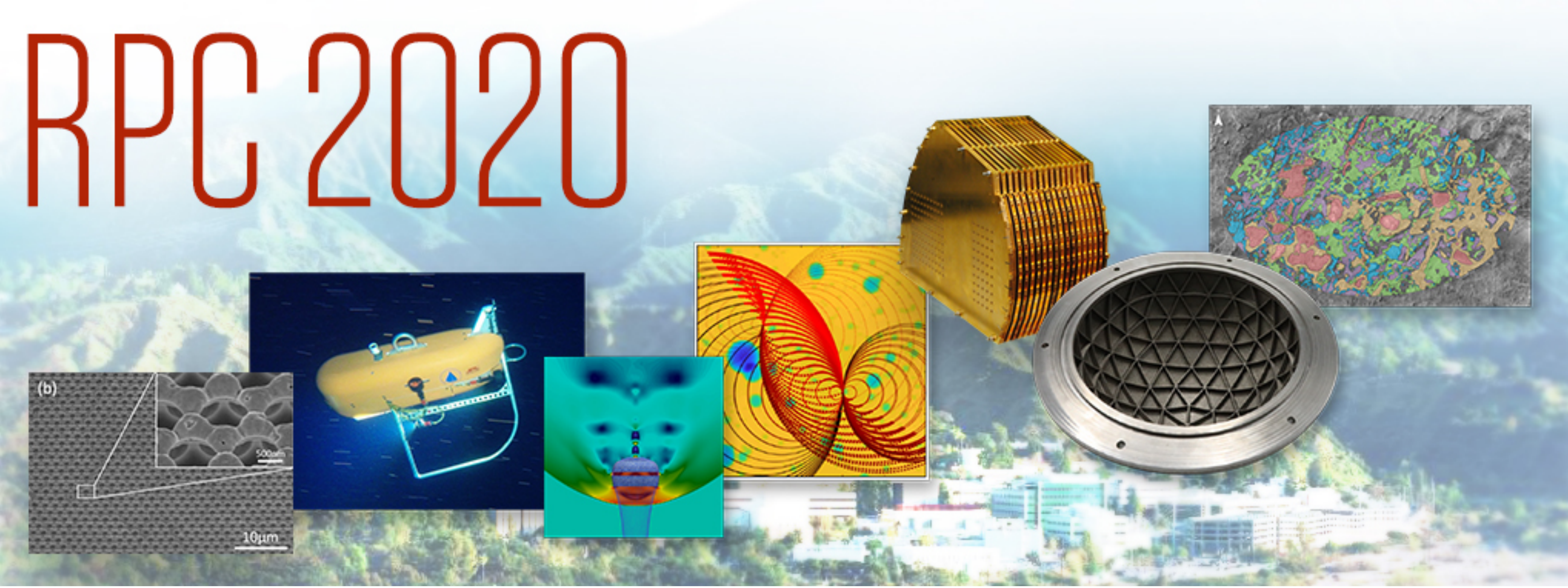


Virtual Research Presentation Conference

A Scalable, Flexible Instrument Simulation Toolkit for Mission Design

Principal Investigator: Brian Wilson (174B)
Co-Is: Derek Posselt, Rachel Storer, Noppasin Niamsuwan, Benyng Tang, Berlin Chen
Program: Topic

#R19135

Jet Propulsion Laboratory
California Institute of Technology

Highlights:

- Using Cluster computing to run 100+ Million OSSE's to characterize science return and quantify uncertainties for various mission designs.
- Led to a won **AIST'18** Project: Evaluating Designated Observables for the Aerosol and Cloud, Convection & Precipitation (ACCP) Mission
- Developed open-Source **PARMAP** Library: Re-deployable Map-Reduce parallel computing with a choice of backend schedulers: multicore, Spark, Dask, and AWS on-demand Lambda functions.

