MapReduce for Statistical NLP/Machine Learning

Sameer Maskey

Week 7, October 17, 2012

Topics for Today

- MapReduce
- Running MapReduce jobs in Amazon Hadoop
- MapReduce for Machine Learning
  - Naïve Bayes, K-Means, Regression
- MapReduce with your code in Amazon Hadoop

Announcement

- Intermediate Report I due tonight (11:59 pm)
- Homework 2 due (date to be decided)
- Please use courseworks to discuss Homework 2
- Reading Assignment for next class
  - Neural Networks 5.1 to 5.3 in Bishop Book
- Graded Homework 1 will be available sometime next week

MapReduce

- Programming model to handle "embarrassingly parallel" problems
- Programmers specify two functions:

Map (k1, v1) $\rightarrow$ list(k2, v2)

Applied to all pairs in parallel for input corpus
(e.g. all document for example)

Reduce (k2, list(v2)) $\rightarrow$ list (v2)

Applied to all groups with the same key in parallel
(e.g. all words for example)

Components of MapReduce Program

- Driver
- Mapper
- Reducers
- May also contain
  - Combiners (often same as Reducers) and Partioners

Data to Mapper : InputFormats

- FileInputFormat
- TextInputFormat
  - Each \n terminated is treated as a value
  - Key is the byte offset
- KeyValueTextInputFormat
  - Map each \n terminated lines into
    - Key SEPARATOR Value format
- SequenceFileInputFormat
- OutputFormat can be set as well

Partioners and Combiners

- Partioners decide what key goes to what reducer
  - Default - hash-based
  - For our word count we can have N different reducers for N words
- Combiners
  - Combines mapper output before sending to reducer
  - Many times similar to reducer

Jobs and Tasks

- Job : user submitted map and reduce
- Task
  - Single mapper
  - Single Reducer
- Job-cooridnater
  - Manages task delegation, retries, etc
- Task tracker
  - Manages task in each slave nodes

Naïve Bayes Experiment from Homework 1

- wget http://www.cs.columbia.edu/~smaskey/CS6998-0412/homework/hw1.tar.gz
- We had small data set which we used to build Naïve Bayes Classifier
- Let us build Naïve Bayes for much larger corpus

Naïve Bayes Classifier for Text

$$P(Y = y_k|X_1, X_2, ..., X_N) = \frac{P(Y=y_k)P(X_1,X_2,..,X_N|Y=y_k)}{\sum_j P(Y=y_j)P(X_1,X_2,..,X_N|Y=y_j)}$$

$$= \frac{P(Y=y_k)\Pi_i P(X_i|Y=y_k)}{\sum_j P(Y=y_j)\Pi_i P(X_i|Y=y_j)}$$

$$Y \leftarrow argmax_{y_k} P(Y = y_k)\Pi_i P(X_i|Y = y_k)$$

Naïve Bayes Classifier for Text

- Given the training data what are the parameters to be estimated?

$P(Y)$	$P(X\|Y_1)$	$P(X\|Y_2)$
Diabetes : 0.8	the: 0.001	the: 0.001
Hepatitis : 0.2	diabetic : 0.02	diabetic : 0.0001
	blood : 0.0015	water : 0.0118
	sugar : 0.02	fever : 0.01
	weight : 0.018	weight : 0.008
	…	…

We Simply Want to Count

- To compute posteriors of P(C|wi) we prepared P(wi|C) for each class
- We compute P(wi|C) by “counting” # of documents the word wi occurred in and dividing by the total # of documents
- Can we put this in MapReduce Framework easily?

$$\frac{\#\mathrm{D}\{\mathrm{X}_i = x_{ij} \& Y = y_k\}}{\#\mathrm{D}\{\mathrm{Y}=\mathrm{y}_k\}}$$

