Journal of Applied Geophysics 124 (2016) 91–105

Contents lists available at ScienceDirect

Journal of Applied Geophysics

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

Macro-parallelisation for controlled source electromagnetic applications

Andrew Pethick *, Brett Harris

Department of Exploration Geophysics, Curtin University, GPO Box U1987, Perth 6845, Western Australia

ARTICLE INFO

Article history:
Received 25 June 2014
Received in revised form 11 November 2015
Accepted 13 November 2015
Available online 30 November 2015

Keywords:
MCSEM
TEM
Integral equation
Parallelisation
Super computing
Cray

ABSTRACT

Many geophysical computational problems can be referred to as "embarrassingly parallel". Parallel computing utilises linked CPU cores to solve computational problems. We create a "macro" parallelisation method that rapidly recovers solutions to large scale electromagnetic forward and inverse modelling problems. The method involves software operating above a generic electromagnetic data structure. Two examples are provided.

The first example quantifies the reduction in computational time where macro-parallelisation is applied to forward modelling of data generated during synthetic marine controlled source electromagnetic surveys. In the second numerical experiment we apply macro-parallelisation to recover the subsurface conductivity distribution from a large airborne transient electromagnetic survey spanning more than 2000 km^2. The computation time for inverting 98 thousand soundings with a serial batch approach on an i7 with a single thread was 65 h. Computational time from inverting 98 thousand soundings on a single thread of a standard i7 processor was 65 h. A 1700 times improvement in computation time was achieved through macro-parallelisation across just 350 cores of a Cray XC30 Supercomputer. Inversion of data for the full AEM survey took just 135 s.

Parallel computing is rapidly becoming an essential for geophysicists. We provide description, sequence diagrams, pseudo-code and examples to illustrate its implementation. In summary we present applied parallelisation for the masses.

1. Introduction

High performance computing methods are beneficial for geophysical modelling in earth science applications. Parallel electromagnetic computing methods are used to alleviate memory or speed limitations or both. We have set out to develop, test and convey practical methods for taking advantage of parallel computing and embarrassingly parallel electromagnetic forward and inverse solvers in geophysics. The phrase "embarrassingly parallel" refers to a computational process which can be easily divided into smaller processes and computed independently on separate threads, without impacting the final outcome. We aim to develop and implement a macro-parallelisation method. Our macro-parallelisation approach enables single-threaded electromagnetic algorithms to operate in a parallel environment. It is a 'macro' process because a wrapper is formed around the algorithm so the internal EM modelling code is not altered. Our objective was to develop software incorporating parallel computing and network methodology with the following baseline requirements:

i It must be simple to understand and implement.
ii It should require no modification of source EM code.
iii Several orders of magnitude improvement in computational speed should be achievable for larger problems requiring the solution of tens of thousands of sub-problems.
iv The application must be scalable from a few networked desktop PCs to clusters with thousands of nodes

Parallelisation of electromagnetic modelling algorithms is not new. Key and Ovall (2011) utilised MPI routines to reduce 3D forward modelling finite element problems to seconds on an 800 node cluster. Commer et al. (2008) imaged large scale field marine controlled source electromagnetic (MCSEM) data using a parallel algorithm on the IBM Watson Research Blue Gene/L supercomputer. Commer et al. (2008) reduced computation time from 4 months of processing time on distributed clusters to 24 h on a super computer utilising 32,768 tasks/processor. Newman and Alumbaugh (1997) created and tested a parallel conjugate gradient 3D electromagnetic inverse problem to alleviate memory issues and to improve execution time. Joint and cooperative inversion methods (e.g., Seismic and EM data) are more practical because of the advancements made in massively parallel computing (i.e., Takam Takougang et al., 2015). Key and Ovall (2011) present a similar parallelisation technique to the one we present, however our parallelisation removes the need to modify existing algorithmic code. This is particularly useful for legacy, compiled, closed source, license restricted or poorly document electromagnetic modelling code.

We present a parallelisation technique to improve computational time for forward and inverse controlled source electromagnetic

---

* Corresponding author.
E-mail addresses: Andrew.Pethick@curtin.edu.au (A. Pethick), B.Harris@curtin.edu.au (B. Harris).

http://dx.doi.org/10.1016/j.jappgeo.2015.11.013
0926-9851/

problems. Our approach integrates existing EM codes as subroutines within a high level multi-threaded electromagnetic framework. We illustrate our methods with a 3D forward modelling and 1D inversion example. In the first example we create a macro-parallelisation wrapper for a MCSEM 3D integral equation code and simulate a MCSEM survey distributing the load over a network of computers. In the second example, our parallelisation approach is applied to the inversion of a large Airborne TEM survey in a sedimentary basin. The approach is tested on both a networked grid of PC's, connected via ethernet, and on a Cray XC30 super computer.

