AFRL-AFOSR-VA-TR-2020-0102

Energy Aware Time Change Detection using Synthetic Aperture Radar on High-Performance Heterogeneous Architectures: A DDDAS Approach

Sanjay Ranka
UNIVERSITY OF FLORIDA
207 GRINTER HALL
GAINESVILLE, FL 32611-5500

07/20/2020
Final Report

DISTRIBUTION A: Distribution approved for public release.

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
20-07-2020	Final Performance	01 May 2015 to 30 Apr 2019

4. TITLE AND SUBTITLE
Energy Aware Time Change Detection using Synthetic Aperture Radar on High-Performance Heterogeneous Architectures: A DDDAS Approach

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-15-1-0047

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Sanjay Ranka

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF FLORIDA
207 GRINTER HALL
GAINESVILLE, FL 32611-5500 US

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 N. Randolph St. Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR RTA2

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-VA-TR-2020-0102

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Detecting areas of change in multiple snapshot images of a ground area (called change detection) is important in a variety of surveillance applications used by the Air Force and Department of Defense. For many such applications, images are constructed using Synthetic Aperture Radar (SAR). SAR collects information about a scene by repeatedly collecting the response from pulses transmitted toward an area of interest. A Backprojection algorithm is used to construct high quality 2-dimensional images from SAR data. The focus of this project is the development of novel sequential and parallel algorithms for change detection using SAR imagery that utilize modern architectures and minimize energy. This enables real-time change detection on an airborne device. When backprojection image frames are used for change detection, the computational burden can be mitigated by rendering areas of low activity in low resolution (called multiresolution processing). In this project, we extensively utilized this feature and developed novel change detection algorithms for multiresolution SAR.

15. SUBJECT TERMS
SAR Synthetic Aperture Radar, multi-pulse multi-resolution back-projection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			BLASCH, ERIK
Unclassified	Unclassified	Unclassified	UU		

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

				19b. TELEPHONE NUMBER *(Include area code)* 703-696-7311

Energy-Aware Time Change Detection Using Synthetic Aperture Radar On High-Performance Heterogeneous Architectures: A DDDAS Approach

---

Final Report

Contract No. FA9550-15-1-0047

30 April 2019

Sanjay Ranka, Ph.D. (PI) **Sartaj Sahni, Ph.D. (Co-PI)** **Mark S. Schmalz, Ph.D. (Co-PI)**

University of Florida
Department of CISE
Gainesville, FL 32611-6120

em: ranka@cise.ufl.edu
ph: 352-514-4213

Abstract

University of Florida's research in Energy Aware Time Change Detection Using Synthetic Aperture Radar On High-Performance Heterogeneous Architectures: A DDDAS Approach under this contract includes:

(1) Testing/Optimization of Video Synthetic Aperture Radar (VSAR) Hierarchical Reconstruction Algorithm (HRA) -- for efficient heterogeneous parallel multicore processor implementation.

***Summary*:** Multiresolution SAR reconstruction algorithms were developed, tested and optimized on the ORNL Titan supercomputer for energy-efficient multicore CPU/GPU architectures [Ahm12,Wija16, Wija17]. Performance exceeds our previous SAR reconstruction algorithms.

***DDDAS Support*:** Our VSAR HRAs optimize multi-frame SAR reconstruction to dynamically enhance performance on heterogeneous multicore architectures, based on time-varying scene and pulse dataset content. This implements DDDAS "Dynamic" and "Adaptive System" emphases.

(2) Development and Testing of VSAR Video Sequence Simulator -- for testing VSAR Reconstruction Algorithm and Change Detection Algorithm.

***Summary*:** A SAR pulse data simulator was developed, having shadowing and flare, sensor noise, communication channel or packet dropout, packet mis-placement, and erroneous discretization effects to support realism in simulated imagery. Our simulator generated SAR pulse data for testing our VSAR HRAs (discussed in 1), above), as well as testing our Change Detection algorithms.

***DDDAS Support*:** The VSAR Video Sequence Simulator produces copious amounts of BigData imagery with precisely controllable attributes, for "Data-Driven" emphasis of DDDAS.

(3) Development, Testing, Analysis, and Parallel Implementation of Change Detection (CD) Algorithm -- for Object Detection in Video SAR.

***Summary*:** We developed and demonstrated enhanced CD algorithms that exploited multiresolution hierarchical processing strategies inherent in our VSAR HRAs. Energy-efficient porting of VSAR HRAs and CD algorithms to heterogeneous clusters of multicore processors was demonstrated [Ahm12,Wija16,Wija17]. Application of our Hierarchical CD algorithm to simulated VSAR imagery from 1) and 2), above, successfully demonstrated efficient performance

given variable noise, channel effects, and resolution. The successful classification of moving and stationary targets incurred few false positives.

***DDDAS Support*:** Our CD algorithms successfully demonstrated "Dynamic", "Data-Driven", and "Adaptive" system processing, given time-varying environmental, scene, and processor constraints.

Keywords: Big Data, Synthetic Aperture Radar, Change Detection, High-performance computing (HPC), Multicore processors, Parallel computing

Table of Contents

Description **Page**

List of Figures **5**

List of Tables **7**

1 Summary **8**

2 Introduction **9**

3 Conformance to Schedule of Work **10**

3.1 Task 1 – Develop Multiresolution Change Detection Algorithms 10
3.2 Task 2 – Hybrid Processing 11
3.3 Task 3 – Energy-Aware Algorithms 11
3.4 Task 4 – Software Framework 12
3.5 Task 5 – Performance and Quality Benchmarking 12
3.6 Task 6 – Management, Reporting, and Dissemination 13

4 Methods, Assumptions, and Procedures **14**

4.1 Methods 14
4.2 Assumptions 15
4.3 Key Procedures 17

5 Results and Discussion: Hierarchical VSAR Reconstruction Algorithms **19**

5.1 Background: SAR Pulse Data, Image Formation, and Processing 19
5.2 Data and Image Partitioning Strategies 23
5.3 Implementation on Heterogeneous CPU/GPU Networks 28
5.4 Implementation on Intel Knights Landing Processors and their Clusters 36

6 Results and Discussion: Simulation and Change Detection **48**

6.1 Simulation of SAR Scenes 48
6.2 Change Detection and Hierarchically Partitioned SAR Reconstruction 50
6.3 Results of Applying Change Detection to Simulated SAR Imagery 57

7 Conclusions **61**

References **63**

List of Symbols, Abbreviations, and Acronyms **68**

List of Figures

Description **Page**

Figure 1. High-level view of a simulated SAR sensor in flight, with relevant variables. 19

Figure 2. Pseudocode formulation of sequential (naïve) backprojection algorithm. 23

Figure 3. Notional diagram of our method for partitioning pulse data ... 24

Figure 4. Four static schemes for distributing 100 identical resolution atoms ... 25

Figure 5. Dynamic range dimension partitioning selects pulse data to minimize memory latency as the sensor's viewing axis moves around the scene. 26

Figure 6. Example of multiresolution image tile partitioning ... 27

Figure 7. Illustration of Oak Ridge National Laboratory (ORNL) Titan supercomputer ... 28

Figure 8. Pulse- and image-based partitioning strategies for multi-GPU architectures. 29

Figure 9. High-level schematic diagram of DDDAS architecture for multiresolution SAR 29

Figure 10. List Assignment and Longest Processing Time algorithms for scheduling of SAR reconstruction computations on multiple GPUs … 30

Figure 11. Pulse- and image-based partitioning strategies for multi-GPU architectures. 31

Figure 12. Scalability of our SAR reconstruction algorithms with multiple GPUs. 31

Figure 13. Communication optimization for multiple GPUs. 32

Figure 14. Theoretical framework for performance optimization for multiple GPUs. 33

Figure 15. Model prediction error for performance optimization, where $N = 321$ cases. 33

Figure 16. Model prediction error (in percent) for CPU and GPU power consumption. 34

Figure 17. Modeling energy consumption versus runtime [for multi-GPU architectures]. 35

Figure 18. Predictive model for CPU power consumption, where f denotes frequency. 35

Figure 19. Overview of KNL architecture and memory modes. 38

Figure 20. KNL backprojection algorithm and test results for runtime (top graph) and power consumption (bottom graph) versus number of cores and threads. 42

Figure 21. KNL backprojection algorithm test results for energy consumption ... 42

List of Figures (cont'd)

Description **Page**

Figure 22. KNL backprojection algorithm performance results for different KNL modes. 44

Figure 23. KNL backprojection algorithm and test results for runtime (top graph) and energy consumption (bottom graph) versus resolution and number of pulses. 45

Figure 24. Conceptual summary of our findings re: computational optimization. 46

Figure 25. Example of projective distortion in aerial scenes ... 49

Figure 26. Example of simulated VSAR frames at various noise and channel error levels. 50

Figure 27. High-level view of change detection architecture employed in CD algorithm. 51

Figure 28. High-level view of supervoxel segmentation architecture in our CD algorithm. 52

Figure 29. Example of optical flow based target segmentation in our current CD algorithm. 53

Figure 30. Example of target localization using block analysis in our current CD algorithm. 54

Figure 31. Example of target localization using supervoxel based analysis in CD algorithm. 54

Figure 32. Schematic diagram of convolutional neural network (CNN) based approach to predicting object or target labels and locations via supervised learning. 55

Figure 33. Power consumption as a function of runtime for 4Kx4K SAR output image frame reconstructed from 5,000 input pulses via our algorithm run on the ORNL Titan architecture. 58

List of Tables

Description **Page**

Table 1. Power consumption as a function of runtime for 4Kx4K SAR output image frame reconstructed from 5,000 input pulses via our algorithm run on the ORNL Titan architecture. 27

Table 2. KNL backprojection algorithm and test results for runtime as a function of resolution. 43

1 Summary

Detecting areas of change in multiple snapshot images of a ground area (called *change detection*) is important in a variety of surveillance applications used by the Air Force and Department of Defense. For many such applications, images are constructed using Synthetic Aperture Radar (SAR). SAR collects information about a scene by repeatedly collecting the response from pulses transmitted toward an area of interest. A Backprojection algorithm is used to construct high quality 2-dimensional images from SAR data.

Backprojection is an algorithm of $\mathbf{O}(N^3)$ computational complexity that requires tens of Teraflops per second for the reconstruction of realistic image sizes. The focus of this project is the development of novel sequential and parallel algorithms for change detection using SAR imagery that utilize modern architectures and minimize energy. This enables real-time change detection on an airborne device. When backprojection image frames are used for change detection, the computational burden can be mitigated by rendering areas of low activity in low resolution (called *multiresolution processing*). In this project, we extensively utilized this feature and developed novel change detection algorithms for multiresolution SAR.

This project also supports dynamic, data-driven adaptive system (DDDAS) concepts and development. Our dynamic data-driven approach uses a real time feedback loop to reduce the total amount of computation required. To realize computational efficiency, we developed parallel approaches for our multiresolution change detection algorithm using a cluster of hybrid multicore processors (HMPs). Currently and throughout the foreseeable future, HMPs with both CPU and GPU cores are poised to deliver high computational throughput with low energy requirements.

For example, we achieved several hundred Gigaflops for a double-precision floating-point SAR computation on a Tesla GPU. This technology was incorporated into our parallel algorithms that scale upward linearly, realizing Teraflops to hundreds of Teraflops on networks of GPUs and HMPs – for example, on the Oak Ridge National Laboratory (ORNL) Titan supercomputing system.

The key scientific challenges that we addressed include intelligent task-to-core mapping, energy-aware task scheduling on networks of distributed multicore processors, and minimization of communication and synchronization cost via interleaving of computational and memory or high-

latency I/O tasks. Our scheduling algorithms and software framework are designed, and have been proven theoretically and in practice, to minimize the overall execution time and energy or power consumption of SAR image reconstruction algorithms via backprojection. This pragmatic objective has been achieved in this project while managing memory requirements of concurrent tasks, and turning off cores to reduce energy requirements. A modular software approach was designed to ensure portability and scalability, but a final commercializeable product was not developed, due to Year-4 funding being curtailed by AFRL personnel.

2 Introduction

The goal of finding targets of interest in multiframe SAR video sequences involves operations such as SAR data acquisition or simulation; SAR image reconstruction; as well as image-based detection, segmentation, indexing, search, and classification operations. In this project, University of Florida researchers developed SAR image reconstruction algorithms, change detection algorithms, algorithm-to-architecture scheduling technology, and performance measurement or estimation techniques and software for change detection (CD) from multiresolution SAR imagery and videos. This Final Report summarizes UF's work.

Section 3 provides verification of completion of the Statement of Work (SOW), on a per-task basis. Our methods, assumptions, and procedures are summarized in Section 4, which overview the technologies developed, the types of data employed, and how our developed technologies were implemented on various computing systems, from a single-GPU desktop computer to the very large ORNL Titan supercomputer.

Discussion of technical advances and results begins in Section 5, which emphasizes our development and testing of hierarchical video SAR (VSAR) reconstruction algorithms. We begin by overviewing the background of SAR pulse data acquisition, SAR image formation, and processing issues thereof. A key component of SAR image reconstruction is the development of an optimal or near-optimal pulse data and image partitioning strategy, which we achieved by a combination of pulse domain projection and image tiling. Section 5 also summarizes the implementation of our hierarchical VSAR reconstruction algorithms on single CPU and/or GPU architectures, networks of heterogeneous CPU/GPU processors, and HMPs such as Intel's Knight's Landing architecture.

Section 6 discusses our achievements in SAR pulse simulation, which was required for this project due to our very limited access to SAR data – much of which is restricted-access or classified. We overview the open literature's discussion of SAR pulse data simulation, as well as effects such as layover distortion and its remediation. Simulation results that are representative of work done for this project are exemplified.

Section 7 presents a discussion of our change detection algorithm development, firstly as a conceptual and theoretical process based on hierarchically partitioned SAR data and reconstructed SAR imagery. We then discuss the status of change detection research as-of the end of our Year-3 funding – our Year-4 funding having been curtailed by AFRL personnel. Conclusions and suggestions for future work are given in Section 8.

3 Conformance to Schedule of Work

In order to demonstrate completion of the schedule of work (SOW) proposed for this project, within the limitations of reduced time and support monies due to curtailment of our Year-4 funding increment, we present the following task-based decomposition of the SOW, with explanation of work performed.

3.1 Task 1 - Develop Multiresolution Change Detection Algorithms

We researched and developed multiresolution change detection algorithms for simple (planar) terrains. As discussed in Sections 1-8, the objective of multiresolution change representation is to render areas of probable change at high resolution, areas of hypothesized change at medium resolution, and areas of little change or no meaningful change at low resolution. Testing of our algorithms was performed using in-house (simulated) and AFRL SAR datasets, to verify performance claims. Although our SAR simulator development effort strove to accommodate complex (corrugated) terrains, and to account for effects of shadowing by terrain objects at low- or grazing-incidence look angles, we were not able to fully develop this capability due to curtailment of Year-4 funding. Throughout the project, we focused on meta techniques for SAR reconstruction algorithm development that incorporated energy, performance and accuracy constraints. This development effort, together with its associated theoretical modeling tasks,

allowed us to choose an appropriate algorithm or technical approach (among several possibilities) based on multiple objectives.

***Conformance to SOW*:** Task 1 was completed on schedule, with the exception of incorporating corrugated terrain models into our SAR simulator, as well as compensation for effects resulting from corrugated terrain into our SAR reconstruction algorithms.

3.2 Task 2 - Hybrid Processing

In this task, we developed and implemented strategies for mapping task partitions called *atoms* to computational resources, for CPU/GPU clusters, HMPs, clusters of HMPs, and large machines comprised of many such clusters.

In particular, we researched and developed algorithm-to-architecture mapping technology for multiresolution SAR reconstruction and change detection applications mapped to simple (CPU-GPU) devices. We also considered HMPs such as the Intel Knight's Landing architecture as well as hypothetical processors that we modelled, which combine a multicore CPU with multiple GPUs. We also conducted experiments with the Nvidia Tesla and Kepler GPUs, developing implementations of SAR backprojection and change detection algorithms for selected configurations of these different architectures. Testing was performed using in-house codes and in-house or AFRL datasets, to verify performance claims. Test platforms ranged from desktop systems with a multicore CPU and/or GPU to the ORNL Titan system having many heterogeneous clusters.

***Conformance to SOW*:** Task 2 was completed on schedule**.**

3.3 Task 3 – Energy-Aware Algorithms

In this task, we researched and developed algorithms for controlling energy consumption of multicore processors, by shutting down unused cores on CPUs, GPUs, or HPUs that supported this control feature. We extended and enhanced our energy consumption control software to implement a static algorithm with dynamic voltage scaling (DVS) that was subsequently revised to yield a dynamic algorithm with DVS. Due to curtailment of our Year-4 funding, we were not able to fully integrate these developments into our software framework, in prototype form suitable for commercialization.

***Conformance to SOW*:** Task 3 was completed on schedule, with the exception of complete

integration of our developments into a commercializeable software prototype – which was not completed due to curtailment of our Year-4 funding**.**

3.4 Task 4 – Software Framework

We were able to produce workable code that supported our research in Tasks 1-3 and Task 5 (described below). Due to curtailment of our Year-4 funding, we were not able to incorporate our research codes into a software prototype that could be commercialized for public release and marketing.

***Conformance to SOW*:** Task 4 was begun but only 30 percent completed (commercializeable prootype not produced), due to curtailment of our Year-4 funding.

3.5 Task 5 – Performance and Quality Benchmarking

In this task, we conducted incremental tests of our software (as developed through Year-3, from Tasks 1-3) on synthetic imagery generated in-house, on single multicore CPUs, GPUs, and the Intel Knight's Landing HMP processor. We subsequently expanded these tests to include realistic SAR data provided as Government furnished information (GFI), with computation on clusters of CPUs, GPUs, and/or HMPs. We also tested our software on a large machine (ORNL Titan) comprised of multiple heterogeneous clusters of multicore CPUs, GPUs, and HMPs. Performance metrics emphasized runtime, I/O and memory access time, space complexity, energy consumption as a function of time, and overall power consumption. Although we developed change detection (CD) algorithms that exploited the multiresolution nature of our SAR reconstruction algorithms, due to curtailment of our Year-4 funding we were not able to fully characterize these CD algorithms in terms of a suite of metrics. For example, given complete funding for this project, accuracy metrics for change detection would have included probability of detection (Pd), rate of false alarms (Rfa), percentage correct classification, and so forth.

***Conformance to SOW*:** Task 5 was carried out to the extent of characterizing performance of our research codes for multiresolution SAR image reconstruction on single multicore processors and clusters of such processors. Results were published in several conference and journal papers. However, detailed characterization of our change detection algorithm was not completed in detail, due to curtailment of our Year-4 funding.

3.6 Task 6 – Management, Reporting, and Dissemination

Throughout the project, we provided coherent management of research, development, and test/analysis activities, as well as technical interchange meetings and periodic technical or budget reports. Prototype software release and/or commercialization, as well as information dissemination via website, was not conducted due to curtailment of funding in Year-4.

***Conformance to SOW*:** Task 6 was completed as scheduled, with the exception of prototype software release, dissemination via website and/or commercialization – the latter tasks not being completed due to curtailment of funding in Year-4.

4 Methods, Assumptions, and Procedures

4.1 Methods

In this project, we conducted research according to the following four methods:

1. Algorithm Design, Development, and Implementation
2. Algorithm Test and Optimization
3. Data Simulation
4. Performance Measurement and Analysis

4.1.1 Algorithm Design, Development, and Implementation focused on efficient algorithms for (a) SAR image reconstruction, (b) mapping of algorithms to processor architectures with optimal scheduling and load balancing, and (c) change detection.

SAR image reconstruction algorithm accepts sensor parameter data such as altitude and look angle at each radar pulse and a pulse return representation. Given such data for multiple pulses, the algorithm then produces a reconstruction of the scene within the SAR sensor footprint.

Optimal scheduling and load balancing algorithms map an application such as SAR image reconstruction to a network of heterogeneous processors. Constraints of energy and power consumption can be applied, as well as spatial resolution constraints. The result of this algorithm is an optimal or near-optimal mapping that meets space, time, error, energy, and power (STEEP) constraints.

Change detection algorithm accepts a multiframe SAR video (also called VSAR) sequence with sensor parameters including but not limited to altitude and look angle, then applies target detection technology to highlight possible (candidate) targets within the VSAR sequence.

