Adapting Parallel Backprojection to an FPGA Enhanced Distributed Computing Environment

Albert Conti, Ben Cordes,
Miriam Leeser, Eric Miller
Northeastern University

Richard Linderman
Air Force Research Labs

High Performance Embedded Computing
22 SEP 2005

Outline

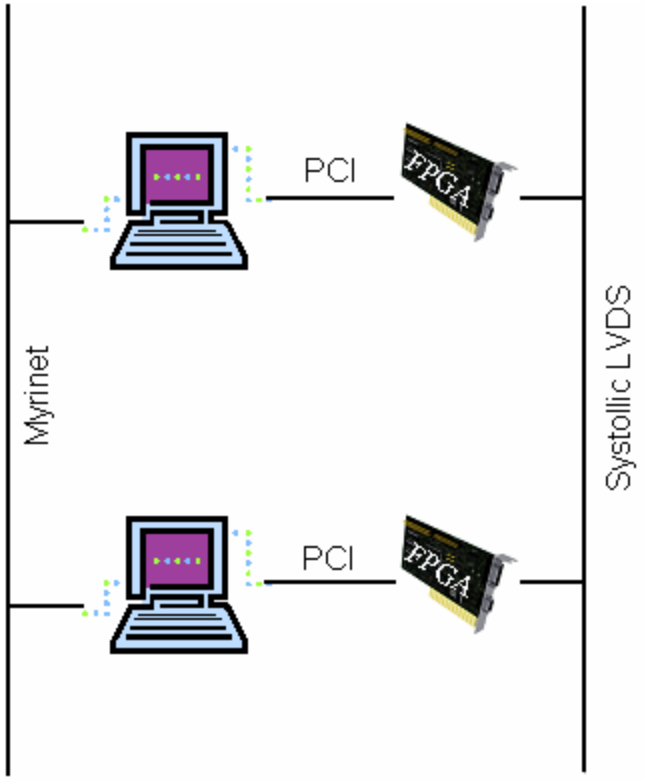
- HHPC Architecture
- Synthetic Aperture Radar and Backprojection
- Implementing Backprojection on the HHPC
- Results
- Analysis

HHPC Processing

- HPTi 48 Node Beowulf Cluster
- Each node
  - Dual 2.2 GHz Xeon processors
  - 4 Gigabytes of RAM
  - Annapolis Wildstar II FPGA board
    - 2 x Virtex II 6000
    - 12 Megabytes of SRAM
    - 128 Megabytes of DRAM
    - 32 bit PCI to host PC

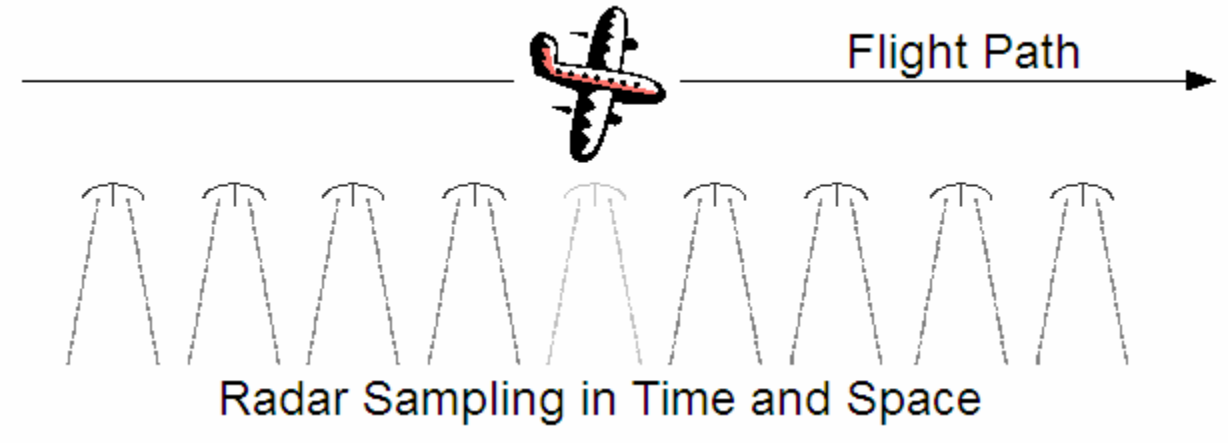
HHPC Communication

- MPI programming interface commands two communication channels
  - 10/100/1000 Ethernet
  - Multi-Gigabit Myrinet
- Each FPGA board has a daughter card that supports LVDS. LVDS interconnections are daisy chained.



Synthetic Aperture Radar

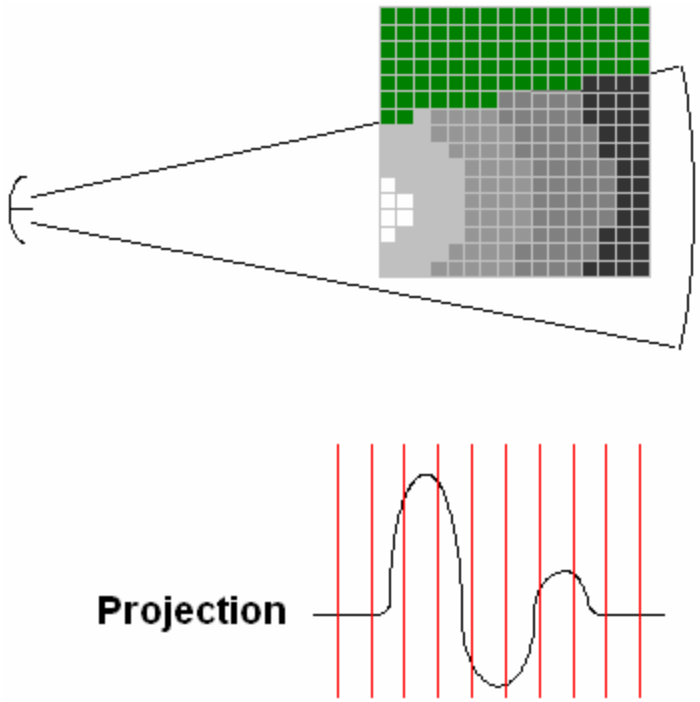
- Synthetic Aperture Radar (SAR) is a method through which high resolution images can be rendered with a single transceiver.
- Data from samples at different points in time and space are combined to synthetically generate an array of transceivers.



Radar Sampling in Time and Space

SAR Data

- SAR data exists as an array of time indexed projections of the target area as seen from the multiple sample points in space.
- Data in each time index of a projection represent the reflectivity of points in space a known distance away from the transceiver.
- Each pixel in the target area is a function of data at different time indices of each projection.



