TEXAS STATE UNIVERSITY
SAN MARCOS

Department of Computer Science
San Marcos, TX 78666

Report Number TXSTATE-CS-TR-2010-24

Clustering in the Cloud

Xuan Wang

2010-05-05

Clustering in the Cloud

- Clustering Algorithms Adaption to Hadoop Map/Reduce Framework

Xuan Wang
Advisor: Anne Hee Hiong Ngu

Independent Study Report
Spring 2010

Contents

Table of figures

1. Introduction

1.1 Apache Hadoop

Inspired by a series of Google's MapReduce papers, Apache Hadoop is a Java-based software framework that enables data-intensive application in a distributed environment. It is released under the free license – Apache License Version 2.0. Hadoop enables applications to work with thousands of nodes and terabyte of data, without concerning the user with too much detail on the allocation and distribution of data and calculation.

1.2 MapReduce

Google introduced and patented MapReduce- a software framework to support distributed computing on large data sets on clusters of computers. The name "MapReduce" and the inspiration came from map and reduce functions in functional programming.

In MapReduce, the Map function processes the input in the form of key/value pairs to generate intermediate key/value pairs, and the Reduce function processes all intermediate values associated with the same intermediate key generated by the Map function, as shown in the figure below.

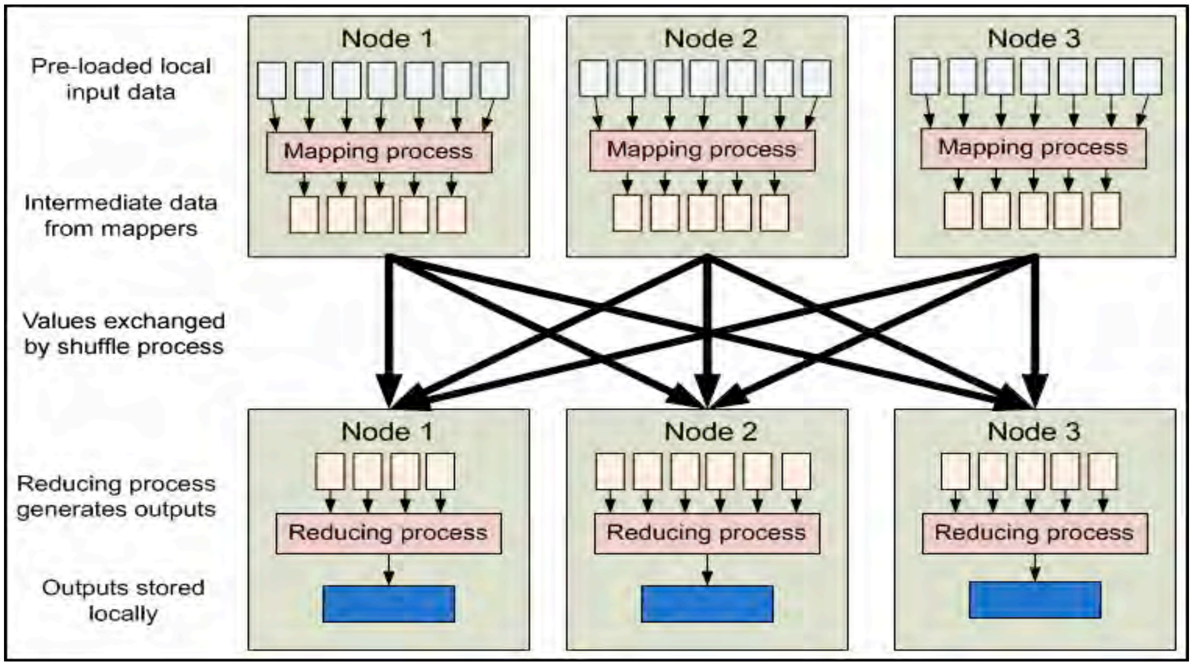

Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Node 2
Node 3
Mapping process
Reducing process

Figure 1 - MapReduce illustration

The MapReduce framework automatically parallelizes and executes on a large cluster of machines. Partitioning the input data, scheduling the program's execution across a set of machines, handling faults, and managing the required inter-machine communication are all handled by the run-time system. This enables programmers with no experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

1.3 Project motivation

Currently, many well established MapReduce implementations, such as "Inverted Indexing", and sorting algorithms are very intuitively MapReduce-able. For instance, in "Inverted Indexing", content from all documents is parsed into tokens by map function, and relayed to the reducer function in the form of key-value pairs, with the key being the token (normally a word), and the integer “1” as the key. Recall in Map/Reduce framework, all values with the same key will be sent the same Reducer. So in the reduce function, for each word, there are the number of its occurrence key-value pairs. The reduce function simply adds all “1”s up, and the result is the appearance count of this word in the collection. This process fits perfectly in the MapReduce framework, because “token” as key and frequency as value to construct key-value pairs are intuitive. As a matter of fact, the MapReduce framework was first designed to help indexing the internet for web search engine.

As the popularity of cloud computing keeps growing, and as Hadoop being one of the most influential cloud computing framework (and free too), researchers from more and more disciplines who need to deal with large quantity of data processing are attracted to cloud computing with the Hadoop framework. The question remains: “How applicable is Hadoop, or essentially the MapReduce framework, to my own area of work? Can my applications be converted to the MapReduce style?”

