SciSpark Tutorial 101

Introduction to Spark
Super Cool Parallel Computing
In-Memory Map-Reduce
It Slices, It Dices, It Minces, ...
So Fast, You Won't Believe It!!!

ORDER NOW!!!

Agenda for 101:

- Intro. to SciSpark Project
- Access your SciSpark & Notebook VM (personal sandbox)
- What is Spark?
  - Lightning-Fast Cluster (Parallel) Computing
  - Simple programming in python or scala
- SciSpark Extensions
  - sciTensor: N-dimensional arrays in Spark
  - sRDD: scientific RDD's
- Notebook Technology
- Tutorial Notebooks:
  - 101-0: Spark Warmup
  - 101-1: Intro. to Spark (in scala & python)
  - 101-2: Spark SQL and DataFrames
  - 101-3: Simple sRDD Notebook (optional)
  - 101-4: Parallel Statistical Rollups for a Time-Series of Grids (pyspark)

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

Agenda for 201:

- Access your SciSpark & Notebook VM (personal sandbox)
- Quick recap. of SciSpark Project
  - What is Spark?
- SciSpark Extensions
  - sciTensor: N-dimensional arrays in Spark
  - sRDD: scientific RDD's
- Implementation of Mesoscale Convective Complexes (MCC) Analytics
  - Find cloud elements, analyze evolution, connect the graph
- Tutorial Notebooks:
  - 201-1: Using sciTensor and sRDD
  - 201-2: More SciSpark API & Lazy Evaluation
  - 201-3: End-to-end analysis of Mesoscale Convective Systems
  - 201-4: MCS Cloud Element Analytics

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

SciSpark AIST Project

AIST-14-0034 Funded by Mike Little

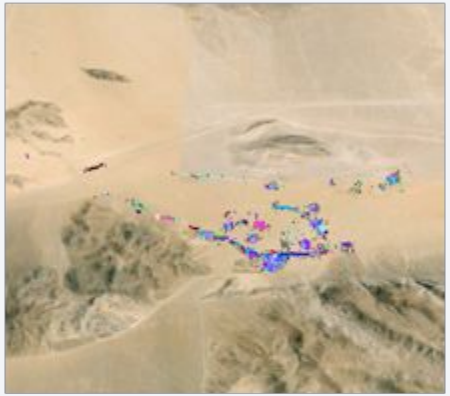
PI: Christian Mattmann
Project Mgr: Brian Wilson
JPL Team: Kim Whitehall, Sujen Shah, Maziyar Boustani, Alex Goodman, Wayne Burke, Gerald Manipon, Paul Ramirez, Rishi Verma, Lewis McGibbney
Interns: Rahul Palamuttam, Valerie Roth, Harsha Manjunatha, Renato Marroquin

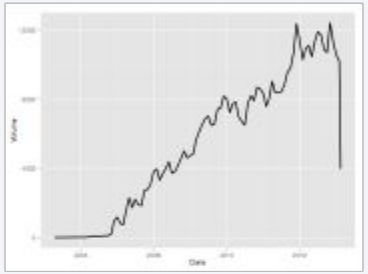
How did we get involved with Spark?

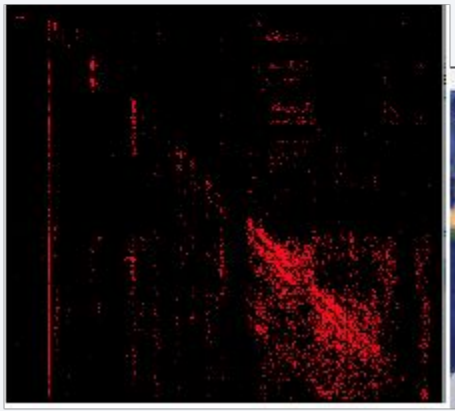
DARPA XDATA

- Grab the principals behind the leading infrastructure/viz technologies
  - Shove them in a tight space
  - Provide ~~beer~~ coffee and snacks
  - Provide awesome data and challenges
  - Provide infrastructure and connectivity
- Check in every day and 1x a week
- Wall of Shame/Fame
- New Challenges Each Week
- Midterm Presentations
  - Peanut Gallery
- Make people talk/socialize
- Put that all together

DARPA XDATA: Analytics + Viz







Met these fine people

- Ion Stoica
  CEO, DataBricks
  Co-Director,
  AMP Lab
- Matt Massie
  Dev Manager,
  AMP Lab
- Dr. Chris White, DARPA
  XDATA PM

