Jet Propulsion Laboratory
California Institute of Technology

SciSpark

SciSpark 201

Searching for MCCs

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

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

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

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

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

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?

SciSpark: A Two-Pronged Approach to Spark

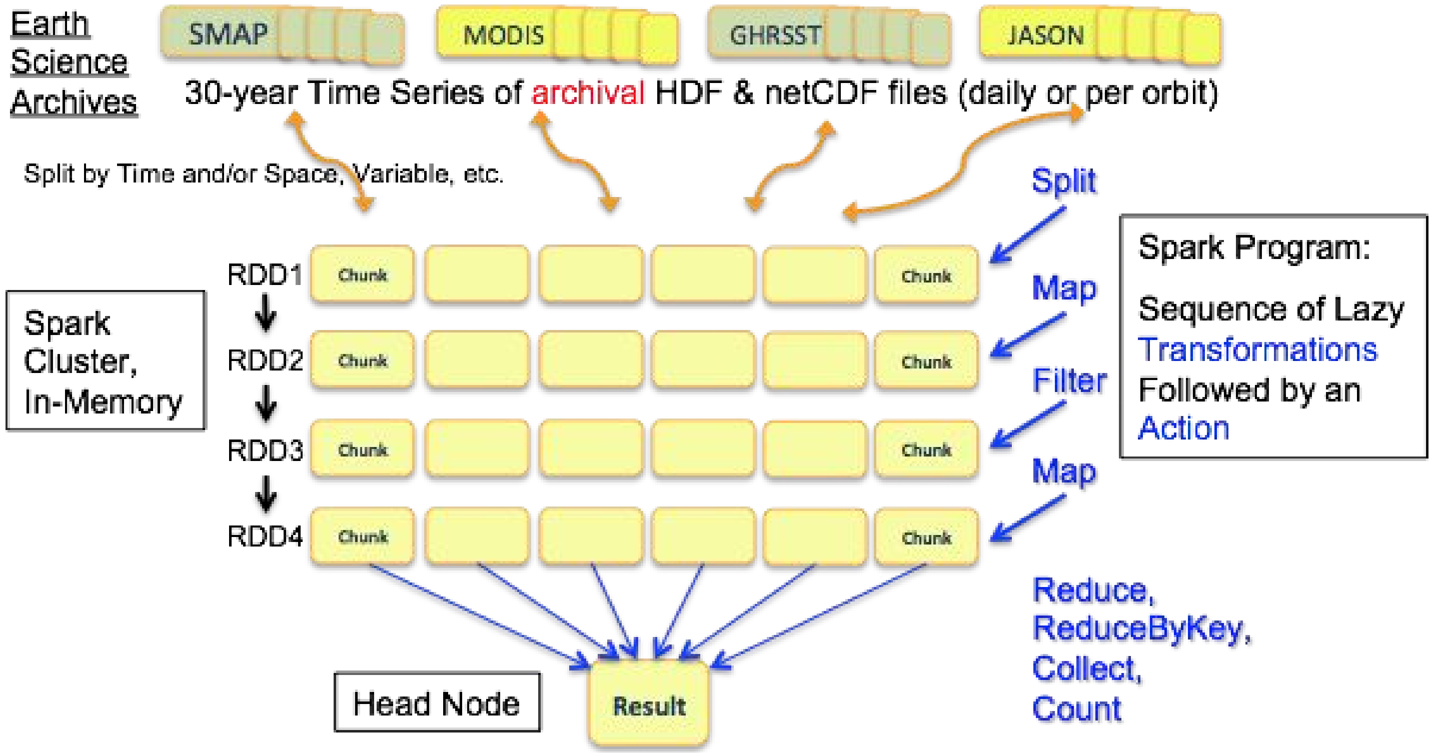
- Extend Native Spark on the JVM (scala, java)
    - Handle Earth Science geolocated arrays (variables as a function of time, lat, lon)
    - Provide netCDF & OPeNDAP data ingest
    - Provide array operators like numpy
    - Implement two complex Use Cases:
        - Mesoscale Convective Systems (MCS) - 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

