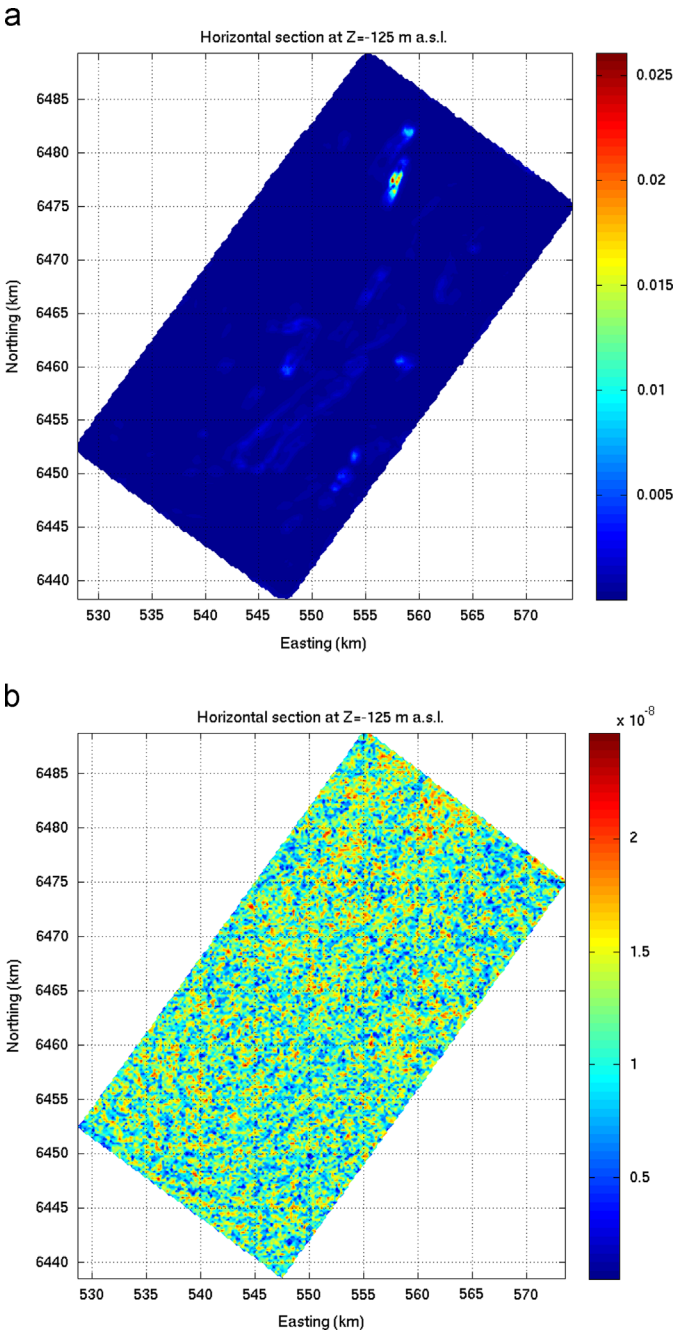

Fig. 3. a Magnetization vector absolute value of the Broken Hill survey area 125 m below sea level obtained with double precision inversion. Fig. 3 b Difference between magnetization vector absolute value of the Broken Hill survey area 125 m below sea level obtained with double precision and mixed single–double precision inversion.

