RC25313 (WAT1210-019) October 3, 2012
Computer Science

IBM Research Report

Optimization of Reverse Time Migration For Fast and Scalable Seismic Imaging on Blue Gene

Ligang Lu, Lurng-Kuo Liu, Karen Magerlein,
Pascal Vezolle*, Michael Perrone
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 208
Yorktown Heights, NY 10598
USA

*IBM France

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Optimization of Reverse Time Migration
For Fast and Scalable Seismic Imaging on Blue Gene

Ligang Lu, Lurng-Kuo Liu, Karen Magerlein, Pascal Vezolle*, Michael Perrone
Computational Science Center, IBM Research
Yorktown Heights, NY 10598
*IBM France

***Abstract*—We present our optimization work on a Reverse Time Migration (RTM) code for fast and scalable seismic imaging on Blue Gene. RTM is a seismic imaging algorithm increasingly used in industry for oil exploration. Our work is novel not only in that it uses the Blue Gene supercomputing systems torus network, high inter-node communication bandwidth, and large resident memory in each node to convert an embarrassingly parallel problem into one that can be efficiently solved using massive domain partitioning; but it is also novel in that it develops effective schemes to keep intermediate data snapshots in internal memory and fast interpolation for sub-sampling that eliminate bottlenecks in the forward and backward pass and significantly improved performance while maintained image quality. With further optimizations in boundary condition calculation, OpenMP, load balance, MPI communications, etc., our final code achieved ~10X overall performance improvement over the base line code. The success of our optimization can be easily extended to next-generation imaging algorithms currently being developed. In the workshop, we will include a BG/Q architecture overview and comparative BG/Q results depending on availability.**

Keywords- Seismic Imaging, Reverse Time Migration Optimization;Finite Difference Methods, Data Movement

I. INTRODUCTION

Reverse Time Migration (RTM) is increasingly used in oil industry for petroleum exploration. Because the drilling success rate is vital to the oil companies, complex variations of algorithms are being developed and used. To reduce the time to oil, high performance computers are widely employed. To be cost effective, optimal performance is critical to oil companies. In this paper, we present our optimization work on a Reverse Time Migration (RTM) code for fast and scalable seismic imaging on Blue Gene. RTM is a seismic imaging algorithm increasingly used in industry for oil exploration. Our work is novel in that it not only took the advantages of the Blue Gene supercomputing systems in 3D torus network, high inter-node communication bandwidth, and large resident memory in each node to convert an embarrassingly parallel problem into one that can be efficiently solved using massive domain partitioning, but also developed effective schemes to keep the wavefield snapshots in internal memory and fast interpolation for sub-sampling that eliminated the bottlenecks in the forward pass and backward pass and significantly improved performance while maintained the image quality. With further optimizations in boundary condition calculation, OpenMP, load balance, MPI communications, etc, our final code achieved ~10X overall performance improvement over the base line code. The success of our optimization can be easily extended to next-generation imaging algorithms currently being developed. In this paper we will also include a BG/Q architecture overview and comparative results on BG/P. In this paper, we explore these ideas in the context of production seismic imaging: physics-based signal processing used by the energy industry to find oil and gas reservoirs. We begin with an introduction to the problem of seismic imaging and a description of the widely used Reverse Time Migration (RTM) method, comparing and contrasting our approach to the one commonly used in the industry. We then describe our implementation of RTM, the optimizations that were performed, the experimental setup and the performance results, comparing where appropriate to other RTM implementations.

II. SEISMIC IMAGING

Seismic imaging is the process of converting acoustic measurements of the Earth into images of the Earth's interior, much like ultrasound for medical imaging. It is widely used in oil and gas exploration and production to identify regions that are likely to contain hydrocarbon reservoirs and to help characterize known reservoirs to maximize production. These methods have become critical to the energy industry as known reserves are used up and new reserves become increasingly difficult (and expensive) to find and are increasingly in technically challenging areas, like the deep sea.

For the past several decades, the energy industry has tried to balance the need to image quickly and the need to image accurately. The need for accuracy is driven by the high cost of drilling a "dry" well due to poor imaging (a deep sea well can cost over $100 million) and the need for quick imaging is driven by the cost of not finding new reserves (i.e., bankruptcy). To minimize these costs, the industry relies on supercomputing clusters and regularly increases compute power, enabling both faster imaging on existing algorithms and the practical implementation of more accurate imaging. Thus, the development of fast, efficient methods for imaging is of high importance to the industry.