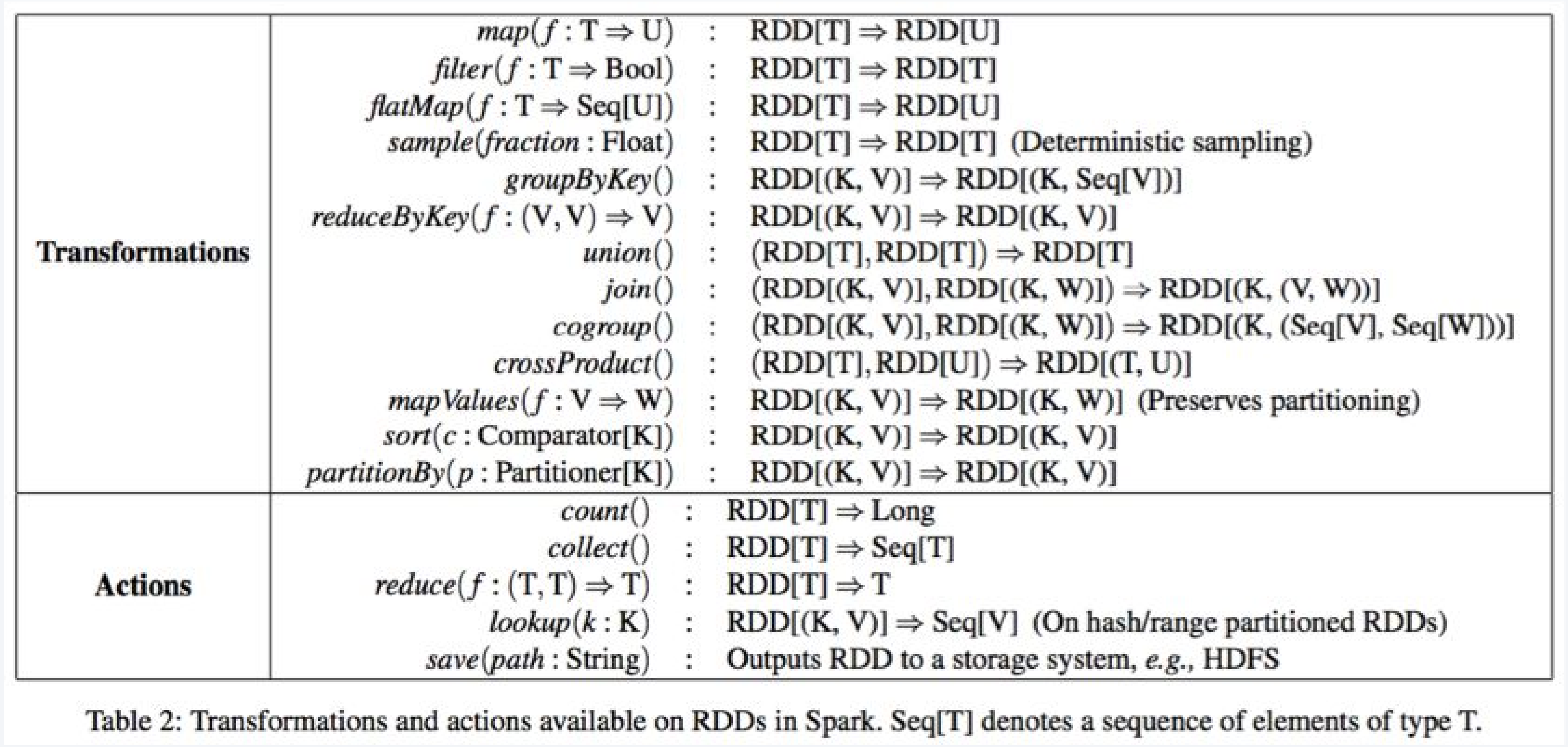
Sources of Parallel Work (Independent Data)

- 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


Earth Science Archives
SMAP
MODIS
GHRSST
JASON
30-year Time Series of archival HDF & netCDF files (daily or per orbit)
Split by Time and/or Space, Variable, etc.
Split
Map
Filter
Map
Spark Cluster, In-Memory
RDD1
RDD2
RDD3
RDD4
Chunk
Reduce, ReduceByKey, Collect, Count
Head Node
Result
Spark Program:
Sequence of Lazy Transformations Followed by an Action

Apache Spark Transformations & Actions



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

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

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

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

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

What is the goal of scientific RDD (sRDD)?

The goal of the sRDD was to:

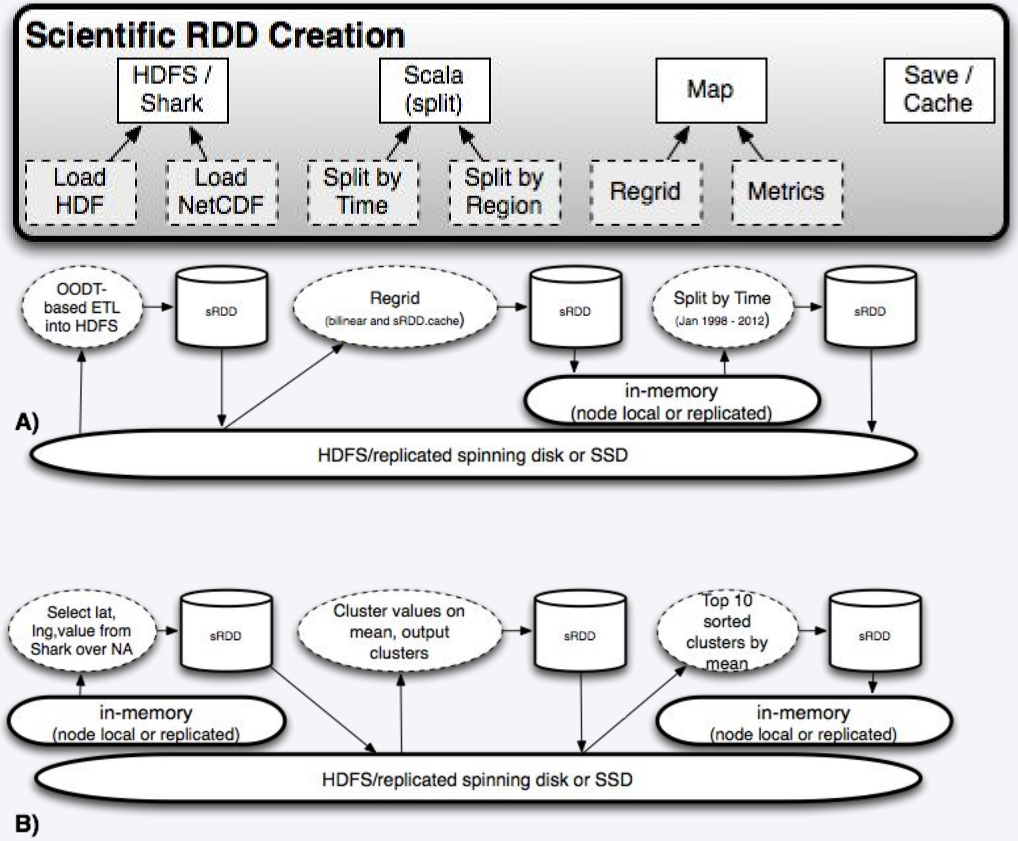
Create a scientific partition aware dataset that could be:

1. ETL'ed from HDF, NetCDF
2. Split by Time/Region
3. Mapped e.g., in Regrid, or Metric
4. Save/Cache'd

Example Scenarios

Two scenarios demonstrating intelligent data caching and access in SciSpark. A) a multi-stage operation to generate a time split, B) a multi-stage operation to select data from Shark (now SparkSQL), and cluster by deviation from mean.

Automatically capture: **provenance**

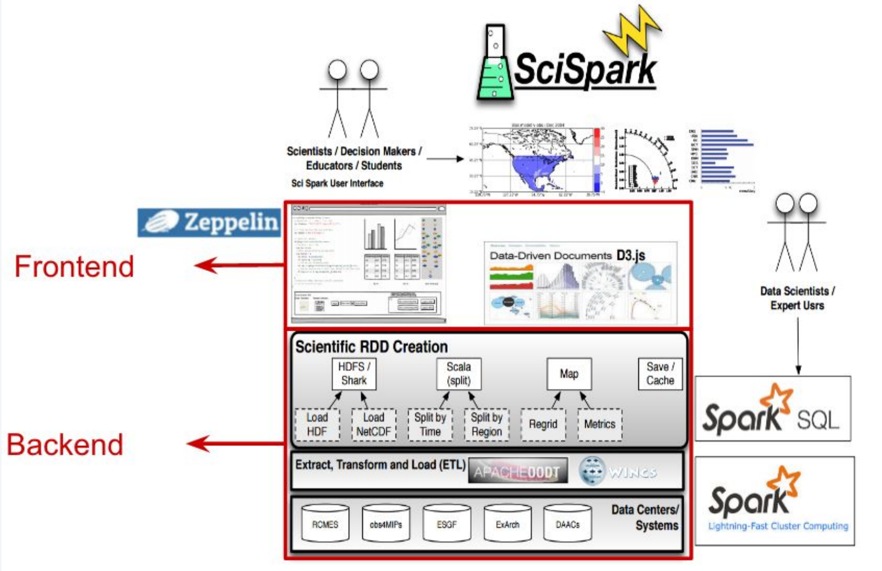

Scientific RDD Creation
HDFS / Shark
Scala (split)
Map
Save / Cache
Load HDF
Load NetCDF
Split by Time
Split by Region
Regrid
Metrics
A)
OODT-based ETL into HDFS
sRDD
Regrid (bilinear and sRDD.cache)
sRDD
Split by Time (Jan 1998 - 2012)
sRDD
in-memory (node local or replicated)
HDFS/replicated spinning disk or SSD
Select lat, lng,value from Shark over NA
sRDD
Cluster values on mean, output clusters
sRDD
Top 10 sorted clusters by mean
sRDD
in-memory (node local or replicated)
in-memory (node local or replicated)
HDFS/replicated spinning disk or SSD
B)

Jet Propulsion Laboratory
California Institute of Technology

Why SciSpark?

To understand the climate variability and change, and extreme weather events we need to be able perform complex analysis on voluminous and heterogeneous highly-resolved datasets. Analysis involves:

- Acquiring and ingesting data
- Resolving profiles and point sources within spatial and/or temporal coverage
- Gridding the data (for analysis and/or comparison)
- Quantifying changes in resolutions according to regridding techniques Calculating a range of statistics
- Visualizing data at all steps
- Ensuring reproducibility and data provenance

