A massively parallel implementation of the common azimuth pre-stack depth migration

H. Calandra
F. Bothorel
P. Vezolle

When accompanied by the appropriate algorithmic approach, seismic imaging is an application that can take advantage of massively parallel computer systems. Three-dimensional (3D) pre-stack time migration (PSTM) and pre-stack depth migration (PSDM) are key components of seismic imaging and require very large computing resources. In this paper, we show that execution of these algorithms can be dramatically accelerated by massive parallelism. Many oil exploration and service companies purchase supercomputing clusters for performing 3D PSTM and PSDM seismic imaging. The common azimuth migration (CAM) algorithm, ported to many architectures, is particularly well suited to offshore marine applications. This paper describes the porting of the CAM algorithm to the IBM Blue Gene/L™ supercomputer, which requires introducing a second level of parallelism, building a parallel 3D-FFT (fast Fourier transform) routine, optimizing a tri-diagonal solver for SIMD (single-instruction, multiple-data) floating-point units, and addressing various I/O concerns. We present results obtained by using up to 16,368 processors for actual data provided from a marine seismic acquisition. Finally, we provide recommendations for porting other pre-stack algorithms to a massively parallel environment.

Introduction

The extraordinary challenges that the oil industry must face in hydrocarbon research require the development of leading-edge technologies. The use of seismic-reflection technology is currently an essential and strategic method to reconstitute the three-dimensional (3D) structure of portions of the earth. This reconstruction is a key function of seismic exploration because it is the basis on which geologists search for new oil deposits. Today, this analysis is possible because of the extraordinary revolution in deep-imaging techniques over the last 20 years. The industry has experienced a very large leap in the ability to process seismic data and to build an increasingly accurate image of the structure of the earth. One of the key components of this revolution is high-performance computing. Indeed, the fast evolution of computers has enabled the development of specialized algorithms that address the steadily increasing volumes of data generated by seismic acquisitions.

The pre-stack depth migration (PSDM) technique is currently the most preferred method used for 3D seismic imaging. The common azimuth migration (CAM) technique refers to a PSDM algorithm that is particularly suitable for the treatment of marine data acquisition because of its speed and the quality of its results. The characteristics of this method make its implementation effective in symmetric multiprocessor clusters. Using the computing and communications capabilities of the IBM Blue Gene/L* (BG/L) system, we show that it is possible to modify the CAM algorithm to obtain a massively parallel version of this algorithm.

In this paper, we first outline the seismic-reflection method for oil exploration and its challenges. Next, we present the PSDM and describe in detail the CAM technique and its algorithms. Then, we discuss how to take advantage of some of the features in the IBM BG/L system in order to implement a massively parallel version of the CAM algorithm, which can scale up to tens of

0018-8646/08/$5.00 © 2008 IBM

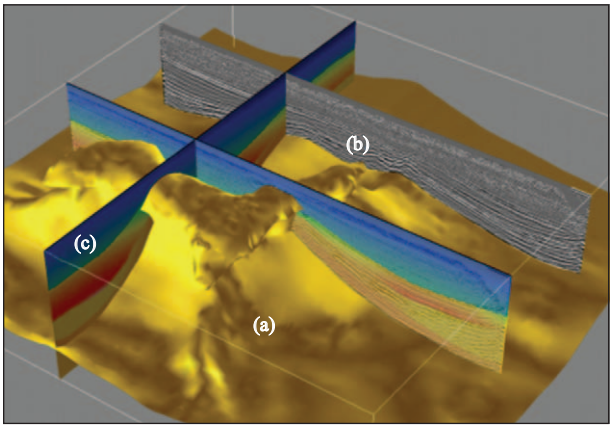

Figure 1

Different features that are part of the depth imaging process: (a) the definition of the geological structure; (b) the migration (or reflectivity) profile; (c) the velocity model profile.

thousands of processors with great throughput and performance. Finally, we present results for the BG/L supercomputer with up to 16,368 processors, achieved with data provided from a marine seismic acquisition.

The challenge of oil exploration by the seismic-reflection method

The search for hydrocarbons by using seismic-reflection methods can be considered as a scan of a portion of the earth. Seismic reflection is an indirect measurement technique that involves recordings on the surface of the earth produced by the echoes of a detonated explosion that propagate in the subsurface. Each signal is emitted by a source and is recorded by many hydrophones in the case of a marine acquisition. Each explosion for a given source generates a collection of reflected waves that are recorded by all the seismic sensors belonging to the acquisition device. A collection of recordings is called a *shot point*.

In order to illuminate the various geological layers as effectively as possible and to obtain the most consistent coverage possible, tens of thousands of shot points are produced during a seismic survey. The size of acquisition surveys has been steadily increasing from a few hundred square kilometers in the beginning of the 1990s to several thousand square kilometers today. The latest generation of seismic acquisition ships records several thousand shot points by using a large number of hydrophones placed in a dozen parallel casings that are 6 to 8 km in length and positioned behind the ship. These giant acquisitions generate an enormous amount of data (up to several tens of terabytes) that has been increasing dramatically since 2000. Thus, the oil industry needs powerful algorithms to precisely and rapidly process these large amounts of data in a global setting in which the search for hydrocarbons is extremely competitive.

