Data Clustering on the Parallel Hadoop MapReduce Model

Dimitrios Verraros

Overview

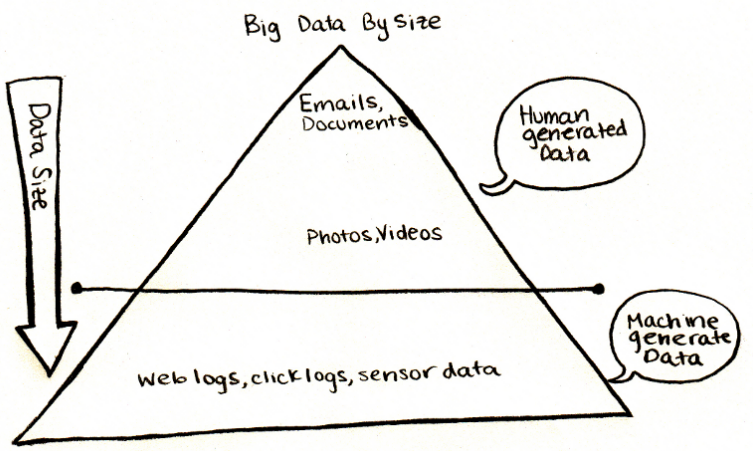
The purpose of this thesis is to implement and benchmark the performance of a parallel K-means clustering algorithm on the Apache Hadoop framework

- Data Clustering and K-means Algorithm
- Apache Hadoop Framework
- Parallel K-means implementation on the MapReduce model
- Experiment results

Motivation

Big Data explosion

- Vast amounts of data that needs to be processed and analyzed
  - Human generated data
  - Machine generated data



- Increased demand for efficient storage and processing solutions

Data Clustering

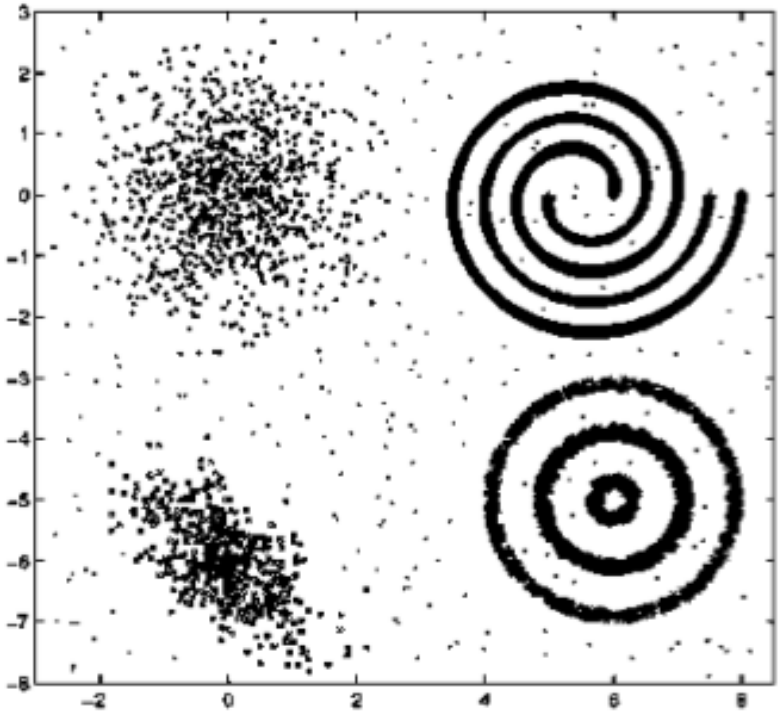
Machine Learning techniques focus on organizing data into sensible groupings