4.1.2 Algorithm Test and Optimization focused on testing algorithms developed under the scope of this project, to determine their performance with respect to the following parameters, or a subset thereof: Space (including memory requirement), Time (wall-clock, CPU, GPU, and I/O times), Error (for example, reconstruction error inherent in mission-critical scene objects or regions), Energy and Power consumption (as determined by on-board processor monitoring routines).

Algorithm optimization emphasized refactoring of computer software codes to more efficiently map an application such as SAR image reconstruction to a processor architecture, for example, a network of heterogeneous processors, under constraints of execution time, energy and power consumption as well as spatial resolution. The purpose of this optimization effort is to optimally or near-optimally map algorithms to processing architectures in a manner that meets space, time, error, energy, and power (STEEP) constraints for application execution.

4.1.3 Data Simulation produced synthetic test data for the SAR image reconstruction algorithm and for the change detection algorithm. SAR pulse data were simulated by constructing a 3D model of a scene using POVray™ software, then measuring the distance from each point in the scene to a simulated radar pulse emitter/receiver pair. The distances were accumulated in a histogram-like data structure that was scaled for physical constants including the speed of light. For the change detection application, simulated pulse data were perturbed to simulate sensor noise effects and transmission-channel dropouts. This resulted in a noisy reconstructed SAR image, which was used to challenge and probe the detection abilities of the various change detection algorithms under test.

4.1.4 Performance Measurement and Analysis focused on measuring quantitites such as space, time, SAR image reconstruction error, energy and power consumption. The purpose of these measurements and the statistical analyses we conducted on such data is to support development of algorithms to optimally or near-optimally map SAR applications to processing architectures in a manner that meets space, time, error, energy, and power (STEEP) constraints for application execution. This process was overviewed in Section 4.1.2.

4.2 Assumptions

Our research and development effort in this project was based on the following primary assumptions:

1. *Availability of Data and Metadata* – We assume that SAR pulse data and associated metadata are available as Government-furnished information (GFI), or can be simulated, in a manner that accurately portrays the physical entities being sensed. This assumption is foundational to the study described in this report – without accurate, reliable SAR

pulse data, development of applications to reconstruct SAR imagery would potentially yield inaccurate results.

2. *SAR Reconstruction by Superposition* – It is assumed that the SAR reconstruction algorithms described in [Gor10] and/or furnished as GFI produce a visually acceptable reconstruction of a scene that has been sensed by SAR data. It is further assumed that the metadata (e.g., sensor position and look angle) that accompany a SAR pulse dataset are sufficiently accurate to support such reconstruction.
3. *Knowledge of Processing Architecture Parameters* – In order to optimize a given implementation of an application or algorithm on a specific processing architecture, one needs to know the parameters of the architecture (e.g., number and configuration of processing nodes or clusters, attributes of processors and support software, and so forth). We assume that such parameters can be determined *a priori* and that other, dynamic parameters such as processor status, queued or scheduled workload, and so forth, can be determined by querying the processors, nodes, or clusters involved. This assumption facilitates dynamic monitoring of architecture status and transmission of such knowledge to processes that dynamically control optimal scheduling and/or load balancing.
4. *Ergodicity of Architectural Parameters per Sensing Epoch* – We further assume that a given processing architecture can be characterized in terms of its parameters, which do not change statistically over a given interval of time called a *sensing or processing epoch*. For example, if one is processing SAR data with four clusters, each comprised of one CPU and two GPUs, then the functional status of these processors is not expected to change temporally over the time required to process the given SAR pulse dataset. For example, this assumption ensures that N CPUs and GPUs processing SAR data will remain functional and available for processing (within dynamic load constraints) for the duration of the processing epoch. Although our optimized scheduling and load balancing algorithms developed herein could be extended to include dynamically changing processor status (e.g., function drop-outs, restarts including process rollback, etc.) and dynamic processing network connectivity, such behaviors were not the primary focus of this study and are therefore omitted by this assumption.

We further assume that certain types of processors are able to measure their own energy or power consumption, and that software is available to translate memory and energy or power consumption parameters for each processor into machine-tractable test results data. This assumption is essential for dynamically optimizing application software implemented on such architectures in terms of space, time, energy and power.

5. *Detectability of Targets in SAR Change Detection Applications* – For purposes of developing and refining our change detection algorithms, we assume that a given VSAR sequence contains moving or other time-varying targets that differ statistically from their respective backgrounds. For example, consider a vehicle moving along a road or city street, or a person moving slowly in foliage being blown about by a breeze.

4.3 Key Procedures

A wide variety of procedures were employed in this study. The two principal procedures for algorithm development, testing, and enhancement are described below: (1) Static and dynamic algorithm analysis, and (2) Pareto-optimal modeling.

4.3.1 Algorithm Analysis. Central to the successful conduct of the research and development tasks inherent in this project is our analysis of algorithms that comprise applications under development or optimization. This procedure has significant importance, and contains the following procedures:

1. *Static Profiling* involves determining the number and type of operations in each portion of the algorithm, and throughout a given algorithm. Space (memory) requirement is analyzed to determine if processor memory is sufficient for declared data structure, or if paging or swapping strategies are indicated. The algorithm is then transformed into a dataflow graph, which serves as the basis for further characterization (dynamic profiling and analysis) in terms of space, time, error, energy and power constraints. Static profiling can also reveal memory access patterns in terms of data structural access such as array reads and writes. In particular, when an array indexing scheme (e.g., row- or column-major) does not match a target memory storage scheme (resp. column- or row-major), then significant memory latencies can result.

2. *Dynamic Profiling* utilizes one or more runtime profilers to determine the actual mix of operations and data structural partitions employed in the execution of a given algorithm over prespecified data. Such profilers are capable of probing an algorithm while processing is occurring (on real or simulated architectures) to determine actual space, time, energy, and power consumption. An audit trail can be generated that depicts an algorithm's runtime behavior, for subsequent analysis to determine statistical performance measures and to detect behavior patterns (for example, patterns of memory access operations). This procedure is also central to the measurement of execution time in terms of CPU, GPU, I/O, and total-elapsed (wall clock) times. From these temporal measurements, we are able to determine effective strategies for time-optimized algorithm enhancement.
3. *Energy and Power Analysis* – Given sufficient support for processor performance measurement, we measured GPU energy and power consumption using the *gprof* profiler. This supported joint optimization for space, time, energy, and power consumption.

4.3.2 Pareto-Optimal Modeling. An additional technique employed in algorithm performance optimization involved executing many runs of an algorithm with different algorithmic and architectural parameters. These runs were then assembled and represented in graphical format. A multilinear representation called a *Pareto curve* was constructed that bounded the best achieved optimization result for a given value of the independent variable(s). This represents the optimization frontier, beyond which further improvement is not possible. Pareto modeling was employed in demonstrating improvements achieved in the ORNL Titan implementation of our SAR image reconstruction algorithm, as discussed in Sections 5 and 6.

5 Results and Discussion: Hierarchical VSAR Reconstruction Algorithms

In this section, we discuss our development and testing of hierarchical synthetic aperture radar (SAR) and video SAR (VSAR) algorithms. Section 5.1 provides background in terms of pulse data, image formation processes, and how the SAR reconstruction algorithms are formulated and designed to be processed on computer. Section 5.2 discusses one of the key advantages of our approach, namely, our optimized techniques for partitioning pulse data and image arrays to effect a tensor-product-based computation of SAR image reconstruction. In Section 5.3, we present our results pertinent to implementation of our SAR algorithms and optimization of their performance on networked clusters of CPU-GPU nodes. Section 5.4 summarizes implementation of our SAR algorithms and optimization of their performance on the Intel Knight's Landing hybrid multiprocessor architecture.

5.1 Background: SAR Pulse Data, Image Formation, and Processing

In this study, SAR data acquisition in the field is modeled as occurring in the context of a monostatic or bistatic emitter/receiver pair that has a known trajectory (in the monostatic case) or independent trajectories for the emitter and receiver (bistatic case), as diagrammed schematically in Figure 1. Target(s) in the sensor's field of view, if present, have a given bidirectional reflectivity distribution function (BRDF) and are referenced in terms of a ground plane **X**.

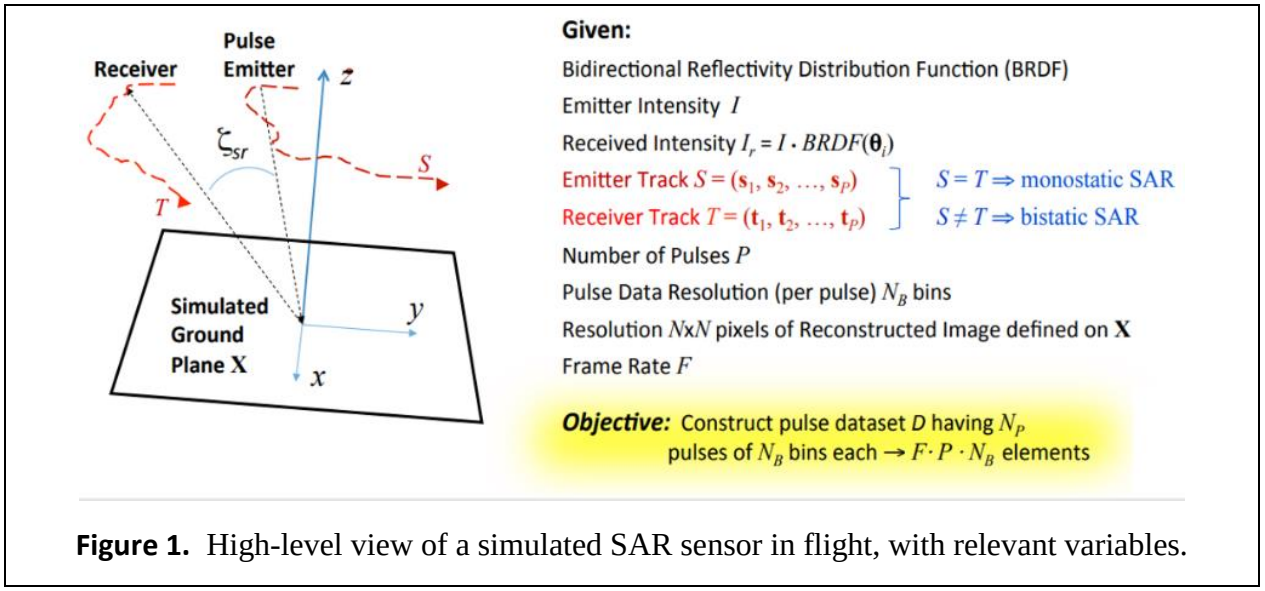


Figure 1. High-level view of a simulated SAR sensor in flight, with relevant variables.

Given the number of simulated SAR pulses P, the emitter follows a three-dimensional track denoted by $S = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_i, \ldots, \mathbf{s}_P)$, where $\mathbf{s}_i$, $i = 1 \,..\, P$, is in world coordinates $\mathbf{W}$; and receiver track is given by $T = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_i, \ldots, \mathbf{t}_P)$, where $\mathbf{t}_i \in \mathbf{W}$. When $S = T$, we have monostatic SAR in the context of our simulation model, otherwise bistatic configuration. Given simulated SAR emitter intensity I, the simulated SAR receiver has look angle $\boldsymbol{\Theta} = (\theta,\phi)$ and received intensity $I_r = I \cdot BRDF(\boldsymbol{\Theta})$. Each of the P received pulses is assumed to be discretized into N_B bins (which can also be denoted elsewhere in this report as B, for notational simplicity).

In a simple case, for the i-th pulse, where $i = 1 \,..\, P$, reconstruction of the SAR image involves computation of a backprojection algorithm that produces an NxN-pixel real-valued image $\mathbf{a}_i$ on $\mathbf{X}$, denoted mathematically by $\mathbf{a}_i \in \mathbb{R}^{\mathbf{X}}$. In this case, video SAR (VSAR) is modeled as a sequence of P reconstructed image frames $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_i, \ldots, \mathbf{a}_P)$. In other less simple cases, groups of adjacent pulses can be aggregated to produce $\underline{P} < P$ pulses (sometimes called *superpulses*), and the VSAR sequence is denoted by $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_i, \ldots, \mathbf{a}_{\underline{P}})$ with $\underline{P}$ frames.

In our basic SAR simulation model, in addition to input metadata including but not limited to $\boldsymbol{\Theta}$, S, T, I, P (or $\underline{P}$) and $BRDF$, the input *pulse data matrix* has P rows and N_B columns. These data are input to the SAR reconstruction algorithm, which produces the output NxN-pixel reconstructed image frame $\mathbf{a}_i$ or the SAR video sequence $\mathbf{a}$ that contains P NxN-pixel frames.

5.1.1 Simple Model of SAR Imaging and Reconstruction

In our simple model employed in this study, SAR reconstruction finds the distance to each ground object from the receiver antenna's phase center by transmitting electromagnetic waves that are reflected by ground objects according to the objects' respective BRDFs. Given transmitted intensity I and reflectivity r computed from look angle $\boldsymbol{\Theta}$ and a ground object's BRDF, the received intensity I_r is given by

$$I_r = I \cdot r \cdot e^{-j2\pi f t} \,, \tag{1}$$

where $j = (-1)^{1/2}$, f denotes the pulse carrier frequency and the round-trip travel time from emitter to the ground object to the receiver is denoted by t. Given the distance d from emitter to ground object and speed of light c, one can write

$$I_r = I \cdot r \cdot e^{-j2\pi f(2d/c)} \,. \tag{2}$$

Given the receiver location $\mathbf{t}_i = (x_r, y_r, z_r)$ and ground object location $\mathbf{g} = (x(t), y(t), z(t))$, the Euclidean distance between the receiver and the ground object can be calculated as

$$d = \sqrt{(x(t) - x_0)^2 + (y(t) - y_0)^2 + (z(t) - x_0)^2} \quad (3)$$

Letting the pulse carrier frequency vary linearly from f_{min} to f_{max}, and denoting by N_f the number of frequency samples per pulse, frequency step size is given by

$$\Delta f = \frac{f_{max} - f_{min}}{N_f} . \quad (4)$$

For a single pulse having carrier frequency f_k, the received intensity is given by

$$I_r(f_k) = I \cdot r \cdot e^{-j2\pi f_k(2d/c)} , \quad (5)$$

and the received intensity for the i-th pulse p_i is given by

$$I_r(p_i) = \sum_{k=0}^{N_f} I \cdot r \cdot e^{-j2\pi f_k\left(\frac{2d}{c}\right)} . \quad (6)$$

The differential range (ΔR) is defined as the distance between an object and the scene origin. The phase associated with the object at the scene origin is set to zero for all frequencies. Given d_a, the distance between the scene origin and the receiver, the distance of the object may be referenced to zero and differential range may be calculated as $\Delta R = d - d_a$, where d was expressed in Equation (3).

Alias response controls the range of the imaged scene. Alias free time range is given as $1/\Delta f$ where the frequency step size Δf was given in Equation (4). Maximum alias free range extent W_r is the distance that a pulse can travel in the alias free time range [Cruz14], calculated as

$$W_r = \frac{c}{2\Delta f} . \quad (7)$$

Given total pulse bandwidth $B_p = (N_f - 1) \cdot \Delta f$, range resolution [Mor13] can be expressed as

$$\delta_r = \frac{c}{2B_p} . \quad (8)$$

5.1.2 Naïve Backprojection Algorithm

Backprojection integrates intensities of the collected radar pulses over a specific time interval to produce a reconstructed output image [Sou99,Des92,Ula03]. For a single ground object situated

at distance d from the scene center, the amplitude of a pulse acquired at carrier frequency f_k is given by

$$A = I \cdot e^{-j2\pi f_k(2d/c)}, \tag{9}$$

The distance between the scene center and maximum alias-free range extent can be divided into equally-sized small distances called range bins. For an object at distance d for all frequency values, the amplitude value $A(m)$ at the m-th discrete range bin is computed as

$$A(m) = \sum_{k=1}^{N_f} I_r(f_k) \cdot e^{j2\pi f_k(2d/c)} \quad . \tag{10}$$

Total amplitude value for an object at distance d for all frequency values and emitted pulses is the sum of $A(m)$ for each pulse, as follows:

$$A_{total} = \sum_{i=1}^{P} \sum_{k=1}^{N_f} I_r(f_k, p_i) \cdot e^{j2\pi f_k(2d/c)} . \tag{11}$$

Complexity of Naïve Backprojection: Per Equation (11) complexity is $\mathbf{O}(N^2 \cdot P \cdot N_f)$, where Equation (11) is necessarily computed for each of the N^2 pixels in the output (reconstructed) SAR image. In [Gor10], it is modified to run in $\mathbf{O}(N^2 \cdot P)$ time using an inverse discrete Fourier transform (IFFT), which can be expressed as

$$A_{total} = \sum_{i=1}^{P} N_{\text{fft}} \cdot \text{fftshift}[\text{ifft}(I_r(f, \tau)] \cdot e^{j4\pi f(\Delta R/c)} \quad . \tag{12}$$

where N_{fft} denotes the size of the IFFT and τ denotes a phase shift value.

Algorithm Formulation. Algorithm 1, listed in pseudocode form in Figure 2, presents a basic sequential computation of backprojection. Inputs for the algorithm are the pulse data array, which contains phase history; (2) location array containing S, the 3D coordinates of the receiver at each pulse; N_{fft}, previously defined; W_r, the maximum alias free range extent defined in Equation (7). The structure of Algorithm 1 is simple: there are two main loops that can be parallelized, which are (1) a loop that iterates through every pulse in the pulse array, and (2) a loop that calculates the value of each pixel. This leads to the discussion of data partitioning in Section 5.2.

Algorithm 1 Algorithm for sequential backprojection

Input: Pulse Array, Location Arrays, N_{fft}, W_r
Output: $N_x \times N_y$ image

1: **for** $i = 1$ to Number of Pulses in Pulse array **do**
2: Calculate IFFT for the pulse
3: Perform FFTShift for the result of IFFT and store as pData array.
4: **for** $j = 1$ to $N_x \times N_y$ **do**
5: Calculate ΔR w.r.t. pixel location.
6: let $index = \Delta R \cdot \frac{N_{fft}}{W_r}$ //Find the range bin that is closest to ΔR.
7: let $v1 = pData[index]$ *and* $v2 = pData[index+1]$
8: Interpolate $v1$ and $v2$ to get the value (v) that corresponds to ΔR.
9: let $v3 =$ multiplication of v with phase correction.
10: Accumulate the calculated value for the current pulse $(v3)$ with that value calculated for previous pulses.
11: **end for**
12: **end for**
13: Normalize computed pixel values.
14: **return**

Figure 2. Pseudocode formulation of sequential (naïve) backprojection algorithm.

5.2 Data and Image Partitioning Strategies

In practice, the pulse array size P can be large (e.g., on the order of 10^3 or 10^4 – or more). Depending on desired range resolution (distance increment from the emitter to the target), the number of range bins N_B can be of similar order. In practice, the ground plane within a 1 km^2 to 10 km^2 sensor footprint may be discretized into cells ranging in resolution from one meter or smaller. Thus, the image dimension N (equal to N_x or N_y in Algorithm 1, above) can be on the order of 4K to 16K (or more), where "K" denotes $2^{10} = 1024$.

This means that the pulse data matrix can have 10^8 or more elements (e.g., on the order of gigabytes to tens of gigabytes of data), while a single frame of the output VSAR sequence can have at least $(4K)^2$ pixels $\geq$ 64MB (megabytes) at single precision floating-point representation. For shared-memory parallel architectures with small local cache for each processor, this

indicates prohibitive storage requirements. Thus, partitioning of the pulse matrix and reconstructed image is required, to produce smaller data arrays that can be stored in local memory, thereby reducing memory access latency. Therefore, the following partitioning strategies were formulated and tested.

5.2.1 Data Partitioning Strategies

In tensor-product based problems such as SAR image reconstruction, one encounters two datasets A and B that are combined to produce an output data structure C. As discussed following Equation (11), the complexity of the naïve algorithm is $\mathbf{O}(|A| \cdot |B|)$, where $|A|$ denotes the cardinality (size) of A. As noted above, SAR image reconstruction can have input data structures in the size range $|A| \cdot |B| \approx 10^{16}$ bytes. Thus, we consider partitioning both input datasets, as diagrammed notionally for each method (pulse and data partitioning) in Figure 3.