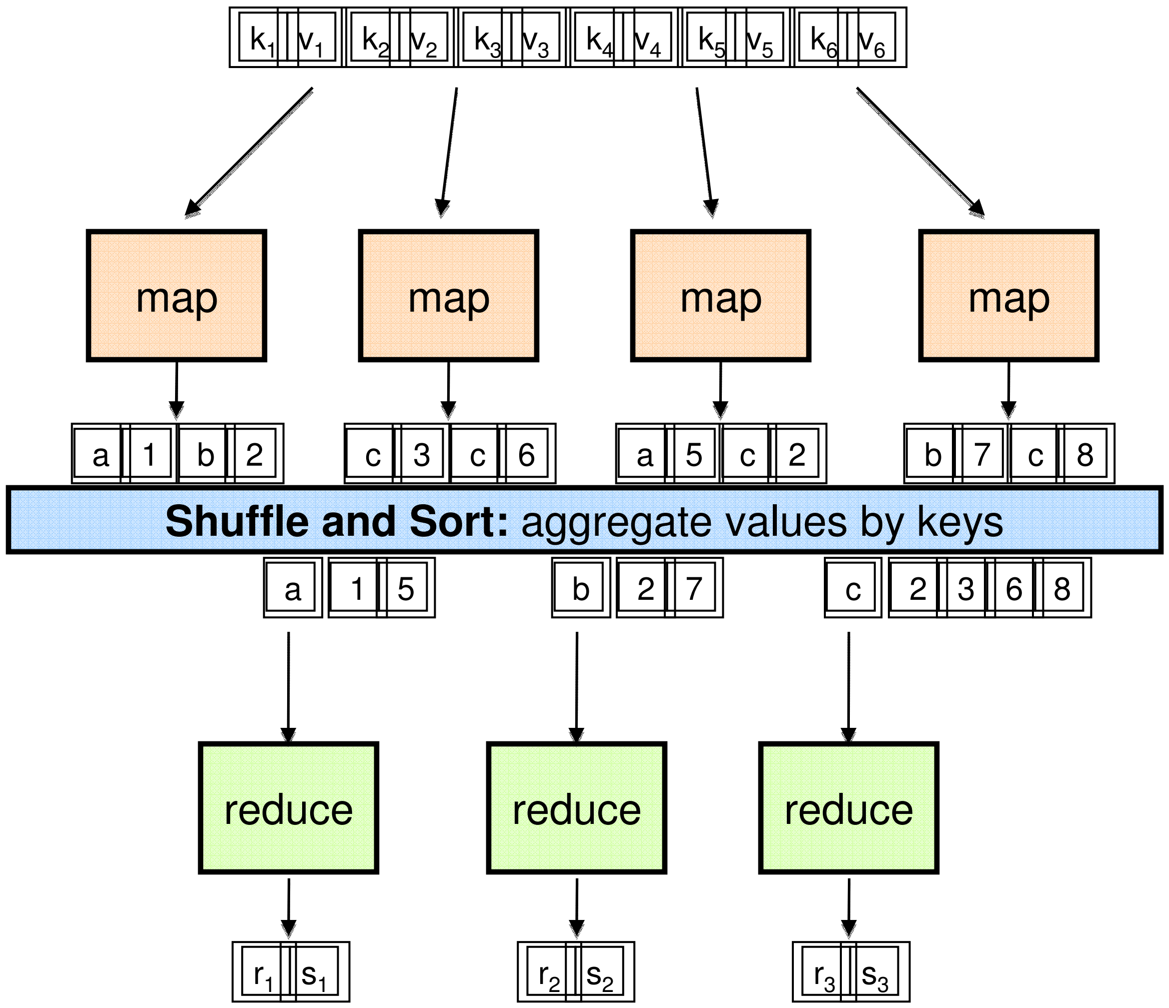
Naïve Bayes for 2 Billion Documents

- Assume you have 2 billion documents
- 1 billion in Hockey class
- 1 billion in Baseball class
- We want to compute numerator of the equation in the last slide
- We want to compute
  - Number of ‘x=1’ where x=1 if word x occurs in the document – number of documents that has word x
  - Multinomial Naïve Bayes Version : sum of ‘x=N’ where N is the number of times word occurs in a document
- What will be the Map function?
- What will be the Reduce function?

Remember our Word Count Example

- Last lecture we talked about word counts in MapReduce framework
- What will be Map and Reduce function for Wordcount Example?

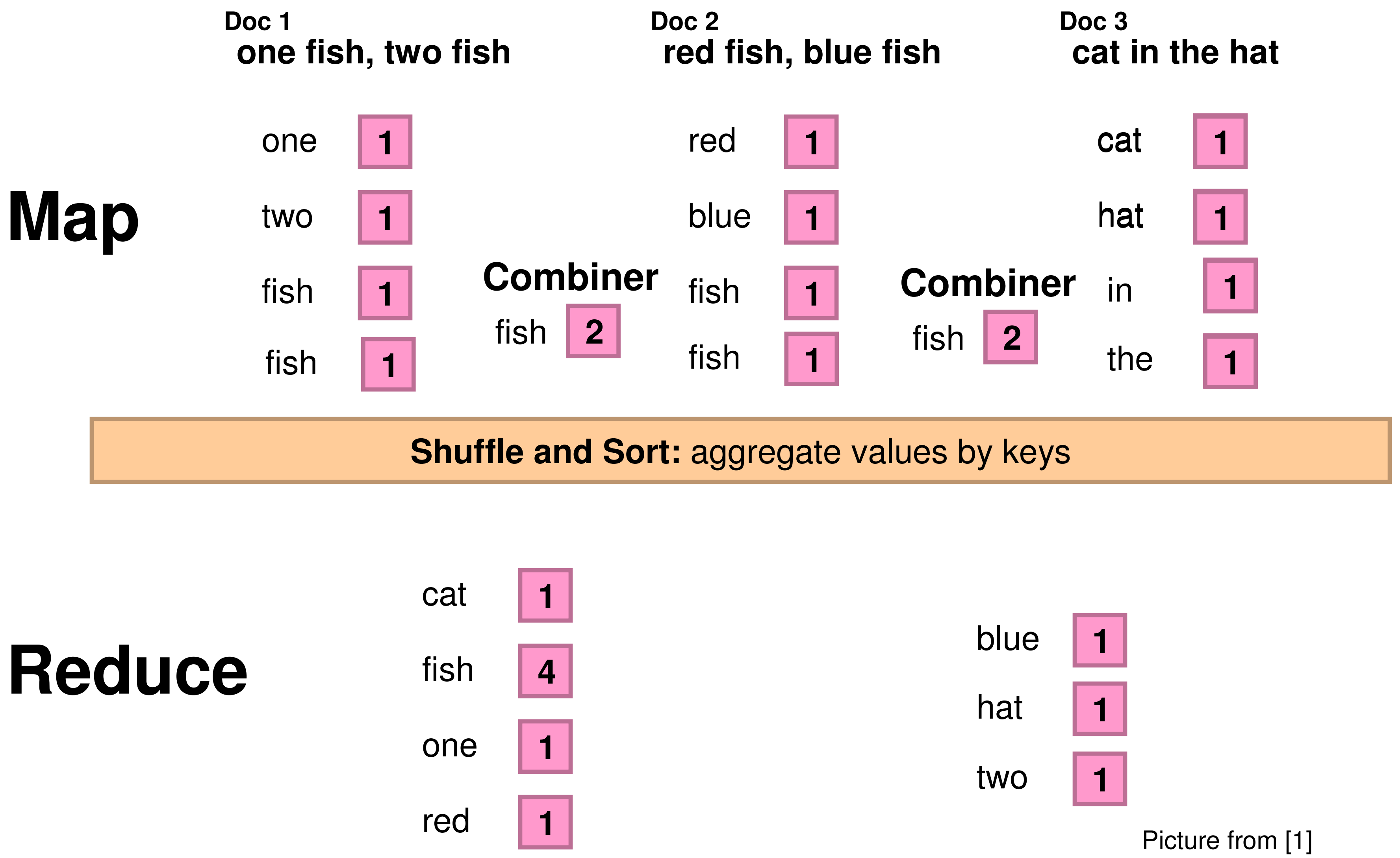
Remember MapReduce Picture



k_1 v_1 k_2 v_2 k_3 v_3 k_4 v_4 k_5 v_5 k_6 v_6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
r_1 s_1
r_2 s_2
r_3 s_3

Picture from [1]

MapReduce for Word Count



Picture from [1]

WordCount Example in AWS MapReduce

- Let’s try to run Amazon’s in built example for word count
- Login to your AWS accounts follow instructions in the class

Using AWS

- **https://stanlpcolumbia.signin.aws.amazon.com/console**

- UserID Password emailed to you
- Access Keys emailed you to as well

- LogIn
- Create Instance with LAMP stack AMI or whatever AMI you want
- Create certificates and store it in a safe place
- Access Virtual Server

- Create S3 bucket
- Try MapReduce Example

Setting up your AWS Machine

- We went through this last time
- Setup LAMP stack if you want
- Setup any other AMI you want
- Install any software you want
  - Ubuntu – apt-get
  - CentOS – yum

```
sudo su
yum install httpd
yum install mysql-server mysql
yum install php php-mysql
service httpd start
service mysqld start
/usr/bin/mysqladmin -u root password 'password'
```

Basic AWS EC2 setup

- Addusers, groups, etc
- Edit /etc/ssh/sshd_config setting where
  - PasswordAuthentication yes
- Restart the sshd
  - sudo service sshd restart
- Find online tutorials for more sysadmin related topics

