

Introduction to MapReduce

Csci E185 Big Data Analytics
Zoran B. Djordjević

@Zoran B. Djordjevic

1

Serial vs. Parallel Programming Model

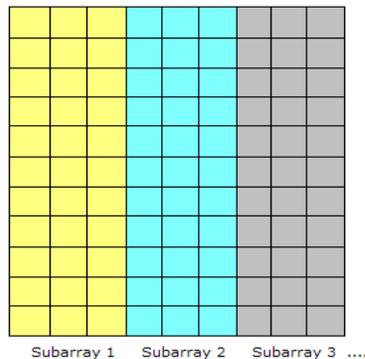
- Many or most of our programs are *Serial*.
 - A *Serial Program* consists of a sequence of instructions, where each instruction executes one after the other.
 - Serial programs run from start to finish on a single processor.
- *Parallel programming* developed as a means of improving performance and efficiency.
 - In a *Parallel Program*, the processing is broken up into parts, each of which could be executed concurrently on a different processor. Parallel programs could be faster.
 - Parallel Programs could also be used to solve problems involving large datasets and non-local resources.
 - Parallel Programs are usually ran on a set of computers connected on a network (a pool of CPUs), with an ability to read and write large files supported by a distributed file system.

@Zoran B. Djordjevic

2

Common Situation

- A common situation involves processing of a large amount of consistent data.
- If the data could be decomposed into equal-size partitions, we could devise a parallel solution. Consider a huge array which can be broken up into sub-arrays



If the same processing is required for each array element, with no dependencies in the computations, and no communication required between tasks, we have an ideal parallel computing opportunity, the so called *Embarrassingly Parallel problem*. A common implementation of this approach is a technique called *Master/Worker*.

@Zoran B. Djordjevic

3

MapReduce Programming Model

- MapReduce programming model derives from the map and reduce combinator in Lisp programming language.
- In Lisp, a `map` takes as input a function and a sequence of values. It then applies the function to each value in the sequence. A `reduce` combines all the elements of a sequence using a binary operation. For example, it can use "+" to add up all the elements in the sequence.
- MapReduce was developed within Google as a mechanism for processing large amounts of raw data, for example, crawled documents or web request logs.
- This data is so large, it must be distributed across thousands of machines in order to be processed in a reasonable time. This distribution implies parallel computing since the same computations are performed on each CPU, but with a different dataset.
- MapReduce is an abstraction that allows Google engineers to perform simple computations while hiding the details of parallelization, data distribution, load balancing and fault tolerance.

@Zoran B. Djordjevic

4

MapReduce Library

- Map function, written by a user of the MapReduce library, takes an input key/value pairs and produces a set of intermediate key/value pairs.
- The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function.
- The Reduce function, also written by the user, accepts an intermediate key and a set of values for that key. It merges together these values to form a possibly smaller set of values.

@Zoran B. Djordjevic

5

Why Does Google Need Parallel Processing

- Google's search mechanisms rely on several matrices of sizes that are really big: $10^{10} \times 10^{10}$
- Google needs to spread its processing on tens and hundreds of thousands of machines in order be able to rank the pages of World Wide Web in "real time".
- Today we will just indicate some features of Google mechanisms.

@Zoran B. Djordjevic

6

MapReduce Execution

- The `Map` invocations are distributed across multiple machines by automatically partitioning the input data into a set of `M` splits or **shards**.
- The input shards can be processed in parallel on different machines.
- `Reduce` invocations are distributed by partitioning the intermediate key space into `R` pieces using a partitioning function (e.g., `hash(key) mod R`).
- The number of partitions (`R`) and the partitioning function are specified by the user.

@Zoran B. Djordjevic

7

Vocabulary and Number of Words in all Documents

- Consider the problem of counting the number of occurrences of each word in a large collection of documents

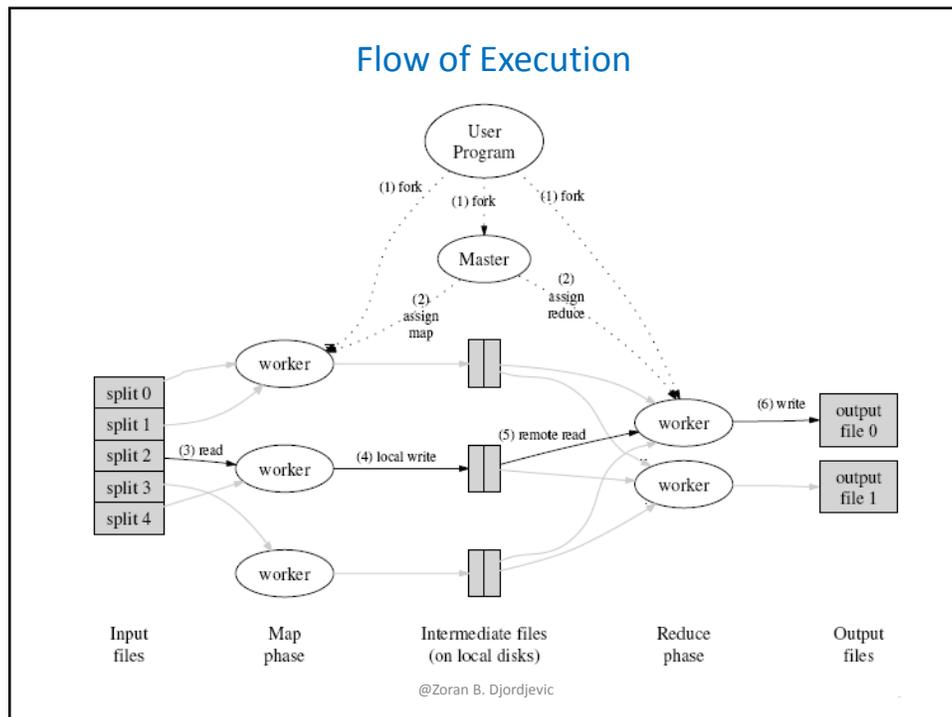
```
map(String documentName, String documentContent):
  //key: document name, value: document content
  for each word w in documentContent:
    //key: word, value: number of occurrences
    EmitIntermediate(w, wordCount);
```

```
reduce(String w, Iterator values):
  // key: a word, // values: a list of counts
  int result = 0;
  for each v in values:
    result += v;
  Emit(w, result);
```

- The map function emits each word plus an associated count of occurrences in a document.
- The reduce function sums all the counts for every word.

@Zoran B. Djordjevic

8



Master and Worker

Role of the MASTER is to:

1. Initialize the array and splits it up according to the number of available WORKERS
2. Send each WORKER its sub-array
3. Receive the results from each WORKER
4. Perform some final calculation if needed.

Role of the WORKER is to:

- Receive the sub-array from the MASTER
- Perform processing on the sub array
- Return results to the MASTER

MapReduce Steps

1. The MapReduce library in the user program first shards the input files into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece. It then starts up many copies of the program on a cluster of machines.
2. One of the copies of the program is special: the Master. The rest are workers that are assigned work by the Master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.
3. A worker who is assigned a map task reads the contents of the corresponding input shard. It parses key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory.
4. Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers.

