

# Application-level Benchmarking with Synthetic Aperture Radar

Chris Conger, Adam Jacobs, and Alan D. George  
{conger, jacobs, george}@hcs.ufl.edu

High-performance Computing and Simulation (HCS) Research Laboratory  
Department of Electrical and Computer Engineering, University of Florida

## Abstract

*Space-based radar applications continue to receive increasing interest from the research community, and new technologies are emerging that will help to make the vision of real-time, on-board, high-volume data processing a reality for next-generation space platforms. Isolated kernel benchmarks may not accurately capture true system performance in the context of a full application, and so application-level benchmarking is needed to measure true system performance. This research presents a full Synthetic Aperture Radar (SAR) application benchmark, written in ANSI-C and using only the Gnu Science Library (GSL) and Message Passing Interface (MPI) libraries. The SAR application forms high-resolution images from raw radar data, and incorporates many computational kernels of interest to the HPEC community. We present performance results from two different parallelizations of the application, executed on a cluster of PowerPC processors connected via Gigabit Ethernet. In addition to presenting the performance results on our system, the benchmark source code and example data is made publicly available.*

## Introduction

The HPEC Challenge benchmark suite was created to give developers a rigorous set of tests that can be used to evaluate different embedded architectures [1]. The kernel benchmarks in the suite are intended to represent a wide range of common applications, such as radar processing and hyper-spectral imaging. The SAR application benchmark that is included with the HPEC Challenge suite implements one fixed parallelization, though it may be desirable to explore the performance of alternate decompositions. The SAR benchmark presented in this work offers unique features and configurability, as well as multiple parallel decompositions. With newer embedded systems for space, such as the Dependable Multiprocessor [2], featuring a more traditional cluster architecture, accurate parallel performance evaluation will play an important role in system design.

This application-level benchmark implements *strip-map* SAR, where the radar platform is moving over the Earth while the radar itself remains pointed downward in a fixed direction. As the platform moves, an image is formed of the strip of ground underneath the satellite. Additional details on the mathematics of the SAR algorithm itself can be found in [3]. The processing flow of strip-map SAR is amenable to parallelization on multiple levels of granularity, and the benchmark presented in this research is capable of taking advantage of this multi-level parallelism. The basis of SAR used for this benchmark was derived from sequential reference code provided by the Scripps

Institute of Oceanography. Along with the source code, a genuine ERS-2 satellite image file was provided to serve as an example input data set for the application. Figure 1 shows the output image produced by the parallel SAR benchmark using the ERS-2 satellite file as input.

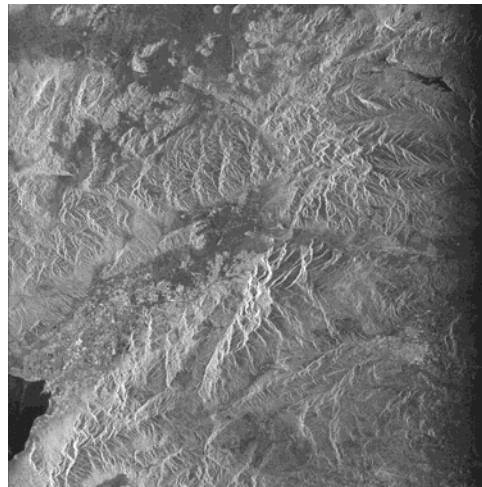


Figure 1: Image formed by benchmark using real ERS-2 data

## Benchmark Description

The benchmark consists of a sequential baseline and two different parallel algorithms for the SAR application. Each design is parameterizable, providing control over numerical precision, image size, and radar/satellite characteristics. For each parallel design, there is also an optional “control thread” that may be enabled to cause small messages to be passed between nodes, a feature inspired by simulative research performed at the University of Florida in [4]. These small messages are intended to imitate latency-sensitive control traffic that must share the network with data traffic from the SAR processing. Latency statistics for these small messages are gathered and reported to help characterize the effect of SAR data movement on critical control traffic for a given system. Additionally, there are utility programs associated with the benchmark, including a data generation utility to generate arbitrary-sized images for input, and a bitmap generation utility to generate viewable image files from the benchmark output.

The SAR application consists of four processing stages, and requires a transpose of the 2-dimensional image between each stage. This benchmark processes the input image in patches, with overlapping patch boundaries and each patch being processed independently. By keeping only a portion of the fully processed data from each patch, the output of consecutive patches can be appended together seamlessly to form the complete output image. All parallel versions of the benchmark reserve one node to serve as the I/O node,

which is solely responsible for reading the input data from file and sending it out to nodes for processing as well as receiving processing results from nodes and writing them to an output file. The first parallel version of the SAR application takes the simplistic approach of assigning one node to each patch to be processed, and letting all patches be processed in parallel. The second parallel version of the benchmark organizes all of the nodes in the system into a selectable number of groups. Each group of nodes receives a patch from the I/O node in a round-robin fashion, and processes it in a data-parallel manner. By choosing the number of groups to be one, this parallelization reduces to a true system-wide, data-parallel decomposition. Figure 2 illustrates the processing flow of the SAR application.

