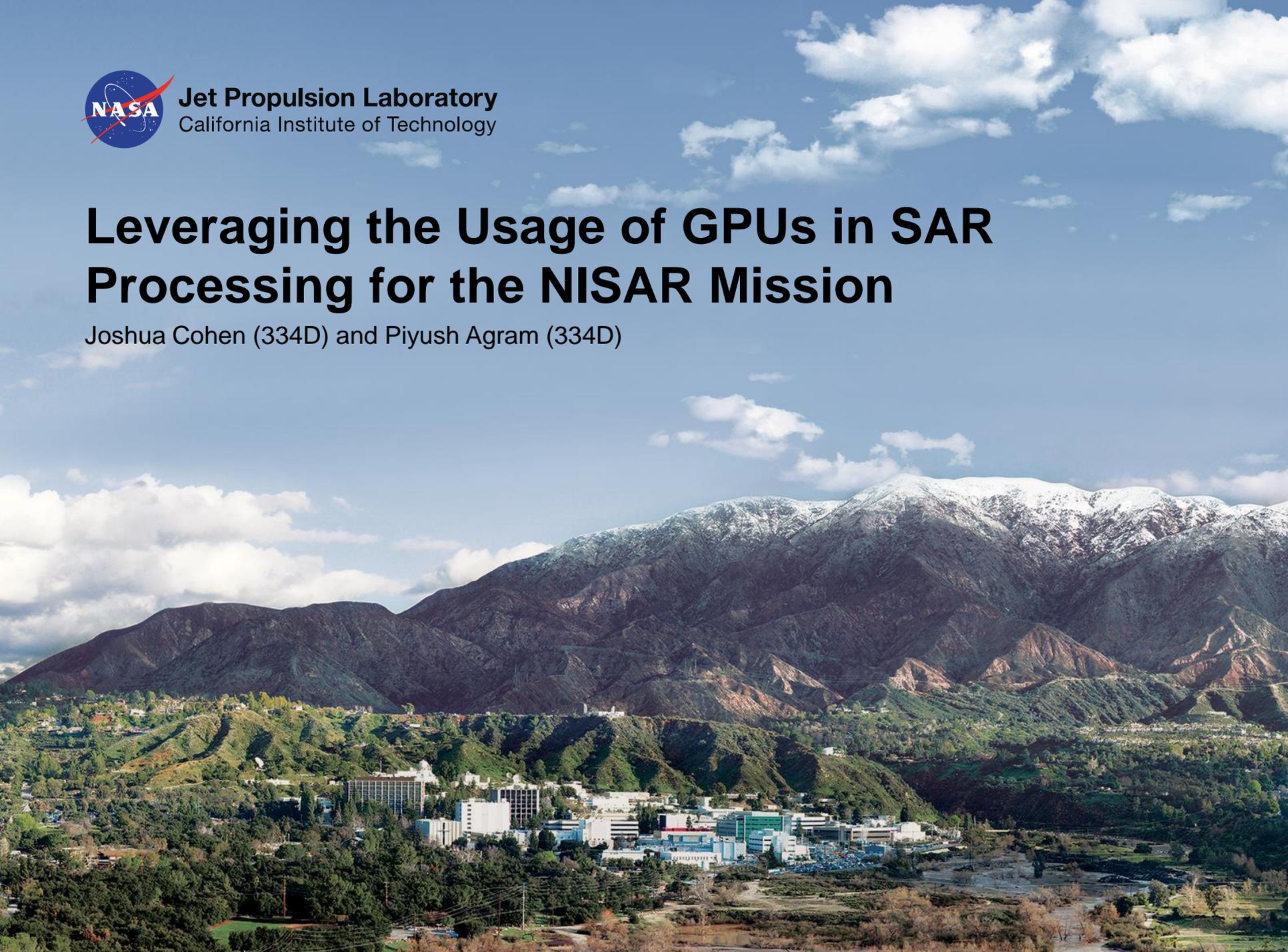




**Jet Propulsion Laboratory**  
California Institute of Technology

# Leveraging the Usage of GPUs in SAR Processing for the NISAR Mission

Joshua Cohen (334D) and Piyush Agram (334D)