This independent study, besides familiarizing the author with the framework, explores the applicability of the MapReduce paradigm to areas of high-volume data computing, specifically data mining. Two algorithms are studies and successfully converted into MapReduce programs. An attempt was made to evaluate the efficiency gain with the framework.

In this study, Hadoop’s stand-alone distribution was used for development, and Amazon’s Elastic Cloud was used as the experiment set up as a real distributed environment.

2. Methodology

2.1 K-Means Clustering

K-Means clustering is a method of cluster analysis which aims to partition dataset (or entries, observations) into k clusters in which each data point belongs to the cluster with the nearest mean.

The choice of K-Means is mostly because of its simplicity.

2.1.1 Original Algorithm

As shown in Figure 2, the algorithm works as follows:

1. Randomly generate K points (call them cluster centers), K being the number of clusters desired.

2. Calculate the distance between each of the data points to each of the centers, and assign each point to the closest center.

3. Among each collection of points that have been assigned to the same center, and also calculate the new center (coordinates, or values).

4. With the new centers, repeat step 2. If the assignment of cluster for the data points changes, repeat 3 & 4. If cluster formation didn’t change, the clustering has finished.

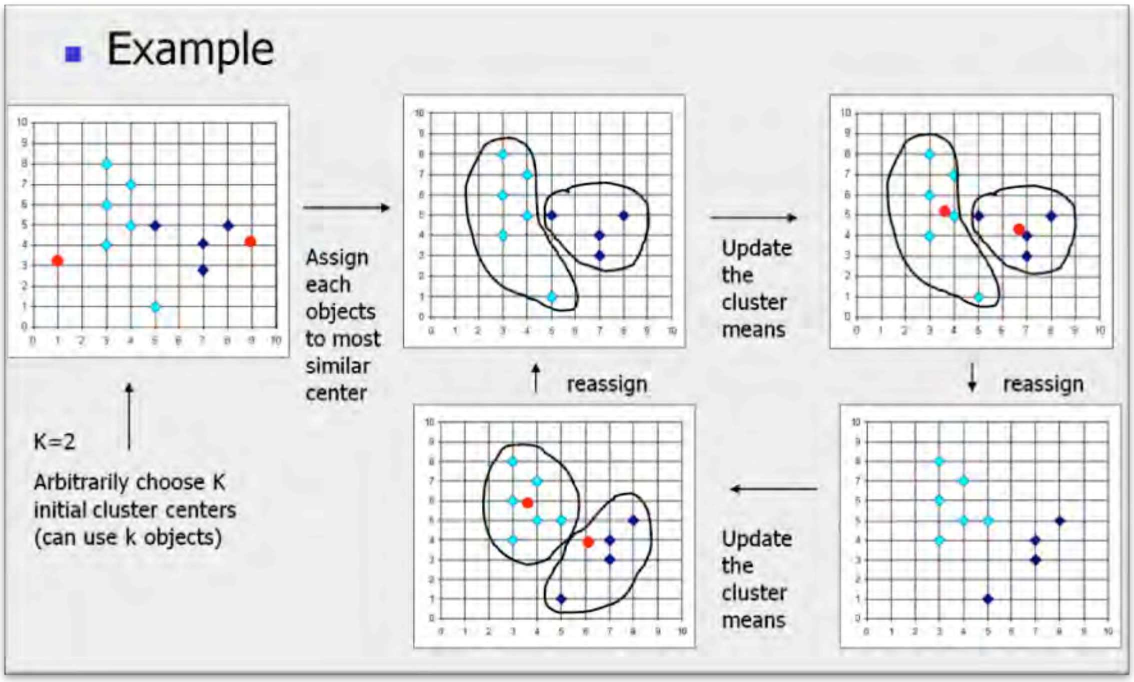

Example
Assign each objects to most similar center
Update the cluster means
reassign
reassign
Update the cluster means
K=2
Arbitrarily choose K initial cluster centers (can use k objects)

Figure 2 - Original K-Means clustering algorithm illustration

2.1.2 K-Means adaption to MapReduce framework

As shown in Figure 3, the MapReduce adaption of the algorithms works as follows:

0. To simplify the work, K points (K is the number of the clusters we wish to get out of the input dataset) are randomly selected from the input data file as initial cluster centers.

1. Input data set (through a data file) is partitioned into N parts (controlled by the run-time system, not your program). Each part is sent to a mapper.

2. In the map function, the distance between each point and each cluster center is calculated, and each point is labeled with the center index to which the distance is the smallest. Mapper outputs the key-value pairs of label assigned to each point, and the coordinates of the point.

3. All data points of the same current cluster (based on the current cluster centers) are sent to a single reducer. In the reduce function, new cluster center coordinates are easily computed. The output of the reducer is consequently the cluster index and its new coordinates.

4. The new cluster coordinates are compared to the original ones. If the difference is within a preset threshold, then program terminates, and we have found the clusters. If not, use the newly generated cluster centers and repeat step 2 to 4.