@Zoran B. Djordjevic

11

MapReduce Steps

5. When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. If the amount of intermediate data is too large to fit in memory, an external sort is used.
 6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is appended to a final output file for this reduce partition.
 7. When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the MapReduce call in the user program returns back to the user code.
- After successful completion, the output of the MapReduce execution is available in the R output files

@Zoran B. Djordjevic

12

Usage of MapReduce at Google

- **Distributed Grep:** The map function emits a line if it matches a given pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.
- **Count of URL Access Frequency:** The map function processes logs of web page requests and outputs <URL, 1>. The reduce function adds together all values for the same URL and emits a <URL, total count> pair.
- **Reverse Web-Link Graph:** The map function outputs <target, source> pairs for each link to a target URL found in a page named "source". The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: <target, list(source)>.
- Most of the rest of Google functionality.

@Zoran B. Djordjevic

13

Open Source MapReduce

- We will not build MapReduce frameworks
- We will learn to use an open source MapReduce Framework called Hadoop which is offered by Amazon and available at *Apache.org*.

@Zoran B. Djordjevic

14

Hadoop and Amazon Elastic MapReduce

@Zoran B. Djordjevic

15

What is MapReduce

- MapReduce is simple parallel programming model (framework) designed for scalability and fault-tolerance.
- MapReduce is Pioneered by Google
 - Google processes 20 petabytes of data per day
- Popularized and further developed by the open-source project Hadoop.
 - Used at Yahoo!, Facebook, Amazon, ...



@Zoran B. Djordjevic

16

What is MapReduce used for?

- At Google:
 - Index construction for Google Search
 - Article clustering for Google News
 - Statistical machine translation
- At Yahoo!:
 - “Web map” powering Yahoo! Search
 - Spam detection for Yahoo! Mail
- At Facebook:
 - Data mining
 - Ad optimization
 - Spam detection
- At New York Times
 - Moving typeset into PDF

@Zoran B. Djordjevic

17

What is MapReduce used for?

- In research:
 - Astronomical image analysis
 - Bioinformatics
 - Analyzing Wikipedia conflicts
 - Natural Language Processing
 - Particle physics
 - Ocean currents simulation
 - Weather analysis, etc.
 - Electioneering
 - Customer behavior analysis
 - ...

@Zoran B. Djordjevic

18

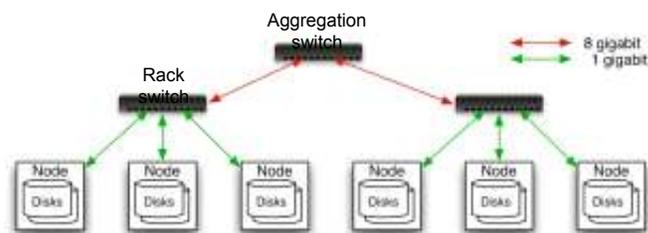
MapReduce Design Goals

1. **Scalability to large data volumes:**
 - 1000's of machines, 10,000's of disks
2. **Cost-efficiency:**
 - Commodity machines (cheap, but unreliable)
 - Commodity network
 - Automatic fault-tolerance (fewer administrators)
 - Easy to use (fewer programmers)
3. **Bring Processing to Data**
 - Every job (map or reduce) is performed on the processor adjacent to the disk containing data.
 - Data transfers are minimized.

@Zoran B. Djordjevic

19

Typical Hadoop Cluster



- 40 nodes/rack, 1000-4000 nodes in cluster
- 1 Gbps bandwidth within rack, 8 Gbps out of rack
- Node specs (Yahoo terasort):
8 x 2GHz cores, 8 GB RAM, 4 disks (= 4 TB?)

@Zoran B. Djordjevic

20

Typical Hadoop Cluster



- If you do not own one of these warehouses, though luck. ☺

@Zoran B. Djordjevic

21

Challenges

- 1. Cheap nodes fail, especially if you have many**
 - Mean time between failures for 1 node = 3 years
 - Mean time between failures for 1000 nodes = 1 day
 - Solution: Build fault-tolerance into system
- 2. Commodity network = low bandwidth**
 - Solution: Push computation to the data
- 3. Programming distributed systems is hard**
 - Solution: Data-parallel programming model: users write “map” & “reduce” functions, system distributes work and handles faults

@Zoran B. Djordjevic

22

Hadoop Components

- **Distributed file system (HDFS)**
 - Single namespace for entire cluster
 - Replicates data 3x for fault-tolerance
 - Allows writes, deletes and appends. Does not allow updates of data blocks.

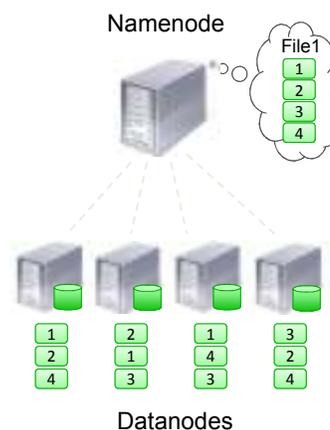
- **MapReduce framework**
 - Executes user jobs specified as “map” and “reduce” functions
 - Manages work distribution & fault-tolerance

@Zoran B. Djordjevic

23

Hadoop Distributed File System

- Files split into 64MB-128MB *blocks*
- Blocks replicated across several *datanodes* (usually 3)
- Single *namenode* stores metadata (file names, block locations, etc)
- Optimized for large files, sequential reads
- Files are append-only



@Zoran B. Djordjevic

24

MapReduce Programming Model

- Data type: key-value *records*
- Map function:

$$(K_{in}, V_{in}) \rightarrow \text{list}(K_{inter}, V_{inter})$$
- Intermediate keys do not have to be related to the initial keys in any way.
- Reduce function is fed collection of intermediate values for each intermediate key.

$$(K_{inter}, \text{list}(V_{inter})) \rightarrow \text{list}(K_{out}, V_{out})$$
- Reduce function transforms that collection into a final result, a list of key-value pairs.

@Zoran B. Djordjevic

25

Example: Word Count

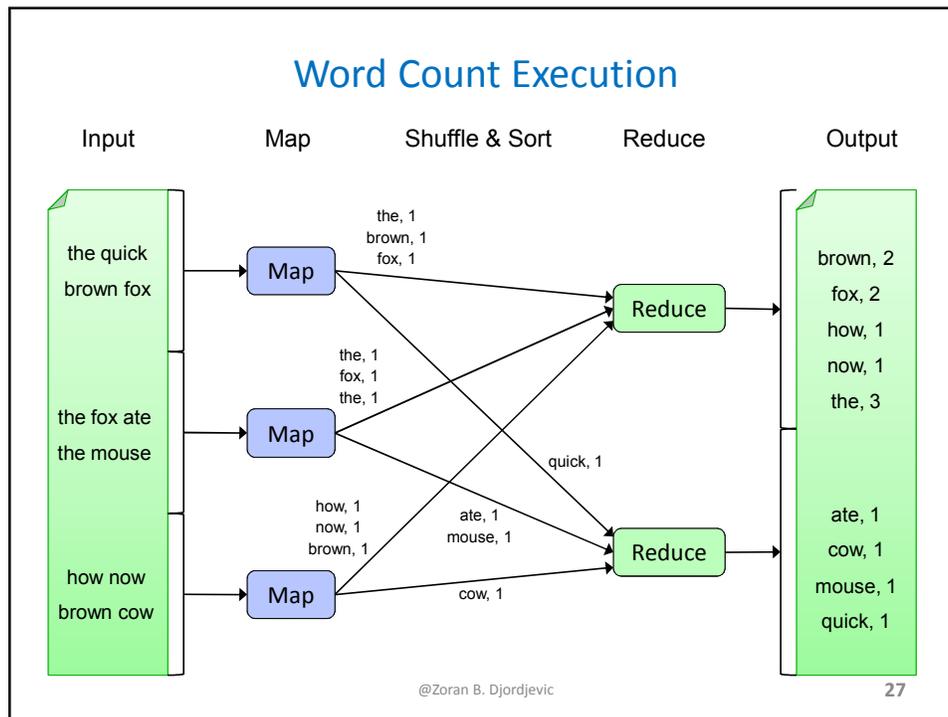
You are given a document with many lines.