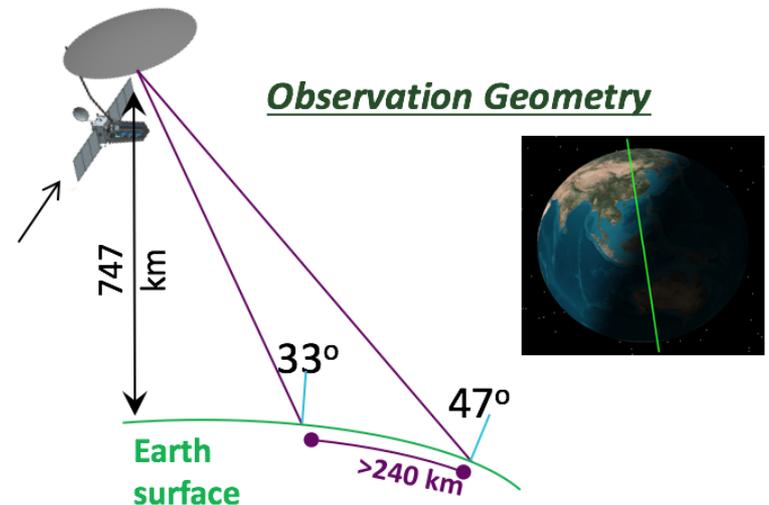
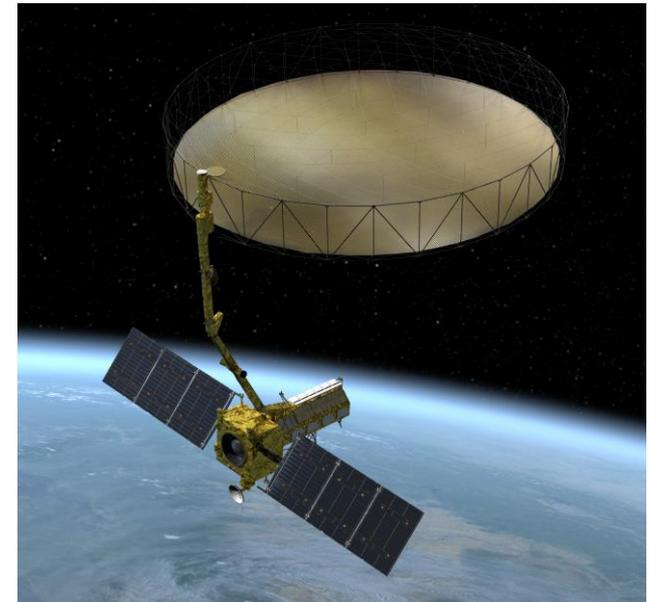
# Outline

- NASA-ISRO Synthetic Aperture Radar (NISAR) mission
- InSAR Scientific Computing Environment (ISCE)
- Topographic correction for SAR imagery
- Upgrading from a Single-Instruction/Single-Data (SISD) to a Single-Instruction/Multiple-Data (SIMD) design
- Benchmarking results and planned future work

# NASA-ISRO Synthetic Aperture Radar (NISAR)

## Mission Overview, Background

NISAR Characteristic:	Would Enable:
L-band (24 cm wavelength)	Low temporal decorrelation and foliage penetration
S-band (12 cm wavelength)	Sensitivity to light vegetation
SweepSAR technique with Imaging Swath > 240 km	Global data collection
Polarimetry (Single/Dual/Quad)	Surface characterization and biomass estimation
12-day exact repeat	Rapid Sampling
3 – 10 meters mode-dependent SAR resolution	Small-scale observations
3 years science operations (5 years consumables)	Time-series analysis
Pointing control < 273 arcseconds	Deformation interferometry
Orbit control < 500 meters	Deformation interferometry
> 30% observation duty cycle	Complete land/ice coverage
Left/Right pointing capability	Polar coverage, north and south