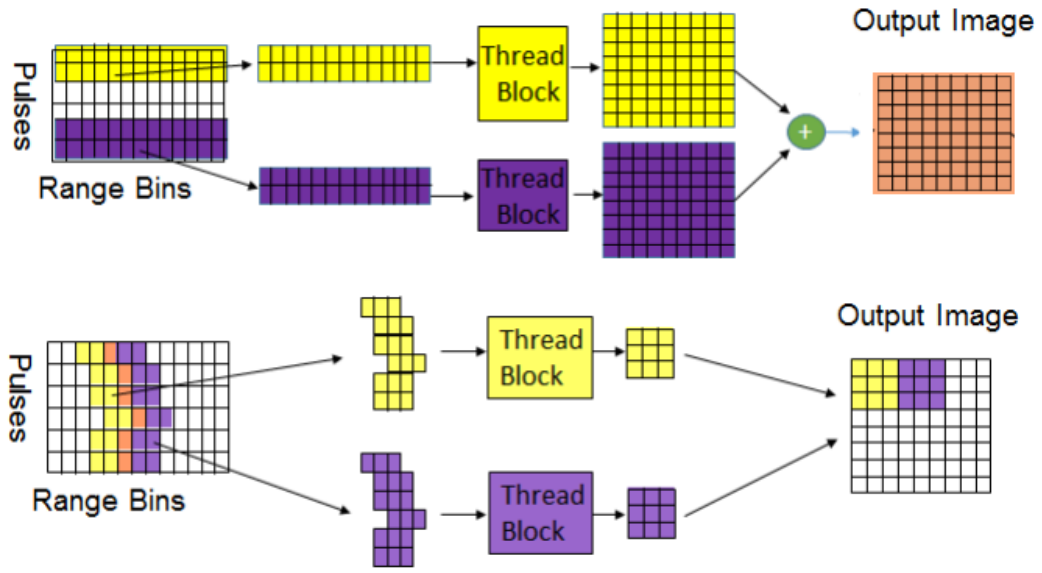


Figure 3. Notional diagram of our method for partitioning (top) pulse data, and (bottom) image data via tiling, for efficient SAR image reconstruction.

Thus, in practice, backprojection is decomposed (1) along the output image dimension, so each processing device renders an output image tile using all pulse data, or (2) each processing device renders an image using a subset of the pulses. Output image tiling is crucial to efficient

algorithm performance – the selection of optimal tile size is done empirically. Pulse data selection is likewise key to reconstruction algorithm efficiency. However, due to the size of the data structures involved, neither of these partitioning methods can be employed individually as memory requirements are too large for efficient retrieval from local memory, so we therefore combine pulse data and image data partitioning. In practice, we prefetch pulses that contribute to each output (tiled) pixel.

A further consideration involves the effect of sensing geometry on pulse selection and prefetching. Figure 4 illustrates the various pulse selection methods we developed and tested. Range dimension partitioning attempts to select pulse data that correspond to a given range along the look angle direction from the receiver's entrance pupil (numbers labeled "Bin" denote relatively-indexed range bins). Cross-range dimension partitioning selects data at 90 degrees to the look angle. Locality partitioning requires less computation than range- or cross-range-dimension partitioning, as a working set is selected based on previous access patterns. Round-robin partitioning configures selection choices into an interleaved checker-board like pattern.

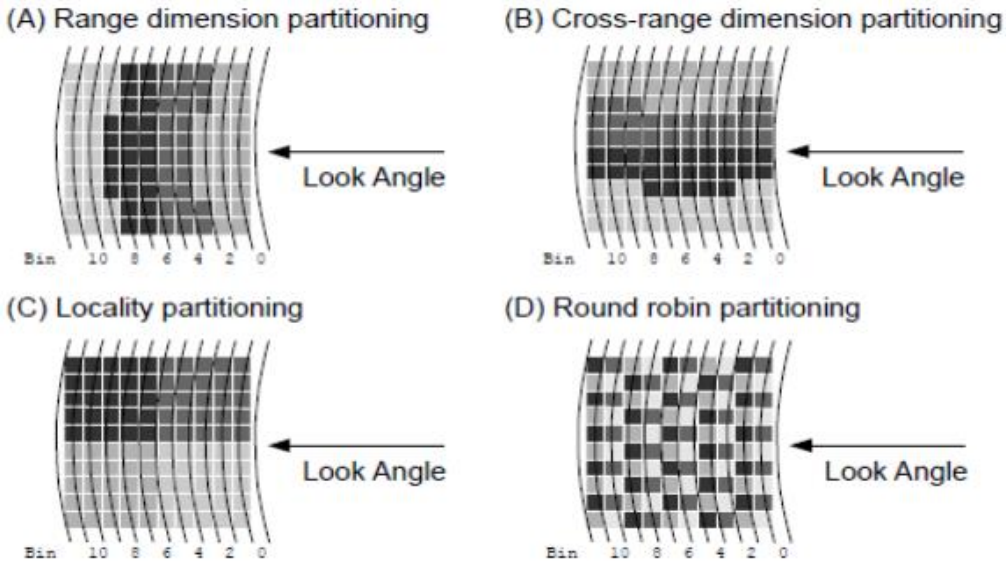


Figure 4. Four static schemes for distributing 100 identical resolution atoms, each with an equivalent pulse set, to four homogeneous processing devices.

The orientation of the sensing geometry with respect to the selection method also plays a significant role in pulse data selection. In Figure 5, two dynamic selection methods are diagrammed notionally – these are taken from a large number of dynamic selection strategies that we explored. For example, in the case of locality partitioning (Figure 4c), if the location of pulse data was dynamically correlated with the portion of a locality-based partition in which the backprojected pulse data were likely to occur (as in the right-hand portion of Figure 5), then significant memory access efficiency could be realized – versus retrieving the entire locality partition (shown as the block boundary or as the highlighted data in the left-hand portion of Figure 5).

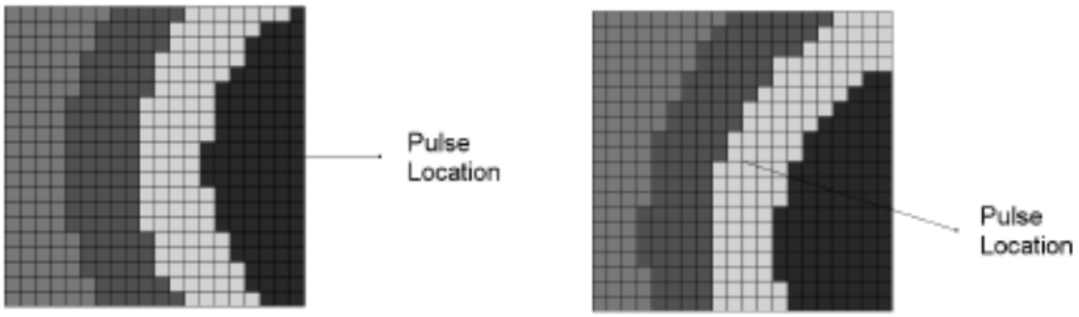


Figure 5. Dynamic range dimension partitioning selects pulse data to minimize memory latency as the sensor's viewing axis (denote by "pulse location") moves around the scene.

5.2.2 Effect of Multiresolution Image Partitioning on Backprojection Algorithm Performance

Further efficiencies in data communication, access, and processing can be realized in SAR image reconstruction via the use of multiresolution techniques for output image tiling. For instance, when a change detector is applied to a reconstructed VSAR sequence, it is possible to estimate which areas of a successive image frame may have activity of interest. These areas can then be reconstructed at higher resolution. Similarly, a statistical estimate could be employed to detect areas of higher (resp. lower) spatial variance that could be reconstructed at higher (resp. lower) resolution. As an example, consider a sandy field (low spatial variance) versus an urban street (higher variance due to the presence of pavement, sidewalk, building, vehicular, and pedestrian

objects). If one prefers to detect pedestrians, then it makes sense to reconstruct the urban sidewalk or street portions of a scene at higher resolution.

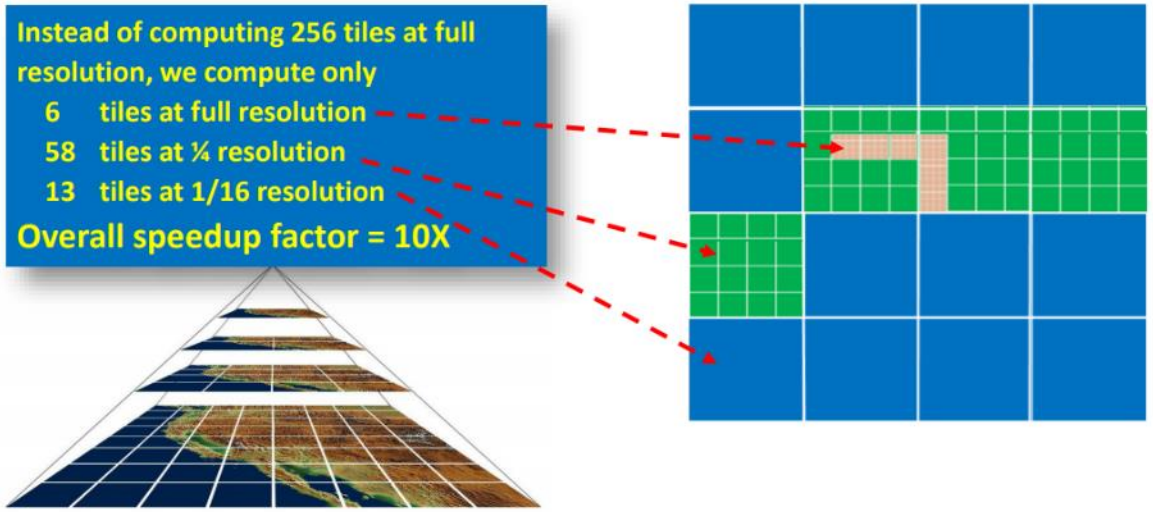


Figure 6. Example of multiresolution image tile partitioning via a combination of full, 1/4, and 1/16 of full resolution. The full-resolution tiles would represent areas of greater interest, e.g., higher variance, more features of interest, or greater estimated featural change.

In tests conducted on the ORNL Titan supercomputer, we verified that multiresolution processing significantly speeded up SAR reconstruction processing, as shown in Table 1. Here,

Table 1. Example performance data for our multiresolution image partitioning scheme, on the ORNL Titan supercomputer. All execution times are wall-clock time, in seconds.

Reconstructed Image Size (pixels)	Number of SAR Pulses	High Resolution Execution Time (sec)		Medium Resolution Execution Time (sec)		Low Resolution Execution Time (sec)	
		Single Res	Multi-Res	Single Res	Multi-Res	Single Res	Multi-Res
8192x8192	1000	10.5	4.385	2.7	1.18	0.51	0.28
	5000	53.75	22.36	13.36	6.01	2.56	1.37
4096x4096	1000	2.79	1.18	0.72	0.303	0.16	0.073
	5000	13.72	5.83	3.52	1.56	0.76	0.36

manual tuning was employed to increase performance by (a) reducing register usage by reordering instructions, (b) optimizing global memory read operations via usage of a *float4* array versus four *float* arrays, (c) using a single array to store coordinates specific to sensor platform location, and (d) increasing the L1 cache size to 48KB. We achieved a maximum throughput of 700 GFLOPs per GPU. Our implementation benefited from our novel architecture for data partitioning and load balancing that is discussed in the following subsection.

5.3 Implementation on Heterogeneous CPU/GPU Networks

Our algorithms and partitioning approaches for SAR image reconstruction were tested on multi-GPU networks, for example, the ORNL Titan supercomputer illustrated in Figure 7. Partitioning

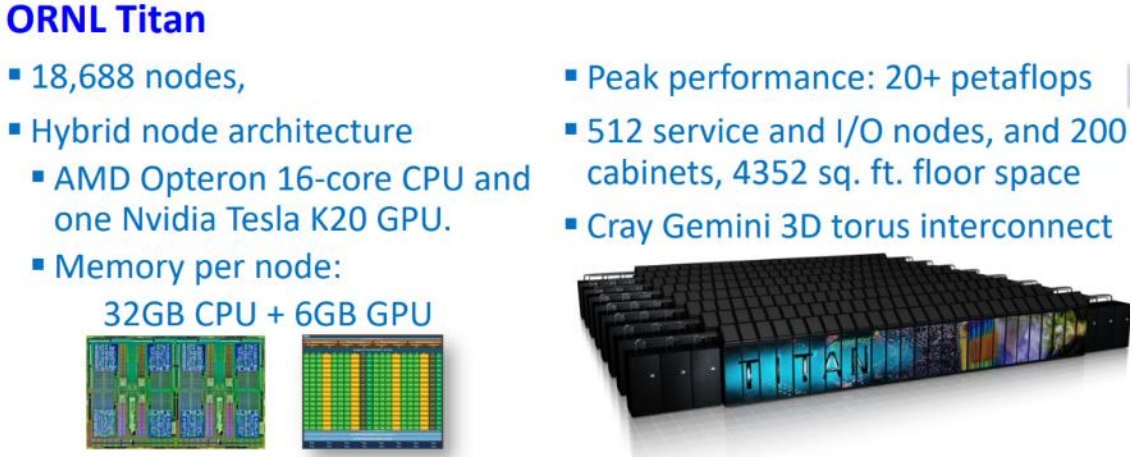


Figure 7. Illustration of Oak Ridge National Laboratory (ORNL) Titan supercomputer employed in the multi-GPU architectural tests of our SAR image reconstruction algorithms.

strategies for multiple GPUs built on the findings of our research with single GPUs controlled by a CPU. For example, the multi-GPU partitioning strategies for pulse and image (tile-based) emphases are illustrated schematically in Figure 8. Here, the single-GPU pulse- and image-based partitioning strategies of Figure 3 are replicated over multiple GPUs. These techniques are instantiated in the architecture diagrammed schematically in Figure 9, whereby a master control module (which can be omniscient, centralized or distributed) controls all functions of the SAR image reconstruction algorithm and (not shown in Figure 9) the change detection algorithm.

This control and execution methodology can be distributed over multiple networked GPUs in architectures from a desktop multi-GPU system to a supercomputer such as the ORNL Titan. In

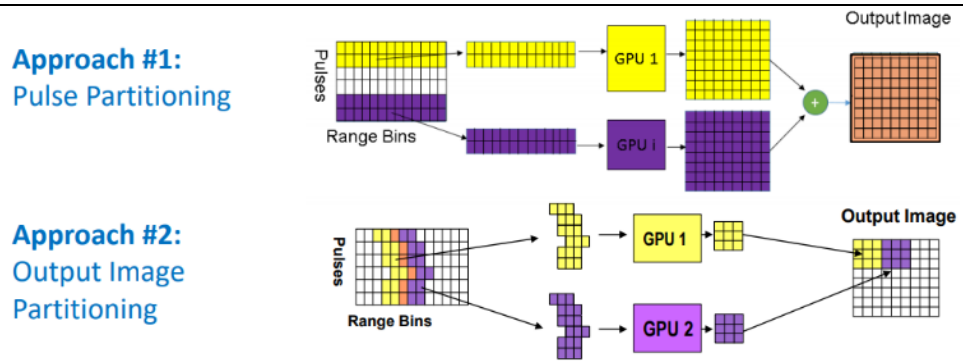


Figure 8. Pulse- and image-based partitioning strategies for multi-GPU architectures.

Figure 8, multiresolution images create load imbalances, whereas tile distribution advantageously balances computational load. Bin packing approaches are also employed.

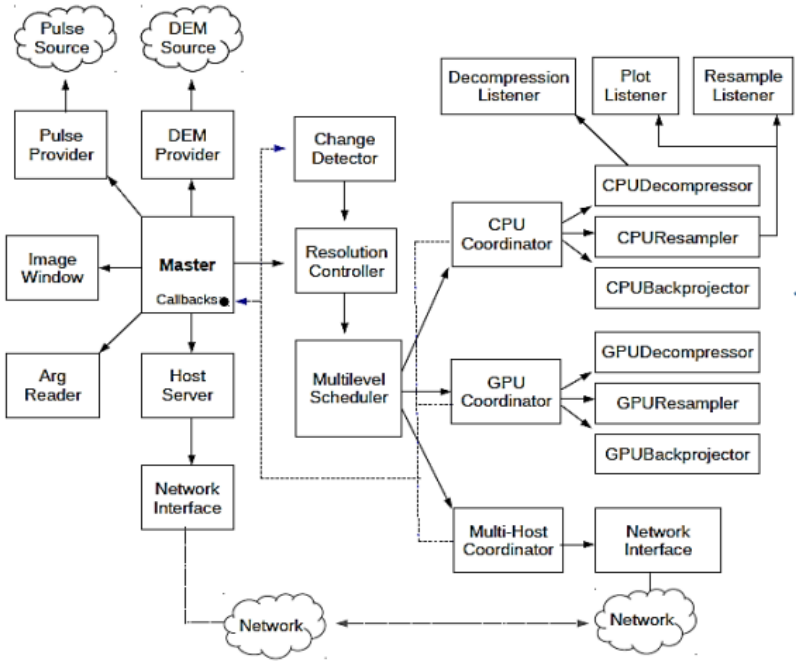


Figure 9. High-level schematic diagram of DDDAS architecture for multiresolution SAR image reconstruction on multiple GPU architectures.

In the architecture of Figure 9, DDDAS criteria are fulfilled via multi-resolution image domain partitioning and intelligent scheduling, as follows. Firstly, the Master (control module) decomposes the problem into atoms (pulses rendered onto one tile of the output image, including all data and computational work). Secondly, the Resolution Controller determines spatial resolution for rendering each tile, as discussed in Section 5.2. Thirdly, the Master sends each atom to the Multilevel Scheduler that balances load for heterogeneous devices and maintains locality of access for efficiency.

The algorithms that power our intelligent Multilevel Scheduler are shown in Figure 10. The List Assignment (LA) algorithm orders then selects tiles according to total work (TW), while the Longest Processing Time (LPT) algorithm orders then selects and assigns tiles according to the workload specified for each tile. Figure 11 compares these algorithms with the naïve algorithm.

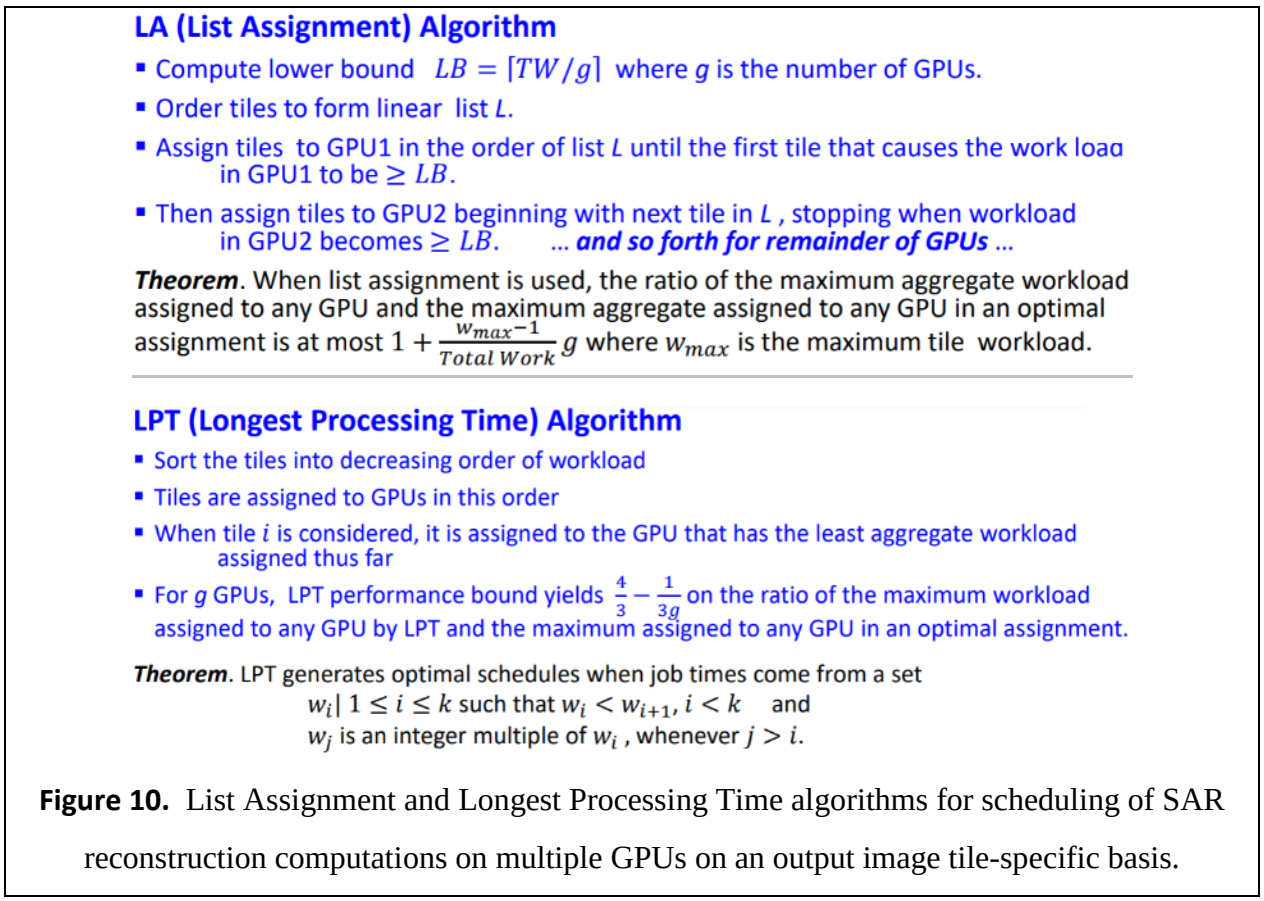


Figure 10. List Assignment and Longest Processing Time algorithms for scheduling of SAR reconstruction computations on multiple GPUs on an output image tile-specific basis.