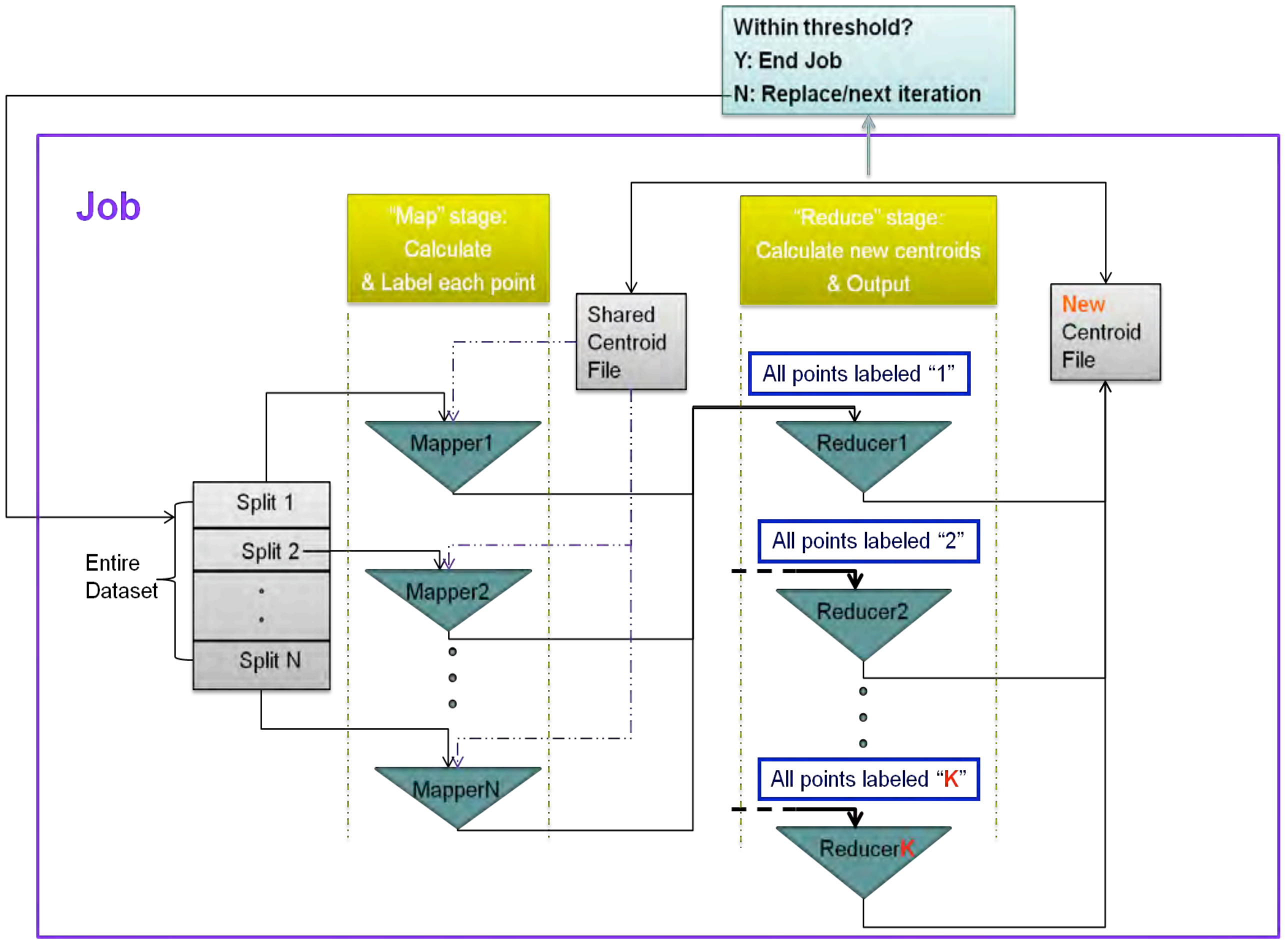


Figure 3 - MapReduce K-Means algorithm illustration

2.2 Divisive hierarchical clustering in MapReduce

The idea of divisive clustering (A.K.A. Diana) is pretty straightforward, as shown in Figure 4. The initial cluster is split into two partitions. Then each of the sub clusters is split into two clusters, one after another. The process continues until each of the data point forms a cluster of just itself. Any number of clusters can be achieved by selecting a proper cut-off height (or depth). (Refer to the dendrogram in Figure 5)

The MapReduce adaption of the divisive clustering algorithm is inspired by the previously explained K-Means clustering algorithm: the former could be implemented as simple iterations of the latter, with cluster number K set to be 2 for each iteration.