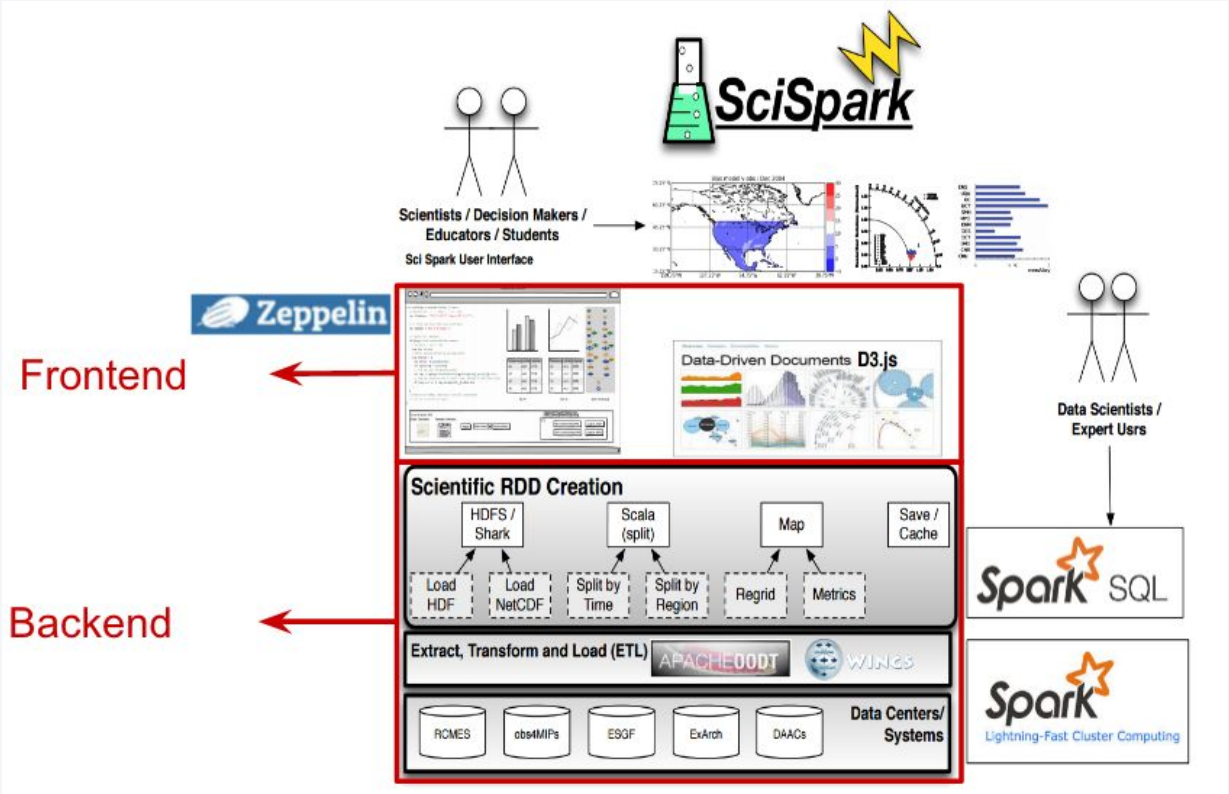
Motivation for SciSpark

•Experiment with in memory and frequent data reuse operations

–Regridding, Interactive Analytics such as MCC search, and variable clustering (min/max) over decadal datasets could benefit from in-memory testing (rather than frequent disk I/O)