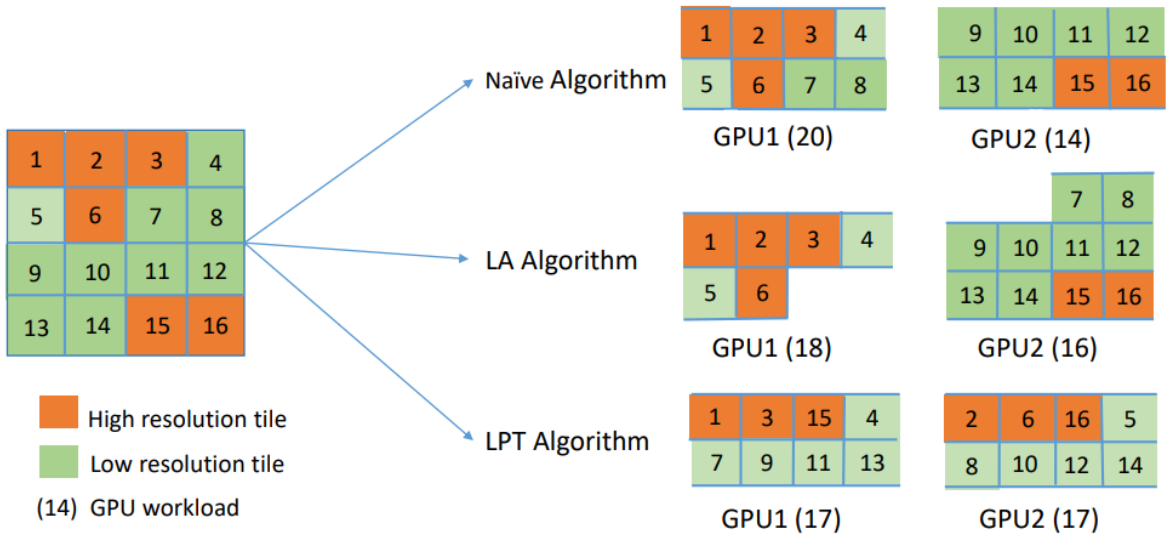


Figure 11. Pulse- and image-based partitioning strategies for multi-GPU architectures.

Scalability is an important property of parallel algorithms; hence, we tested the scalability and the ability of our algorithms to interleave computational (e.g., arithmetic) operations and I/O operations. Figure 12 graphs scalability in terms of execution time as a function of random versus sorted output tiles as well as type of algorithm and number of GPU nodes. Here, 60 percent of the tiles are high resolution, 20 percent are medium resolution, and 20 percent are low resolution for image size of 32,768 x 32,768 pixels for a SAR dataset of 5,000 pulses. The tiles appear randomly (Naïve algorithm) or in a sorted manner (LA or LPT algorithm). Results showed

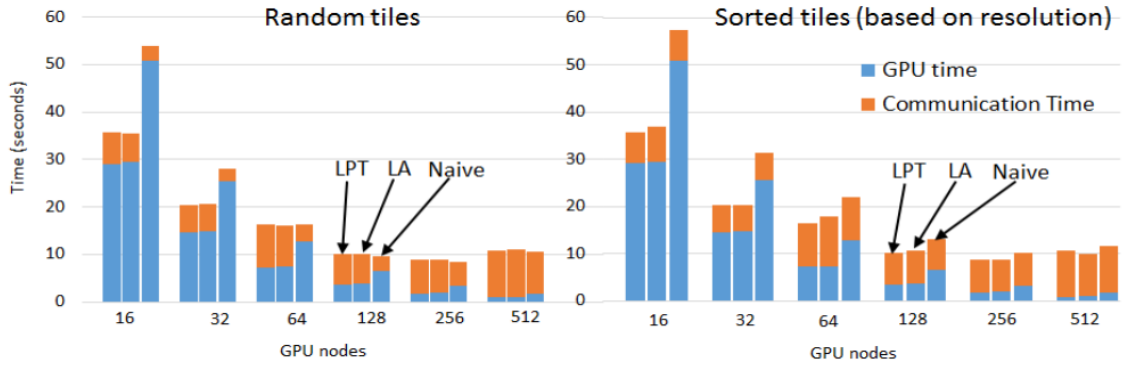


Figure 12. Scalability of our SAR reconstruction algorithms with multiple GPUs.

that LPT and LA algorithms have comparable performance, and perform better than the Naïve algorithm.

We found it important to increase performance by increasing CPU and GPU utilization. Figure 13 graphs test results whereby communication is overlapped with GPU Computation using CUDA streams. When GPU computation time is sufficient, both MPI broadcasting and CPU-to-GPU communication times are covered by GPU computation time. Here, 60 percent of the tiles are high resolution, 20 percent are medium resolution, and 20 percent are low resolution for image size of 32,768 x 32,768 pixels for a SAR dataset of 5,000 pulses. The tiles appear randomly (Naïve algorithm) or in a sorted manner (LA or LPT algorithm). Similar in concept to Figure 12, results showed that LPT and LA algorithms have comparable performance, and perform better than the Naïve algorithm.

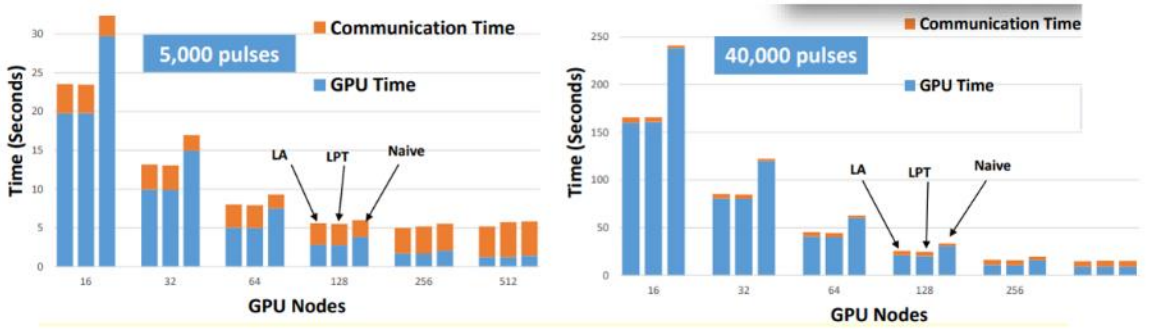


Figure 13. Communication optimization for multiple GPUs.

Performance optimization strategies were designed with the help of a theoretical model illustrated in Figure 14. A total of 321 test cases were run to determine the error in this theoretical model, as shown in Figure 15. Here, absolute modeling errors of 1.71 percent or less were measured for predicted CPU time based on maximum CPU frequency f_{max}, with minimum error of zero percent, mean error of We also found the expected result to hold, namely, that CPU computation time is inversely proportional to CPU clock frequency. Thus, our model is accurate as well as physically faithful.

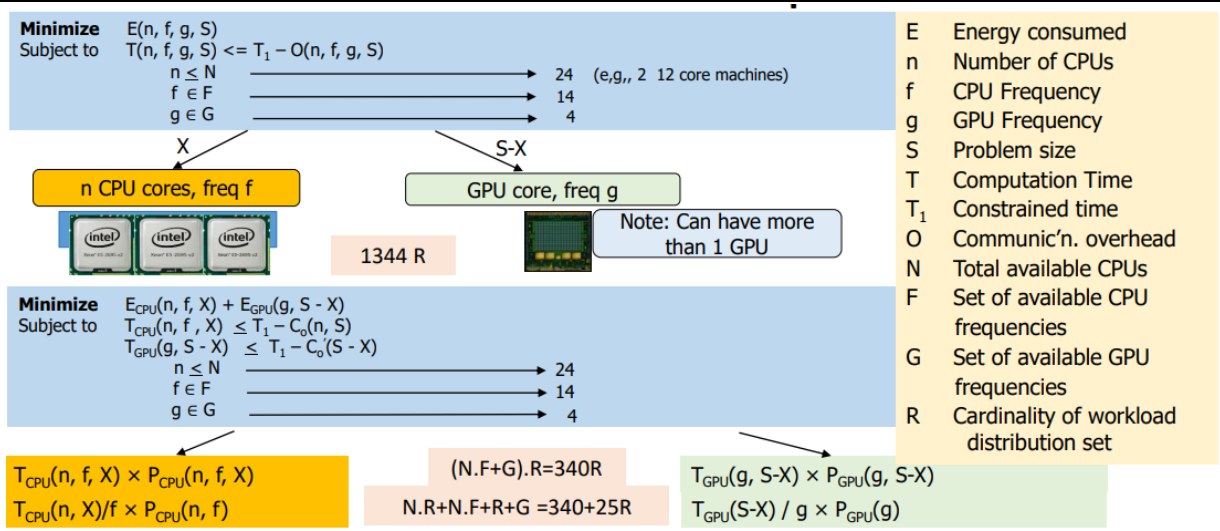


Figure 14. Theoretical framework for performance optimization for multiple GPUs.

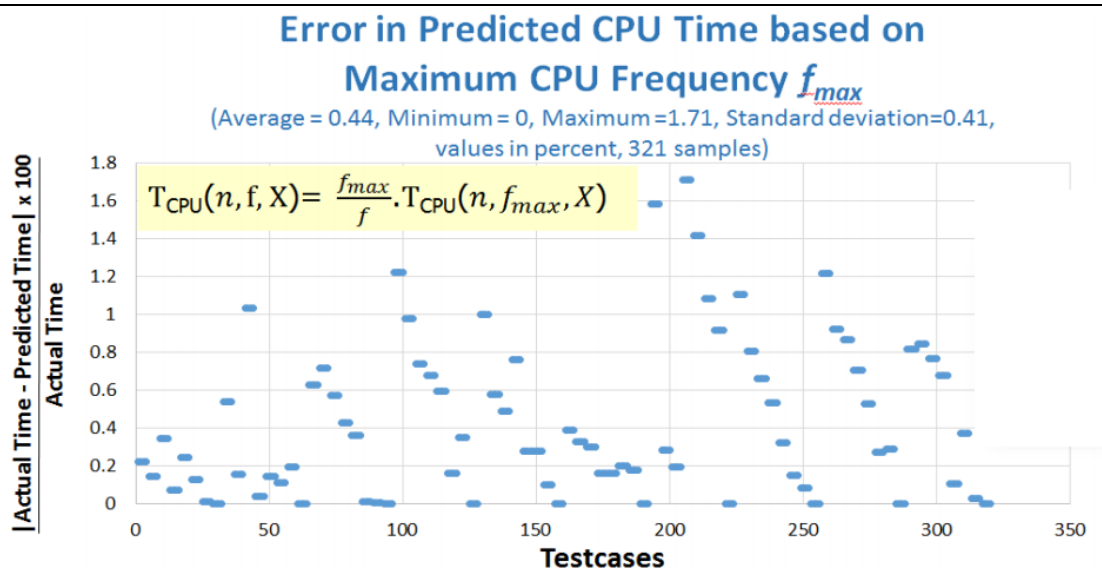


Figure 15. Model prediction error for performance optimization, where N = 321 cases.

Similar measurements were made for prediction accuracy with respect to energy and power consumption. Figure 16 graphs CPU and GPU power consumption as a function of clock frequency f and problem size, for 5,000 test cases. As expected, we found that CPU and GPU power consumption does not depend much on problem size.

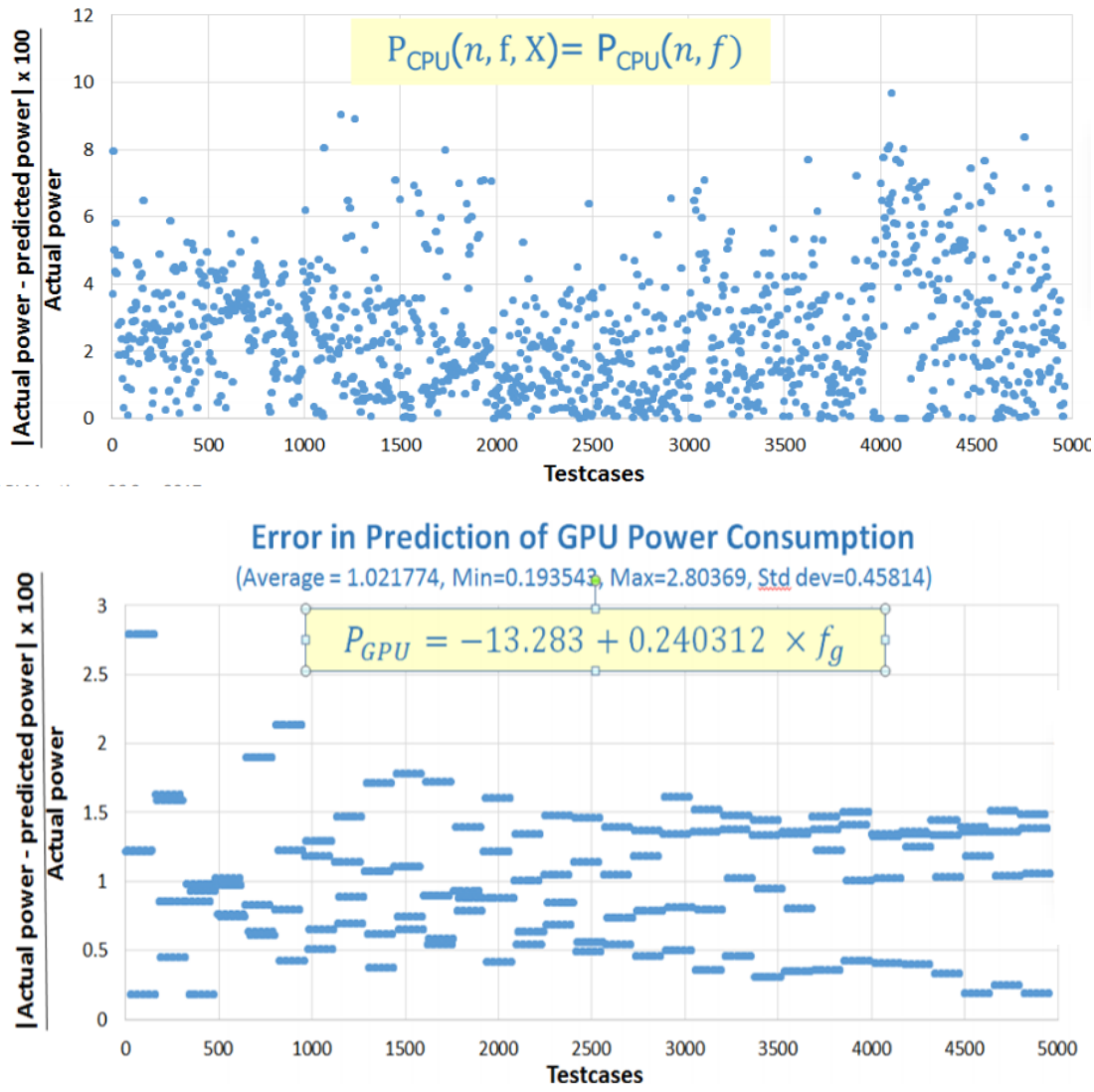


Figure 16. Model prediction error (in percent) for CPU and GPU power consumption.

Thousands of test cases were simulated, and we marshalled the resulting performance measurements to form well-populated graphs from which Pareto-optimal curves were derived. A Pareto curve, in this context, delineates the optimization horizon, beyond which additional optimization efforts yield no further increases in performance. Figure 17 illustrates the process of multiple simulations (top graph) from which Pareto curves (bottom graph) are derived for modeling energy versus runtime. Similar results were obtained for power consumption.

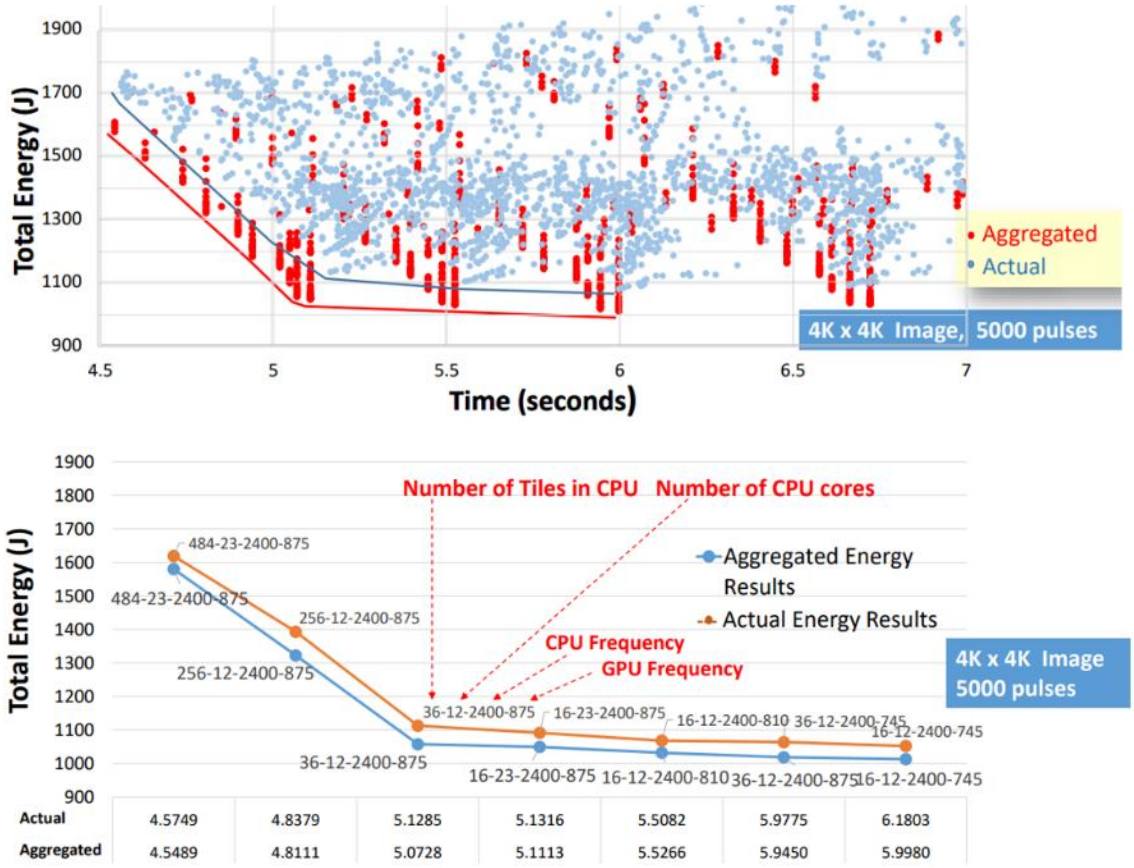


Figure 17. Modeling energy consumption versus runtime, where *Aggregated* denotes model prediction, and *Actual* denotes measured performance.

From these simulation results, we developed a method for describing the Pareto curves in terms of piecewise-continuous linear equations. An example is shown in Figure 18a. These Pareto mod-

Number of CPU Cores

$$P_{CPU}(1, f) = 46.58979 + 0.008488 \times f$$
$$P_{CPU}(2, f) = 43.32728 + 0.009557 \times f$$
$$P_{CPU}(4, f) = 39.42679 + 0.011428 \times f$$
$$P_{CPU}(12, f) = 33.68219 + 0.022718 \times f$$
$$P_{CPU}(23, f) = 74.73546 + 0.043811 \times f$$

Figure 18. (a) Predictive model for CPU power consumption, where *f* denotes frequency.

