High-performance computing for seismic imaging; from shoestrings to the cloud

Sverre Brandsberg-Dahl, PGS

Introduction

The process of creating an image of the Earth's interior from surface seismic measurements has always required some sort of computation. Norm Bleistein was the first to impressed on me the richness in approaches and methods to solving this rather complex problem, back in the mid 90's when I was a young graduate student. Ten years earlier, Bleistein (1987) had introduced to the field of geophysics, a rigorous treatment of Kirchhoff migration through the tools and language of asymptotic analysis. This provided a roadmap for how one could implement in a computer, an imaging algorithm that would yield true-amplitude images of reflectors in the Earth. However, at this point in time, Norm was also truly fascinated by how geophysicists had discovered many of these principles years earlier, mainly through ingenuity and a deep understanding of the physics behind the seismic experiment and how to map the reflective layers in the sub-surface using reflections.
The earliest imaging algorithms had solved the problem of mapping reflectors in the subsurface, not through asymptotic analysis, but by using a string and pencil. One of the earliest examples of this can be found in the mapping "computer" of Haggerdorn (1958). Together with a fellow student at the time, we actually constructed such a "computer", not using a single transistor, but rather shoestring, nails and a pencil. With this we were able to map the classic "seismic bowtie" from a common offset section into its corresponding syncline in the subsurface space.

Following such early innovations, the groundbreaking work of John Claerbout (1971, 1985) caused nothing short of a seismic shift in the way industry and academia approached the subject of reflector mapping or migration. With the introduction the exploding reflector principle and a series of new and innovative ways of imaging subsurface reflectors, the computational era has truly arrived for the field of seismic imaging. Rapid innovation by the industry and academia introduced several innovative and robust approaches to how one could use a computer to effectively create reflector maps from surface seismic data. From these very early days until the 90's, this state of the art seismic imaging was using the available state of the art computers to perform this task. That meant that seismic imaging was being performed using mainframes and the many different supercomputers of the era. Computers from Thinking Machine, SGI, IBM and Cray were all commonplace as the tools of choice for seismic processing and imaging. The seismic companies and oil companies alike were big customers of high end computing equipment, and the imaging algorithms being developed were evolving along with the ever-increasing computational power, see Figure 1. The investment levels at the time in such hardware must have been substantial, and some of the biggest commercial HPC installations from then until present day have been dedicated to seismic imaging. Around the mid 2000's compute clusters based on commodity hardware were introduced to the field, and rapidly concurred all aspects of seismic imaging. By connecting together hundreds or even thousands of individual computers, pretty much anyone could build and have access to supercomputer power. Clusters have remained the mainstay of the industry until today. The latest addition to this trend is the move to more use of cloud computing, where cluster computing is offered as a service by a variety of providers.

According to the trend shown in Figure 1, the seismic industry should soon be entering the era of exascale computing. When this will happen is still highly uncertain, given market forces and reduced investment levels, but also due to technical challenges facing the scalability of seismic imaging algorithms. Until today, it has been pretty much a hunt for Flops, but dataset sizes and input/output are maybe now the biggest obstacles to continuing on the trend from the past. Also, more complex, heterogeneous computer architectures, are introducing challenges around programmability, so it might not be as easy now to adopt new algorithms to the hardware of the day as in the past. There is an important lesson to be had from the now vintage machines of the 90's, in that they were working with a nicely balanced ratio of I/O to compute. I will come back to this later when discussing some of the trends we might face in the years ahead.

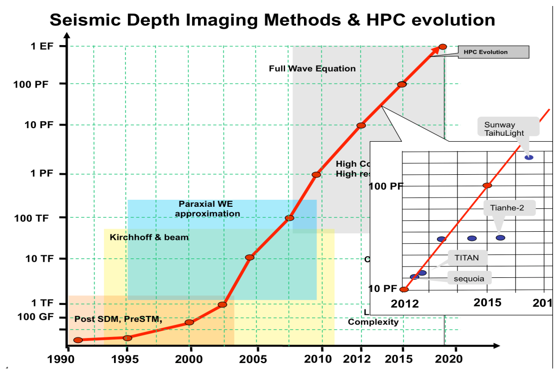

Figure 1: *The evolution of seismic imaging algorithms and relative compute power availabiltiy. The zoom shows some of the most powerful supercomputers of the last 5 years. (taken from a presentation by Calandra et al., 2017, Rice Oil & Gas HPC Conference)*

