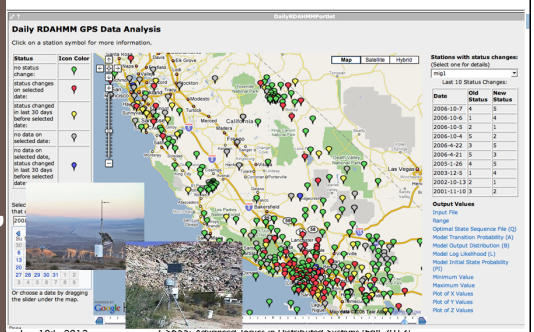
CS 655: Advanced Topics In Distributed Systems [Granules]

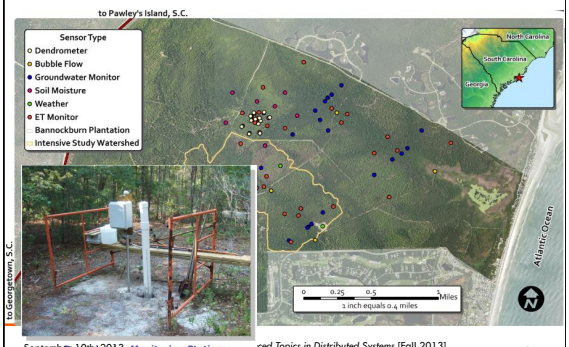
Shrideep Pallickara
Computer Science
Colorado State University

Projects utilizing NaradaBrokering

QuakeSim



Ecological Monitoring: PISCES



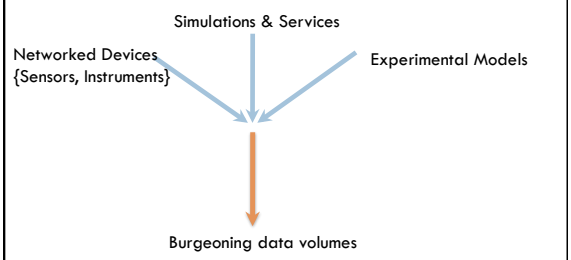
Projects utilizing NaradaBrokering

Sensor Grids

Lessons learned from multi-disciplinary settings

- Framework for processing streaming data
- Compute demands will outpace availability
- Manage computational load transparently

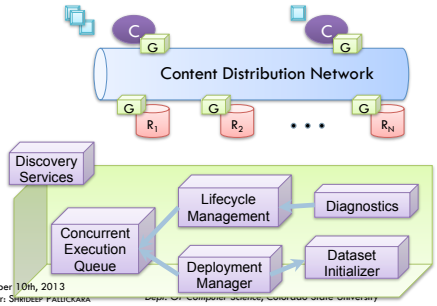
Big Picture



Fueled the need for a new type of computational task

- Operate on dynamic and voluminous data
- Comparably smaller CPU-bound times
  - Milliseconds to minutes
- BUT concurrently interleave 1000s of these *long running* tasks
- Granules provisions this

Granules is dispersed over, and permeates, distributed components



An application is the sum of its computational tasks that are …

- Agnostic about the resources that they execute on
- Responsible for processing a subset of the data
  - Fragment of a stream
  - Subset of files
  - Portions of a database

Granules does most of the work for the applications except for …

① Processing Functionality
② Specifying the Datasets
③ Scheduling strategy for constituent tasks

Granules processing functionality is domain specific

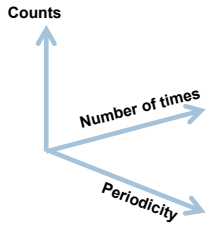
- Implement just one method: `execute()`
- Processing **1 TB** of data over 100 machines is done in **150 lines** of Java code.