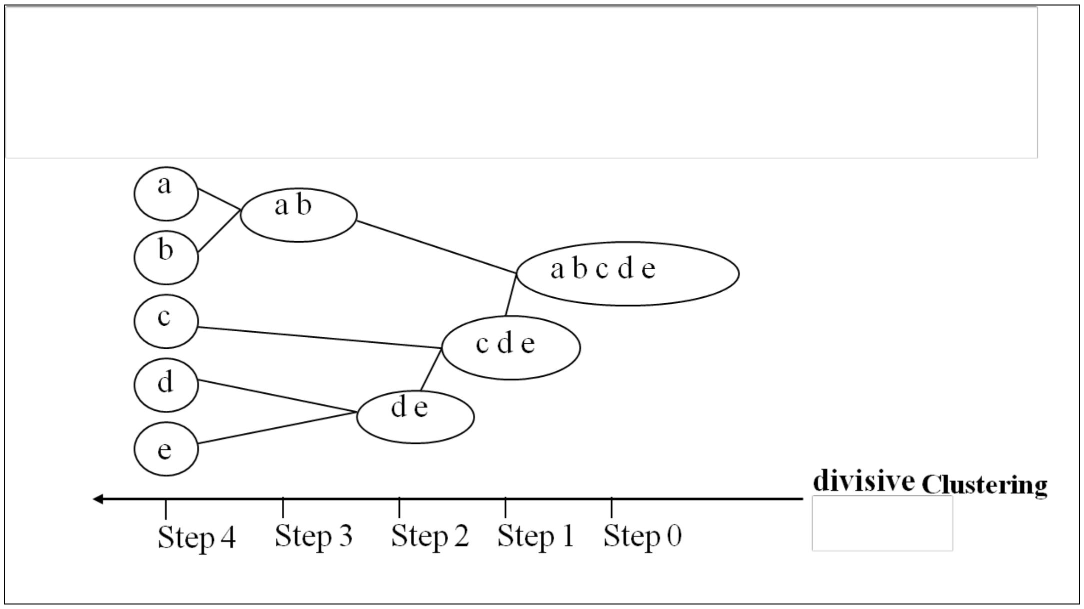


Figure 4 - Divisive clustering concept illustration

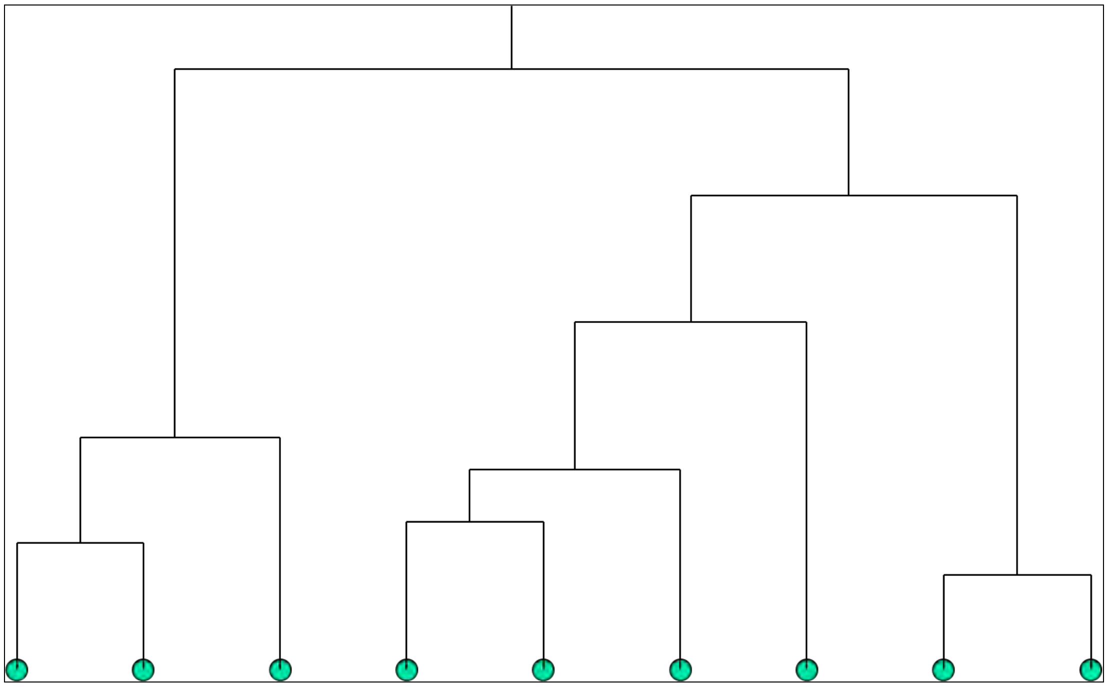


Figure 5 - Dendrogram showing divisive clustering

3. Results

The author generated 36K two-dimensional data points as the input dataset. The following results and conclusions are based on this data set.

- Correct clustering result (KMeans, Divisive Clustering)

The MapReduce version of both the KMeans and the Divisive Clustering algorithms give the correct the results, as verified both visually with plot and by other clustering tools.

- Minor performance gain on experiment dataset

  Both programs were verified on Amazon's cloud with multiple parameters setup (2, 5, or 10 clusters nodes launched). There is only very minor performance gain on the 2 cluster nodes setup, compared to running on the sudo-distributed system on a single machine (1 min 36 sec compared to 1 min 45 sec). There is no performance gain in 5 cluster nodes and 10 cluster nodes setups.

- Analysis
  - ➢ Possibly due to small size of dataset.

    The "normally recommended block size is 100MB". So it's very likely that the sample dataset was not big enough to have the efficiency gain by parallel computing overweigh the overhead between split partitions.

  - ➢ Further optimization needed.

    During the process of developing the two algorithms discussed above, the author noticed that there exist some potential optimizations that can increase efficiency.

    For instance, "weighted local cluster centers can be produced by each map function, and the output could be maximally the number of clusters, instead of the data points, which should reduce a lot of data traffic between the mapper and reducer.