Breaking with a long standing paradigm; clusters

Seismic imaging went through a paradigm shift about 20 years ago when the first compute clusters were introduced. This caused a rapid and decisive shift in industry compute platform; away from the earlier generations of mainframes and supercomputers to relatively cheap clusters built using commodity components.
On the algorithmic front, this helped facilitate the shift to shot based migrations, where rather than treating the data in CMP order or as offset classes, the seismic data was simply imaged one shot at the time. This ushered in the era of shot migration and introduction of the term "embarrassingly parallel" application, as scalability was achieved by simply adding more computers to do more shots at a time. An illustration of data and some legacy hardware is shown in the top portion of Figure 2. Each shot could be handed off to an individual computer, migrated 'in the box', and then the image is formed by collecting all the individual sub-images into a final stack/image. The fidelity of the algorithms, and problem size (aperture and frequency) was then basically tracking the evolution of improving CPU speed and increasing memory size. This trend is also observable on the graph shown in Figure 1. As long as the individual shot fit inside the footprint of the individual server, this was indeed embarrassingly parallel, and enabled a rapid growth in capacity and capability of imaging algorithms. With the introduction of GPUs this trend continued, and even more compute flops were available to solve the imaging problems faster or more accurately.

It is not until recent days that this model has started to be challenged, mainly due to the slower growth in memory size and performance. As larger and more finely sampled computational domains are required, applications like RTM started to push at the boundaries of what is physically possible to do inside a single server. This trend was also helped along, or maybe pushed along, by the ever more advanced seismic surveys and larger data volumes acquired by the industry. The introduction of very long offsets in marine seismic (Long et al., 2014) pushed the problem size up by a factor 4x as offsets increased from a typical 8km to more than 16km and in a full-azimuth fashion, requiring a doubling of a typical computational aperture, just to capture the input data. Computational performance for algorithms like RTM is obviously still key, but the true hotspot when imaging such datasets is now shifting to lack of available address space (memory) to efficiently hold the problem inside the computer. With this backdrop, a logical challenge to the prevailing shot parallel implementations was introduced, and pushed us to look for solutions that could provide access to very large memory footprints.
This move is also supported by the rapid adaptation of blended data acquisition; where data instead of being acquired as individual shot records, is recorded as continuous records capturing multiple shots. Such records do no lend themselves to the classical processing schemes of cluster-based RTM, unless some form of data de-blending is performed (Berkhout, 2009). To treat the continuous records directly is putting further pressure on compute architecture and algorithms, in that the time axis now becomes much longer. An illustration of this is shown in the bottom part of Figure 2. This also shows the back pane of a Cray XC30 computer, where the massive wire bundles represent the network infrastructure provided to support the shared memory architecture of this machine. With sufficient memory, one can imagine that the complete seismic experiment can be treated as one, offering hope for moving beyond the reliance on individual treatment of shot records that prevail in most algorithms today. The choice of algorithms closely follows the trends in available hardware, so while the clusters have reigned supreme for quite some time, there might be a change on the horizon towards new algorithms and infrastructure to support them.

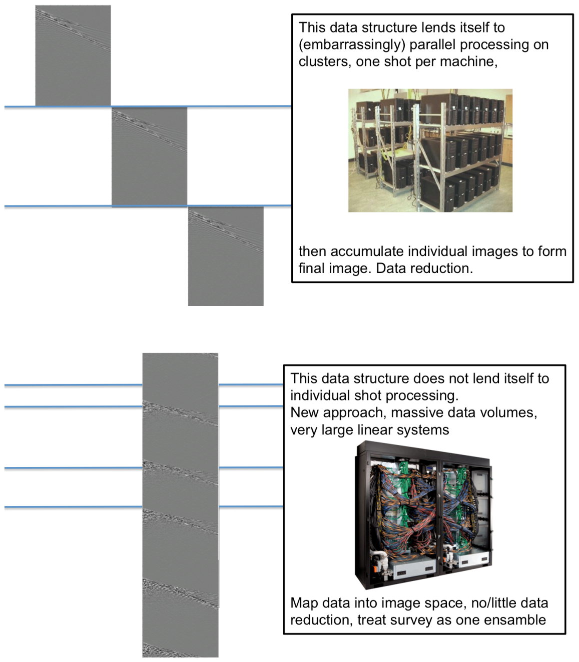
This data structure lends itself to (embarrassingly) parallel processing on clusters, one shot per machine,
then accumulate individual images to form final image. Data reduction.
This data structure does not lend itself to individual shot processing.
New approach, massive data volumes, very large linear systems
Map data into image space, no/little data reduction, treat survey as one ensamble

Figure 2: *A comparison of data structure and computaational models for the classical shot-parallel data and hardware (top), as compared to continous data recordings with blended acquisition, and an example of compute hardware to support the use of shared memeory, a Cray XC30 system.*

Implementing RTM on a shared memory architecture

In this section, I will review the effort undertaken by a team of geophysicists and computer engineers to port a RTM code originally developed to run on a cluster to run on a shared memory architecture machine. The RTM code that was originally designed to run on dual-socket servers with local scratch disk for storage of snapshots, and the goal is to port it to run on a Cray XC40 machine. The XC40 is also built around dual socket compute nodes, but no local scratch is available, and the nodes are networked to allow for rapid access also to memory not local to any node. The same CPUs were used in both platforms; so one could maybe easily conclude that the best possible outcome would be parity between running a shot inside a single node in the cluster versus across several nodes on the XC40.