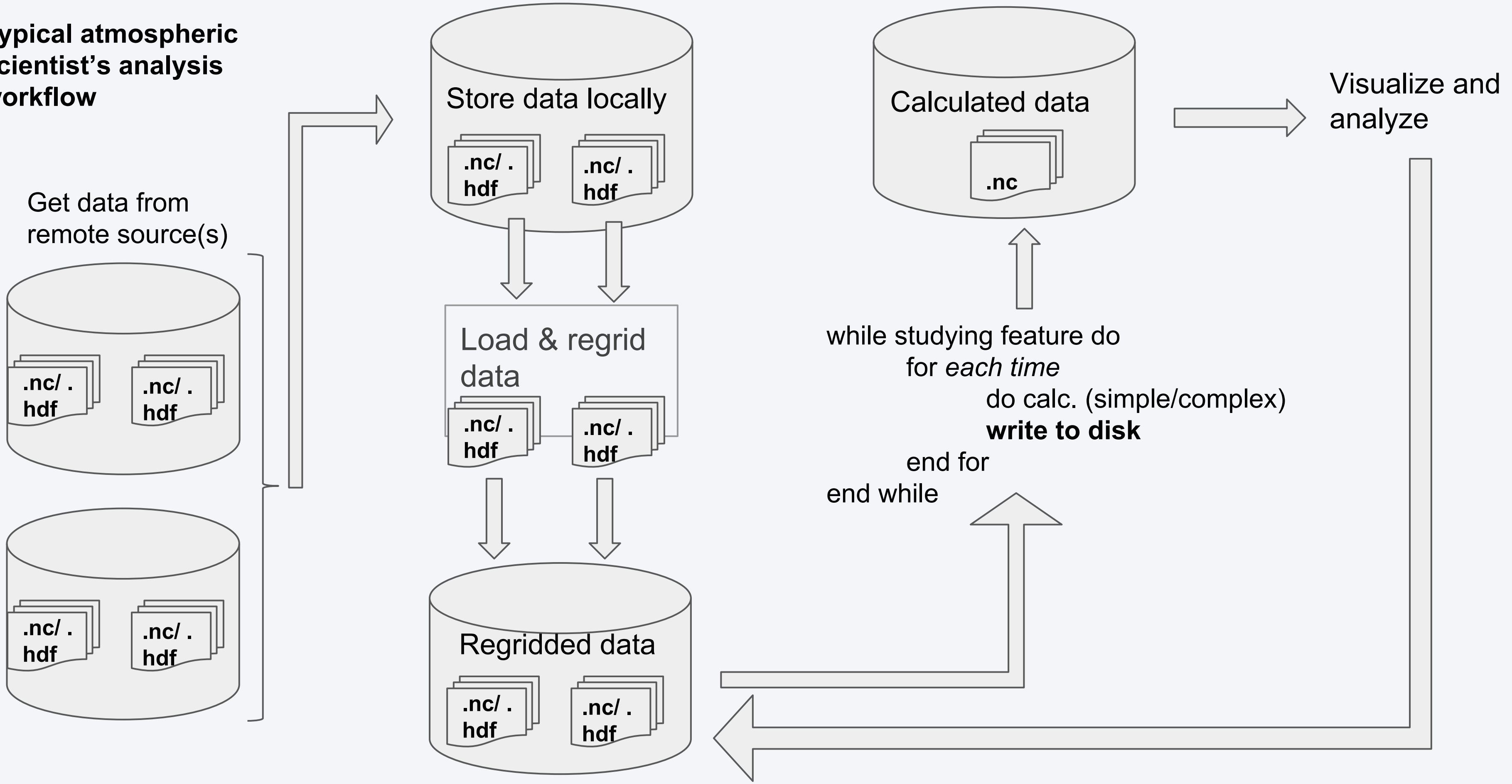


Work funded under 14-AIST-14-0034 - PI: Christian Mattmann

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

Typical atmospheric scientist's analysis workflow

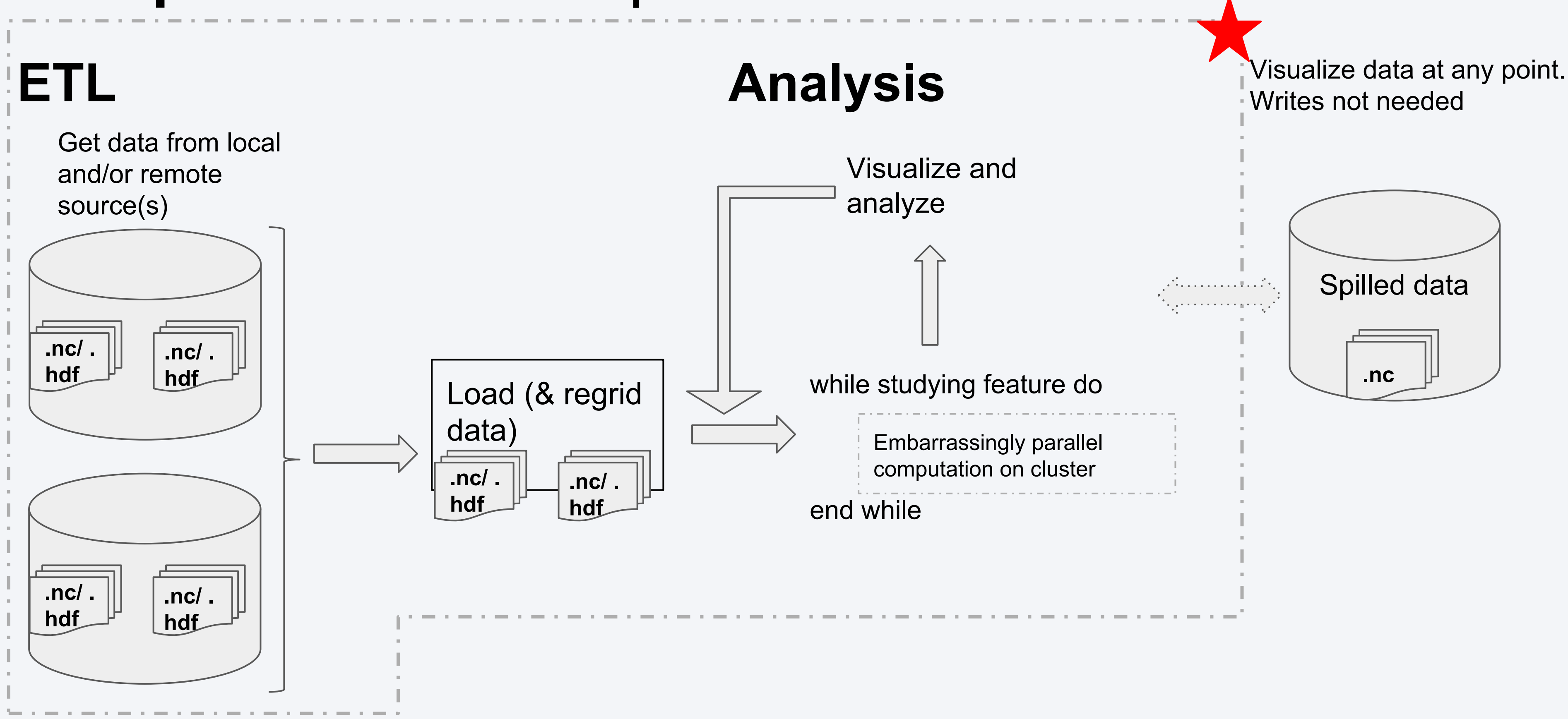


```
while studying feature do
        for *each time*
                do calc. (simple/complex)
                **write to disk**
        end for
end while
```

