National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

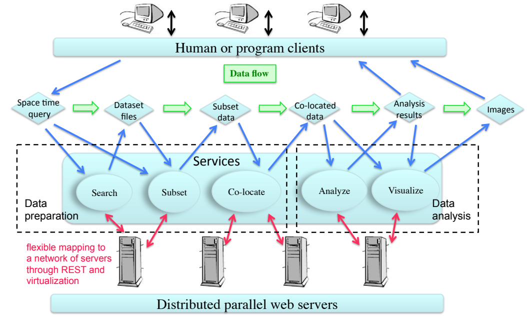
# Software infrastructure and algorithms to facilitate co-location of observation and model data

**Lei Pan*, Seungwon Lee, Gary Block, Robert Morris, Jui-lin Li, and Duane Waliser**
**Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91101**
*lei.pan@jpl.nasa.gov

## Background and Goal

- Observation data (e.g., from A-Train satellites) and model data (e.g., ECMWF) live in their own respective grid space, have different sampling characteristics, and use different formats and structures in archiving
- Scientists expect synergistic usage of these datasets, e.g., to calibrate one instrument using another, or to conduct model data assimilation using observation
- Co-location, which is to interpolate from a source data grid onto that of a target, provides a bridge for the above gap
- To built a software tool to facilitate co-location
- To make it easy to use and fast by leveraging technologies (e.g., web services, virtualization, and parallelization), and implementing efficient and scalable core algorithms

## Functional Architecture




## Web Services and Virtualization

- Reason for services: huge datasets from satellite and model in PBs
- Open source web services solutions rich and simple: Flask (Python microframework), Gunicorn (Python WSGI server), Tornado (scalable web server)
- Separate app. traffic (Gunicorn) from static HTTP traffic (Tornado)
- RESTful interfaces using ROA: URI to carry "scoping info." and HTTP to convey "method info."; stateless for parallel deployment
- Virtualization using VMWare: service replication made easy

## Parallelization

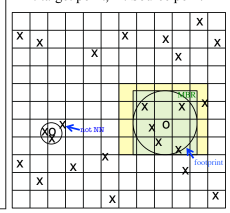
- Co-location of multiple data granules inherently data parallel
- Python Multiprocessing for multicore:
  - Multicore machines affordable and available
  - Shared memory programming relatively easy
  - Effective memory recycling when child processes join
  - Limited scalability
- Parallel Python for cluster (under construction):
  - Scalable to 100s of cores and beyond
  - Remote data shipping either via parallel file system (e.g., PVFS) or using pre-fetching

## Core Algorithm: Nearest Neighbor Search

Our algorithm has a linear complexity of O(n+m), where n is the number of target points and m the number of source points

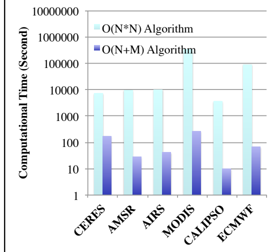
```
1.  Overlay k by k grid on target points
    where k=sqrt(n) (O(k))
2.  Register each target point to a grid
    cell (O(n))
3.  For each source point S (O(m))
   1.find minimum bounding rectangle
     (MBR) of its footprint
   2.Find grid cells that cover MBR
   3.For each grid cell G (O(C))
       • For each target point T in G
          • store S to T's candidate list
            (sorted by distance=|T-S|)
4.  For each target point T (O(n))
   1.Compute average among the first L
     in T's candidate list
```

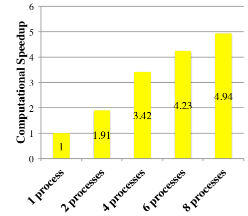
X: target point, O: Source point




- ✧ Nearest neighbor not necessary in same grid cell
- ✧ C: a constant bounded by max "density" of target pts in MBR, which is not proportional to n
- ✧ When L==1, nearest neighbor; when L>1, average of L nearest neighbors

## Co-location Performance



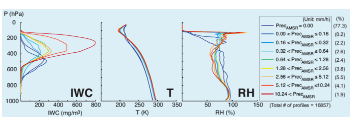




- nearest neighbor search algorithm improves computational performance by 100-1000 times over the naive O(N*N) algorithm
- parallel co-location implementation has a speedup of 5 on 8 cores

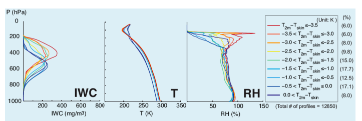
## Applying to Science

Precipitating Deep Convection: A-Train vs. YOTC-ECMWF model forecast

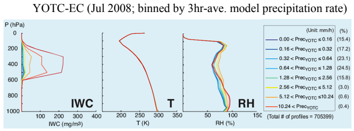
CloudSat + colocated ECMWF (Jul 2007; binned by AMSR-E precip. rate)



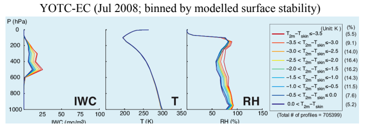

CloudSat + colocated ECMWF (Jul 2007; binned by EC surface stability)




YOTC-EC (Jul 2008; binned by 3hr-ave. model precipitation rate)




YOTC-EC (Jul 2008; binned by modelled surface stability)




A comprehensive A-Train-ECMWF co-located data set can be used to characterize deep convection cloud structures, properties, and its environmental context. Reasonable agreements between observations and YOTC-EC model forecast are found.

## Conclusions and Future Work

- Web services efficient: large data is where heavy number crunching is, along with powerful CPUs; small result is downloaded by clients; cost of coarse communication overshadowed by gain; light-weight clients both in terms of processing power and number of software packages
- Web services easy to use: similar to function calls that scientists are familiar with, using a browser or a python program, with samples provided
- Efficient algorithm & data structure and parallelization both important for performance
- Demonstrated value of the co-located datasets to scientific research
- ATrain/ECMWF co-located data products released to public in 2011
- Future work: use Parallel Python for scalable distributed computing
- Future work: build data analysis web services (funded by NASA ROSES 2011)

***Acknowledgements:*** **This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with NASA.**

**Presented at AGU Fall Meeting, San Francisco, Dec 3-7, 2012**