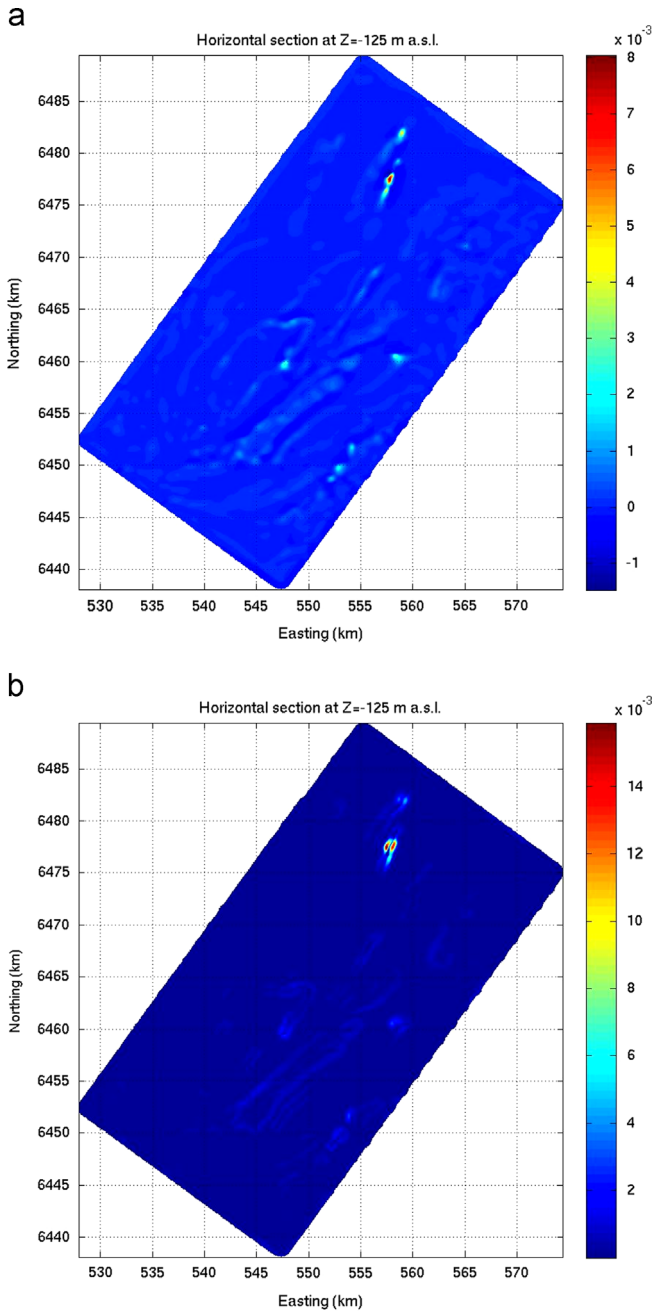

Fig. 4. a Induced part of the magnetization vector from the fine grid inversion of the Broken Hill area survey at 125 m below sea level. Fig. 4 b Remanent part of the magnetization vector from the fine grid inversion of the Broken Hill area survey at 125 m below sea level.

differences are minimal, with the recovered model magnetization vector root mean square difference (RMSD) of 1.63×10^{-6}. The misfits for the DP and SPDP are the same to the fifth significant digit. The predicted data values have RMSD 1.66×10^{-4}, which brings assurance that the DP and SPDP results are almost identical

Finally, we take advantage of the speedup provided by the OpenACC mixed single–double precision implementation to invert the Broken Hill dataset for the magnetization vector on the fine 50 m^3 grid. The 70 inversion iterations took 14.6 h to complete on four cluster nodes with two Tesla M2090 accelerators each, a similar CPU only inversion would require hundreds of nodes to be done in the same time.

In Fig. 4 we present the induced and remanent components of the magnetization. The induced part is caused by the present Earth's magnetic field, and is aligned parallel to this field. The remanent component is plotted as a projection of the recovered magnetization vector perpendicular to the Earth's magnetic field. Most of the anomalies in this survey area show both induced and remanent component which suggests demagnetization effects rather than pure remanence. Strong magnetic anomaly tends to shift the direction of the magnetization vector away from the inducing field. Inversion for magnetization vector is capable of recovering anomalies with both remanence and demagnetization while inversion for scalar susceptibility in these cases is problematic.