A. *Seismic Data*

Seismic imaging data varies widely depending on how and where the data is collected (e.g., on land, at sea, at the ocean surface, at the ocean floor, below ground, electromagnetically, etc). We focus here on the data collection method that is most relevant to the RTM algorithm analyzed in this paper: towed hydrophone receiver arrays for ocean seismic data collection. The basic idea is shown in Figure 1. A ship is shown towing a 2D array of hydrophones spaced about every 25m on 1 to 16 trailed streamers. Every 15 or so seconds, an air cannon is fired into the water, creating an acoustic wave that propagates through the water and into the Earth. Reflections from various surface and subsurface boundaries cause echoes that reflect back and are recorded by each hydrophone in the array. The recording of a single hydrophone in time as a trace and the collection of traces for a single firing of the air cannon is called a **common shot gather**, or **shot**. As the ship moves, a large set of spatially overlapping shots is recorded. Depending on the size of the survey region to be imaged, this data collection can take a month or more and is designed to get the maximal coverage of the area to be imaged. For our purposes, we need to know that we have lots of shots, potentially hundreds of thousands, and that the receiver data collected is the result of some source data at a particular location. A sample of artificial shot data is shown in Figure 2.

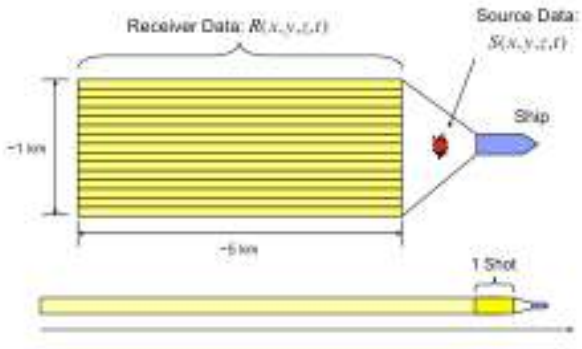

Figure 1: A ship collecting seismic data using a towed hydrophone receiver array

B. *The RTM Algorithm*

The Reverse Time Migration (RTM) algorithm is widely used in the industry because of its superior imaging accuracy for difficult subsurface structures like salt domes which are poorly imaged by other algorithms but which are very effective at trapping oil and gas. Several variants of RTM exist with differing degrees of approximation to reality, all of which use single-precision arithmetic. For this paper we implemented isotropic, acoustic RTM which assumes the wave velocity is independent of wave direction and that no energy is absorbed by the medium.

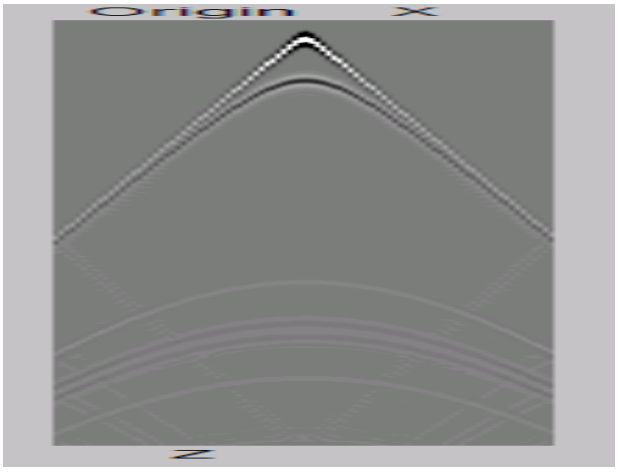

Figure 2: Sample shot data for a 1D array of hydrophones showing time on the Y-axis and spatial offset on the X-axis. The direct source signal propagates out linearly in time (from the center of the array) and appears as straight lines. The recorded reflections appear as curved lines.

The RTM algorithm arises from the observation that pressure waves should be correlated at reflection boundaries; so RTM proceeds by correlating two pressure waves (called the forward and backward waves) to find those boundaries. To generate the waves for correlation, RTM simulates wave propagation using the wave equation below for a wave U(x,y,z,t) with a source term S(x,y,z,t):

$$(1) \qquad \frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} + S$$

The **forward wave** is the wave generated from the air cannon firing and propagating forward in time using a "**velocity model**" represented by C(x,y,z), which specifies the wave velocity at each point in space and represents the various material properties and boundaries of the volume being imaged. The air cannon firing is treated as a wavelet impulse localized in time and space. The **backward wave** is generated by using the shot data recorded by the hydrophone array as the source term for the wave equation and propagating that backward in time. These two waves are then multiplied point-wise at each time step to generate an image, using the following "**imaging condition**":