Backprojection

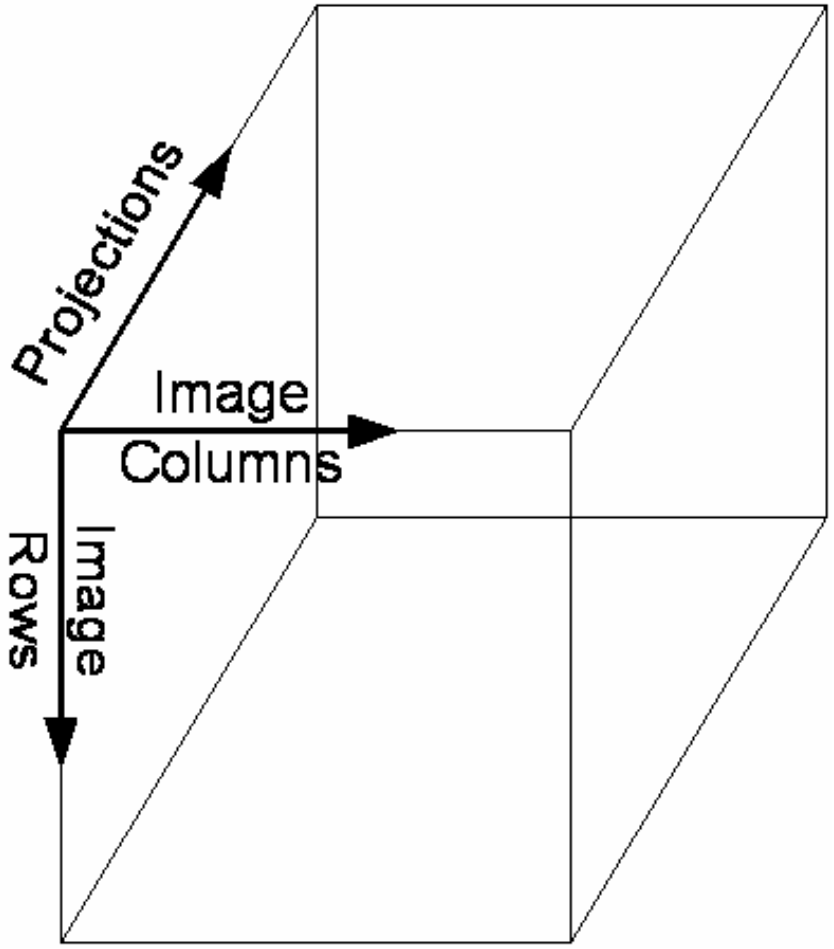
- Backprojection is an algorithm to reconstruct images from SAR data.
- For each pixel in the target image, data from each projection corresponding to that pixel is accumulated.
- Result is a collective value of reflectivity for each pixel which can be directly translated to a normalized RGB value to create an image.
- Backprojection is a highly parallelizable algorithm that filters out physical effects of radar.

Motivation for Acceleration

- With current radar technology, transceivers have the ability to collect gigabytes of data per second.
- Reconstructing images from such large amounts of data is computationally prohibitive in terms of processing and input bandwidth requirements.
- Currently, data is collected and then processed later on the ground.
- Clearly, processing in real time and on-site is desirable.

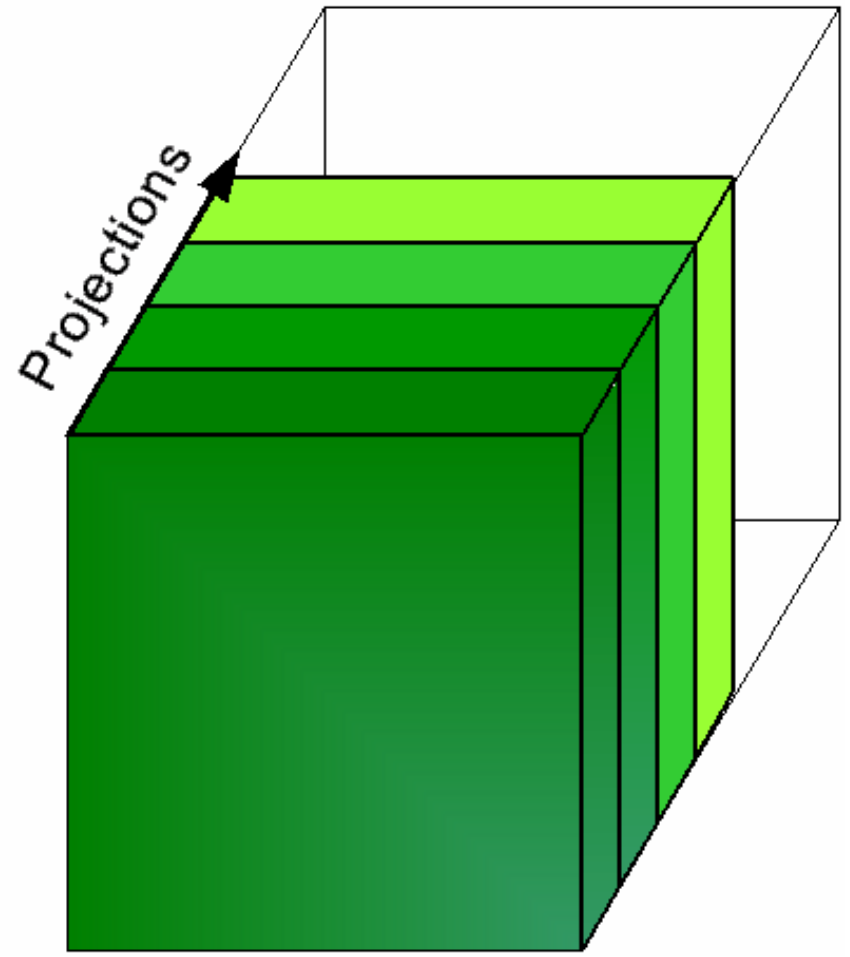
Parallelism

- Processing in parallel can provide for a gain in performance.
- Data dependencies reduce opportunity for acceleration due to parallel processing.
- Few data dependencies exist in backprojection algorithm for SAR.
- Both pixel level and projection level parallelism can be exploited with backprojection.