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

SciSpark & the atmospheric scientist's workflow



Challenges of handling Earth Science data in Hadoop environment

- Storage on the Apache Hadoop Distributed FileSystem (HDFS)
  - How to chunk up the file?
  - How to re-assemble it?
- How to access the data?
  - Loading **self describing, multidimensional & multivariable** data
  - How to keep that metadata?
  - How to extract variable from file?
  - Handling multidimensional data as Hadoop env is Java based
- How to compute on the data?
  - Matrix and scalar operations
  - Basic, complex statistics and user-defined algorithms

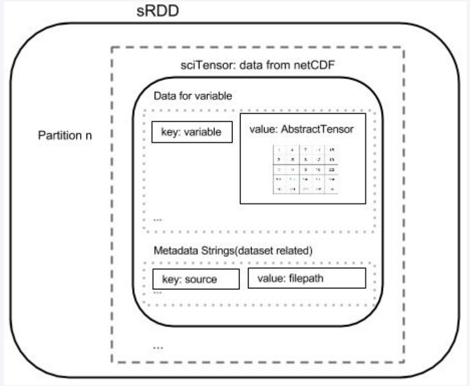
Not **SciSpark**'s focus
See: SciHadoop, sciDB

The sRDD

- The sRDD is an immutable, partitioned collection of ***sciTensor*** elements, that can be operated on in parallel within the cluster environment.
- The sRDD ***extends*** the RDD i.e. it has all the non-private variable & methods available to the RDD e.g. getPartitions, persist, transformations: map, filter, etc.
- All transformations in the sRDD are ***lazy***, which helps to optimize the disk and memory usage of the cluster.

The sciTensor

- The sciTensor is the datatype used in SciSpark to read in data from hierarchical data formats such as netCDF. The sciTensor:
  - is a self-documented array
  - keeps a list of variable arrays
  - keeps metadata in a hashmap
  - users can query the hashmap and perform matrix operations



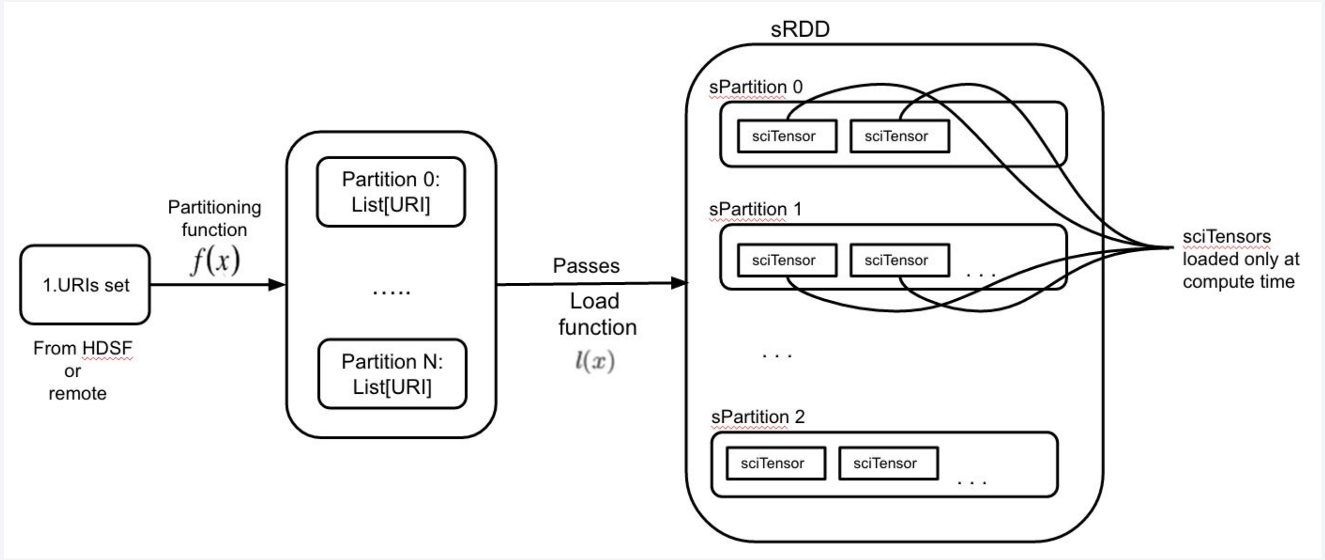
- The sciTensor is required in the SciSpark environment because:
  - SciSpark/ Spark is built on the Hadoop environment that is Java based, which inherently does not handle multidimensional arrays well (as compared to C, Fortran, Python)
- The sciTensor is read into the sRDD (i.e. the memory of the cluster) and the data within is operated on via arithmetic and relational operations.

