Jet Propulsion Laboratory
California Institute of Technology

SciSpark 301

Build Your Own Climate Metrics

Agenda for 301:

- Access your SciSpark & Notebook VM (personal sandbox)
- Quick recap. of SciSpark Project
- PDF Clustering Use Case
  - K-means Clustering of atmospheric behavior using a full probability density function (PDF) or just moments: histogram versus (std dev, skewness)
- SciSpark Challenges & Lessons Learned
- Tutorial Notebooks:
  - 301-1, 2, 3: PDF Clustering in PySpark, in Scala, Side by Side
  - 301-4: Regional Climate Model Evaluation (RCMES) - (optional)
  - 301-5: CMDA Workflow Example
  - 301-6: Lessons Learned Notebook
- Wrap-up Discussion: How do you do X in SciSpark?
  - Bring your Use Cases and questions

Claim your SciSpark VM

- Go to: http://is.gd/scispark
  - You'll have to click through!
- Enter your name next to an IP address to claim it
- Enter the URL in your browser to access the SciSpark Notebook
  - Chrome or Firefox preferred

SciSpark AIST Project

AIST-14-0034 Funded by Mike Little

PI: Christian Mattmann
Project Mgr: Brian Wilson
JPL Team: Kim Whitehall, Sujen Shah, Maziyar Boustani, Alex Goodman, Wayne Burke, Gerald Manipon, Paul Ramirez, Rishi Verma, Lewis McGibbney
Interns: Rahul Palamuttam, Valerie Roth, Harsha Manjunatha, Renato Marroquin

Goals for expanding SciSpark outside of MCC, PDF and other cases

Data Reuse Tasks

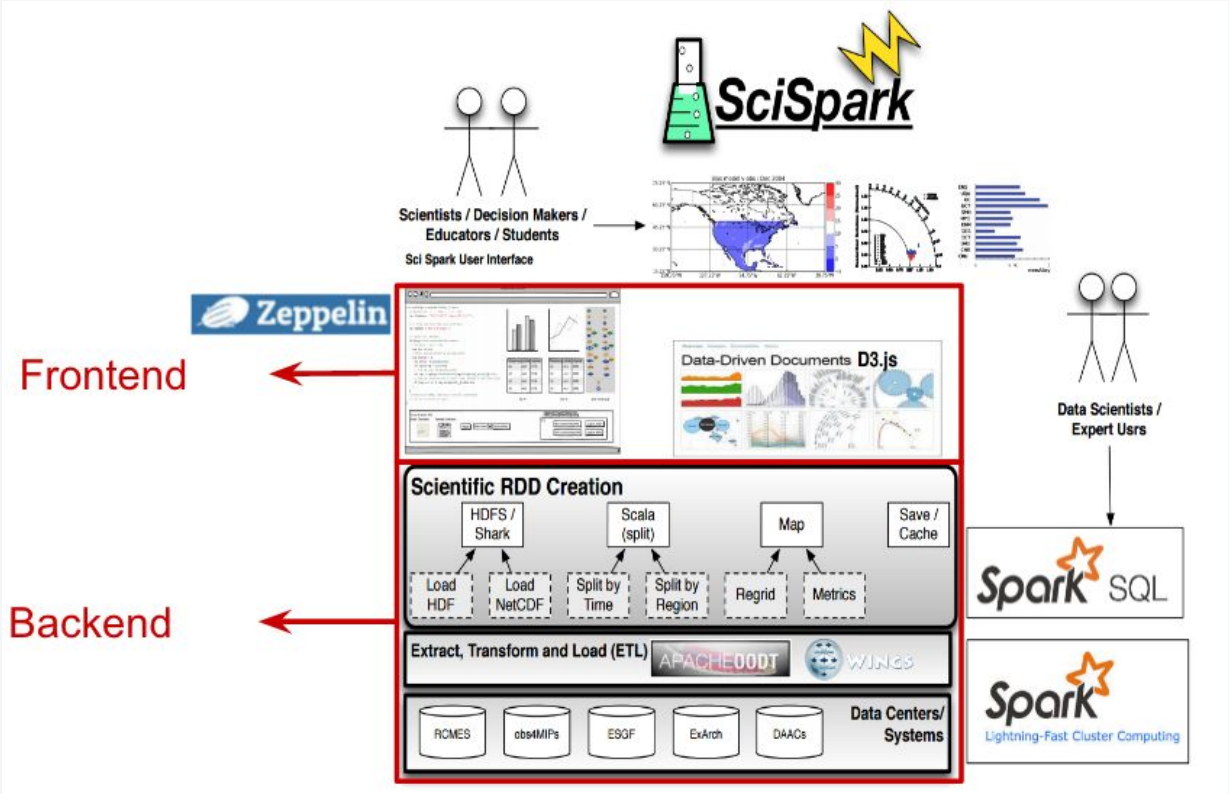
- Sparkler, Data Science
  - http://github.com/USCDataScience/sparkler/
  - Web Crawling on Spark - http://irds.usc.edu/
- Science
  - Search Analytics - AIST augmentation task - “Deep Data” extraction
  - Regional Climate Model Evaluation System (RCMES), Apache Open Climate Workbench (OCW), http://climate.apache.org/