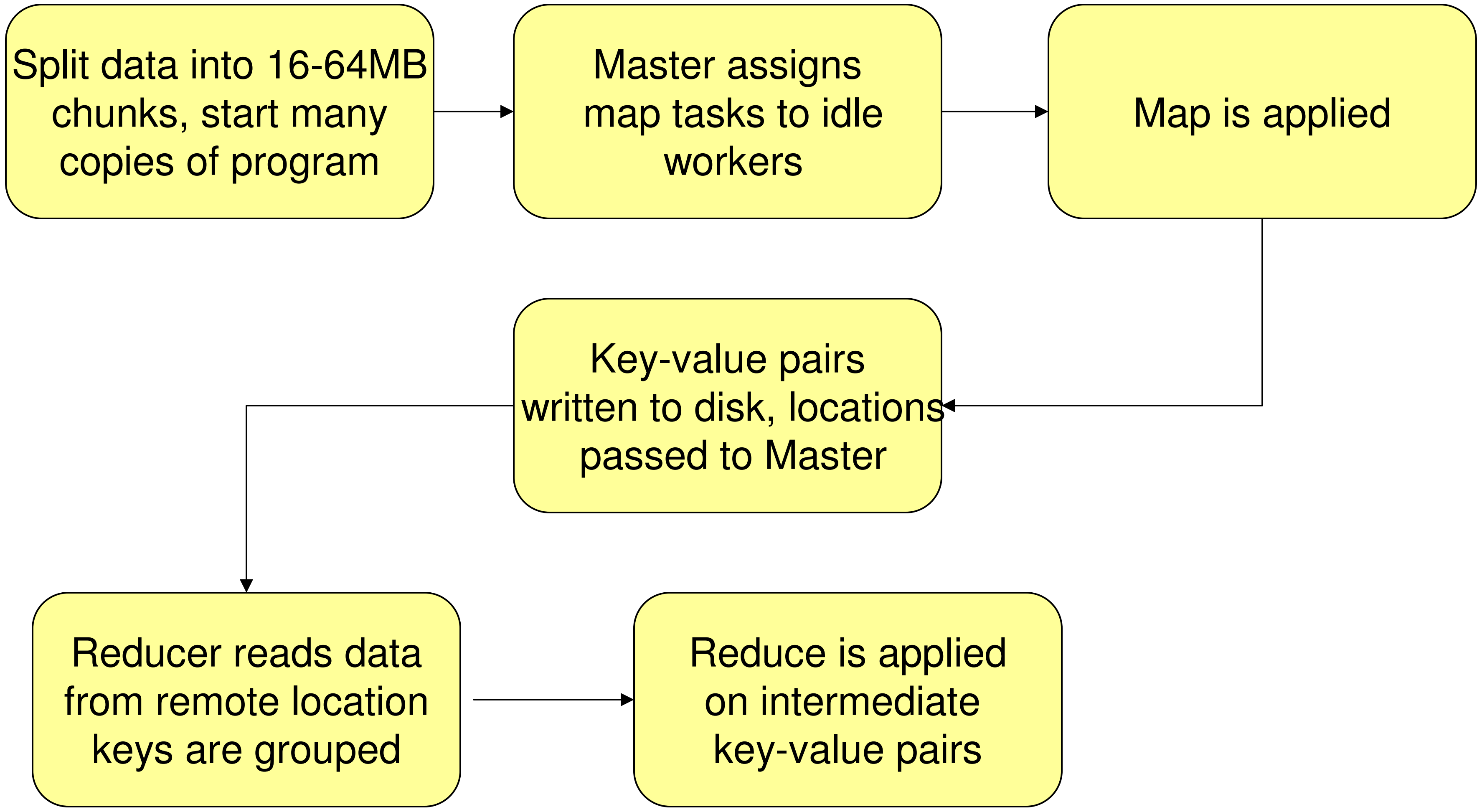
Setup S3

http://www.s3fox.net/DownloadPage.aspx
firefox plugin

```
wget http://s3tools.org/repo/CentOS_5/s3tools.repo
yum install s3cmd

s3cmd --configure
```

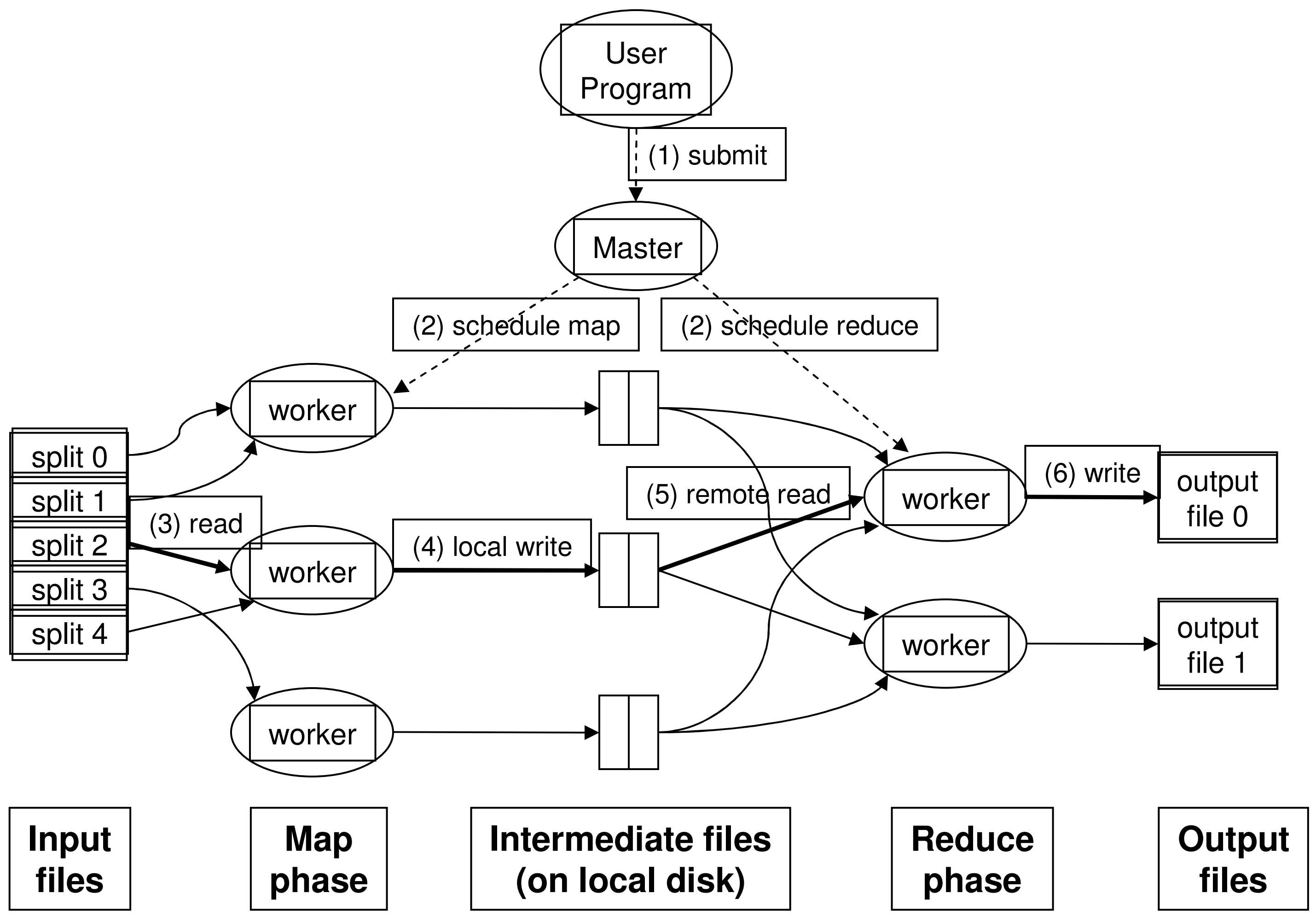
```
Enter new values or accept defaults in brackets with Enter.
Refer to user manual for detailed description of all options.

Access key and Secret key are your identifiers for Amazon S3
Access Key: XXXXXXXXXXXXXXXXXXXX          (enter your access key here)
Secret Key: SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS       (enter your secret key here)

Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3
Encryption password: PPPPPPPPPPP              (enter a password of your choice here)
Path to GPG program [/usr/bin/gpg]:

When using secure HTTPS protocol all communication with Amazon S3
servers is protected from 3rd party eavesdropping. This method is
slower than plain HTTP and can't be used if you're behind a proxy
Use HTTPS protocol [No]:

On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't conect to S3 directly
HTTP Proxy server name:
```