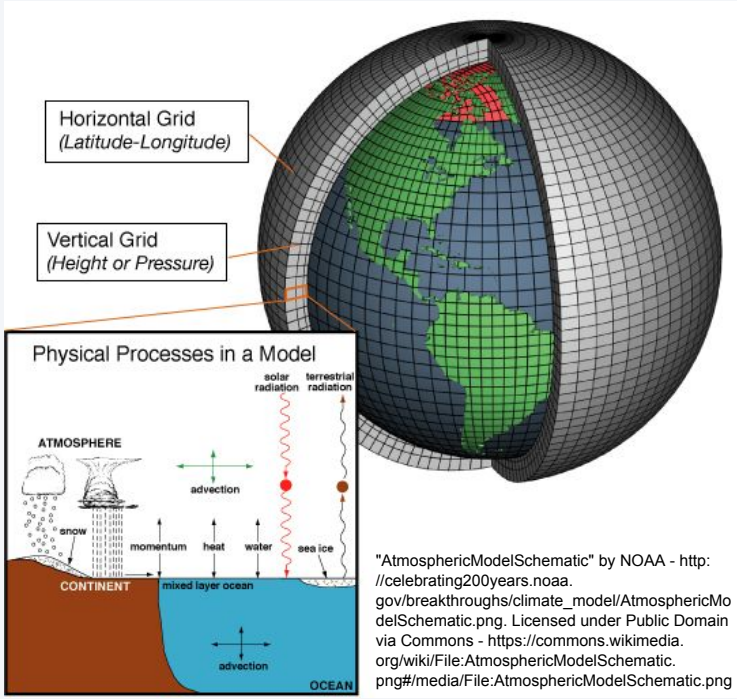
–Data Ingestion (preparation, formatting)

SciSpark: Envisioned Architecture



SciSpark: Motivation

- Climate and weather directly impacts all aspects of society
- To understand average climate conditions, and extreme weather events, we need data from climate models and observations
- These data are voluminous & heterogeneous - varying spatial and temporal resolutions in various scientific data formats
- Provides important information for policy- and decision- makers, as well as for Earth science and climate change researchers



"AtmosphericModelSchematic" by NOAA - http://celebrating200years.noaa.gov/breakthroughs/climate_model/AtmosphericModelSchematic.png. Licensed under Public Domain via Commons - https://commons.wikimedia.org/wiki/File:AtmosphericModelSchematic.png#/media/File:AtmosphericModelSchematic.png

SciSpark: an analytical engine for science data

- I/O is major performance bottleneck
  - Reading voluminous hierarchical datasets
- Batch processing limits interaction and exploration of scientific datasets
  - Regridding, interactive analytics, and variable clustering (min/max) over decadal datasets could benefit from in-memory computation
- Spark Resilient Distributed Dataset (RDD)
  - Enables re-use of intermediate results
  - Replicated, distributed, and reconstructed
  - In memory for fast interactivity

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

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

Three Challenges using Spark for Earth Science

- Adapting Spark RDD to geospatial 2D & 3D (dense) arrays
  - sRDD, with loaders from netCDF & HDF files
  - sciTensor using ND4J or Breeze
- Reworking complex science algorithms (like GTG) into Spark's map-, filter-, and reduce- paradigm
  - Generate parallel work at expense of duplicating some data
  - Port or redevelop key algorithms from python/C to Scala
- Performance for bundles of dense arrays in Spark JVM
  - Large arrays in memory require large JVM heap
  - Garbage collection overheads