*OSSE = Observation System Simulation Experiment

Parallel OSSE Toolkit: A Collaboration between Scientists & Technologists

Technologists:
PI Brian Wilson
Sujen Shah
Chris Mattmann

Computing Technologies

Apache PySpark
SciSpark – AIST14
Py Mat/Vec for GPU

Instrument Simulation Codes

QuickBeam
NEOS3
Statistical Analysis

OSSE Team:
Derek Posselt
Rachel Storer
Ethan Nelson
Noppasin Niamsuwan
Benyang Tang

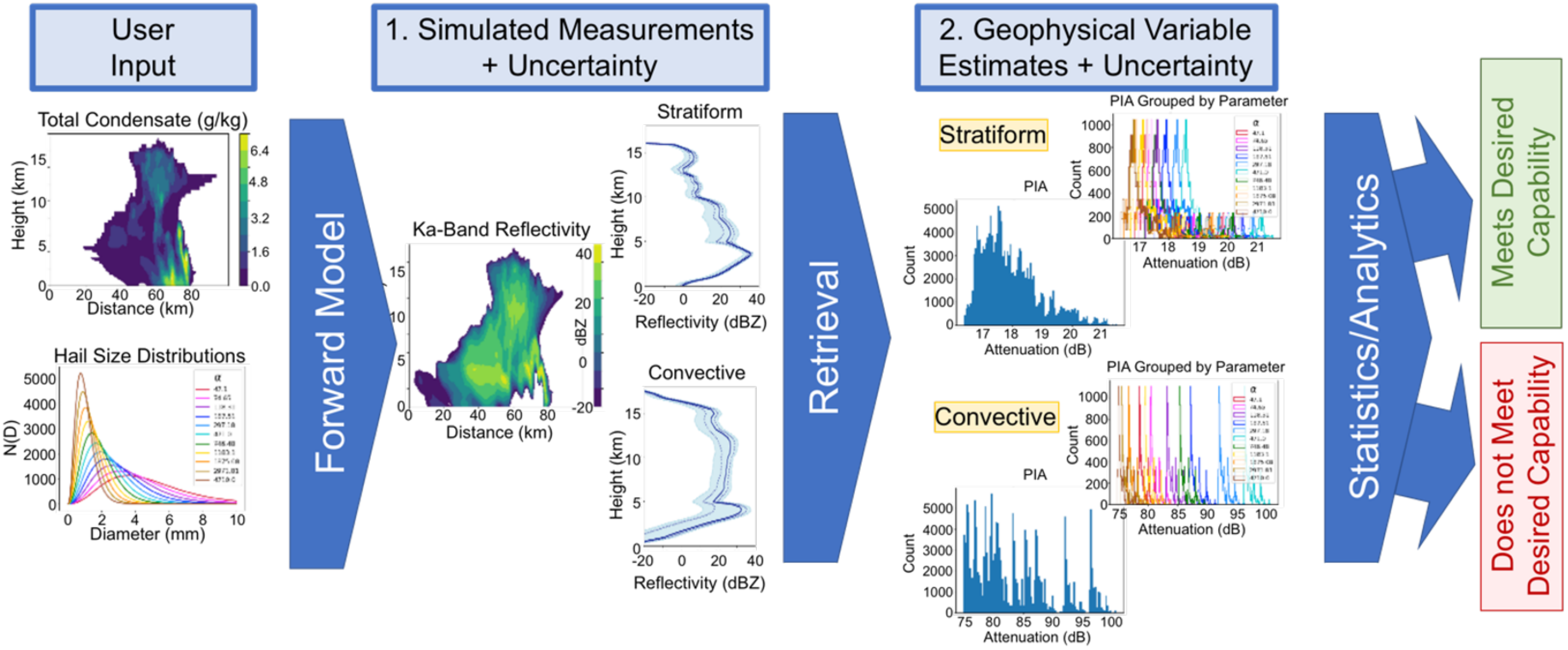