Computational tasks often need to cope with multiple datasets

- TYPES: Streams, Files, Databases & URIs
- INITIALIZE: Configuration & allocations
- ACCESS: Permissions and authorizations
- DISPOSE: Reclaim allocated resources

Computational tasks specify their lifetime and scheduling strategy

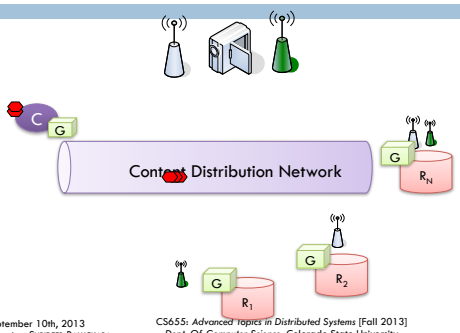
- **Combine** along any of these dimensions
- **Change** during execution
- **Assert** completion



Granules discovers resources to deploy computational tasks

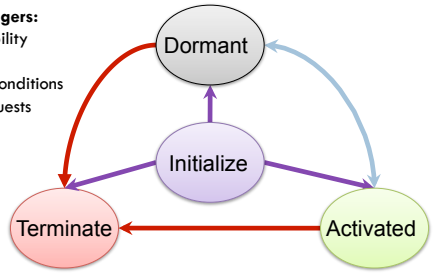
- **Deploy** computation instance on multiple resources
- **Instantiate** computational tasks & execute in Sandbox
- **Initialize** task's STATE and DATASETS
- **Interleave** multiple computations on a given machine

Deploying computational tasks

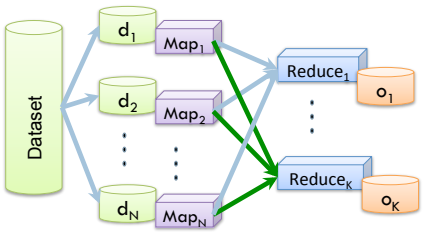


Granules manages the state transitions for computational tasks

Transition Triggers:
- Data Availability
- Periodicity
- Termination conditions
- External Requests



Map-Reduce enables concurrent processing of large datasets



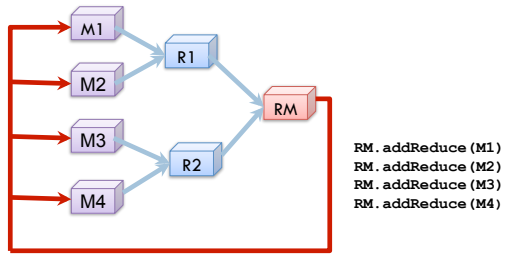
Substantial benefits can be accrued in a streaming version of Map-Reduce

- File-based = Disk IO → **Expensive**
- Streaming is much **faster**
  - Allows access to intermediate results
  - Enables time bound responses
- Granules Map-Reduce based on streams

In Granules MAP and REDUCE are two roles of a computational task

- **Linking** of MAP-REDUCE roles is easy
  - `M1.addReduce(R1)` or `R1.addMap(M1)`
  - Unlinking is easy too: `remove`
- Maps **generate** result streams, which are consumed by reducers
- Reducers can track outputs from Maps

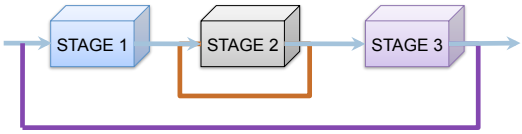
In Granules Map-Reduce roles are interchangeable



Scientific applications can harness Map-Reduce variants in Granules

- Iterative/Recursive: Fixed number of times **Or** Till termination condition
- Periodic
- Data Availability Driven

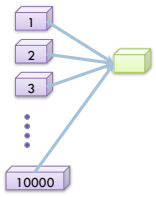
Complex computational pipelines can be set up using Granules



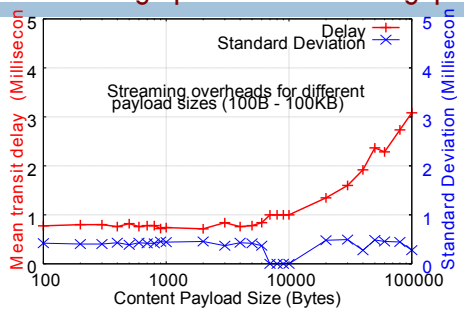
- Iterative, Periodic, Recursive & Data driven
- Each stage could comprise computations dispersed on multiple machines