$$(2) \quad I(x,y,z) = \sum_t U_{Forward}(x,y,z,t) U_{Backward}(x,y,z,t)$$

This process is repeated for all shots in the seismic survey and the images generated are summed to create a final image of the reflecting boundaries, which represent the subsurface structure. It is important to note that the time

summation in the imaging condition implies that the first time step of the forward wave needs to be correlated with the last time step of the backward wave. This constraint is typically handled in one of two ways: either the forward wave is saved to disk (called a "**snapshot**") every several time steps and read in for imaging when the backward wave is computed, or the forward propagation is run twice – once forward in time and once in reverse time using boundary data saved from the forward pass to recreate the forward pass in reverse – and then imaging proceeds with the backward wave and the reverse forward wave. The first method requires significant disk storage and can be bottlenecked on disk I/O, while the second requires 50% more computation and additional memory space to save the boundary data.

Following standard practice in the industry [2], we simulate the wave propagation of Equation (1) using the finite difference approximation in Equation (3) where we select the coefficients to implement 2nd order accuracy in time and 8th order accuracy in space. These coefficients are scaled to satisfy the CFL condition [5]. This approach gives rise to the 25-point stencil shown in Figure 3.

$$U_{i,j,k,t+1} = 2U_{i,j,k,t} - U_{i,j,k,t-1} + c^2_{i,j,k} \sum_{n=-4}^{n=4} (\alpha_n U_{i+n,j,k,t} + \beta_n U_{i,j+n,k,t} + \delta_n U_{i,j,k+n,t}) \quad (3)$$

In practice, the size of production RTM models varies widely, but the universal desire is to grow models larger to get more resolution and to run the models longer to enable deeper imaging since echoes take longer to reflect from deeper structures. Typically, velocity models for individual shots are 512^3 to 1024^3 elements or larger and the number of time steps can be 10,000 or more in both the forward and backward propagation phases.

Seismic imaging is typically computed using single precision arithmetic and we take that approach here. Some practitioners believe that the RTM method described above, that avoids the need to save snapshots, must be run in double precision; however, we do not implement that version here.

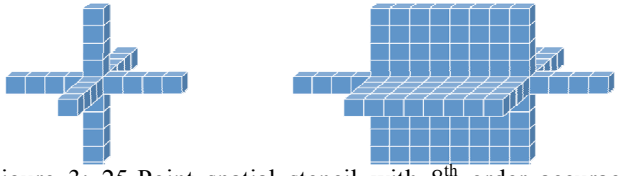

Figure 3: 25-Point spatial stencil with 8th order accuracy shown in isolation on the left and as it moves along the stride-1 dimension of the model

C. *Embarrassisngly Parallel RTM*

Industrial implementations of RTM are embarrassingly parallel. They typically run individual shots on one to two nodes of a compute cluster and run many shots in parallel (See Figure 4). These clusters have minimal network connectivity because it is not needed: the individual shots run independently and asynchronously. A simple work queue is used to manage runs and if a run for a shot fails, it is simply re-run, as it doesn't impact any of the other runs. A master process of some kind is needed to manage the work queue and to merge the partial images that are generated from each shot. Additionally, other image processing might be included in this process, but for our purposes here we ignore these details as the RTM calculation is the main computational workload.

RTM compute clusters have significant per-node scratch disk requirements for saving snapshot data, which for a 1024^3 model and 10,000 time steps would require 40TB of snapshot storage – per shot! In practice, snapshot sub-sampling is used to reduce both disk requirements and disk IO bottlenecks; however sub-sampling results in image degradation and must be balanced with performance. Compression can be used to trade computation for disk IO, but if lossy, compression can also degrade image quality.

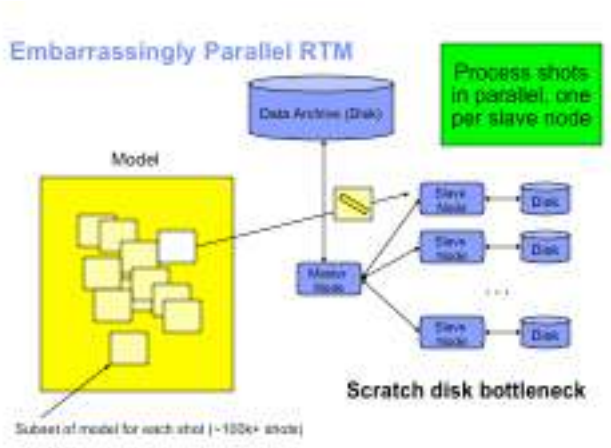

Figure 4: Embarrassingly parallel RTM implementation

As FLOPS per processor increase, the embarrassingly parallel implementations become disk IO bound [2]. It is possible to improve performance by partitioning single shot gathers over multiple nodes; however, such implementations typically use only a handful of nodes. We have developed an RTM implementation that extends domain partitioning over thousands of nodes on a Blue Gene/P supercomputer and results in dramatic performance improvements.