Depth migration is the processing step in which reflections in seismic data are moved to their correct spatial locations, for example, relative to shot points, in areas where there are significant and rapid changes in wave-propagation velocity that distort the image. The accurate reconstruction of the subsurface is key in prospecting for hydrocarbon deposits. The migration process can be assimilated into an inverse problem, which consists of finding the geological structure that explains the data as accurately as possible. **Figure 1** illustrates three different stages of seismic reflection: the definition of the geological structure, the depth migration (or reflectivity) processing, and the construction of a velocity model.

The overall reconstruction process is iterative. After each depth migration, the resulting image is analyzed, the velocity model is modified, and a new depth migration is realized until process convergence is achieved. In this iterative process, the depth migration is the fundamental and the most computation-intensive stage.

Depth migration based on the common azimuth method

The oil industry has invented many methods of PSDM. Among these methods, the CAM method is particularly interesting for narrow azimuth marine acquisitions. Before describing the CAM method, we outline the principle of depth migration. Let us start by presenting the imaging principle as formulated by John Claerbout [1] in the early 1970s: "In any point on the reflectors, the up-going wave field is equal to the down-going field multiplied by the reflection coefficient." Thus, by calculating two wave fields, one up-going from measurements and the other down-going starting from the source, we can define the reflectors (the coefficient of reflection) at their intersection points. In practice, this consists of summing the contribution of all the shot points for all the subsurface points. The source field is propagated, whereas the receiving field is reverse propagated. The correlation of these two fields gives the image reflectors at $t=0$. The basic tool for depth migration is the wave equation. The wave equation differential operator is recursively applied to propagate the different wave fields in the depth direction. It is possible to distinguish two families of recursive methods by the wave equation:

- The shot-point method (also referred to as *shot profile migration*), in which the propagation is independently realized to each shot point.
- The method based on the principle of "survey sinking," or "double square root" (DSR) [1], which propagates the overall acquisition information at the same time. Biondi and Palacharla [2] developed the

CAM approach from this formulation, which is also known as the *DSR PSDM* method.

The wave equation in DSR formulation in real space can be expressed as

$$\frac{\partial P}{\partial z} = \left(\sqrt{\left(\frac{1}{V^2} - \left(\frac{\partial t}{\partial g} \right)^2 \right)} + \sqrt{\left(\frac{1}{V^2} - \left(\frac{\partial t}{\partial s} \right)^2 \right)} \right) \frac{\partial P}{\partial t} \quad (1)$$

and in Fourier space, $P(t,s,g) \rightarrow P(\omega,k_s,k_g)$, as

$$\frac{\partial P}{\partial z} = \left(\sqrt{\left(\frac{\omega^2}{V^2} - k_g^2 \right)} + \sqrt{\left(\frac{\omega^2}{V^2} - k_s^2 \right)} \right) P. \quad (2)$$

Here, (s,g) represent the surface localization of the sources and hydrophones; t is the time; V is the velocity of propagation; ω is the pulsation frequency; (k_s,k_g) are the wave numbers; and P is the wave field. These equations represent the simultaneous propagation in the z direction of seismic device sources and hydrophones.

We may apply the change of variables in the two-dimensional (2D) horizontal plane $(s,g) \rightarrow (m,h)$, where (m,h) represents the midpoint offset coordinate system defined by

$$m = \frac{s+g}{2}, \quad h = \frac{g-s}{2} \quad (3)$$

and in Fourier space by

$$k_m = \frac{k_g - k_s}{2}, \quad k_h = \frac{k_g + k_s}{2}. \quad (4)$$

Here, m, h, k_m, and k_h are 2D vectors in the horizontal plane: i.e., (m_x,m_y), (h_x,h_y), (k_{mx},k_{my}), and (k_{hx},k_{hy}).

Finally, the DSR Equation (2) in the midpoint offset coordinate system is

$$\frac{\partial P}{\partial z} = \left(\sqrt{\left(\frac{\omega^2}{V^2} - \frac{1}{4}\left[(k_{mx} - k_{hx})^2 + (k_{my} - k_{hy})^2 \right] \right)} + \sqrt{\left(\frac{\omega^2}{V^2} - \frac{1}{4}\left[(k_{mx} + k_{hx})^2 + (k_{my} + k_{hy})^2 \right] \right)} \right) P, \quad (5)$$

where (mx, my) are the 2D spatial dimensions, and (hx, hy) correspond to the offset. A solution to Equation (5) is

$$P(z + \Delta z) = e^{\Delta z k_z} P(z) \quad (6)$$

$$k_z = \left(\sqrt{\left(\frac{\omega^2}{V^2} - \frac{1}{4}\left[(k_{mx} - k_{hx})^2 + (k_{my} - k_{hy})^2 \right] \right)} + \sqrt{\left(\frac{\omega^2}{V^2} - \frac{1}{4}\left[(k_{mx} + k_{hx})^2 + (k_{my} + k_{hy})^2 \right] \right)} \right). \quad (7)$$

