

# Treating gridded geospatial data as point data to simplify analytics

Christine Smit<sup>1,3</sup>, Hailiang Zhang<sup>2,3</sup>, Mahabaleshwara Hegde<sup>2,3</sup>,  
Faith Giguere<sup>2,3</sup>, Long Pham<sup>3</sup>

(1) Telophase Corporation

(2) Adnet Systems, Inc.

(3) NASA

# Background: NASA GES DISC

Goddard Earth Sciences (GES) Data and Information Services Center (DISC)

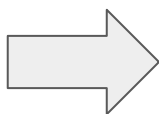
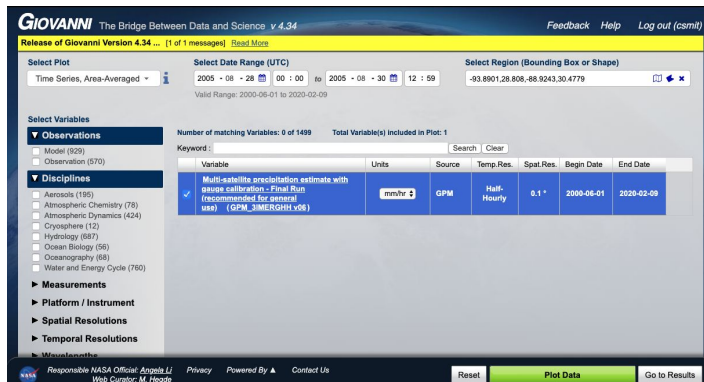
- Archives datasets
  - atmospheric composition
  - water and energy cycles
  - climate variability
- Provides data services

# Background: Giovanni

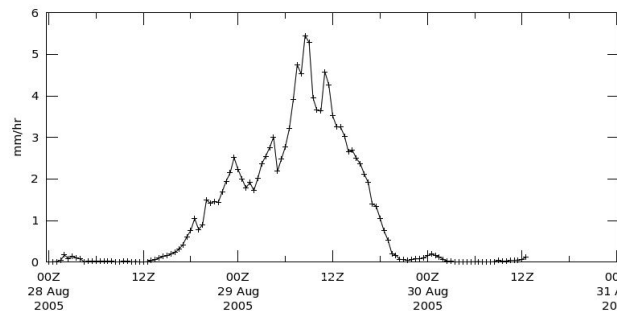
Geospatial Interactive Online Visualization ANd aNalysis Infrastructure (Giovanni)

- Simplify data → science
- Visualizes data in web browser

# Background: Giovanni

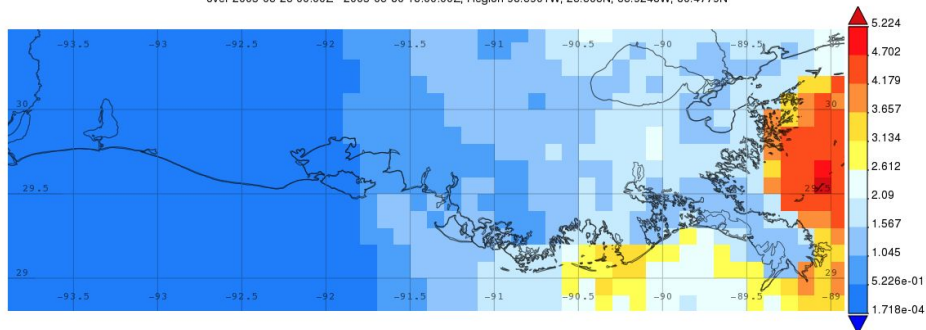


Time Series, Area-Averaged of Multi-satellite precipitation estimate with gauge calibration - Final Run (recommended for general use) half-hourly 0.1 deg. [GPM GPM\_3IMERGHH v06] mm/hr over 2005-08-28 00:00Z - 2005-08-30 13:00:00Z, Region 93.8901W, 28.808N, 88.9243W, 30.4779N



- The user-selected region was defined by 93.8901W, 28.808N, 88.9243W, 30.4779N. The data grid also limits the analyzable region to the following bounding points: 93.85W, 28.85N, 88.95W, 30.45N. This analyzable region indicates the spatial limits of the subsetted granules that went into making this visualization result.

Time Averaged Map of Multi-satellite precipitation estimate with gauge calibration - Final Run (recommended for general use) half-hourly 0.1 deg. [GPM GPM\_3IMERGHH v06] mm/hr over 2005-08-28 00:00Z - 2005-08-30 13:00:00Z, Region 93.8901W, 28.808N, 88.9243W, 30.4779N



- Selected date range was 2005-08-28 00:00Z - 2005-08-30 12:59Z. Title reflects the date range of the granules that went into making this result.

