

Spark on Hadoop Multinode Cluster

Here I'm recording critical steps on achieving movie recommendation on spark on multiple AWS EC2 instances. Totally there are 3 primary steps as following:

1.Set up multi node hadoop cluster on 4 ubuntu instances:

- Namenode (master)
- SecondaryNamenode (back up)
- Datanode (slave 1)
- Datanode (slave 2)

2. Installing spark and run pyspark in both 'local' and 'yarn' mode

3. Achieving movie recommendation with Spark Mllib ALS algorithm

Here we go.

1.Set up multi node hadoop cluster on 4 ubuntu instances:

1.1 Choosing AWS EC2 Ubuntu Server 18.04

The screenshot shows the AWS Management Console interface for creating an EC2 instance. The 'Step 1: Choose an Amazon Machine Image (AMI)' screen is active. On the left, there is a sidebar with 'My AMIs', 'AWS Marketplace', and 'Community AMIs'. The 'Free tier only' filter is selected. The main area displays a list of AMIs:

- Amazon Linux 2 AMI (HVM), SSD Volume Type** - ami-01e3b8c3a51e88954 (64-bit x86) / ami-0871ae9e379c8be5b (64-bit Arm). It includes Linux kernel 4.14, GCC 7.3, and Binutils 2.26.
- Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type** - ami-0080e4c5bc078760e. It includes AWS command line tools, Python, Ruby, Perl, and Java.
- SUSE Linux Enterprise Server 15 (HVM), SSD Volume Type** - ami-06ea7729e394412c8. It includes Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules.
- Red Hat Enterprise Linux 7.6 (HVM), SSD Volume Type** - ami-011b3ccf1bd6db744 (64-bit x86) / ami-0e3688b4a755ad736 (64-bit Arm). It includes Red Hat Enterprise Linux version 7.6.
- Ubuntu Server 18.04 LTS (HVM), SSD Volume Type** - ami-0ac019f4fcb7cb7e6 (64-bit x86) / ami-01ac7d9c1179d7b74 (64-bit Arm). It includes Ubuntu Server 18.04 LTS.

The 'Ubuntu Server 18.04 LTS' AMI is selected. At the bottom, there is a prompt: 'Are you launching a database instance? Try Amazon RDS.' The footer of the console shows 'Feedback', 'English (US)', and copyright information for 2008-2019.

1.2 Configure 4 instances

console.aws.amazon.com

aws Services Resource Groups

Xiaodan N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances

4

Launch into Auto Scaling Group

Purchasing option

☐ Request Spot instances

Network

vpc-a7d63f3c (default)

Create new VPC

Subnet

No preference (default subnet in any Availability Zone)

Create new subnet

Auto-assign Public IP

Use subnet setting (Enable)

Placement group

☐ Add instance to placement group

Capacity Reservation

Open

Create new Capacity Reservation

IAM role

None

Create new IAM role

Shutdown behavior

Stop

Enable termination protection

☐ Protect against accidental termination

Monitoring

☐ Enable CloudWatch detailed monitoring

Additional charges apply.

Tenancy

Shared - Run a shared hardware instance

Additional charges will apply for dedicated tenancy.

Elastic Inference

☐ Add an Elastic Inference accelerator

Additional charges apply.

Cancel

Previous

Review and Launch

Next: Add Storage

Feedback

English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

1.3 Configure security group

console.aws.amazon.com

aws Services Resource Groups

Xiaodan N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:

☒ Create a new security group

☐ Select an existing security group

Security group name:

launch-wizard-32

Description:

launch-wizard-32 created 2019-01-14T17:34:45.817-05:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule

Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel

Previous

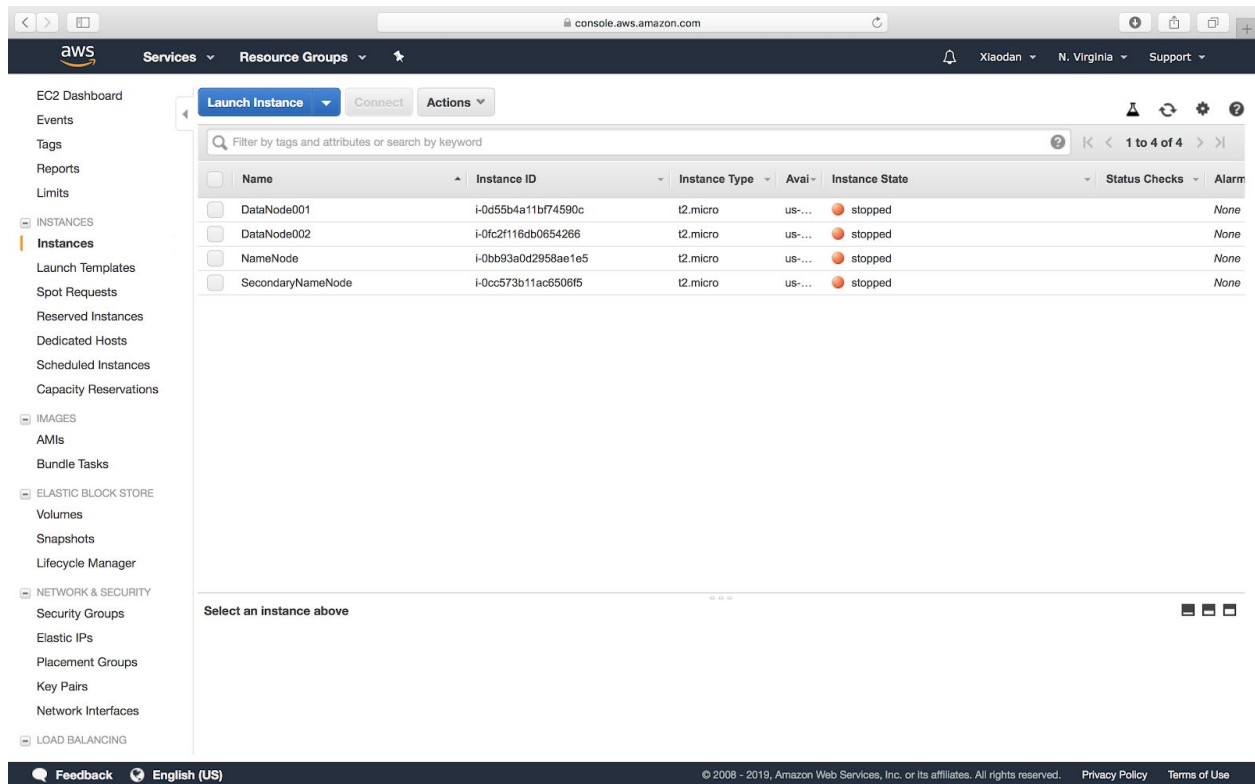
Review and Launch

Feedback

English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

1.4 Rename 4 instances and get it started later on



1.5 SSH to these 4 instances

```
Desktop — ubuntu@ip-172-31-95-26: ~ — ssh -i xiaodanchen.pem ubuntu@ec2-54-234-195-249.compute-1.amazonaws.com
...untu@ip-172-31-95-26: ~ — ssh -i xiaodanchen.pem ubuntu@ec2-54-234-195-249.compute-1.amazonaws.com
Xiaodans-MacBook-Pro:~ xiaodanchen$ cd Desktop
Xiaodans-MacBook-Pro:Desktop xiaodanchen$ ssh -i "xiaodanchen.pem" ubuntu@ec2-54-234-195-249.compute-1.amazonaws.com
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
```