Equations (6) and (7) clearly illustrate the recursive process of DSR migration. The formulas concern a five-dimensional space (k_{mx}, k_{my}, k_{hx}, k_{hy}, ω), making this algorithm computationally intensive for large-acquisition datasets. In marine acquisition, the azimuth variation is small, and with suitable data pretreatment, the acquisition allows a constant azimuth. The application of the stationary phase in the hy direction is used to reduce the overall problem size by one dimension. This reduction corresponds to DSR migration with constant azimuth and is referred to as *CAM*. Using the principle of the stationary phase, we obtain a new equation [Equation (8)] that is comparable to a series of 2D pre-stack migrations in the in-line direction and post-stack migration in the cross-line direction [2]. In particular, we have

$$P(z + \Delta z) = e^{\Delta z \bar{k}_z} P(z),$$

with

$$k_z = \sqrt{\left(\frac{\omega^2}{V^2} - \frac{1}{4}\left[(k_{mx} - k_{hx})^2 \right] \right)} + \sqrt{\left(\frac{\omega^2}{V^2} - \frac{1}{4}\left[(k_{mx} + k_{hx})^2 \right] \right)},$$

$$\bar{k}_z = \sqrt{k_z^2 - k_{my}^2}. \quad (8)$$

This equation holds in the case of a homogeneous propagation domain. In the general case, with lateral velocity variation, we introduce the Fourier finite difference (FFD) [3] approximation of $\bar{k}_z$:

$$\bar{k}_z \approx k_{z0} + \omega\left(\frac{1}{V_S} + \frac{1}{V_G} - \frac{2}{V_0} \right) - \frac{\frac{1}{4\omega} A_1 \alpha k_{mx}^2}{1 - \frac{1}{4\omega^2} B_1 \beta k_{mx}^2} - \frac{\frac{1}{4\omega} A_1 \alpha k_{hx}^2}{1 - \frac{1}{4\omega^2} B_1 \beta k_{hx}^2} - \frac{\frac{1}{2\omega} A_2 \alpha k_{my}^2}{1 - \frac{1}{4\omega^2} B_2 \beta k_{my}^2} \quad (9)$$

V_0 is the velocity of propagation in the reference environment. The first term (k_{z0}) is called *phase shift correction*, the second term is called *thin lens correction*, and the last three terms represent the *wide-angle correction* constants A, B, and α and β.

Massively parallel implementation of CAM

The implementation of CAM is an iterative process over the depth and it consists of two steps. In the first step, the wave field is propagated using Equations (6) and (8). In the second step, the image is computed and stored.

The main data structures used by the algorithm are as follows:

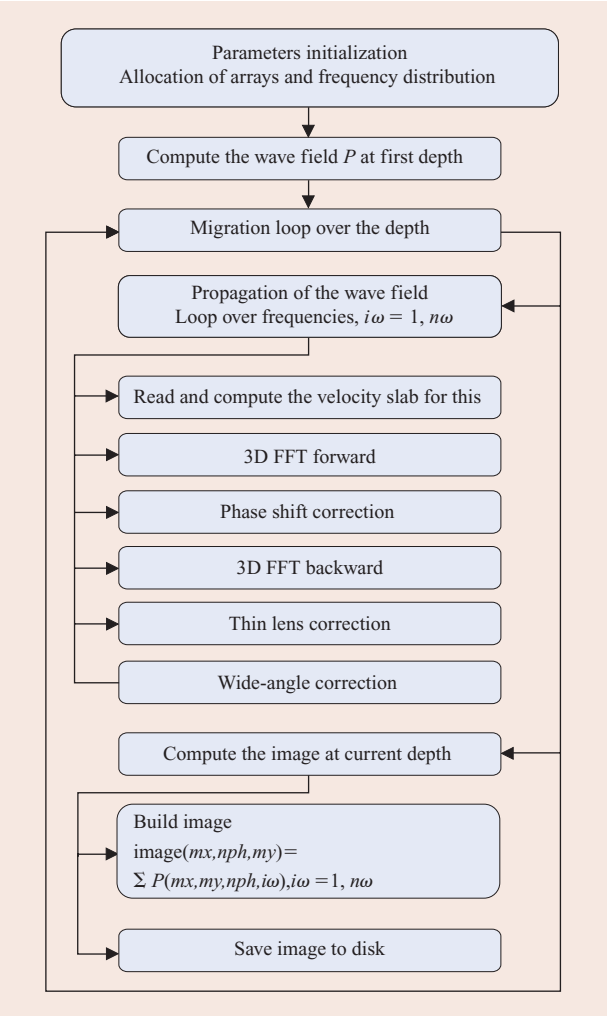

Figure 2

Flow diagram for the CAM code. The wide-angle correction block is discussed further in the section "Parallel domain decomposition of the wide-angle correction on the BG/L system."