MapReduce Computation


Split data into 16-64MB chunks, start many copies of program
Master assigns map tasks to idle workers
Map is applied
Key-value pairs written to disk, locations passed to Master
Reducer reads data from remote location keys are grouped
Reduce is applied on intermediate key-value pairs

MapReduce Framework

- We just write mapper and reducer
  - Hand over many details to the framework
- How to assign workers to the map tasks?
- How to assign workers to the reduce tasks?
- Handle synchronization across many map tasks
- Handle failures, errors
- How to distribute data in the file system

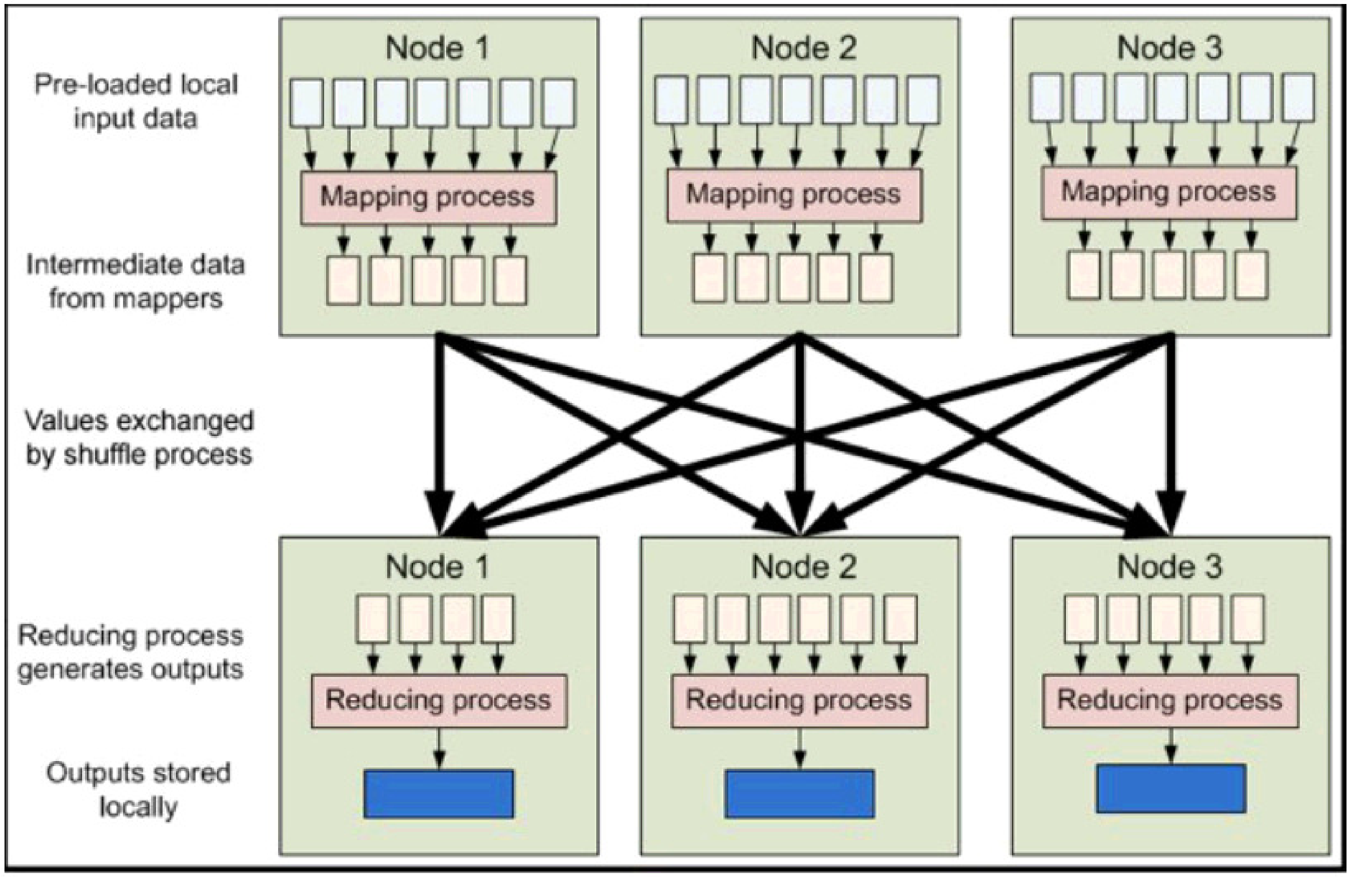


Adapted from (Dean and Ghemawat, OSDI 2004)

MapReduce Implementations

- Google has a proprietary implementation in C++
  - Bindings in Java, Python
- Hadoop is an open-source implementation in Java
  - Development led by Yahoo, used in production
  - Now an Apache project
  - Rapidly expanding software ecosystem
- Lots of custom research implementations
  - For GPUs, cell processors, etc.

Shuffle and Sort


Node 1
Node 2
Node 3
Pre-loaded local input data
Mapping process
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Reducing process
Outputs stored locally

Can we use locality information for Reducer?

How about Combiner?
Sum vs Mean?

