SciSpark: Highly Interactive & Scalable Model Evaluation and Climate Metrics

Brian Wilson, Chris Mattmann, Duane Waliser, Jinwon Kim, Paul Loikith,
Huikyo Lee, Lewis John McGibbney, Maziyar Boustani, Michael Starch, Kim Whitehall
Jet Propulsion Laboratory / California Institute of Technology

Summary

Under a NASA AIST grant, we are developing a lightning fast Big Data technology called SciSpark based on Apache Spark. Spark implements the map-reduce paradigm for parallel computing on a cluster, but emphasizes in-memory computation, "spilling" to disk only as needed, and so outperforms the disk-based Apache Hadoop by 100x in memory and by 10x on disk, and makes iterative algorithms feasible. This 2nd generation capability for NASA's Regional Climate Model Evaluation System (RCMES) will compute simple climate metrics at interactive speeds, and extend to quite sophisticated iterative algorithms such as machine-learning (ML) based clustering of temperature PDFs, and even graph-based algorithms for searching for Mesocale Convective Complexes (MCC's). The goals of SciSpark are to: (a) Decrease the time to compute comparison statistics and plots from minutes to seconds; (b) Allow for interactive exploration of time-series properties over seasons and years; (c) Decrease the time for satellite data ingestion into RCMES to hours; (d) Allow for Level-2 comparisons with higher-order statistics and/or full PDF's in minutes to hours; and (e) Move RCMES into a near real time decision-making platform.

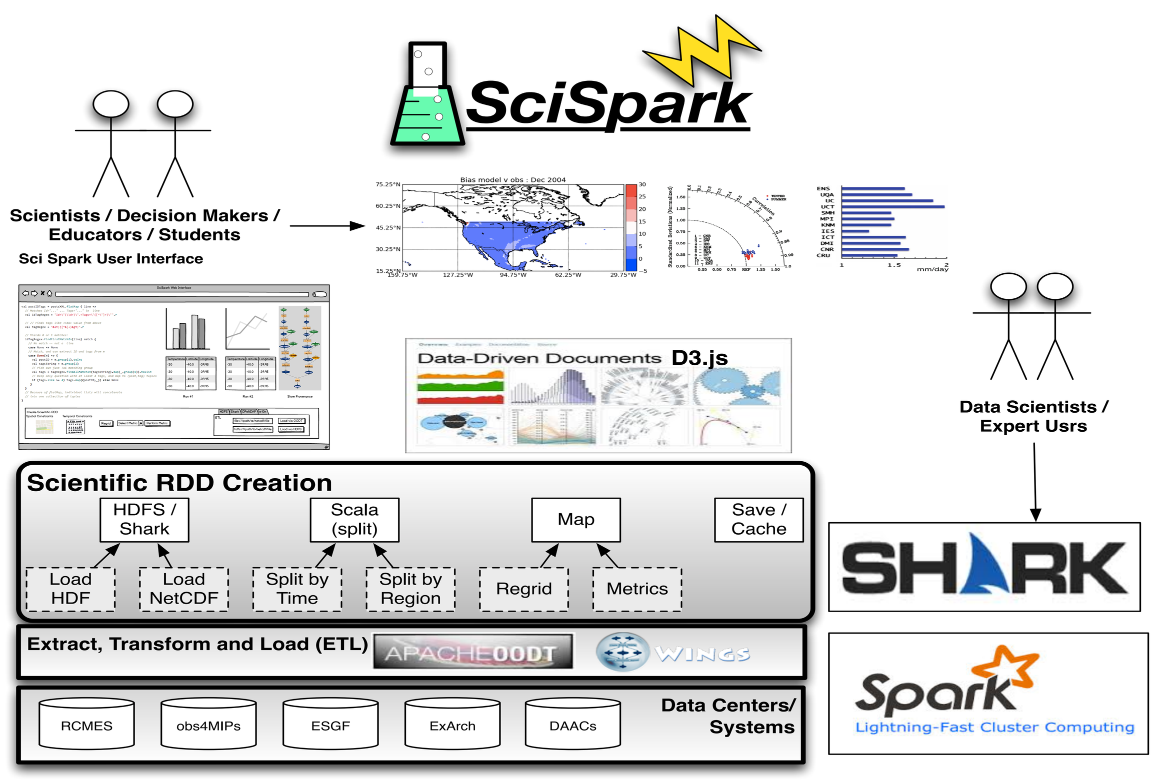
The capabilities of the SciSpark compute cluster will include:

1. On-demand data discovery and ingest for satellite (A-Train) observations and model variables (from CORDEX and CMIP5) by using OPeNDAP and webification (w10n) to subset arrays out of remote or local HDF and netCDF files;
2. Use of HDFS, Cassandra, and SparkSQL as a distributed database to cache variables/grids for later reuse with fast, parallel I/O back into cluster memory;
3. Parallel computation in memory of model diagnostics and decade-scale comparison statistics by partitioning work across the SciSpark cluster by time period, spatial region, and variable;
4. An integrated browser UI that provides a "live" code window (python & scala) to interact with the cluster, interactive visualizations using D3 and webGL, and search forms to discover & ingest new variables.

