



National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California



**National Aeronautics and
Space Administration**

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Workshop: Using Jupyter for Cloud-based Analysis

Presented By: Frank Greguska (JPL) and Joe Jacob (JPL)

Supported By: Sean Gordon (HDF Group) and Keith Maull (UCAR)

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099, U.S.A.



Outline

- Introductions
- Brief Introduction to Jupyter
- Server Assignment
- Data Analysis with Apache Science Data Analytics Platform (SDAP)



Introduction to Jupyter

- This workshop will use an instance of JupyterHub hosted by ESIP





Server Assignment

- Setup your Jupyter Notebook hosted by ESIP
 - Sign in to esiphub.ndslabs.org
- You will also be connecting to a cloud service running on AWS
 - Navigate to the URL below and put your name next to an instance
 - This will be the instance you will be using for the workshop

<https://is.gd/bawexu>





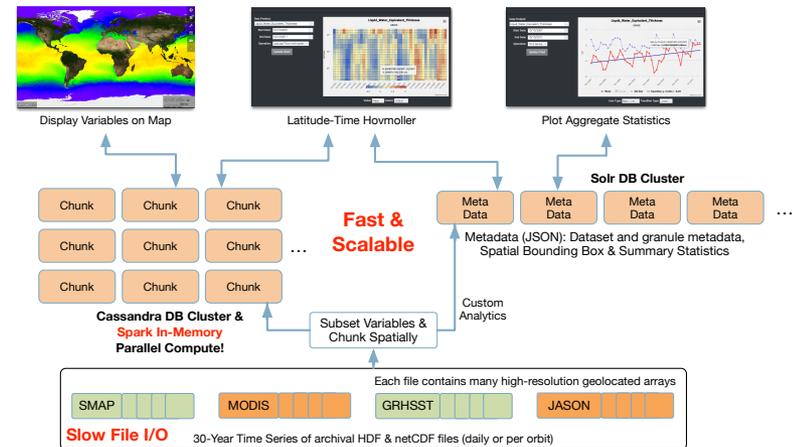
Apache Science Data Analytics Platform (SDAP)

- **OceanWorks** is to establish an **Integrated Data Analytics Center** at the NASA Physical Oceanography Distributed Active Archive Center (PO.DAAC) for Big Ocean Science
- Focuses on technology integration, advancement and maturity
- Collaboration between JPL, Center for Atmospheric Prediction Studies (COAPS) at Florida State University (FSU), National Center for Atmospheric Research (NCAR), and George Mason University (GMU)
- Bringing together PO.DAAC-related big data technologies
 - Big data analytic platform
 - Anomaly detection and ocean science
 - Distributed in situ to satellite matchup
 - Dynamic datasets ranking and recommendations
 - Sub-second data search solution and metadata translation and services aggregation
 - Quality-screened data subsetting
- All code open-sourced as Apache Science Data Analytics Platform (SDAP)



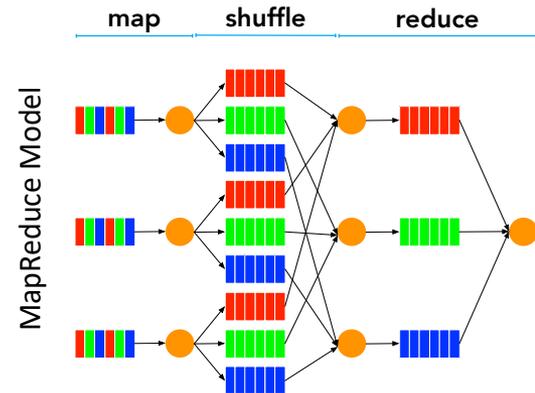
SDAP Cloud Analytics: NEXUS

- **NEXUS** is a data-intensive analysis solution using a new approach for handling science data to enable large-scale data analysis
 - Streaming architecture for horizontal scale data ingestion
 - Scales horizontally to handle massive amount of data in parallel
 - Provides high-performance geospatial and indexed search solution
 - Provides tiled data storage architecture to eliminate file I/O overhead
 - A growing collection of science analysis webservices



NEXUS' Two-Database Architecture

- **MapReduce**: A programming model for expressing distributed computations on massive amount of data and an execution framework for large-scale data processing on clusters of commodity servers. - J. Lin and C. Dyer, "Data-Intensive Text Processing with MapReduce"
 - **Map**: splits processing across cluster of machines in parallel, each is responsible for a record of data
 - **Reduce**: combines the results from Map processes





