Applying Advanced Computing to Improve High-Fidelity Radar Data Simulations

HPEC 2006 - High Performance Rapid Prototyping

Chris Hulbert cch@isl-inc.com
Jamie Bergin jsb@isl-inc.com
Paul Techau pmt@isl-inc.com
Outline

• Introduction
  – radar system analysis
  – clutter signal model and simulation problem size

• Implementations
  – Matlab
  – MEX functions
  – parallel Matlab simulations
  – Standalone code

• Performance Results
  – Matlab
  – standalone

• Summary and Future Work
Radar System Analysis

- Radar signal modeling has proven valuable in characterizing system performance
- Most performance measures are evaluated over short time intervals
- Other performance measures such as tracking require longer processing intervals
- Simulating enough data for tracking analysis can take weeks or even months on single nodes
- Some signal models reduce fidelity to decrease required simulation run times
- Another approach is to implement high-fidelity models and apply high performance computing resources
Clutter Signal Model

- A typical clutter signal is

\[ x_c(t) = \sum_{p=1}^{P} \alpha_p s_p(t) \circ v_p \circ u_p \in \mathbb{C}^{NM \times 1} \]

- \( \alpha_p \) is the complex scattering amplitude of the \( p^{th} \) patch
- \( s_p(t) \) is an arbitrary waveform varying for each pulse and channel
- \( v_p \) is the space-time steering vector for the \( p^{th} \) patch
- \( u_p \) is a unit energy modulation mechanism accounting for temporal and spatial subspace leakage

- The signal model accounts for real-world effects including
  - range walk and bandwidth effects
  - environment databases (e.g. terrain and land cover data)
  - platform acceleration during a CPI (e.g. satellite orbits)

- Since all parameters are independent from patch to patch, this problem is embarrassingly parallel
Problem Size

• All terms, except $\alpha_p$, are $M*NxL$ complex matrices
  – $M$ is the number of pulses in a CPI
  – $N$ is the number of receive array channels
  – $L$ is the number of range bins in the radar data

• There are on the order of $4*M*N*L*P$ floating point multiplications and $M*N*L*(P-1)$ complex additions ($P$ is the number of clutter patches in the problem)

• Typical parameter values
  – $M*N = 1000$
  – $L = 1000$
  – $P = 35e6$
  – 910 TFLOPS

• Comparatively space-time adaptive processing of a comparably sized data cube is on the order of 5 GFLOPS
Outline

• Introduction
  – radar system analysis
  – clutter signal model and simulation problem size

• Implementations
  – Matlab
  – MEX functions
  – parallel Matlab simulations
  – Standalone code

• Performance Results
  – Matlab
  – standalone

• Summary and Future Work
Implementations

- Matlab toolbox implementation
  - serial
  - MEX functions
  - parallel simulations using MatlabMPI/pMatlab

- Standalone implementation
  - based on same code as MEX
  - parallel implementations using MPI and OpenMP

- Implementation summary
Matlab Implementation

- The radar simulation code is divided into three Matlab toolboxes
- Splatter, Clutter, and Target Signal (SCATS) toolbox
  - RF phenomenology functions
  - site-specific modeling including terrain and land cover effects
- Radar simulation toolbox
  - radar simulation framework including task/parameter initialization and loop over patches
  - uses function handles to run the tasks allowing easy changing of models including of user-written models
  - IQ data and RF phenomenology outputs
- Space-Based Radar toolbox
  - space-based scenarios and clutter modeling
  - space-based simulation tools including the clutter signal model described earlier
MEX Function Implementations

• Even though Matlab is built on many well-optimized libraries, some functions can perform better when written in a compiled language (e.g. C and Fortran)

• Matlab recognizes this and provides a convenient API for interfacing code written in C and fortran to Matlab functions

• These interfaces are known as MEX functions

• Other benefits of MEX functions
  – can exploit multi-core processors using threads and OpenMP
  – easier transition to Matlab-independent software
  – external code implementations are constantly and thoroughly exercised by analysts using Matlab simulations
VSIPL Library

- The MEX functions are based on ISL C libraries that use the Vector, Signal, and Image Processing Library (VSIPL) for portable and efficient vector, matrix, and signal processing operations
- ISL has optimized parts of the VSIPL library for AMD/Intel architectures
  - OpenMP parallelization
  - AMD Core Math Library (including the vector math functions)
  - Intel Math Kernel Library
- Additional optimizations will be implemented based on profile results
  - more BLAS functions
  - VSIPL FFT functions may use FFTW3 or the AMD/Intel library FFT functions
Parallel Radar Simulations