- The complex wave field array P of size (mx, my, hx, $n\omega$), where mx and my are the 2D spatial dimensions, hx is the number of offsets, and $n\omega$ is the number of frequencies. This array is initialized with the frequency files from acquisition and updated at every depth.
- The real image array of size (mx, nph, my), where nph is the offset ray parameter.
- The real velocity array of size (mx_velo, my_velo). This array is computed at every depth using data from velocity files.

The global structure of the CAM implementation is illustrated in **Figure 2**. The typical dimensions for a real acquisition are $mx > 2{,}000$, $my > 1{,}500$, $hx < 128$, and $200 \leq n\omega \leq 500$. The dimensions are extended to match FFT (fast Fourier transform) nearest values.

The natural way to parallelize this algorithm is to distribute the frequencies accross Message Passing Interface (MPI) tasks, because the propagation step is fully independent of the frequencies. The compute-image step requires a global reduction in the number of frequencies and, therefore, in the number of MPI tasks. This current implementation of CAM in production, mainly on SMP clusters, has only one level of parallelism over the frequencies. In this standard cluster implementation, the number of tasks is limited by the number of frequencies. Shared memory parallelism has been applied to the low-level loops using multithreaded libraries and OpenMP** (Open Multiprocessing) directives. This new level of parallelism provides the opportunity to increase the number of processors involved in a single migration. We cannot expect to use more than four threads per task because the program efficiency starts to drop beyond this number. Even in the optimal configuration with one frequency per MPI task, we are limited to a relatively small number of processors (e.g., ~2,500 processors) and cannot take advantage of the large number of processors available on the Blue Gene/L system.

Moreover, this implementation requires a high memory footprint per process. The computation is globally dimensioned by the size of wave field array P (which depends on the 2D size of the surface, the number of offsets, and the number of frequencies per MPI task). For real data, this complex array contains more than 1.5 GB per frequency and per process. Thus, the memory per node of the Blue Gene/L system is too small to fit the required storage.

In order to take advantage of massively parallel systems, and especially those with low-latency networks, we introduce a new level of parallelism using domain decomposition. This decomposition is done in a single dimension, and when coupled with the frequency distribution, it allows us to utilize numerous processors (e.g., >100,000) while linearly reducing the per-processor memory footprint. This new level of parallelism is fully compatible with the multithreading layer already implemented. The spatial partitioning can be done in the y (i.e., my) direction. Most of the changes are independent of the decomposition direction, and the domain decomposition can be extended easily to the mx direction. To implement this new level of parallelism, the processes are distributed in frequency groups of size $nb\_cluster$. Each group is assigned a set of frequencies.

The wave field array is globally distributed throughout the frequency groups and within a group locally partitioned along the y (my) direction. The local wave field array is defined by $P(mx, myL = my/nb\_cluster, hx,$

nw/*nb_group*), where *nb_group* is the number of frequency groups, and the letter *L* denotes "local." Thus, the local data storage required is divided by the group size. The maximum number of processors involved in a migration is *nb_cluster* times *nb_group*. A distributed memory version of the 3D-FFT and wide-angle-corrections procedures is required in the propagation algorithm.

Parallel one-dimensional decomposition 3D FFT on the BG/L system

One of the most computationally intensive components of the common azimuth method is the volumetric FFT. Several studies have demonstrated that the most scalable and natural implementation of the 3D FFT on a BG/L system is based on a volume domain decomposition [4]. This method achieves scalability to a larger number of nodes (up to 16,368 nodes) because it allows the distribution of work for an $N \times N \times N$ FFT over the natural torus topology of the BG/L system.

Such a method cannot be applied within the CAM because of the natural parallelism over the frequencies and the limited size of the FFT. Note that we do not have to compute one large FFT but *nb_group*-independent 3D FFTs. For each frequency, a 3D FFT is computed by a limited number of processors (*nb_cluster*). Our aim is to create the most efficient and scalable 3D-FFT implementation for 128 through 256 processors. In order to minimize any change in the overall application, we have implemented a 3D FFT that is based on the slab decomposition in which the input data is partitioned in one direction in accordance with the general data decomposition.

We have developed two parallel versions of the 3D FFT depending on the input and output decompositions. In the first version, called *homogeneous I/O partitioning*, the output array has the same decomposition as the input array. This version always assumes that data is partitioned in the *y* direction. Each process stores the local array, $P(mx, myL = my/nb\_cluster, hx)$. The one-dimensional (1D) FFTs in the *x* and *z* directions are performed by multiple sequences of 1D FFTs. The 1D FFT in the *y* direction is realized in three steps: a global all-to-all communication to perform a decomposition in the *x* direction with storage in the array $P(mxL = mx/nb\_cluster, my, hx)$, multiple sequences of 1D FFTs in the *y* direction, and the inverse all-to-all communication to return to the original decomposition. In order to optimize the communication (via the use of *mpi_alltoall* or *mpi_alltoallv*), the last stage is realized iteratively over the *z* dimension. (The *mpi_alltoall* routine sends data from all processes to all processes. The *mpi_alltoallv* routine adds flexibility to *mpi_alltoall* in that the location of data for the send is specified and the location of the placement of the data on the receive side is specified.)