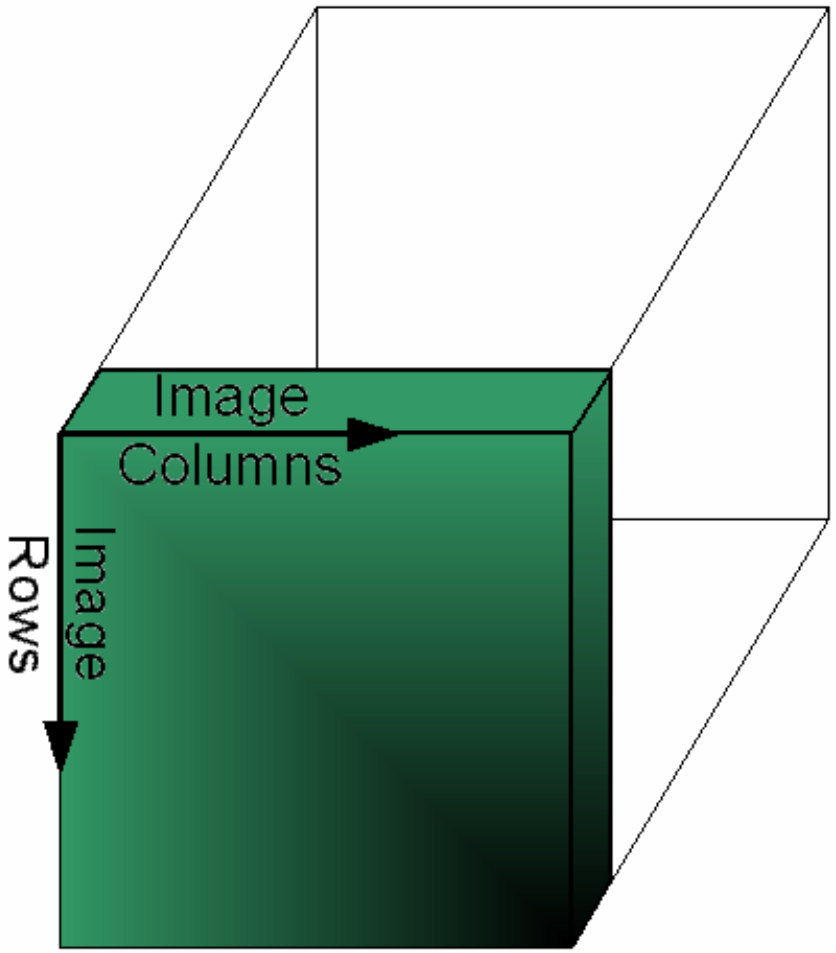
Projection Level Parallelism

- Projections can be indexed in parallel and data can be accumulated sequentially.
- Projection level parallelism is limited by input memory bandwidth as well as available processing elements.



Pixel Level Parallelism

- The target image can be partitioned spatially so that pixels can be processed in parallel.
- Pixel level parallelism is limited by input memory bandwidth.
- Our system exploits both pixel level and projection level parallelism.



Mapping to FPGAs

- Processing requirements are simple for backprojection
  - LUTs can be used to determine which index into each projection corresponds to each pixel in the target area.
  - After data is fetched, adders are all that is required for computation.
- Virtex II has more than enough resources to efficiently support parallel backprojection.
- Total size of target image that can be processed by a single FPGA is limited by the memory available to store the intermediate results.
- Global target image can be partitioned spatially and distributed between multiple FPGAs.

Implementation

- Single-Processing node is configured to backproject the maximum size target image.
- Exploit projection level parallelism using fine grain parallelism with FPGAs and on-board SRAM.
- Multi-node system partitions global target image according to single-node implementation and distributes processing.
- Exploit pixel level parallelism using coarse grain parallelism with MPI software.

Processing Node Data-Flow

1. Host loads projection data into memory from a shared disk.

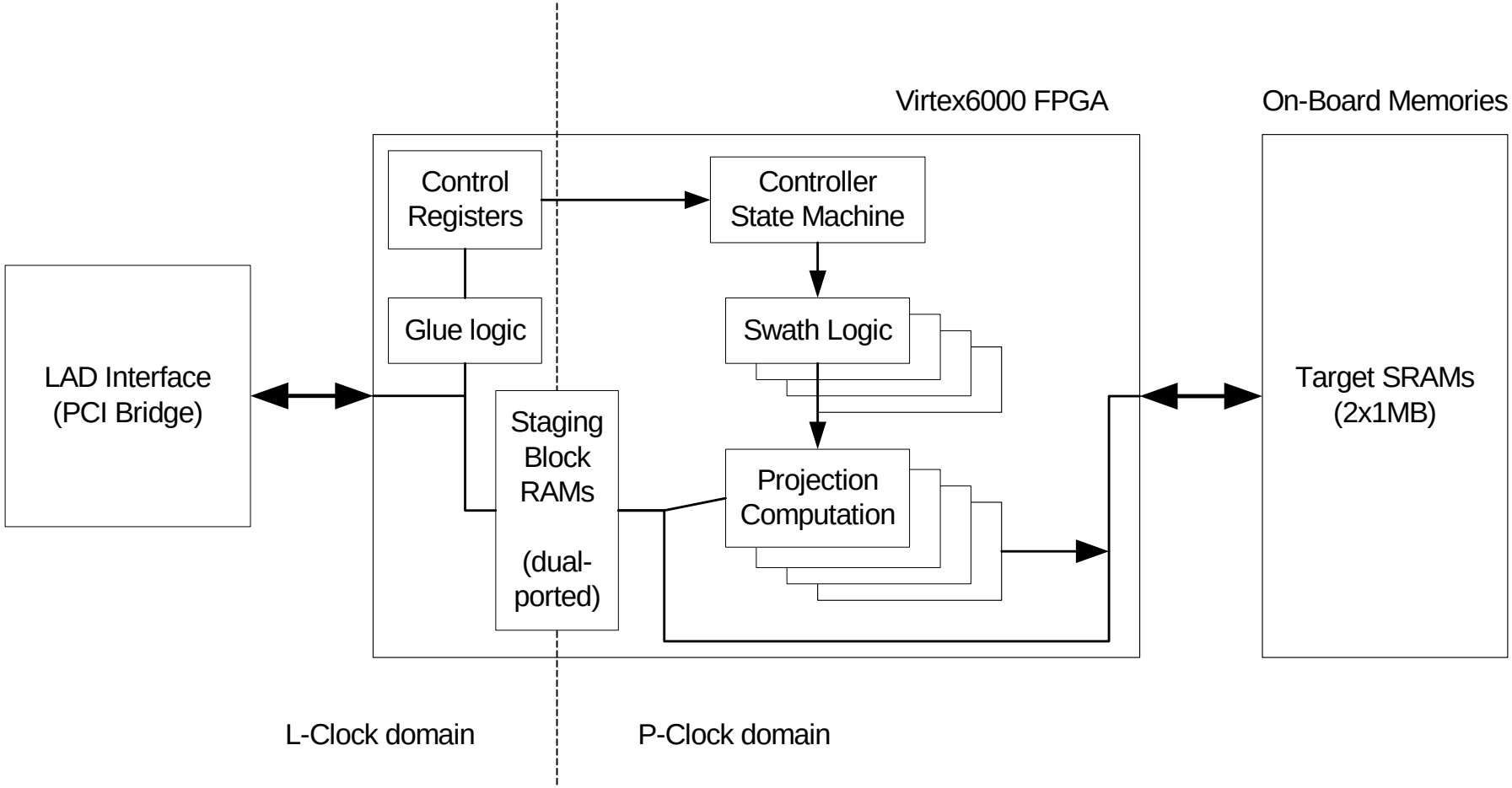
2. Host programs FPGA board with bitstream.