Define functions mapper and reducer:

```
def mapper(line):
    foreach word in line.split():
        output(word, 1)
    • word is the key, 1 is the value
def reducer(key, list(values)):
    output(key, sum(values))
```

@Zoran B. Djordjevic

26



MapReduce Execution Details

- Single *master* controls job execution on multiple *slaves*
- Mappers preferentially placed on same node or same rack as their input block
 - Minimizes network usage
- Mappers save outputs to local disk before they are served to reducers by Shuffle and Sort phase
 - Allows recovery if a reducer crashes
 - Allows having more reducers than nodes

@Zoran B. Djordjevic

28

An Optimization: The Combiner

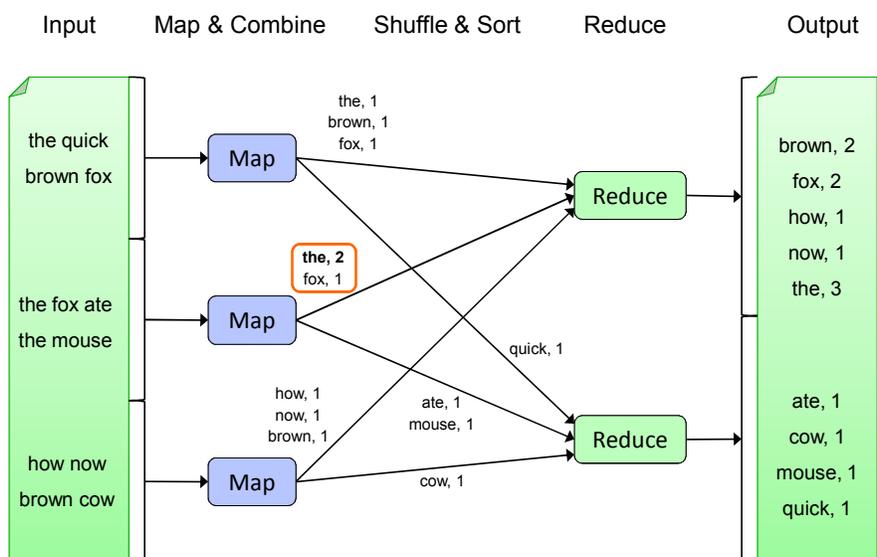
- Shuffle and Sort phase typically introduces combiner, a local aggregation function, for repeated keys produced by same map
- Combiner works with associative functions like sum, count, max
- Decreases size of intermediate data
- Example: map-side aggregation for Word Count:

```
def combiner(key, values):
    output(key, sum(values))
```

@Zoran B. Djordjevic

29

Word Count with Combiner



@Zoran B. Djordjevic

30

Fault Tolerance in MapReduce

1. If a task crashes:
 - Retry on another node
 - OK for a map because it has no dependencies, just do it again, duplicate data are saved somewhere else
 - OK for reduce because map outputs are on disk, or several disks, new reduce could start over.
 - If the same task fails repeatedly, fail the job or ignore that input block. Quite often tasks are statistical in nature, no one would notice a slight error, anyway.
2. If a node crashes:
 - Re-launch its current tasks on other nodes
 - Re-run any maps the node previously ran
 - Necessary because their output files were lost along with the crashed node

@Zoran B. Djordjevic

31

Fault Tolerance in MapReduce

3. If a task is going slowly (straggler):
 - Launch second copy of task on another node ("speculative execution")
 - Take the output of whichever copy finishes first, and kill the other
- Surprisingly important in large clusters
 - Stragglers occur frequently due to failing hardware, software bugs, misconfiguration, etc
 - Single straggler may noticeably slow down a job
- **For these fault tolerance features to work, your map and reduce tasks must be side-effect-free**

@Zoran B. Djordjevic

32

Takeaways

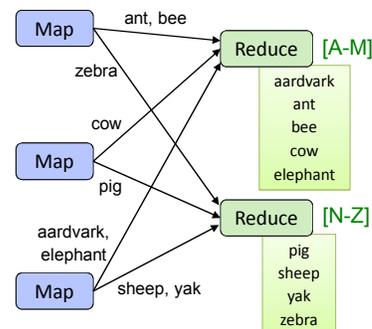
- By providing a data-parallel programming model, MapReduce can control job execution in useful ways:
 - Automatic division of job into tasks
 - Automatic placement of computation near data
 - Automatic load balancing
 - Recovery from failures & stragglers
- User focuses on application, not on complexities of distributed computing

@Zoran B. Djordjevic

33

1. Sort

- Input:** (key, value) records
- Output:** same records, sorted by key
- Map:** identity function
- Reduce:** identify function
- Trick:** Pick partitioning function h such that $k_1 < k_2 \Rightarrow h(k_1) < h(k_2)$



@Zoran B. Djordjevic

34

2. Inverted Index

- **Input:** (filename, text) records
- **Output:** list of files containing each word
- **Map:**

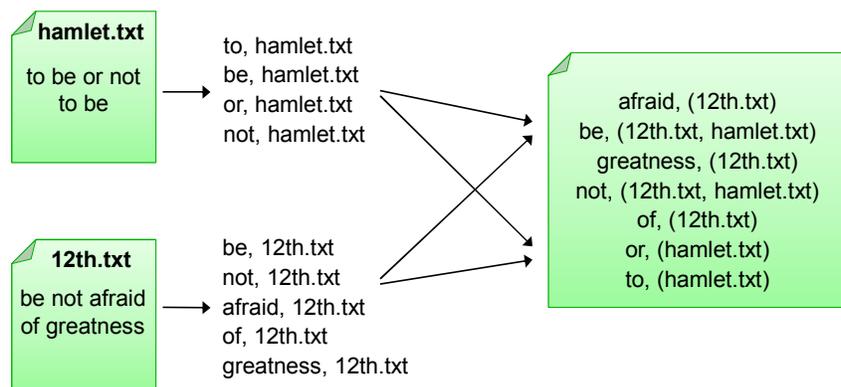
```
foreach word in text.split():
    output(word, filename)
```
- **Combine:** uniquify filenames for each word
- **Reduce:**

```
def reduce(word, filenames):
    output(word, sort(filenames))
```

@Zoran B. Djordjevic

35

Inverted Index Example



@Zoran B. Djordjevic

36

Getting Started with Hadoop

- Download from hadoop.apache.org
- To install locally, unzip and set JAVA_HOME
- Details: hadoop.apache.org/core/docs/current/quickstart.html

- Several ways to write jobs:
 - Java API
 - Hadoop Streaming (for Python, Perl, etc)
 - Pipes API (C++)
- If you want to do very sophisticated work and create special map/reduce procedures you have few options. Learn one of Hadoop's Api-s

@Zoran B. Djordjevic

37

Elastic MapReduce

- Amazon Elastic MapReduce is a web service that utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).
- Using Amazon Elastic MapReduce, you can instantly provision as much or as little capacity as you like to perform data-intensive tasks for applications such as web indexing, data mining, log file analysis, machine learning, financial analysis, scientific simulation, and bioinformatics research.

