Graphics Processor Clusters for High Speed Backpropagation

Daniel P. Campbell, Daniel A. Cook
Georgia Tech Research Institute / Sensors and Electromagnetic Applications Laboratory
{dan.campbell, dan.cook}@gtri.gatech.edu

Abstract

This paper describes the use of GPU clusters to accelerate backpropagation for Synthetic Aperture Sonar (SAS) systems. We extended a GPU-based implementation[1] of backpropagation to support clusters of GPU-enhanced nodes. The GPU accelerated implementation formed a 3,936 x 3,936 SAS image from 60s of sonar data in under 12s using a single GTX480, and under 3s on a small cluster with 8 Tesla C2050s. Our presentation will discuss the details of our implementation, describe our optimizations, and show performance for various image sizes and execution platforms.

Sonar Backpropagation

The classical approach to both radar and sonar synthetic aperture image reconstruction is known as backprojection, or backpropagation. The scheme is straightforward. Each point in the scene being imaged contributes reflections to throughout the recorded data. Any point has a corresponding locus of echo returns in the observed data. In order to compute the value of a single output pixel in the reconstructed image, all that is required is to integrate the data along this locus while multiplying by the complex conjugate of the expected locus. For each output point this operation has the form of an inner product, and the reconstructed image can be thought of as resulting from a spatially-varying correlation operation. The inner product serves as a measure of how similar the measured data is to the expected locus, and a strong correspondence results in a bright output pixel. Backpropagation is an $O(n^3)$ algorithm with image edge size. For certain collection geometries, there exist Fourier-based reconstruction techniques with a lower order complexity than backpropagation, but these all lose accuracy when the motion of the collection platform deviates from the idealized path, and are unsuitable for wide beam collections.

Motion compensation to enable simpler algorithms is generally not a problem for small perturbations of the collection path, but large distortions result in useless data. This is illustrated in Fig. 1 which shows a typical SAS mission. Imaging software based on, for example, the ω-k approach cannot use the data in the maneuvers at all. Although the drawing in Fig. 1 is not to scale, the maneuvering area can represent a substantial lost opportunity for image collection because battery life limits the duration of the mission for the vehicle.

In addition, motion compensation works best for sonars with narrow beams. Wide beam sensors are also of interest, but motion compensation processing can limit their operating envelope in terms of tolerable deviations from straight-line flight[2]. Lastly, the typical platform speed (1.5 m/s) is an appreciable fraction of speed of sound underwater (1,500 m/s). Image reconstruction algorithms must account for the fact that the sensor can move between the times of transmission and reception, adding a level of complexity not present in Synthetic Aperture Radar (SAR) systems.

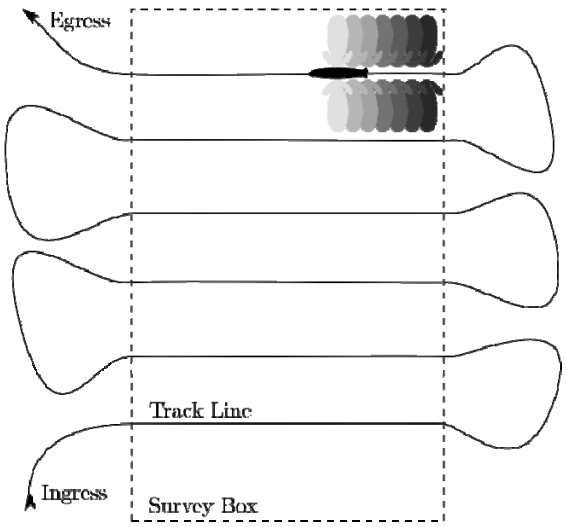

Figure 1: Typical mission for autonomous synthetic aperture sonar data collection. The ends of the 'dog bone' tracks represent lost opportunities for image generation when traditional Fourier-based techniques are used.

Application

GTRI maintains a MATLAB-based synthetic aperture processing toolbox that supports the study of new approaches for radar and sonar knowledge extraction. This toolbox includes a wide variety of primitives to support experimental and prototype processing approaches, including backpropagation. For any but the smallest output images, execution time for the backpropagation model is prohibitively long. Long runtimes limit the ability of algorithm researchers to study approaches that include backpropagation. Algorithm prototyping must be done with much smaller images, inferior image formation approaches, or with long test and iterate cycles.