4. Conclusion

- Clustering algorithms can adapt to M/R framework w/. linked multiple M/R jobs

  The single most important finding of this independent study is that some more complex (than inverted indexing) data intensive applications can be adapted to the MapReduce framework. The key point is to identify the distributable portion, which should also be the most data intensive and consequently the bottleneck of program efficiency improvement). And then write those portions into inter-linkable MapReduce jobs, with the appropriate intermediate results between jobs to relay the calculation results down.

5. Future works

- Optimization of converted algorithms

  As mentioned in part 3, the algorithms can be further optimized in a couple of ways.

- More efficient data passing alternatives between linked jobs

  Hadoop is aware of the need to custom outputs of mapper function and reducer functions. There is more efficient data format that the intermediate results can be stored as (such as sequence file format), etc. These options will enable faster read and write.

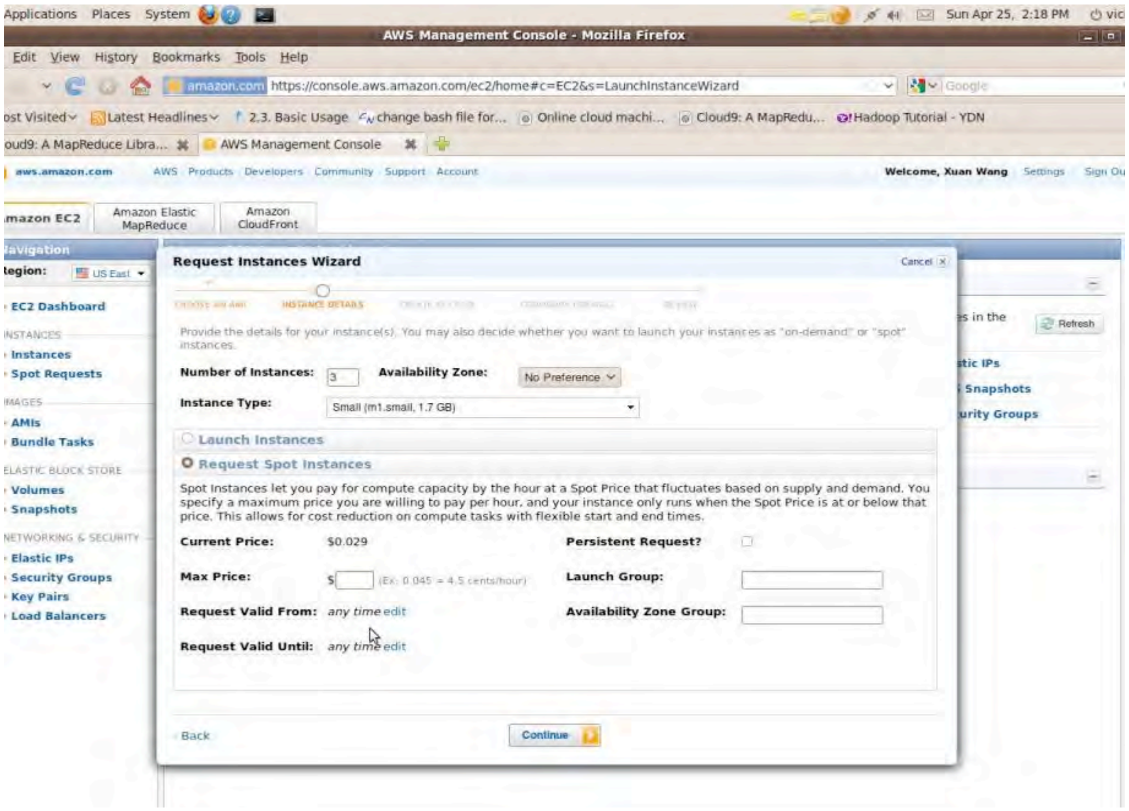
- Use available packages to build your own algorithms

A series of commercially available packages (Analytics1305 as mentioned in the reference being one of them) with optimized MapReduce implementation of some universal algorithms such as K-Means, etc. If applicable, complex algorithms can be built on top of those packages.

- Hadoop framework on TXSTATE SunGrid clusters (instead of paying Amazon, etc)

6. Appendix -- some screen shots

- Launching instances from Amazon EC2 UI:



Request Instances Wizard

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

Number of Instances: 3 Availability Zone: No Preference

Instance Type: Small (m1.small, 1.7 GB)