Spark Cluster

Parallel OSSE Toolkit for Mission Design

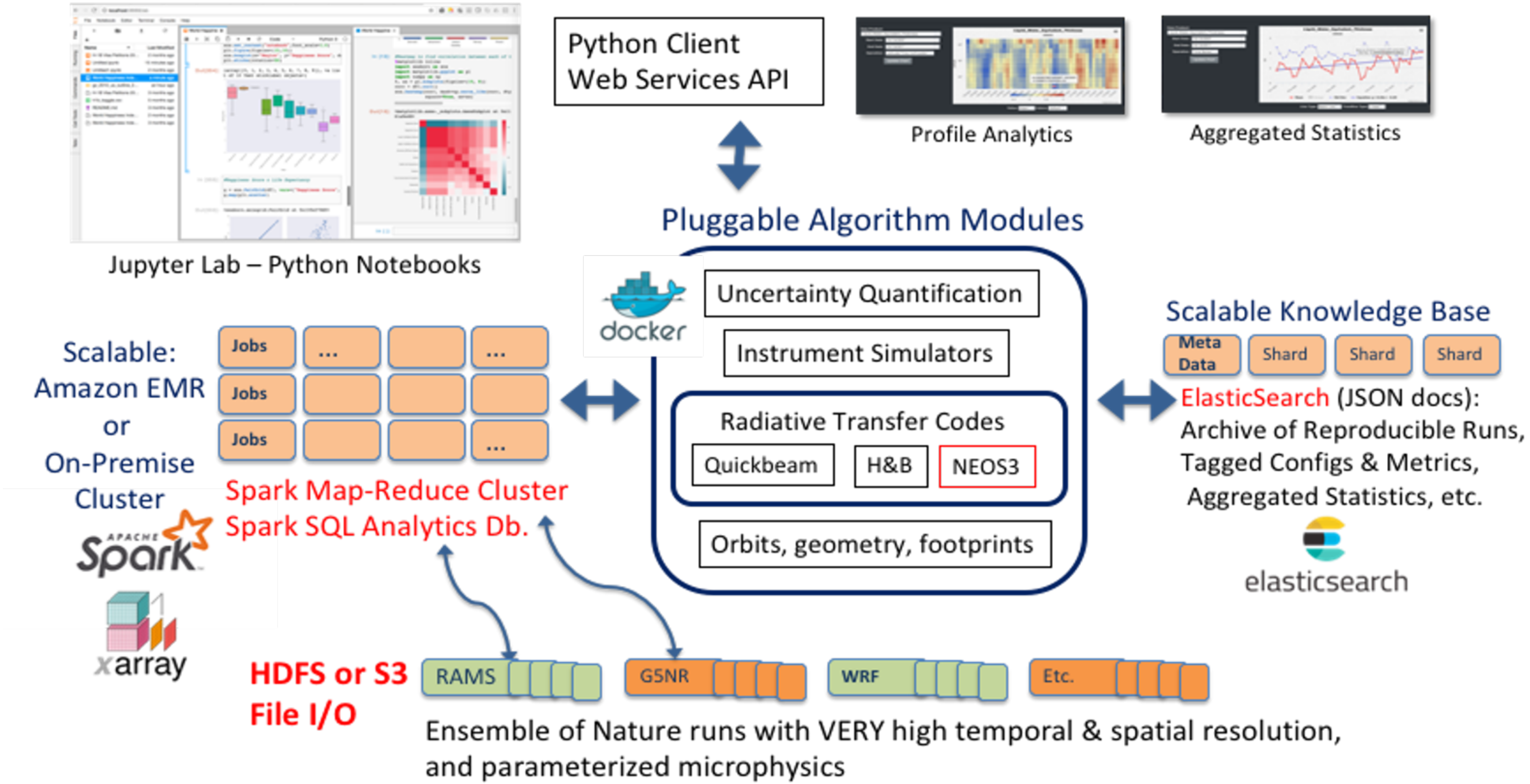
Integrated Toolkit in Python
End-to-end automated analysis
16 to 1024-way Ensemble Parallelism
"Quick look" Exploration
High-Fidelity Simulations
Parallel Analytics (or on GPU)
Explore larger Science Trade-Space

Applications:
- 2017 Decadal Survey-proposed CCP "required" observations
- Radar constellations
- All future instrument simulation & mission design at JPL