SciSpark: A Two-Pronged Approach to Spark

- Extend Native Spark on the JVM (scala, java)
  - Handle Earth Science geolocated arrays (variables as a function of time, lat, lon)
  - Provide netCDF & OPeNDAP data ingest
  - Provide array operators like numpy
  - Implement two complex Use Cases:
    - Mesoscale Convective Systems (MCS) - discover cloud elements, connect graph
    - PDF Clustering of atmospheric state over N. America

- Exploit PySpark (Spark jobs using python gateway)
  - Exploit numpy & scipy ecosystem
  - Port and parallelize existing python analysis codes in PySpark
    - Statistical Rollups over Long Time-Series
    - Regional Climate Model Evaluation System (RCMES)
    - Many others

SciSpark: Envisioned Architecture



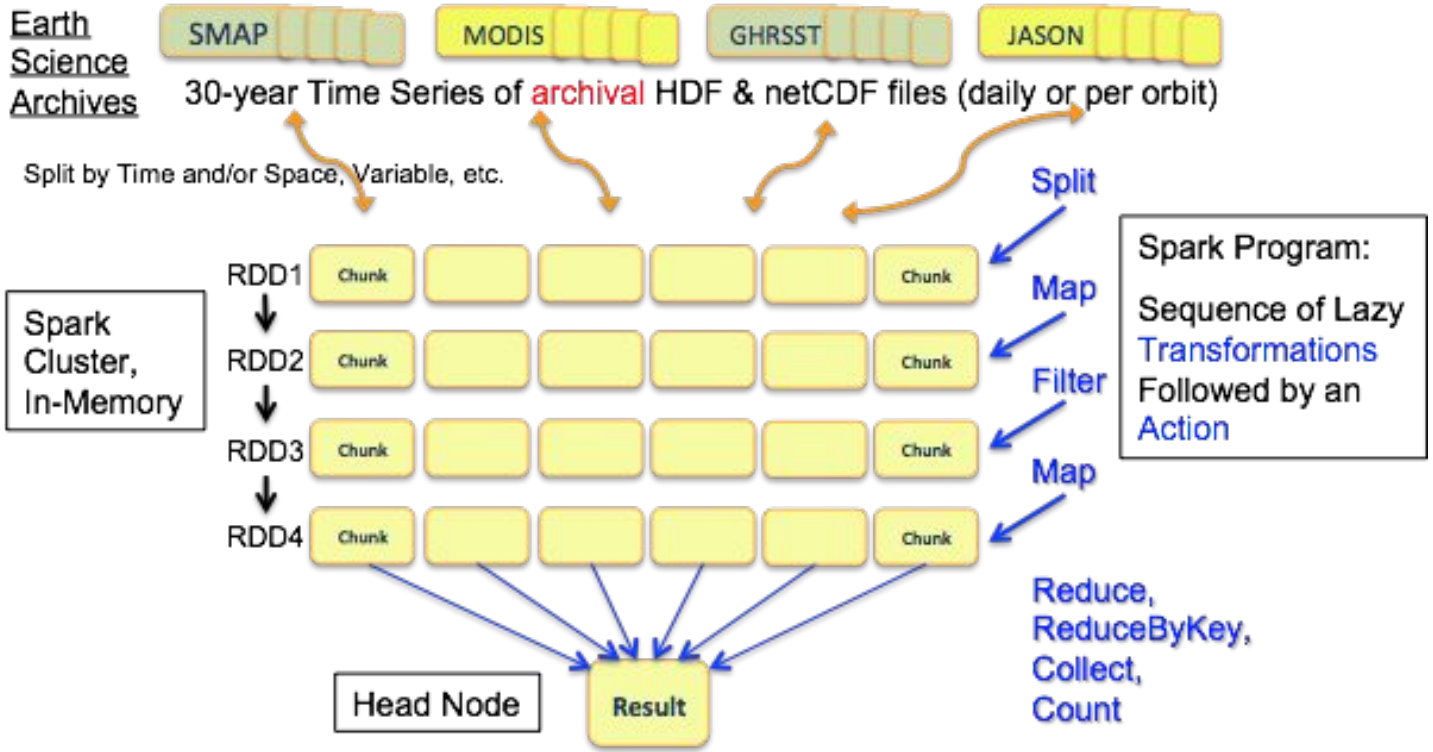
PySpark Gateway

- Has full Python ecosystem available
  - Numpy N-dimensional arrays
  - SciPy algorithms
- However, now have two runtimes: Python & JVM
  - Py4J gateway moves simple data structures back-and-forth
  - At expense of communication overhead, format conversions
- Apache Spark actually supports three languages:
  - Scala, Python, Java
  - Can translate from Python to Scala almost line-by-line

Sources of Parallel Work (Independent Data)

- Parallelize Over Time:
  - Variables (grids) over a Long 10-30 Year Time-Series (keep grid together)
- Parallelize Over Space (and/or Altitude):
  - By Pixel: Independent Time-Series, Massive Parallelism
  - By Spatial Tiles: Subsetting, Area Averaging, Stencils (NEXUS)
  - By Blocks: Blocked Array operations (harder algorithms)
- Parallelize Over Variable, Model, Metrics, Parameters, etc.
  - Ensemble Model evaluation with multiple metrics
  - Search Parameter Space with many analysis anruns