# NASA-ISRO Synthetic Aperture Radar (NISAR)

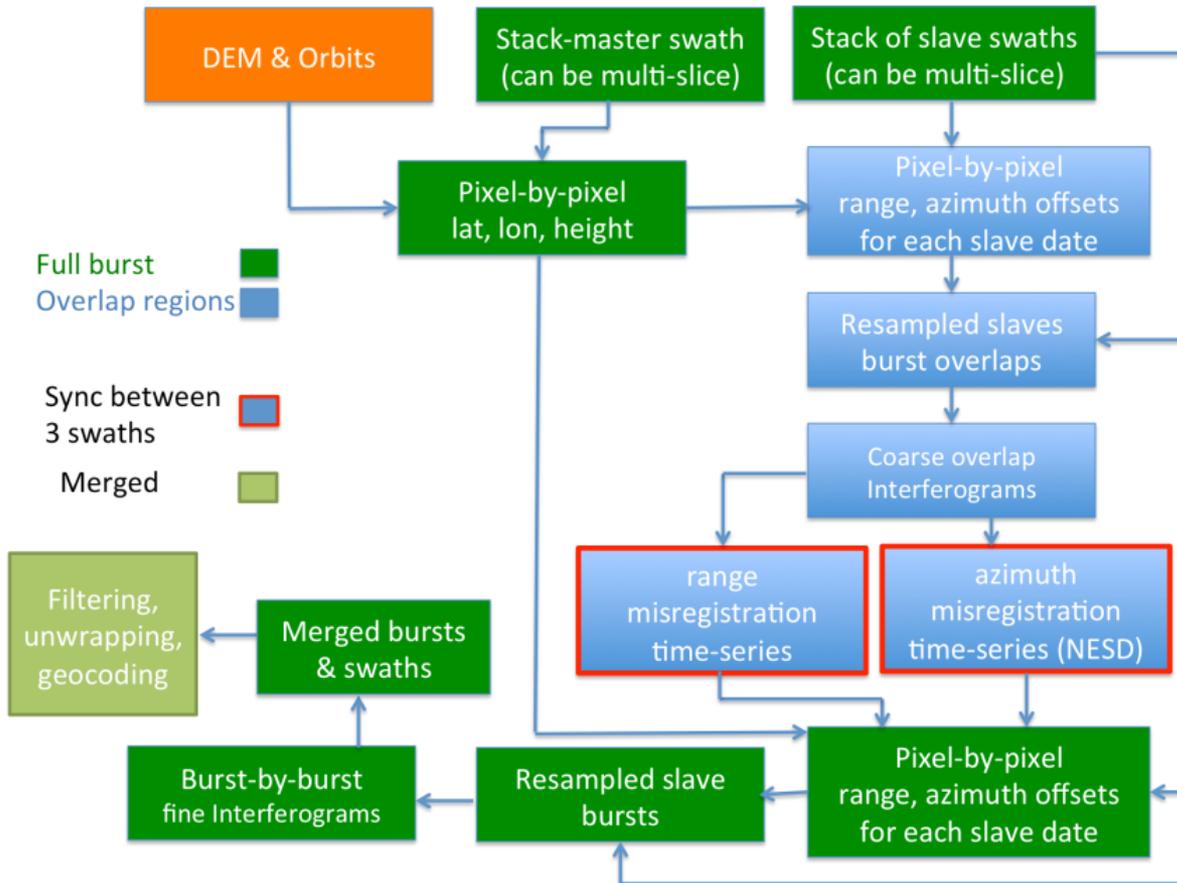
## Mission Overview, Big Data Problem

Key Driving Requirements	NISAR L3 MS/Algorithm Requirements
<b>Data Acquisition Volumes</b>	<ul style="list-style-type: none"> <li>• 26 Tbits/day (3.25 TB)</li> </ul>
<b>Prime Data Products</b> * excluding ancillary /auxiliary products	<ul style="list-style-type: none"> <li>• 14 (2 Level 0's, 2 Level 1's, 10 Level 2's)</li> </ul>
<b>Data Product Availability</b> * from availability of LOA data & needed inputs at JPL SDS	<ul style="list-style-type: none"> <li>• Level 0B none</li> <li>• Level 0B, Urgent Response 2 hrs</li> <li>• Level 1 30 days</li> <li>• Level 2 30 days</li> </ul>
<b>Processing/Reprocessing of science products</b>	<ul style="list-style-type: none"> <li>• Forward processing: Daily keep up</li> <li>• Post-Val reprocessing (optional): 4X speed               <ul style="list-style-type: none"> <li>- 8 months worth of data, all products (@Science + 8 months)</li> </ul> </li> <li>• End of mission reprocessing (optional): 1X speed               <ul style="list-style-type: none"> <li>- 12 months worth of data, all products (@Science Phase + 36 months)</li> </ul> </li> </ul>
<b>Mission Support</b>	<ul style="list-style-type: none"> <li>• Rolling Storage (leverage DAAC as the primary archive &amp; long-term back up)</li> <li>• Self-sufficient storage size for production</li> </ul>
<b>Total Data Product Volume</b>	<ul style="list-style-type: none"> <li>• Daily volume Forward processing: 95 TB/day               <ul style="list-style-type: none"> <li>- During Bulk Reprocessing (peak): 475TB/day</li> </ul> </li> <li>• Mission total (Forward): 105PB / 3 yr               <ul style="list-style-type: none"> <li>- Mission total (Forward + Reprocessing): 163 PB</li> </ul> </li> </ul>