Jupyter Integration

- Python 3 module for easy integration
 - Source code: <https://github.com/apache/incubator-sdap-nexus/tree/master/client>
 - API Documentation: <https://htmlpreview.github.io/?https://github.com/apache/incubator-sdap-nexus/blob/master/client/docs/nexuscli/nexuscli.m.html>
- Exposes HTTP endpoints as functions
- Marshalls function input to JSON
- Unmarshalls server response to objects

```
def time_series(datasets, bounding_box, start_datetime, end_datetime,  
               spark=False)
```

Send a request to NEXUS to calculate a time series.

datasets Sequence (max length 2) of the name of the dataset(s)
bounding_box Bounding box for area of interest as a `shapely.geometry.polygon.Polygon`
start_datetime Start time as a `datetime.datetime`
end_datetime End time as a `datetime.datetime`
spark Optionally use spark. Default: `False`
return List of `TimeSeries` namedtuples

[SHOW SOURCE ▾](#)

```
def daily_difference_average(dataset, bounding_box, start_datetime,  
                             end_datetime)
```

Generate an anomaly Time series for a given dataset, bounding box, and timeframe.

dataset Name of the dataset as a String
bounding_box Bounding box for area of interest as a `shapely.geometry.polygon.Polygon`
start_datetime Start time as a `datetime.datetime`
end_datetime End time as a `datetime.datetime`
return List of `TimeSeries` namedtuples



Analysis on the Cloud

- SDAP NEXUS makes it easy to do scientific analysis across many types of datasets
- Common algorithms are available

The screenshot shows a Jupyter Notebook titled "NEXUS Time Series Example" running on a secure Jupyter server. The code defines a URL for the NEXUS service, sends a GET request, and processes the JSON response to extract dates and mean SST values. A plot at the bottom shows the resulting time series data from 2008 to 2015, with temperature on the y-axis and time on the x-axis. The plot shows a clear seasonal cycle with peaks around 14 and troughs around 6.

```
# Request NEXUS to compute SST Time Series 2008/9/1 - 2015/10/1
# for the "blob" warming off Western Canada and plot the means
...
ds='AVHRR_OI_L4_GHRSSST_NCEI'

url = ... # construct the webservice URL request

# make request to NEXUS using URL request
# save JSON response in local variable
ts = json.loads(str(requests.get(url).text))

# extract dates and means from the response
means = []
dates = []
for data in ts['data']:
    means.append(data[0]['mean'])
    d = datetime.datetime.fromtimestamp((data[0]['time']))
    dates.append(d)

# plot the result
...
```

https://oceanworks.jpl.nasa.gov/timeSeriesSpark?spark=mesos,16,32&ds=AVHRR_OI_L4_GHRSSST_NCEI&minLat=45&minLon=-150&maxLat=60&maxLon=-120&startTime=2008-09-01T00:00:00Z&endTime=2015-10-01T23:59:59Z

It took: 2.0984323024749756 sec

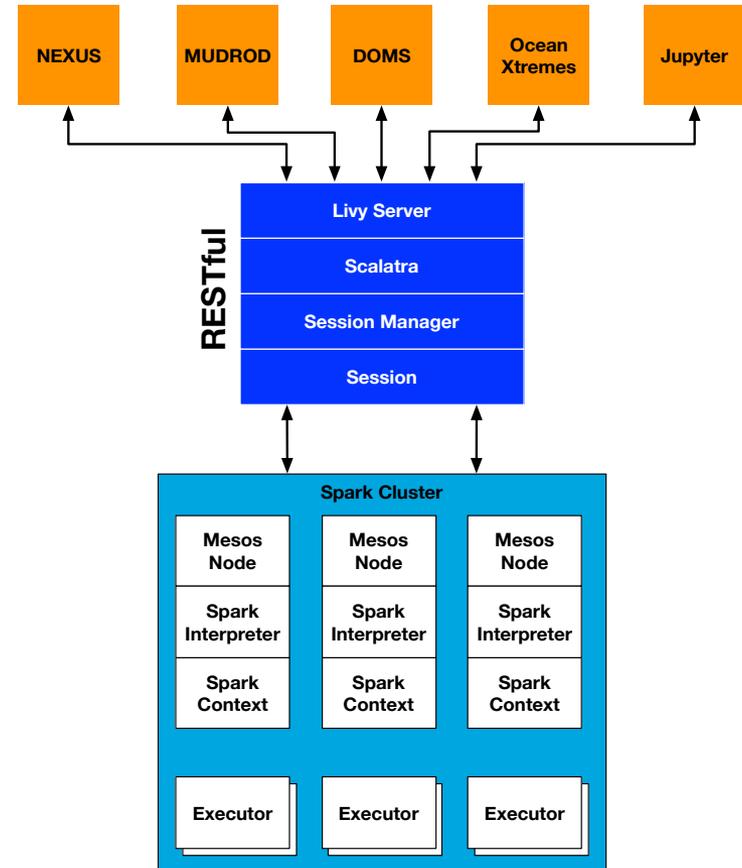


Session 1

Let's Get Started!