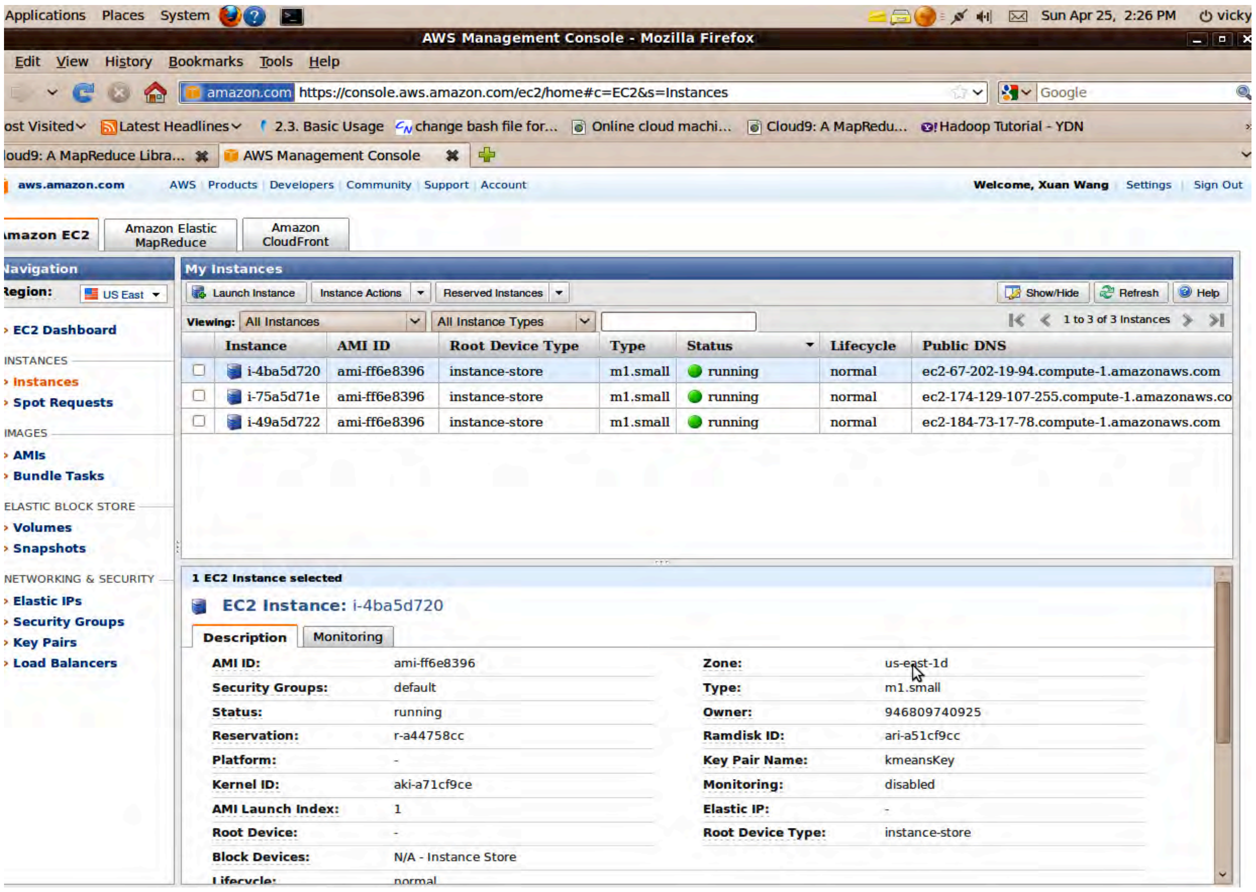
Launch Instances

Request Spot Instances

Spot Instances let you pay for compute capacity by the hour at a Spot Price that fluctuates based on supply and demand. You specify a maximum price you are willing to pay per hour, and your instance only runs when the Spot Price is at or below that price. This allows for cost reduction on compute tasks with flexible start and end times.

Current Price:	$0.029	Persistent Request?	☐
Max Price:	$ (Ex: 0.045 = 4.5 cents/hour)	Launch Group:	
Request Valid From:	any time edit	Availability Zone Group:	
Request Valid Until:	any time edit		

- Running instances


Applications Places System
Sun Apr 25, 2:26 PM vicky
AWS Management Console - Mozilla Firefox
Edit View History Bookmarks Tools Help
amazon.com https://console.aws.amazon.com/ec2/home#c=EC2&s=Instances
ost Visited Latest Headlines 2.3. Basic Usage change bash file for... Online cloud machi... Cloud9: A MapRedu... Hadoop Tutorial - YDN
loud9: A MapReduce Libra... AWS Management Console
aws.amazon.com AWS Products Developers Community Support Account
Welcome, Xuan Wang Settings Sign Out
mazon EC2
Amazon Elastic MapReduce
Amazon CloudFront
Navigation
Region: US East
EC2 Dashboard
INSTANCES
Instances
Spot Requests
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORKING & SECURITY
Elastic IPs
Security Groups
Key Pairs
Load Balancers
My Instances
Launch Instance
Instance Actions
Reserved Instances
Show/Hide
Refresh
Help
Viewing: All Instances
All Instance Types
1 to 3 of 3 Instances

	Instance	AMI ID	Root Device Type	Type	Status	Lifecycle	Public DNS
☐	i-4ba5d720	ami-ff6e8396	instance-store	m1.small	● running	normal	ec2-67-202-19-94.compute-1.amazonaws.com
☐	i-75a5d71e	ami-ff6e8396	instance-store	m1.small	● running	normal	ec2-174-129-107-255.compute-1.amazonaws.co
☐	i-49a5d722	ami-ff6e8396	instance-store	m1.small	● running	normal	ec2-184-73-17-78.compute-1.amazonaws.com