What is Apache Spark? - In-memory Map-Reduce



What is Apache Spark? - In-memory Map-Reduce

- Datasets partitioned across a compute cluster by key
  - Shard by time, space, and/or variable
- RDD: Resilient Distributed Dataset
  - Fault-tolerant, parallel data structures
  - Intermediate results persisted in memory
  - User controls the partitioning to optimize data placement
- New RDD's computed using pipeline of transformations
  - Resilience: Lost shards can be recomputed from saved pipeline
- Rich set of operators on RDD's
  - Parallel: Map, Filter, Sample, PartitionBy, Sort
  - Reduce: GroupByKey, ReduceByKey, Count, Collect, Union, Join
- Computation is implicit (Lazy) until answers needed
  - Pipeline of Transformations implicitly define a New RDD
  - RDD computed only when needed (Action): Count, Collect, Reduce
- Persistence Hierarchy (SaveTo)
  - Implicit Pipelined RDD, In-Memory, On fast SSD, On Hard Disk

Apache Spark Transformations & Actions

Transformations	*map*(*f* : T ⇒ U)	:	RDD[T] ⇒ RDD[U]
	filter(*f* : T ⇒ Bool)	:	RDD[T] ⇒ RDD[T]
	flatMap(*f* : T ⇒ Seq[U])	:	RDD[T] ⇒ RDD[U]
	sample(*fraction* : Float)	:	RDD[T] ⇒ RDD[T] (Deterministic sampling)
	groupByKey()	:	RDD[(K, V)] ⇒ RDD[(K, Seq[V])]
	reduceByKey(*f* : (V, V) ⇒ V)	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	union()	:	(RDD[T], RDD[T]) ⇒ RDD[T]
	join()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (V, W))]
	cogroup()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (Seq[V], Seq[W]))]
	crossProduct()	:	(RDD[T], RDD[U]) ⇒ RDD[(T, U)]
	mapValues(*f* : V ⇒ W)	:	RDD[(K, V)] ⇒ RDD[(K, W)] (Preserves partitioning)
	sort(*c* : Comparator[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	partitionBy(*p* : Partitioner[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
Actions	*count*()	:	RDD[T] ⇒ Long
	collect()	:	RDD[T] ⇒ Seq[T]
	reduce(*f* : (T, T) ⇒ T)	:	RDD[T] ⇒ T
	lookup(*k* : K)	:	RDD[(K, V)] ⇒ Seq[V] (On hash/range partitioned RDDs)
	save(*path* : String)	:	Outputs RDD to a storage system, *e.g.*, HDFS

Table 2: Transformations and actions available on RDDs in Spark. Seq[T] denotes a sequence of elements of type T.

SciSpark Project Contributions

- Parallel Ingest of Science Data from netCDF and HDF
  - Using OPeNDAP and Webification URL's to slice arrays
- Scientific RDD's for large arrays (sRDD's)
  - Bundles of 2,3,4-dimensional arrays keyed by name
  - Partitioned by time and/or space
- More Operators
  - ArraySplit by time and space, custom statistics, etc.
- Sophisticated Statistics and Machine Learning
  - Higher-Order Statistics (skewness, kurtosis)
  - Multivariate PDF's and histograms
  - Clustering, Graph algorithms
- Partitioned Variable Cache (in development)
  - Store named arrays in distributed Cassandra db or HDFS
- Interactive Statistics and Plots
  - "Live" code window submits jobs to SciSpark Cluster
  - Incremental statistics and plots "stream" into the browser UI

Spark Split Methods

- Sequence files: text files, log files, sequences of records
  - If in HDFS, already split across the cluster (original use for Hadoop & Spark)
- sc.parallelize(list, numPartitions=32)
  - Split a list or array into chunks across the cluster
- sc.textFile(path, numPartitions=32), sc.binaryFile()
  - Split a text (or binary) file into chunks of records
  - Each chunk is set of lines or records
- sc.wholeTextFiles(), sc.wholeBinaryFiles()
  - Split list of files across the cluster, whole files are kept intact
- Custom Partition methods
  - Write your own

SciSpark Extensions for netCDF

- sciSparkContext.NetcdfFile([path or DAP URL], numPartitions=32)
  - Read multiple variables (arrays) & attributes into a sciTensor
- sciSparkContext.NetcdfDFSFile([list of netCDF files in local dir. or HDFS])
  - Split a list or array into chunks across the cluster
- sciSparkContext.openPath([all netCDF files nested under top directory])
- sciSparkContext.readMERGFile()
  - Read a custom binary file

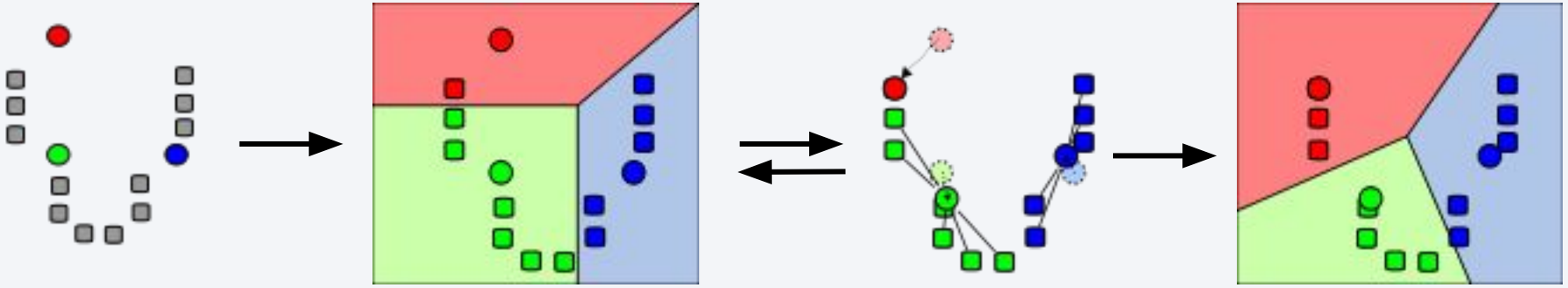
Discussion at End of Session: How could I use SciSpark for X? (Stump the chumps!)