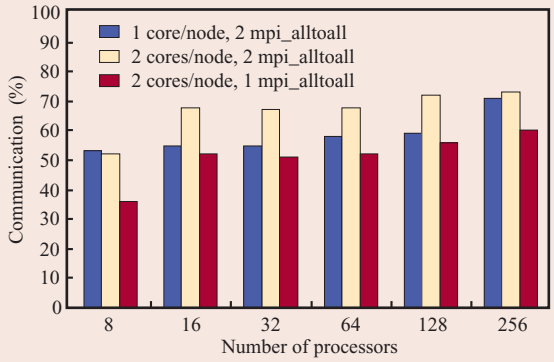

Figure 3

Percentage of overall time devoted to communications on the BG/L system for an FFT of size 2,304 × 1,680 × 64 using one-dimensional data decomposition. Shown here is the dependence on the number of cores per compute node, with homogeneous and nonhomogeneous output partitioning. The yellow and red bars differ in the number of mpi_alltoall steps in the FFT. Red indicates nonhomogeneous and yellow indicates homogeneous I/O partitioning.

Our measurements for relevant problem size for seismic migration show a communication cost that is more than 50% of the overall time, which dramatically limits the scalability of our application. It is important to note that our implementation of the volumetric FFT uses the ESSL (Engineering Scientific Subroutine Library) 1D serial FFT, which is specially tuned for the POWER* processor but increases the communication load.

In the second version of the 3D FFT, we have removed the last transformation, allowing us to return to the original *y* decomposition. This version, called *nonhomogeneous I/O partitioning*, accepts an input array with decomposition in the *x* or *y* direction and delivers results in the opposite decomposition. For instance, if the local input array is $P(mx, myL, hx)$, the resulting FFT is stored in an array $P(mxL, my, hx)$. This method divides the overall communications by a factor of 2. In order to limit the memory footprint, the implementation uses input and output arrays that are located at the same address.

Figure 3 shows the percentage of overall time devoted to communications on the BG/L system for a 3D FFT of size 2,304 × 1,680 × 64 for a varying number of processors in configurations with one or two cores per node. The BG/L ASIC (application-specific integrated circuit) node has two cores (or processors) [5]. The figure indicates the impact of the number of cores per node. The program runs that use two cores per node increase the ratio of computation to communication time compared to runs using one core per node, where MPI bandwidth

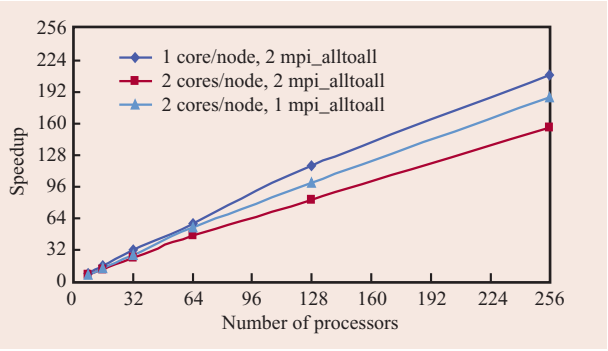

Figure 4

Three-dimensional FFT speedup with a size of 2,304 × 1,680 × 64 on the Blue Gene/L system, depending on the number of cores per compute node, with homogeneous (red plot) and nonhomogeneous (light blue plot) output partitioning.

is completely dedicated to a single core. This clearly emphasizes the utility of increasing the number of cores per node while keeping the network bandwidth constant. Our tests show around 35% downturn of the total time on the BG/L system, essentially due to the communication part. With the traditional clusters, the increase in the number of cores will be a significant impediment to a massively parallel approach for seismic algorithms based on 3D FFT.

As shown in **Figure 4**, the speedup on the BG/L system occurs with up to 256 processors, with a steady efficiency of greater than 0.8, with one core per node for the homogeneous version and two cores per node for the nonhomogeneous version.

In order to integrate the fastest 3D FFT into the overall algorithm, a phase shift correction routine was modified to manage data decomposition in an x or y direction. The communication part, in the sequence of three steps (FFT, then phase shift correction, followed by inverse FFT), can be divided by two, with a 30% or greater improvement in the overall 3D-FFT time.