Picture from [2]

Examples

- Distributed Grep
  - Map <line, 1>
  - Reduce <line, 1>
- URL Access Frequency Count
  - Map <URL, 1>
  - Reduce <URL, total count>
- Reverse Web-Link Graph
  - Map <target, source>
  - Reduce <target, list(source)>

Examples

- Term-Vector per Host
  - Map <hostdocment, tf vector>
  - Reduce <hostdocument, tf vector>
- Inverted Index
  - Map <word, document id>
  - Reduce <word, list<document id>

Distributed File System

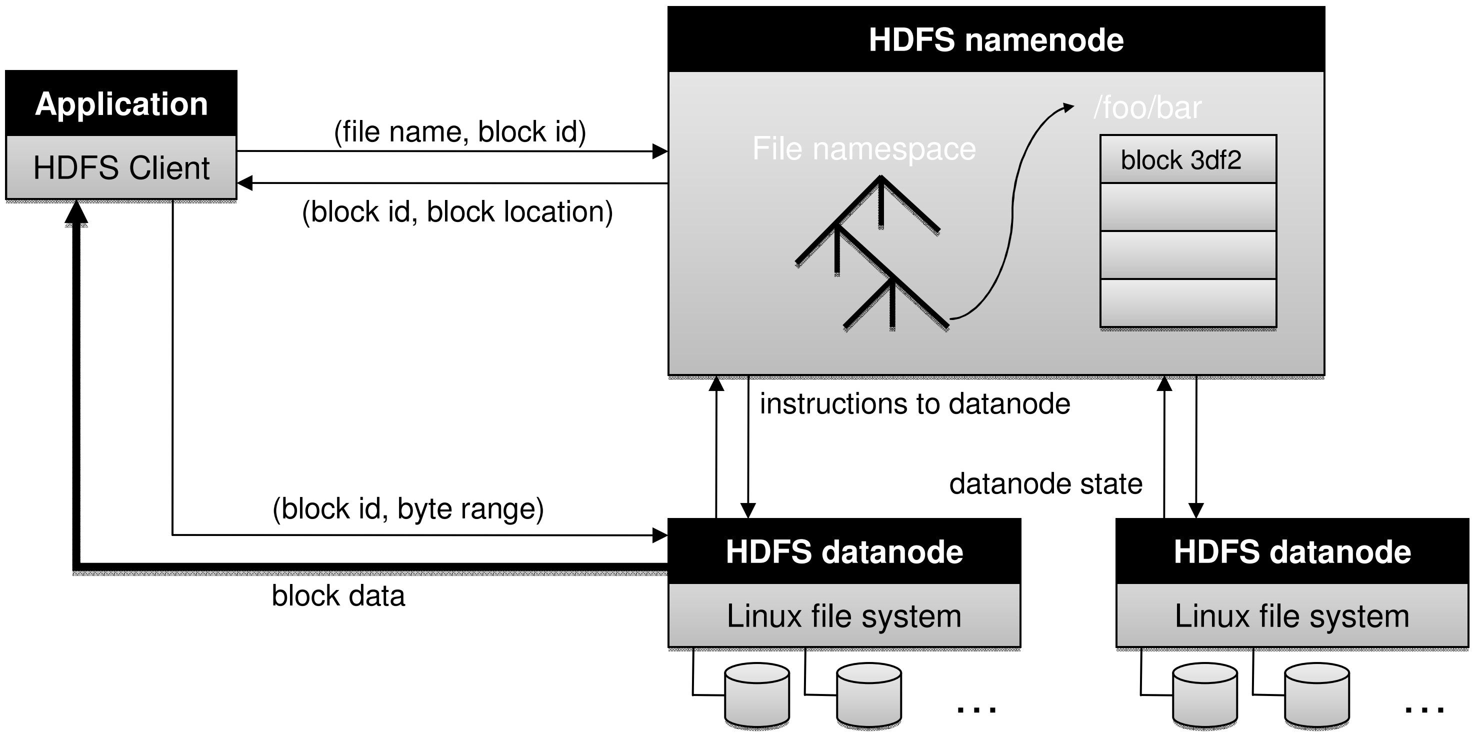
- We don’t want to move data (remember : we can store a file of 5TB in HDFS)
  - Put data in local disks and move the workers to the local disk
- A distributed file system allows to put data in a distributed fashion allowing worker to move where the data chunk resides
  - GFS (Google File System) for Google’s MapReduce
  - HDFS (Hadoop Distributed File System) for Hadoop

HDFS File System Designs

- Use fixed chunk size
  - Fixed size (64MB)
- Replicated 3 times across different nodes
- Centralized management with datanode that keeps track of namenodes
- No caching

HDFS = GFS clone (same basic ideas)

HDFS Architecture

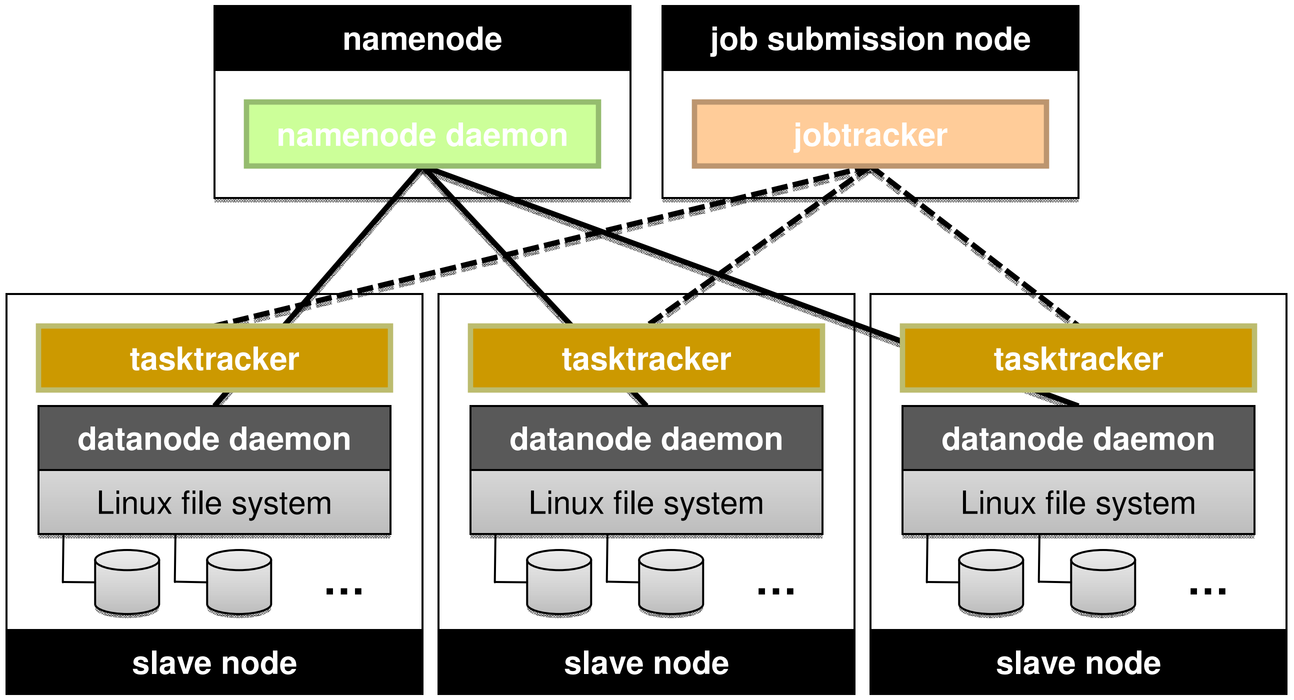


Picture from [1]