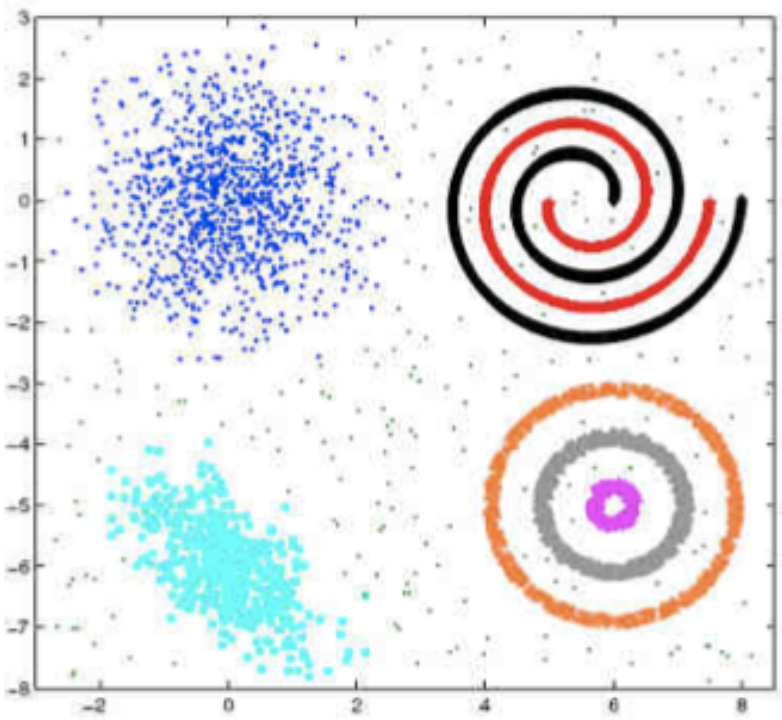
- Unsupervised learning
- Grouping of objects based on their relationship instead of preset tags or labels

Clustering Challenges

- Definition of Similarity
- Familiarity with the available data
  - Data preprocessing
- Cluster Diversity
- Data Representation
  - Number of Clusters
  - Data type
  - Data scale

Cluster Diversity





K-means Clustering

- One of the most popular machine learning algorithms utilized for clustering data sets