Parallel domain decomposition of the wide-angle correction on the BG/L system

The wide-angle correction is the most computationally intensive component of the CAM. This algorithm is a correction of the wave field and depends on the lateral velocity gradient. Within a frequency group, the correction consists of sequentially applying the corrections corresponding to the last three terms of Equation (9), with one correction per direction. These corrections are achieved by solving a large number of the linear systems $Ap = Bp$, of size mx, hx, and myL, respectively, where A and B are two complex tri-diagonal matrices, and p is the corresponding vector in the wave field array. This algorithm has exactly the same structure as the volumetric FFT presented above, where the 1D FFT is replaced by the construction of the linear system followed by a tri-diagonal complex solver.

The complex tri-diagonal linear systems are solved by an LU direct solver, an excellent general-purpose equation solver that is well known in the literature. The communication functions are shared with the volumetric FFT. The build image algorithm, which directly follows the wide-angle correction, has a preferential decomposition direction in y, allowing it to reach optimal levels of performance for I/O and communications. Thus, unlike the 3D FFT, the second set of transformation data cannot be removed. Nevertheless, the scalability of the overall algorithm on the BG/L system is slightly higher than that for the 3D FFT, with efficiency greater than 0.9 for up to 256 processors, because of a better ratio of computations to communications.

The computation part of the direct tri-diagonal solver is of critical interest, and it is well known that the actual performance of a direct tri-diagonal solver is very low compared to the processor peak performance. Our aim is, therefore, to achieve the best sequential performance on the BG/L platform by changing the structure of the algorithm to address the following points:

1. The computation strongly depends on the memory latency and bandwidth because of the ratio of loads and stores per flops.
2. The backward dependency, in which a loop iteration depends on a previous iteration, does not allow the compiler to fully optimize the process loops and penalizes the filling of the floating-point unit (FPU) pipes.
3. The current version of the compiler for the BG/L system is not able to generate SIMD (single instruction, multiple data) for complex operations.
4. The weight (i.e., the cost) of the division operations is significant.

Points 1 and 2 are significantly improved by introducing a block solver that solves several linear systems at the same time. The matrices and the solution originally stored in 1D arrays are gathered by block in 2D arrays in which the first dimension is the block size (number of systems solved simultaneously), and the second dimension is the corresponding size of the problem. This data structure allows us to unroll the main loop while reducing the impact of the memory accesses on overall runtime. (When we use the phrase "unroll the loop," we mean the process in which the instructions that are called in multiple iterations of the loop are combined into a single iteration.)

The BG/L ASIC processor includes two double-SIMD 64-bit multiply–add FPUs, one for each core, allowing the chip to deliver up to 4 flops per cycle and per SIMD FPU. Unlike standard POWER chips, the BG/L processor chip does not provide two independent FPUs; the SIMD unit includes parallel primary and secondary arithmetic pipes, each with its own 32- × 64-bit floating-point registers (FPRs). The primary pipe executes the standard instructions and the SIMD instructions, while the second pipe executes only the SIMD instructions. The double FPU implemented on the BG/L chip offers more capabilities than a pure SIMD unit. Some instructions cause two different operations to be performed in the two pipes. For example, the instructions allow efficient support for complex cross products (for more details, see [5, 6]). As the Fortran compiler is not able to generate SIMD instructions for complex operations, we have developed a complex block tri-diagonal LU direct solver that solves four systems at the same time, where all the complex operations are replaced by SIMD intrinsic functions. The data structures are based on 2D arrays of dimension (4,*n*). This data structure requires that each column be aligned on a 16-byte boundary, and it generates quadword (16-byte) load and store instructions. It turns out that solving four systems simultaneously allows filling the pipes and getting the highest number of flops per cycle. The computation part for the division operations is divided by a factor of 5 using the parallel hardware estimation of the BG/L processor [7]. Our measurement has shown that the optimal block size is four with respect to reusing the 32 FPRs; a block of six generates a register starvation. All the operations are realized in 64-bit precision, but the input data can be in single or double precision. The use of single precision is about 15–20% faster because of the reduction in data for memory transfers.

All of these optimizations can be applied to architectures in addition to the architecture of the BG/L system. For instance, on x86 microprocessor architectures, the SIMD intrinsic functions can be replaced by SSE (streaming SIMD extensions) intrinsic functions.

Optimizing I/O for the BG/L massively parallel approach

The BG/L system delivers excellent I/O performance by distributing the I/O through dedicated I/O nodes that are independent of the compute nodes. Only the I/O nodes have direct access to the file systems. The compute nodes cannot directly access storage and must access data through the I/O nodes. Several compute nodes are connected to a single and dedicated I/O node using the tree network [5]. This subset of compute nodes that are physically connected to a single I/O node is called a *pset*. The ratio of compute nodes to I/O nodes depends on the node configuration, which is from 8 to 128 on the BG/L platform.

Compared with the standard computing cluster, the individual performance per I/O node is relatively limited, and the overall I/O bandwidth is achieved by using the maximum number of I/O nodes. The CAM implementation makes use of three types of files:

- *Input frequency files*—The total size of these files is typically greater than 300 GB. During the first step of the migration process, file reads are performed simultaneously (in parallel) over the frequency groups.
- *Velocity files*—Records from these files are taken at every depth and are performed in parallel over the frequency groups.
- *Image file*—This file contains the final image and is updated at every depth, in parallel.