Namenode Functions

- Managing namespace of the filesystem
  - Metadata, permissions, etc
- File Operations coordinate through namenodes
  - Which datanode to write on
  - Which datanode to read from
- Maintain overall health
  - Pings to datanodes to check if they are up
  - Garbage collection

HDFS and MapReduce



Picture from [1]

Map Reduce for Multiple Linear Regression

$$J(\theta) = \frac{1}{2N} \left\| \underset{\substack{\uparrow \\ \mathbf{Y}}}{\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ Y_N \end{bmatrix}} - \underset{\substack{\uparrow \\ \mathbf{X}}}{\begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1K} \\ 1 & x_{21} & x_{22} & \dots & x_{2K} \\ & & \dots & & \\ & & \dots & & \\ & & \dots & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix}} \cdot \underset{\substack{\uparrow \\ \boldsymbol{\theta}}}{\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_K \end{bmatrix}} \right\|^2$$

$$\theta^* = (X^T X)^{-1} X^T Y$$

Map Reduce for Multiple Linear Regression

$$\theta^* = (\underbrace{X^T X}_{A})^{-1}\underbrace{X^T Y}_{b}$$

$$\begin{tikzcd}
 & A=\sum_i^m w_i(x_i x_i^T) \arrow[dl, "\text{Mappers}"'] \arrow[d] \arrow[dr] & \\
\sum_{subgroup} w_i(x_i x_i^T) \arrow[dr] & \cdots \arrow[d] & \sum_{subgroup} w_i(x_i x_i^T) \arrow[dl] \\
 & \text{Reducer} \arrow[d] & \\
 & A &
\end{tikzcd}$$

Compute b similarly

MapReduce for Naïve Bayes

- We need to sum over all xj where j=1 to M and xj=1 if xj occurs in document D of class 1
- We need to sum over all xj where j=1 to M and xj=1 if xj occurs in document D of class 1

$$\begin{tikzcd}
 & \sum_1^M 1\{x_j = k|y=1\} \arrow[dl, "\text{Mappers}"'] \arrow[d] \arrow[dr] & \\
\sum_{subgroup} 1\{x_j = k|y=1\} \arrow[dr] & \ldots \arrow[d] & \sum_{subgroup} 1\{x_j = k|y=1\} \arrow[dl] \\
 & \text{Reducer} &
\end{tikzcd}$$

Compute $\sum_1^M 1\{y=1\}$

MapReduce for K-Means

- Minimize J with respect to r_{nk}
  - Keep μ_k fixed

Step 1

- Optimize for each n separately by choosing r_{nk} for k that gives minimum $||x_n - r_{nk}||^2$

$$r_{nk} = 1 \text{ if } k = argmin_j ||x_n - \mu_j||^2$$
$$= 0 \text{ otherwise}$$

This requires assigning value of 0 or 1 to each data point
We just need to split the data with mapper to many split

Map : <ClusterID, point>

MapReduce for K-Means

- Minimize J with respect to μ_k
  - Keep r_{nk} fixed

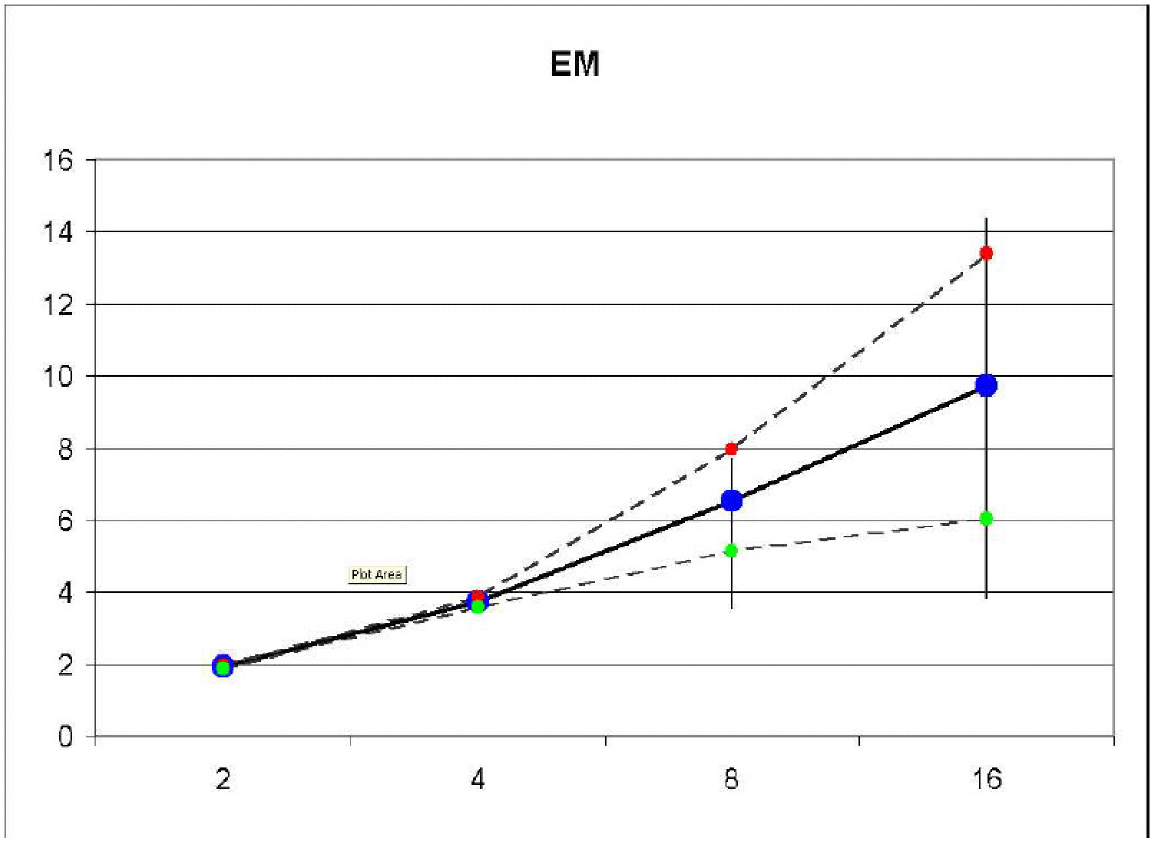
Step 2

- J is quadratic in μ_k. Minimize by setting derivative w.rt. μ_k to zero

$$\begin{tikzcd} & \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \arrow[dl] \arrow[d] \arrow[dr] & \\ \sum_{subgroup} r_{nk} x_n & \dots & \sum_{subgroup} r_{nk} x_n \end{tikzcd}$$