Granules manages pipeline communications complexity

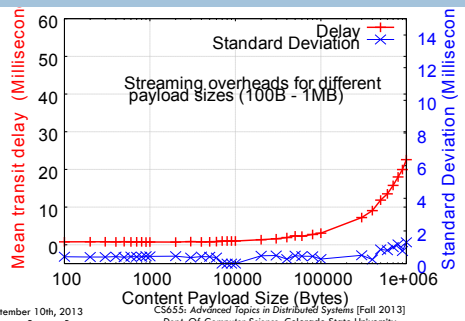
- No arduous management of **fan-ins**
- Facilities to **track** outputs
- **Confirm** receipt from all preceding stages.



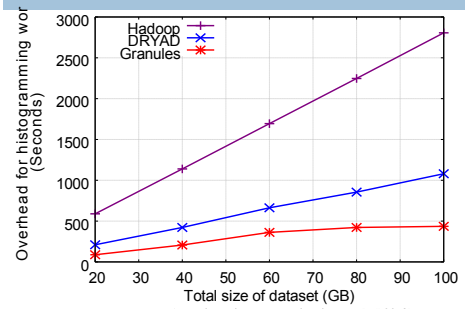
The streaming substrate provides consistent high performance throughput



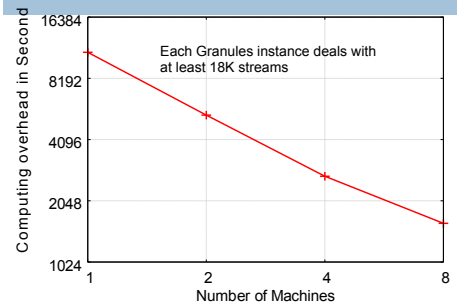
The streaming substrate provides consistent high performance throughput



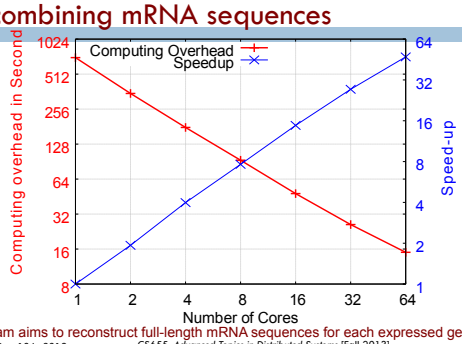
Granules outperforms Hadoop & Dryad in a traditional IR benchmark



Computing the product of two 16Kx16 K matrices using streaming datasets

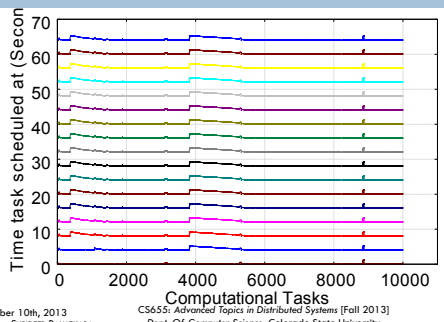


Maximizing core utilizations when combining mRNA sequences



Program aims to reconstruct full-length mRNA sequences for each expressed gene

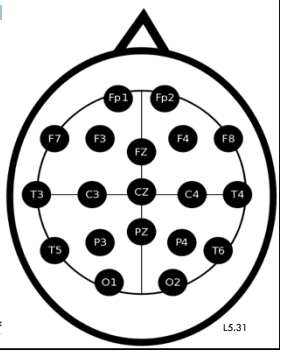
Preserving scheduling periodicity for 104 concurrent computational tasks



EEG Stream Processing

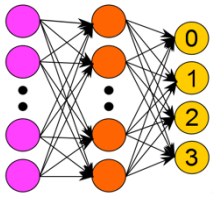
Electrode Placement

- Non-invasive
  - Cap holds electrodes to the scalp
- Electrode placement follows international 10-20 system of electrode placement