Parallel Computing Styles

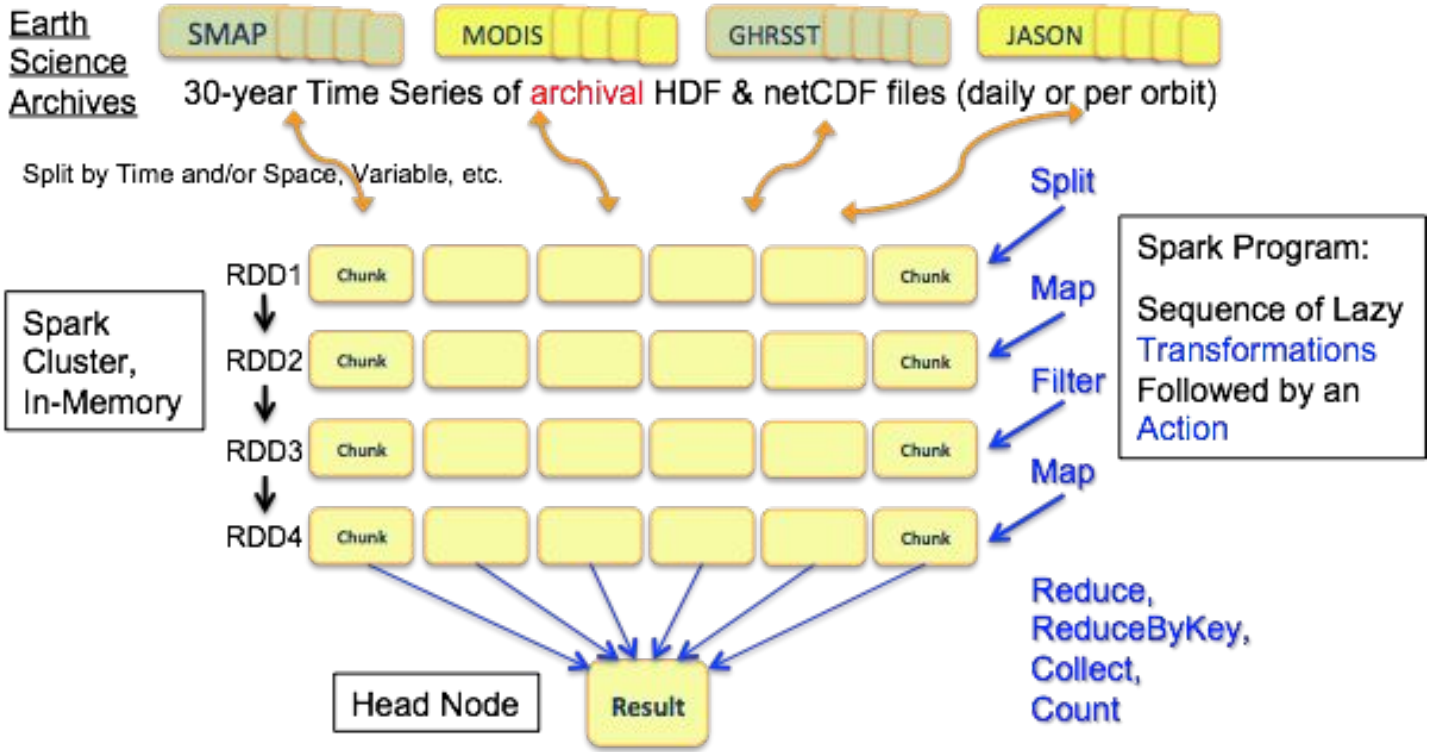
- Task-Driven:
  - Recursively Divide work into small Tasks
  - Execute them in parallel

- Data-Driven:
  - First, Partition the data across the Cluster
  - Then Compute using map-filter-reduce
  - Repartition (shuffle) data as necessary using groupByKey