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

Loading data into the sciTensor

- Data can be loaded into the sciTensor from:
  - Apache Hadoop Distributed FileSystem (HDFS)
  - OpenDap server
  - Local Filesystem



Notebook 201-1:

Using SciTensor and sRDD

- Loading data from HDFS
- Using relational operators
- Using arithmetic operators
- Parallel graphic generation
- Visualization

Notebook 201-2:

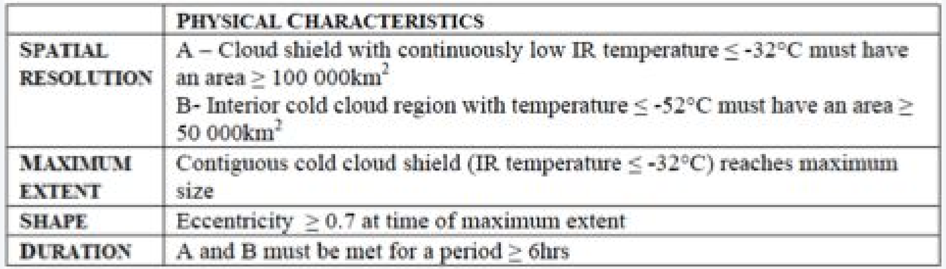
More SciSpark API & LazyEvaluation

- Explaining lazy evaluation
- Loading data from OpenDap

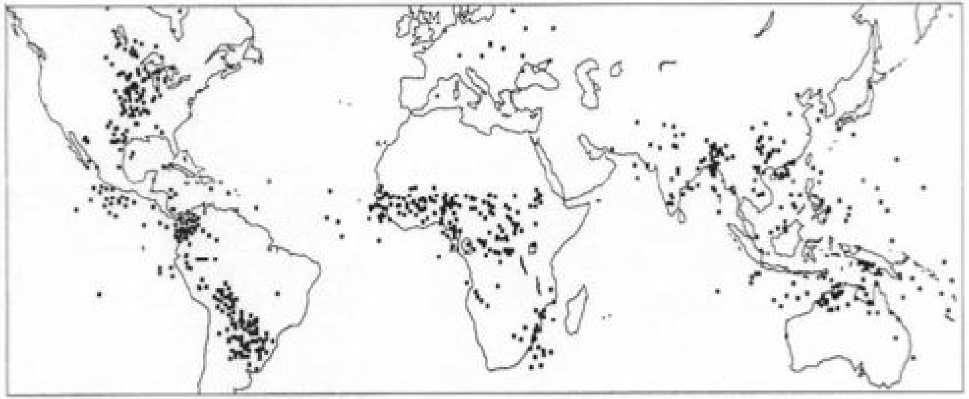
Jet Propulsion Laboratory
California Institute of Technology

SciSpark

SciSpark usecase: Mesoscale convective complexes

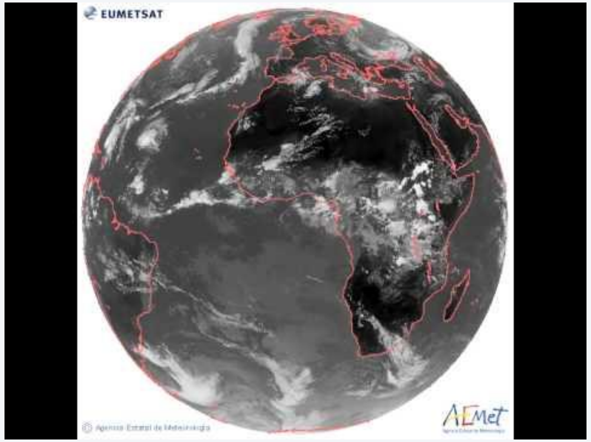


	PHYSICAL CHARACTERISTICS
SPATIAL RESOLUTION	A – Cloud shield with continuously low IR temperature ≤ -32°C must have an area ≥ 100 000km^2 B- Interior cold cloud region with temperature ≤ -52°C must have an area ≥ 50 000km^2
MAXIMUM EXTENT	Contiguous cold cloud shield (IR temperature ≤ -32°C) reaches maximum size
SHAPE	Eccentricity ≥ 0.7 at time of maximum extent
DURATION	A and B must be met for a period ≥ 6hrs



2. Locations of MCCs based upon 1-3 year regional samples of satellite imagery (from Laing and Fritsch, 1997). Locations are shown for the time of maximum extent of the cold cloud shield. Additional data concerning the samples is given in Appendix A.