*Need fast, large-scale scientific processing*

# NASA-ISRO Synthetic Aperture Radar (NISAR)

## Interferometric SAR (InSAR) Processing Workflow



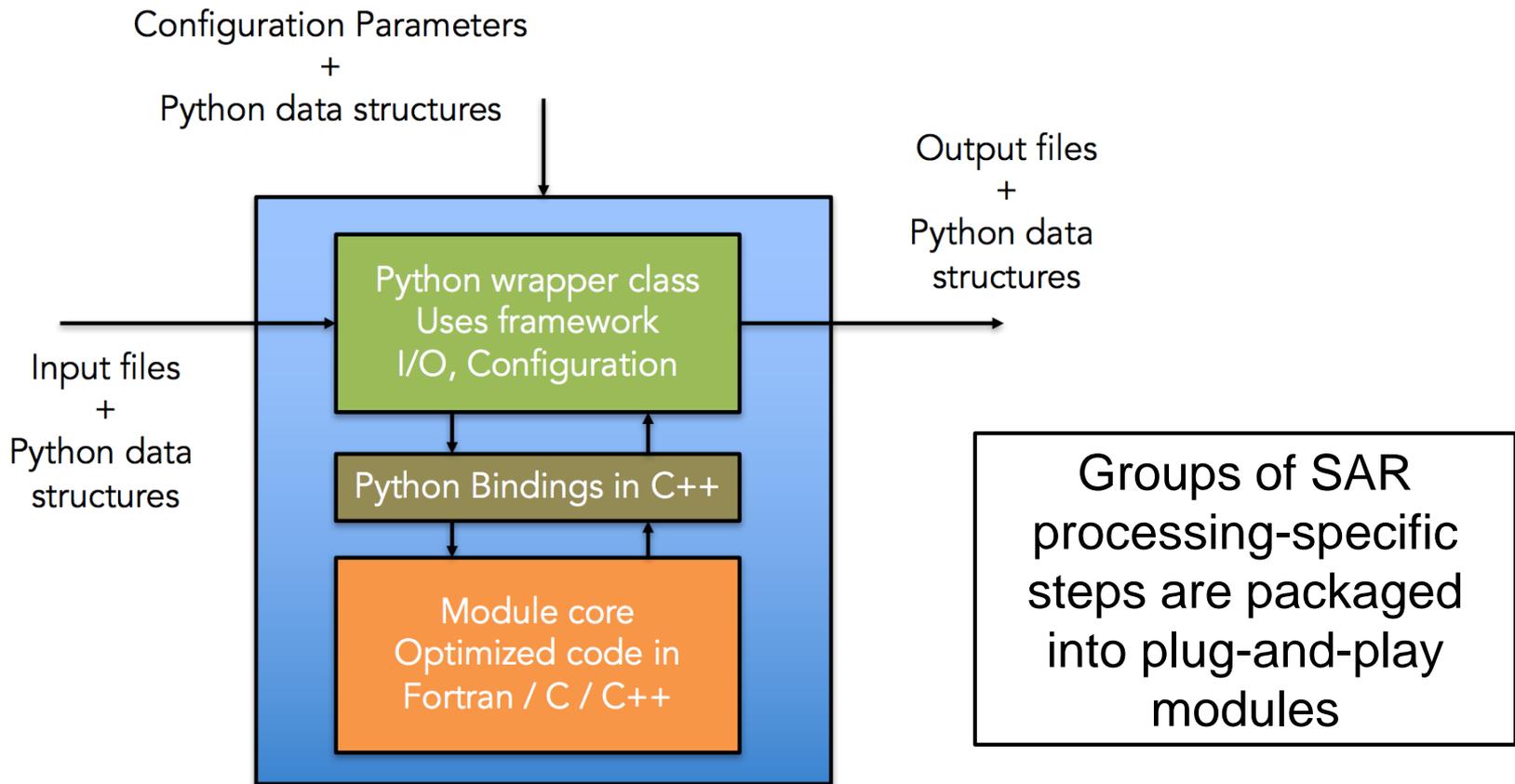
- Multiple compartmentalized steps
- Each step might take multiple inputs and produce as much if not significantly more output data
  - Ex: “Pixel-by-pixel lat, lon, height” takes a radar image, a DEM, and orbit information and produces 5-7 output images of the same size as the input image

