Code Porting to Cray XC40
Lessons Learned

Jim McClean and Raj Gautam

jim.mcclean@pgs.com

raj.gautam@pgs.com

Abstract— **We present a case study of porting seismic applications from the Beowulf cluster using Ethernet networking to the Cray XC40 cluster with Aries networking. The applications in question are Tilted Transverse Anisotropic Reverse Time Migration (TTI RTM), Kirchhoff Depth Migration (KDMIG) and Wave equation migration (WEM). The primary obstacle in this port was that TTI RTM and WEM used local scratch disk heavily, and imaging is performed one shot per node as in map-reduce applications. The Cray nodes do not have local scratch disks and depend on the remote file system for storage. The primary obstacle in KDMIG was its heavy IO usage from permanent disk due to the constant reading of Travel Time Maps (TT). We briefly explain how these algorithms were refactored to be less dependent on scratch disk and to fully utilize the advanced networking in the Cray XC40. In the case of KDMIG, we explain how its IO load was reduced via a memory pool concept. We also provide some details concerning the management of striping issues on Lustre file systems and other IO load management issues.**

I. Introduction

Years ago when PGS first started as a seismic acquisition and processing company, its first large computer was based on the Intel 860 chip. This was considered a large distributed computer in its day, similar in concept to the CM2 or CM5. Like most companies, PGS switched to the Beowulf concept in the early 1990's. This switch was due to the superior performance relative to cost characteristics of the Beowulf cluster compared to monolithic computers of the day.

Today, we see evidence of a trend toward more monolithic computer architecture in order to achieve the integration and reliability needed to achieve Exascale computing. To solve new physics challenges, the HPC community is seeing increased usage of fiber optics and various other high-speed networking in order to achieve the sustained performance needed across nodes. . This paper relates our experiences in trying to switch from an embarrassingly parallel paradigm to a more distributed computing paradigm.

II. Background

To acquire seismic data, PGS tows source and receiver arrays behind marine seismic vessels. The receiver array consists of many sensors and acts as a giant antenna receiving the reflected sounds from the sub surface. These reflected sound waves are received and recorded on the surface boundary. The HPC challenge is to project these sound waves recorded on the surface boundary back into a simulated mathematical earth model using the wave equation in order to form a useful image of the earth's sub surface.

PGS employs three common techniques to do this: Kirchhoff depth imaging, one-way wave equation, and reverse time migration. Industry practice is to use an acoustic and not an elastic approximation to this physics problem. This is largely due to the fact that the marine environment does not support shear waves but only compressional sound waves. Also the elastic approximation that might be used in the onshore case is more computationally expensive.

In the marine environment, current industry practice is to perform a one-shot experiment every 25 meters or so. A single shot consists of one discharge of the compressed air gun source array, and the collection of the reflected subsurface data upon the receiver antenna. In a large marine survey that is 100 kilometers squared in size, up to 1.6 million shot experiments can be recorded, each of which can be up to ½ gigabyte of captured shot experiment data. The total size of the collected data can run into the petabytes for a large survey. The outputs from imaging this input data are also of a similar size.

Past industry practice was to treat or image each shot separately using one of the accepted algorithms mentioned. This allowed us to use an embarrassingly parallel paradigm for the most part, making it possible to compute the image result using one node per shot experiment. However this becomes challenging as the frequency increases, both from a main memory and compute cycle point of view. Also the industry is evolving to use continuous shooting which causes cross talk in the received results, thereby making the embarrassingly parallel paradigm more difficult to use in practice. We will discuss in order our experiences in restructuring reverse time migration, Kirchhoff, and one-way wave equation to the XC40.

III. Reverse Time migration

Reverse Time Migration (RTM) is based on the idea of sending the recorded sound energy back into the earth model, using the wave equation from the surface boundary, and correlating this energy with a synthetically generated forward source model. The computation cost of RTM scales as the 4th power of the maximum frequency imaged. The main memory requirements of RTM scales as the 3rd power of the maximum frequency imaged. This is due to the wave equation's three spatial dimensions and one time dimension, and the relationship to the size of the computational bin/cell

bin_size ~ Vmin / (Fmax * points_per_wavelength)   [1]

Bin_size decreases as Fmax increases, and also the time stepping are inversely proportional to the maximum frequency to be imaged. The implementation of TTI RTM we are using is a spectral domain implementation and requires the use of FFTs to perform differentiation.

When we started our investigation into porting RTM from a Beowulf cluster to the Cray XC30, we started from the somewhat naive view of simply recompiling the code for the Cray and check its performance. At the time we had access to a two cabinet XC30 cluster. The program ran successfully using 9 nodes and produced a valid result. We then proceeded to try to scale up to 100 nodes and discovered that a job with 100 nodes ran much slower than the corresponding job on a Beowulf cluster.

We determined the slowness was due to IO on the Lustre disk. Our initial starting point was the embarrassingly parallel paradigm where one shot is imaged in one node, and we were saving the forward modeled wave field to scratch disk. The Cray XC does not have scratch disk on each node, so the main Lustre file system had to be used instead. So jobs running on hundreds of nodes, each of which is saving the forward modeled wave fields to Lustre disk, did not scale well.