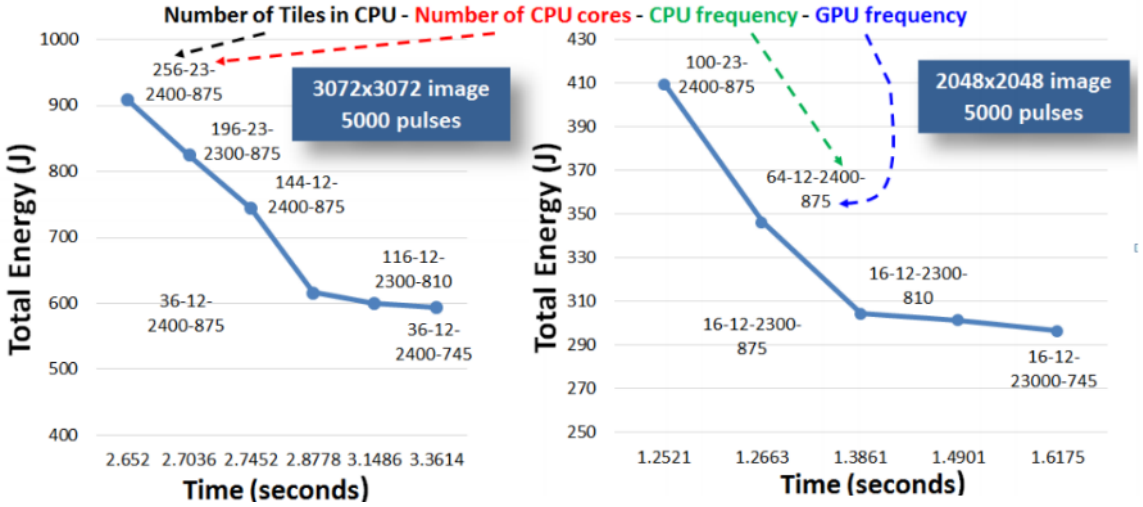


Figure 18. (b) Aggregated models for multi-node CPU-GPU energy consumption.

eling methods were applied to energy and power consumption for CPUs, GPUs, and nodes with GPUs controlled by a CPU. Our aggregated models as a function of image size, for a pulse data set of size 5,000 pulses, are presented in Figure 18b.

Following our research in CPU-GPU node optimization with tests on the ORNL Titan architecture, we moved to implementation and optimization of SAR image reconstruction on the Intel Knight's Landing hybrid multiprocessor (HMP) and clusters thereof. Our technique and results are described in the following subsection.

5.4 Implementation on Intel Knights Landing Processors and their Clusters

Thus far, we have employed multiresolution tile selection and portioning techniques to improve the performance of SAR image reconstruction. Multiresolution images play a significant role in real-time change detection, where lower resolution may be sufficient for background regions and higher resolution can be used for more sensitive regions. In this paper, we present an evaluation of performance, power and energy consumption for SAR image reconstruction on the Intel Knight's Landing (KNL) hybrid multi-processor.

5.4.1 KNL Processor Architecture

The Intel Knights Landing (KNL) is a many-core processor targeted toward high performance

computing applications. It exhibits significant improvements in scalar and vector performance over its predecessor Intel Knights Corner architecture: theoretical peak performance of 3 TFlops for double precision and 6 TFlops for single precision can be achieved using KNL [Soda15]. A KNL processor consists of 36 tiles interconnected by a 2D mesh. Each tile holds 2 cores and a 1MB L2 cache that is shared between two cores in the tile. Each core is based on Intel's Silvermont microarchiteccture and contains 2 Vector Processing Units (VPU). Each core is capable of handling 4 SMT threads and is equipped with 32KB instruction and data caches with 64 bytes cache line length [Cod17], which can hold 16 single-precision and 8 double- precision floating-point (FP) operands. A high-level architectural diagram of KNL is shown in Figure 19. KNL introduced a new 512-bit vector instruction set architecture (AVX 512), that supports 512 bit floating point and integer vector operations while still supporting all legacy instructions (e.g., SSE, AVX, AVX2). The AVX unit in KNL is comprised of AVX-512F, AVX-512CD, AVX-512PF, and AVX-512ER. AVX-512F contains 32 registers and 8 mask registers with gather and scatter support and supports 512- bit floating point and integer vectors. AVX-512PF introduces new prefetch instructions with gather and scatter. AVX-512ER is used for exponential and reciprocal instructions. AVX512CD introduces efficient conflict detection and improves vectorization.

One of the major improvements of KNL is the introduction of the Multi Channel DRAM (MCDRAM), which is a 16GB on-chip high bandwidth memory. MCDRAM is a 3D stacked DRAM accessed by 8 high speed memory controllers [Cod17] which provides a bandwidth of over 400 GB per second [Soda15]. In addition, KNL provides 384 GB of DDR4 memory, which can be accessed by two 3-channel memory controllers, and has a bandwidth of 90 GB per second. The KNL memory can be configured in three different modes, which can be set by BIOS options at boot time. MCDRAM can be configured in one of three modes: as shown in Figure 4, available memory mode options are *Flat* mode, *Cache* mode, and *Hybrid* mode which we describe as follows.

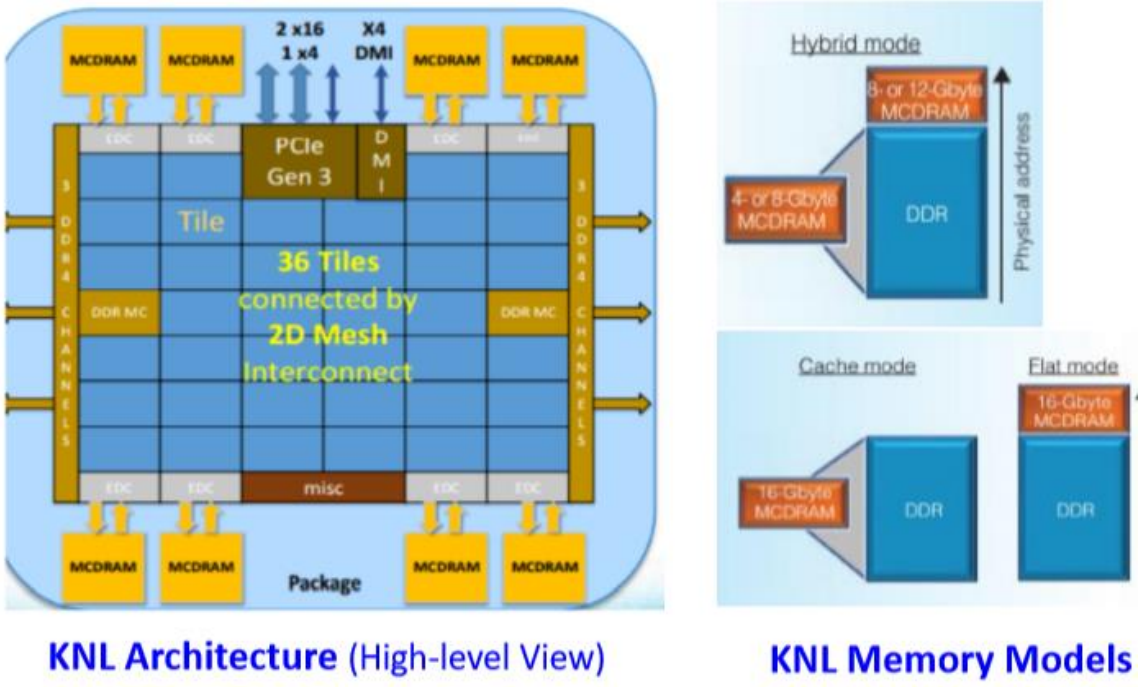


Figure 19. Overview of KNL architecture and memory modes.

1) *Flat Mode*: Flat mode allows MCDRAM to work as a part of regular memory and have a physical address space. This increases the total memory size of the system, such that a software developer can decide which memory to use for a given application. If the total memory requirement of the application is less than the size of the MCDRAM (16GB), then flat mode is a preferred option as data can reside completely on the MCDRAM. If the total memory requirement of the application is larger than the size of the MCDRAM, careful consideration must be given to memory allocation. To increase performance, a software developer must explicitly allocate critical data on the MCDRAM by using fast *malloc* functions from the *memkind* library [Can15].

2) *Cache Mode*: Here, the OS considers the MCDRAM as an L3 cache that can store data from DDR4. If the application requires a larger memory size but accesses smaller size data for a continuous period, then cache mode will increase performance as memory requests will hit the L3 cache. Also, if the memory requirement is larger than the size of the MCDRAM and critical parts of the data cannot be readily recognized, then it is recommended that the MCDRAM be employed in cache mode. Another advantage of the cache mode is that the OS handles

MCDRAM as the L3 cache, so the software developer does not need to consider handling the MCDRAM. On the other hand, if the application causes many cache misses, cache mode will not provide good performance as each cache miss is needed to issue a request to DDR memory after communicating with the die [Cod17], which increasess the latency.

3) *Hybrid Mode*: Hybrid mode is a combination of both Cache and Flat modes, which splits the MCDRAM into two parts, where one part can be used in cache mode and the other part can be used in flat mode. Note that 4 or 8 GB of MCDRAM can be used as a cache and 8 or 12 GB of MCDRAM can be used in flat mode [Soda15]. When the application requires caching in general and also some critical data requires frequent access, hybrid mode tends to give the best performance.

5.4.2 KNL Clustering Modes

In the KNL architecture, a *tile* is comprised of two cores and an L2 cache; tiles are interconnected by a 2D mesh. Modified, Exclusive, Shared, Invalid and Forward (MESIF) protocol is used to maintain cache coherency [Kan07]. Due to the 2D mesh architecture, latency can increase significantly for an L2 cache miss. KNL supports five modes of tile clustering to provide different levels of address affinity. This helps the user to reduce latency by reducing the distance which the memory request travels on the mesh, thus improving overall performance. These clustering modes can be selected from the BIOS at boot time, requiring a reboot when a change in the mode is needed. The clustering modes are briefly described below.

1) *SNC4*: In SNC4 mode, the KNL chip is divided into four nonuniform memory access (NUMA) domains. SNC4 mode provides affinity between the tile directory and memory. As most of the traffic will be inside the local domain, this mode provides the best latency only if data is not shared among the domains. There is an additional overhead if cache data has to cross NUMA boundaries.

2) *SNC2*: Similar to SNC4 mode, but the chip is divided into two NUMA domains.

3) *All-to-All*: This is the most general mode of all clustering modes. In All-to-All mode, there are no specific requirements in memory allocation. But it can deliver poor performance when L2 cache misses occur. In general, upon an L2 miss, a request is sent to the distributed directory to check if the data is locally held by another core. If there is a miss in the directory, then the data

request is forwarded to the memory and the memory sends data to the requester. When the clustering mode is All-to-All, addresses are uniformly hashed across all distributed directories. This increases mesh traffic when the requester tile and the corresponding tag directory tile are far apart.

4) *Quadrant*: In quadrant mode, the chip is divided into four parts, which are spatially local to the memory controllers. This mode provides affinity between the tile directory and memory. Quadrant mode also provides lower latency and higher bandwidth than all-to-all mode.

5) *Hemisphere*: In Hemisphere mode, the chip is divided into two parts as opposed to four parts in Quadrant mode.

5.4.3 SAR Reconstruction Algorithm Optimizations for KNL Architecture

Different versions of the SAR image reconstruction algorithm were analyzed [Wija17]. Given output image sizes N_x and N_y, tile sizes T_x and T_y, as well as number of pulses N_p, the following is the ordering of *for* loops that gives the best performance:

```
for i = 1 to N_x/T_x
  for j = 1 to N_y/T_y
    for k = 1 to T_x
      for l = 1 to T_y
        for p = 1 to N_p
          Calculate pixel location corresponds to i, j, k and l
          Calculate pixel values as stated in Algorithm 1
```

In practice, during processing, the output image domain is partitioned into tiles, then the two outermost *for* loops iterate through the image tiles first in the *x* dimension and then in the *y* dimension. The third and fourth *for* loops iterate through the *x* and *y* dimensions of a tile, and the last *for* loop iterates through the pulses.

Loop fusion and permutation were employed as follows. The two outermost loops were fused, followed by a fusion of the next two loops, resulting in an algorithm with three *for* loops. Finally, the innermost *for* loop was swapped with the second *for* loop, yielding an algorithm that gives the best performance on KNL.

Compilation was based on the Intel C++ compiler with fast math calculations and **`-O3`** optimization. Improved Advanced Vector Extensions (AVX) in KNL are able to compute up to 16 32-bit floating points concurrently, which represents a two-fold improvement from legacy AVX that was able to compute 8 32-bit registers concurrently [Lom11,Tan12]. The aforementioned Intel C++ Compiler performs auto-vectorization on the SAR reconstruction CPU code with the use of the **`xMIC-AVX512`** flag.

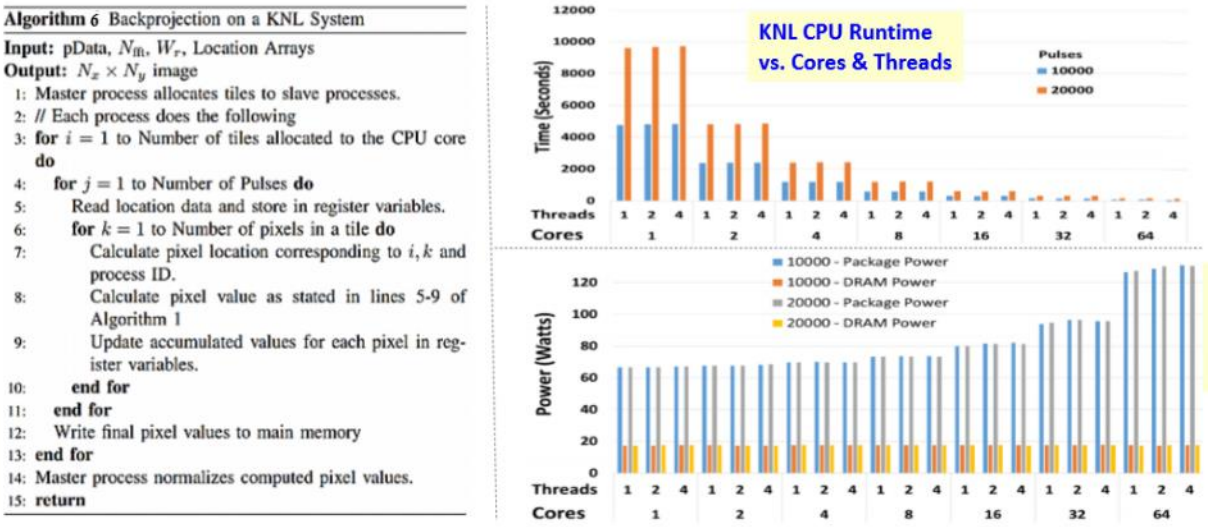
Each iteration of our optimized code requires six memory read accesses and one memory write access, which affects the performance of the SAR reconstruction. KNL has an on-chip high-bandwidth memory with throughput of more than 400GB per second [Soda15], which is five times more than the speed of a DDR4 memory. We investigated different memory modes available in KNL to find the best configuration to improve the performance of SAR image reconstruction by taking maximum advantage of the available MCDRAM.

Our KNL system has 64 cores that can be used for computing; each core supports up to four threads using hyperthreading. Work for each core is allocated by the master process, while the slave processes compute the image tiles, which are then saved in the main memory. At the end of the computation, the master process normalizes the computed pixel values to create the final output image. In order to maximize the computational parallelism, an efficient mechanism to distribute the work among CPU cores is required. Results in [Wija17] indicate that the image partitioning approach yields better results than the pulse partitioning approach for our KNL SAR Algorithm. Hence, we used only image partitioning on our KNL based system.

5.4.4 KNL Test Results

We conducted performance tests of the backprojection algorithm shown in Figure 20, which also displays graphs of runtime and power consumption versus number of KNL cores and threads. We used three different resolution levels for our multi-resolution experiments. Each reconstructed SAR image is partitioned into equal-size tiles where each tile has a specific resolution. For the resolution of 1 (i.e., highest resolution), every pixel in the tile is being computed from pulse data. At one-fourth resolution, a 2×2 grid is overlaid over the tile and only the top left pixel is computed. At one-sixteenth resolution, the overlay is done using a 4×4 grid

and the top left pixel in each grid cell is computed. We also determined optimal tile allocation for KNL cores when multiple resolution levels are considered. Our Intel KNL processor is ver-



Algorithm 6 Backprojection on a KNL System

Input: pData, N_{fft}, W_r, Location Arrays
Output: $N_x \times N_y$ image

1: Master process allocates tiles to slave processes.
2: // Each process does the following
3: **for** $i = 1$ to Number of tiles allocated to the CPU core **do**
4: **for** $j = 1$ to Number of Pulses **do**
5: Read location data and store in register variables.
6: **for** $k = 1$ to Number of pixels in a tile **do**
7: Calculate pixel location corresponding to i, k and process ID.
8: Calculate pixel value as stated in lines 5-9 of Algorithm 1
9: Update accumulated values for each pixel in register variables.
10: **end for**
11: **end for**
12: Write final pixel values to main memory
13: **end for**
14: Master process normalizes computed pixel values.
15: **return**

Figure 20. KNL backprojection algorithm and test results for runtime (top graph) and power consumption (bottom graph) versus number of cores and threads.

sion 7210, which has 64 cores where each core supports four hyperthreads. CPU base clock frequency is 1.30 GHz and L2 cache size is 32 MB, with 16GB on-chip high-bandwidth MCDRAM

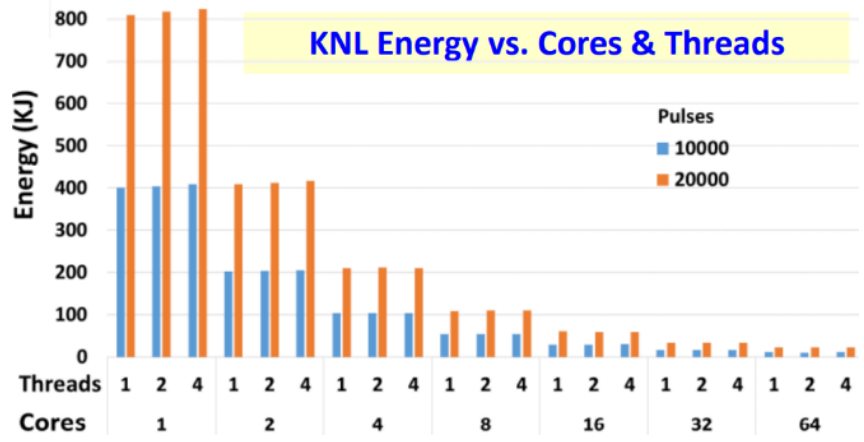


Figure 21. KNL backprojection algorithm test results for energy consumption as a function of cores and threads.

and 384GB of DDR4 memory. The Thermal Design Power (TDP) is 215 Watts. To measure power consumption, we used the **`perf stat`** command in Linux.

Figures 20 and 21 present the runtime, power, and energy consumption for KNL-based SAR reconstruction using tile partitioning as a function of number of cores, number of threads per core, and pulse sizes. SAR reconstruction was performed on KNL with parameters selected as follows. The tile size was set to 16×16 pixels and the reconstructed image size was set to 8192×8192 pixels. We used 1, 2, 4, 8, 16, 32 and 64 CPU cores; 1, 2 and 4 threads per core; as well as 10,000 and 20,000 pulses. Figure 20 shows CPU processing time for our SAR reconstruction algorithm for different number of cores when uniform resolution is used. Our experiment indicates that runtime is linearly proportional to the number of pulses N_p which is expected, as workload is $\mathbf{O}(N^2 \cdot N_p)$ – thus illustrating the strong scalability of our approach.

Table 2. KNL backprojection algorithm and test results for runtime as a function of resolution.