Schematic of the Mission Design Workflow



Architecture of the Parallel OSSE Framework



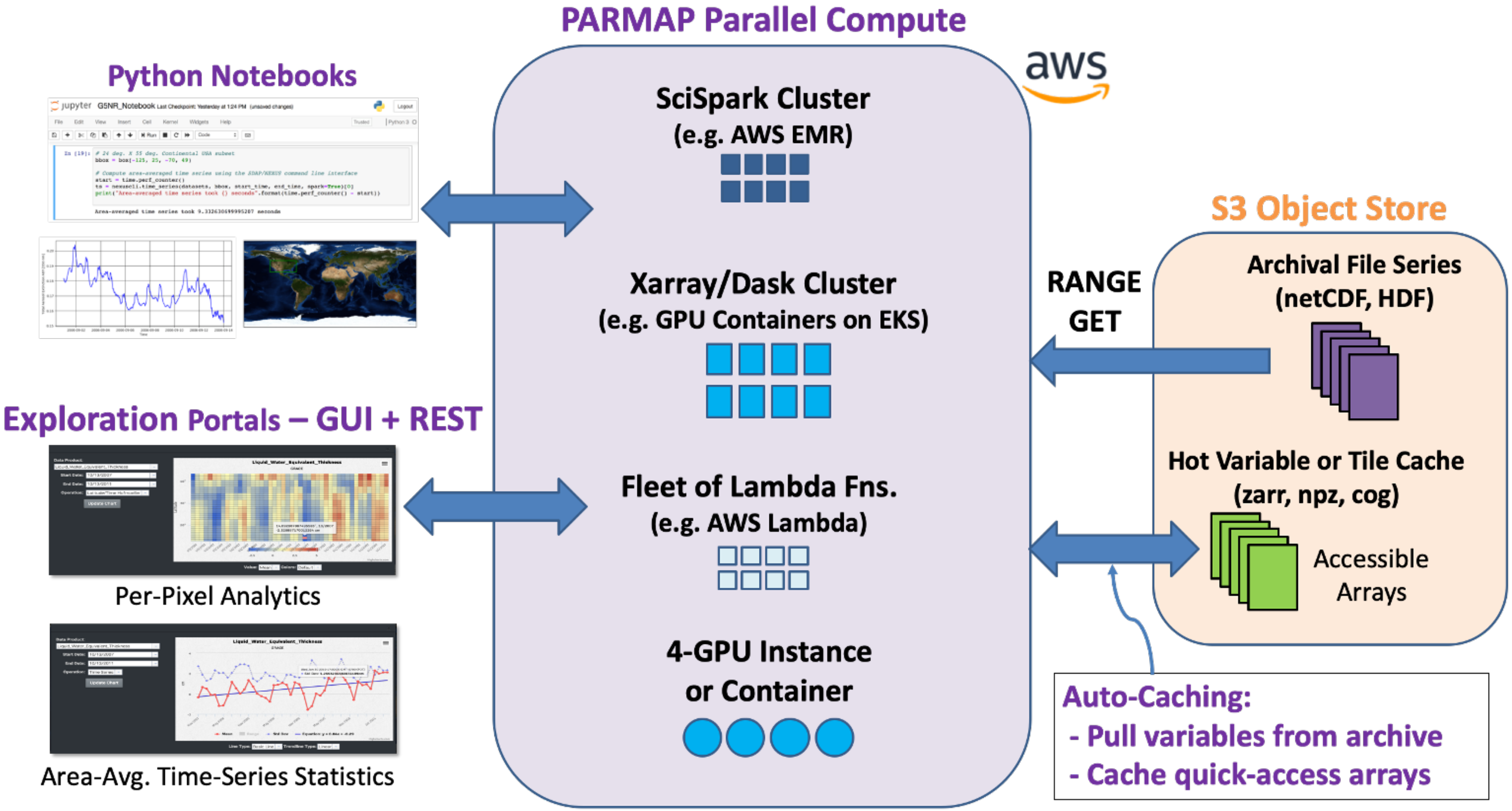
Applying PARMAP Python Library: Easy, Redeployable Parallel Computing for Everyone

- **PARMAP library (easy parallel computing in Python):**
  - Provides Map-Reduce operations over data, files or S3 URL's
  - Enables parallel analytics over time, spatial tiles, variables, model ensembles, multiple experiments, etc.
- **Deploy in multiple modes (without code changes):**
  - Multicore or GPU parallelism (single node)
  - PySparkling (pure Python library)
  - Full Apache PySpark Cluster
  - Xarray / Dask Cluster (dask.distributed scheduler)
  - AWS Lambda Functions (fleet of workers)
- **Choose type of Parallelism at deployment or run time**
  - Mode string = 'multicore', 'spark', 'dask, or 'lambda'
  - Credentials encapsulated in a 'config' dictionary
  - Combine parallelism patterns: Multicore & Distributed
  - Tailor scale of parallelism to data size & data locality

Simple coding for computing analytics in parallel over a file / tile / url series:

```
def metricFn(ncFile, variable):
    …
results =
  parmap(metricFn, urls,
            mode='lambda',
            numWorkers=32,
            config)
```