- Sources of Parallelism: time, space, model/variable, metric, etc.
- Parallel Statistical Rollup is an example of key techniques.
  - Notebook 101-4

Example Use Case: PDF Clustering

- Problem: We wish to thoroughly test climate models with respect to observations in the ability to simulate climate extremes (eg heat waves, heavy rainfall events), and hence, probability distributions of such climate variables
- However, visualizing PDFs for every location (grid point) can be cumbersome
- Solution: Use k-means clustering to group grid points together based on similarities in PDF structure (for example, standard deviation and skewness)

K-means clustering algorithm



Step 1
Pick k initial centroids (here k=3)

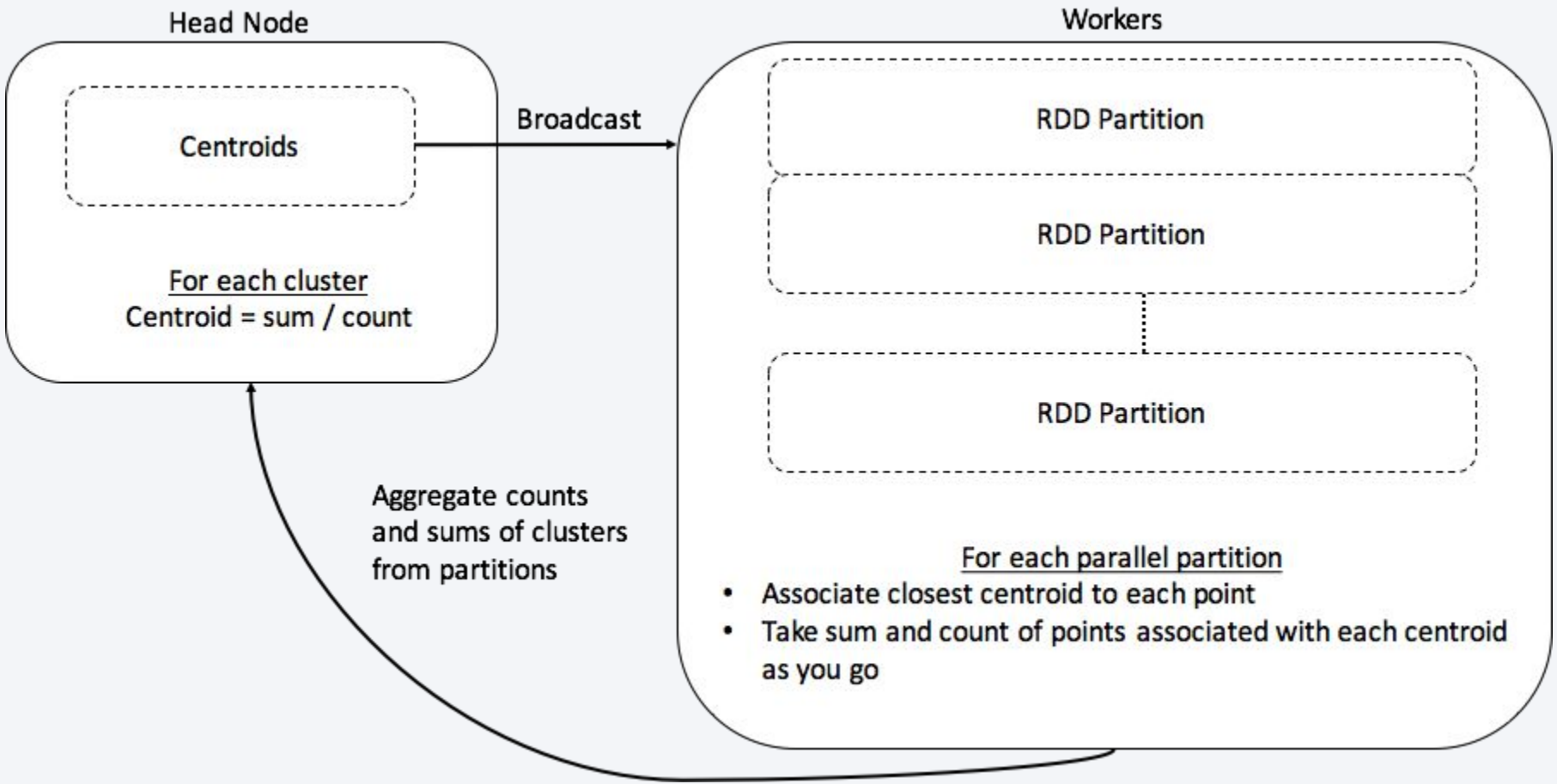
Step 2
Group points into clusters based on closest distance to each centroid