Image size	Pulses	High	Medium		Low	
		time(s)	time(s)	speedup	time(s)	speedup
8192×8192	10000	77.29	19.73	3.92	5.41	14.30
	20000	155.79	39.73	3.92	10.81	14.41
	40000	313.52	80.13	3.91	21.84	14.36
$16K \times 16K$	10000	307.92	78.68	3.91	21.45	14.36
	20000	622.03	158.37	3.93	43.14	14.42
	40000	1252.05	319.02	3.92	86.96	14.40

Key elements of CPU power consumption can be categorized as memory and processor power. Our experiments show that DRAM power consumption has a constant value of 18 Watts. Figure 20 shows that the package power varies with the number of active cores; starting with a baseline power consumption of 66W, package power increases linearly with the number of cores, but does not vary significantly with the number of threads per core. The power numbers represent total power, including both the static and dynamic power components. Figure 21 shows that the energy consumed decreases in most cases even though the power required increases with the number of cores. This result is not anomalous, but occurs because of the corresponding decrease in runtime and the inverse relationship between energy, power, and time.

In order to demonstrate the performance improvement using medium and low resolution levels as opposed to high resolution, we conducted experiments with full resolution, 1/4th of full resolution and 1/16th of full resolution for image sizes 8192×8192 pixels and 16384×16384 pixels, as well as pulse counts of 10,000, 20,000 and 40,000 pulses. The results are presented in Table 2, where it can be readily seen that if resolution is reduced by a factor f_r, then processing is speeded up by a factor of f_r because of the reduced number of computations.

Figure 22 shows the performance of a combination of different KNL memory and cluster modes. Flat memory mode with quadrant clustering mode gives us the best performance, while hybrid memory mode with 50 percent cache configuration and SNC4 clustering mode performed the worst. For our application, the difference between the best and worst modes was 7.5 seconds over a range of 309 sec to 316.5 sec, or approximately 2.4 percent. This is because our application is compute bound and has a L2 cache hit ratio of 99 percent.

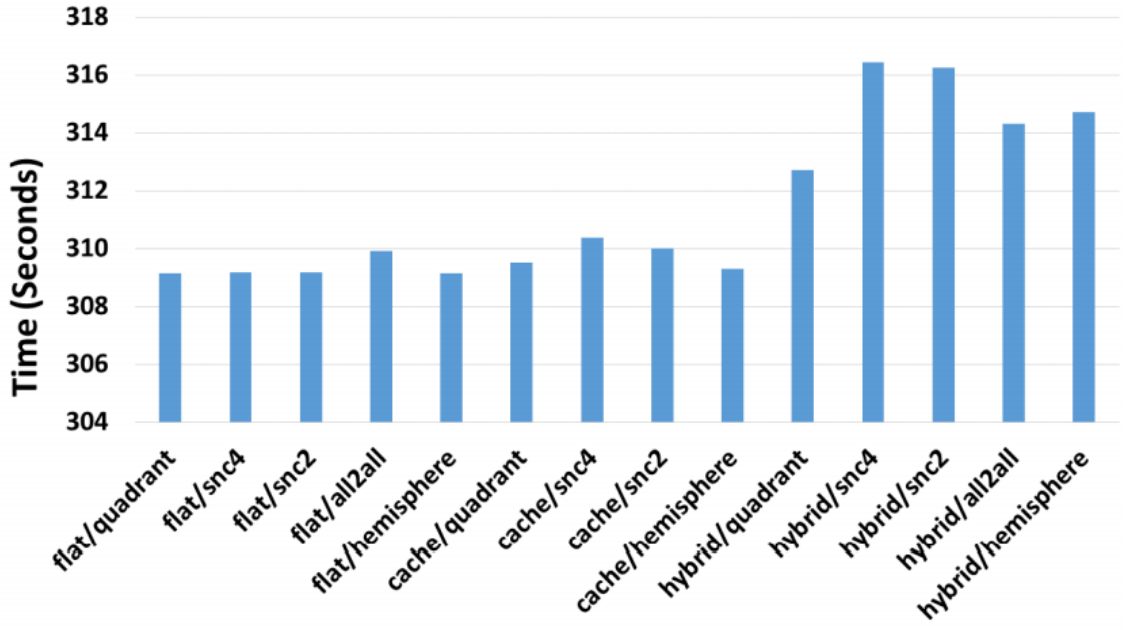


Figure 22. KNL backprojection algorithm performance results for different KNL modes.

We compared our KNL results against Intel Xeon (Ivy Bridge) and NVIDIA Tesla K40M GPU. Figure 23 shows the performance of the described architectures. The Tesla K40m is

comprised of 2,496 CUDA cores and 13 multiprocessors, where cores are distributed evenly among the multiprocessors. It has a maximum boosted clock speed of 875MHz and a default clock speed of 745MHz. The GPU card has up to 48KB of shared memory per block and 64K registers per block. We used Nvidia System Management Interface (*nvidia-smi*) command line utility for power and energy measurements. The Intel Xeon CPU E5-2695 V2 (Ivy Bridge) is a dual socket system where each socket has 12 cores. It has 2.4GHz base frequency with a 115W thermal design point (TDP). The Intel Xeon CPU E5-2695 V2 has a memory of 768 GB and a cache of 30,720 KB with 64-byte alignment.

The Running Average Power Limit (RAPL) based PowerGadget was used to measure power, and we employed 8192×8192-pixel output imagery with 10,000, 20,000 and 40,000 pulses, with full resolution, 1/4th, and 1/16th of full resolution. Figure 23 (top graph) shows the run time of KNL, Tesla K40m and Intel Xeon architectures, where KNL yields better results than Ivy Bridge but has a higher runtime than Tesla K40m. Average power consumption for KNL, Ivy Bridge and Tesla K40m is 145 Watts, 180 Watts and 140 Watts, respectively. Figure 23 (bottom graph) shows energy consumption of the three architectures, where Tesla K40m has the lowest, KNL has the second lowest, and Ivy Bridge has the highest energy consumption.

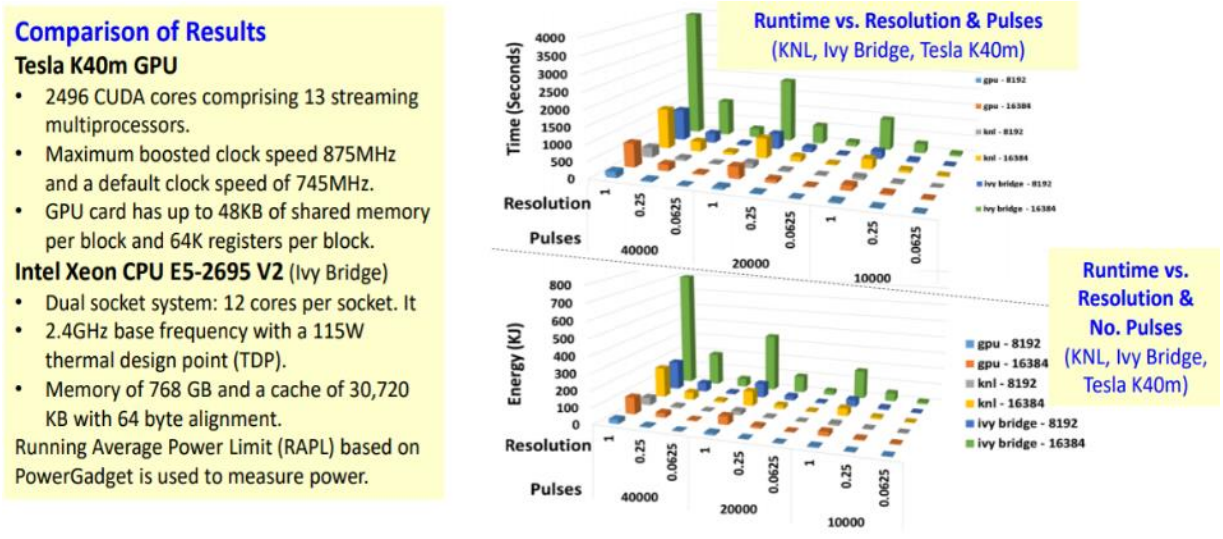


Figure 23. KNL backprojection algorithm and test results for runtime (top graph) and energy consumption (bottom graph) versus resolution and number of pulses.

5.4.5 Summary of Development and Test Results

We conducted research and development of SAR image reconstruction with backprojection algorithms, for networked multi-GPU and multi-node architectures, where each node contained a multicore CPU and one or more GPUs. Similarly, we developed SAR reconstruction algorithms for Intel's Knight's Landing (KNL) hybrid multi-processor (HMP). In each case, we were able to employ our unified software framework to implement, test, and demonstrate SAR reconstruction algorithms that proved (theoretically and in practice) to be linearly scalable in terms of number of cores and number of networked nodes. Power and energy consumption scaled likewise, and reconstruction of SAR images as large as 16K×16K pixels was demonstrated. Due to premature termination of our funding, we were not able to conduct detailed error analysis on the reconstruction algorithms or their computational optimizations.

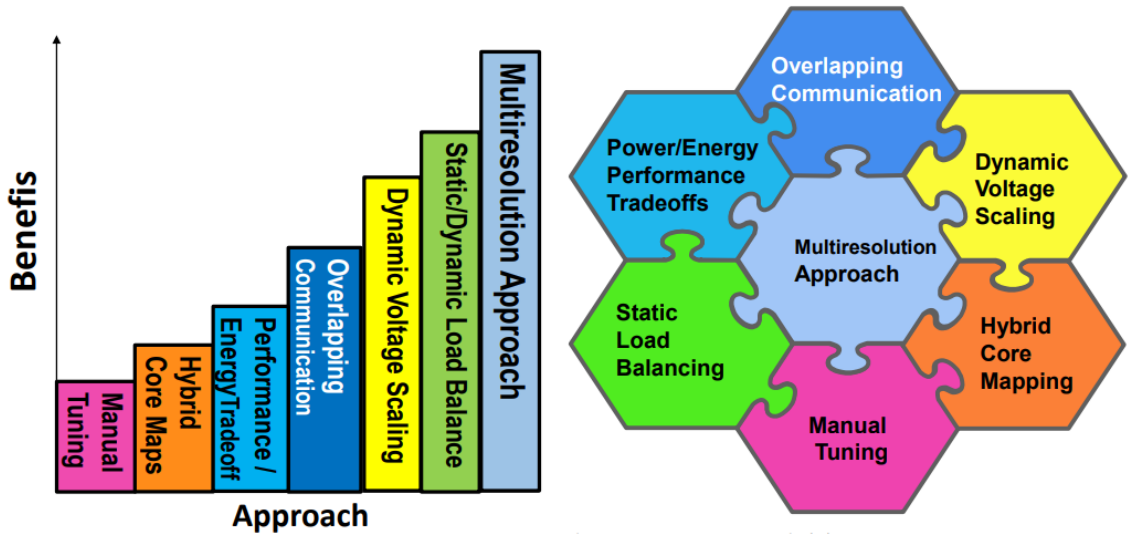


Figure 24. Conceptual summary of our findings re: computational optimization.

We developed a multiresolution approach to SAR image reconstruction that was linearly scalable in terms of spatial resolution. That is, an f_r -fold reduction in local spatial resolution yielded an f_r-fold computational savings (space, time, energy, and power consumption), as theory predicted. Our multiresolution approach was found to be superior to other methods of SAR reconstruction algorithm optimization. Together with tiling of the output image and

physics-based strategies for optimal pulse subset selection, our technique demonstrates the feasibility of large-scale parallel processing of tensor-product-based algorithms on networks of heterogeneous multicore processors.

6 Results and Discussion: Simulation and Change Detection

6.1 Simulation of SAR Scenes

The analysis of change detection is facilitated in this project via simulation of synthetic SAR imagery using POVray™ rendering software [POVray] and the SAR image reconstruction algorithms discussed in Section 5. The purpose of image synthesis is not to test all physical or pathological scenarios or image instances, but to provide data to our change detection algorithm development process. For realism and to test performance of change detection algorithms in the presence of noise and data errors, we simulated effects such as receiver noise, uncertainty in quantizing the range bins, and transmission errors such as packet corruption or dropouts.

Our video SAR (VSAR) sequences used for testing the change detection algorithms were mostly produced by the following simulation process:

Step 1: Construct scene in POVray data format, whereby streets, trees, buildings, structural features such as windows and doors, and vehicles such as automobiles and trucks were obtained from various online POVray-compatible libraries [POVlib], then inserted into the scene to be simulated.

Step 2: Establish SAR sensor entrance pupil location, and compute distance to each scene point, by which we mean that given the SAR receiver entrance pupil $\mathbf{p}_e \in \mathbb{R}^3$ and a point $\mathbf{p}_o \in \mathbb{R}^3$ in the constructed 3D scene, the Euclidean distance d from entrance pupil to scene point is determined as $d = \| \mathbf{p}_o - \mathbf{p}_e \|$, where ($\| x \|$) denotes the norm of x. We then collect the various distances d into a set S_d that characterizes the projection of the scene to the SAR sensor. By appropriately perturbing the distances d in S_d, one can simulate sensor noise and quantization errors. Addition of an intensity component I_d allows the expansion of S_d to be a collection of tuples of form (d, I_d), which allows various reflectance effects to be simulated at the scene level, and SAR receiver effects such as noise and look-angle dependent response to be accounted for.

Step 3: Compute simulated SAR pulse dataset D from S_d and apply SAR reconstruction algorithm to create a reconstructed SAR image frame $\mathbf{a}_i$. With many such frames taken at different sensor positions , one can construct a video sequence $\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_i, \ldots, \mathbf{a}_N\}$.

Examples of simulation results are presented in the following figures.

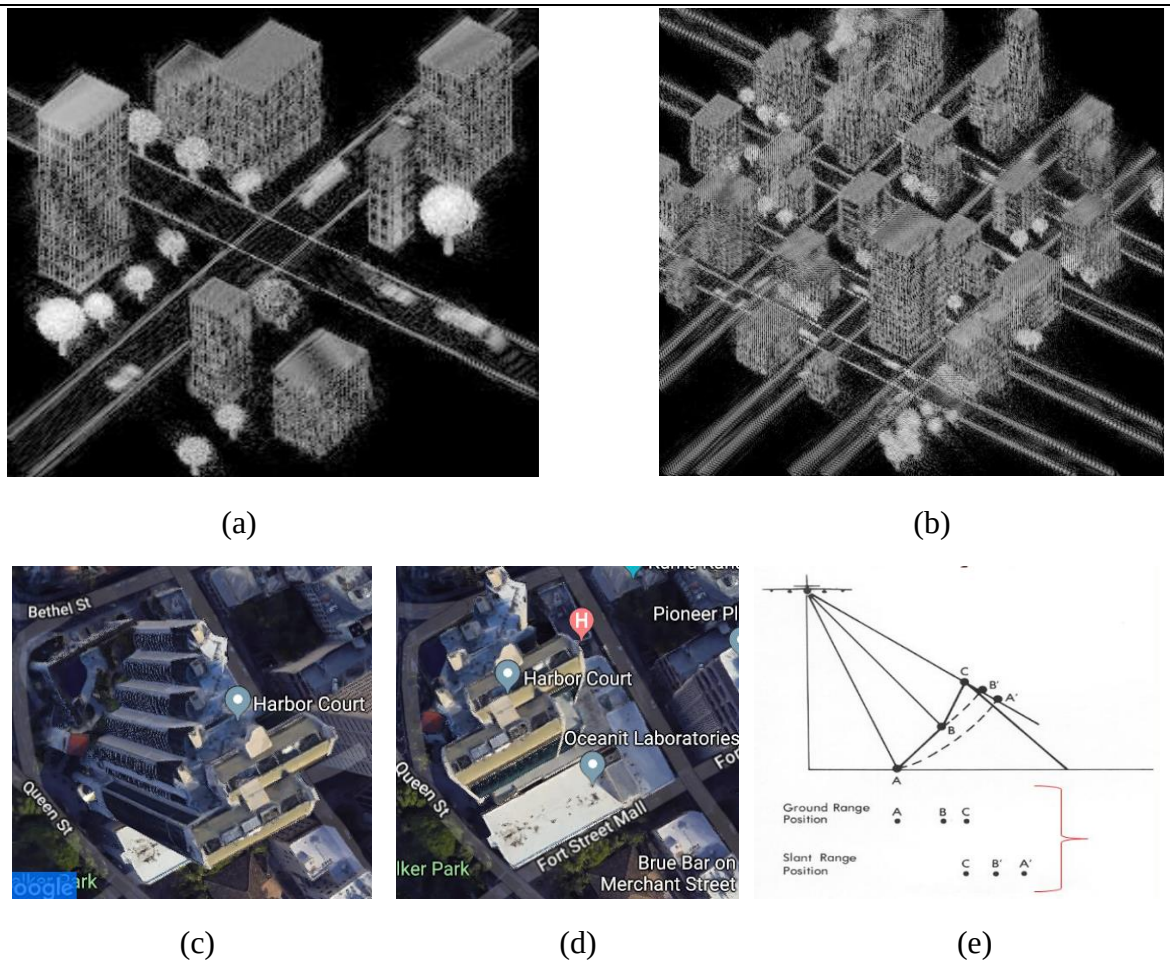


Figure 25. Example of (a) simple and (b) more complex simulated SAR street scenes. (c,d) Example of projective distortion in aerial scenes with visible-wavelength sensor [Google Maps: Honolulu]; (e) SAR layover effect - positions of features A, B, and C are reversed on the radar image. Hence, feature C' is said to "lay over" features B' and A'.

Figure 25c-d illustrates a simple and a more complex street scene, with some noise and blur interjected into the scene. Note that the buildings do not appear to have "layover distortion", which is a reconstruction artifact that causes proximal scene features to be over-emphasized spatially versus more distant scene features. An example of layover distortion in optical imagery is given in a Google aerial view of two nearby towers in an office development near Honolulu harbor, as shown in Figures 25a and 25b. Corresponding SAR geometry is given in Figure 25e.

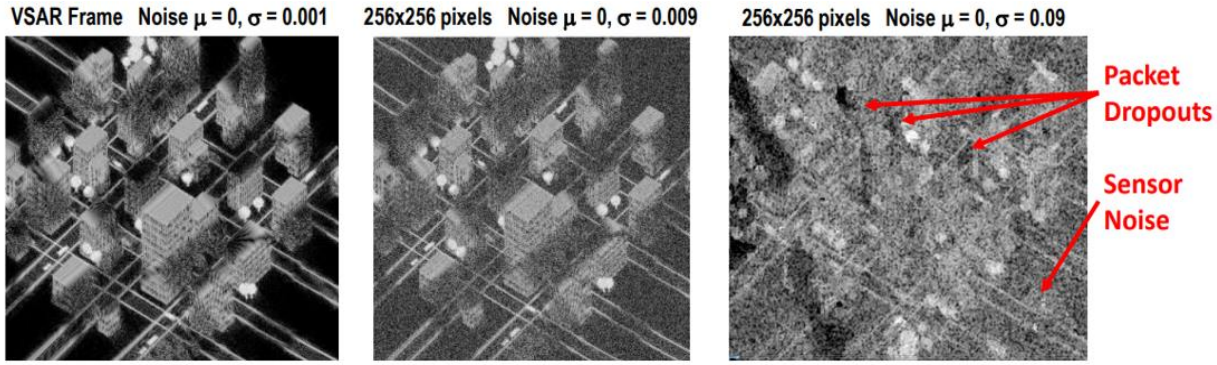


Figure 26. Example of simulated VSAR frames at various noise and channel error levels.

As described in [Wang13], it is possible to reduce SAR layover distortion via a series of inverse transformations. However, due to premature termination of our funding, we did not have the opportunity to simulate the effects of layover distortion or the distortion reduction methods described by Wang et al. Therefore, we concentrated on simulating sensor noise and channel errors, as exemplified in Figure 26.

The above imagery was then input to our change detection algorithm, whose development is described in the following subsection.

6.2 Change Detection and Hierarchically Partitioned SAR Reconstruction

Image-based change detection (CD) is the process whereby two or more frames of a video sequence are analyzed to determine the probability and location of areas where interframe differences are significantly related to mission-specific objectives. Our suite of change detection (CD) algorithms are designed to be applied to the output of our VSAR Reconstruction Algorithm and its postprocessed results (e.g., sets of aggregated superpulses) to implement object detection in video SAR. Image change detection involves the identification of regions of change in multi-temporal datasets. A typical algorithm with pre-processing has the high-level architecture shown in Figure 27. Our approach incorporates optical flow and a deep neural network to estimate and learn the motion attributes of regions or moving objects in images. Once motion estimation is performed, we then compute change detection.