@Zoran B. Djordjevic

38

Benefits of Elastic MapReduce

- Amazon Elastic MapReduce lets you focus on crunching or analyzing your data without having to worry about time-consuming set-up, management or tuning of Hadoop clusters or the hardware capacity upon which they sit.
- Amazon Elastic MapReduce automatically sub-divides the data in a job flow into smaller chunks so that data can be processed (the “map” function) in parallel, and eventually recombining the processed data into the final solution (the “reduce” function).
- Amazon S3 serves as the source for the data being analyzed, and as the output destination for the end results.

@Zoran B. Djordjevic

39

Elastic MapReduce Functionality

- Develop your data processing application.
- Amazon Elastic MapReduce enables job flows to be developed in SQL-like languages, such as Hive and Pig.
- If desired, more sophisticated applications can be run in: Java, Ruby, Perl, Python, PHP, R, or C++.
- Upload your data and your processing application into Amazon S3.
- Log in to the AWS Management Console to start an Amazon Elastic MapReduce “job flow.” Alternatively you can start a job flow by specifying the same information mentioned above via our Command Line Tools or APIs.
- Monitor the progress of your job flow(s) directly from the AWS Management Console, Command Line Tools or APIs.

@Zoran B. Djordjevic

40

Service Highlights

- Amazon Elastic MapReduce enables you to use as many or as few compute instances running Hadoop as you want. You can commission one, hundreds, or even thousands of instances.
- You don't need to worry about setting up, running, or tuning the performance of Hadoop clusters.
- Amazon Elastic MapReduce is built on Amazon's highly reliable infrastructure, and has tuned Hadoop's performance specifically for Amazon's infrastructure environment.
- Amazon Elastic MapReduce is designed to integrate easily with other AWS services such as Amazon S3 and EC2.
- Secure and inexpensive.

@Zoran B. Djordjevic

41

Pricing

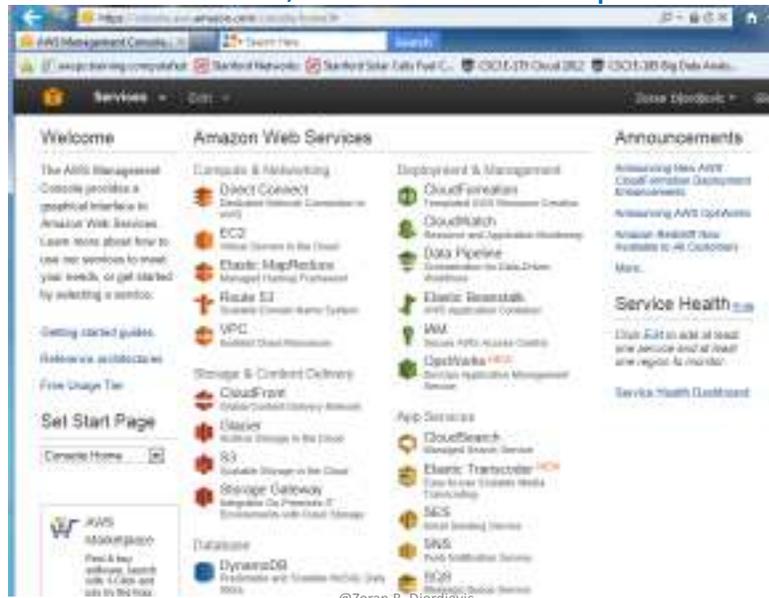
Region: <input type="text" value="US East (N. Virginia)"/>		
Standard On-Demand Instances	Amazon EC2 Price	Amazon Elastic MapReduce Price
Small (Default)	\$0.065 per hour	\$0.015 per hour
Large	\$0.26 per hour	\$0.06 per hour
Extra Large	\$0.52 per hour	\$0.12 per hour
Hi-Memory On-Demand Instances		
Extra Large	\$0.45 per hour	\$0.09 per hour
Double Extra Large	\$0.90 per hour	\$0.21 per hour
Quadruple Extra Large	\$1.80 per hour	\$0.42 per hour
Hi-CPU On-Demand Instances		
Medium	\$0.165 per hour	\$0.03 per hour
Extra Large	\$0.66 per hour	\$0.12 per hour
Cluster Compute On-Demand Instances		
Quadruple Extra Large	\$1.30 per hour	\$0.27 per hour
Cluster Compute Eight Extra Large	\$2.40 per hour	\$0.50 per hour
Cluster GPU On-Demand Instances		
Quadruple Extra Large	\$2.10 per hour	\$0.42 per hour

- EMR prices are atop of instance prices.
- Amazon EC2, Amazon S3 and Amazon SimpleDB charges are billed separately.

@Zoran B. Djordjevic

42

AWS Services, Select Elastic Map Reduce

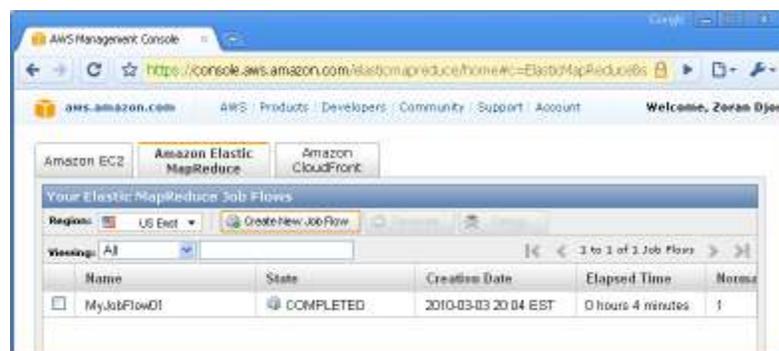


@Zoran B. Djordjevic

43

Create a new Job Flow

- Login into AWS Management Console
- Select Elastic MapReduce service
- Click on Create New Job Flow



@Zoran B. Djordjevic

44

Select Type and Number of Instances

Create a New Job Flow Cancel X

DEFINE JOB FLOW | SPECIFY PARAMETERS | **CONFIGURE EC2 INSTANCES** | ADVANCED OPTIONS | BOOTSTRAP ACTIONS | REVIEW

Specify the master, core and task nodes to run your job flow. For more than 20 instances, complete the limit request form.

Master Instance Group: This EC2 instance assigns Hadoop tasks to core and task nodes and monitors their status.

Instance Type: Request Spot Instance

Core Instance Group: These EC2 instances run Hadoop tasks and store data using the Hadoop Distributed File System (HDFS). Recommended for capacity needed for the life of your job flow.

Instance Count: Instance Type: Request Spot Instance

Task Instance Group (Optional): These EC2 instances run Hadoop tasks, but do not persist data. Recommended for capacity needed on a temporary basis.

Instance Count: Instance Type: Request Spot Instance

[Back](#) **Continue** * Required field

@Zoran B. Djordjevic 47

Select your Key and Logging Options

Create a New Job Flow Cancel X

DEFINE JOB FLOW | SPECIFY PARAMETERS | CONFIGURE EC2 INSTANCES | **ADVANCED OPTIONS** | BOOTSTRAP ACTIONS | REVIEW

Here you enter advanced details about your job flow, such as an EC2 key pair, to use VPC, and your job flow debugging options.

Amazon EC2 Key Pair:
 Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop".

Amazon VPC Subnet ID:
 To run this job flow in a Virtual Private Cloud (VPC), select a subnet. See [Create a VPC](#).

Configure your logging options. [Learn more](#).

Amazon S3 Log Path:
 Optional: To copy log files from the job flow to Amazon S3, specify an Amazon S3 bucket.

Enable Debugging: Yes No
 Yes means EMR will store an index of your logs (requires an Amazon S3 Log Path).

Set advanced job flow options.