3. Host downloads as much projection data as hardware can process in parallel.

4. Repeat step 3 until all projections have been processed by hardware.

5. Host uploads target image data from FPGA board.

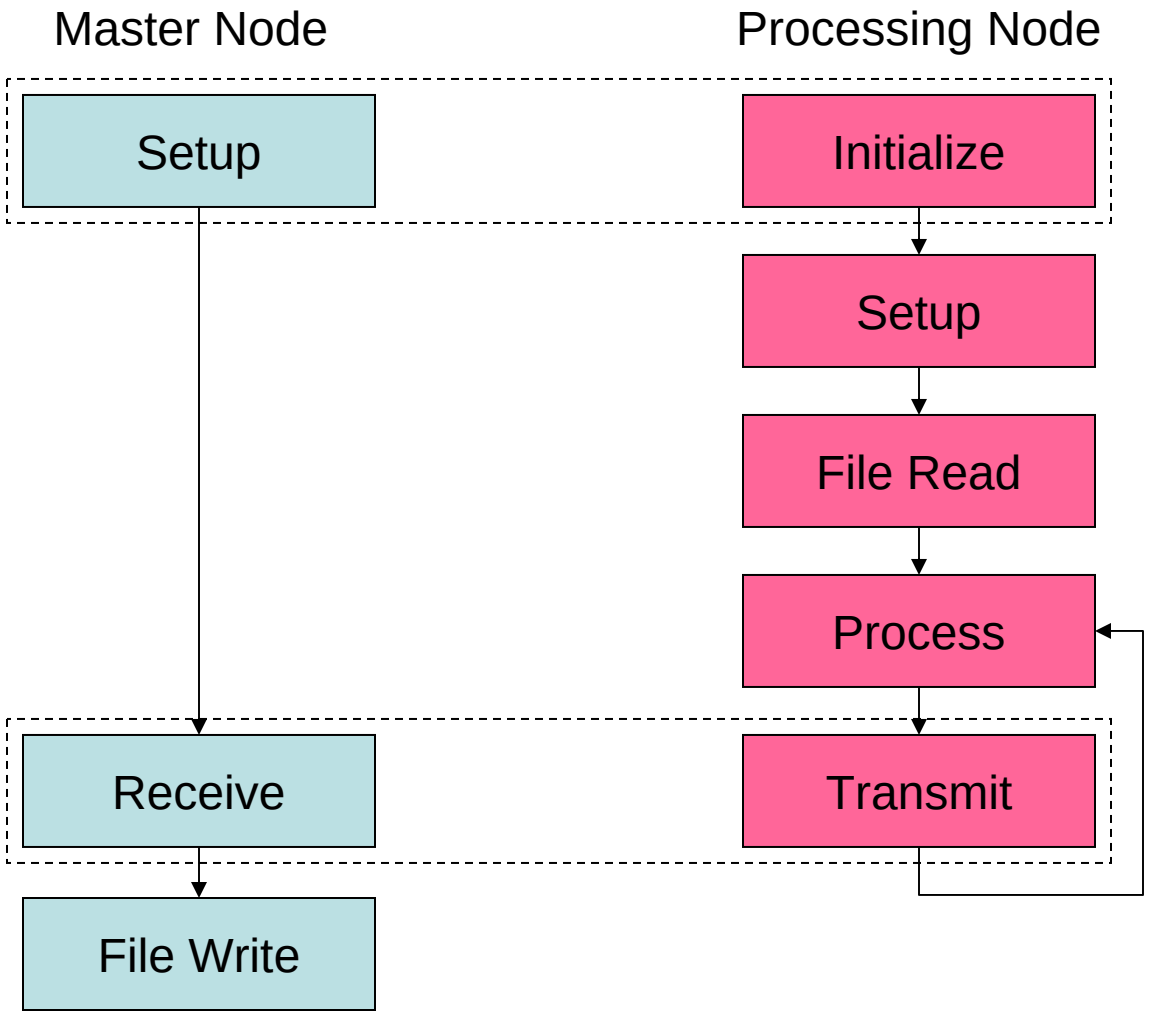
Processing Node Hardware



Using Multiple Processing Nodes

- Global target image is partitioned spatially to minimize (eliminate) the need to transmit redundant data.
- Exploit pixel level parallelism with MPI software.
- Each node processes a fixed size partition of the image. For global images that are too large, partitions are time-multiplexed.
- Master Node(s) is used to control processing and collect resulting images so they can be merged and written to a file.

Process Control



Data, Preprocessing

- To test our system and measure performance, we used Matlab to produce synthetic data.
- Matlab allowed us to test with multiple image dimensions, radar sample rates and plane speeds resulting in different values for image resolution.
- Data preprocessing/filtering/formatting done with C code to transform Matlab output into our system's input.