*Sentinel-1 InSAR processing workflow example (Fattahi et al, 2016)*

# InSAR Scientific Computing Environment (ISCE)

## Overview, Modular Design

A flexible framework of InSAR software applications and algorithms



*Example of an ISCE "module"*

# InSAR Scientific Computing Environment (ISCE)

## InSAR Processing Steps

- Combination of legacy and novel applications to handle geocoding, focusing, calibration, correction, and more
- Four InSAR processing modules relevant to this talk:
  - Topo: Forward mapping geometry algorithm to convert an image in slant-range coordinates to ground-range coordinates, as well as correct for topography
  - Ampcor: Normalized image cross-correlation algorithm to generate a 2D coregistration polynomial
  - Geo2rdr: Inverse of the forward mapping geometry algorithm to convert an image in ground-range coordinates to radar slant-range coordinates
  - Resamp\_slc: Image correction algorithm to resample a slave image against its master image using coregistration polynomials or offset fields
- Each of the four modules is a good candidate for “GPU-ization”

# InSAR Scientific Computing Environment (ISCE)

## Key Feature: OpenMP Parallelization

- Many algorithms repetitively apply a set of operations to single pixels
- Basic acceleration – leverage the cores in your CPU, split the repetition
- OpenMP is a set of code “decorators” that can apply basic automatic parallelization

```
#pragma omp parallel for  
for (i=0; i<100; i++) {  
    ...some work...  
}
```

(assuming 2 cores)

Thread 1: for (i=0; i<25; i++) { ...some work... }

Thread 2: for (i=26; i<50; i++) { ...some work... }

Thread 3: for (i=51; i<75; i++) { ...some work... }

Thread 4: for (i=76; i<100; i++) { ...some work... }

# Project Goals

Driving question: Can we develop new versions of existing ISCE modules (with an eye towards processing large images like NISAR's) that:

- Are vastly more parallelized than those using OpenMP
- Are algorithmically similar to the original modules
- Guarantee the same numerical precision as the original modules