Keep Alive Yes No Yes means the job flow will keep running after processing is complete.

Termination Protection Yes No Yes prevents your nodes from shutting down due to accident or error.

Visible To All IAM Users Yes No Yes means the job flow will be visible to all IAM users under your account.

[Back](#) **Continue** * Required field

@Zoran B. Djordjevic 48

Setup Bootstrap Actions

Create a New Job Flow Cancel

3/1 THE JOB FLOW 2/2 STEP 2 PARAMETERS 1/3 CONFIGURE EL2 INSTANCES 4/4 ADVANCED OPTIONS 5/5 **BOOTSTRAP ACTIONS** 6/6 REVIEW

Proceed with no Bootstrap Actions

I do not want to associate any Bootstrap Actions with this Job Flow.

NOTE: Bootstrap Actions must be associated with a Job Flow upon creation. You will not be able to add these later without creating a new Job Flow.

Configure your Bootstrap Actions

Back Continue * Required field

- Bootstrap actions allow you to pass a reference to a script stored in Amazon S3. This script can contain configuration settings and arguments related to Hadoop or Elastic MapReduce. Bootstrap actions are run before Hadoop starts and before the node begins processing data. Actions are like: Install software on the node, Modify the default Hadoop site configuration, Change the way Java parameters use Hadoop daemons
- You can specify up to 16 bootstrap actions per job flow by providing multiple `--bootstrap-action` parameters from the CLI or API.

@Zoran B. Djordjevic

49

Job need a few minutes to start

Your Elastic MapReduce Job Flows

Regions: US East Create New Job Flow Terminate Show Help Refresh Help

Name	State	Creation Date	Elapsed Time	Normalized Instance Ho
ApacheLogSample	STARTING	2010-03-08 11:23 EST	0 hours 0 minutes	0
MyJobFlow01	COMPLETED	2010-03-03 20:04 EST	0 hours 4 minutes	1

1 Job Flow selected

Id: j-3FG9OPHR7V6V7 **Creation Date:** 2010-03-08 11:23 EST

Name: ApacheLogSample **Start Date:** -

State: STARTING **End Date:** -

Last State Change Reason: Starting instances

Availability Zone: us-east-1b **Instance Count:** 4

© 2008 - 2010, Amazon Web Services LLC or its affiliates. All rights reserved. Feedback Support Privacy Policy Terms of Use

An amazon.com company

@Zoran B. Djordjevic

50

Job is Running, Shutting Down

The screenshot shows the Elastic MapReduce console interface. At the top, there's a navigation bar with 'Create New Job Flow', 'Terminate', and 'Debug' buttons. Below that, a table lists job flows. The 'Pig Interactive Job flow' is highlighted and its state is 'RUNNING'. Below the job flow list, there are tabs for 'Description', 'Steps', 'Bootstrap Actions', 'Instance Groups', and 'Monitoring'. The 'Instance Groups' tab is active, showing a table with the following data:

Instance Group Id	Role	Instance Type	State	Market	Bid Price	Running Count	Request Count	Creation DateTime	Last State Change
ig-1J45E6[7U4YV]	MASTER	m1.small	RUNNING	ON_DEMAND	-	1	1	2012-11-16 13:42 EST	-
ig-3FX82GSEFOHP	CORE	m1.small	RUNNING	ON_DEMAND	-	1	1	2012-11-16 13:42 EST	-

At the bottom of the console, there are tabs for 'Description', 'Steps', 'Bootstrap Actions', 'Instance Groups', and 'Monitoring'. The 'Instance Groups' tab is selected.

@Zoran B. Djordjevic 51

Job Completed, Results

The screenshot shows the Elastic MapReduce console interface. At the top, there's a navigation bar with 'Create New Job Flow', 'Terminate', and 'Debug' buttons. Below that, a table lists job flows. The 'Pig Interactive Job flow' is highlighted and its state is 'TERMINATED'. Below the job flow list, there are tabs for 'Description', 'Steps', 'Bootstrap Actions', 'Instance Groups', and 'Monitoring'. The 'Instance Groups' tab is active, showing a table with the following data:

Instance Group Id	Role	Instance Type	State	Market	Bid Price	Running Count	Request Count	Creation DateTime	Last State Change
ig-1J45E6[7U4YV]	MASTER	m1.small	TERMINATED	ON_DEMAND	-	1	1	2012-11-16 13:42 EST	-
ig-3FX82GSEFOHP	CORE	m1.small	TERMINATED	ON_DEMAND	-	1	1	2012-11-16 13:42 EST	-

At the bottom of the console, there are tabs for 'Description', 'Steps', 'Bootstrap Actions', 'Instance Groups', and 'Monitoring'. The 'Instance Groups' tab is selected.

On the right side of the screenshot, there is a terminal window showing the output of a command. The output is a list of search terms and their counts:

```

value 625
views 426
login 224
search 195
items 112
bigtable 68
google+bigtable 59
?29?2?2Fusr?2Fbin?2Fper1+?w 46
philmont+pictures 45
?29?2?2Fusr?2Fbin?2Fper1 44
philmont 37
poc+instrument 33
google+quick+links 33
pig 30
walla 28
vegas 27
about+me+website 26
pig+the+pc+nerd 26
homemade 24
Fishing 23
comments 23
travis 21
hadoop+0.20 21
google+bigtable 21
seattle 19
miscellaneous 19
mnb@ball+ip06 19
pebble 19
biking 18
seahawks 18
andre+carroll 14
    
```

Below the terminal window, there is a text box containing the following text:

top_50_search_terms from file part-r-00000 in directory top_50_search_terms_from_bing_google

@Zoran B. Djordjevic 52

We can control Job Flow thru EMR Command Line

- Download and install Ruby 1.8.7
 - http://rubyforge.org/frs/?group_id=167&release_id=28426
 - Select rubyinstaller-1.8.7-p398-rc2.exe, perhaps.
 - On Linux, do: \$ sudo apt-get install ruby
- Download elastic-mapreduce-client.zip from
- <http://aws.amazon.com/developertools/2264>
- Unzip into c:\elastic-mapreduce-client
- Add C:\elastic-mapreduce-ruby;C:\Ruby\bin; to your PATH.
- In the above directory create file credentials.json and add:


```
{
  "access_id": "<insert your aws Access Key Id here>",
  "private_key": "<insert your aws Secret Access Key here>",
  "keypair": "<insert path of your amazon ec2 Key Pair file>",
  "log_uri": "s3://name of a bucket in s3 to place logs from job"
}
```

@Zoran B. Djordjevic

53

credentials.json

- Be careful with the content of this file. It must be right.

```
{
  "access_id": "AKGGGGHJT7WWWFDTHJQ",
  "private_key": "gU1aTrEwIrQBsyqh3w6253422cK+FlUeRtBWE",
  "keypair": "ec2_hu",
  "key-pair-file": "C:\AWS\hu\ec2_hu.pem",
  "log_uri": "s3n://zoran1116log01/",
  "region": "us-east-1"
}
```

- .

@Zoran B. Djordjevic

54

Examples of Command Line Usage

Listing Active Job Flows

```
ruby elastic-mapreduce --list
ruby elastic-mapreduce --list --active
ruby elastic-mapreduce --list --all
# create a job flow that requires manual termination
ruby elastic-mapreduce --create --alive
```

To create a job flow that will run a mapper written in python, all one line

```
ruby elastic-mapreduce --create --stream -input \
s3://elasticmapreduce/samples/wordcount/input \
--mapper \
s3://elasticmapreduce/samples/wordcount/wordSplitter.py \
--output s3://zoranbucket01
```

Bucket needs to be there but the output folder may not exist before the command is run. If folder is present you get an error.