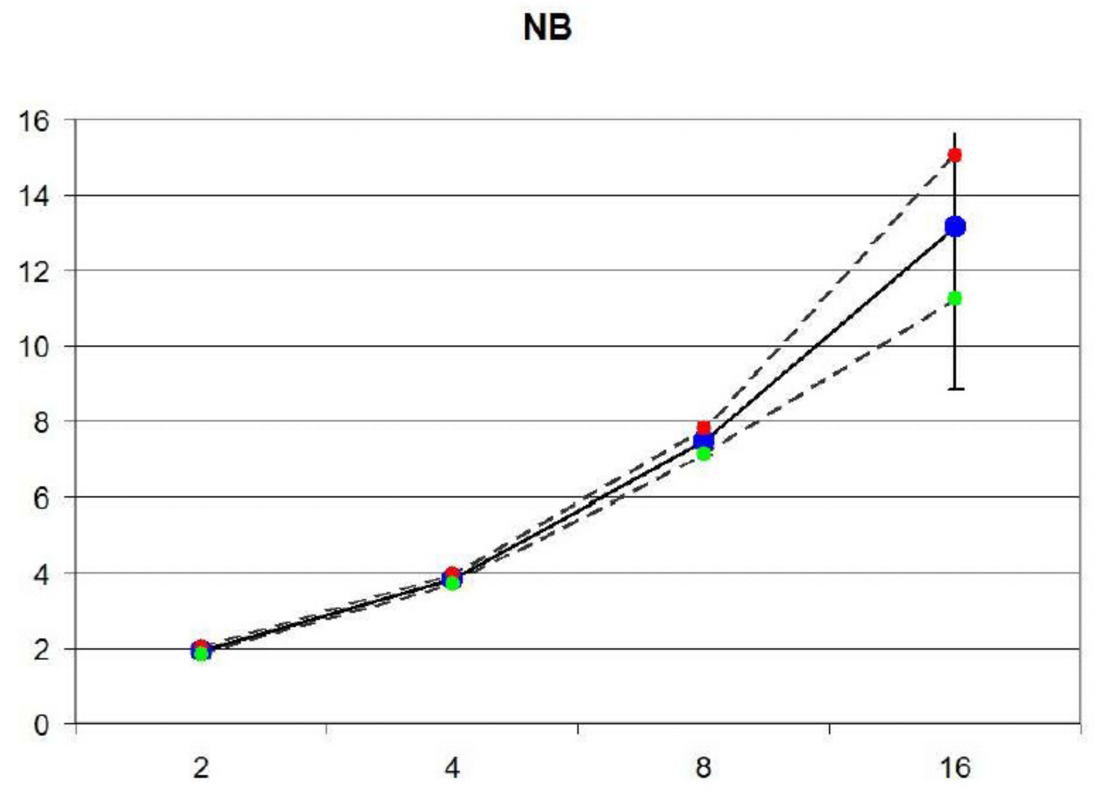
Reduce : <ClusterID, mean>

Speedup with MapReduce - EM



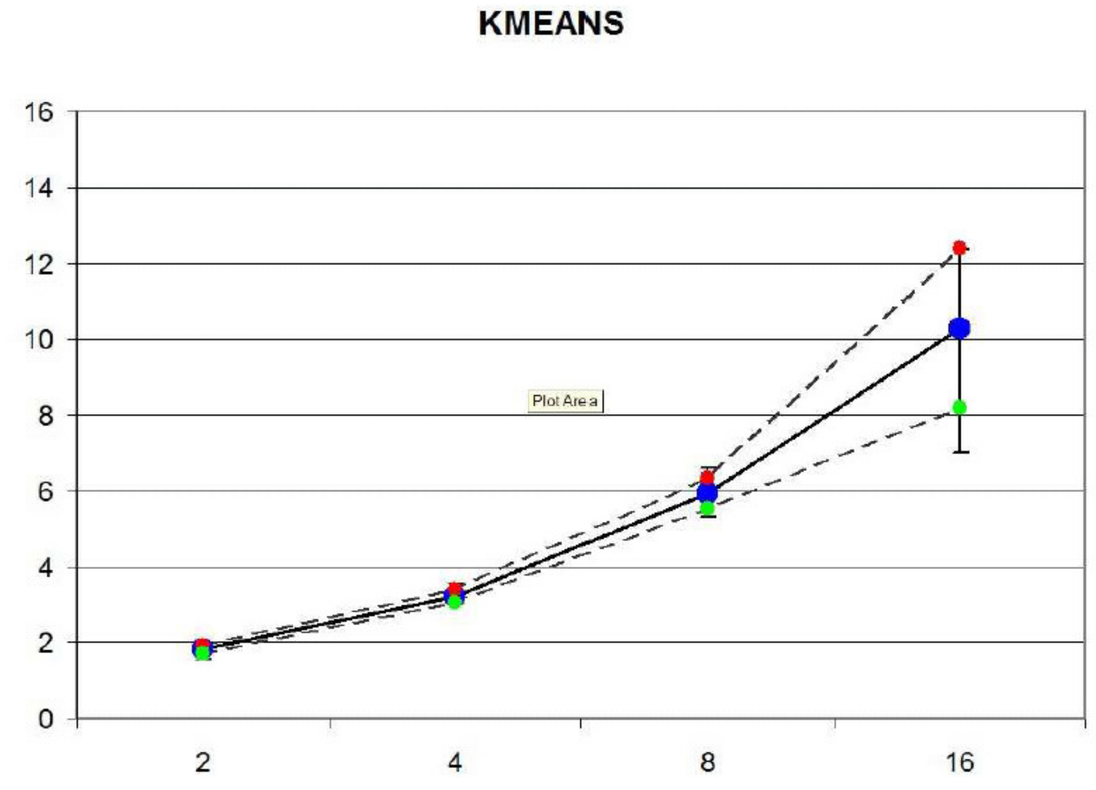
Chu, Cheng-Tao, et. al

Speedup with MapReduce – Naïve Byes



Chu, Cheng-Tao, et. al

Speedup with MapReduce : KMeans



Chu, Cheng-Tao, et. al

Amazon’s Elastic MapReduce

- We want to write ML algorithms we have learned so far in MapReduce Framework
- Use Elastic MapReduce of Amazon
- Naïve Bayes in MapReduce framework to classify our hockey and baseball documents
- We need to loadup data in Amazon’s persistent storage, S3