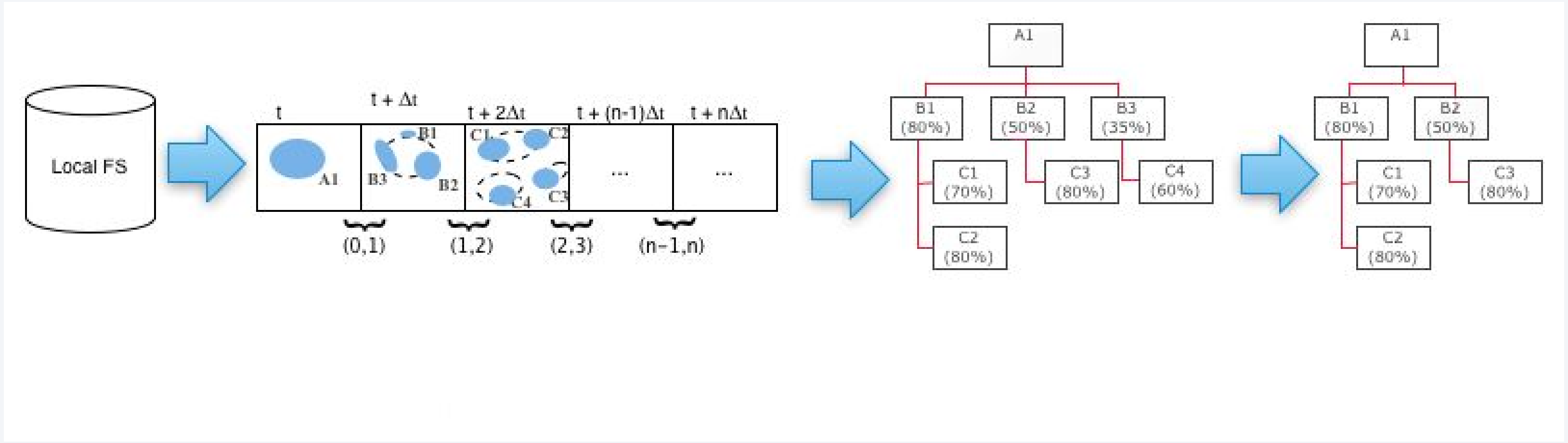
- A subset of mesoscale convective systems (MCSs)
- More than 700 MCCs occur globally, with 66% in the NH
- Maddox 1980 criteria for identification



Jet Propulsion Laboratory
California Institute of Technology

SciSpark

Finding MCSs



- Uses brightness temperature data.
- Nodes are areas with a given brightness temperature value, and of a given size.
- Edges are determined by area overlaps between nodes within consecutive time periods.

Finding cloud elements and overlaps

- Identify areas with a given brightness temperature value, and of a given size - nodes (in a parallelized manner)
- Edges are determined by area overlaps between nodes within consecutive time periods (this can also be parallelized in time).

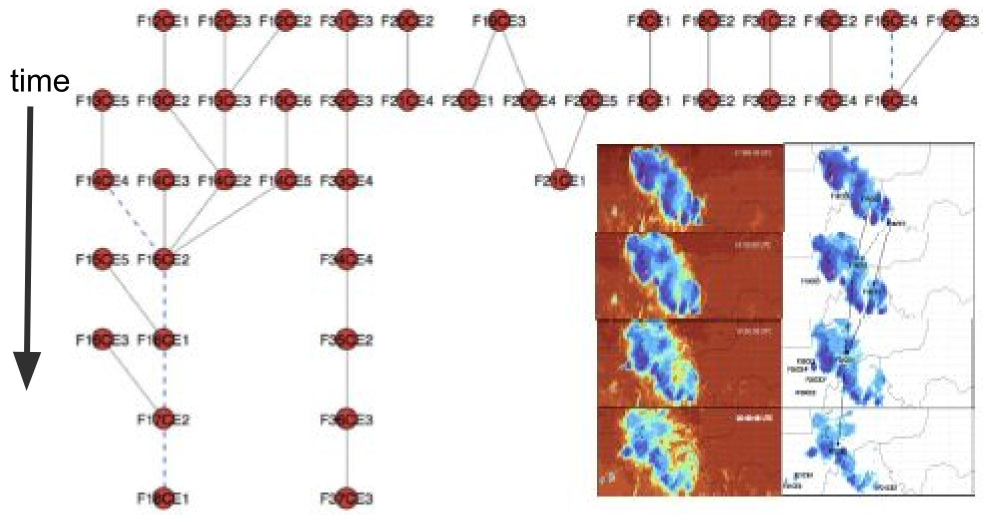
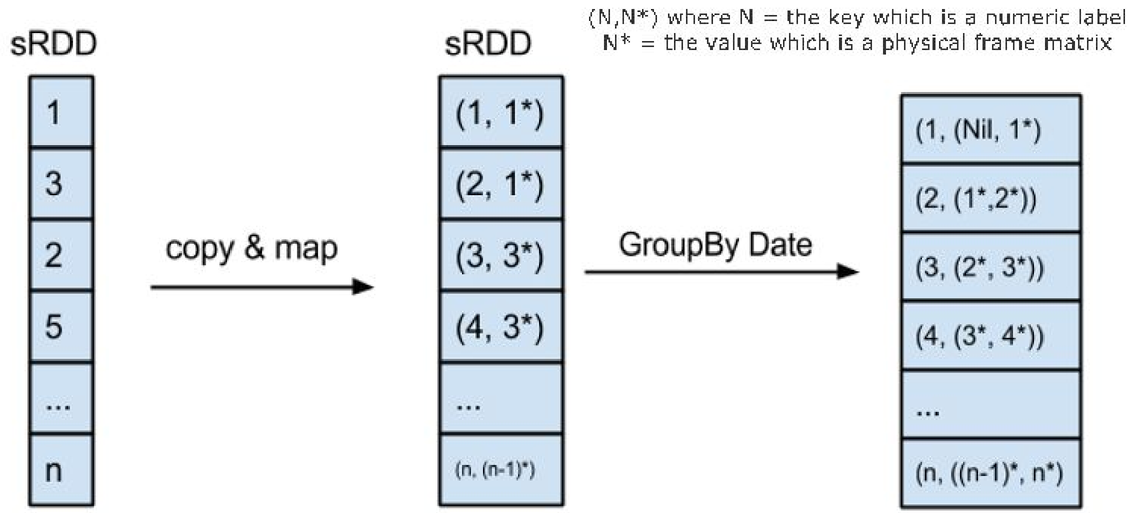


Figure 3. Graph construction from Whitehall et al. [26] GTG algorithm for MCC identification over Niamey, Niger.