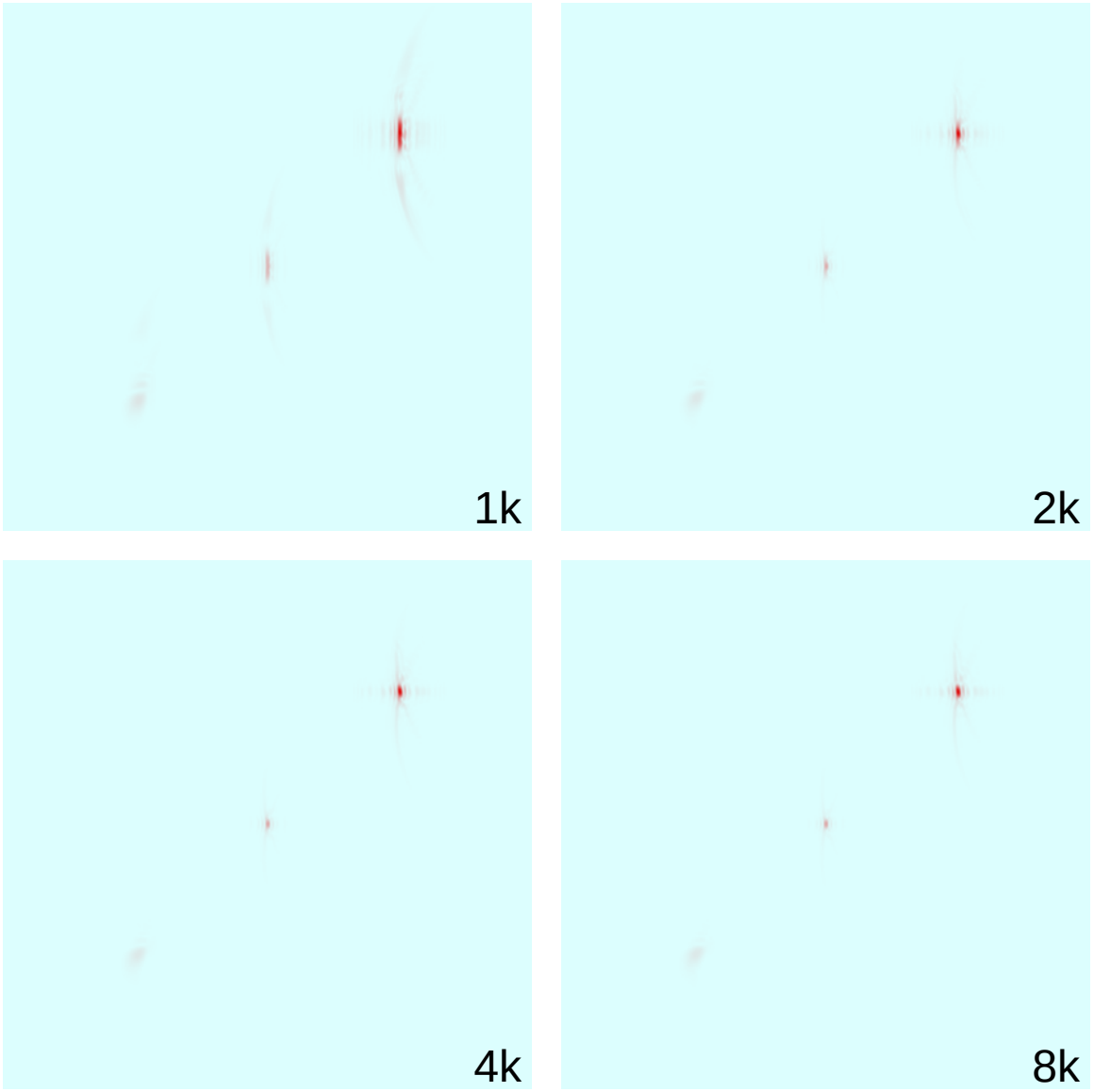
Soumekh, M. "Synthetic Aperture Radar Signal Processing with MATLAB Algorithms", ISBN 0-471-29706-2

Resolution

Images show the result of processing increasing amounts of data per projection.

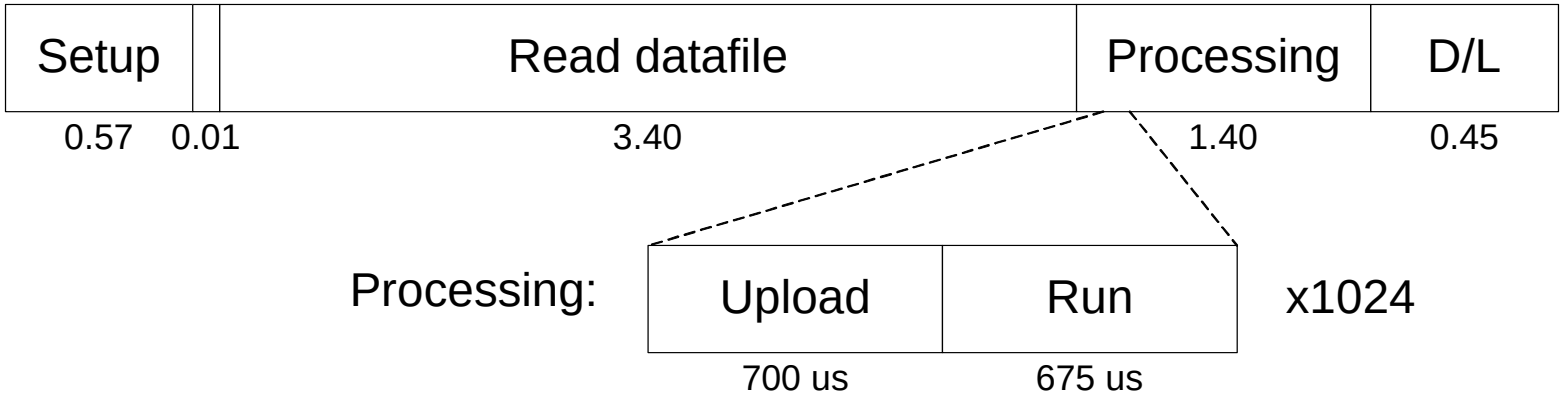
There is much redundancy in projection data.

For target images with equal range and azimuth resolution, we chose 4k.