To terminate all active job flows

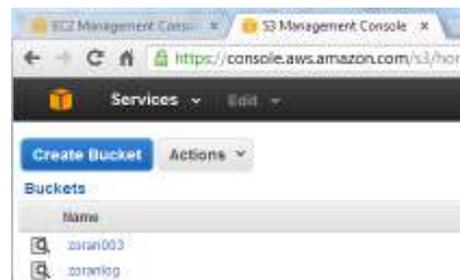
```
ruby elastic-mapreduce --list --active --terminate
# terminate a running job flow
ruby elastic-mapreduce --terminate --jobflow j-2WSXRVDHH08T1
```

@Zoran B. Djordjevic

55

Hadoop Streaming

- Rather than writing scripts for Hadoop in a special language or writing jobs in Java, which is Hadoop's native language, you can write your Map and Reduce routines in almost any language and use a utility called Hadoop Streaming to run them.
- We will demonstrate Hadoop Streaming using a provided example.
- We will need an S3 output bucket or a folder for outputs and perhaps a bucket or a folder for logs
- I created buckets `zoran003` and `zoranlog`

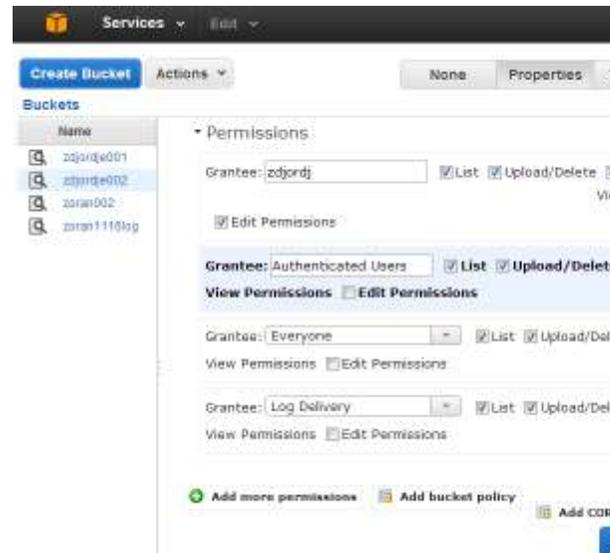


@Zoran B. Djordjevic

56

Add S3 Permissions to all Authenticated Users

- Right click on your bucket and grant permissions to Authenticated users, Everyone and Log Delivery.
- You are a bit more generous than necessary.

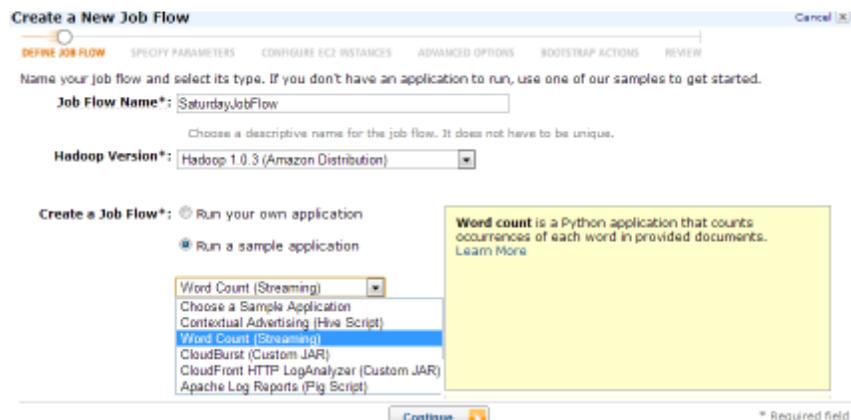


@Zoran B. Djordjevic

57

Create New Job Flow, Streaming Job, Word Count

- Next we go to the Elastic Map Reduce service and create a job flow
- Name your job and select Word Count, a Streaming Application



@Zoran B. Djordjevic

58

Redirect the output to our Bucket

- We change the Output Location to our bucket. Just replace the bucket name, leave folders unchanged. Hit Continue

Create a New Job Flow Cancel

DEFINE JOB FLOW **SPECIFY PARAMETERS** CONFIGURE EC2 INSTANCES ADVANCED OPTIONS BOOTSTRAP ACTIONS REVIEW

Specify the mapper and reducer functions for the job flow. These can be specified either as the name of a predefined streaming command in Hadoop, or you can upload your own code for the mapper and reducer functions to Amazon S3. The format for specifying a location in Amazon S3 is bucket_name/path_name. The location should point to an executable script or binary.

Input Location*: elasticmapreduce/samples/wordcount/input
The Amazon S3 bucket that contains the input files.

Output Location*: zoran003/wordcount/output/2013-02-23
The Amazon S3 bucket that receives the output files. This URL should be unique across all your job flows.

Mapper*: elasticmapreduce/samples/wordcount/wordSplitter.py
The Amazon S3 location of the map function or the name of the Hadoop streaming command to run.

Reducer*: aggregate
The Amazon S3 location of the reduce function or the name of the Hadoop streaming command to run.

Extra Args: Pass extra arguments to the Hadoop streaming program.

[Back](#) [Continue](#) * Required field

@Zoran B. Djordjevic

59

Configure Instances

- These are tiny jobs always chose smallest numbers of everything. Continue.

Create a New Job Flow Cancel

DEFINE JOB FLOW SPECIFY PARAMETERS **CONFIGURE EC2 INSTANCES** ADVANCED OPTIONS BOOTSTRAP ACTIONS REVIEW

Specify the master, core and task nodes to run your job flow. For more than 20 instances, complete the limit request form.

Master Instance Group: This EC2 instance assigns Hadoop tasks to core and task nodes and monitors their status.

Instance Type: Small (m1.small) Request Spot Instance

Core Instance Group: These EC2 instances run Hadoop tasks and store data using the Hadoop Distributed File System (HDFS). Recommended for capacity needed for the life of your job flow.

Instance Count: 2

Instance Type: Small (m1.small) Request Spot Instances

Task Instance Group (Optional): These EC2 instances run Hadoop tasks, but do not persist data. Recommended for capacity needed on a temporary basis.

Instance Count: 0

Instance Type: Small (m1.small) Request Spot Instances

[Back](#) [Continue](#) * Required field

@Zoran B. Djordjevic

60

Provide your key pair and log folder

Create a New Job Flow Cancel [X]

DERIVE JOB FLOW | SPECIFY PARAMETERS | CONFIGURE EC2 INSTANCES | **ADVANCED OPTIONS** | BOOTSTRAP ACTIONS | REVIEW

Here you enter advanced details about your job flow, such as an EC2 key pair, to use VPC, and your job flow debugging options.

Amazon EC2 Key Pair:

Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop".

Amazon VPC Subnet ID:

To run this job flow in a Virtual Private Cloud (VPC), select a subnet. See [Create a VPC](#).

Configure your logging options. [Learn more](#).

Amazon S3 Log Path:

Optional: To copy log files from the job flow to Amazon S3, specify an Amazon S3 bucket.

Enable Debugging: Yes No
Yes means EMR will store an index of your logs (requires an Amazon S3 Log Path).

Set advanced job flow options.

Keep Alive Yes No Yes means the job flow will keep running after processing is complete.

Termination Protection Yes No Yes prevents your nodes from shutting down due to accident or error.

Visible To All IAM Users Yes No Yes means the job flow will be visible to all IAM users under your account.