- Partitioning *n* observations into meaningful *k* **clusters**
- Target is to reduce the *mean* error of all observations to their respective cluster
- *Centroid* clustering
- *Hard* clustering

K-means Algorithm

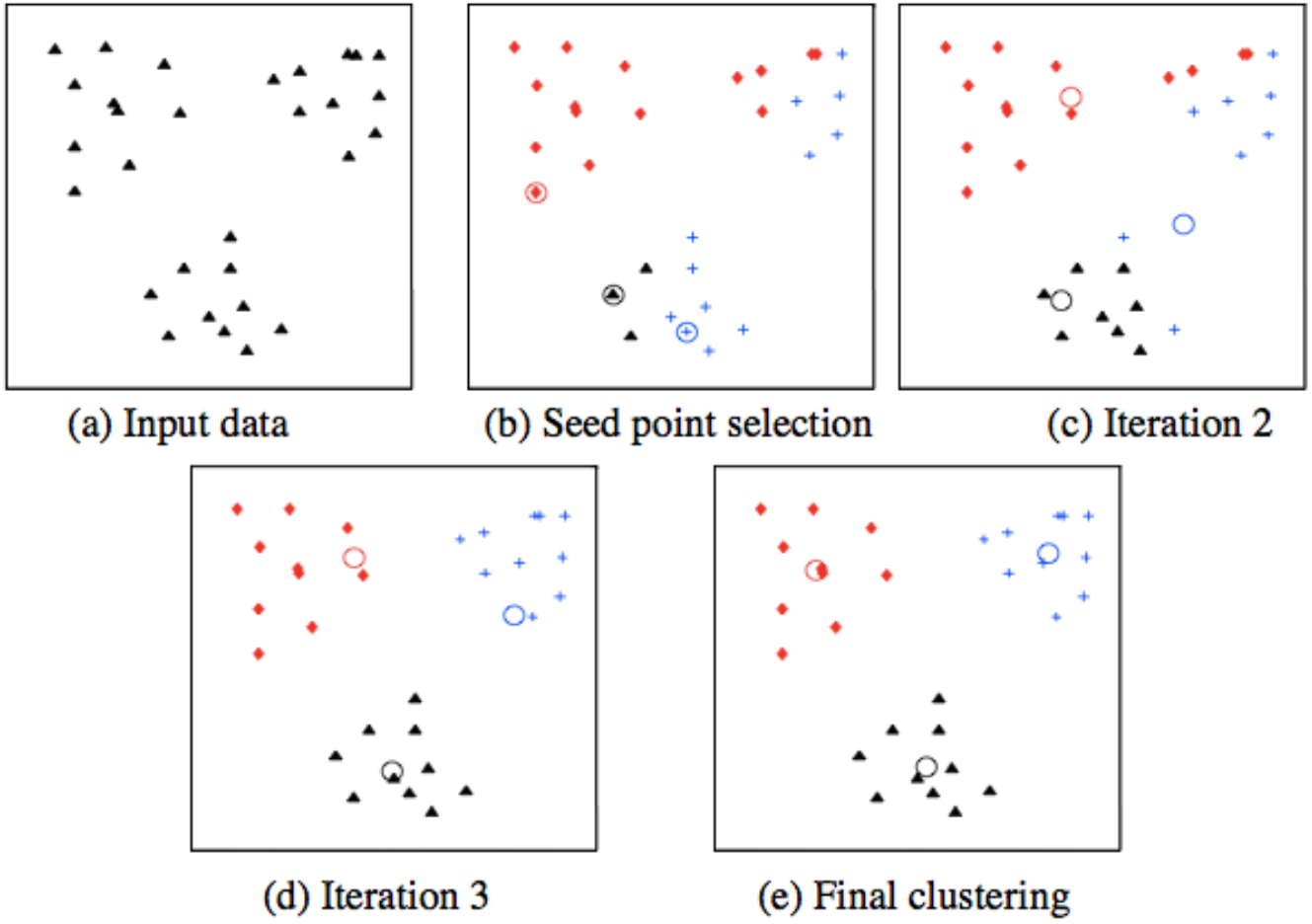
- Define the number of clusters K, and initialize them
- Repeat the following steps until the stopping condition is met
  - Assign every point* to its closer cluster
  - Recalculate cluster centers