Under Development

- Developed independently, all the major services in OceanWorks require Apache Spark cluster
- If OceanWorks simply deploy these services to Amazon, it will require dedicated Apache Spark cluster for each
- Too many cluster and very costly, since Apache Spark recommends high memory machine instances
- Looking at the Amazon's EMR model. It is designed to be a job execution solution, and the jobs could from different applications
- Apache Livy provides a RESTful interface to Apache Spark cluster. It is a drop-in service to enable applications to interact with Spark cluster using RESTful api.
- The Apache Livy API also allows users to submit ad hoc map and reduce logics to be handled by the Spark cluster
- Through Apache Livy, scientists could use Jupyter environment to design their analytic algorithms that will be executed in the OceanWorks' Spark Cluster





Under Development

- Provide scientist a platform to develop algorithms to execute in OceanWorks' Spark cluster
- A new OceanWorks' RESTful service to offer flexible environment for researchers to experiment with their algorithms and our data, without having to deal with the complexity of Cloud and job management

The screenshot shows a JupyterLab notebook titled 'SDAP Livy' in a browser window. The notebook contains Python code that defines a function to create a Spark driver and execute a time series analysis. The code includes imports for textwrap, webservice, shapely, and numpy, and defines a bounding polygon and a daysinrange. It then uses spark_driver to execute the code and prints the results. The output shows the creation of a Spark session and the submission of code, followed by a large JSON-like output structure.

```
def time_series(ds, min_lon, max_lon, min_lat, max_lat, start_time, end_time,
               spark_nparts, lh):
    code = textwrap.dedent("""
    from webservice.algorithms_spark.TimeSeriesSpark import spark_driver
    from shapely.geometry import Polygon
    import numpy as np
    shortName = "{}"
    bounding_polygon = Polygon(((3, 1), (4, 1), (4, 2), (3, 2), (3, 1)))
    daysinrange=np.arange(5, {6}+1)
    spark_nparts_needed = {7}
    results, meta = spark_driver(daysinrange, bounding_polygon, shortName,
                                spark_nparts_needed=spark_nparts_needed,
                                sc=sc)

    print results
        """.format(ds, min_lat, max_lat, min_lon, max_lon,
                    start_time, end_time,
                    spark_nparts))

    ans = lh.run_code(code)
    pprint.pprint(ans)

# Create a SDAP handler.
lh = SDAPHandler()

# Run a NEXUS time series.
ds = 'AVHRR_OI_L4_GHRSSST_NCEI'
min_lon = -150
max_lon = -120
min_lat = 45
max_lat = 60
start_time = 1220227200
end_time = 1221523199
spark_nparts = 16
time_series(ds, min_lon, max_lon, min_lat, max_lat, start_time, end_time,
            spark_nparts, lh)

# Close the SDAP handler
lh.close()

Creating Spark session...
Submitting code...
Running code...
{'text/plain': [{"std": 1.0067574523434835, 'cnt': 4841, 'min': 9.6900024, "
"max": 15.940002, 'time': 1220227200, 'mean': "
"12.469548225402832}, {'std': 1.1194929679587775, 'cnt': 4841, "
"min": 9.7200012, 'max': 15.899994, 'time': 1220313600, "
"mean": 12.436386108398438}, {'std': 1.2530827115264895, "
"cnt": 4841, 'min': 9.6900024, 'max': 16.019989, 'time': "
"1220400000, 'mean': 12.5492582321167}, {'std': "
"1.2141109231112326, 'cnt': 4841, 'min': 9.3399963, 'max': "
"15.73999, 'time': 1220486400, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1220572800, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1220659200, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1220745600, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1220832000, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1220918400, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221004800, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221091200, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221177600, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221264000, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221350400, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221436800, 'mean': 12.61837100982666}, "
"std": 1.286464696098635, 'cnt': 4841, 'min': 9.3299866, "
"max": 15.73999, 'time': 1221523200, 'mean': 12.61837100982666}]}

```

Webmaster: Thomas Huang



Wrap Up

- This year's workshop materials are available online
 - <https://github.com/ESIPFed/Using-Jupyter-for-Cloud-based-Analysis>
- Documentation for NEXUS can be found online
 - <http://incubator-sdap-nexus.readthedocs.io/en/latest/index.html>
- Apache Science Data Analytics Platform (SDAP) website
 - <http://sdap.apache.org/>

Thank You!