Step 3
Define means of points in each cluster as new centroids

Step 4
Repeat previous two steps until convergence

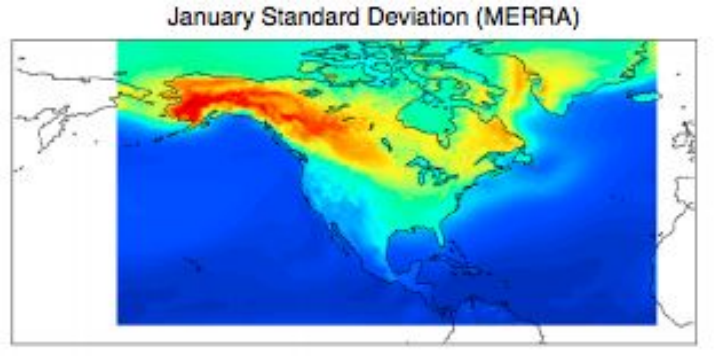
Image Source: Wikipedia

Parallelized k-means (steps 2-3)



PDF Clustering with real data (Loikith et al., 2013)

- Data
  - NASA MERRA reanalysis daily mean surface temperature over North America for January, 1979-2011
  - Inputs to k-means: histogram bin counts OR standard deviation and skewness, k=5 clusters
- PDF Moments:



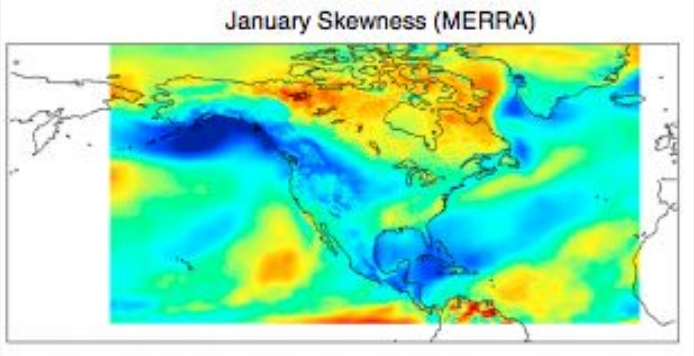


Image Source: Loikith et al., 2013

Review: PDF Moments

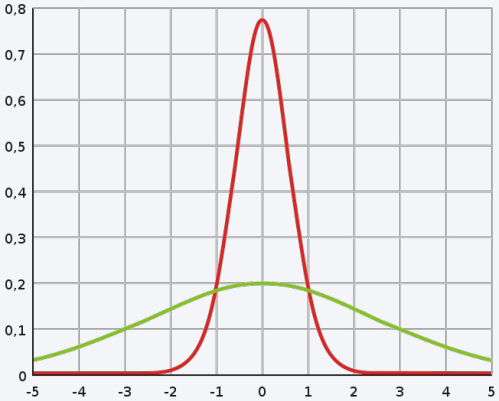
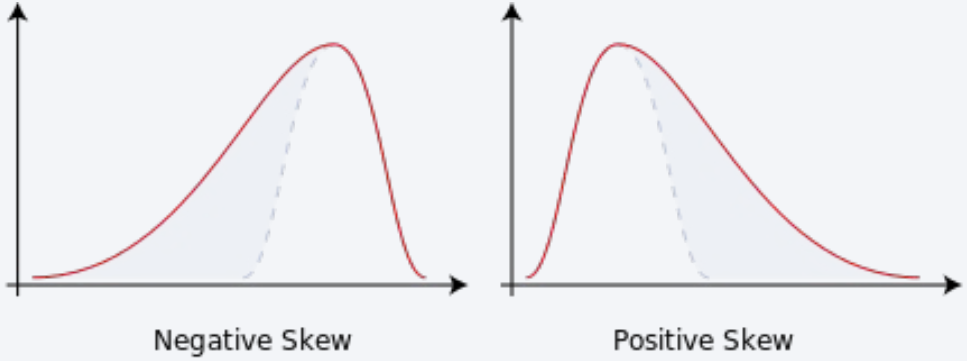


Image Source: Wikipedia

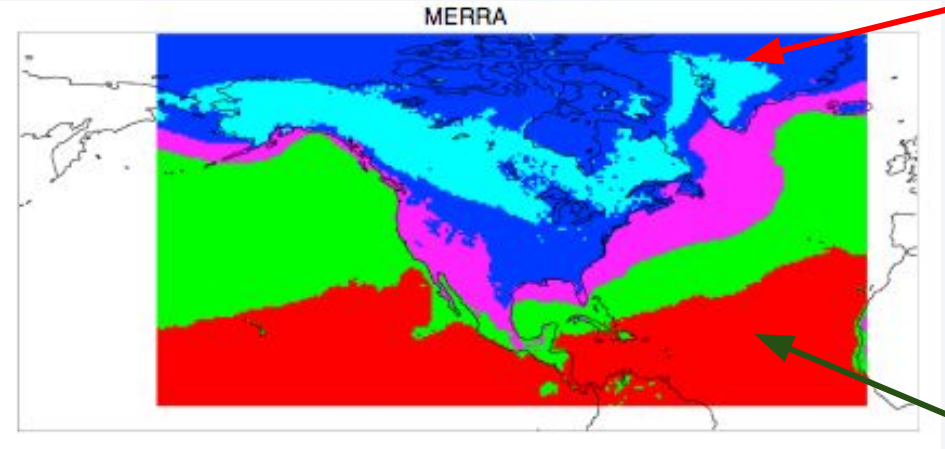
Standard Deviation (2nd Moment)
Width of PDF