The research described here was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Spark: In-Memory Map-Reduce

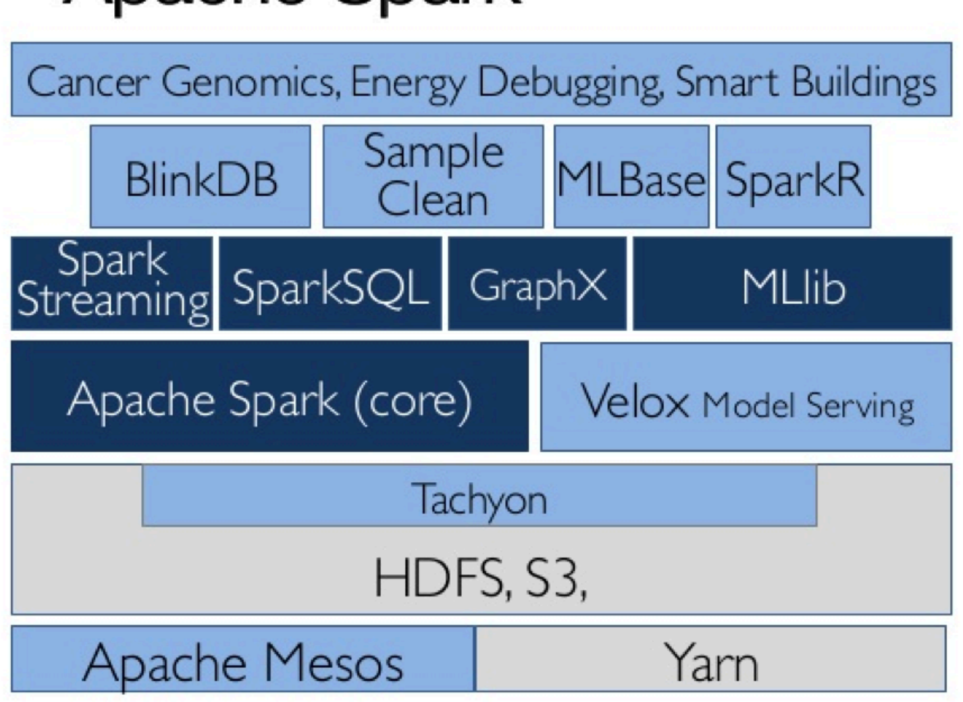
- Datasets partitioned across a compute cluster by key
  - Shard by time, space, and/or variable
- RDD: Resilient Distributed Dataset
  - Fault-tolerant, parallel data structures
  - Intermediate results persisted in memory
  - User controls the partitioning to optimize data placement
- New RDD's computed 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



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

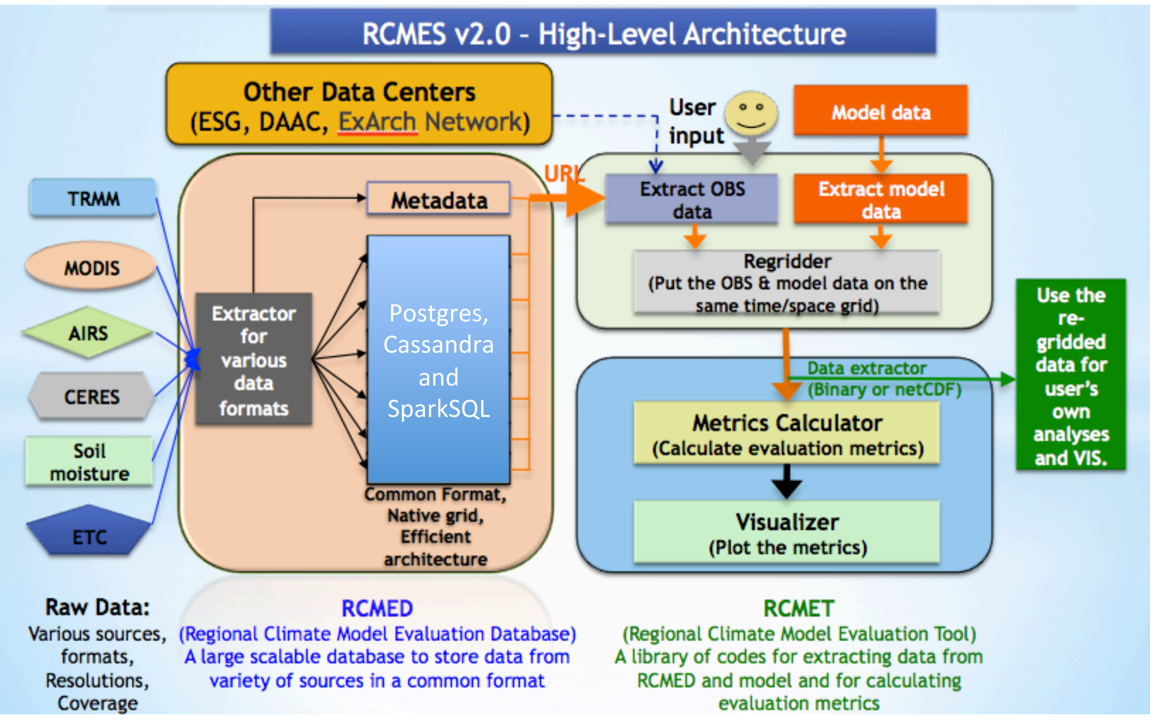
Table 2: Transformations and actions available on RDDs in Spark. Seq[T] denotes a sequence of elements of type T.