Naïve Bayes Experiment from Homework 1

- wget http://www.cs.columbia.edu/~smaskey/CS6998-0412/homework/hw1.tar.gz
- s3cmd put baseball_hockey_train.txt s3://srmbucket/exp1/input/baseball_hockey_train.txt

Step Back : Word Count Example

- Map

```
lines = sys.stdin

for line in lines:
    words = line.split()
    for word in words:
        print word, "\t", 1;
```

Key = wordID
Value = 1

Word Count Example

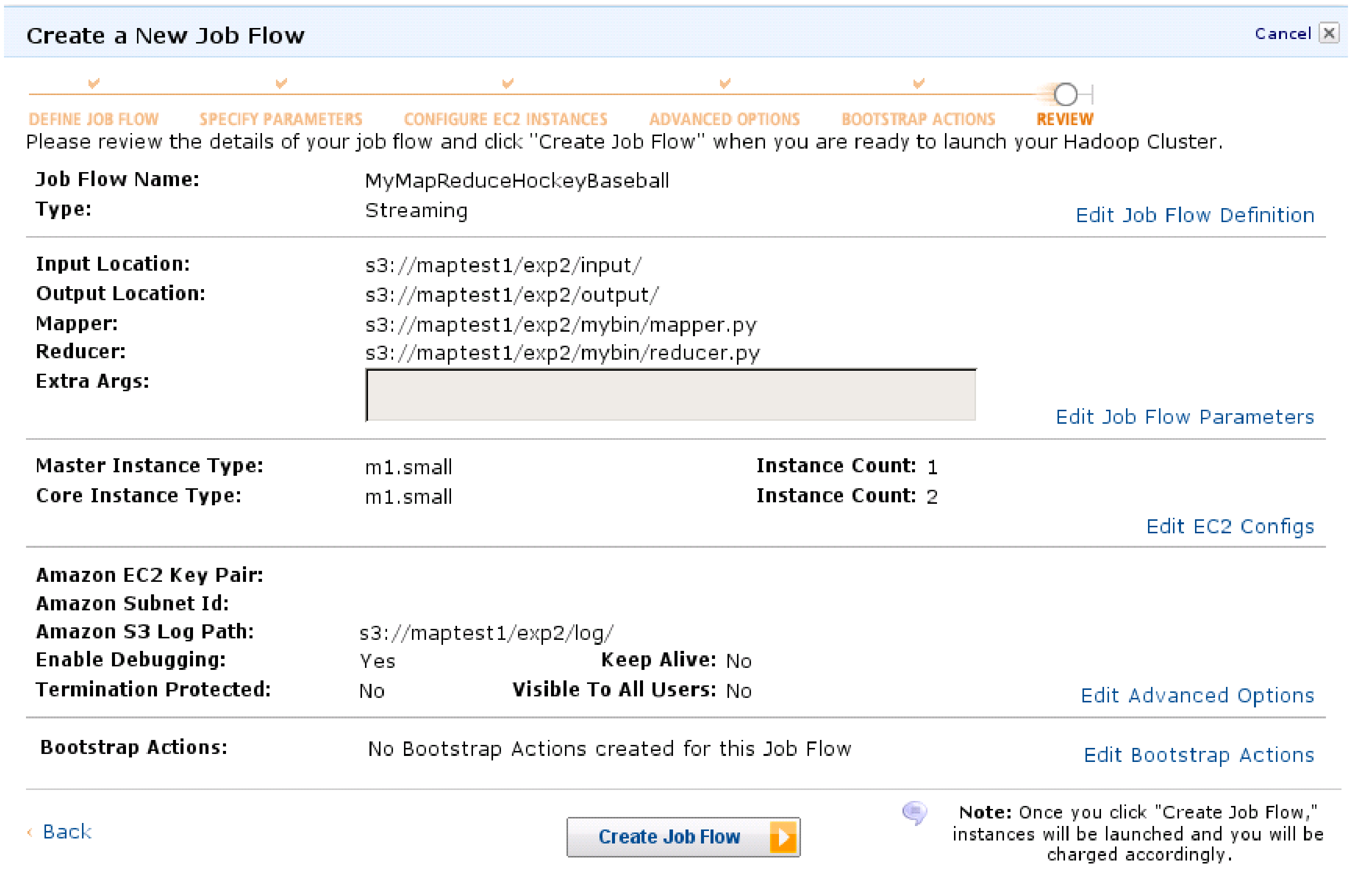
- Reduce

```
wordFreq={}

lines = sys.stdin

for line in lines:
    words = line.split("\t")
    key = words[0]
    val = words[1]
    if key in wordFreq:
        wordFreq[key] = wordFreq[key]+1
    else:
        wordFreq[key] = 1

for key in wordFreq.keys():
    print key, "\t", wordFreq[key]
```

Running MapReduce based WordCount in Amazon

- Copy over mapper.py and reducer.py to s3
  - s3cmd --recursive put mybin s3://maptest1/exp2/mybin
- Copy over data to s3
- s3cmd put baseball_hockey_train.txt s3://maptest1/exp2/input/baseball_hockey_train.txt

Setup MapReduce Job Flow



Please review the details of your job flow and click "Create Job Flow" when you are ready to launch your Hadoop Cluster.

Job Flow Name:	MyMapReduceHockeyBaseball		
Type:	Streaming		
Input Location:	s3://maptest1/exp2/input/		
Output Location:	s3://maptest1/exp2/output/		
Mapper:	s3://maptest1/exp2/mybin/mapper.py		
Reducer:	s3://maptest1/exp2/mybin/reducer.py		
Extra Args:			
Master Instance Type:	m1.small	**Instance Count:**	1
Core Instance Type:	m1.small	**Instance Count:**	2
Amazon EC2 Key Pair:			
Amazon Subnet Id:			
Amazon S3 Log Path:	s3://maptest1/exp2/log/		
Enable Debugging:	Yes	**Keep Alive:**	No
Termination Protected:	No	**Visible To All Users:**	No
Bootstrap Actions:	No Bootstrap Actions created for this Job Flow		

Note: Once you click "Create Job Flow," instances will be launched and you will be charged accordingly.

Commandline Interface

- elastic-mapreduce --create --stream
- --num-instances 1
- --mapper s3://yourBucket/bin/mapper.py
- --reducer s3://yourBucket/bin/reducer.py
- --input s3://yourBucket/data
- --name 'My WordCount Job'
- --output s3://yourBucket/output

References

- [1] Jimmy Lin, http://www.umiacs.umd.edu/~jimmylin/cloud-2010-Spring/syllabus.html