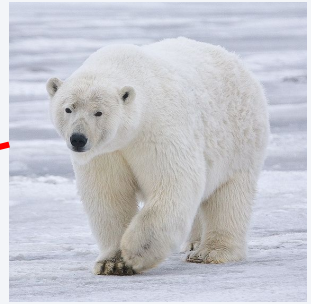
Skewness (3rd Moment)
Asymmetry of PDF



Results and Interpretation

Image Source: Loikith et al., 2013





Blue clusters: cold, arctic climate with larger deviations in temperature. Also note larger deviations over land than ocean.

Red and green clusters: Warm, tropical climate with smaller deviations in temperature

In the following notebook, we will reproduce these figures from Loikith et al. 2013 using pyspark's built-in implementation of k-means clustering.

Notebook 301-1:

PDF Clustering using PySpark

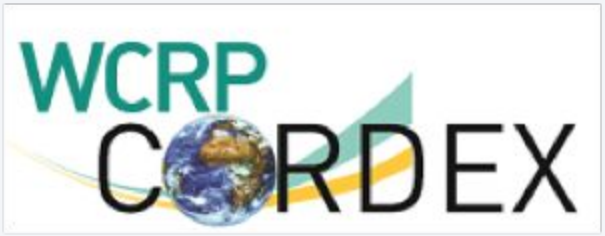
Notebook 301-2:

PDF Clustering using Scala

Notebook 301-3:

PDF Clustering Side by Side

Session D2:
Training Session on Regional Climate Model Evaluation System (RCMES)



ICRC-CORDEX 2016
May 19th, 2016
14:00-17:00

http://rcmes.jpl.nasa.gov

http://climate.apache.org

Notebook 301-4:

Parallel RCMES/OCW (optional)

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

Climate Model Diagnostic Analyzer

Carnegie Mellon University

- PI: Jia Zhang (Carnegie Mellon University)
- Co-Is: Seungwon Lee, Chris Mattmann (NASA Jet Propulsion Laboratory)
- Collaborators: Jonathan Jiang, Zhangfan Xing (NASA Jet Propulsion Laboratory)
- Team Members: Lei Pan, Benyang Tang, Chengxing Zhai, Qihao Bao, Namrata Malarout

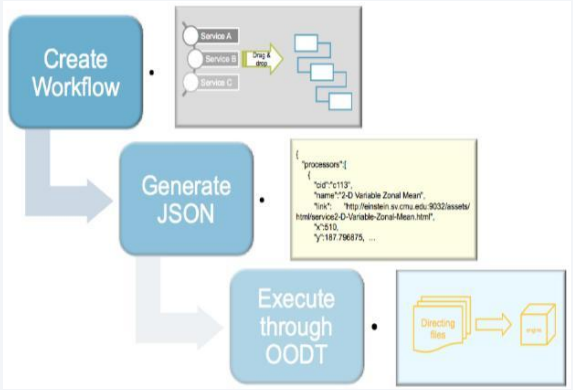
CMDA System

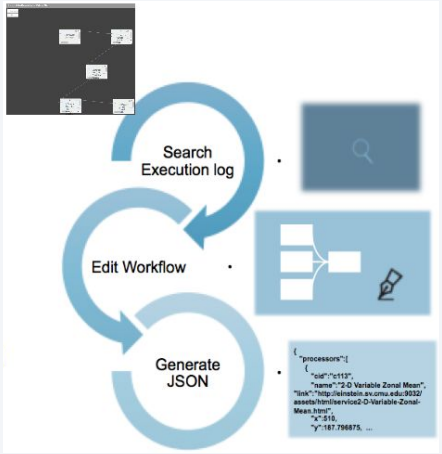
Carnegie Mellon University

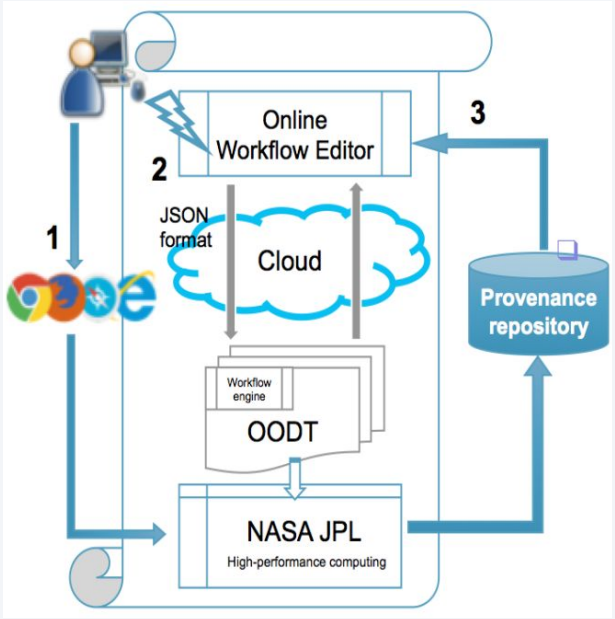
- Goal: Develop a technology to help Earth scientists create and manage workflows for scientific calculations.
- Objectives
  - Understand how Earth scientists conduct data analytics in their daily research
  - Develop a CMDA workflow design tool which can chain web services
  - Develop a community-oriented Climate Model Diagnostic Analyzer (CMDA 2.0)
  - Develop a data-centric provenance repository with recommender system (CMU Team)
  - Develop a cloud-based governance layer to ensure scalability and performance