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 analyses and runs

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

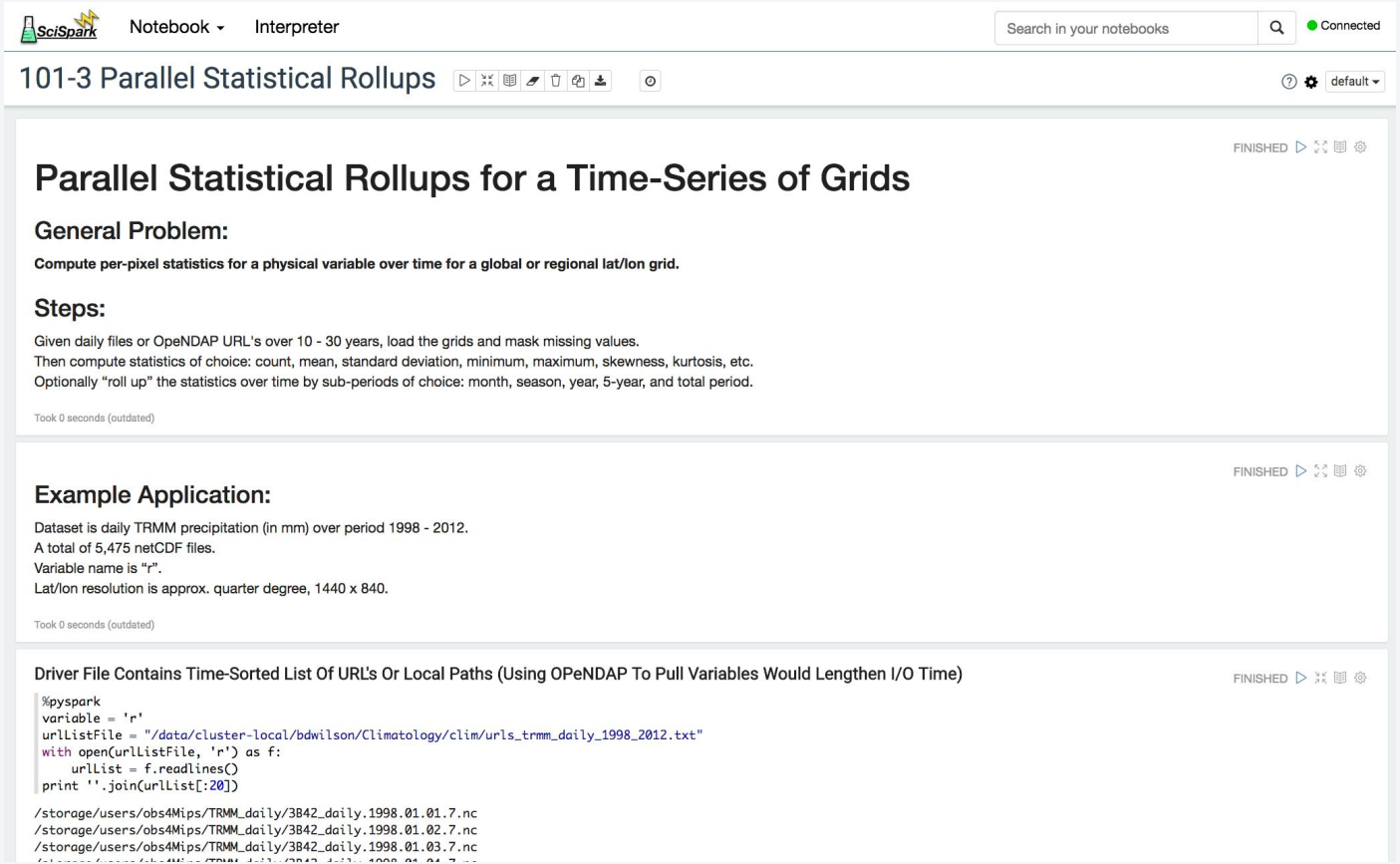
Spark Split Methods

- 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

SciSpark Front-end: Notebooks

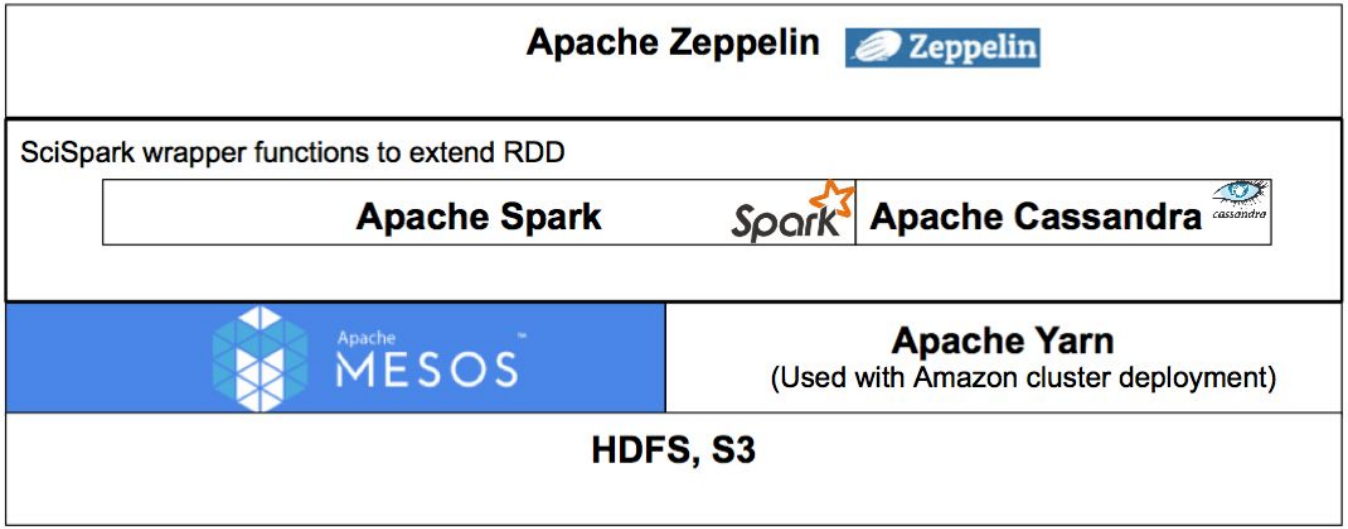


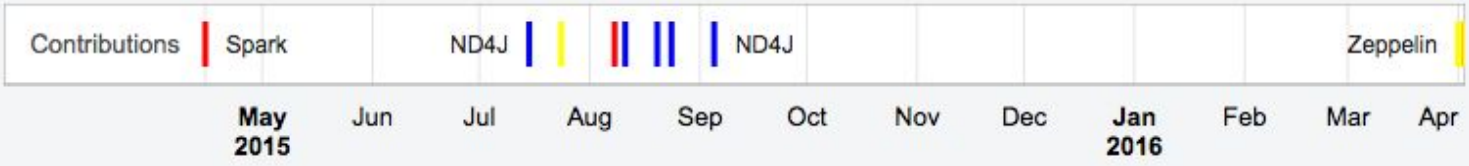
SciSpark Front-end: Notebooks

- Multi-interpreter support (by cell)
  - Scala (default), Python (%pyspark), Spark SQL (%sql), etc.
  - Documentation cells in markdown (%md), HTML, with images
- Notebook automatically connects to spark-shell (cluster)
  - SparkContext available as ‘sc’
  - Progress bars while executing Spark tasks
- Other features
  - Auto-saved as you type
  - Can share, copy & modify, group edit & view
  - Run your own Notebook server using Spark standalone

SciSpark Infrastructure Stack