$$e_k^2 = \sum_{i=1} \left(x_i^{(k)} - \mu^{(k)}\right)^2$$

$$E_K^2 = \sum_{k=1}^{K} e_k^2$$

*point: *n* dimensions vector

K-means Steps



Apache Hadoop

Open source framework for storage and large scale processing of data-sets on computer clusters

- Web-scale
  - terabytes to petabytes of data
  - hundreds to thousands of machines
- Used and developed by the industry leaders
  - Yahoo!
  - Facebook
  - Netflix

Hadoop Components

- Hadoop Common
  - tools, libraries and utilities used by the other Hadoop modules
- Hadoop Distributed File System – HDFS
- Hadoop YARN
  - Resource Manager
  - Task Manager
- Hadoop MapReduce Implementation
- Hadoop Ecosystem
  - underlying platform of many Apache “Big Data” projects

Hadoop HDFS

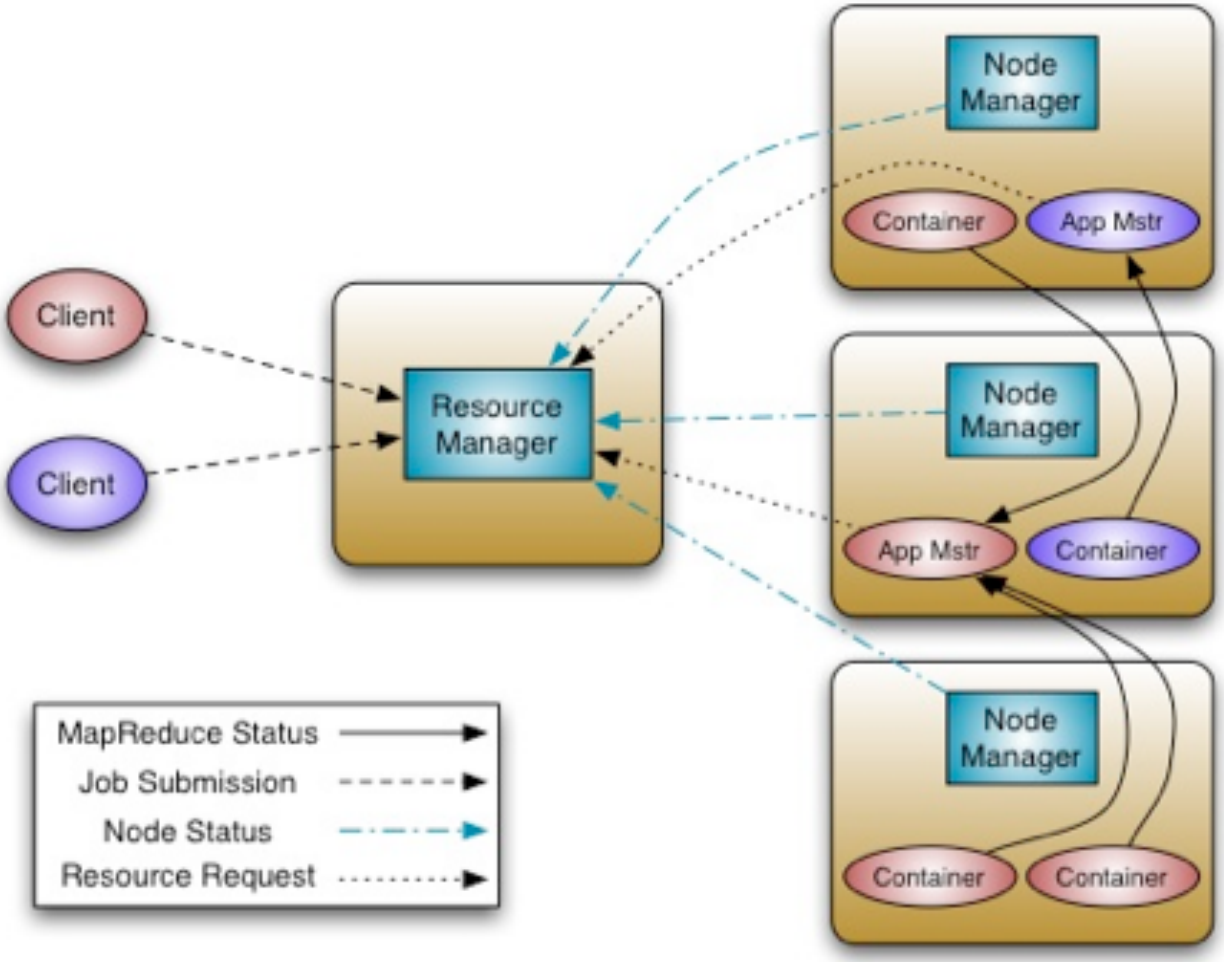
- Distributed File-system after Google's GFS
- Runs on commodity hardware
  - handles failures
- Master/Slave architecture
  - single point of failure?
- Designed for **BIG** files
  - 64 -128 MB default block size
- Replication across all nodes that serve as block pools