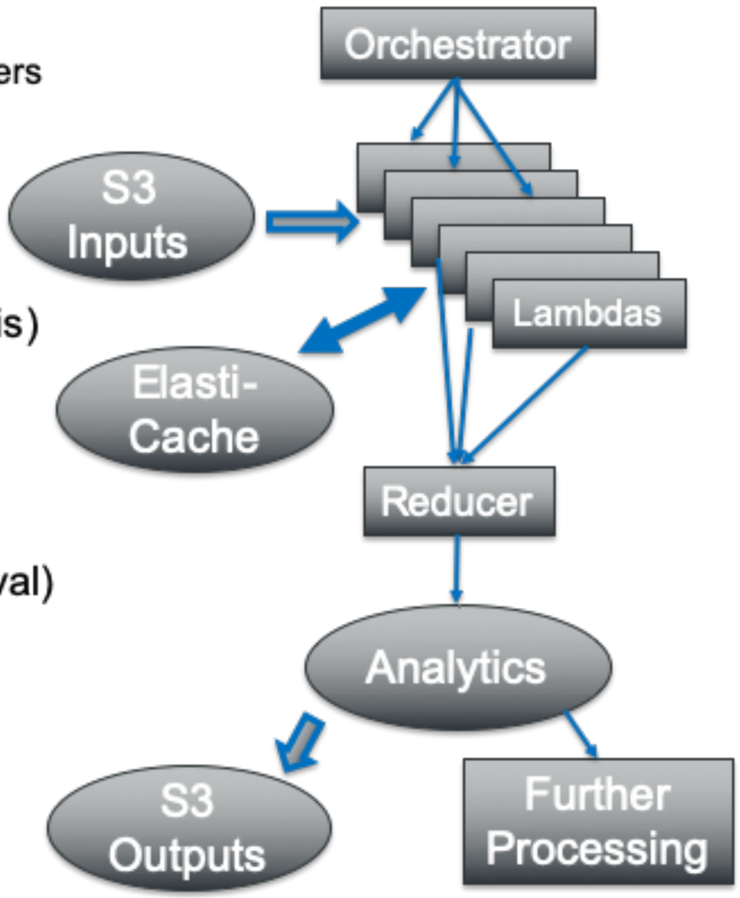
Notebook Analytics: PARMAP Cloud Architecture



PARMAP Serverless: Use Lambda Functions at AWS

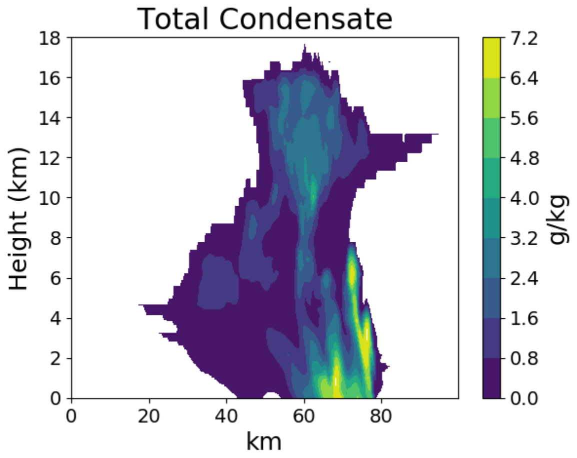
- **Cloud Paradigm:**
  - NetCDF / HDF files series reside in S3 (NASA pays for storage)
  - Hot Variables can be tiled & cached for even better performance
  - User pays compute costs for parallel analytics – Lambdas or Containers
  - Simple parallel coding using PARMAP library in Jupyter Notebooks
- **Serverless Backend (transparent to user):**
  - Parallel Analytics computed in fleet of Lambda Workers
  - Aggregation/reduce "in memory" using AWS ElastiCache (redis)
  - Only final results saved on disk in S3 (no disk bottleneck)
  - AWS Lambda functions bill every 1/10th of a second
- **Near-Zero Costs Except When Actually Computing**
  - User pays for Lambda Workers & transient ElastiCache (key/val)
  - No permanent Spark / Cassandra cluster needed
  - At NCCS, using on-demand Dask Clusters for compute
  - No permanent databases needed for most datasets (except time/space lookups for satellite swaths)