Artificial Neural Networks

- Number of input and output nodes are defined by the data
- Number of hidden units can vary
  - More hidden units can model more complex data
  - More hidden units take longer to train
- Weights are added between input and hidden and hidden and output layers



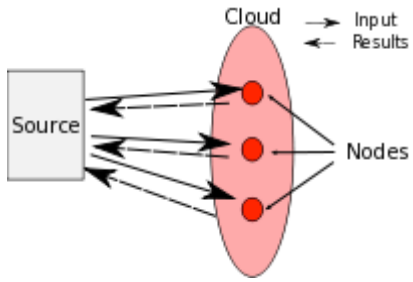
Approach

- R backend
  - Optimized for matrix multiplication
  - Existing code available for EEG manipulation, as well as neural network code
- Group of experts approach
  - Fits the map reduce framework
    - Mappers classify
    - Reducer produces expert opinion
- 3 sets of experiments:
  - Baseline times in R
  - Cloud communication overhead with Snowfall
  - Cloud and bridge communication overhead with Granules and JRI

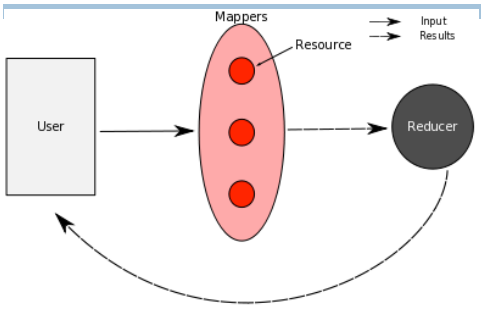
Frameworks Used

- Snowfall
  - Parallel computing package for R
  - Builds on the Snow package
  - Executes sequential code on multiple machines simultaneously
  - Does not require strong parallel computing background
- Granules
  - Lightweight cloud computing runtime
  - Java based
  - Allows user to specify run semantics – can enter a dormant state while waiting for more data to become available
- JRI
  - Java R Interface
  - Allows R computations to be run through Java
  - Communication is string-based

Snowfall Network Setup



Granules Network Setup



Baseline Results

Loading a single training set (200MB) in ms

Mean(ms)	Min(ms)	Max(ms)	SD(ms)
6581.602	6439.742	6822.34	101.376

Training a neural network from 4 training sets in ms

Mean(ms)	Min(ms)	Max(ms)	SD(ms)
194463.7	192433.3	197094.9	1300.87

Classification times with 1 neural net in ms

Stream Time	Mean(ms)	Min(ms)	Max(ms)	SD(ms)
5s	23.0432	22.17889	23.56791	0.4734237
1s	5.28194	4.909039	11.16085	0.8568976
250ms	1.710529	1.673937	1.926184	0.03777157

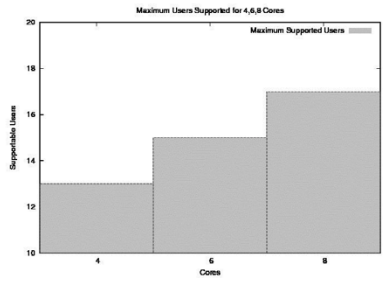
Snowfall and Granules Training Comparisons

NNs	Training Sets	Approach	Mean(ms)	Min(ms)	Max(ms)	SD(ms)
1	*1*	*Snowfall*	409860.5	403364.3	419965.7	4216.875
		Granules	313304.2	306402.0	329149.0	5141.031
3	*1*	*Snowfall*	462626.2	401475.9	483235.9	23512.64
		Granules	675968.8	610550.0	772679.0	52823.21
1	*4*	*Snowfall*	1001631	971224	1020680	17743.27
		Granules	1933540	1782531	2057664	110686.70
3	*4*	*Snowfall*	988410.4	964499	1023549	17513.37
		Granules	1964255.0	1779853	2131574	136452.10