5. Conclusions

Thanks to the code structure that was developed for the OpenMP multithreading, the OpenACC implementation was straightforward and resulted in up to 22× double precision speedup on a node equipped with two NVidia Tesla M2090 cards as compared to a 2.8 GHz dual CPU 12 core Intel Westmere EP cluster node. Mixed single–double precision increases this speedup to up to 32×, as the scalar CPU implementation does not benefit from this approach. From this experience we believe that directive based approaches such as OpenACC, or future OpenMP for accelerators, can be a good choice for scientists since they require significantly less coding efforts than the alternatives (CUDA or OpenCL). A well designed threaded OpenMP code should be fairly straightforward to port to OpenACC and run on a GPU, and other future accelerators.

The magnetization vector inversion of the Broken Hill area survey demonstrates the capability of the accelerated inversion code and offers a more concise interpretation of the magnetic features in the area. Our massively parallel inversion method coupled with the OpenACC implementation enables geologists to get a complete 3D picture of density, susceptibility or magnetization vector covering very large areas.

Acknowledgments

We acknowledge support of the University of Utah's Center for High Performance Computing (CHPC). the Consortium for Electromagnetic Modeling and Inversion (CEMI), and TechnoImaging. We furthermore thank Mat Colgrove from PGI for help and useful comments, and Brian Haymore from CHPC for comments on the manuscript. Finally, we are grateful to the editor and two anonymous reviewers for comments that helped to improve the manuscript.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2013.10.004.

References

Beiki, M., 2010. Analytic signals of gravity gradient tensor and their application to estimate source location. Geophysics 75, I59–I74, http://dx.doi.org/10.1190/1.3493639.

Beiki, M., Pedersen, L.B., 2010. Eigenvector analysis of gravity gradient tensor to locate geologic bodies. Geophysics 75, I37–I49, http://dx.doi.org/10.1190/1.3484098.

Chen, Z., Meng, X., Guo, L., Liu, G., 2012. GICUDA: a parallel program for 3D correlation imaging of large scale gravity and gravity gradiometry data on graphics processing units with CUDA. Comput. Geosci. 46, 119–128.

Choi, J., Demmel, J., Dhillon, I., Dongarra, J., Ostrouchov, S., Petitet, A., Stanley, K., Walker, D., Whaley, R.C., 1996. ScaLAPACK: a portable linear algebra library for distributed memory computers—design issues and performance. Comput. Phys. Commun. 97, 1–15.

Cox, L.H., Zhdanov, M.S., 2007. Large-scale 3D inversion of HEM data using a moving footprint. Presented at 77th Annual Meeting, SEG, Expanded Abstracts, San Antonio, Texas.

Cox, L.H., Wilson, G.A., Zhdanov, M.S., 2010. 3D inversion of airborne electromagnetic data using a moving footprint. Explor. Geophys. 41, 250–259, http://dx.doi.org/10.1071/EG10003.

Čuma, M., Wilson, G.A., Zhdanov, M.S., 2012. Large-scale 3D inversion of potential field data. Geophys Prospect. 60, 1186–1199, http://dx.doi.org/10.1111/j.1365-2478.2011.01052.x.

Davis, K., Li, Y., 2011. Fast solution of geophysical inversion using adaptive mesh, space-filling curves and wavelet compression. Geophys. J. Int. 185, 157–166, http://dx.doi.org/10.1111/j.1365-246X.2011.04929.x.

Dransfield, M., Zeng, Y., 2009. Airborne gravity gradiometry: terrain corrections and elevation error. Geophysics 74, I37–I42, http://dx.doi.org/10.1190/1.3170688.

Fedi, M., 2007. DEXP: A fast method to determine the depth to the sources of potential fields. Geophysics 72, L1–L11, http://dx.doi.org/10.1190/1.239945215.

Hornby, P., Boschetti, F., Horowitz, F.G., 2002. Analysis of potential field data in the wavelet domain. Geophys. J. Int. 137, 175–196, http://dx.doi.org/10.1046/j.1365-246x.1999.00788.x.

Kirkendall, B., Li, Y., Oldenburg, D., 2007. Imaging cargo containers using gravity gradiometry. IEEE Trans. Geosci. Remote Sens. 45, 1786–1797.

Lee, J.B., 2001. FALCON gravity gradiometer technology. Explor. Geophys. 32, 247–250, http://dx.doi.org/10.1071/EG01247.