Accelerated Implementation

Synthetic aperture sonar image formation via backpropagation is well suited to parallelization on many platforms. While the mathematical operations to form the image are pleasingly parallel, each input datum is used by many output points, and the relationship between output point location and input data used is non-linear. Because of these factors, it is straightforward to create an implementation of backpropagation that benefits from increased parallelism, but challenging to fully optimize it for a specific platform.

Our implementation of the SAS image formation module is accelerated with nVidia GPUs, and was developed using CUDA. The implementation launches one thread per output pixel location, with two-dimensional blocks to improve intra-block range coherency. Each thread loops over the input pulses synchronously within a block, and accumulates the correlated output as it traverses the input set. Linear interpolation of return data is implemented with texture sampling operations in order to exploit dedicated logic. Our implementation accelerated the formation of a test image from an estimated 50 days using the original (unoptimized) toolbox implementation to 12s on an nVidia 480GTX-enhanced workstation, and under 2s using a small cluster enhanced with 16 Tesla C2050s. Reoptimization of the algorithm and communication structures for larger cluster is ongoing.

Performance Testing & Results

We tested the software on a small cluster with each node containing two Intel Xeon 5660 six-core processors operating at 2.8GHz, and two Tesla C2050s. For our tests, we formed a 3,936x3,936 pixel output image from data sets containing 3,936 pulses each. The input data for each image corresponded to a typical sonar data collection performed over a period of approximately 60 seconds. We tested using from one to eight nodes, using one or two of the available Telsa C2050 per node, as well as using all twelve available CPU cores for a lightly optimized C-based implementation.

The measured runtimes are summarized in Table 1, below. The reported flops/s was estimated by treating square roots, divisions, and trigonometric functions as single floating point operations. Under these assumptions, the minimum flops per output pixel 102 per input pulse for SAS. Synthetic aperture radar is similar to SAS, but the much higher ratio of propagation speed to platform speed allows simplifications that reduce the load to 34 flops per pixel per pulse.

	Runtime(s)		
Nodes	1 GPU/node	2 GPU/node	flops/s
1	14.57	8.68	716.5E+9
2	8.65	5.04	1.234E+12
3	6.23	3.64	1.709E+12
4	4.92	2.98	2.087E+9
8	2.87	2.00	3.110+12
12xCPU no GPU	925.8		6.718E+9

Table 1 - Benchmark Results

These results demonstrate a speedup of 63x to 463x for our implementation relative to a lightly optimized, 12-core C based implementation, and a speedup of several orders of magnitude relative to the baseline MATLAB implementation. We found that in contrast to our single-node, 8-GPU implementation, our cluster implementation did not scale as well, achieving only 65% of linear speedup moving from one to eight nodes, and from 72% to 85% of linear speedup moving from one to two GPUs per node. Scaling in both nodes and GPUs per node resulted in a speedup of only 45% using 16 GPUs.

Improvements and Future Work

The lack of strong scaling in the current implementation limits the utility of the implementation. For wide-area collections, much larger images are desirable. The effective use of larger clusters, and more GPUs per node is needed in order to achieve suitable runtimes for larger images with larger input data sets.

The backpropagation algorithm requires that every pixel consider the effect on its final value from every input pulse, resulting in a communication cost that increases with the number of nodes. Depending on the collection and beam geometries it is often possible to identify in advance which pulses, or return samples within each pulse are required by regions of the output image. For collection geometries where every pixel is within the range swath of every pulse, digital spotlighting techniques may be used to synthesize smaller data sets that are functionally equivalent to the full data set for specific regions of the image, reducing both the communication costs, as well as the total amount of computation per node. In future implementations, we will implement one or more of these techniques to reduce communication costs.

Our current implementation suffers from a per-host-thread initialization cost that is appreciable. This cost varies by system but was at least 0.2s in the results shown in Table 1, and as high as 2.5s on larger clusters. Improvements in the separation between initialize and execute are available to remove this cost.

Future implementations will incorporate these improvements, as well as further intra-node optimizations to reduce multi-GPU overheads and resource contentions. We will demonstrate the findings of these optimizations on larger clusters with 64 or more GPUs.

References

[1] D.P. Campbell, D.A. Cook, *Using Graphics Processors to Accelerate Synthetic Aperture Sonar Imaging via Backpropagation* , High Performance Embedded Computing Workshop, Lexington MA, 2010

[2] H. J. Callow, *Signal Processing for Synthetic Aperture Sonar Image Enhancement*. PhD thesis, Department of Electrical and Electronic Engineering, University of Canterbury, Christchurch, New Zealand, 2003.