The first step in this effort was to simply run the code, as is, on the XC40. As there is not local scratch disk in the nodes, this would rely on storing the snapshots on a centralized, parallel file system instead. The relative performance of this is shown in Figure 3, indicating that such a straight port resulting in about a 3x slowdown. The next step was to mitigate the lack of local disk. This was done by instead leveraging the very large memory pool available in the XC40 for storing the snapshots. This obviously required the use of multiple compute nodes per shot migration to create a sufficient memory footprint to hold both the computational domain as well as the snapshots. To facilitate this, the FFTs used in the pseudo analytical extrapolator (Etgen and Brandsberg-Dahl, 2009) were replaced with distributed versions that run across all threads in the node pool allocated for each shot. At this stage, this was performed at math library level, switching out the standard Intel MKL FFT with the distributed MKL FFT. As can be seen in Figure 3, this helped performance, reducing the runtime by about 40% relative to the version relying on the parallel file system. However, this was still not sufficient to get performance on par with running one shot per node.

Continued analysis of performance of the now distributed FFT-based implementation showed that further optimization was possible by rearranging some of the computational kernels of the code as well as optimizing the transposes used when applying the TTI anisotropic terms in the extrapolator. After these items were addressed, the relative run time for the distributed code on XC40 was less than when running one shot per node. Surprising maybe, but this reflect the fact that snapshots are now stored in memory with much faster access than even what was provided by high-performance, in node scratch disk.

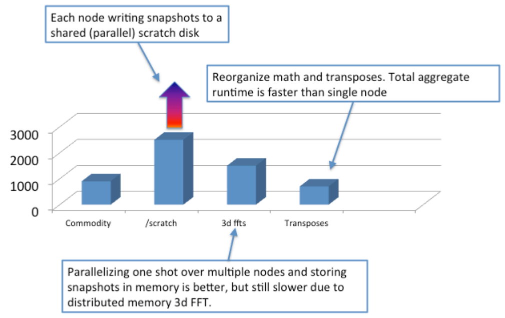

Figure 3: *Relative performance of RTM when run fully inside a dual socket commodity node and on a shared memory massively parallel computer. A straight port, versus re-engineering of the code.*

Having successfully ported RTM to a shared memory architecture and obtained better relative performance compared to the one shot per node reference, the next step was to add functionality to output azimuth sectored angle gathers from the RTM (Frolov et al., 2016). This is further challenging the classical cluster compute model, as the output space is now increasing from a typical 3D stack image volume to a 4D or 5D volume, depending on how