Making note of the fact that in our implementation of pseudo analytic TTI RTM, most of the fourth derivatives just depend on two space axes only. Only 3 of the fourth derivatives depend on all three space axes. So most of the derivatives just require the use of 2D FFTs only and not 3D FFTs. This caused us to use the concept of slabs in this algorithm.

Having seen the disk I/O scalability issue, we were also up against the Cray's 128 gigabyte-per-node memory limitation in some of our business projects, so we decided to restructure the algorithm. Since most of the FFTs in use were 2D FFTs, the pressure wave field could be divided into slabs, and each slab distributed to different nodes. Further, since we did not want to put undue stress on the Lustre file system, we decided to keep all the forward extrapolated wave fields in memory. In the collection of nodes processing a single shot, each node would keep its corresponding pressure slabs. The same is true for the forward extrapolated model and also the output image – only slabs appropriate for that node are kept in that node's memory. Using this technique, we had access to terabytes of memory to process a single shot experiment.

The only serious problem with this approach was the 3 or 4 of the 3D FFTs required to perform each time step of the algorithm. We tried a distributed 3D cluster-based FFT that used MPI but found it was too slow for this purpose. So in collaboration with Cray, we developed a 3D cluster-based FFT based on the shared memory protocol. Since SHMEM is a one-sided protocol, not a two-sided protocol like MPI, this SHMEM FFT turned out to be faster than the normal 3D FFT based on MPI.

Once this framework was in place, we discovered that the new algorithm in Cray was scaling better than N, the number of nodes used in older code running in Beowulf cluster. It means that compared with one node per shot, the N nodes per shot in the new algorithm run faster than 1 / N time. The speed increase was because we were not storing the forward wave fields on scratch disk.

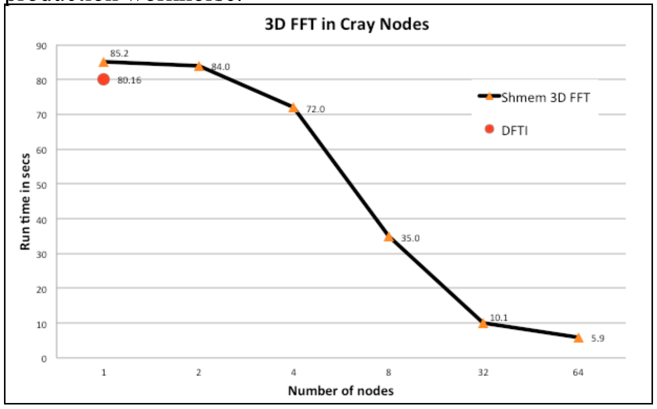
Additionally, the old implementation used numerous master nodes to contain the output image for the job, and this often introduced a large overhead in resources needed for each job. This master node overhead was also eliminated in the new scheme as the new peer-to-peer MPI job was an improvement over the old master-slave MPI job, where the master(s) contained the output image(s). This proved to be very fortuitous as the new scheme exhibits strong scaling. So as long as the problem fits in certain number of N nodes, then N can vary over a fairly wide range and the run time per shot will change proportionally.

The new scheme is a truly distributed solution and effectively frees us from limitations on the maximum frequency of a problem that can fit in the memory of one node.. For example, if 25 Hz problems could be processed with 12 nodes, then the corresponding 50 Hz problem would require 96 nodes to solve. We have tested such scenarios and see that they do scale properly. When we double the maximum frequency to be processed, each of the spatial dimensions decreases by half and also the time stepping decreases by half. So the amount of compute work to be done goes up by a factor of 16.

So in practice there is sufficient compute work to keep the SIMD units busy. Also, the slabs do not get too thin in each node due to the decreased bin sizes. There is intermodal communication overhead in order to perform the 3D distributed FFT that is needed in every time stepping of the algorithm. We measured the speed of a 3D distributed FFT on a regular Beowulf cluster with 1 gigabit leaf switches versus the same test on the Cray XC30. We found that this type of scheme would run wholly inadequately on a regular Beowulf cluster but has strong scaling on a Cray with the Aries networking.

Numerous commercial projects have now been run on our production Cray XC40, and we also retain a small XC30 system for testing and development. The new distributed TTI

RTM implementation running on the Cray XC40 is now our production workhorse.



IV. Kirchhoff Depth Imaging

The Kirchhoff Depth imaging algorithm we have implemented operates as a master-slave MPI program that also uses OPENMP. In the original implementation, the master rank would broadcast both the bulk input data and its associated travel time information. Each slave rank has ownership of a portion of the output space. These portions are staggered. So each subordinate rank receives the bulk data. Then each subordinate rank examines whether it should generate a response in the output space over which it has ownership. This is done for each trace in the group of traces that was broadcasted to it. If a response is to be generated in the output space, then the travel time information is used to do so.

All such output responses reside in the memory space of the collection of subordinate nodes. So in this sense Kirchhoff Depth migration is a distributed MPI program. It turns out that there was a bottleneck when reading travel time information from disk and broadcasting it to all nodes in the job, and the job would not scale after some point. This was true on both the regular Beowulf clusters and the Cray XC40 cluster. The behavior on the Cray XC40 was better due to the faster networking in the XC40.