Finding graph edges between two time epochs

- Bullet
-



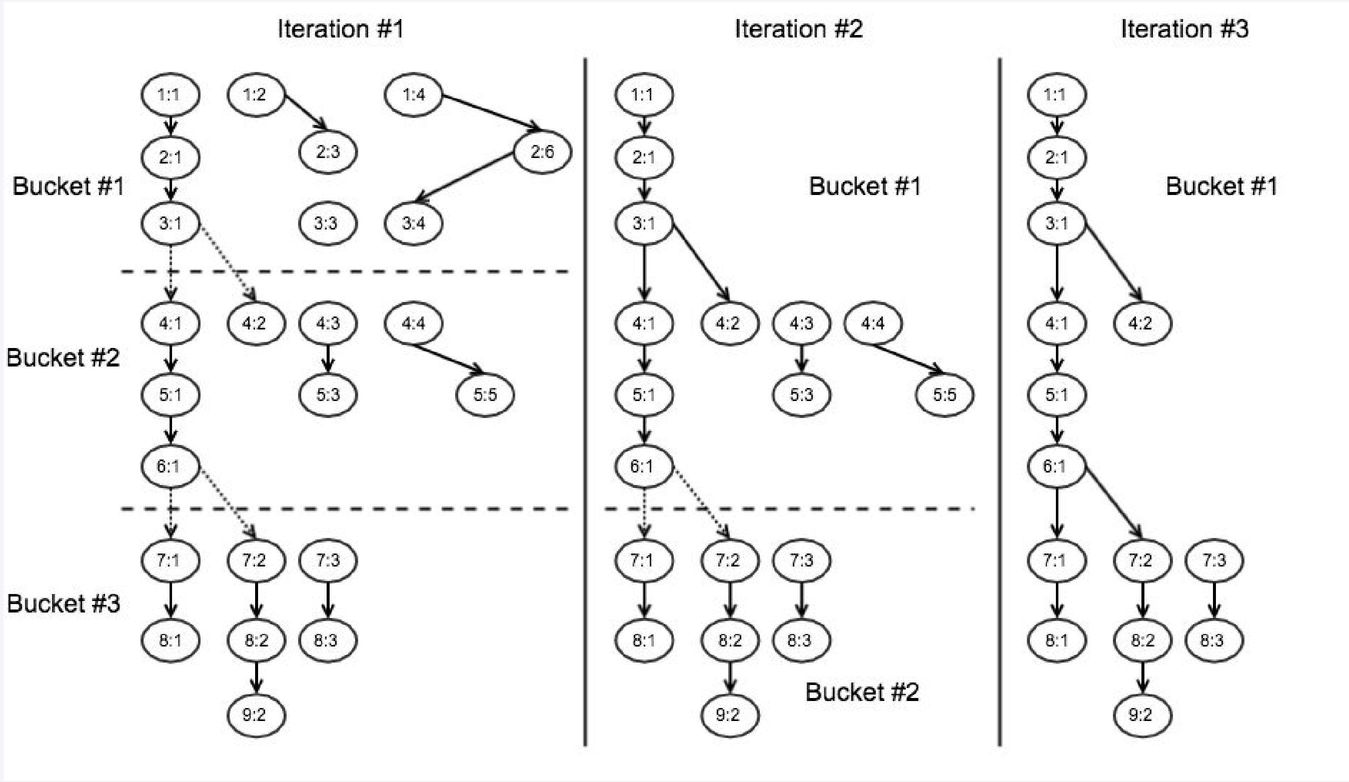
```
RDDDoubleKey = InputRDD.flatMap(p => List((p_f , p), (p_f + 1, p)))
ConsecutivePairRDD = RDDDoubleKey.GroupByKey
ConsecutivePairRDD.map((A, B) => {
      ...
})
```

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

Traversing graph by recursive merge

- Smartly partition: nodes with greatest connectivity
- For each partition: eliminate only “full” graphs
- Merge consec. partitions and repeat above step
- Iterate until one partition or nodes = 0



Complexities of this use case:

- Keeping time sequence in the sRDD: Our graph has an implicit time dependence, thus we need to consider the how the problem will be parallelized
- Keeping data & metadata flowing through the algorithm: As nodes are determined, there is some “irregular” subsetting of the initial dataset.
- Traversing the graph in the SciSpark engine. Our graph is:
    - Sparse, disconnected and directed.
    - There are no cycles

Jet Propulsion Laboratory
California Institute of Technology

SciSpark

Notebook 201-3:

End-to-end MCS part 1

- Loading data from HDFS
- Maintaining the time sequence
- Creating the graph object: Identifying cloud elements and edges
- Visualizing the cloud elements from the run leveraging SparkSQL

Notebook 201-4:

End-to-end MCS part 2

- Traversing the graph object to identify MCSs (i.e. subgraphs)
- Visualizing the graph and results (D3)

Notebook Exercises

- Try some of the exercises
- Just Play
- Ask Questions
- Notebook exercises:
    - 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