Provenance-powered Workflow Experiment

- 3 ways of workflow creation
  - Web-based service execution
  - Visual workflow creation
  - Provenance-based automatic workflow generation





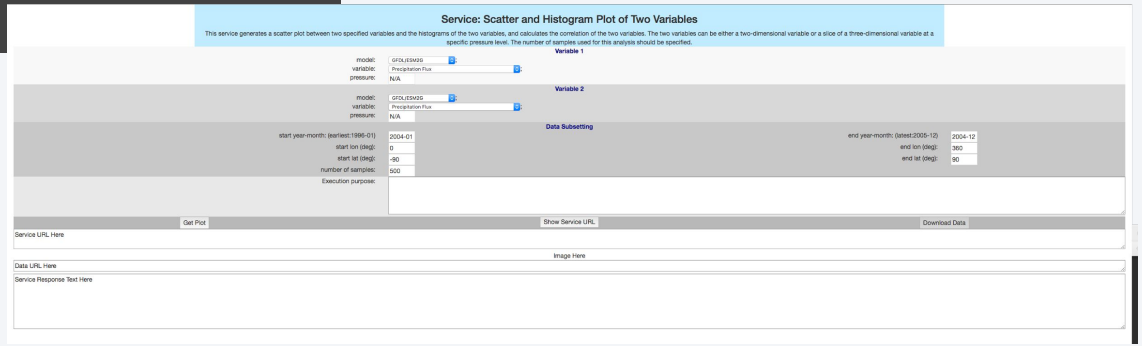


Jet Propulsion Laboratory
California Institute of Technology

SciSpark

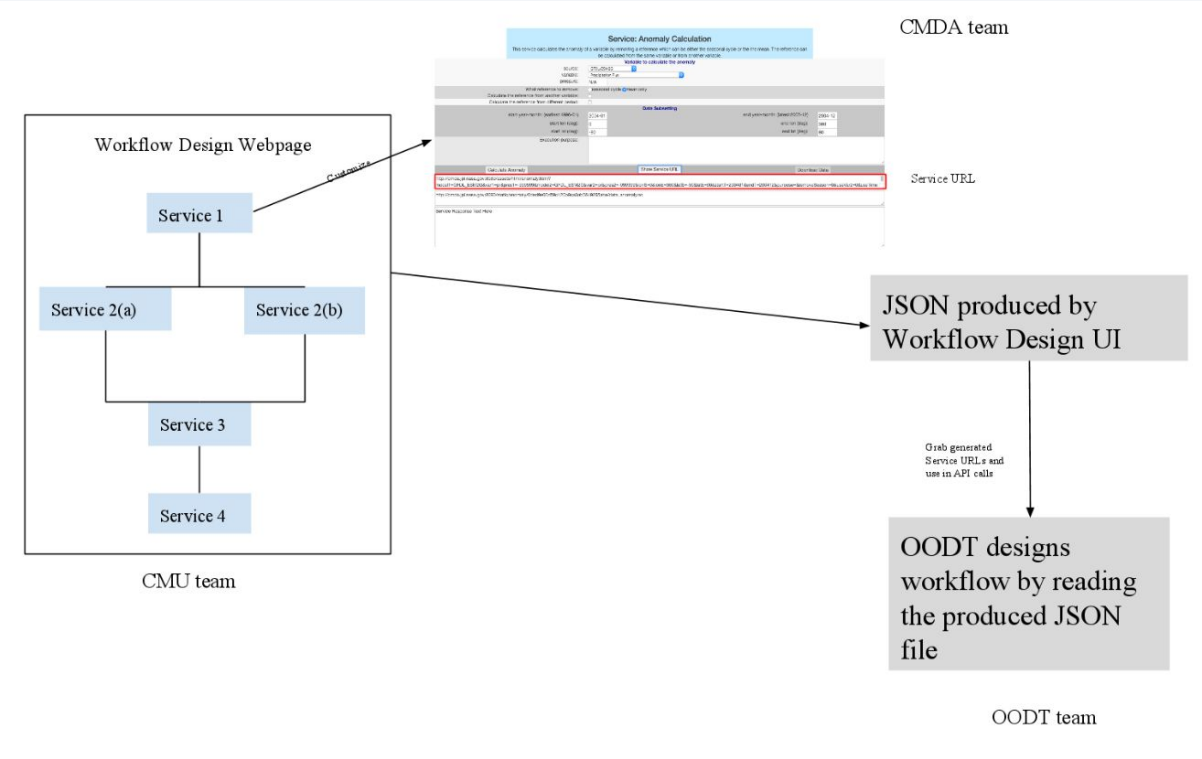
Online CMDA Workflow Editing Tool

Carnegie Mellon University



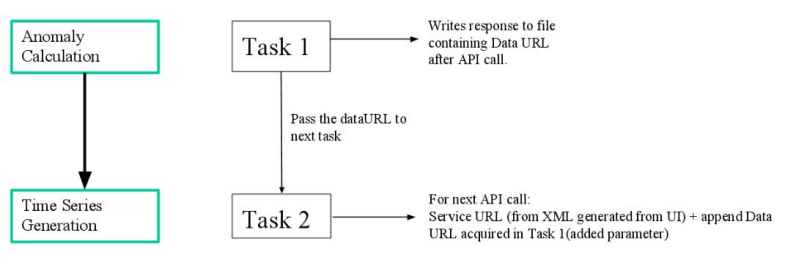
SciSpark

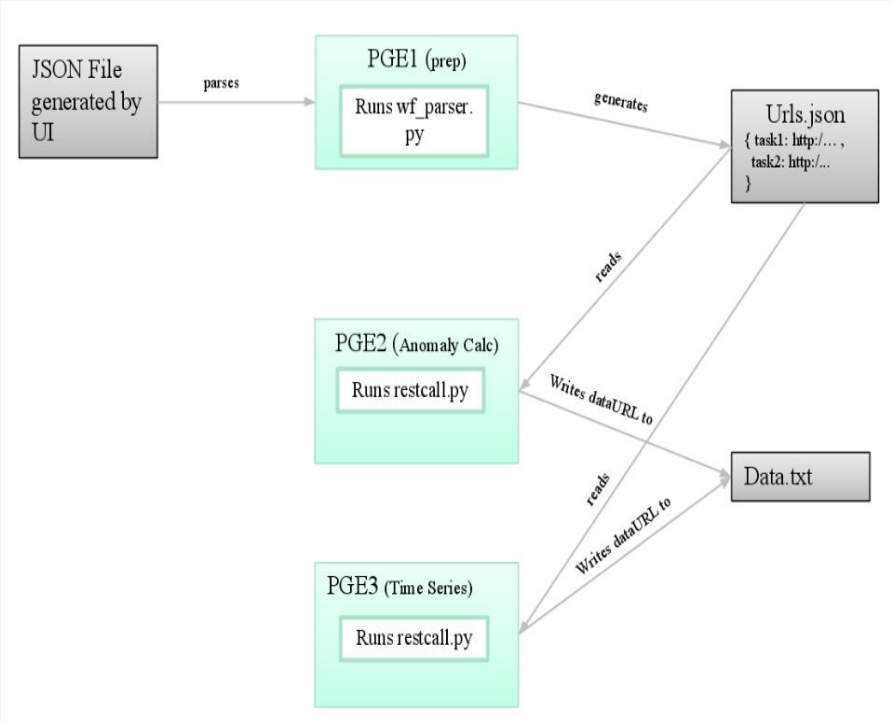
Workflow Implementation Design Decision



Simple Three Step Example

1. Prep :- Reads the JSON file generated by the frontend
   and creates a list of URLs for the workflow
2. Anomaly Calculation web service call:- Runs python script which makes REST call to the web service and extracts the dataURL from the JSON response.
3. Time Series web service call:- Uses dataURL from previous call as a parameter in Time Series API call.





Notebook 301-5:

CMDA Workflow Example

Spark Idioms / Lessons Learned (1)

- Partition data as needed to main "data locality" for your algorithm (e.g. here monthly rather than random).
- Write simple functions that you can pass to Map.
- Try to use the longest pipeline of Map and Filter functions.
- Only Reduce or ReduceByKey at the end of your pipeline.
- Almost never use Collect (for BIG DATA, the result will be too big to collect).
- Can use Collect to look at intermediate results to debug (run on small subset of the data).
- Often better to write a sequential version of your code FIRST to debug.
