Geofísica Internacional (2015) 54-1: 31-48

Original paper

Forward modeling of gravitational fields on hybrid multi-threaded cluster

Carlos Couder-Castañeda*, José Carlos Ortiz-Alemán, Mauricio Gabriel Orozco-del-Castillo and Mauricio Nava-Flores

Received: October 18, 2013; accepted: March 11, 2014; published on line: December 12, 2014

Resumen

La solución analítica de las componentes del tensor gravimétrico, utilizando la ecuación del potencial gravitacional para un ensamble volumétrico compuesto de prismas de densidad constante, requiere un alto costo computacional. Esto se debe a que el potencial gravitacional de cada uno de estos prismas tiene que ser calculado para todos los puntos de una malla de observación previamente definida, lo cual resulta en una carga computacional de gran escala. En este trabajo introducimos un diseño híbrido y su implementación paralela basada en OpenMP y MPI, para el cálculo de las componentes vectoriales del campo gravimétrico (G_x, G_y, G_z) y las componentes del tensor gravimétrico (G_{xx}, G_{xy}, G_{xz}, G_{yy}, G_{yz}, G_{zz}).El rendimiento obtenido conlleva a óptimas relaciones del speed-up, ya que el tiempo de cómputo es drásticamente reducido. La técnica de paralelización aplicada consiste en descomponer el problema en grupos de prismas y utilizar diferentes espacios de memoria por núcleo de procesamiento, con el fin de evitar los problemas de cuello de botella cuando se accesa a la memoria compartida de un nodo del cluster, que se producen generalmente cuando varios hilos de ejecución acceden a la misma región en OpenMP. Debido a que OpenMP solo puede utilizarse en sistemas de memoria compartida es necesario utilizar MPI para la distribución del cálculo entre los nodos del cluster, dando como resultado un código híbrido OpenMP+MPI altamente eficiente con un speed-up prácticamente perfecto. Adicionalmente los resultados numéricos fueron validados con respecto a su contraparte secuencial.

Palabras clave: gravedad, gradiometría, OpenMP, MPI, hyper-threading, clusters.

Abstract

The analytic solution of the gravimetric tensor components, making use of the gravitational potential equation for a three-dimensional volumetric assembly composed of unit prisms of constant density, demands a high computational cost. This is due to the gravitational potential of each one of these prisms must be calculated for all of the points of a previously defined observation grid, which turns out in a large scale computational cost. In this work we introduce a hybrid design and its parallel implementation, based on OpenMP and MPI, for the calculation of the vectorial components of the gravimetric field and the components of the gravimetric tensor. Since the computing time is drastically reduced, the obtained performance leads close to optimal speed-up ratios. The applied parallelization technique consists of decomposing the problem into groups of prisms and using different memory allocations per processing core to avoid bottleneck issues when accessing the main memory in one cluster node, which are generally produced when using too many execution threads over the same region in OpenMP. Due OpenMP can be only used on shared memory systems is necessary to use MPI for the calculation distribution among cluster nodes, giving as a result a hybrid code (OpenMP+MPI) highly efficient and with a nearly perfect speed-up. Additionally the numerical results were validated with respect to its sequential counterpart.

Keywords: gravity, gradiometry, OpenMP, MPI, hyper-threading, clusters.

C. Couder-Castañeda*
J. C. Ortiz-Alemán
M. Gabriel Orozco-del-Castillo
Mexican Petroleum Institute
Eje Central Lázaro Cárdenas, 152
San Bartolo Atepehuacan, Gustavo A. Madero
07730, Ciudad de México
*Corresponding author: ccouder@esfm.ipn.mx

M. Nava Flores
División de Ingeniería en Ciencias de la Tierra
Facultad de Ingeniería
Universidad Nacional Autónoma de México
Ciudad Universitaria
Delegación Coyoacán, 04510
México D.F., México

Introduction

The shared memory architecture is becoming more common every day in the high-performance computing market. With the hardware technology advances allowing us to have a great number of cores with access to the same memory locations, nowadays it is not that expensive to have systems with forty or sixty cores using shared memory. OpenMP is now a standard for symmetric multiprocessing systems (SMP) (even can be used transparently in the Xeon Phi architecture (Calvin *et al*., 2013)) sustained by a combination of function and compiler directives, a standard for the symmetric multiprocessing (SMP) systems (Dagum and Menon, 1998; Curtis-Maury *et al*., 2008). OpenMP has proven to be a powerful tool for SMP due to several reasons: it is highly portable; it allows fine and medium granularity, each thread can access to the same global memory; and has their own private memory, and it also has a greater level of abstraction than MPI model (Brunst and Mohr, 2008).

MPI is a library supported on the Same Program Multiple Data (SPMD) model and on the message passing model, with an explicit control of the parallelism. The processes can only read and write in their respective local memories and the data in these memories is transferred through calls to functions or procedures which implement the message passing model. Among the principal characteristics of MPI are that it can run in architectures of shared and distributed memory, is convenient for medium to coarse granularity and that employment is widely extended, making it extremely portable among platforms (Krpic *et al*., 2012).

Using a hybrid programming model we can take advantage of the benefits of two programming models OpenMP and MPI. MPI is normally used to control the parallelism among cluster nodes, while OpenMP is applied in the creation of threads of fine granularity tasks within each node. Most applications developed in hybrid model involves a hierarchical model: MPI is for the higher level and OpenMP for the lower one (Smith, 2000).

One of the potential benefits of using hybrid model programming consists of getting rid of the barrier of scaling that each model has. Generally, in MPI the scaling is limited by the communications cost, because an application is affected by the overload of communication when the number of processes is increased. In OpenMP the performance of an application is affected by cache coherence problems and access to shared memory which may lead to bottleneck issues between the execution threads when trying to access memory. By mixing these methodologies of parallel programming (OpenMP and MPI), we can obtain a more diverse granularity of the application and therefore a better performance than by using each one on its own.

There are different applications which use this programming paradigm: OpenMP with MPI. For example, in the solution of sparse linear systems (Mitin *et al*., 2012), in graph-coloring algorithms (Sariyuce *et al*., 2012), in some models of fluid dynamics (Amritkar *et al*., 2012; Couder-Castañeda, 2009) and finite element methods (Boehmer *et al*., 2012), in the simulation of turbulent fluids (Jagannathan and Donzis, 2012), even in the simulation of combustion chambers (Környei, 2012) and the implementation of neural networks (Gonzalez *et al*., 2012). As can be observed, there are numerous computational implementations using OpenMP with MPI, nevertheless, this type of design is supported on a natural decomposition of the domain (Carrillo-Ledesma *et al*., 2013), based on data. For our particular problem, each one of the processing units accesses all of the computational domain points.

In Figure 1 is depicted a domain decomposition, where each task (process or thread) is given some data subset on which to work. This domain decomposition is commonly used for example in finite differences problems where computational domains divided disjointly among the different tasks.

On the other hand, in the direct conformation of gravimetric data, an initial model for the source body is constructed from geological-geophysical information. The anomaly of such model is calculated and compared to the observed anomaly, after which the parameters are adapted to improve the adjustment between them. These three steps that arrange the model properties — *anomalies calculation, comparison* and *adjustment* — are repeated up to the observed and calculated anomalies are similar enough.

A mass volume can be approximated by a set of rectangular prisms; if chosen sufficiently small, each prism can be considered to have a constant density. Because of the superposition principle, the gravitational anomaly of a body can be approximated at anypoint by summing the effects of all the prisms over that point. Even though this methodology appears simple (by reducing the size of the prisms to better adjust the source body),