- 22 summary plots
- > 1000 physical parameters
- > 50 kinds of measurements
- > 20 measurement platforms and instruments

# Background: moving to the cloud

## On premises

- Technology > 10 years old
- Single server

Background: moving to the cloud

Question #1: Data format?

# Sidebar: data in the cloud

1. Large datasets → object storage
2. Object storage →
  - a. Libraries must support object API.
  - b. Data formats must be subsettable in place.

# Background: Traditional Data Formats HDF5 and NetCDF4

The screenshot shows the Panoply application window titled "Panoply — Sources". The interface includes a toolbar with "Create Plot", "Combine Plot", and "Open Dataset" buttons. Below the toolbar are tabs for "Datasets", "Catalogs", and "Bookmarks". A table lists datasets with columns for Name, Long Name, and Type. The selected dataset is "GPM\_3IMERGHH\_06\_precipitationCal", which is a "Geo2D" type. The table lists variables: lat (1D), lat\_bnds (2D), lon (1D), lon\_bnds (2D), time (—), and time\_bnds (1D). To the right of the table, the variable "GPM\_3IMERGHH\_06\_precipitationCal" is selected, and its metadata is displayed in a text box. The metadata includes the file path "scrubbed.GPM\_3IMERGHH\_06\_precipitationCal.20050828000000.nc" and a list of attributes such as DimensionNames, Units, FillValue, CodeMissingValue, origname, fullnamepath, standard\_name, quantity\_type, product\_short\_name, product\_version, long\_name, coordinates, units, and ChunkSizes.

Name	Long Name	Type
scrubbed.GPM_3IMERGHH_06_pr...	scrubbed.GPM_3IMERGHH_06_precipitatio...	Local File
GPM_3IMERGHH_06_precipitat...	Multi-satellite precipitation estimate with g...	Geo2D
lat	latitude	1D
lat_bnds	lat bnds	2D
lon	longitude	1D
lon_bnds	lon bnds	2D
time	time	—
time_bnds	time bnds	1D

**Variable "GPM\_3IMERGHH\_06\_precipitationCal"**

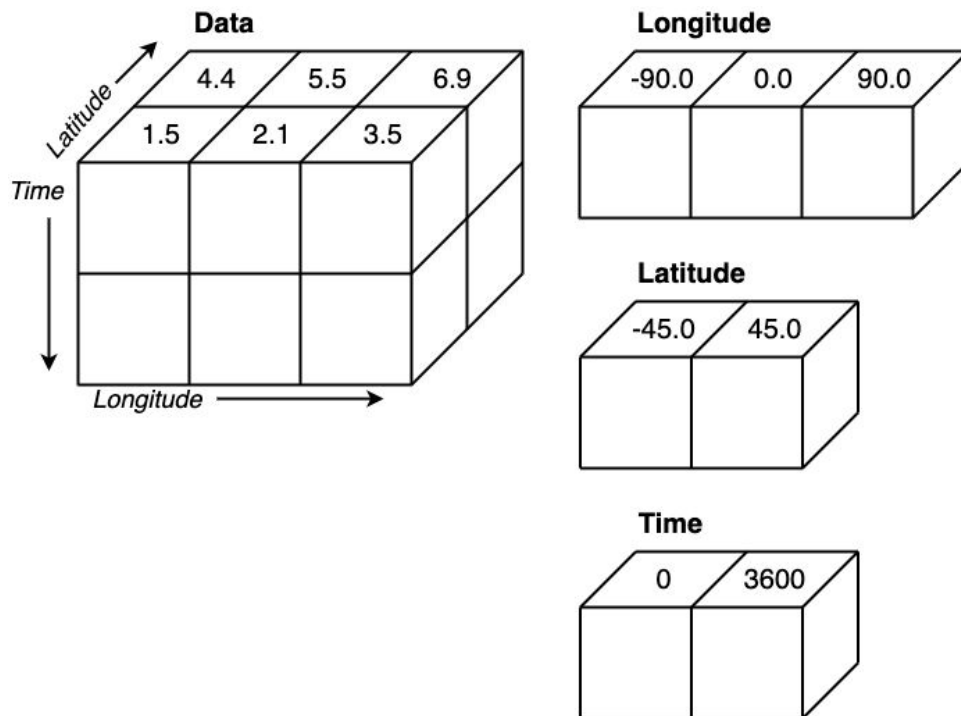
In file "scrubbed.GPM\_3IMERGHH\_06\_precipitationCal.20050828000000.nc"

```
float GPM_3IMERGHH_06_precipitationCal(time=1, lat=1800, lon=3600);
:DimensionNames = "time,lon,lat";
:Units = "mm/hr";
:_FillValue = -9999.9f; // float
:CodeMissingValue = "-9999.9";
:origname = "precipitationCal";
:fullnamepath = "/Grid/precipitationCal";
:standard_name = "precipitationcal";
:quantity_type = "Precipitation";
:product_short_name = "GPM_3IMERGHH";
:product_version = "06";
:long_name = "Multi-satellite precipitation estimate with gauge calibration - Final Run (recommended for general use)";
:coordinates = "time lat lon";
:units = "mm/hr";
:_ChunkSizes = 1U, 600U, 1200U; // uint
```