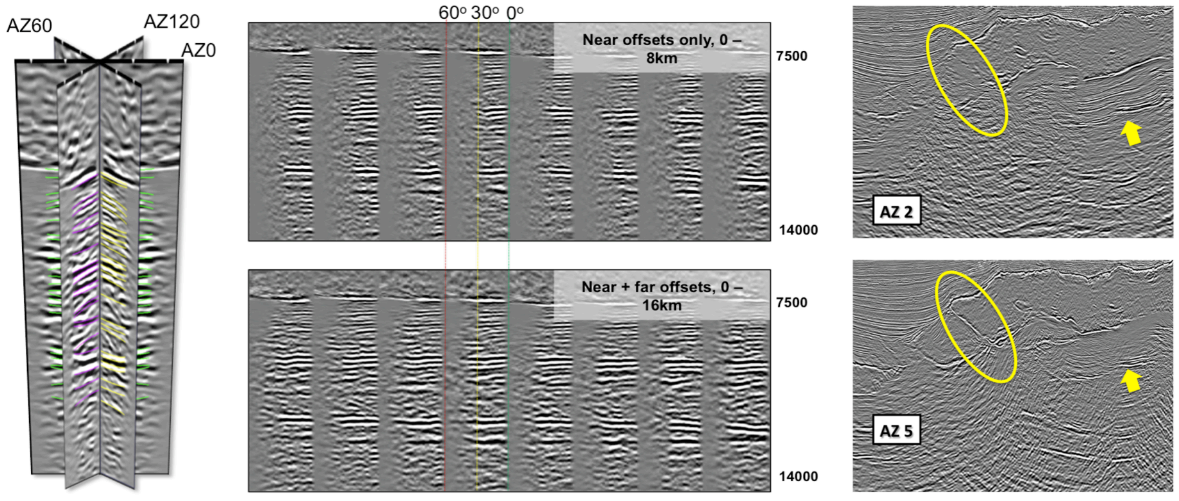

Figure 4: *An example of azimuth sectored angle gathers from RTM, left, with RMO tomography picks overlayed. The center image shows angle gathers from a Gulf of Mexico dataset, where the angle domain is used to combine data from two separate datasets. The right portion shows an example of how the sub-surface azimuth direction can be very useful for image*

many angular dimensions are used for the output image. An example of an azimuth sectored angle gather from RTM is shown in Figure 6 along with examples of how the angle domain can be leverage for anything from velocity model building to advanced image optimization.

After successfully moving RTM to run on a shared memory computer, performance was retained, and new abilities introduced. When dealing with very large, densely sampled input datasets and also wanting to keep data redundancy (pre-stack information) on output, a just as important challenge becomes memory size, bandwidth and overall system I/O performance. For the Gulf of Mexico FAZ survey described by Frolov et al. (2016), each output location has 276 angle traces, so the RTM output volumes for the about 12,000km^2 survey was 276 times larger than a stacked image. Data management and I/O will maybe be as important as availability of pure compute Flops when it comes to addressing the seismic imaging applications of the future.

With a need for more refined data decompositions and analysis of data in multiple domains, it should be a safe bet to expect that problem sizes will grow. Add to this the trends towards more use of inversion algorithms, the era of embarrassingly parallel computing might soon be behind us.

Programmability versus performance; the future of HPC in the seismic industry

If we as an industry are to continue to follow the trend of compute consumption shown in Figure 1, there are challenges ahead. The growth shown here was mainly driven by a few key trends in HPC that the seismic industry was able to follow and exploit fully. However, there is a risk today that seismic imaging industry and HPC might diverge. On one hand, the true mega compute centers of today are server farms run by the Internet giants like Amazon, Google and Microsoft, and they tend to have hardware profiles very different from what we are used to in the seismic or Oil & Gas industries. But with the rapid growth seen in cloud computing, it might just be a necessity for us to jump onto the trend and adapt our problems to the prevailing hardware trends and programming models used there. On the other hand, it is still perfectly plausible that the classical HPC vendors will continue to innovate and provide hardware platforms that will prove themselves cost competitive also in the future. I believe there will be continue focus on cost and turnaround, so whichever compute platform can offer an edge will prevail.

An important component in creating such a competitive edge comes from the programming model and software stack associated with the hardware. This is particularly true when one considers the growing heterogeneity in hardware with mixes of CPU, GPU and other accelerators. For a computer to be truly useful, it must have an acceptable programming model. If a model can offer better flexibility and reduced turnaround in software development from concept to at-scale implementation, it might help offset the cost of the pure hardware components when evaluated with a view to cost performance and cost of ownership. The use of FPGSs comes to mind as a good example, notoriously hard to program, but with great performance. Their popularity in our industry has been on and off for quite some time, and we will for sure see them again in the near future, but hopefully with a much better programming model as part of the package.

Maybe the key item that will help set the direction for the future is how fast any hardware/software combination can get an organization from concept to at-scale application. Obviously cost performance will be a factor, but there seems to be a trend towards an ever more rapid development cycle for imaging technologies, that then need to be run on a large computer system to be applicable to the datasets from modern seismic surveys. People resources enters this picture as well, so a holistic view on everything from programming language, compilers and libraries, to hardware configuration and power consumption will have to be addressed in any successful solution.

Acknowledgement

There is a large team of people to thank at PGS for having made the move to share memory computing possible. Of a too long list to print here, I would like to acknowledge Raj Gautam, Sean Crawley and Pat Manning. Thanks to Cray Inc. for support throughout the process, and finally thanks to PGS for support and permission to present this work.

EDITED REFERENCES

Note: This reference list is a copyedited version of the reference list submitted by the author. Reference lists for the 2017 SEG Technical Program Expanded Abstracts have been copyedited so that references provided with the online metadata for each paper will achieve a high degree of linking to cited sources that appear on the Web.

REFERENCES

Claerbout, J., 1985, Imaging the Earth's interior: Blackwell Science.

Bleistein, N., 1987, On the imaging of reflectors in the Earth: Geophysics, **52**, 931–942, http://dx.doi.org/10.1190/1.1442363.