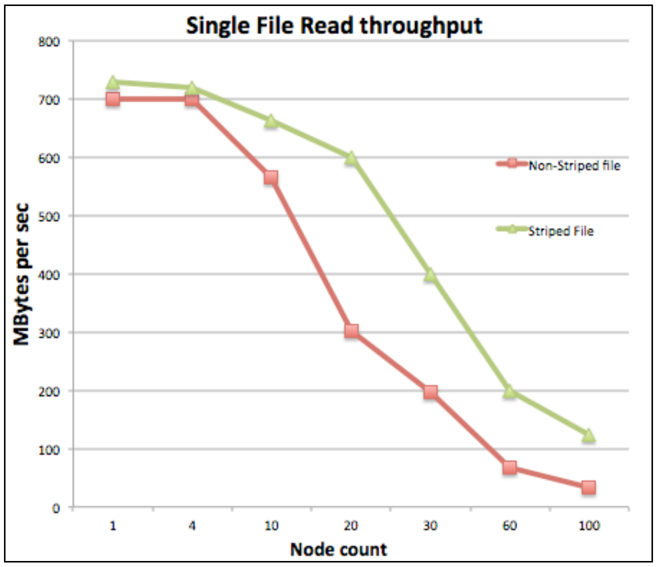
The solution to this problem was to stripe the travel time file across all the Lustre file system's object storage targets (OST) and to read all the travel time information into a shared memory (SHMEM) pool. From this shared memory pool, any node could access the travel times on demand using the networking in the Cray XC40. The travel times are only read once when the job starts, and thereafter they are accessed from any node in the job from shared memory using SHMEM one-sided communication. The amount of memory dedicated to the shared memory pool for travel times is about 10% of each subordinate rank.

V. One-Way Wave Equation Imaging

Wave equation migration (WEM) also used a model where one seismic shot was imaged per node on a Beowulf cluster. The intermediate imaged depth slices and the TTI earth model were kept on scratch disk if there was not sufficient memory. Whether there was sufficient memory depended on the maximum frequency to be imaged. The memory requirements also varied as the third power of the maximum frequency, which caused considerable efficiency loss when scratch disk was used. To remove this limitation, we decided to eliminate the utilization of scratch disk altogether by using more than one node to image a single seismic shot. Typically only 2 or 3 nodes are required to completely contain the problem in memory if the nodes have 128 gigabytes of memory. In this new model, we divided up the problem by depth slices. So the first node is tasked with imaging the early depth slices and later nodes in a group processing a shot are responsible for processing the later depth slices. Frequency slices are passed between nodes as processing occurs. There is some overhead in doing so, but in practice this is far less overhead than reading and writing to scratch disk, as the networking is fast enough to accommodate this.

VI. File System Lessons learned

In the process of restructuring algorithms to run efficiently in Cray XC 30/40, we observed that it is better to stripe the earth model and travel time parameter files. This is because there could be hundreds of jobs accessing the same earth model parameters, and unless the files are striped it would put an excessive burden on a single OST. Some output files from an accumulation process are also being striped over two OSTs to better distribute the IO load.



ACKNOWLEDGMENT

We would like to acknowledge the effort of Cray in devising a 3D FFT based on shared memory.

REFERENCES

[1] Sean Crawley, Sverre Brandsberg-Dahl, and Jim McClean (2010) 3D TTI RTM using the pseudo-analytic method. SEG Technical Program Expanded Abstracts 2010: pp. 3216-3220.

[2] John T. Etgen (2012) 3D Wave Equation Kirchhoff Migration. SEG Technical Program Expanded Abstracts 2012: pp. 1-5.

[3] A. A. Valenciano, C. C. Cheng, N. Chemingui, and S. Brandsberg-Dahl (2009) Implicit wave-equation migration in TTI media using high order operators. SEG Technical Program Expanded Abstracts 2009: pp. 3005-3009..

[4] W. A. Mulder and R.-E. Plessix (2004). ”A comparison between one-way and two-way wave-equation migration.” GEOPHYSICS, 69(6), 1491-1504.

[5] Phil Kitchenside, Uwe Albertin, Wenfong Chang, Clement Kostov, Alexandre Kleitz, Nick Moldoveanu, Ananthanaraya Sugavanum, and David Yingst (2001) Comparing finite-difference and Kirchhoff prestack depth migration. SEG Technical Program Expanded Abstracts 2001: pp. 917-920.

[6] Becker, Donald J and Sterling, Thomas and Savarese, Daniel and Dorband, John E and Ranawak, Udaya A and Packer, Charles V, "BEOWULF: A parallel workstation for scientific computation", in Proceedings, International Conference on Parallel Processing vol. 95, (1995).

[7] *Poole, Stephen (2011). "OpenSHMEM - Toward a Unified RMA Model". Encyclopedia of Parallel Computing: 1379–1391. Retrieved 2013-01-15*.

[8] Costian, Calin r. and Marinescu, Dan C., "A Distributed Memory Algorithm for 3-D FFT" (1993). Computer Science Technical Reports. Paper 1071