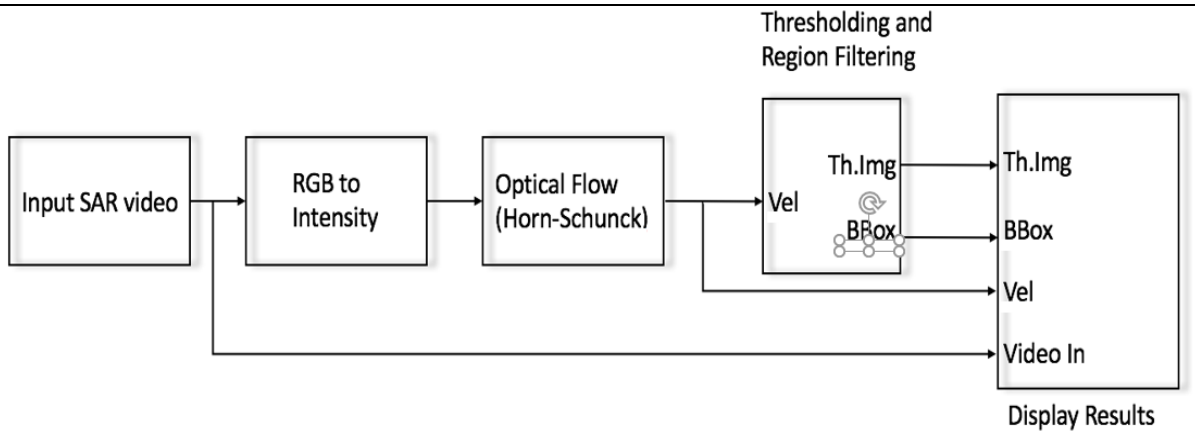


Figure 27. High-level view of change detection architecture employed in our CD algorithm.

We developed and tested enhanced CCD algorithms with parallelism analysis. Development of hierarchical processing strategies for energy-efficient porting of these CD algorithms to different types of multicore processors and clusters of these processors was undertaken but stopped prematurely due to decreases in proposed funding. Thus, our Hierarchical CCD algorithm was successfully applied to a limited database of simulated VSAR imagery obtained by reconstruction of VSAR pulse data (as described in Section 6.1), to provide support for more extensive testing and optimization of CCD algorithm performance (which did not occur due to the aforementioned funding cuts).

Challenges and Solutions: Due to data characteristic different from general image change detection, the challenges of change detection in SAR video/image sequences include: moving sensor, variations in pulse intensity, variations in target size and posture, and variable image resolution (itself a feature of our SAR reconstruction approach). To address the change detection problem for VSAR, our team developed two computer vision algorithms: (1) Optical flow and foreground detection, as well as (2) Supervoxel based foreground segmentation. We discuss these algorithms, as follows.

Optical Flow and Foreground Segmentation Algorithm: Optical flow quantifies the motion of objects in a video stream and has been broadly applied to numerous motion-based object detection and tracking systems. The algorithm uses an optical flow estimation technique (we use the Horn-Schunck method [Horn81]) to estimate motion vectors in each frame of a VSAR sequence. By thresholding the motion vectors, the model creates a binary feature image

containing blobs of moving objects. Median filtering is used to remove scattered noise; a morphological *close* operation is performed to remove small holes in blobs. An underlying model helps locate targets in each binary feature image using Blob Analysis. Then, our *Draw Shapes* module draws a color rectangle around the targets indicated by Blob Analysis. Figure 28 provides a high-level diagram of this detection algorithm.

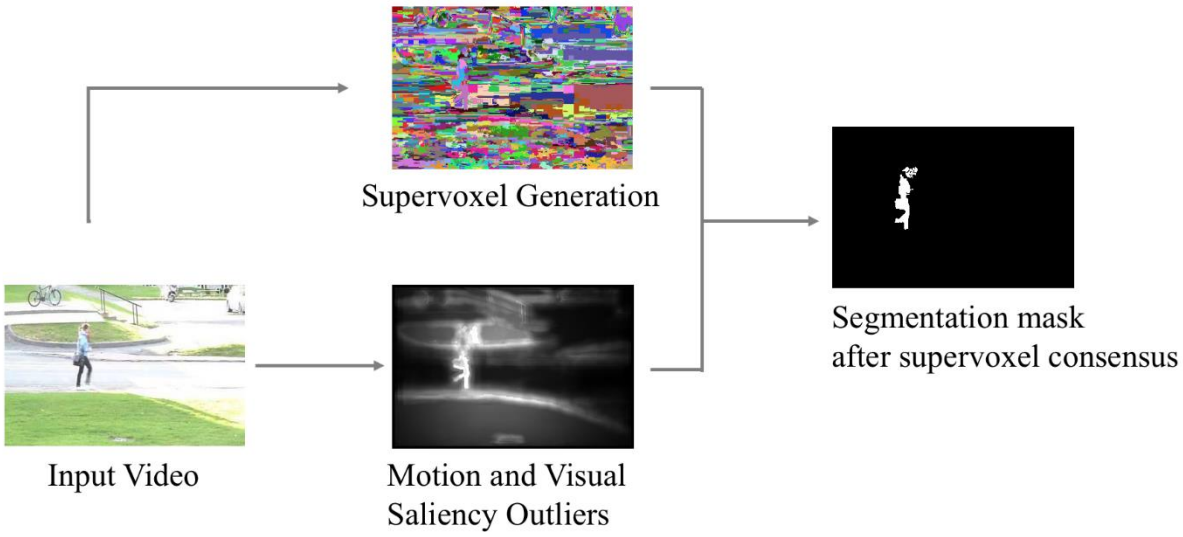


Figure 28. High-level view of supervoxel segmentation architecture in our CD algorithm.

Supervoxel-based Foreground Segmentation Algorithm: We began development of the following 3D-UCM supervoxel method to generate supervoxels from a video stream such as VSAR. However, this method was not fully developed due to the aforementioned funding decrease. Firstly, we build an initial quantitative estimate of *foregroundness* using a combination of motion (optical flow) and visual saliency cues that are weighted using a straightforward statistical measure of *outlierness*. Secondly, based on the initial estimate, we construct an *internal or local consensus* (in terms of supervoxels) that is relayed across the video among a set of nearest neighbors; this result forms an additional non-local consensus. Figures 27 and 28 diagram a pipeline representation of this algorithm using image data from the changedetection.net dataset.

Given this basic algorithm development approach, we then refined our change detection algorithm and applied it to progressively more complex datasets. Figure 30 shows the result of optical flow segmentation on low-resolution VSAR data (256x256-pixel frame size) simulated in-house (see Section 6.1) and higher-resolution simulated VSAR data (512x512-pixel frames). In these examples, no noise or channel errors were simulated, so as to establish a base set of performance results in the absence of input perturbations.

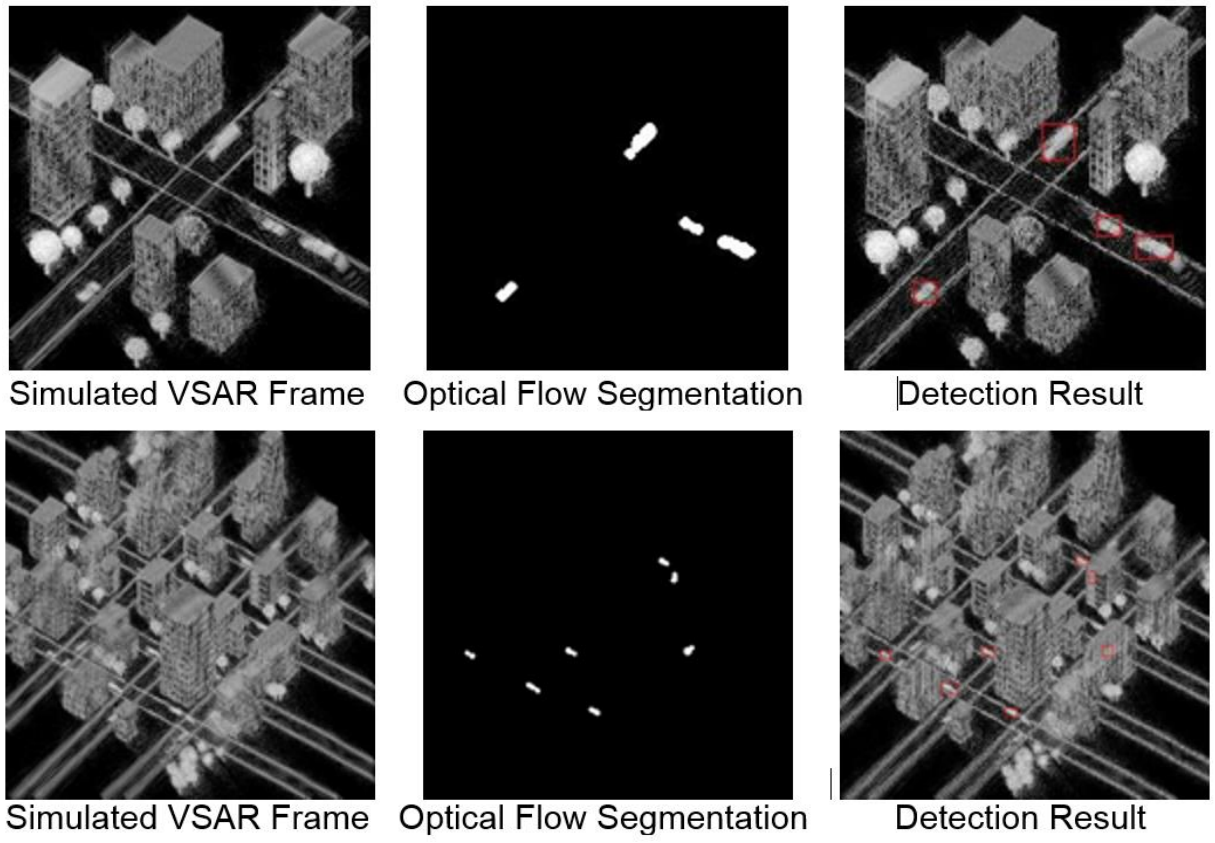


Figure 29. Example of optical flow based target segmentation in our current CD algorithm: (top row) 256x256-pixel video frame size; (bottom row) 512x512-pixel frame size.

Regions of change were next localized using block analysis – Figure 30 illustrates these results for vehicular detection (similar to Figure 29) in terms of (a) the initial analysis of 256x256-pixel imagery and (b) subsequent analysis of 512x512-pixel VSAR imagery. Note the two false positives in Figure 30b, which are located on the sides of buildings. We believe these artifacts

are due to simulated sensor motion effects, which we planned to investigate in Year-4 of this project, but could not due to the aforementioned decrease in planned funding.

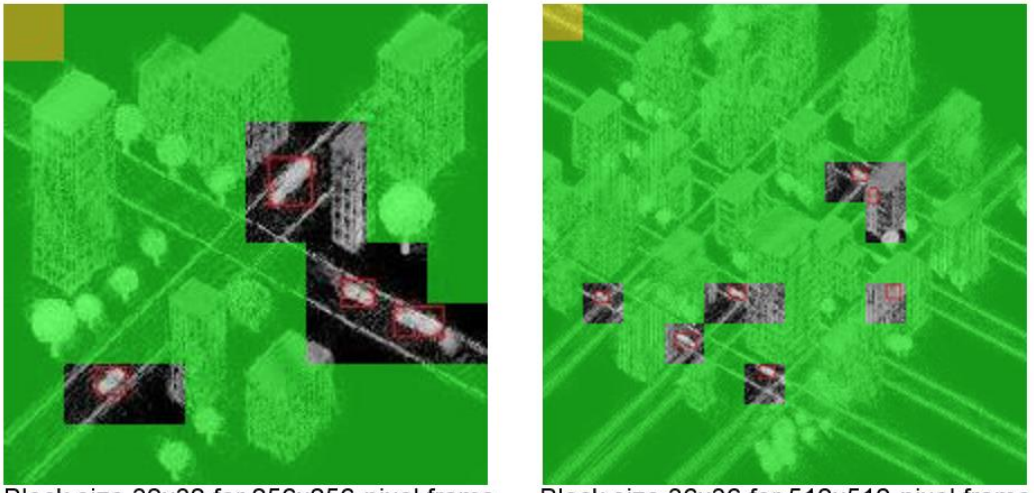


Figure 30. Example of target localization using block analysis in our current CD algorithm: (a) 256x256-pixel video frame size; (b) 512x512-pixel frame size.

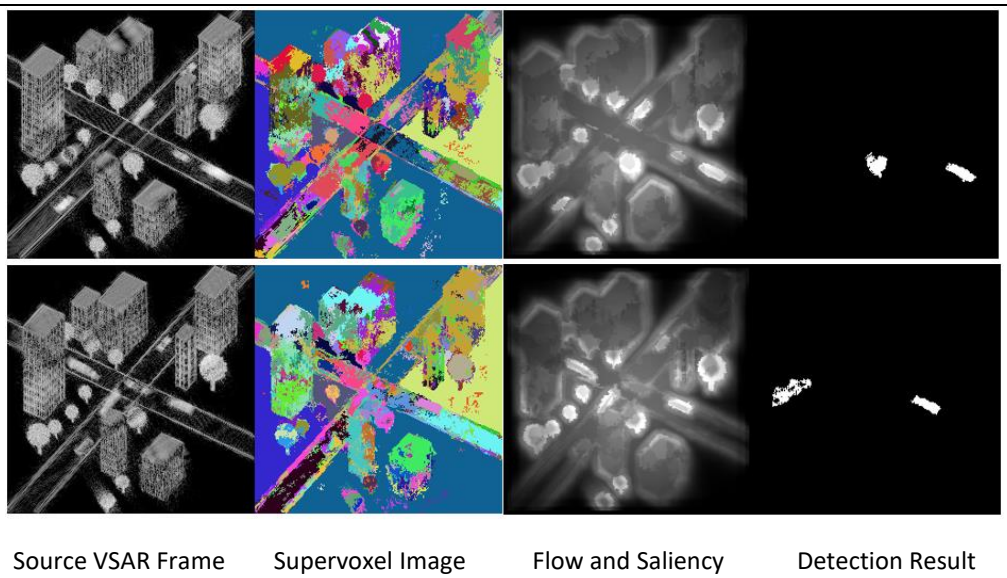


Figure 31. Example of target localization using supervoxel based analysis in our current CD algorithm, applied to two 256x256-pixel video frames taken from a simulated VSAR sequence.

The supervoxel computation and target detection step is next applied, to produce a map of target presence that can be focused on a specific target attribute. For example, Figure 31 illustrates the detection of a speeding car (right-hand detected object) in two different frames of a VSAR sequence with 256x256-pixel frame size.

Deep CNN-based Target Detection Approach: We began to integrate the preceding methods with a deep segmentation convolutional neural network (CNN) to learn changes in multi-temporal images, but could not complete this work item due to the aforementioned funding decrease. Our approach is motivated by the need for more frequent and efficient updates in remote sensing and aerial image change detection. It is an end-to-end approach which trains on registered image pairs in a segmentation CNN (illustrated schematically in Figure 32) for object-based change detection. We utilize a deep convolutional (DCNN) encoder/decoder architecture for object-based labelling and can segment images with small changes in objects or regions. This DCNN learns to predict object-based labels from supervised learning. Therefore, it requires a dataset of input images with corresponding ground truth labels. Our experiments employed the Change Detection Benchmark in Aerial Imagery Dataset (AICD Dataset [AICD]) for our experiments that supported development of our change detection algorithm.

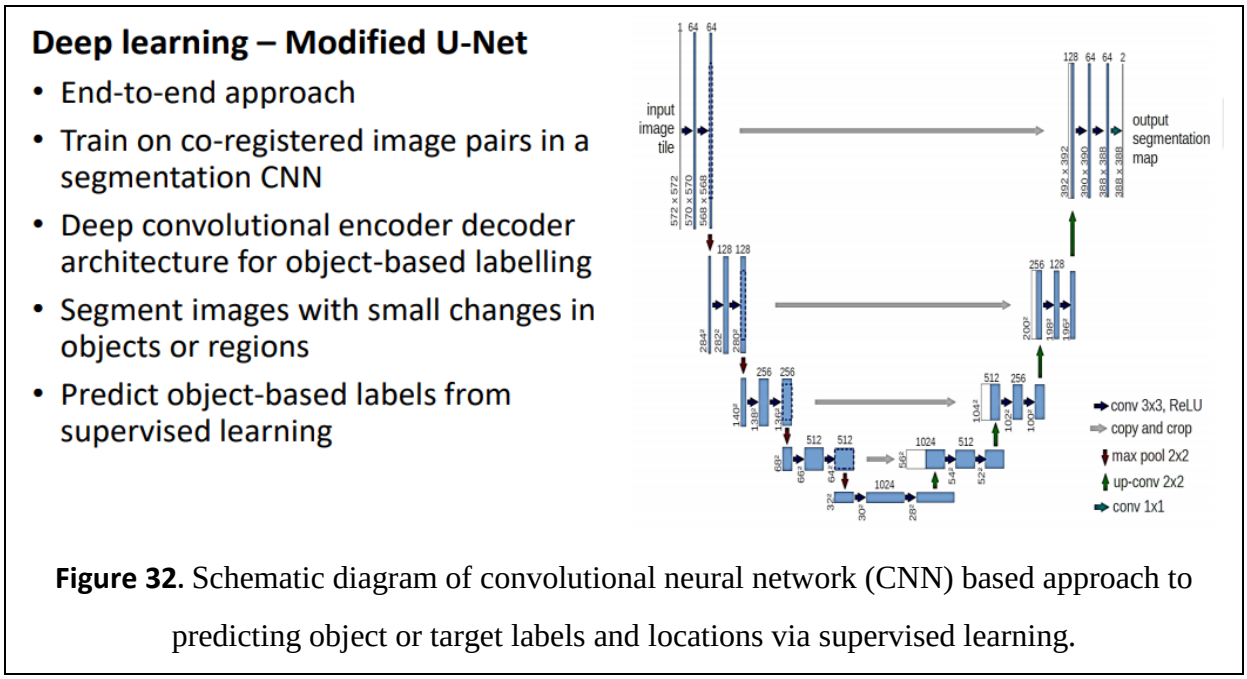


Figure 32. Schematic diagram of convolutional neural network (CNN) based approach to predicting object or target labels and locations via supervised learning.

DDDAS Related Features: Our enhanced change detection algorithm, which is currently being enhanced, has the following features of interest to DDDAS and to parallel implementation of change detection:

- *CD Algorithm Developed at UF Highlights Regions of Interest*, to avoid expending scarce computing resources (especially in embedded or other energy-constrained systems) on areas where targets are not likely to occur. Here, we can use context and history to determine target presence in future superpulse sets [Chap13, Sch11]. As an example of context, a vehicle is not likely to be found on the surface of a pond. Similarly, a dismount in a given (x,y) position will not likely be found 100 meters away from that position in the next VSAR frame (assuming, say, one frame per second rate) – because a human cannot move that fast. These simple examples are but two of a wide variety of scenarios that we have constructed from our extensive previous work in parallel computing of automated target recognition operators for a wide variety of surveillance applications.
- *Isolation of Regions Containing Moving Vegetation* (e.g., high frequency spatiotemporal variance) is important to rule out areas where there is low probability of target presence. In these areas, computation of the CD algorithm can proceed at lower spatial resolution, thereby reducing computation time and energy consumption. However, our aforementioned contextual model takes into account the possibility of (for example) dismounts in defilade in the contextual of a grassy background – which is especially emphasized if there is a history of target presence in such areas. Similarly, trees with moving foliage are detected, segmented, and portrayed at significantly lower resolution – especially (for example) if one is tracking vehicles that context tells us are not found in trees.
- *Construction of Multiresolution (Pyramidal) Scene Representation* to facilitate the rigorous and computationally efficient querying of data structures derived from the simulated or actual VSAR imagery. As is the case with our Hierarchical SAR Reconstruction algorithm, theory and experimental results lead us to expect an f-fold reduction in spatial resolution to produce an f-fold increase in computational efficiency.
- *Identification of Future Target Movement Regions* (by optical flow, variance, and context) using our aforementioned context models and predictive target motion models.

- *Statistical Identification and Tracking of Targets by Spatiotemporal Variance*, to support predictive selection of areas-of-interest for processing future frames in a VSAR sequence. As before, this predictive and adaptive technique is employed to circumvent wastage of costly computational resources, energy, and execution time.