< Back Continue [Next] * Required field

- Change log path. Select EC2 Key Pair. Notice **Keep Alive** is YES. Leave it.
- Continue and then Proceed with no Bootstrap Actions

@Zoran B. Djordjevic 61

Review your settings and Create Job Flow

Create a New Job Flow Cancel [X]

DERIVE JOB FLOW | SPECIFY PARAMETERS | CONFIGURE EC2 INSTANCES | ADVANCED OPTIONS | BOOTSTRAP ACTIONS | **REVIEW**

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: SaturdayJobFlow
Type: Word Count (Streaming) [Edit Job Flow Definition](#)

Input Location: s3n://elasticmapreduce/samples/wordcount/input
Output Location: s3n://zoran003/wordcount/output/2013-02-23
Mapper: s3n://elasticmapreduce/samples/wordcount/wordSplitter.py
Reducer: aggregate
Extra Args: [Edit Job Flow Parameters](#)

Master Instance Type: m1.small **Instance Count:** 1
Core Instance Type: m1.small **Instance Count:** 2 [Edit EC2 Configs](#)

Amazon EC2 Key Pair: ec2_hu
Amazon Subnet Id:
Amazon S3 Log Path: s3n://zoranlog
Enable Debugging: Yes **Keep Alive:** Yes
Termination Protected: No **Visible To All Users:** No [Edit Advanced Options](#)

Bootstrap Actions: No Bootstrap Actions created for this Job Flow [Edit Bootstrap Actions](#)

< Back Create Job Flow [Next]

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

@Zoran B. Djordjevic 62

Watch the job flow

- Your jobs will take a few minutes to start, run, wait or fail

The image shows three sequential screenshots of the AWS Elastic MapReduce console. Each screenshot displays a table of job flows. The first screenshot shows the job flow in the 'STARTING' state. The second screenshot shows the job flow in the 'RUNNING' state. The third screenshot shows the job flow in the 'WAITING' state.

Name	State	Creation Date	Elapsed Time	Normalized Instance Hours
SaturdayJobFlow	STARTING	2013-02-23 14:02 EST	0 hours 0 minutes	0
SaturdayJobFlow	RUNNING	2013-02-23 14:02 EST	0 hours 1 minute	3
SaturdayJobFlow	WAITING	2013-02-23 14:02 EST	0 hours 4 minutes	3

@Zoran B. Djordjevic

63

Once Job Flow is Waiting you are Done

- Once the job flow moves to Waiting state you can go to your S3 buckets and examine the results

The image shows a screenshot of the AWS S3 Management Console. The browser address bar shows the URL: <https://console.aws.amazon.com/s3/home?region=us-east-1>. The console displays a list of files in the bucket 'zorano03 / wordcount / output / 2013-02-23'.

Name	Storage Class	Size	Last Modified
_SUCCESS	Standard	0 bytes	Sat Feb 23 14:11:15 GMT-000 2013
part-00000	Standard	97.3 KB	Sat Feb 23 14:11:00 GMT-000 2013
part-00001	Standard	98.6 KB	Sat Feb 23 14:10:58 GMT-000 2013
part-00002	Standard	97.1 KB	Sat Feb 23 14:11:11 GMT-000 2013

- You can download any of the files, like part-00000 and read the word count. Result is presented on the

@Zoran B. Djordjevic

64

Word Count Output

```

part-00000 (C:\CLASS-1 of word count) - GVM
File Edit Tools Syntax Buffers Window Help
bypass 3
byproduct 1
byr 6
bystrica 3
c1425gm 3
cabanatusan 3
cabbage 6
cabbages 6
cabin 3
cabinet 1106
cable 676
cabet 1
cabreera 12
cacao 10
cacn 17
cactus 12
cadets 3
caguet 3
caborra 6
caixa 3
1271,1 12%

```

@Zoran B. Djordjevic

65

Select EC2 Service and Review State of Your Instance

- Select EC2 Service first and then on the EC2 Dashboard, select Instances

The screenshot shows the AWS Management Console interface. The 'Services' menu is open, and 'EC2' is selected. The 'EC2 Dashboard' is displayed, showing the 'INSTANCES' link highlighted with a red box. The 'Resources' section shows the following statistics for the US East (N. Virginia) region:

Resource Type	Count
Running Instances	0
Elastic IPs	0
Volumes	0
Snapshots	1

66

Instances Supporting the Job Flow

- There is always a master instance and as many slave instances

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with 'Viewing: Running Instances' and 'All Instance Types'. Below that is a table of instances. The table has columns for Name, Instance, AMI ID, Root Dev, Zone, Type, State, Status Checks, Alarm St, Monitor, Security, Key Pair, and Virtualiz. Three instances are listed, all with a state of 'running'. The first instance is 'empty' with ID 'i-1ac7f', the second is 'empty' with ID 'i-18c7f', and the third is 'empty' with ID 'i-14ce'. The third instance is selected, and its details are shown below the table. The details include the instance ID 'i-14ce7f67', the public DNS name 'ec2-23-23-57-236.compute-1.amazonaws.com', and various configuration parameters like AMI, Zone, Type, and Security Groups.

Name	Instance	AMI ID	Root Dev	Zone	Type	State	Status Cl	Alarm St	Monitor	Security	Key Pair	Virtualiz
empty	i-1ac7f	ami-e543i	instance	us-east-1	m1.small	running	2/2 c	none	basic	ElasticMc	ec2_hu	paravirtua
empty	i-18c7f	ami-e543i	instance	us-east-1	m1.small	running	2/2 c	none	basic	ElasticMc	ec2_hu	paravirtua
empty	i-14ce	ami-e543i	instance	us-east-1	m1.small	running	2/2 c	none	basic	ElasticMc	ec2_hu	paravirtua

1 EC2 Instance selected.

EC2 Instance: i-14ce7f67
ec2-23-23-57-236.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

AMI: Amazon Elastic MapReduce 2013-02-04-17-11-31-32 pvm/s3 (ami-e543d48c) | Alarm Status: none

Zone: us-east-1b | Security Groups: ElasticMapReduce-master, view rules

Type: m1.small | State: running

Scheduled Events: No scheduled events | Owner: 951414138794

VPC ID: - | Subnet ID: -

Source / Dest. Check: - | Virtualization: paravirtua

Source / Dest. Check: - | Subnet ID: - | Virtualization: paravirtua

@Zoran B. Djordjevic

67

Let us SSH to the Master Instance

- In the instance view, right click on the master instance and select Connect. From the wizard that pops up copy the public DNS name of your instance. In my case the DNS name was:

ec2-23-23-57-236.compute-1.amazonaws.com

- Open Cygwin window in the folder

```
zdzordjr@zdzordjr-PC /cygdrive/c/AWS/hu
```

- where you keep your key pair (ec2_hu.pem in my case) and type:

```
$ ssh -i ec2_hu.pem hadoop@ec2-23-23-57-236.compute-1.amazonaws.com
```

- You will get a Linux prompt in the home directory of hadoop user:

```
hadoop@domU-12-31-39-00-69-A7:~$ pwd
```

```
/home/hadoop
```

- On the Linux prompt you can run standard Linux (Unix) commands. Since you are not a root, you might have to run your commands as a sudo user. Just prefix your commands with `sudo`.