D. *Domain-Partitioned RTM*

We have developed, implemented and tested a 3D isotropic RTM code that uniformly partitions the wave equation domain in blocks over thousands of nodes (see Figure 5). The partitioning over so many nodes means that the size of

the model on each node is about 1000 times smaller than for standard RTM on a handful of nodes and provides five main benefits: (1) all forward wave snapshots can be stored in main memory, removing the need for disk and thereby improving per-node data bandwidth from hundreds of MB/s (for disk) to tens of GB/s (for main memory), effectively removing the disk I/O performance bottleneck; (2) the partitioned models can fit in processor cache, allowing processing to proceed at the speed of the cache memory bandwidth instead of main memory bandwidth which can be an order of magnitude larger on some systems; (3) we can load entire 3D seismic surveys into memory on one or more racks, enabling "in-memory" processing of algorithms that require multiple passes through the data set, such as Full Waveform Inversion (FWI), and thereby avoiding additional disk I/O bottlenecks and enabling better performance; (4) keeping the models in the processor's cache means that snapshot reading/writing has full access to main memory bandwidth and is not a bottleneck; and (5) this method can run an entire velocity model instead of a subset, as is typically done in standard RTM, allowing us to easily extend this method to include multisource processing with minimal code changes and significant potential performance gains [4].

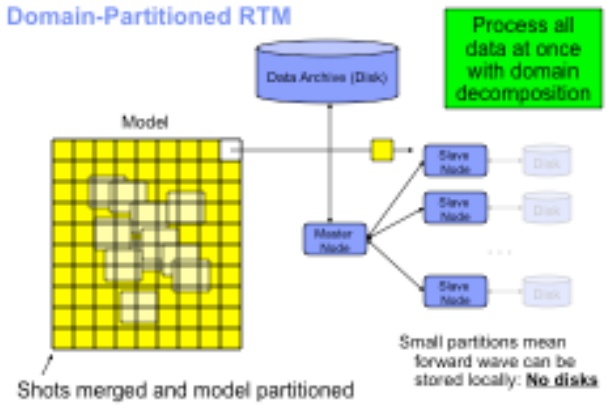

Figure 5: Domain-partitioned RTM Implementation

A critical aspect of domain-partitioned RTM is that current wave data from neighboring sub-domain boundaries is required for stencil calculations at each time step. Since this boundary data transfer grows with the amount of partitioning and with the size of the stencil used, it can easily become a performance bottleneck. To avoid communication bottlenecks, we implemented our partitioned RTM on a Blue Gene/P supercomputer, which is designed specifically for extremely efficient inter-node communication.

III. BELUE GENE ARCHITECTURE

Our performance measurements were all performed on subsets of two racks of Blue Gene/P. It is beyond the scope of this paper to give a full description of these machines. Instead we focus on those features that are relevant. More details can be found elsewhere [1].

The Blue Gene/P (**BGP**) supercomputer has 1024 nodes per rack running at 0.85GHz. Each node has 4 single-threaded cores, 4GB of RAM per node (4TB per rack) and an extremely high-bandwidth, low-latency, nearest-neighbor 3D torus topology network in which each node is connected to each of its 6 nearest neighbor nodes by 850MB/s of send+receive bandwidth (i.e., 5.1GB/s per node and 5.22TB/s of communication bandwidth per rack). Because of this massive bandwidth, BGP is ideally suited for physical modeling involving extensive nearest-neighbor communication and synchronization – like RTM. The nearest neighbor latency for 32B data transfers is about 0.1 microseconds and is essentially amortized away for larger block transfers required by RTM. Each compute node core has a 32KB L1 cache with a 32B cache line and a shared 8MB L3 cache with a 128B cache line. Each node has two memory channels with an aggregate bandwidth of 13.6 GB/sec to main memory. BGP compute notes are connected via dedicated I/O nodes to a GPFS file system based on three DDN S2A9900 couplets attached to the BGP I/O nodes via 10 Gigabit Ethernet connections, providing ~16GB/s of disk I/O bandwidth per rack. Each node can operate in **SMP** mode as a unit, or as four "virtual" nodes. The Virtual Node (**VN**) model avoids the need to explicitly use multithreading at the node level and thereby eases programmability. Each core has a 2-way SIMD unit.