2. Developing a macro electromagnetic parallel modelling framework

We have developed an electromagnetic modelling framework to provide higher level functionality to most electromagnetic modelling algorithms. This framework provides methods to store, retrieve and sort electromagnetic data and survey information. It also controls network protocols and execution functionality. A higher level server program was designed to automatically divide a larger electromagnetic problem into smaller tasks and then to distribute them amongst a number of linked processors to be solved concurrently. Both the client and server are built upon this framework.

2.1. Programming heuristics

Java is an example of a modern high level programming language. We have used the Java programming language for its versatility and ease of use. Java is a cross platform programming language with automatic memory management (i.e., garbage collection) and provides access to most system call. Java has been selected because graphical user interface, network IO, file IO, memory management, execution and threading capabilities are provided without the need for external third party algorithms. The traditional geophysical programming stables, Fortran and C, offers limited graphical user interface (GUI), network or multi-threading capacity without the help of third party libraries, which may be platform dependent. Java overcomes this problem without the need for third party libraries which reduces install time and improves cross platform versatility and development time. While the program is written in Java, the steps discussed for our macro-parallelisation method were deliberately generic to enable readers to replicate the method in programming languages other than. We avoided language specific grid/parallel computing libraries such as JavaGCL (He et al., 2003) for two reasons. (i) Firstly, to demonstrate that the process can be replicated without the use of advanced features and (ii) secondly, to be replicated in programming languages other than Java.

2.2. Electromagnetic data structure

A generic electromagnetic data structure can be used to enable the storage, sorting and retrieval of any electromagnetic survey component (e.g., survey geometry and data). This generic structure enables integrations of third-party EM compiled codes. Our Java code performs survey division and system execution and control of these third party codes. This data model is used to “wrap” the electromagnetic algorithm in a sophisticated data management package. The data structure is open source license and is freely available (Pethick, 2012). We summarise the key elements of our electromagnetic data structure in Fig. 1 using UML class diagrams (Pilone and Pitman, 2005). Common controlled source electromagnetic survey features include

i system or survey configuration;
ii transmitter arrays;
iii receiver arrays;
iv transmission waveforms;
v receiver windows (for time domain systems);
vi geo-electrical earth models;
vii output electromagnetic data.

The system, or survey configuration, defines the transmitter and receiver relationship. The transmitter array describes transmitter properties and geometry. The receiver array describes receiver properties and geometry. The geo-electrical model defines the conductivity distribution. The transmission waveform describes the electromagnetic field generated from the transmitter. The output electromagnetic data is recorded for each transmitter–receiver location for each transmission frequency or time window. These elements form a controlled source electromagnetic survey.

We have created an electromagnetic project (i.e., the ‘ElectromagneticProject’ class) to map real world electromagnetic structures into a coherent data model. The electromagnetic project stores all waveforms, transmitters, receivers and earth models (see Fig. 1). The electromagnetic project can contain multiple survey instances. Each survey instance (i.e., ‘surveyInstance’ class) contains a single instance of a survey. The survey instance class contains all the components required to create a CSEM survey. Only one system configuration, earth model, transmitter and receiver pattern, waveform and receiver window setting is allowed per survey instance. Frequency and time domain data are stored in the survey instance rather than the electromagnetic project because the data is specific to the survey instance. The survey instance enables the conversion between the generic data structure and the external algorithm's specific input and output files.

To integrate an EM modelling and inversion algorithm into the macro-parallelisation framework a new class implementing the ‘Algorithm’ interface is required. (see Fig. 2). The interface is in essence a strict template. Every third party algorithm typically requires its own input, output and execution file formats. The ‘Algorithm’ interface enables the conversion between the electromagnetic framework and the algorithm's own input and output file formats and enables custom execution sequences to be created for each algorithm executable. The ‘FileFormat’ interface covers import, export and execution functionalities.

2.3. Macro-parallelisation

Any modern computer can forward model a controlled source electromagnetic response. The forward and inverse solutions need to be completed in a practical time frame, especially when many forward models are computed, analyzed and updated based on previous results. More computing power is required with increasing survey and geo-electrical complexity. A sequence diagram is provided in Appendix B to show the order of processes/steps contained within the macro-parallelisation algorithm. Pseudo-code is also included so our process can be replicated. Our macro-parallelisation method involves several steps:

Step 1 Initialise a group of connected networked processors.
Step 2 Create independent execution environments.
Step 3 Divide the survey into an optimal number of input files.
Step 4 Compile results.

The first step of the macro-parallelisation process is to interconnect a network of computers and to establish the number of available cores. A server application was developed to handle connections and distribute tasks (see Fig. 3). The server creates a connection by opening a new server socket on a predetermined port. A server thread is then created to listen for incoming client connections. Upon each client connection an individual thread is added. This thread controls incoming and outgoing requests and file transfers. Upon the successfully connection between client and server sockets, the server's client thread sends a “success” command to the client. Upon retrieval of the “success” command the client transmits its known IP-address, number of maximum