Apache Spark



SciSpark Contributions

- Parallel Ingest of Science Data from HDF & netCDF
  - 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
  - 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



The SciSpark Cluster

Apache Spark is an in-memory map-reduce platform. http://spark.apache.org/

Spark features include:

- Stream processing
- SQL Query Syntax
- Integration with Apache Mesos cluster manager
- Spark grew out of the Berkeley AMP Lab (Mattmann is on steering com)
  - Algorithms, Machines and People, investment from 80+ industry partners, DARPA XDATA and NSF CISE Expeditions in Computing

SciSpark is a deployment used to develop scientific Spark processing workflows.

The SciSpark test cluster provides:

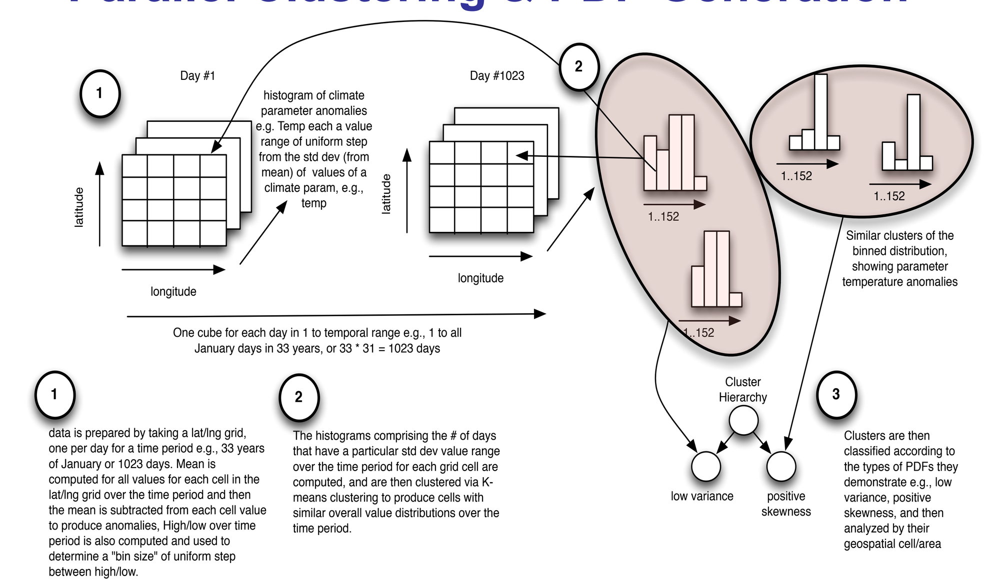
- Multiple nodes for parallel computation (4 nodes, 32 cores)
- Spark processing environment (Python, Scala, Java)
- Distributed file system (HDFS, Cassandra)
- Apache Mesos cluster manager