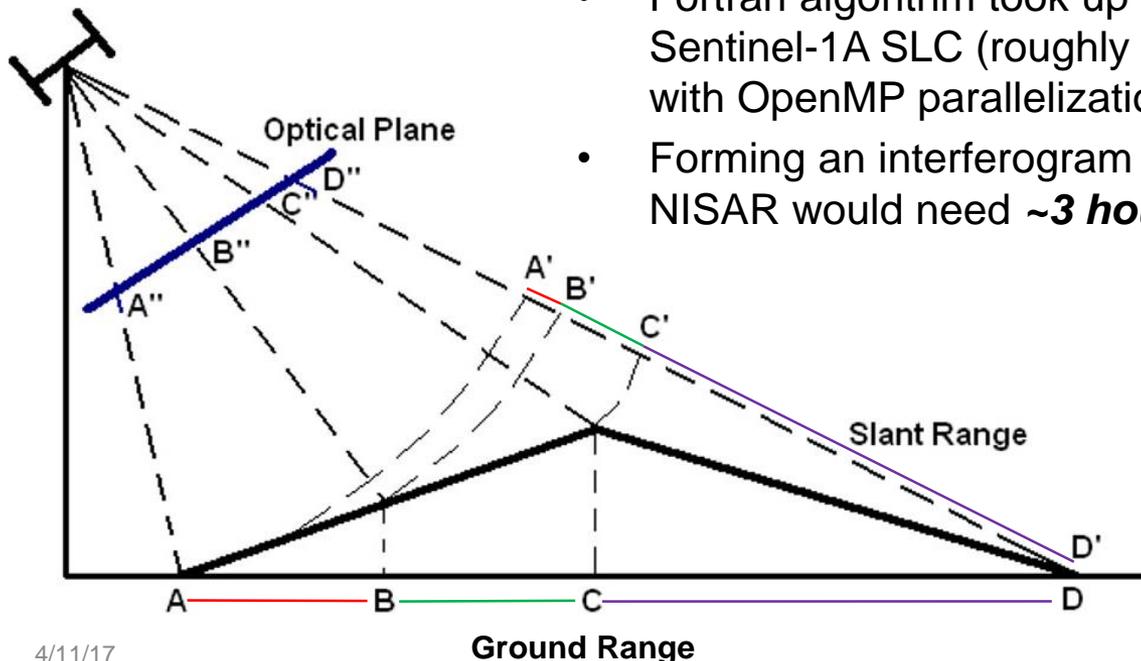
*This talk focuses on one highly parallelizable and computationally intensive module: “Topo”*

# ISCE's Forward Geometry "Topo" Module

## Overview

- Standard SAR processing module to convert a given input image in slant-range coordinates to corresponding ground-range coordinates
  - Module essentially solves:
$$R_{ground} \propto R_{slant} \cdot \sin \theta_{incidence}$$
for each pixel in the image using a reference DEM
  - Corrects for shadow, layover, foreshortening effects
  - Fortran algorithm took up to ~90 minutes for a large Sentinel-1A SLC (roughly 20,000 x 60,000 pixels), even with OpenMP parallelization
  - Forming an interferogram needs two runs of Topo, so for NISAR would need **~3 hours** for this step alone

$$\begin{aligned} \overline{C'D'} &\cong \overline{CD} \\ \overline{B'C'} &< \overline{BC} \\ \overline{A'B'} &\ll \overline{AB} \end{aligned}$$

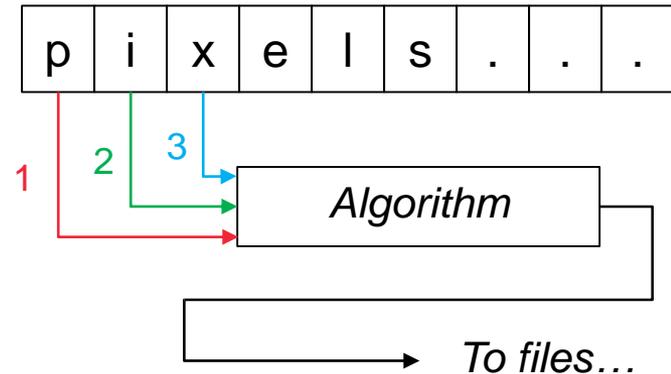


# Single-Instruction/Single-Data to /Multiple-Data

## SISD Algorithm Design

- In SISD, a single algorithm operates on single pixels
- General purpose CPUs are inherently SISD optimized
- For a given algorithm, wall-clock runtime scales directly with number of pixels
- Can be characterized by:

$$T_{CPU} = N_{total\_pixels} \times T_{pixel}$$



Algorithm	Input
Step 1	Pixel 1
Step 2	Pixel 1
Step 3	Pixel 1
...	Pixel 1
Write-to-file	Pixel 1
Step 1	Pixel 2
Step 2	Pixel 2

etc...