In a communicator, I/O is done by the master task. In the BG/L system, the communicators are created to optimize the number of I/O nodes. The masters in each communicator are dynamically assigned during the runtime to a different pset, maximizing the I/O bandwidth. For the compute-image step, the master stores a partial image corresponding to the local y in individual files. This method reduces the overhead of file-descriptor management and the impact of the General Parallel File System* (GPFS*) block. The reconstitution of the global file can be done simultaneously by a program running on any front-end node.

It is important to note that in order to achieve optimal performance using this implementation, the I/O configuration must be optimized as well. In the BG/L system, this implementation requires a suitable file system and configuration of the appropriate number of compute nodes per I/O node.

Results

To validate this massively parallel implementation of the CAM, and to demonstrate its efficiency, we have utilized up to eight BG/L racks (corresponding to 16,368 processors) installed at the IBM T. J. Watson Research Center. These results were obtained in collaboration with Total, a French oil company, using an actual production seismic model on a POWER5* 16-way 1.5-GHz cluster at the E&P (Exploration and Production) division of Total. The dimension of the dataset was 47 km × 33.6 km (~1,580 km^2), with 64 offsets and approximately 400 GB of input data. The simulation was realized with 264 frequencies, from 1,500 m down to 10,000 m in depth, by 10-m steps. The physical grid size of 1,881 × 1,346 × 64

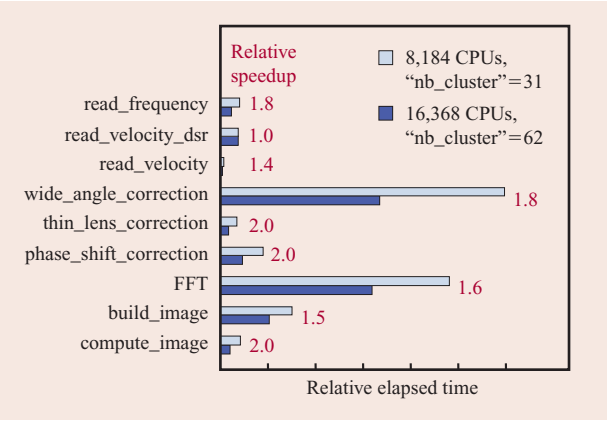

Figure 5

Time distribution for the test case on 8,184 and 16,368 Blue Gene/L processors. The label "nb_cluster" indicates the number of cores (processors) per frequency group.

was modified to $2,304 \times 1,684 \times 64$ (the dimension of the wave field arrays) to fit ESSL FFT requirements.

Figure 5 presents the elapsed time for the main parts of the CAM algorithm on 8,184 CPUs (four BG/L racks, $nb\_cluster = 31$) and 16,368 CPUs (eight BG/L racks, $nb\_cluster = 62$), as well as the relative speedup between the two runs. The global efficiency is greater than 0.8. The figure shows that more than 60% of the overall time is spent in the FFT and the wide-angle correction.

We also notice that efficiency depends mainly on the FFT, the overhead of the velocity correction (handled in a step called *read_velocity_dsr*), and the global reduction in the build image phase. The phase shift and thin lens corrections functions are linear up to 16,368 processors. It is important to note that we used the homogeneous version of the FFT; the nonhomogeneous version will be able to reduce the overall FFT part by about 20% while providing better scalability.

The performance and parallelism of the I/O steps are rather good. The velocity initialization at each depth has two steps: the *read_velocity* step consists of a parallel read of the velocity files, and the *read_velocity_dsr* step, which is not parallel and interpolates the raw velocity data to match the domain grid points. The velocity reads can be improved by taking advantage of the free memory and the performance of the BG/L network in order to map the velocity file in memory and replace the standard reads by parallel communications. Nevertheless, the most significant part corresponds to the *compute-image* phase, which requires a global reduction within the *hy* direction. As we explained in the previous paragraph, the I/O was optimized depending only on the number of nodes in a pset. The BG/L system installed at the IBM T. J. Watson Center has one I/O node for 64 compute nodes (the optimal configuration is eight compute nodes per I/O node). Thus, a configuration with more I/O nodes will significantly reduce the I/O time.

On the basis of the results obtained on the Watson BG/L system, and using the same version (e.g., code, implementation, or executable), we can estimate a global efficiency of 0.75 on 32,736 processors (16 BG/L racks).

Summary

This paper confirms that massively parallel architectures such as the IBM BG/L system are well suited to solving wave-equation algorithms and are capable of providing the computing power necessary for seismic imaging while also maintaining reasonable infrastructure costs in terms of square meters of floor space, electricity consumption, RAS (reliability, availability, and serviceability), and administration.