Show: All variables



# Background: Traditional Data Formats HDF5 and NetCDF4



# Background: HDF5 and NetCDF4

Not (currently) cloud friendly \*

- Libraries don't support object storage
- Data can't be subset

\* HDF5 has a cloud effort called HSDS, but this requires a data server cluster to be up and running any time you want to access data.

# Cloud-friendly data format: Apache Parquet

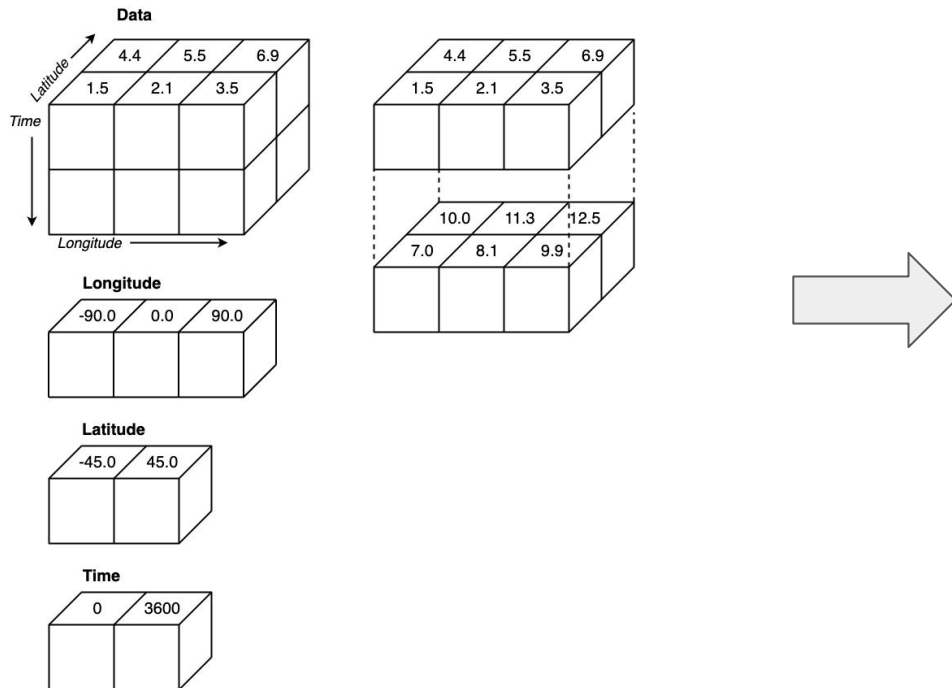
- Columnar data format
- Released 2013
- Part of the Hadoop ecosystem

# Cloud-friendly data format: Apache Parquet

- Simple data model
- Erases distinctions between different levels and formats of data

# Cloud-friendly data format: Apache Parquet

Multi-dimensional array with dimension variables



Point data (Data frame)

Longitude	Latitude	Time	Data
-90.0	-45.0	0	1.5
0.0	-45.0	0	2.1
90.0	-45.0	0	3.5
-90.0	45.0	0	4.4
0.0	45.0	0	5.5
90.0	45.0	0	6.9
-90.0	-45.0	3600	7.0
0.0	-45.0	3600	8.1
90.0	-45.0	3600	9.9
-90.0	45.0	3600	10.0
0.0	45.0	3600	11.3
90.0	45.0	3600	12.5

# Example results

Time series with a global average for each day:

- 10.5 years
- Daily,  $0.1^\circ$  data
- ~ 25 billion points

On premises Giovanni:

- ~16500 seconds/4.6 hours
- ~4000 lines of analytics code

# Example results: Dask\* + Apache Parquet

- **3x faster:** ~4.6 → ~1.4 hours
- **50x less code:** ~4000 → ~75 lines of code