Credit: Mike Starch

Demo CSV File and PySpark Code

Sample File: Sample Code: Credit: Mazi Boustani

Parallel Clustering & PDF Generation



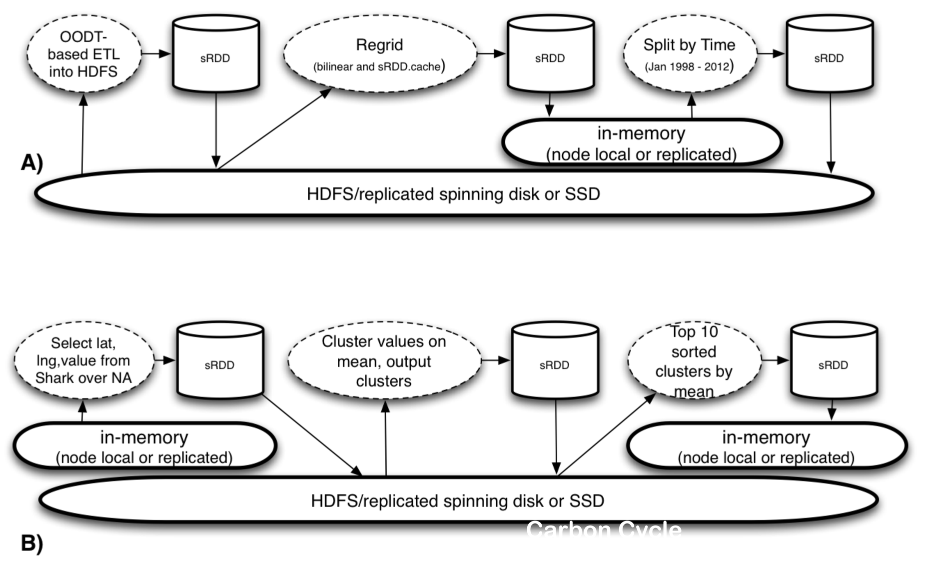
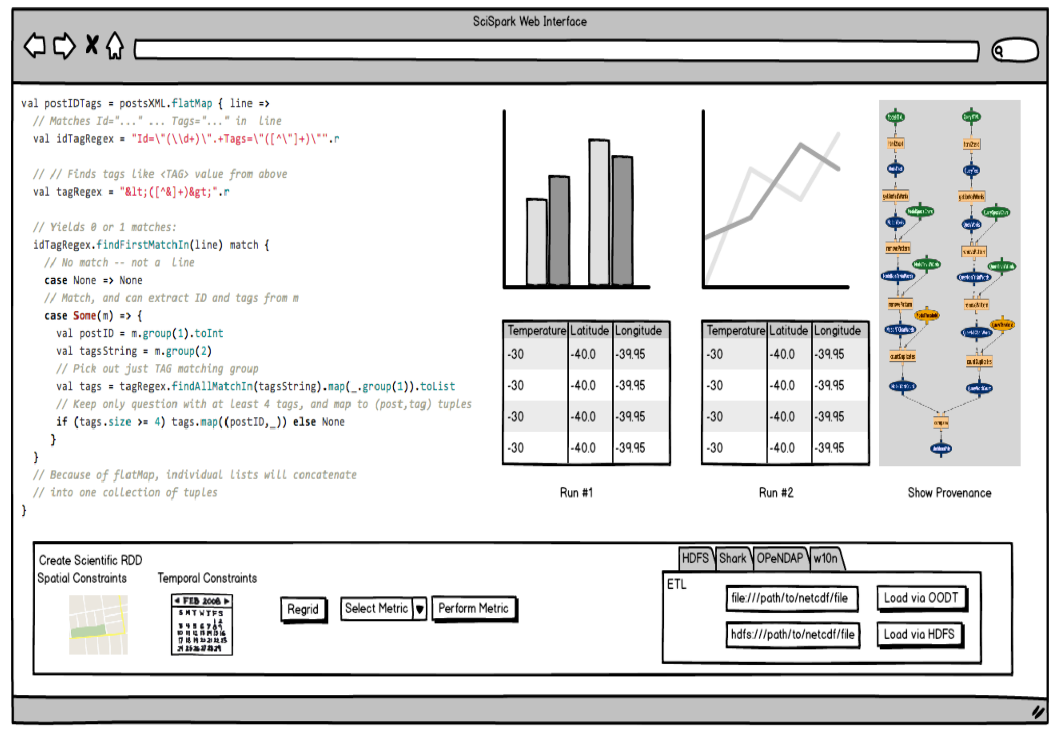


Figure 1. Two scenarios demonstrating intelligent data caching and access in SciSpark. A) a multi-stage operation to generate a time split of regridded data, B) a multi-stage operation to select climate parameters from Shark, to cluster by deviation from mean value, and then to output the first 10 clusters sorted by size.



Progress & Plans

- Set up test compute cluster
  - Installed Mesos, Spark, Cassandra
- Software Prototypes
  - Ingest global station data in CSV format, exercise SparkSQL, stats
  - Integrated code for reading arrays from netCDF, HDF, and DAP
- Architecture & Design
  - Designing data structures for scientific RDD's
  - Challenge: Interoperate between Python/numpy arrays and Java/Scala arrays (format conversion)
  - Prototyping Cassandra as key/value store for named arrays
- Next Steps
  - Reproduce prior RCMES model diagnosis runs in SciSpark paradigm
  - Quantify speedups
  - Implement custom statistics algorithms and "scale up" the cluster
  - Develop & Integrate the browser UI: live code, interactive viz.