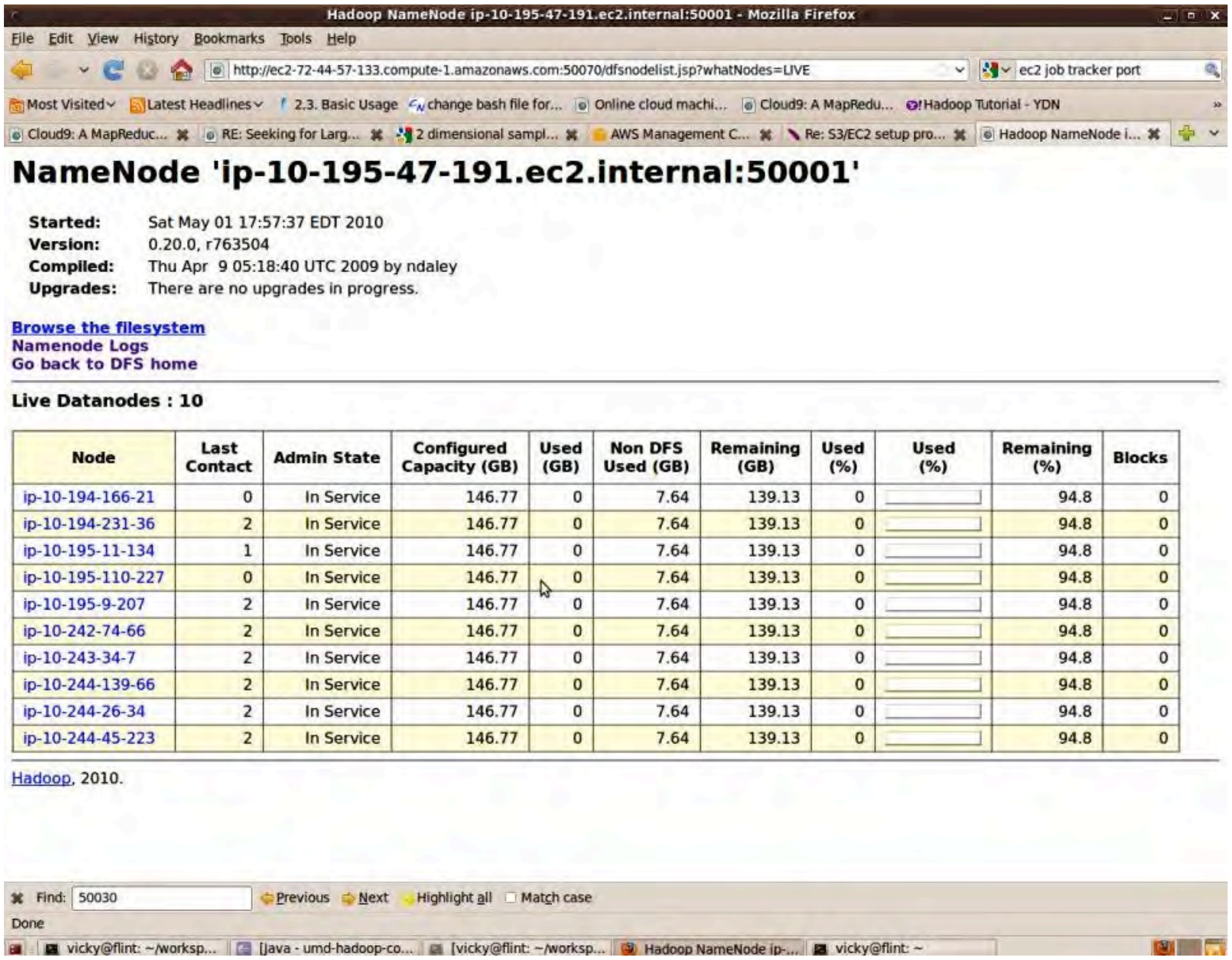
1 EC2 Instance selected

EC2 Instance: i-4ba5d720

Description | Monitoring

AMI ID:	ami-ff6e8396	**Zone:**	us-east-1d
Security Groups:	default	**Type:**	m1.small
Status:	running	**Owner:**	946809740925
Reservation:	r-a44758cc	**Ramdisk ID:**	ari-a51cf9cc
Platform:	-	**Key Pair Name:**	kmeansKey
Kernel ID:	aki-a71cf9ce	**Monitoring:**	disabled
AMI Launch Index:	1	**Elastic IP:**	-
Root Device:	-	**Root Device Type:**	instance-store
Block Devices:	N/A - Instance Store		
Lifecycle:	normal		

- Name node tracker (port 50070)


Hadoop NameNode ip-10-195-47-191.ec2.internal:50001 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://ec2-72-44-57-133.compute-1.amazonaws.com:50070/dfsnodelist.jsp?whatNodes=LIVE
ec2 job tracker port
Find: 50030
Done

NameNode 'ip-10-195-47-191.ec2.internal:50001'

Started:	Sat May 01 17:57:37 EDT 2010
Version:	0.20.0, r763504
Compiled:	Thu Apr 9 05:18:40 UTC 2009 by ndaley
Upgrades:	There are no upgrades in progress.