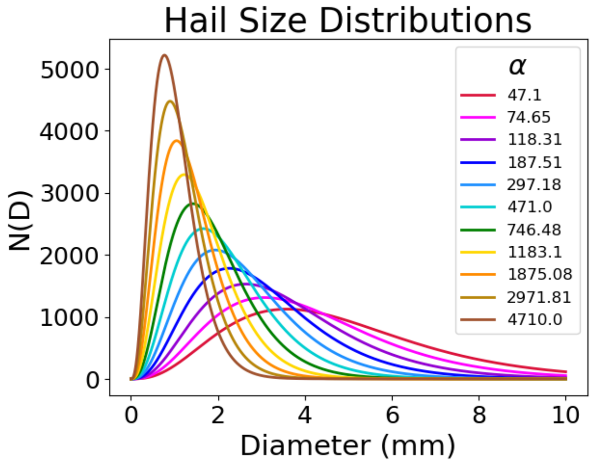
AWS Serverless Design



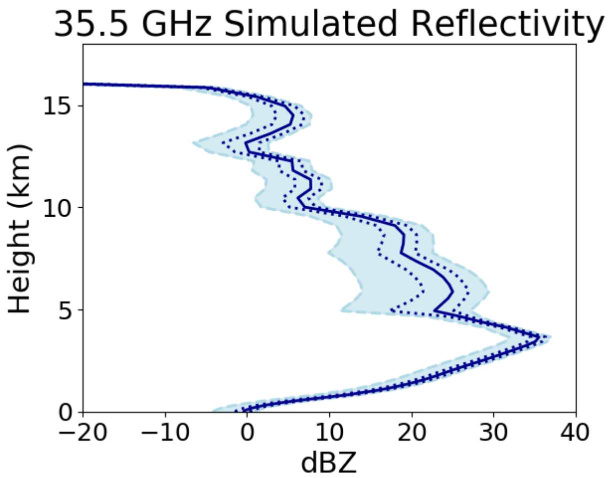
Example of Forward Model Results

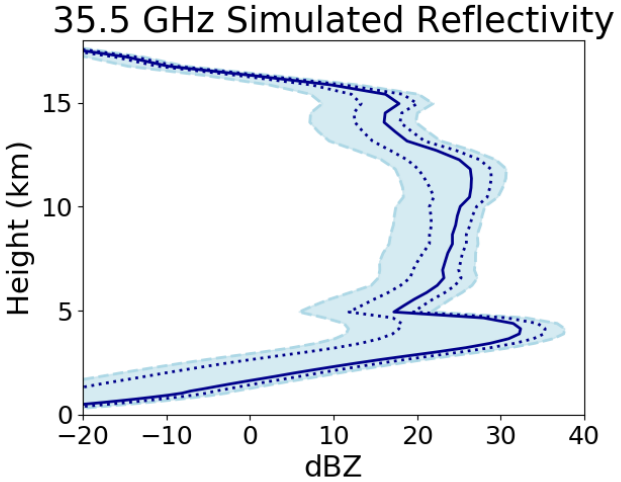
- Inputs:

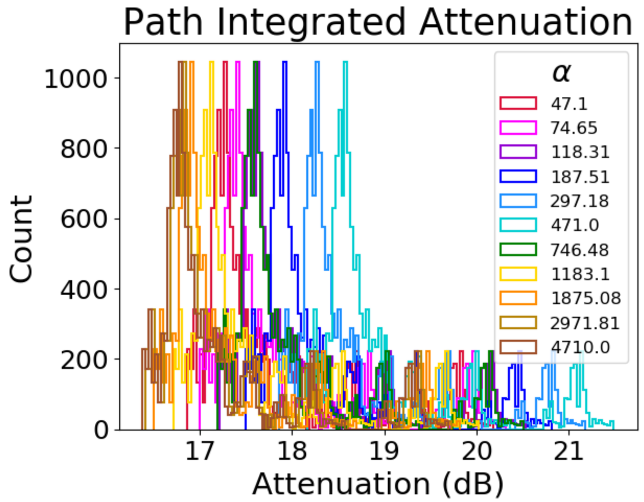




- Outputs:







Bayesian Optimal Estimation Retrieval

- Given a measurement **Y** we want to know the most likely state **X**

$$P(x|y) = P(y|x)\, P(x)$$

P(y|x) – Forward model **y = F(x)**, linearized as **y=Kx**

P(x) – Prior knowledge $\mathbf{x_a}$

each term has associated error matrix **S**

- Minimize cost function

$$(y - Kx)^T S_e^{-1} (y - Kx) + (x - x_a)^T S_a^{-1} (x - x_a)$$

Example Case:
Retrieve Cloud Water Path in shallow clouds

- Observation **y**
  - Radar reflectivity from Quickbeam forward model
  - Cloud optical depth
- State **x** to be retrieved
  - Profile of cloud water content
  - Constant value of droplet number in height
- Assumed prior knowledge $\mathbf{x}_a$
  - Constant profile of cloud mixing ratio 1 g/kg
  - Cloud droplet number concentration 300 /mg

Optimal Estimation Retrieval Validation

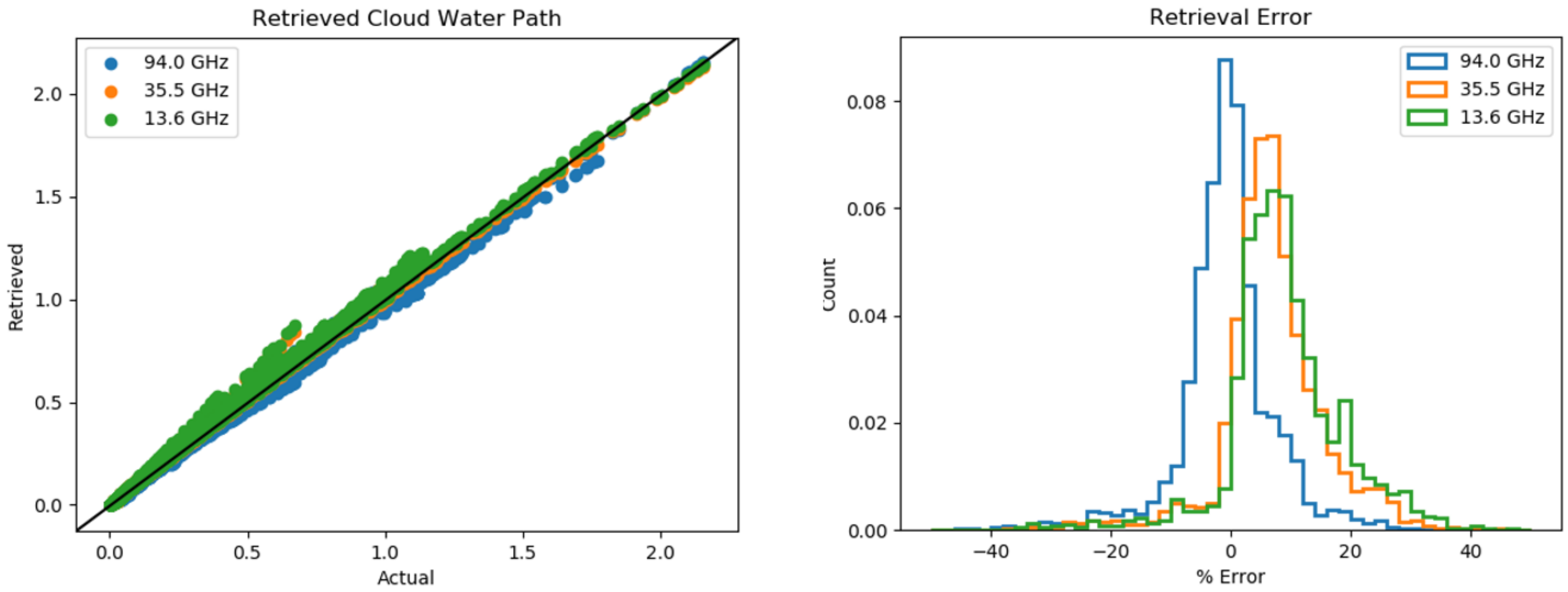


Figure 2. Optimal Estimation retrieval of Cloud Water Path for many profiles. (a) Scatter plot of retrieved vs. actual for three frequencies. (b) Histogram of percentage error for the same 3 frequencies.

Complete set of 100m+ OSSE experiments to be run in the won AIST-18 grant; cross-product of:

- 3 radar simulators
- 3 frequencies
- 75 RAMS model runs
- 15 time slices
- Particle size distributions
- 3 radar sensitivities
- 2 footprints
- Etc.