- Leverages open-sourced projects

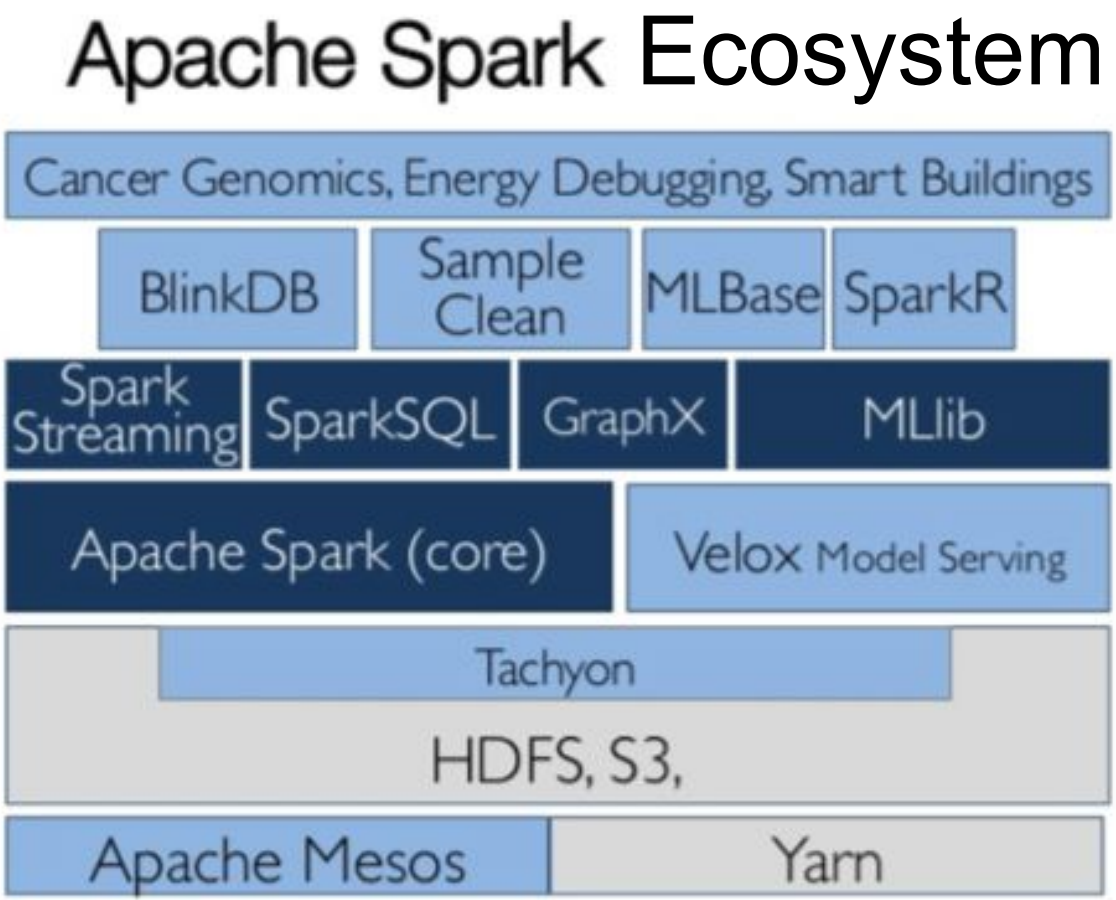




SciSpark Extensions To Core

SciSpark

sRDD
sciTensor



Notebook 101-0:

Spark Warmup

Notebook 101-1:

Intro. to Spark

Spark also does SQL and DataFrames

- This is useful for Earth Science Datasets because:
- Heterogeneity of data:
  - In delimited file formats such as .csv files (e.g. NOAA's NCEI StormData, some station data)
  - In text formats (e.g. NHC reports)
  - In SQL databases (e.g. NWS and WMO databases for climate variables)
- Analysis with these datasets is usually limited to single core processing in one environment and involves:
  - subsetting (required) data from the DB or the files and storing in tabular format
  - performing analysis available for tables
  - transferring the results to final locations. This process is usually not very reproducible.