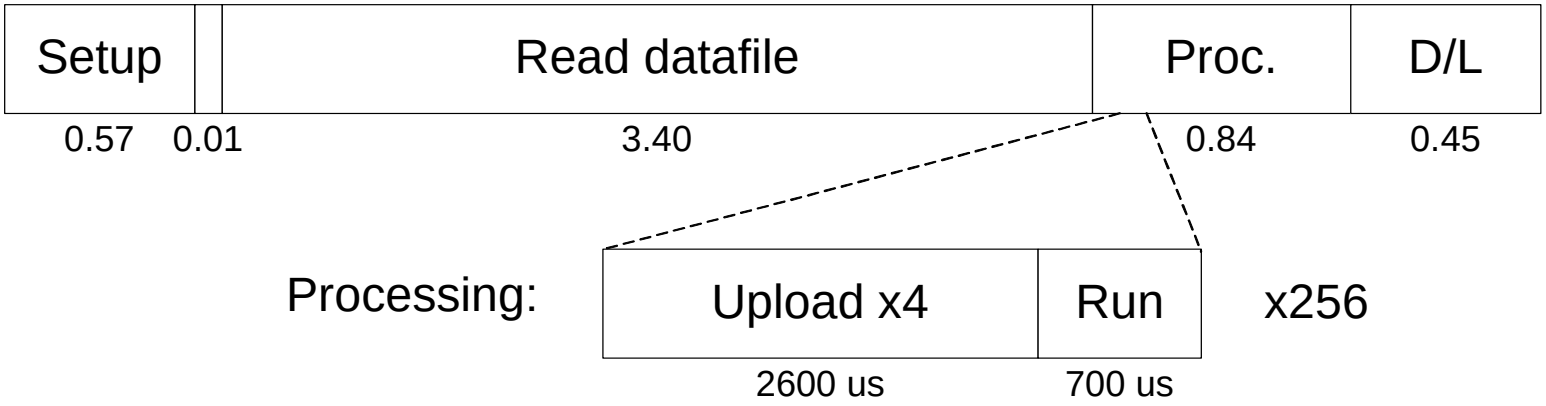


Processing Node Timing

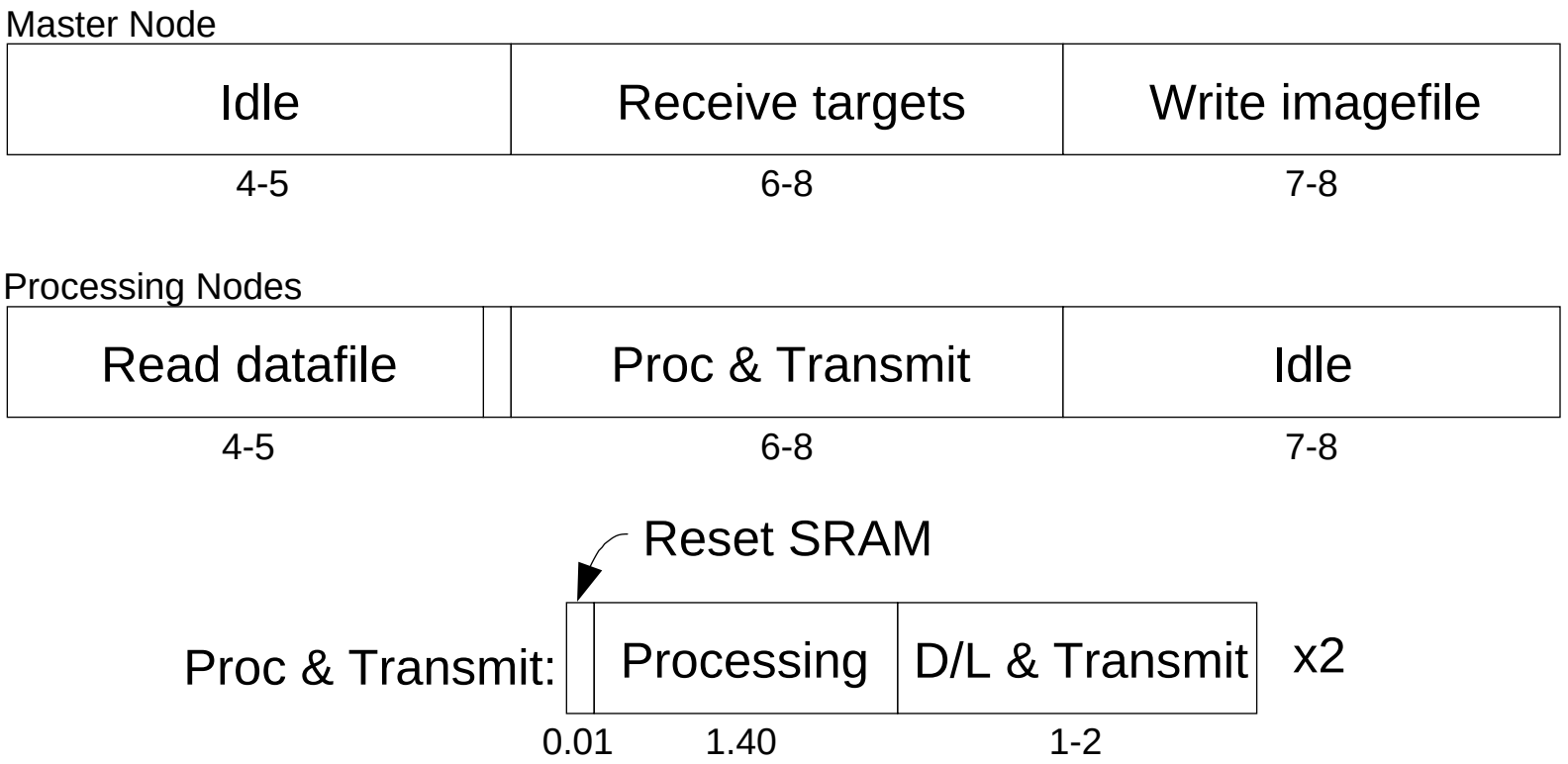
1 Projection Pipeline



4 Projection Pipeline



Multi-node Timing



Runtime diagram for 32 processing nodes reconstructing a global target area 64 times the size a single node can process.

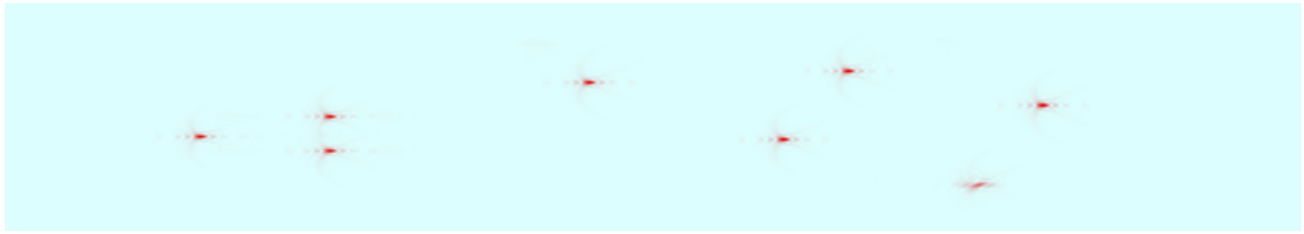
Benchmark, Comparison Basis

- Three sets of benchmark data were used to measure accuracy and runtime performance.
- Matlab code generates complex double precision floating point data after a transform to and from the frequency domain.
- Preprocessing filter transforms Matlab data into 16 bit fixed point data.
- Multiple parallel instances of our system as well an efficient software solution run on a single processor of the HHPC were tested.