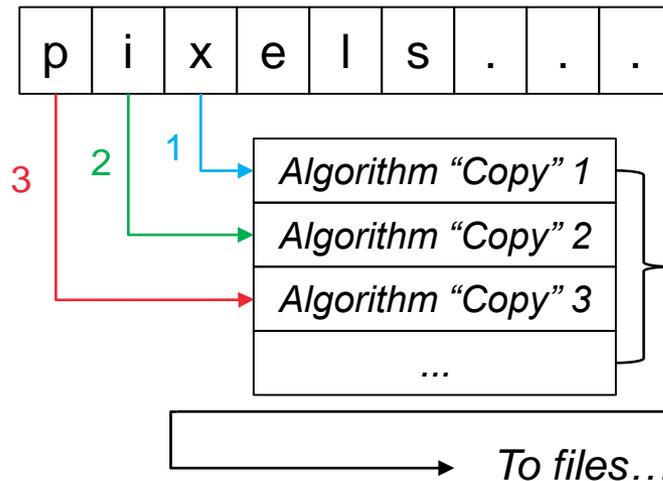
# Single-Instruction/Single-Data to /Multiple-Data

## SIMD Algorithm Design

- In SIMD, many identical copies of an algorithm operate on multiple “data streams”
- Instruction execution is slower than that of CPUs due to explicit operations that are performed implicitly in CPU hardware for SISD...
- ...but “hyper-parallelism” can easily mitigate this slowdown
- Characterized by:

$$T_{GPU} = \frac{N_{total\_pixels} \times T_{pixel}}{N_{pixels\_per\_block}}$$

Note: OpenMP is pseudo-SIMD, and already implemented in ISCE



Algorithm	AC 1 Input	AC 2 Input	AC 3 Input	...
Step 1	Pixel 1	Pixel 2	Pixel 3	...
...	Pixel 1	Pixel 2	Pixel 3	...
Write-to-file	Pixel 1	Pixel 2	Pixel 3	...
Step 1	Pixel 4	Pixel 5	Pixel 6	...

etc...

# Single-Instruction/Single-Data to /Multiple-Data

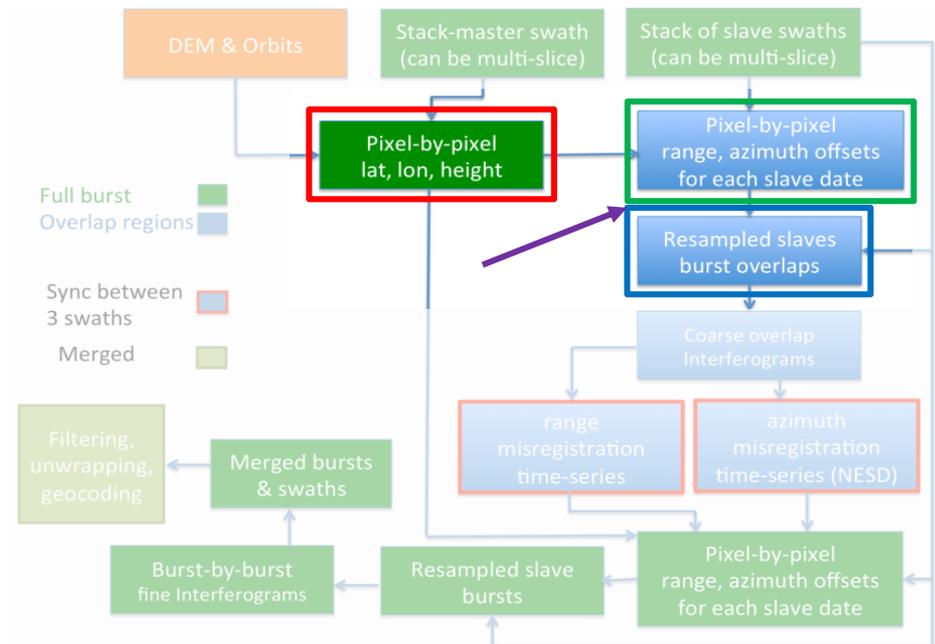
## GPU-based SAR Processing

- GPUs have shown to be effective accelerators for standard SAR image formation and processing before, ex:
  - “Synthetic Aperture Radar imaging on a CUDA-enabled mobile platform” (Fatica and Phillips, 2014) – SAR image formation and L1 processing on an embedded GPU platform
  - “SAR Image Processing using GPU” (Maddikonda and Sundaram, 2014) – SAR L1.5 image processing, Digital Beam Forming, applying a “smoothing” filter (denoising)
  - “Techniques for Mapping Synthetic Aperture Radar Processing Algorithms to Multi-GPU Clusters” (Hayden et al, 2012) – SAR L1 image formation and processing, multi-GPU processing load distribution
- Question: Can we extend these types of results by accelerating InSAR (at least L2) processing steps such as geocoding and coregistration using GPU-enabled algorithms?