Spark also does SQL and DataFrames

- The notebook interface backed by Spark allows for:
  - Loading (all) the data into the parallelized environment
  - Discovery through the ability to access to all the data within an SQL type of environment or a frame
  - Analysis through querying the data and visualizations

Notebook 101-2:

Spark SQL and DataFrames

Challenge of handling Earth Science data

- Heterogeneity of the data:
  - Some instruments produce point source data - stored in ASCII formats and tabular formats
  - Some instruments produce multi-dimensional data for the FOV as a function of time, level, lat, lon. These are stored in data formats ***hierarchical data formats*** that are self-describing, multidimensional and usually multivariable. Common formats are netCDF, HDF.
  - Models usually produce multi-dimensional data as well.
  - **The Spark engine does not inherently handle N-dimensional arrays**
- Size of the data:
  - As instruments and models improve in resolution, the data grow significantly
- Getting N-dimensional data into SciSpark

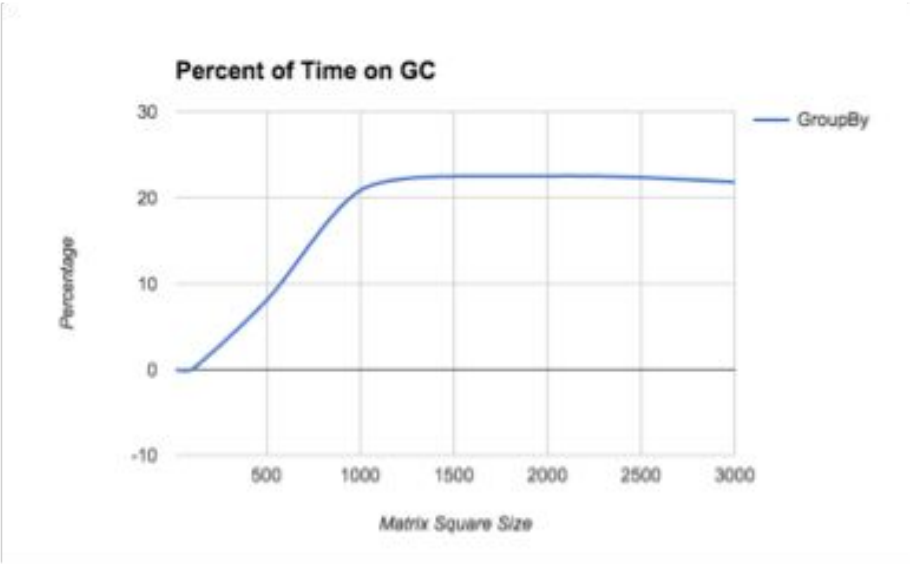
SciSpark Accomplishments

- ✓ sRDD architecture
  - Built sRDD and sciTensor to handle scientific data formats
- ✓ Demonstrated functionality to read hierarchical data
- ✓ Extend N-Dimensional Array operations
  - Provide common interface for both Breeze and ND4J
- ✓ Support loading scientific data from multiple sources & formats
- ✓ SciSpark Use Case: Distributed GTG
  - Generate a graph (nodes are cloud areas & edges are time) using SciSpark (run time 10 - 60 seconds)

SciSpark

Garbage Collection (GC) Performance Lessons

- Avoid unnecessary object creation
- Avoid creating a lot of small objects
- Avoid cartesian product!
- Multiple data copies can be advantageous
  - Large JVM heap leads to large GC overhead