By introducing several levels of parallelism, we have implemented a very efficient version of the CAM method, which scales to tens of thousands of processors and fits the memory footprint per processor, with an overall efficiency of greater than 0.8. The I/O capabilities of the architecture of the BG/L system, coupled with the GPFS, are sufficiently large to manage the steadily increasing data requirements of the PSDM algorithms over the coming years. Despite the 32-bit nature of the seismic implementations (single-floating-point precision), the BG/L ASIC processor achieves excellent performance results, especially because the SIMD FPU is well suited to complex data types.

Similar techniques have already been applied to the shot point algorithm (i.e., shot profile migration) with similar and even better scalability, in which the first level of parallelism over the frequency in the CAM algorithm is replaced by a distribution of the shot points to groups of processors. Within a group of processors, we have introduced a spatial decomposition in order to reduce the memory footprint per processor. The number of shot points is much greater than the number of frequencies, which suggests excellent scalability to extremely large numbers of processors.

Our current objective is to take advantage of the memory size available on the massively parallel systems or to explore global array capabilities, which could be used to implement algorithms such as the Kirchhoff algorithm, a well-known approach currently used by many oil companies. We also aim to study the reverse-time-migration algorithm, which should be particularly suitable to the 3D torus topology of the BG/L system.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of the OpenMP Architecture Review Board in the United States, other countries, or both.

References

1. J. F. Claerbout, *Imaging the Earth's Interior*, Palo Alto, CA, Blackwell Scientific Publications, 1985.
2. B. Biondi and G. Palacharla, "3-D Prestack Migration of Common-Azimuth Data," *Geophysics* **61**, No 6, 1822–1832 (1996).
3. D. Ristow and T. Ruhl, "Fourier Finite-Difference Migration," *Geophysics* **59**, No. 12, 1882–1893 (1994).
4. M. Eleftheriou, B. G. Fitch, A. Rayshubskiy, T. J. C. Ward, and R. S. Germain, "Scalable Framework for 3D FFTs on the Blue Gene/L Supercomputer: Implementation and Early Performance Measurements," *IBM J. Res. & Dev.* **49**, No. 2/3, 457–464 (2005).
5. O. Lascu, N. Allsopp, P. Vezolle, J. Follows, M. Hennecke, F. Ishibashi, and M. Paolini, et al., "Unfolding the IBM eServer Blue Gene Solution," *IBM Redbooks*, September 2005; see *http://www.redbooks.ibm.com/abstracts/sg246686.html*.
6. S. Chatterjee, L. R. Bachega, P. Bergner, K. A. Dockser, J. A. Gunnels, M. Gupta, F. G. Gustavson, et al., "Design and Exploitation of a High-Performance SIMD Floating-Point Unit for Blue Gene/L," *IBM J. Res. & Dev.* **49**, No. 2/3, 377–391 (2005).
7. G. L. Mullen-Schultz and C. Sosa, "IBM System Blue Gene Solution: Application Development," *IBM Redbooks*, June 2007; see *http://www.redbooks.ibm.com/abstracts/sg247179.html*.

Received March 19, 2007; accepted for publication April 15, 2007; Internet publication December 18, 2007

Henri Calandra *Total Exploration and Production Division, Avenue Larribau, 6400 Pau, France (henri.calandra@total.com)*.

In 1984, Dr. Calandra received a Ph.D. degree in mathematics from the University des Pays de l'Adour, France. He joined Cray Research France in 1987 and worked on seismic applications for 2 years. He joined the Applied Mathematics Department of the French Atomic Agency in 1989. In 1990, he started to work for Total. After 12 years of work in high-performance computing and as project leader for pre-stack depth migration research, he became head of the Total USA Geophysics Research Group and now coordinates depth imaging research for the worldwide group.

François Bothorel *IBM France, 2 Avenue Gambetta, 92400 Courbevoie, France (francois.bothorel@fr.ibm.com)*.

In 1982, Mr. Bothorel received an M.Sc. degree in mathematics from the University Paris XI, Orsay, France. He joined Control Data Corporation in 1983 where he worked in a service bureau for 3 years. He joined Gould Computer Systems in 1986 and worked on real-time applications and participated in developing an FFT package for vector computers. Mr. Bothorel worked for Digital Equipment Corporation as a specialist on real-time and scientific applications. In 1999, he joined IBM and is the French Lead Architect for the high-performance computing market. He is an IBM IT Certified Specialist and part of the Deep Computing organization.

Pascal Vezolle *IBM France, Rue de la Vielle Poste, BP 1021, F-34006 Montpellier, France (vezolle@fr.ibm.com)*.

In 1993, Dr. Vezolle received a Ph.D. degree in applied mathematics from the University of Bordeaux. From 1993 to 1995, he worked on distributed parallel computers for the French Atomic Agency. He joined Saint Gobain Research in 1996 as a computing engineer. He joined IBM in 2001. He is an IBM IT-certified specialist and part of Deep Computing organization and Blue Gene* worldwide team.