# Single-Instruction/Single-Data to /Multiple-Data

## SISD-to-SIMD Module Conversion

- For ISCE specifically, four modules were identified for GPU algorithm development
  - **Topo**, **Ampcor**, **Resamp\_slc**, **Geo2rdr**
- GPU modules were developed with 3 steps:
  - Refactor legacy Fortran code and create an object-oriented C++ equivalent
  - Identify and examine the core algorithm, and develop a new GPU-accelerated CUDA code
  - Develop module framework to implement CPU or GPU code depending GPU availability



# Development Foci

Simplicity, Similarity, Precision

- **Simplicity** – To benchmark worst-case scenario, few CUDA- or GPU-specific optimizations applied
  - Example: Some advanced CUDA implementations allow for “dynamic parallelism”, where a set of threads within a parallelized “kernel” can each spawn their own kernel of parallel threads. Potentially faster, but more complex
- **Similarity** – To allow for easier parallel codebase maintenance, algorithms kept as identical as possible
  - Example: Not replacing known functions with unknown ones from the CUDA math libraries, e.g.  $a = \text{sqrt}(\text{pow}(b[0],2) + \text{pow}(b[1],2) + \text{pow}(b[2],2))$  (known) versus  $a = \text{norm}(3, b)$  (CUDA math library)
- **Precision** – To ensure that results align with expected values from C++ algorithm, refrain from implementing non-double-precision functions
  - Example: CUDA math library contains “intrinsic” which are GPU-optimized versions of standard functions, at the potential cost of IEEE-754 compliant precision

# Benchmarking Results

## “Topo” Timing Results

Image Scale	CPU Runtime	GPU Runtime	Speedup
COSMO-SkyMed	18 m	30 s	36x
NISAR	82 m	3 m	27.3x

- Two test scenes used:
  - COSMO-SkyMed; 40 km x 40 km;  $\sim 4 \times 10^8$  pixels
  - Sentinel-1A (NISAR-scale); 240 km x 240 km;  $\sim 1.2 \times 10^9$  pixels
- Hardware details:
  - CPU: High-frequency Intel Xeon E5-2670, 12 threads enabled
  - GPU:  $\frac{1}{2}$  NVIDIA Tesla K80, 1 GK210B GPU and 12 GB memory
- GPU runtime limited by AWS GovCloud I/O throttling, limited memory bus speeds (common to all GPUs)

# Benchmarking Results

## Rough “Topo” Implementation Costs

	<u>CPU</u>	<u>GRID K520</u>	<u>Tesla K80</u>
Runtime (COSMO-SkyMed scale)	18 minutes	3 minutes	30 seconds
Runtime (NISAR/Sentinel-scale)	82 minutes	(est.) 20 minutes	3 minutes
Total “Topo” Runtime (1400 NISAR-scale interferograms)	1913.333 hours	466.666 hours	70 hours
Cost (per node)	\$.702 / hour	\$.702 / hour	\$.900 / hour
Estimated # of nodes per day	80	20	3
Estimated cost per day	\$1,347.84	\$336.96	\$64.80
Total Speedup	1x	~5x	~30x

# Summary

- GPU implementation of radar-to-ground coordinate mapping show ~30x speed improvement over OpenMP implementation on a CPU
- Further optimizations are possible, at the cost of added complexity and dissimilarity to original code
- Three other modules have been “GPU-ized” with similar speedup in all cases
- Module development time decreased significantly as most of the code development focused on creating the proper logical structure for the algorithms
  - Note that these logical structures translated to other algorithms fairly easily

# Planned Future Work

“GPU-izing” the SAR processor

- Plan to implement several algorithms – range-doppler, omega-K, hybrid approaches
- Must address large (30,000 x 30,000) images with large synthetic aperture convolution kernels
- Must address non-uniform sampling along-track for “SweepSAR” implementation
- Should be consistent with simplicity, similarity, and precision principles previously described
- Leverage published GPU methods for SAR processing

Questions?



**Jet Propulsion Laboratory**  
California Institute of Technology

---

[jpl.nasa.gov](http://jpl.nasa.gov)