\* Dask

- scalable analytics
- cluster or single thread

# Dask + Apache Parquet: very simple code

```
def area_average(df, bbox, time_range):  
    """  
    Builds a dataframe with the area average over the bounding box at each time  
    in the time range.  
  
    df: dask dataframe  
    bbox: bounding box string in "west,south,east,north" format.  
    time_range: time in "YYYY-MM-DDThh:mm:ssZ/YYYY-MM-DDThh:mm:ssZ" format  
    """  
    # parse the bounding box  
    west, south, east, north = [float(v) for v in bbox.split(",")]  
  
    # parse the time range  
    start_time, end_time = [string_to_timestamp(s) for s in time_range.split("/")]  
  
    # form the subset and get rid of any NaN values  
    subset = df[(df.lat >= south) & \  
                (df.lat <= north) & \  
                (df.lon >= west) & \  
                (df.lon <= east) & \  
                (df.time >= start_time) & \  
                (df.time <= end_time)].dropna()  
  
    # calculate the cos latitude weight  
    subset['weights'] = dask.array.cos(dask.array.multiply(np.pi/180, subset['lat']))  
  
    # multiply the weights by the data values  
    subset['weighted_data'] = dask.array.multiply(subset.variable, subset.weights)  
  
    # group by the time, which groups all the data points from the same time together  
    grouped = subset.groupby(by="time")  
  
    # calculate the average  
    avg = dask.array.divide(grouped.weighted_data.sum(), grouped.weights.sum()).to_frame().reset_index()  
  
    # rename the 'weighted_data' column to 'area average'  
    return avg.rename(columns={'weighted_data': 'area_average'})
```



Example results: AWS Athena + Apache Parquet  
(+ 3-dimensional accumulation)

## Sidebar: Accumulation to speed up averages

	0	1	2	3	4	5	6	7	8	9
$\vec{X}$	2.3	5.2	8.3	3.0	7.0	1.6	2.2	2.3	6.4	2.3

## Sidebar: Accumulation to speed up averages

	0	1	2	3	4	5	6	7	8	9
$\vec{X}$	2.3	5.2	8.3	3.0	7.0	1.6	2.2	2.3	6.4	2.3

$$\text{avg}(5, 8) = \frac{1}{4} \sum_{i=5}^8 x_i$$

Sidebar: Accumulation to speed up averages

$$acc(\vec{X})_i = \sum_{j=0}^i x_j$$

## Sidebar: Accumulation to speed up averages

	0	1	2	3	4	5	6	7	8	9
$\vec{X}$	2.3	5.2	8.3	3.0	7.0	1.6	2.2	2.3	6.4	2.3
$acc(\vec{X})$	2.3	7.5	15.8	18.8	25.8	27.4	29.6	31.9	38.3	40.6

## Sidebar: Accumulation to speed up averages

	0	1	2	3	4	5	6	7	8	9
$\vec{X}$	2.3	5.2	8.3	3.0	7.0	1.6	2.2	2.3	6.4	2.3
$acc(\vec{X})$	2.3	7.5	15.8	18.8	25.8	27.4	29.6	31.9	38.3	40.6

$avg(5, 8) = \frac{1}{4} \left( acc(\vec{X})_8 - acc(\vec{X})_4 \right)$

## Sidebar: Accumulation to speed up averages

	0	1	2	3	4	5	6	7	8	9
$\vec{X}$	2.3	5.2	8.3	3.0	7.0	1.6	2.2	2.3	6.4	2.3
$acc(\vec{X})$	2.3	7.5	15.8	18.8	25.8	27.4	29.6	31.9	38.3	40.6

$avg(5, 8) = \frac{1}{4} \left( acc(\vec{X})_8 - acc(\vec{X})_4 \right)$

O(1) algorithm!

## Sidebar: Accumulation to speed up averages

n-dimension accumulation  $\rightarrow 2^n$  points  $\rightarrow$  still  **$O(1)$**  !



# Example results: AWS Athena + Apache Parquet (+ 3-dimensional accumulation)

- **3000x faster:** ~4.6 hours → ~5 seconds\*
- **10x less code:** ~4000 → ~300 lines of code

\* Reported in more detail at the American Geophysical Union (AGU) Fall 2019 Meeting. Abstract:  
<https://agu.confex.com/agu/fm19/meetingapp.cgi/Paper/596820>, Poster:  
<https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20200000534.pdf>

# Results

## Dask + Apache Parquet

- Code complexity ↓↓↓
- Speed ↑ (*sometimes*)

## AWS Athena + Apache Parquet + accumulation

- Code complexity ↓↓
- Speed ↑↑↑

# Results

Why not Dask + Apache Parquet + accumulation?

- Poor subsetting\*

Why not AWS Athena + Apache Parquet?

- Cost

\* Data frame divisions + calculating offsets is fast, but we then can't use main dask API.

# Conclusions

- Apache Parquet + high level specification = effortless parallelization
- Data frames = simple code

# Looking forward ... Zarr?

- + Cloud-native
  - + Has multi-dimensional arrays
  - + Supports metadata (w/ xarray)
- 
- Library support

Thanks!

NASA GES DISC