Classification Times

Method	Stream Time	Mean(ms)	Min(ms)	Max(ms)	SD(ms)
Snowfall	5s	8884.60	8797.745	9069.47	85.82
Granules	5s	141.69	136.42	266.63	12.75
Snowfall	1s	5825.71	5815.38	5857.98	10.09
Granules	1s	93.16	47.51	492.68	32.13
Snowfall	250ms	2831.32	2830.38	2849.83	2.03
Granules	250ms	87.25	48.57	92.67	4.49

Maximum Supported Users on a Single Machine

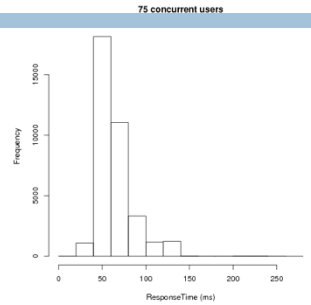


Scaling to multiple machines

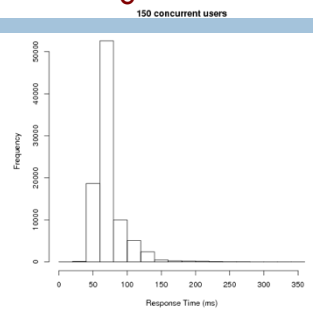
- Gathered statistics for classification on 5 and 10 machines
- Each machine supported 15 users
- While 17 users per 8-core machine could be supported, the network was swamped with 150 simultaneous users
  - 12MB/s 1GB/83s 1TB/23h

	Mean(ms)	Min(ms)	Max(ms)	SD(ms)
75 Users	64.33	21.69	268.30	20.51
150 Users	69.81	22.01	352.82	22.49

Stress Histograms – 75 users



Stress Histograms – 150 users



Clustering with Mahout in Granules

Clustering

- Unsupervised machine learning technique
- Finds logical 'clusters' to group data into
  - Not guaranteed to make sense to humans
- We work with a set of news articles
  - Reuters-21578 text categorization collection dataset
  - From the UCI Machine Learning Repository

Clustering

- Clustering is an iterative task
  - Start with a set of clusters
  - Put data in closest cluster
  - Modify cluster points to better match observed data
    - Repeat…

Clustering in Hadoop

- Mahout is a Java-based machine learning library built to work with Hadoop
  - We are only looking at the supported clustering algorithms
- Hadoop is execute-once and relies on disk to store state information
  - Since clustering usually runs for many rounds, there is a heavy disk I/O overhead

Clustering in Granules

- Granules allows computations to build state over multiple rounds of execution
  - Do not need to write to disk between iterations
    - Could vastly speed up Mahout runtimes
- Granules is also Java-based
  - We should be able to perform Mahout operations with little modification to boiler-plate code
    - Do not need to touch Mahout algorithms

Analyzing the Data

- Both Granules and Hadoop versions use the same data
  - Data is broken from text files into sequence files
- Produce vectors of normalized bigrams
  - With term frequency inverse document frequency (TF-IDF) weighting
  - Bigrams a sets of two adjacent words
    - There are 95,000+ unique bigrams in our data
  - TF-IDF helps to lower the importance of words which occur across all documents
    - He, She, The etc.
- 21,578 points to cluster in 95K+ dimensions

Clustering techniques

- K-Means
  - Most basic clustering algorithm supported by Mahout
- Fuzzy K-Means
  - Slightly more complex, allows data to belong to more than one group
- Dirichlet
  - No set k - it adds and removes clusters as it deems necessary
  - Can support data where clusters should not have a normal distribution
- Latent Dirichlet Allocation (LDA)
  - Similar to Dirichlet
  - Generative Model
  - Closest to Human Clustering