@Zoran B. Djordjevic

68

Hadoop Environment

- `hadoop` is also an executable which actually controls your cluster.
You can for example, run the following Linux commands

```
hadoop@domU-12-31-39-00-69-A7:~$ pwd
/home/hadoop
hadoop@domU-12-31-39-00-69-A7:~$ which hadoop
/home/hadoop/bin/hadoop
hadoop@domU-12-31-39-00-69-A7:~$ which java
/usr/bin/java
hadoop@domU-12-31-39-00-69-A7:~$ ls
PATCHES.txt          hadoop-core-1.0.3.jar      hadoop-tools.jar
bin                   hadoop-core.jar           lib
conf                  hadoop-examples-1.0.3.jar lib64
contrib              hadoop-examples.jar       libexec
etc                   hadoop-minicluster-1.0.3.jar native
hadoop-ant-1.0.3.jar  hadoop-test-1.0.3.jar    sbin
. . . . .
```

@Zoran B. Djordjevic

69

Distributed File System, `dfs` command

- Hadoop has access not only to the local, Linux, file system. It also has its own distributed file system (HDFS Hadoop Distributed File System)
- We access that file system through hadoop file system shell, `dfs`.
Type
\$ `hadoop dfs`
- and you will get a long list of options. We will present those options on the next slide. Some of those resemble Unix (Linux) commands. Some are different.
- We use those commands to create directories in the HDFS, copy files between HDFS and the local file system, Internet and AWS S3 buckets.
- When you use `dfs`, you always prefix it with `hadoop`.

@Zoran B. Djordjevic

70

File system shell `dfs`

```
hadoop@domU-12-31-39-00-69-A7:~$ hadoop dfs
Usage: java FsShell
    [-ls <path>]
    [-lsr <path>]
    [-du <path>]
    [-dus <path>]
    [-count[-q] <path>]
    [-mv <src> <dst>]
    [-cp <src> <dst>]
    [-rm [-skipTrash] <path>]
    [-rmr [-skipTrash] <path>]
    [-expunge]
    [-put <localsrc> ... <dst>]
    [-copyFromLocal <localsrc> ... <dst>]
    [-moveFromLocal <localsrc> ... <dst>]
    [-get [-ignoreCrc] [-crc] <src> <localdst>]
    [-getmerge <src> <localdst> [addnl]]
    [-cat <src>]
    [-text <src>]
    [-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
    [-moveToLocal [-crc] <src> <localdst>]
```

@Zoran B. Djordjevic

71

File system shell `dfs`

```
[-moveToLocal [-crc] <src> <localdst>]
[-mkdir <path>]
[-setrep [-R] [-w] <rep> <path/file>]
[-touchz <path>]
[-test -[ezd] <path>]
[-stat [format] <path>]
[-tail [-f] <file>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-help [cmd]]
```

Generic options supported are

```
-conf <configuration file>    specify an application configuration file
-D <property=value>          use value for given property
-fs <local|namenode:port>    specify a namenode
-jt <local|jobtracker:port>  specify a job tracker
-files <comma separated list of files> specify comma separated files to
be copied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar
files to include in the classpath.
```

@Zoran B. Djordjevic

72

File system shell `dfs`

`-libjars <comma separated list of jars>` specify comma separated jar files to include in the classpath.

`-archives <comma separated list of archives>` specify comma separated archives to be unarchived on the compute machines.

The general command line syntax is

`bin/hadoop command [genericOptions] [commandOptions]`

- For example, we can use `dfs` to fetch the Python script used in our job flow:

```
$ hadoop dfs -copyToLocal\
s3://elasticmapreduce/samples/wordcount/wordSplitter.py .
```

- The last dot (`.`) on the line is significant. This is “this” directory.
- Notice that options following `dfs` shell always start with a dash.
- Examine local directory and see the local copy of `wordSplitter.py`

```
hadoop@domU-12-31-39-00-69-A7:~$ ls -la wordSplitter.py
-rw-r--r-- 1 hadoop hadoop 294 Feb 23 19:50 wordSplitter.py
```

@Zoran B. Djordjevic

73

`wordSplitter.py`

- We can `vi` the Python script or transfer it to our local Windows or Mac terminal and discover that it reads like:

```
#!/usr/bin/python
import sys
import re

def main(argv):
    pattern = re.compile("[a-zA-Z][a-zA-Z0-9]*")
    for line in sys.stdin:
        for word in pattern.findall(line):
            print "LongValueSum:" + word.lower() + "\t" + "1"

if __name__ == "__main__":
    main(sys.argv)
```

@Zoran B. Djordjevic

74

Input Data

- We could similarly use good services of hadoop's distributed file system shell `dfs` and first examine the input folder and then fetch the input data we used in the job flow.

```
hadoop@domU-12-31-39-00-69-A7:~$ hadoop dfs -ls
s3://elasticmapreduce/samples/wordcount/input
Found 12 items
-rwxrwxrwx 1 2392524 2009-04-02 02:55 /samples/wordcount/input/0001
-rwxrwxrwx 1 2396618 2009-04-02 02:55 /samples/wordcount/input/0002
-rwxrwxrwx 1 1593915 2009-04-02 02:55 /samples/wordcount/input/0003
-rwxrwxrwx 1 1720885 2009-04-02 02:55 /samples/wordcount/input/0004
-rwxrwxrwx 1 2216895 2009-04-02 02:55 /samples/wordcount/input/0005
-rwxrwxrwx 1 1906322 2009-04-02 02:55 /samples/wordcount/input/0006
-rwxrwxrwx 1 1930660 2009-04-02 02:55 /samples/wordcount/input/0007
-rwxrwxrwx 1 1913444 2009-04-02 02:55 /samples/wordcount/input/0008
-rwxrwxrwx 1 2707527 2009-04-02 02:55 /samples/wordcount/input/0009
-rwxrwxrwx 1 327050 2009-04-02 02:55 /samples/wordcount/input/0010
-rwxrwxrwx 1 8 2009-04-02 02:55 /samples/wordcount/input/0011
-rwxrwxrwx 1 8 2009-04-02 02:55 /samples/wordcount/input/0012
```

@Zoran B. Djordjevic

75

Copy input file 0001 to Local File System

```
hadoop@domU-12-31-39-00-69-A7:~$ hadoop dfs -copyToLocal
s3://elasticmapreduce/samples/wordcount/input/0001 .
13/02/23 20:06:21 INFO s3native.NativeS3FileSystem: Opening
's3://elasticmapreduce/samples/wordcount/input/0001' for
reading
hadoop@domU-12-31-39-00-69-A7:~$ ls -la 0001
-rw-r--r-- 1 hadoop hadoop 2392524 Feb 23 20:06 0001
hadoop@domU-12-31-39-00-69-A7:~$
```

- Unix (Linux) utilities `tail` and `head` will tell us what are the lines at the end and beginning of file `001`.

@Zoran B. Djordjevic

76

tail 0001, head 0001

```

hadoop@domU-12-31-39-00-69-A7:~$ tail 0001
                                males age 16-49: 7,322,965
                                females age 16-49: 6,859,064 (2008 est.)
                                Manpower fit for military service:
                                    males age 16-49: 4,886,103
                                    females age 16-49: 5,525,764 (2009 est.)
                                Manpower reaching militarily significant
age annually:
                                male: 365,567
                                female: 352,643 (2009 est.)
                                Military expenditures:
                                1.6% of GDP (2006)
hadoop@domU-12-31-39-00-69-A7:~$ head 0001
CIA -- The World Factbook -- Country Listing
    World Factbook Home
    The World Factbook
    &nbsp;
    Country Listing . . .

```

@Zoran B. Djordjevic

77

Terminate all Instances

- Since we had enough for the day, we should terminate all instances, so that we stop incurring additional charges.
- Select all instances, and under Actions select Terminate, and Yes Terminate.