Phase 1:			
Quickbeam			
Radar Frequencies	3 (Ku, Ka, W)		
Parameters Varied	5 (ice hydrometeor PSDs)		
Parameter Values	5 (0.1 to 10 times the default)		
Model Simulations	75		
Time Slices	15 (5 in each of 3 life stages)		
Profiles per Time	200 (100 convective, 100 stratiform)		
Total	3*5^5*75*15*200 = 2109375000 (2.1e9)		
Per Profile	Total Time	With 48 Cores	With 1024 Cores
1s	24414 Days	509 Days	24 Days
H&B			
Radar Frequencies	3 (Ku, Ka, W)		
Parameters Varied	5 (ice hydrometeor PSDs)		
Parameter Values	2 (largest, smallest values)		
Scattering Choices	2 (single, multiple scattering)		
Hydrometeor Shapes	2 (spherical, non-spherical)		
Model Simulations	75		
Time Slices	15 (5 for each of 3 life stages)		
Profiles per Time	200 (100 convective, 100 stratiform)		
Total	3*2^5*2*2*75*15*200 = 86400000 (8.6e7)		
Per Profile	Total Time	With 48 Cores	With 1024 Cores
2s	2000 Days	42 Days	2 Days
Total Phase I	Total Time	With 48 Cores	With 1024 Cores
	26414 Days	551 Days	26 Days

Phase 2:			
'Observations' (NEOS)			
Radar Frequencies	3 (Ku, Ka, W)		
Model Simulations	75		
Time Slices	15 (5 in each of 3 life stages)		
Total	3*75*15 = 3375		
Per Slice	Total Time	With 48 Cores	With 1024 Cores
5 Hours	703 Days	15 Days	16.5 Hours
Footprints	5 (1-5km)		
Total	3375*5 = 16875		
Per Slice	Total Time	With 48 Cores	With 1024 Cores
2.5 Hour	1758 Days	37 Days	2 Days
Retrieval (Quickbeam or H&B)			
Radar Frequencies	7 (All combinations of 1, 2, or 3 frequencies)		
Radar Sensitivity	9 (3 values for each frequency)		
Footprints	6 (Native 250m, 1-5km)		
Model Simulations	75		
Time Slices	15 (5 for each of 3 life stages)		
Retrievals per Time	20 (10 convective, 10 stratiform)		
Total	7*9*6*75*15*20 = 8505000 (8.5e6)		
Per Retrieval	Total Time	With 48 Cores	With 1024 Cores
1 min	5906 Days	123 Days	6 Days
Total Phase II	Total Time	With 48 Cores	With 1024 Cores
	8367 Days	175 Days	9 Days

Table 1: Summary of all of the various configurations to be implemented in Phases 1 and 2, along with the expected computational time necessary to implement each phase. Compute times are based on proof-of-concept testing on the local JPL SciSpark cluster. Estimates of time necessary to run on 1024 cores is based on the assumption of linear scaling from 48 to 1024 cores.

Publications and References

[1] Wilson, B., R. Palamuttam, K. Whitehall, C. Mattmann, A. Goodman, M. Boustani, S. Shah, P. Zimdars, and Paul Ramirez, “SciSpark: Highly Interactive In-Memory Science Data Analytics”, peer-reviewed proceedings of the *IEEE Big Data Conference*, December 5, 2016.

[2] Haynes, J.M., R.T. Marchand, Z. Luo, A. Bodas-Salcedo, and G.L. Stephens, “A multi-purpose radar simulation package: Quick-Beam.”, *Bull. Amer. Meteor. Soc.*, 88, 2007, 1723-1727.

[3] Tanelli, S., et al., "Integrated instrument simulator suites for Earth science", *Proc. SPIE 8529, Remote Sensing and Modeling of the Atmosphere, Oceans, and Interactions IV*, 85290D (8 November 2012); doi: 10.1117/12.977577.

[4] Niamsuwan, N., S. Tanelli, M. P. Johnson, D. Dao, J. Jacob, S. Jaruwatanadilok, S. Oveisgharan, M. Simard, F. J. Turk, N. Majurec, and L. Tsang, "NASA Earth Observing System Simulator Suite (v 2.0)", *Earth Science and Technology Forum 2014*, Leesburg, VA, Oct 29, 2014.

[5] Rodgers, C. D., “Inverse Methods for Atmospheric Sounding”, *Theory and Practice, World Sci.*, Singapore, 2000.