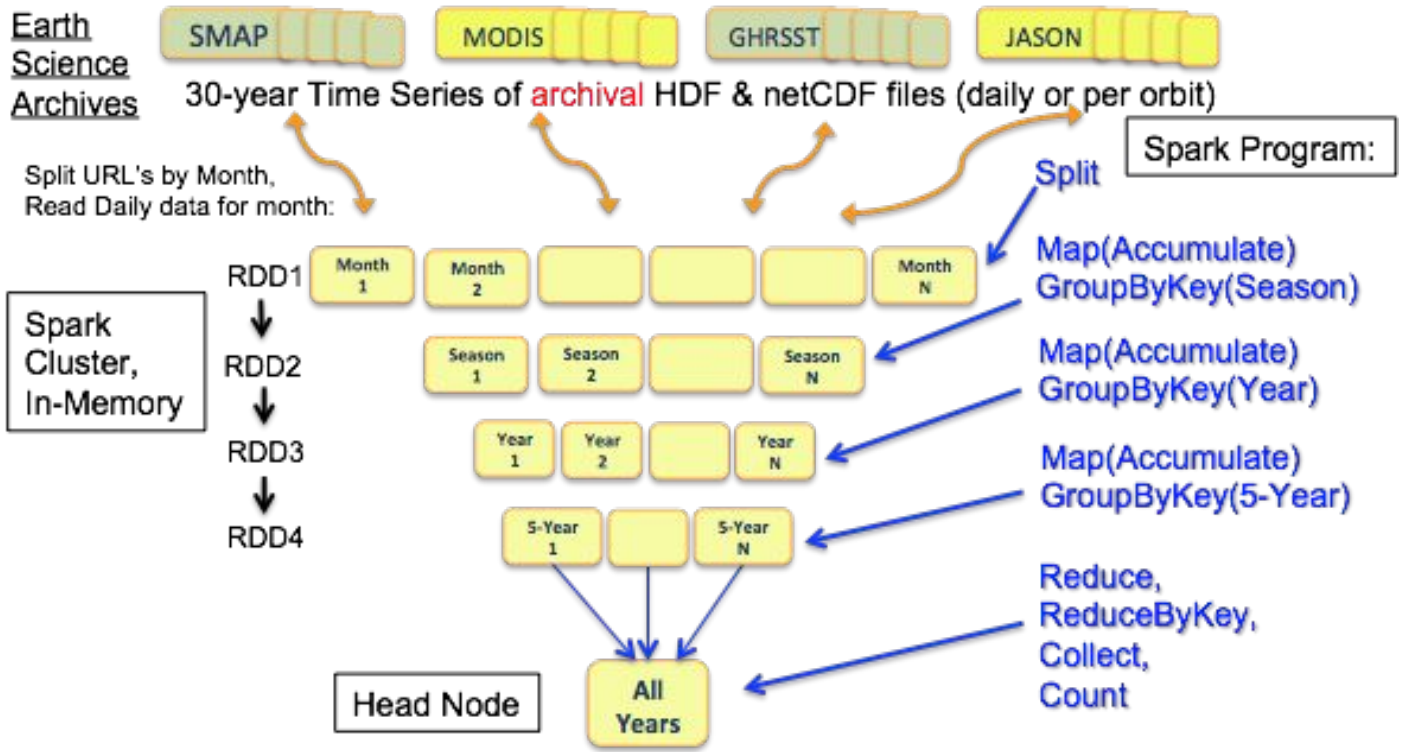
Notebook 101-3:

Simple sRDD Notebook

Parallel Statistical Rollups for a Time-Series of Grids

- General Problem:
  - Compute per-pixel statistics for a physical variable over time for a global or regional lat/lon grid.
- Specific Steps:
  - Given daily files or OpenDAP URL's over 10 - 30 years, load the grids and mask missing values.
  - Then compute statistics of choice: count, mean, standard deviation, minimum, maximum, skewness, kurtosis, etc.
  - Optionally "roll up" the statistics over time by sub-periods of choice: month, season, year, 5-year, and total period.

Parallel Statistical Rollups for Earth Time-Series



Notebook 101-4:

Parallel Statistical Rollups for Time-Series of Earth Science Grids

Discussion in 301:
How do you do X in SciSpark?

- What are your Use Cases?
  - Bring them to SciSpark 301
- Who is using Spark or Hadoop?
- Questions?

Notebook Exercises

- Try some of the exercises
- Just Play
- Ask Questions
- Notebook exercises:
  - 101-0: Spark Warmup
  - 101-1: Intro. to Spark (in scala & python)
  - 101-2: Spark SQL and DataFrames
  - 101-3: Simple sRDD Notebook
  - 101-4: Parallel Statistical Rollups for a Time-Series of Grids (pyspark)