IV. RTM OPTIMIZATION DETAILS

In this section we describe various optimization details on Blue Gene that were important to our RTM performance improvement.

A. Code Flow

First we describe the RTM's code flow as show in the figure below. The code has two main loops: **the forward pass loop** and **the backward pass loop.** In the forward pass loop, after the step of wavefield computation, the forward wavefield will be sub-sampled and compressed and saved to the disk at a predetermined frequency. In the backward pass loop, after the step of the wavefield computation, the backward wavefield will be sub-sampled and the saved forward wavefield will be fetched in and decompressed to do imaging condition with the backward wavefield at the same frequency.

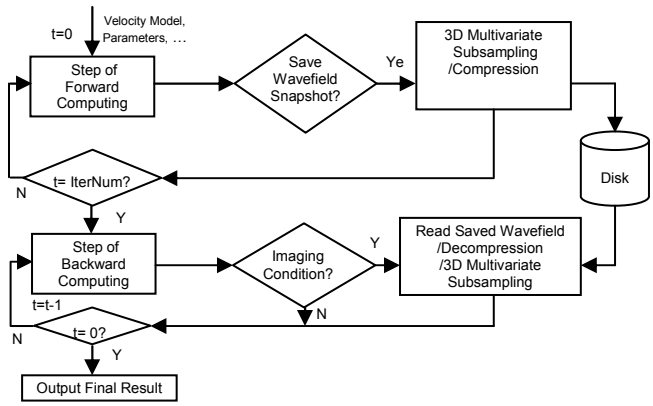

Figure 6: RTM Code Flow

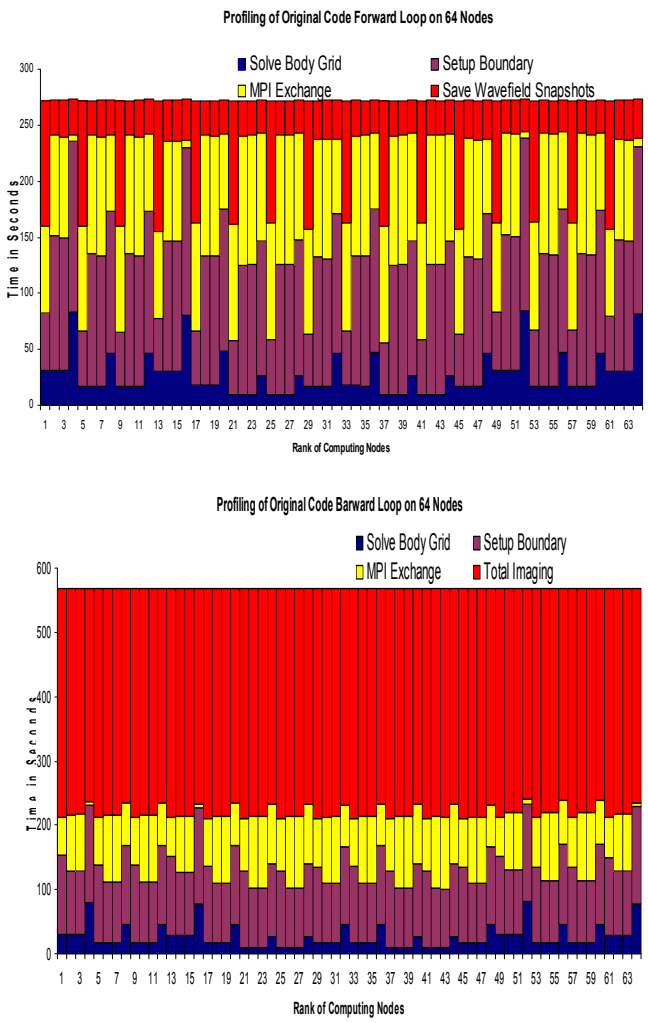

Figure 7: Profiling of Base line code forward pass (top) and backward pass (bottom) results (in seconds) on 64 nodes

B. *Base Line Code Performance Analysis*

Upon porting the base line code to Blue Gene P system, we first conducted scaling performance profiling and analysis to identify performance bottlenecks and guide our optimization process. A high-level breakdown of the base line code (un-optimized) run time for the forward and backward passes is shown in the figures below. The results suggest that source field writing, boundary conditions and MPI are not scaling well. Clearly the base line code does not scale well and needs to be optimized.