- Use HDFS to write out intermediate results.

Spark Idioms / Lessons Learned (2)

- “Never” use GroupByKey unless you absolutely have to (shuffles slow)
  - Prefer ReduceByKey (faster)
  - Good Use: Repartition or replicate data to other nodes
  - Use broadcast for small data
- Use Re-Keying and GroupByKey to Replicate Data (e.g. pairs of two time epochs)
- Use Accumulators instead of Multiple Reduces or Collects
- In Spark shells or Notebooks, Global Variables won't be pulled into Lambda expressions, so use Literals
- Must make an SQLContext to use Spark SQL or DataFrames
- JVM Performance: Increase Max. Memory, Monitor Garbage Collection

Notebook 301-6:

Spark Idioms / Lessons Learned

Discussion: How could I use SciSpark for X? (Stump the chumps!)

- Sources of Parallelism: time, space, model/variable, metric, etc.
- Parallel Statistical Rollup is an example of key techniques.
  - Notebook 101-4

Sources of Parallel Work (Independent Data)

- Parallelize Over Time:
  - Variables (grids) over a Long 10-30 Year Time-Series (keep grid together)
- Parallelize Over Space (and/or Altitude):
  - By Pixel: Independent Time-Series, Massive Parallelism
  - By Spatial Tiles: Subsetting, Area Averaging, Stencils (NEXUS)
  - By Blocks: Blocked Array operations (harder algorithms)
- Parallelize Over Variable, Model, Metrics, Parameters, etc.
  - Ensemble Model evaluation with multiple metrics
  - Search Parameter Space with many analysis anruns

Parallel Statistical Rollups for Earth Time-Series