Li, Y., 2001. 3-D inversion of gravity gradiometry data. Proceedings of the 71st Annual Meeting, SEG, Expanded Abstracts, pp. 1470–1473. doi: 10.1190/1.1816383.

Li, Y., Oldenburg, D.W., 1996. 3-D inversion of magnetic data. Geophysics 61, 394–408, http://dx.doi.org/10.1190/1.1443968.

Li, Y., Oldenburg, D.W., 1998. 3-D inversion of gravity data. Geophysics 63, 109–119, http://dx.doi.org/10.1190/1.1444302.

Li, Y., Oldenburg, D.W., 2003. Fast inversion of large-scale magnetic data using wavelet transforms and a logarithmic barrier method. Geophys. J. Int. 152, 251–265, http://dx.doi.org/10.1046/j.1365-246X.2003.01766.

Moorkamp, M., Jegen, M., Roberts, A., Hobbs, R., 2010. Massively parallel forward modeling of scalar and tensor gravimetry data. Comput. Geosci. 36, 680–686.

Phillips, N., Nguyen, T.H., Thomson, V., Oldenburg, D. and Kowalczyk, P., 2010. 3D inversion modeling, integration, and visualization of airborne gravity, magnetic, and electromagnetic data—the QUEST Project. Presented at the International Workshop on Electrical, Gravity and Magnetic Methods, Capri.

Portniaguine, O., Zhdanov, M.S., 1999. Focusing geophysical inversion images. Geophysics 64, 874–887, http://dx.doi.org/10.1190/ 1.1444596.

Portniaguine, O., Zhdanov, M.S., 2002. 3-D magnetic inversion with data compression and image focusing. Geophysics 67, 1532–1541, http://dx.doi.org/10.1190/1.1512749.

Salem, A., Ravat, D., 2003. A combined analytic signal and Euler method (AN-EUL) for automatic interpretation of magnetic data. Geophysics 68, 1952–1961, http://dx.doi.org/10.1190/1.1635049.

Thompson, D.T., 1982. EULDPH: a new technique for making computer assisted depth estimates from magnetic data. Geophysics 47, 31–37, http://dx.doi.org/10.1190/1.1441278.

Tondi, R., Cavazzoni, C., Danecek, P., Morelli, A., 2012. Parallel, 'large', dense matrix problems: application to 3D sequential integrated inversion of seismological and gravity data. Comput. Geosci. 48, 143–156.

Yang, D., Oldenburg, D.W., 2012. Practical 3D inversion of large airborne time domain electromagnetic data sets. Presented at ASEG 22nd Geophysical Conference and Exhibition, Brisbane.

Zhang, J., Wang, C.-Y., Shi, Y., Cai, Y., Chi, W.-C., Dreger, D., Cheng, W.-B., Yuan, Y.-H., 2004. Three-dimensional crustal structure in central Taiwan from gravity inversion with a parallel genetic algorithm. Geophysics 69, 917–924.

Zhdanov, M.S., 2002. Geophysical Inverse Theory and Regularization Problems. Elsevier Elsevier Science B.V., Amsterdam, The Netherlands.

Zhdanov, M.S., 2009. New advances in 3D regularized inversion of gravity and electromagnetic data. Geophys. Prospect. 57 (4), 463–478, http://dx.doi.org/10.1111/j.1365-2478.2008.00763.x.

Zhdanov, M.S., Wan, L., Gribenko, A., Čuma, M., Key, K., Constable, S., 2011. Large-scale 3D inversion of marine magnetotelluric data: Case study from the Gemini prospect, Gulf of Mexico. Geophysics 76, 77–87.

Zhdanov, M.S., Čuma, M., Wilson, G.A., Polomé, L., 2012. 3D magnetization vector inversion for SQUID-based full tensor magnetic gradiometry. In: Proceedings of the 82nd Annual Meeting, SEG, Expanded Abstracts. doi: 10.1190/segam2012-0740.