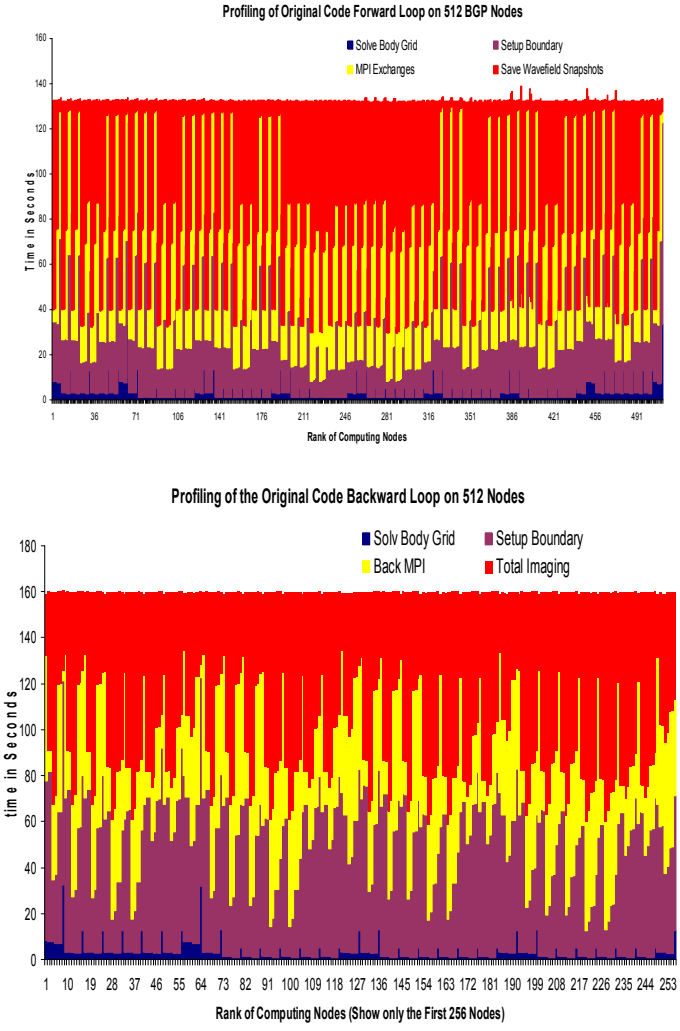

Figure 8: Profiling of Base line code forward pass (top) and backward pass (bottom) results (in seconds) on 512 nodes

Further analysis based on MPI traces shows that the four bottom corner nodes took a significant amount of time and thus their MPI became the late arrival senders in the subsequent MPI communication, causing other nodes to wait in the MPI sendrecv.

C. *Keeping Wavefield Snapshots in Memory*

From the profiling analysis, it is clear that the writing the wavefield snapshots to the disk consumes a significant amount of the time in forward pass (the red coloured part in the forward pass in Figure 7 and Figure 8). Further more it clearly becomes the bottleneck of the forward pass when the number of computing nodes gets large.