Images

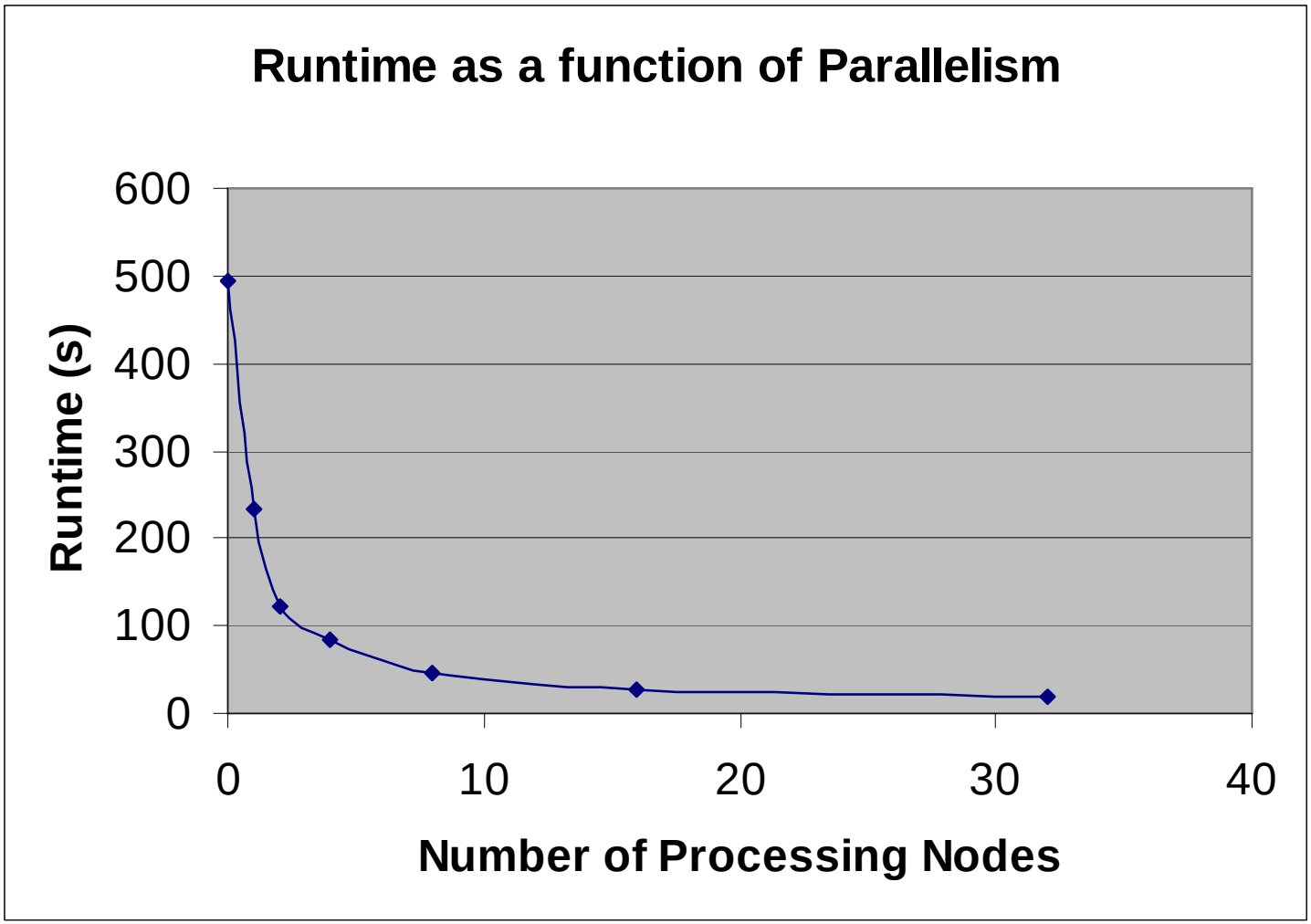
Single Target – High Reflectivity Coefficient