Hadoop YARN

Yet Another Resource Negotiator

- Separated from the MapReduce component since version 2
- Resource Management
- Monitoring
- Task Scheduling
- Job execution
- Failure Management

Hadoop YARN


Client
Client
Resource Manager
Node Manager
Container
App Mstr
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

Hadoop MapReduce

- Programming model for processing large data sets with a parallel, distributed algorithm on a cluster
- Implementation of the technology introduced by Google
- Inspired by *map* and *reduce* methods from functional programming
- Takes advantage of the locality of data
  - parallelism over data

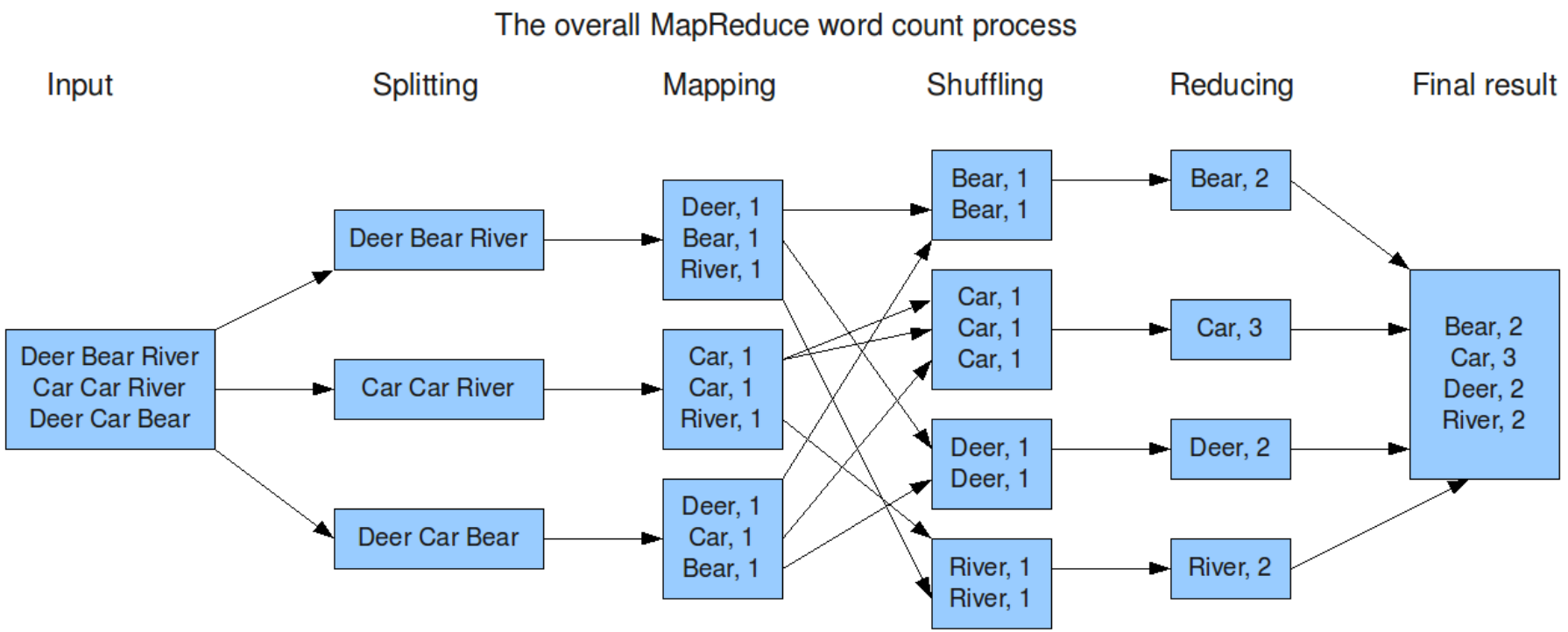
MapReduce

Can be seen as a two step process

- Map step
  - process the input data and extract them as <key, value> pairs

  (*combining* and *shuffling*)
- Reduce step
  - collect the outputs of the mappers, sorted and grouped by their key, and form the final output

WordCount Example



Apache Hadoop

It makes sense for **big** amounts of data

- the amount of mappers comes out by dividing the size of the file with the block size (64 -128 MB)
- every mapper must have enough work to do
- each node should have at least 10 mappers
- practically it makes no sense of processing something smaller than 1GB of data

Apache Hadoop

- Parallel Programming Challenging
- Higher Level Abstractions
  - programming interfaces
  - administration tools
  - data distribution
  - measures against race conditions
- Scalable
- Fault tolerant