- MatlabMPI and pMatlab are parallel programming Matlab toolboxes developed at MIT Lincoln Laboratories
  - http://www.ll.mit.edu/MatlabMPI
  - http://www.ll.mit.edu/pMatlab
- MatlabMPI is a Matlab toolbox that provides a subset of the Message Passing Interface (MPI)
- Launches one or more Matlab instances on multiple nodes
- pMatlab is a toolbox based on MatlabMPI
  - introduces processor maps and data distribution
  - hides most of the details of the parallel communication/coordination
- The actual underlying parallel code of ISL’s toolboxes is abstracted from the user allowing the underlying framework to change without affecting user scripts
Standalone Simulation Software

- Like the Matlab implementation, the standalone code was divided into three libraries
- These libraries are the same ones the MEX functions are based on
- Uses the same flexible simulation style as the Matlab toolbox by using C function pointers for tasks
- The simulation driver program uses a simple INI style configuration file for parameter inputs
- If a runtime linker is available, simulation functions can be changed without recompiling the driver program
- Includes parallel implementation using MPI
- Outputs can be saved as Matlab data files for easier analysis
Implementation Summary

Radar Simulation Driver Programs
C/Fortran

Matlab toolboxes:
matlab_scats,
isl_radar_sim,
and sbr

Radar Simulation Driver Scripts

C/Fortran Libs:
libscats,
libisl_radar_sim,
and libsbr

MEX Functions
Outline

• Introduction
  – radar system analysis
  – clutter signal model and simulation problem size

• Implementations
  – Matlab
  – MEX functions
  – parallel Matlab simulations
  – Standalone code

• Performance Results
  – Matlab
  – standalone

• Summary and Future Work
Performance Results

• Simulation setup

• Matlab performance
  – single node simulation
  – single node with MEX functions
  – parallel simulation
  – parallel simulation with MEX functions

• Standalone performance
  – single node
  – distributed memory parallel performance
Simulation Setup

- Low Earth Orbit (LEO) monostatic simulation

- Problem Size
  - 127 pulses, 10 receive subarrays (MN=1270)
  - 121 range bins (L=121)
  - 250,000 clutter patches (P=0.25e6)
  - 192 GFLOP

- Timing
  - Initialization times the iterative calculation of clutter patches
  - Clutter Signal times clutter signal model shown on slide 4 including the generation of the waveform and spatial steering vector
  - Total Simulation times the entire simulation including initialization, IQ data calculation, and in the parallel cases collecting data from all the nodes.

- Architecture (unless noted) is a 9-node cluster of Intel Pentium 4 2.4 GHz systems with 1 Gb SDRAM per node
Matlab Implementation Performance

- **MEX speedup**
  - 15.9x Initialization speedup
  - 22.1x clutter signal computation speedup
  - 21.6x total simulation speedup

- *As predicted by the problem, speedup is approximately linear*
Standalone Implementation Performance

**Single Node Simulation**

- Initialization: 1.52x speedup over Matlab with MEX functions
- Clutter Signal: 1.65x speedup
- Total Simulation: 1.62x speedup

**Simulation Speedup**

- Standalone speedup over Matlab with MEX functions:
  - 1.52x initialization
  - 1.65x clutter signal
  - 1.62x total simulation

- Standalone speedup is also nearly linear

- Will the speedup hold as more processing nodes are added?
Parallel Standalone (1-60 nodes)

- Space-based radar simulation
- 35 million cells in simulated range swath
- AFRL Heterogeneous High-Performance Computer cluster
  - 44 dual processor Intel Xeon 2.2 GHz processors
  - 4 Gb SDRAM per node
  - Gigabit/Myrinet connections
- Actual performance follows linear speedup

![Graph showing SBR Parallel Simulation Performance]

- Speedup Factor $T_1/T_p$
- Number of nodes
- Ideal linear speedup
- Actual speedup
Outline

• **Introduction**
  – radar system analysis
  – clutter signal model and simulation problem size

• **Implementations**
  – Matlab
  – MEX functions
  – parallel Matlab simulations
  – Standalone code

• **Performance Results**
  – Matlab
  – standalone

• **Summary and Future Work**
Summary

• Radar simulations are a valuable tool in analyzing radar performance
• High-fidelity radar simulations require a large number of operations
• The embarrassingly parallel nature of the problem makes parallel simulations simple
• Using MEX functions and computing clusters greatly improves simulation performance and allow higher-fidelity models to be implemented
• Using VSIPL provides a portable numerical library that can be optimized for various target platforms
Future Work

- Analyze task parallelization performance in conjunction with the data parallel implementation
  - PVL
  - VSIPL++ parallel

- FPGA
  - generally have higher memory bandwidth and sustained FLOPS than conventional microprocessors
  - some recent DSP systems employ FPGA coprocessors
  - the AFRL HHPC has a systolic array of 96 FPGA

- GPU co-processing/clusters
  - well suited to matrix algebra
  - have high memory bandwidth
  - high FLOP/dollar ratio