Three Targets – Ranging Reflectivity Coefficients



Eight Targets – Low Reflectivity Coefficients

Performance Comparison

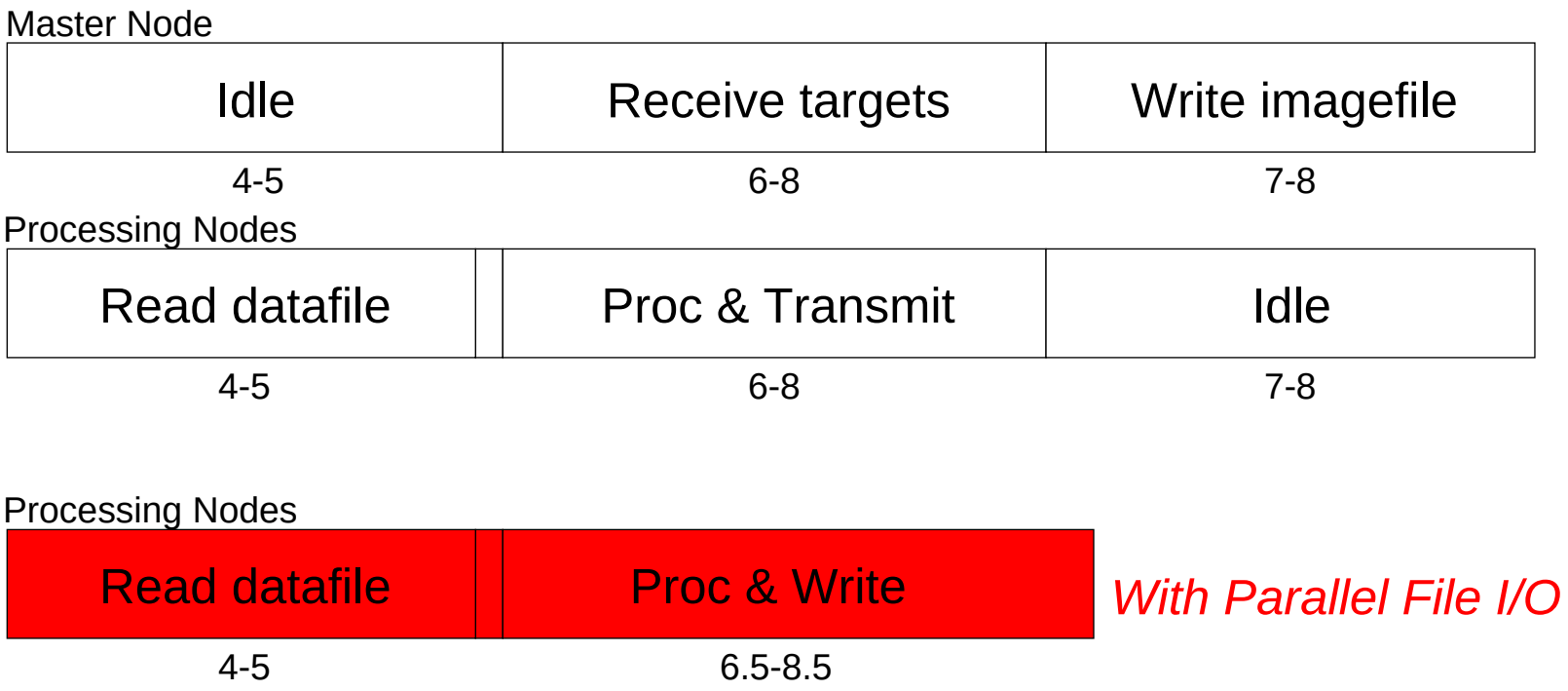


Bottlenecks

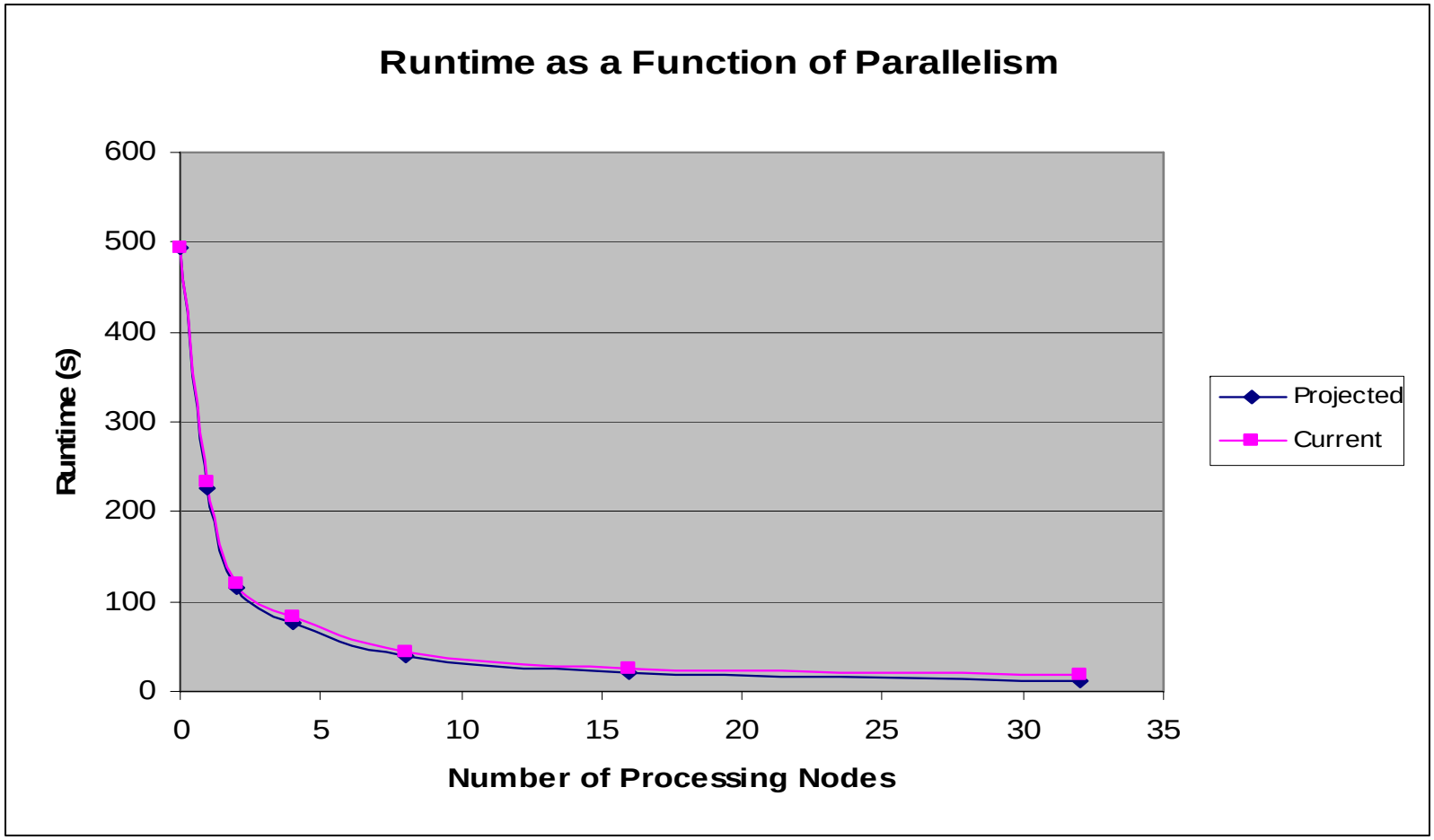
- Backprojection system is restricted by file I/O capabilities of the HHPC.
- Increasing projection level parallelism beyond our current implementation will not improve overall performance because input file read time dominates.
- Increasing pixel level parallelism so that a larger target area can be processed at once will not be able to improve much beyond our current implementation because output file write time dominates.

Parallel File I/O Performance Analysis

Parallel file I/O would allow for parallel backprojection without inter-process communication prior to the write back stage. There would be no need for a master node. Runtime projections are based on experimental results using multiple output files.



Projected Performance with Parallel File I/O



Overall Speedup would increase from 26x to no less than 41x.

Extensions, Improvements

- Current MPI libraries on HHPC do not support parallel file I/O. File system is major bottleneck of overall system.
- Currently, one of the two FPGAs on each Wildstar II is being utilized. Making use of the other FPGA so that access to the additional SRAM would make each node able to process a target image twice as large.
- DMA rather the PIO for data movement between host PC and Wildstar II.
- LVDS interconnect.

Summary

- We successfully mapped backprojection for SAR onto the AFRL HHPC.
- Made efficient use of the Annapolis Wildstar II FPGA board.
- Achieved 26x speedup over an efficient serial solution.
- Current system able to process data as fast as it can be read and written from the file system.

Acknowledgement

“This publication was made possible through support provided by DoD HPCMP PET activities through Mississippi State University under contract No. N62306-01-D-7110.”

For more information:

Northeastern Reconfigurable Computing Lab

www.ece.neu.edu/groups/rpl

AFRL/IF Distributed Center

www.rl.af.mil/tech/facilities/HPC/hpcf.html

Contact

aconti@ece.neu.edu