DDDAS goals have been supported by developing the "Dynamic" and "Adaptive" parts of our change detection algorithm to conform to DDDAS focus, thereby allowing our Hierarchical CCD algorithm to adapt to time-varying mission-specific, environmental, sensor, and computational status/constraints.

6.3 Results of Applying Change Detection to Simulated SAR Imagery

We enhanced and tested our VSAR reconstruction algorithm, an enhanced version of our VSAR simulator, and our enhanced change detection algorithm, on a highly parallel multicore system.

Our Multiresolution VSAR reconstruction algorithm was tested on a variety of architectural configurations that were configured from the ORNL Titan architecture via a permutation-of-parameters methodology. Architectural parameters included problem size, number of CPUs and/or GPUs, CPU and/or GPU frequency, number of thread blocks and threads, and so forth. We showed (see Section 5) that our VSAR reconstruction algorithm scales robustly over problem sizes appropriate for SAR-based surveillance operations (e.g., from 1Kx1K pixels to 32Kx32K pixels) and from one to 512 GPUs. Execution time was approximately linear in the problem size (i.e., number of output image pixels), and scaled well under multiresolution constraints. Given that the average resolution of a single-resolution VSAR image frame is f times the resolution of a multi-resolution SAR image representation, then the resulting computational cost incurred by processing the multiresolution frame is nearly $1/f$ that of processing the single-resolution frame. The approximation is primarily due to message-passing overhead.

Energy and power consumption were modeled via a parametric simulation that estimated energy costs based on a Pareto-optimal analysis, and the model accuracy was checked via hundreds to thousands of runs on the ORNL Titan architecture. Results for energy and power consumption were illustrated in Section 5 and exemplified in Figure 33, where the "aggregated"

results are model-based estimations and the "actual" results are measured from empirical analysis of multiple algorithm runs.

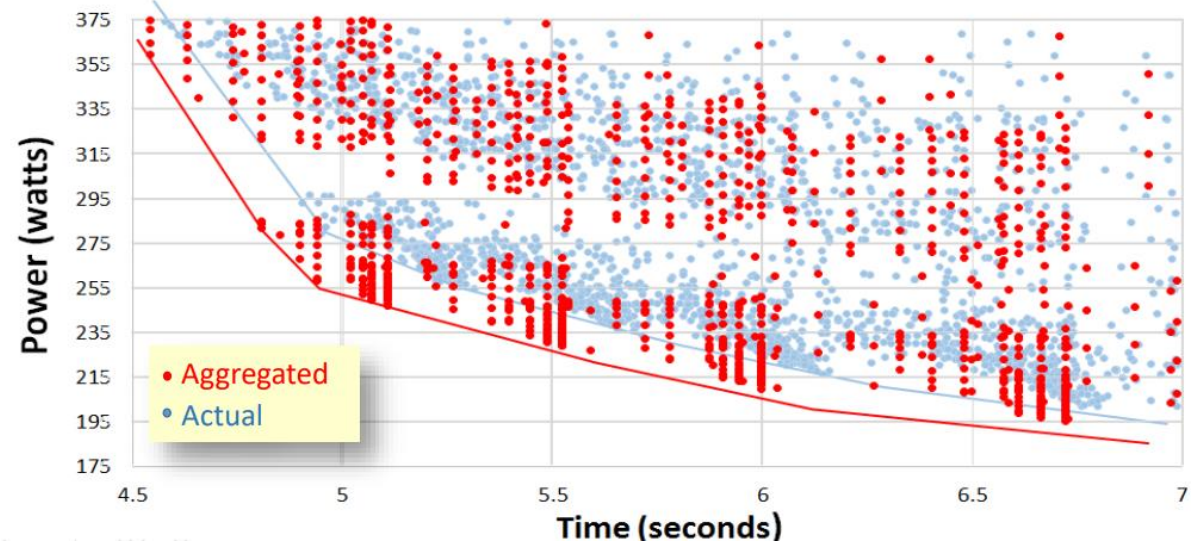


Figure 33. Power consumption as a function of runtime for 4Kx4K SAR output image frame reconstructed from 5,000 input pulses via our algorithm run on the ORNL Titan architecture.

As described in Section 4, for tests described in this report, we employed the Titan computational platform at Oak Ridge National Laboratory, which is a hybrid system consisting of AMD Opteron CPU and Nvidia Kepler GPU in the ratio of 1:1. Thus each of the 18688 nodes has a 16-core 2.2GHz AMD Opteron 6274 processor with 32GB of RAM, and an Nvidia Kepler K20 GPU with 6GB of DDR5 memory. There are 512 service and I/O nodes, and 200 cabinets. The system uses Cray Gemini 3D torus interconnection.

In our tests, continuous streaming SAR pulse data was read from disk at a single Master host and distributed to other hosts using the Host Coordinator discussed in Section 5. Reconstructed image frames of size 4096x4096 pixels were generated at each host, with a mix of variable-resolution tiles per image. Completed image frames were transmitted back to the Master host once per every 5,000 pulses. Throughput was measured as the number of pulses rendered per second, and was also presented as Gflops/sec, assuming 43 operations per pulse, per output (reconstructed) pixel [Chap13,Wija18-19].

Throughout this study, we continued to ask the research question: *Are there other work scheduling systems that could outperform our DDDAS-influenced system?* As before, the only system we tested that came close to the results from our optimizing scheduler is the *Charm++* paradigm. However, Charm has the disadvantages of a 100ms setup time, and is lacking the automatic data partitioning [Dur12] that we have implemented in our system [Chap13,Wija16-17]. Additionally, our system is optimized for the specific computational and data movement requirements of VSAR data processing. For very large operands (e.g., larger than 8Kx8K pixel reconstructed image frames), Charm may indeed provide some additional efficiency. However, the data movement latencies of large operands are not well determined in any published reports of SAR image reconstruction of which we are aware. Thus, the use of software systems such as Charm on very large (BigData) operands, with the underlying requirement of tensor-product based computation upon which VSAR reconstruction is based, remains an open question.

Finally, we compared the performance of our multiresolution-based optimizing scheduler approach with a variety of approaches employed in tensor-product based algorithm optimization. As shown qualitatively in Figure 24, multiresolution based optimization has superior performance vis-à-vis manual tuning and hybrid core maps, traditional methods such as performance/energy tradeoffs as well as overlapping of computational and communication operations. We also found that multiresolution based optimization also outperforms hardware-based methods such as dynamic voltage scaling and software-based methods such as static versus dynamic load balancing.

7 Conclusions

The primary objectives of this project were achieved, namely, (1) development and testing of parallel algorithms for SAR image and video reconstruction on networks of heterogeneous multicore processors, (2) simple simulation of SAR imagery for testing our algorithms, and (3) development and testing (with simulated SAR imagery) of change detection algorithms for finding (for example) vehicular targets in urban imagery.

The following objective was also achieved, due to our research team's in-depth knowledge and investigation of parallel high-performance computing for SAR image reconstruction, namely, (4) significantly improved performance of our SAR reconstruction algorithms, due to the use of multiresolution data structures. In this technique, an *f*-fold reduction in data size via processing at reduced spatial resolution yielded (as theory predicted) an *f*-fold reduction in processing time for the reduced-size data partition. Image regions of greater interest (e.g., candidate target areas) could thus be processed at higher resolution, while regions of lesser interest (e.g., background) could be processed at much lower resolution and thus save considerable computation time, energy and power consumption.

We also achieved our objective of (5) extending our previous work in parallel scheduling to develop an optimizing scheduler for SAR image reconstruction. This was further extended and combined with our previous scheduler technology, to include the following classes of parallel operations: *pointwise arithmetic* and *unary operations*; *global operations* on an image or region, e.g., image sum, maximum, etc.; *inner product* or *dot product*; *convolution* with linear or nonlinear kernels; *matrix multiplication*; and *outer product* or *tensor product* (the primary focus of this work item in the current project).

Unfortunately, our research funding was cut by the sponsor, which did not enable us to pursue the following goals that we currently have allocated to possible future work:

- *Improve CD Algorithm Performance* by increasing Detection Probability and decreasing False Alarm Rate. We also propose to investigate the SIAP performance metrics [SIAP] and receiver operating characteristic (ROC) analysis for directing optimization of CD algorithm performance.
- *Enhancement of Multiresolution Structures to Focus CD Parallel Processing for Efficiency*

- More accurately determine probable target regions at high spatiotemporal resolution;
- Increase false positive rejection via temporal spectral filtering of low resolution regions;
- Decrease energy consumption by focusing energy-consumptive computational resources on fewer areas of interest based on contextual cues, thus allocating more energy resources to processing of other image areas.

- *Measure and Improve* CD Algorithms' energy profiles and power consumption on desktop networks and subsets of the ORNL Titan architecture, via modeling and simulation, laboratory measurements, and more effective scheduling around energy-consumptive computing resources.
- *Simulation/Test/Analysis of CD Algorithms* for Distributed Networked Embedded Applications, especially those controlled from mobile stations (e.g., handheld digital devices) in the field.

In the latter area (simulation and test), we propose to extend and enhance our existing realistic use cases for testing, analyzing and optimizing networked sensor computing scenarios over various ranges of simulated noise and channel error, to support embedded sensing with computation at multiple levels (e.g., at the sensor, with area-of-interest downselection for efficient embedded computing, as well as fast change detection at the sensor and at the ground station). All of these future work items would be performed in the context of energy-aware computing, with performance analysis directed by space, time, error, energy- and power-consumption (STEEP) metrics.

References

[Ahm12] I. Ahmad and S. Ranka, *Handbook of Energy-Aware and Green Computing*, Chapman-Hall, 2012.

[AICD] Aerial Imagery Dataset (AICD) Change Detection Benchmark, Web: http://www.computervisiononline.com/dataset/1105138664 , 2019.

[Ayg09] Ayguade, E., R. M. Badia, D. Cabrera, A. Duran, M. Gonzalez, F. Igual, D. Jimenez, J. Labarta, X. Martorell, R. Mayo, J. M. Perez, and E. S. Quintana-Orti, "A Proposal to Extend the OpenMP Tasking Model for Heterogeneous Architectures". *IWOMP: Evolving OpenMP in an Age of Extreme Parallelism* , vol. 5568, pp. 154–167, Dresden Germany, June 2009.

[Can15] C. Cantalupo, V. Venkatesan, J. Hammond, K. Czurlyo, and S. D. Hammond, "memkind: An extensible heap memory manager for heterogeneous memory platforms and mixed memory policies." Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2015.

[Cha91] Chang, P.P., S.A. Mahlke, and W.W. Hwu, "Using profile information to assist classic code optimizations", *Software – Practice and Experience*, vol. 21, pp. 1301-1321, 1991.

[Chap11a] Chapman, W., S. Ranka, S. Sahni, and M. Schmalz. (2011) "Parallel processing techniques for the processing of synthetic aperture radar data on GPUs", in *Proceedings of the IPDPS 2011 Conference*.

[Chap11b] Chapman, W., S. Ranka, S. Sahni, M. Schmalz, and U. Majumder. (2011) "Two-stage backprojection on synthetic aperture radar data using multiple GPUs", in *Proceeidngs SPIE*, vol. 8051.

[Chap12] Chapman, W., S. Ranka, S. Sahni, M. Schmalz, L. Moore, U. Majumder, and B. Elton. (2012) "Backprojection on Multicore and GPU Architectures", *Multi- and Many-Core Technologies: Architectures, Programming, Algorithms, and Applications*, Chapman-Hall / CRC Pr., Ed. S. Rajasekaran.

[Chap13] Chapman, W.H. (2013) *Multiresolution SAR Image Formation and Change Detection on High-Performance Heterogeneous Architectures*, Ph.D. Dissertation, University of Florida College of Engineering.

[Chap14] W. Chapman, S. Ranka, S. Sahni, M. Schmalz, L. Moore, and B. Elton, "A framework for rendering high resolution synthetic aperture radar images on heterogeneous architectures," in 2014 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2014, pp. 1–6.

[Cod17] V. Codreanu, J. Rodrguez, and O. Saastad, "Best practice guide-knights landing", 2017.

[Cruz14] J. R. Cruz, "Comparison of image processing techniques using random noise radar," DTIC Document, Tech. Rep., 2014.

[Cum05] I. G. Cumming and F. H. Wong, Digital processing of synthetic aperture radar data: algorithms and implementation. Artech house, 2005.

[Des92] M. D. Desai and W. K. Jenkins, "Convolution backprojection image reconstruction for spotlight mode synthetic aperture radar," IEEE Transactions on Image Processing, vol. 1, no. 4, pp. 505–517, 1992.

[Due05] M. I. Duersch, "Backprojection for synthetic aperture radar," 2013.

[Dur12] Duran, A., E. Ayguadé, R.M. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas. (2012) "OmpSs: A proposal for programming heterogeneous multi-core architectures", *Parallel Processing Letters* **21**:173.

[Gor10] L. A. Gorham and L. J. Moore, "Sar image formation toolbox for matlab," in SPIE Defense, Security, and Sensing. International Society for Optics and Photonics, 2010, pp. 769 906–769 906.

[Horn81] Horn, B.K.P., and B.G. Schunck. (1981). "Determining optical flow[J]", *Artificial Intelligence* **17**(1-3):185-203.

[Jak12] C. V. Jakowatz, D. E. Wahl, P. H. Eichel, D. C. Ghiglia, and P. A. Thompson, Spotlight-Mode Synthetic Aperture Radar: A Signal Processing Approach: A Signal Processing Approach. Springer Science & Business Media, 2012.

[Kan07] D. Kanter, "The common system interface: Intels future interconnect," Real World Technologies, pp. 1–4, 2007.

[Li09] Li, Y., S. Ranka, S. Sahni, and M. Schmalz. (2009) "Network centered muliple resource scheduling in e-science applications", ***GridNets***, 2009, LNICST **25**: 37-44, Springer Verlag.

[Li12] Li, J., S. Ranka and S. Sahni. (2012) "GPU Matrix Multiplication," *Multi- and Many-Core Technologies: Architectures, Programming, Algorithms, and Applications*, Chapman Hall/CRC Press. Ed. S. Rajasekaran.

[Lom11] C. Lomont, "Introduction to intel advanced vector extensions," Intel White Paper, 2011.

[Mor13] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, "A tutorial on synthetic aperture radar," IEEE Geoscience and remote sensing magazine, vol. 1, no. 1, pp. 6–43, 2013.

[Mun12] Munir, A., S. Ranka, and A. Gordon-Ross. (2012) "High performance energy-efficient multicore embedded computing", *IEEE Transactions on Parallel and Distributed Systems* **23**(4): 684–700.

[Plan09] J. Planas, R.M. Badia, E. Ayguadé, and J. Labarta. (2009) Hierarchical task-based programming with StarSs, *International Journal of High Performance Computing Applications*, Volume 23, No. 3, pp. 284–299.

[POVlib] F.A. Lohmueller, "POVray objects – ready-made objects for POVray", Web: http://www.f-lohmueller.de/pov_tut/objects/obj_950e.htm, 2019.

[POVray] POVray Rendering Software, Web: www.povray.org, 2019.

[Rig05] B. D. Rigling and R. L. Moses, "Taylor expansion of the differential range for monostatic sar," IEEE Transactions on Aerospace and Electronic Systems, vol. 41, no. 1, pp. 60–64, 2005.

[Rit91] Ritter, G.X. (1991) "Heterogeneous matrix products," in *Image Algebra and Morphological Image Processing II*, vol. 1568 of *Proceedings of SPIE*, (San Diego, CA), pp. 92-100.

[Rit01] Ritter, G.X. and J.N. Wilson. (2001) *Handbook of Computer Vision Algorithms in Image Algebra*, Second Edition, Boca Raton, FL: CRC Press.

[SIAP] Votruba, P., R. Nisley, and R. Rothrock. (2001) "Single Integrated Air Picture (SIAP) Metrics Implementation", Tech. Rpt. No. 2001-003, Joint Staff, Arlington VA.

[Sch97a] Schmalz, M.S. "Automated analysis and prediction of accuracy and performance in ATR algorithms. 1. Requirements, theory, and software implementation", *Proceedings SPIE* 3079:249-260 (1997).

[Sch97b] Schmalz, M.S. "Automated analysis and prediction of accuracy and performance in ATR algorithms. 2. Experimental results and analysis of system performance", *Proceedings SPIE* **3079**:261-272 (1997).

[Sch02] Schmalz, M.S. "Analysis of error in digital image processing algorithms using coincidence measures", *Proceedings of the SCI-2002 World Multiconference on Systemics, Cybernetics, and Informatics* **13**:457-462 (2002).

[Sch04] Schmalz, M.S. "Genetic algorithms for single- and multi-frame blind deconvolution", in *Proceedings of the AMOS 2004 Conference, Maui HI* (2004).

[Sch10] Schmalz M.S., G.X. Ritter, and E. Hayden. (2010) "Image algebra Matlab 2.3 and its application to research in image processing and compression", in *Proc. SPIE* **7799**.

[Sch11] Schmalz, M.S., G.X. Ritter, E. Hayden, and G. Key. "Algorithms for adaptive nonlinear pattern recognition", in *Proceedings SPIE* **8136** (2011).

[Sch14] Schmalz, M.S., W.H. Chapman, E.T. Hayden, S.Ranka, and S.K. Sahni. "Manyscale computing for sensor processing in support of space situational awareness", in *Proceedings of AMOS Conference*, Maui HI (2014).

[Soda15] A. Sodani, "Knights landing (knl): 2nd generation intel R xeon phi processor," in Hot Chips 27 Symposium (HCS), 2015 IEEE. IEEE, 2015, pp. 1–24.

[Sou99] M. Soumekh, Synthetic aperture radar signal processing. New York: Wiley, 1999, vol. 7.

[Tan12] A. Tanikawa, K. Yoshikawa, T. Okamoto, and K. Nitadori, "N-body simulation for self-gravitating collisional systems with a new simd instruction set extension to the x86 architecture, advanced vector extensions," New Astronomy, vol. 17, no. 2, pp. 82–92, 2012.

[Ula03] L. M. Ulander, H. Hellsten, and G. Stenstrom, "Synthetic-aperture radar processing using fast factorized back-projection," IEEE Transactions on Aerospace and electronic systems, vol. 39, no. 3, pp. 760–776, 2003.

[Wal95] G. C. Walter, S. G. Ron, and M. M. Ronald, Spotlight Synthetic Aperture Radar, Signal Processing Algorithms. Boston: Artech House, 1995, 1995.

[Wang13] P. Wang, Q. Ma, J. Wang, W. Hong, Y. Li, Z. ChenAn, "Improved SAR radiometric terrain correction method and its application in polarimetric SAR terrain effect reduction," Prog. Electromagn. Res. B, vol. 54, pp. 107-128, 2013.

[Wija16] A. Wijayasiri, T. Banerjee, S. Ranka, S. Sahni, and M. Schmalz, "Dynamic data driven image reconstruction using multiple gpus," in Signal Processing and Information Technology (ISSPIT), 2016 IEEE International Symposium on. IEEE, 2016, pp. 241–246.

[Wija17] A. Wijayasiri, T. Banerjee, S. Ranka, S. Sahni, and M. Schmalz, "Parallel dynamic data driven approaches for synthetic aperture radar," in High Performance Computing (HiPC), 2017 IEEE 24th International Conference on. IEEE, 2017, pp. 193–202.

List of Abbreviations and Acronyms

AFRL	Air Force Research Laboratory
AOI	Area of Interest
AVX	Advanced Vector Extensions
BRDF	Bidirectional Reflectivity Distribution Function
CPU	Central Processing Unit
CUDA	An architecture for design and programming of Graphics Processing Units (by Nvidia Inc.)
DDDAS	Dynamic Data-Driven Adaptive Systems
DoD	Department of Defense
DRAM	Dynamic Random-Access Memory
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HMP	Hybrid Multiprocessor
KNL	Knight's Landing (computer architecture by Intel)
LA	List Assignment (algorithm)
LPT	Longest Processing Time (algorithm)
MCDRAM	Multi-Channel Dynamic Random-Access Memory
MESIF	Modified, Exclusive, Shared, Invalid and Forward (protocol)
MPI	Message Passing Interface
ORNL	Oak Ridge National Laboratory
SAR	Synthetic Aperture Radar
SOW	Schedule of Work
TFlops	Trillions of Floating-Point Operations Per Second
TW	Total Work
UF	University of Florida
VSAR	Video Synthetic Aperture Radar
W	Watts (measure of power)
WAMI	Wide Area Motion Imaging