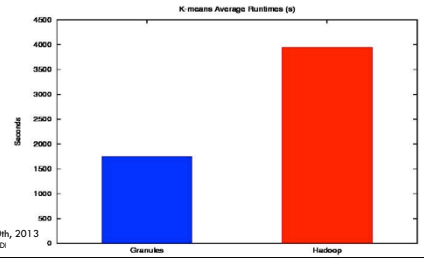
K-Means

- Fixed k, or number of clusters
  - Cluster centers are moved based on the previous iteration of clustering technique
- Larger numbers of clusters would mean finer-grained topics
- Good when you believe data is easily separable
  - An article only belongs to one broad topic
- May not return clusters that make sense to humans
  - It clusters across dimensions we don't take into account

K-Means Results

K-Means clustering in seconds

	Mean	Min	Max	SD
Granules	1749.76	1737.91	1772.52	13.696
Hadoop	3948.14	3922.475	3995.501	21.203



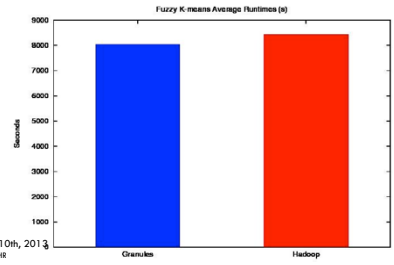
Fuzzy K-Means

- Builds off of K-means
  - Still a fixed k clusters
  - Now data can belong to multiple groups
- Can model more complex relations
  - From our data:
    - An article may discuss oil prices in the Middle East
    - K-means would cluster with either articles about the Middle East, or about prices of raw materials
    - Fuzzy K-means would be able to show this relationship to both
- Each data point receives a probability of belonging to every cluster
  - More time spent in CPU

Fuzzy K-Means Results

Fuzzy k-means clustering in seconds

	Mean	Min	Max	SD
Granules	8024.706	7970.56	8062.34	36.167
Hadoop	8423.78	8414.22	8431.15	5.812



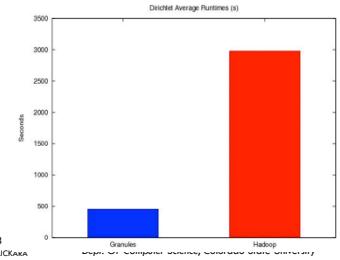
Dirichlet Clustering

- No set number of clusters
- Can support data which does not fit a normal model
- More complex that k-means or fuzzy k-means
  - Longer runtime
- If only run for a few iterations, it can be a good first-pass approach
  - Help to determine the k to give k-means or fuzzy k-means
  - Can explain why k-means is having trouble with the data

Dirichlet Clustering Results

Dirichlet Clustering in Seconds

	Mean	Min	Max	SD
Granules	456.78	437.61	481.64	10.474
Hadoop	2980.24	2830.94	3068.12	46.794



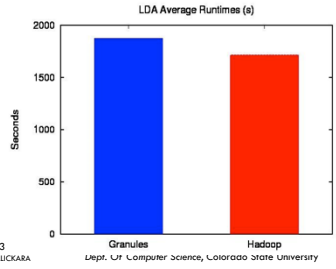
Latent Dirichlet Allocation (LDA)

- Similar to Dirichlet
- Generative model
  - Starts with a known model, and tweaks parameter to fit the model to the data
- LDA clusters documents by word according to "topics"
  - Like fuzzy k-means, a word belongs to all "topics" with a given probability
  - Document belongs to topics based off of the probabilities of all the words
- Uses a k
  - The number of topics
- Generated clusters make sense to humans
- Very CPU intense

LDA Results

LDA Clustering in Seconds

	Mean	Min	Max	SD
Granules	1873.91	1870.01	1878.26	3.057
Hadoop	1712.64	1682.19	1756.12	32.595



Key highlights of Granules

- **Easy** to develop applications
- Support for real-time **streaming datasets**
- Rich **lifecycle** and scheduling support for computational tasks.
- Enforces semantics of complex, distributed computational **graphs**

Conclusions

- **Complexity** should be managed BY the runtime, and NOT by domain specialists
- **Autonomy** of Granules instances allows it to cope well with resource pool dilations
- Provisioning lifecycle metrics for the parts makes it easier to do so for the sum