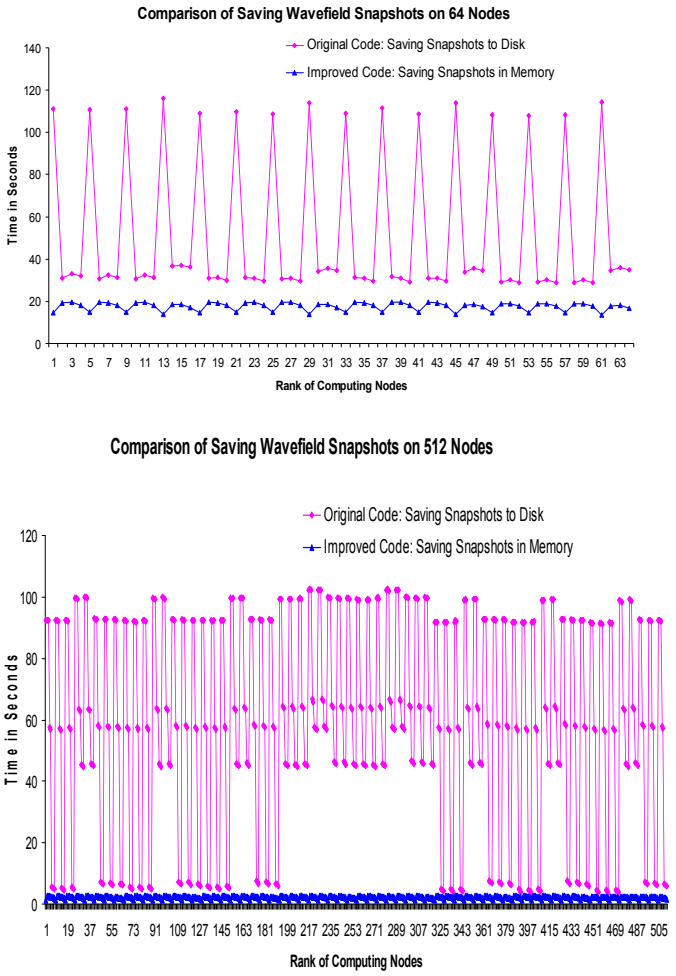

Figure 9: Performance improvements after keeping the wavefield snapshots in memory on 64 nodes (top) and on 512 nodes (bottom)

Taking the advantages of Blue Gene/P 3D torus network, large in-node memory, and fast inter-node communication, in addition to use the domain–partition parallel computing approach, we further developed a scheme to keep the wavefield snapshots in memory so that the disk IO and compression and decompression operations can be reduced or even completely avoided. The following figures show the performance improvements after keeping the wavefield snapshots in memory.

D. *Fast and Effective Subsampling*

From the profiling analysis of backward pass, we also know that the total imaging part is a bottleneck. With further investigation, we found that the largest computation stems from the 3D multivariate interpolations used for sub-sampling. There are three 3D multivariate interpolations for every image point; moreover, each 3D multivariate interpolation requires 22 multiplications, and 14 additions.

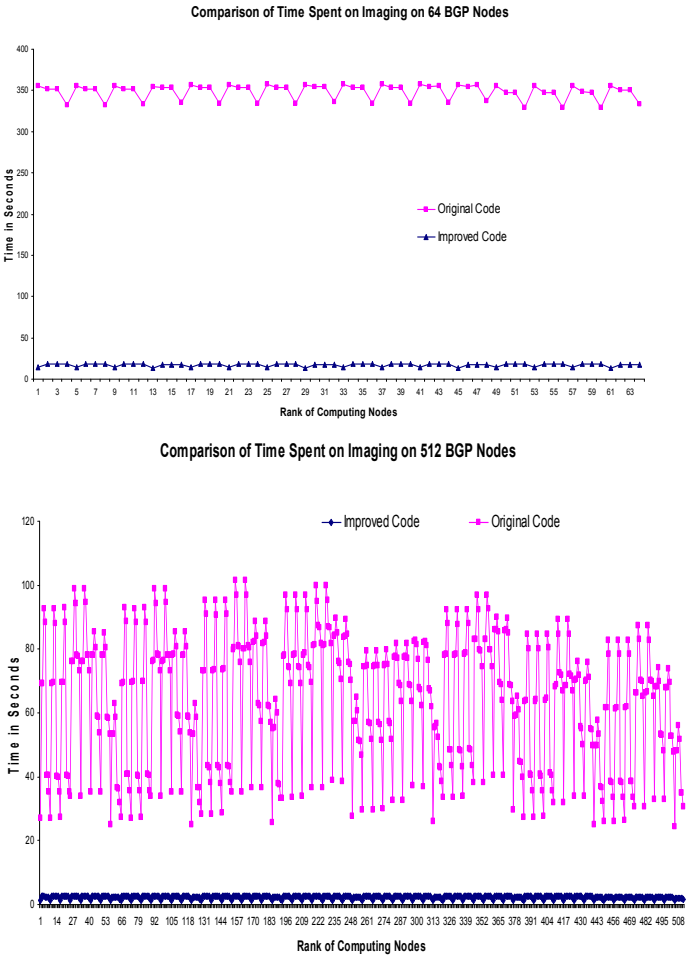

Figure 10: Performance improvements after using the fast interpolation method on 64 nodes (top) and on 512 nodes (bottom)

To improve the performance while maintaining the image quality, we developed a fast linear interpolation method to replace the 3D multivariate interpolation. In Figure 10, we show the improvements using the new interpolation as the

sub-sampling method. Now each image point only requires one multiplication and 7 additions. We compare the final image outputs from the two interpolation methods and the error statistics in Table 1 show that there is virtually no impact to the image quality.