K-means on MapReduce

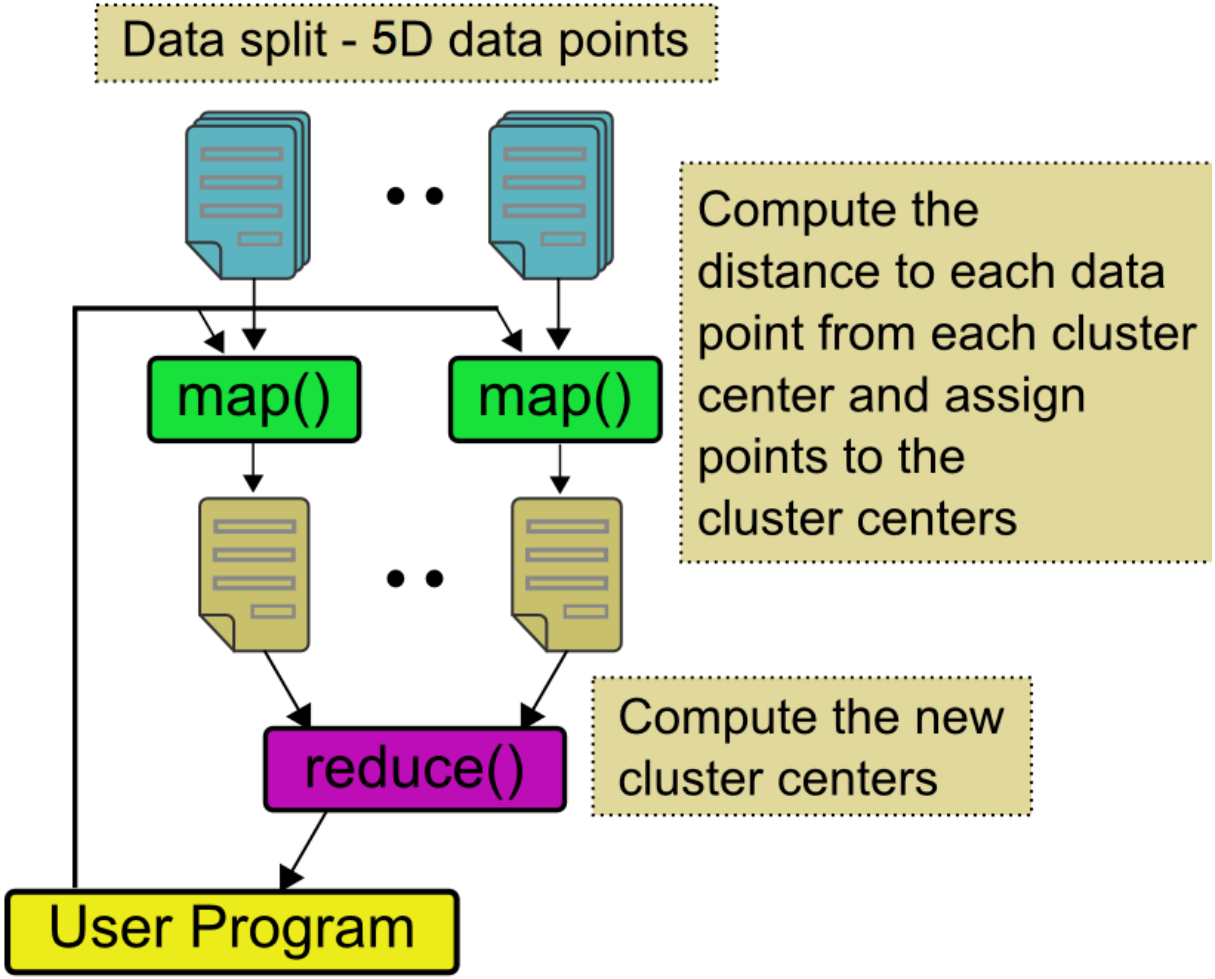
Parallel K-means

- Increased need to parallelize it
- Fits quite well the MapReduce model
  - map step - classification step (data parallel over points)
  - reduce step - recompute centers (data parallel over centers)

Experiment

- MLSP Amazon Access Data Set
  - more than 8 million rows
  - CSV format
- okeanos Cloud Infrastructure
  - Resource pool with 18 cores, 22 GBs of RAM and 260 GBs of HDD
  - Ubuntu Linux 12.04
  - Apache Hadoop 2.2.0
- 5-dimensional data points

K-means on MapReduce



Execution parameters

- Input path
- Sample size – number of data points
- Block size
  a. block size directly affects the number of splits and in turn the amount of mappers
  b. smaller block size => more blocks
  c. the number of blocks equals the number of splits and the number of mappers
- Number of clusters
- Number of iterations

Results

Master node with 1 core and 4 GBs of memory and
Slave node with 1 core and 2 GBs of memory

	Sample Size	Block Size	Clusters	Iterations	Mappers	Avg. Job Time	Total Time
Run 1	1 million	2 MB	100	5	30	5 min 2 sec	27 min 25 sec
Run 2	1 million	16 MB	100	5	4	2 min 36 sec	14 min 27 sec
Run 3	1 million	32 MB	100	5	2	2 min 9 sec	12 min 33 sec
Run 4	1 million	64 MB	100	5	1	1 min 56 sec	11 min 6 sec
Run 5	5 million	32 MB	500	5	10	4 min 34 sec	28 min 45 sec
Run 6	5 million	64 MB	500	5	10	5 min 13 sec	30 min 1 sec
Run 7	8 million	32 MB	500	5	16	6 min 23 sec	39 min 28 sec
Run 8	8 million	64 MB	500	5	8	6 min 7 sec	37 min 32 sec

Scaling up and out

Experiment on 3 nodes with increased resources

K-means on MapReduce

Considerations

- K-means is an iterative algorithm
  - every iteration is a new Job
  - Job initialization produces overhead
  - between every iteration all the data are written back to the disk and then read again
- Needs a LOT of data points (~10m per node)

Alternatives

- Apache Mahout
  - Machine Learning library on top of Hadoop
- Apache Spark
  - Engine for data processing – DAG model
- Apache Hama
  - BSP model
- Stratosphere
  - MapReduce extension

Conclusions

- K-means is a broadly adapted, easy to implement and quite efficient clustering algorithm
- Apache Hadoop is one of the best frameworks for distributed storage and parallel data processing
- MapReduce gets deprecated – better alternatives come up

Data Clustering on Hadoop

Questions