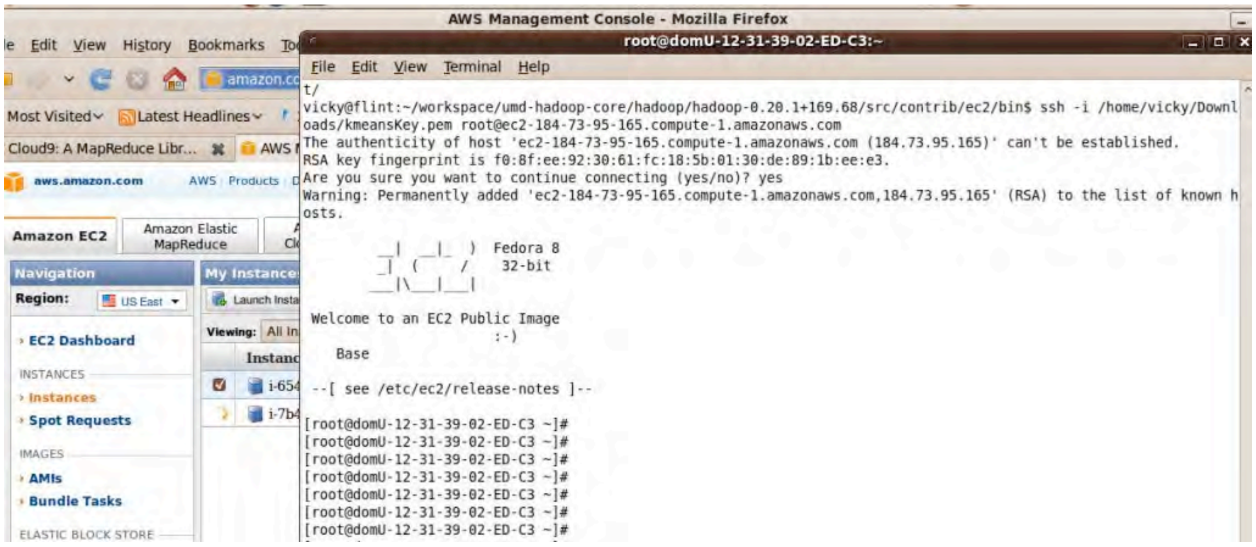
Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 10

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
ip-10-194-166-21	0	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-194-231-36	2	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-195-11-134	1	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-195-110-227	0	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-195-9-207	2	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-242-74-66	2	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-243-34-7	2	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-244-139-66	2	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-244-26-34	2	In Service	146.77	0	7.64	139.13	0		94.8	0
ip-10-244-45-223	2	In Service	146.77	0	7.64	139.13	0		94.8	0

Hadoop, 2010.

- Connection to master node



```
t/
vicky@flint:~/workspace/umd-hadoop-core/hadoop/hadoop-0.20.1+169.68/src/contrib/ec2/bin$ ssh -i /home/vicky/Downl
oads/kmeansKey.pem root@ec2-184-73-95-165.compute-1.amazonaws.com
The authenticity of host 'ec2-184-73-95-165.compute-1.amazonaws.com (184.73.95.165)' can't be established.
RSA key fingerprint is f0:8f:ee:92:30:61:fc:18:5b:01:30:de:89:1b:ee:e3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-184-73-95-165.compute-1.amazonaws.com,184.73.95.165' (RSA) to the list of known h
osts.

        __|  __|_  )  Fedora 8
        _|  (     /    32-bit
       ___|\___|___|

 Welcome to an EC2 Public Image
                       :-)

    Base

 --[ see /etc/ec2/release-notes ]--

[root@domU-12-31-39-02-ED-C3 ~]#
[root@domU-12-31-39-02-ED-C3 ~]#
[root@domU-12-31-39-02-ED-C3 ~]#
[root@domU-12-31-39-02-ED-C3 ~]#
[root@domU-12-31-39-02-ED-C3 ~]#
[root@domU-12-31-39-02-ED-C3 ~]#
[root@domU-12-31-39-02-ED-C3 ~]#
```

7. References

Analytics1305. (2010). *Integrated solutions for large scale data analysis*. Retrieved from Integrated solutions for large scale data analysis: http://www.analytics1305.com/cloud/index.php

Apache. (2010, Feb 19). *Apache Hadoop Quick Start*. Retrieved Mar 2010, from Apache Hadoop: http://hadoop.apache.org/common/docs/current/quickstart.html

Berry, M. (2008, Feb 9). *MapReduce and K-Means Clustering*. Retrieved Mar 2010, from Data Miners Blog: http://blog.data-miners.com/2008/02/mapreduce-and-k-means-clustering.html

Cloudera. (2010). *Hadoop Training*. Retrieved 2010, from Cloudera: Apache Hadoop for the Enterprise: http://www.cloudera.com/resources/?type=Training

GoogleCodeUniversity. (2007, Jul). *Google: Cluster Computing and MapReduce*. Retrieved Mar 2010, from Google Code: http://code.google.com/edu/submissions/mapreduce-minilecture/listing.html

Lin, J. (2010, Apr). *Data-Intensive Information Processing Applications Course(Spring 2010)*. Retrieved Apr 2010, from University of Maryland: http://www.umiacs.umd.edu/~jimmylin/cloud-2010-Spring/index.html

LLC, A. W. (2009, Nov 30). *Get Started with EC2*. Retrieved Mar 2010, from Amazon Elastic Compute Cloud, Getting Started Guide: http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/

Yahoo! (n.d.). *Yahoo! Hadoop Tutorial*. Retrieved Jan-Apr 2010, from Yahoo! Developer Network: http://developer.yahoo.com/hadoop/tutorial/