Table 1 Error Statistics between the Images from the two Interpolation Methods

Min	0.000000e-00
Max	4.135992e-06
Mean	2.188777e-09
MSE	1.522148e-15
Std Div.	3.895328e-08

E. Final Performance

We also conducted optimizations in boundary condition calculation, OpenMP, load balance, MPI communications, etc. The final code achieved up to ~10X performance improvement over the base line code as shown in Figure 11.

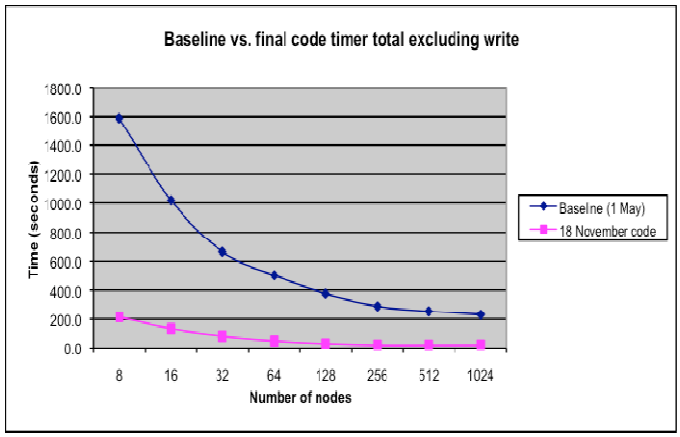

Figure 11: Overall performance improvement of the final code over the base line code

V. CONCLUSIONS

In this paper we have presented our new results on optimizing a RTM base code on Blue Gene/P. We took the advantages of Blue Gene in 3D torus network, inter-node communication bandwidth, and large memory in each node to use the domain partition approach. We further developed a scheme to keep the wavefield snapshots in the memory to eliminate the disk IO bottleneck and saved the compression/decompression time. We also developed a fast interpolation method to replace the time consuming 3D multivariate interpolation for wavefield sub-sampling process and achieved very significant performance improvement. In addition, we also conducted optimizations in boundary condition calculation, OpenMP, load balance, MPI communication, etc. The final code achieved ~10X performance improvement over the base line code on BG/P. Significantly better results on BG/Q will be presented at the workshop.

ACKNOWLEDGEMENTS

The Blue Gene/P project has been supported and partially funded by Argonne National Laboratory and the Lawrence Livermore National Laboratory on behalf of the U.S. Department of Energy under Lawrence Livermore National Laboratory subcontract no. B554331. We also acknowledge the support and collaboration of Columbia University and Edinburgh University.

REFERENCES

[1] Sosa, C. and Knudson, B. 2009. *IBM System Blue Gene Solution: Blue Gene/P Application Development, IBM Redbooks*, DOI= http://www.redbooks.ibm.com/abstracts/sg247287.html

[2] Zhou, H., Fossum, G., Todd, R. and Perrone, M. 2010. Practical VTI RTM. In *Proceedings of 72nd EAGE Conference*.

[3] Higdon, R. L. 1987. Numerical Absorbing Boundary Conditions for the Wave Equation. *Mathematics of Computation*, Vol. 49:179 July,1987, pps. 65-90.

[4] Boonyasiriwat, B. and Schuster, G. 2010. 3D Multisource Full-Waveform Inversion using Dynamic Quasi-Monte Carlo Phase Encoding. *Geophysical Research Abstracts,* Vol. 12, EGU2010-7298, 20.

[5] Trefethen, L. and Bau, D. 1997. *Numerical Linear Algebra*, SIAM.

[6] Perrone, M., Liu, L., Lu, L. and Magerlein, K. 2010. High Performance RTM Using Massive Domain Partitioning, In Proceedings of *EAGE'2011 Conference*, May, 2010.

[7] Abdelkhalek, R., Calandra, H., Coulaud, O., Roman, J., Latu, G. 2009. Fast Seismic Modeling and Reverse Time Migration on a GPU Cluster. In *International Conference on High Performance Computing & Simulation*, 2009. HPCS'09.

[8] Fletcher, R. P., Du, X. and Fowler, P. J. 2009. Reverse time migration in tilted transversely isotropic (TTI) media. *Geophysics,* Vol 74:6, 2009.

[9] Micikevicius, P. 2009. 3D finite difference computation on GPUs using CUDA. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, Washington, D.C., 79-84.

[10] Okamoto, T., Takenaka, H., Nakamura, T. and Aoki, T. 2009. Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition. In *Earth Planets Space*, November, 2010.