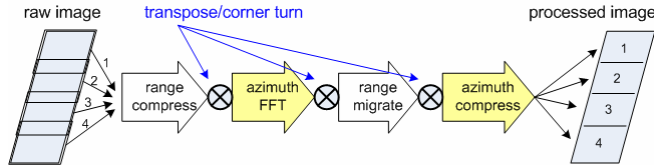


Figure 2: SAR processing flow

## Results

The benchmark was executed on an embedded systems testbed of 10 PowerPC G4 servers connected via Gigabit Ethernet, where size of the ERS-2 satellite image is 5616 by 27990 pixels. Execution times for the first parallelization of SAR, with one patch per node, are shown in Table 1. Recall that for all parallel cases, one node is used for disk I/O and communication only, and thus the number of processing nodes is one less than the total number of nodes.

Table 1: SAR Execution Times (in seconds)

Sequential	2 Nodes	4 Nodes	8 Nodes
640.2	670.9	254.4	185.2

Comparing the two-node case in Table 1 to the sequential performance gives an idea of the overhead introduced through parallelization. In the two-node case, one node is used purely for disk-I/O and the other node is used for processing. Figure 3 shows a graph of overall application speedup vs. number of nodes used for processing. The first parallelization has crippling limitations on maximum attainable performance, and the lack of scalability of this parallelization approach is illustrated by the plateau-like shape of the speedup curve as the number of nodes increases. Due to the round-robin distribution of image patches from the I/O node, if the total number of patches is not evenly divisible by the number of nodes in the system, there may be little to no benefit in using more nodes until reaching the next even factor. Also, since each patch is processed by a single node, the maximum number of nodes that can be used is limited by the number of patches decomposed from the overall image. In order to squeeze more performance out of a given system for this SAR application, each patch can be processed in a data-parallel fashion by a group of nodes, with different patches being processed by different groups in parallel. Due to space limitations, additional results are reserved for the full presentation.

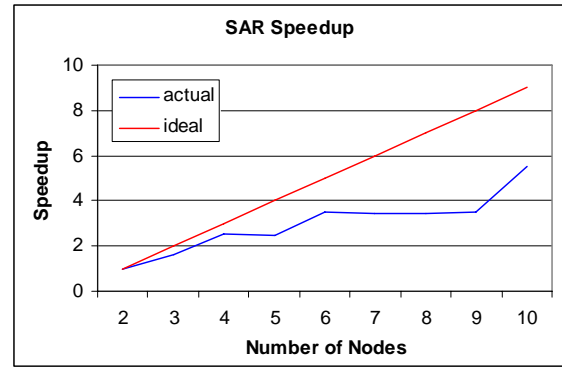


Figure 3: Speedup vs. system size

## Conclusions

We have developed and studied a full application-level benchmark using SAR. With several parallelization approaches possible, we have found that a hybrid data-parallel/patch-based decomposition can provide the best speedup and parallel efficiency on our system. Another interesting outcome of this benchmark is that it can be seen that single-precision, floating-point format provides sufficient precision, as no differences were observed between output images created with single-precision vs. double-precision arithmetic. By using single-precision float, not only are the computational latencies decreased for floating-point operations, but the amount of data that must traverse the system interconnect is reduced by half. For the implementations included in our benchmark, the I/O node becomes a bottleneck as the system size increases, forcing processing nodes to wait while the single I/O node passes out data for processing one node at a time.

## Acknowledgements

We wish to thank Honeywell Electronic Systems Engineering and Applications - Space (ESEA-Space) in Clearwater, FL for their support of this research. We would also like to thank Dr. David Sandwell and the Scripps Institute of Oceanography for graciously providing the original SAR code that served as the basis for the development of our benchmark.

## References

- [1] R. Haney, T. Meuse, J. Kepner and J. Lebak. *The HPEC Challenge Benchmark Suite*, Proc. of the Ninth Annual High-Performance Embedded Computing Workshop (HPEC 2005), Lexington, MA, September 2005.
- [2] J. Samson, J. Ramos, M. Patel, and A. George, *Technology Validation: NMP ST8 Dependable Multiprocessor Project*, Proc. of IEEE Aerospace Conference, Big Sky, MT, March 4-11, 2006.
- [3] C. Miller, D. Payne, T. Phung, H. Siegel, and R. Williams, *Parallel Processing of Spaceborne Imaging Radar Data*, Proc. of IEEE/ACM Supercomputing Conference, San Diego, CA, Aug. 14, 1995.
- [4] D. Bueno, *Performance and Dependability of RapidIO-based Systems for Real-time Space Applications*, Ph.D. Dissertation, Dept. of Electrical and Computer Engineering, University of Florida, Gainesville, FL, 2006.