1.6 Pre-installations for the cluster

```
$ sudo apt-get update
# install java
$ sudo add-apt-repository -y ppa:webupd8team/java
$ sudo apt-get -y install oracle-java8-installer
```

```
# install hadoop
$ wget {hadoop from Apache download page}
$ tar -xzvf hadoop-1.2.1.tar.gz
$ mv hadoop-1.2.1 hadoop

# set environment variable
$ sudo vim ~/.bashrc
$ export PATH=$PATH:{}
$ source ~/.bashrc

# ssh
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
$ ll ~/.ssh
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ chmod 644 authorized_keys
# remote ssh
$ eval `ssh-agent -s`
$ ssh-add ~/.xiaodanchen.pem

# hadoop cluster configuration
$ cd ~/hadoop/etc/hadoop
# configure these files:


- Hadoop-env.sh
- Core-site.xml
- Mapred-site.xml
- Hdfs-site.xml



$ sudo vim masters # add two namenodes ip to masters
$ sudo vim slaves # add two datanode ip to slaves
# copy these files to SecondaryNamenode
# for slave 1: leave masters blank, only copy datanode 1 ip tp slaves
# for slave 2: leave masters blank, only copy datanode 2 ip tp slaves
```

```
# hadoop daemon startup
$ hadoop namenode -format
$ start-all.sh # I've already set environment variable
```

```
Desktop — ubuntu@ip-172-31-95-26: ~ — ssh -i xiaodanchen.pem ubuntu@ec2-18-234-102-17...
...te-1.amazonaws.com  ...te-1.amazonaws.com  ...-1.amazonaws.com  ...-1.amazonaws.com  +

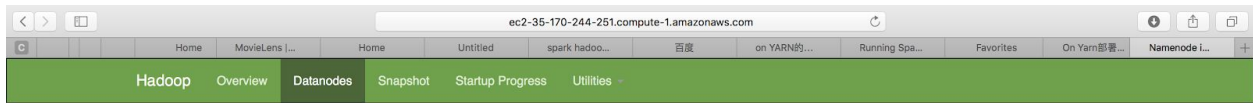
* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

89 packages can be updated.
0 updates are security updates.

Last login: Mon Jan 14 22:47:44 2019 from 24.126.24.184
ubuntu@ip-172-31-95-26:~$ eval `ssh-agent -s`
Agent pid 1228
ubuntu@ip-172-31-95-26:~$ ssh-add ~/xiaodanchen.pem
Identity added: /home/ubuntu/xiaodanchen.pem (/home/ubuntu/xiaodanchen.pem)
ubuntu@ip-172-31-95-26:~$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-namenode-ip-172-31-95-26.out
data1: starting datanode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-84-24.out
data2: starting datanode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-81-42.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-secondary-namenode-ip-172-31-95-26.out
starting yarn daemons
starting resourcemanager, logging to /home/ubuntu/hadoop/logs/yarn-ubuntu-resourcemanager-ip-172-31-95-26.out
data1: starting nodemanager, logging to /home/ubuntu/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-84-24.out
data2: starting nodemanager, logging to /home/ubuntu/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-81-42.out
ubuntu@ip-172-31-95-26:~$ jps
1415 NameNode
2074 Jps
1820 ResourceManager
1677 SecondaryNameNode
ubuntu@ip-172-31-95-26:~$
```

```
Desktop — ubuntu@ip-172-31-84-24: ~ — ssh -i xiaodanchen.pem ubuntu@ec2-3-84-211-20.co...
...te-1.amazonaws.com  ...te-1.amazonaws.com  ...-1.amazonaws.com  ...-1.amazonaws.com  +

Last login: Sun Jan 13 18:36:45 2019 from 24.126.24.184
ubuntu@ip-172-31-84-24:~$ jps
2036 DataNode
2213 NodeManager
2335 Jps
ubuntu@ip-172-31-84-24:~$
```



Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ec2-35-171-160-52.compute-1.amazonaws.com (172.31.81.42:50010)	2	In Service	19.32 GB	15.66 MB	7.76 GB	11.55 GB	25	15.66 MB (0.08%)	0	2.6.0
ec2-52-201-102-64.compute-1.amazonaws.com (172.31.84.24:50010)	2	In Service	19.32 GB	15.66 MB	7.76 GB	11.55 GB	25	15.66 MB (0.08%)	0	2.6.0

Decommissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction
------	--------------	-------------------------	------------------------------	--

Hadoop, 2014.

Legacy UI

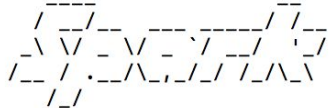
2. Installing spark and run pyspark

```
# install scala
$ wget http://www.scala-lang.org/files/archive/scala-2.11.6.tgz
$ tar xvf scala-2.11.6.tgz
$ sudo mv scala-2.11.6 ~/scala

# set environment variable
$ sudo ~/.bashrc
$ export SCALA_HOME=/home/ubuntu/scala
$ export PATH=$PATH:$SCALA_HOME/bin
$ source ~/.bashrc
```



```
Desktop — ubuntu@ip-172-31-95-26: ~ — ssh -i xiaodanchen.pem ubuntu@ec2-18-234-102-17...  
...mazonaws.com    ...mazonaws.com    ...amazonaws.com    ...zonaws.com    ...zonaws.com  
  
[ubuntu@ip-172-31-95-26:~$ pyspark  
Python 2.7.14 |Anaconda, Inc.| (default, Oct 16 2017, 17:29:19)  
[GCC 7.2.0] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
19/01/15 01:41:22 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform.  
.. using builtin-java classes where applicable  
Welcome to
```



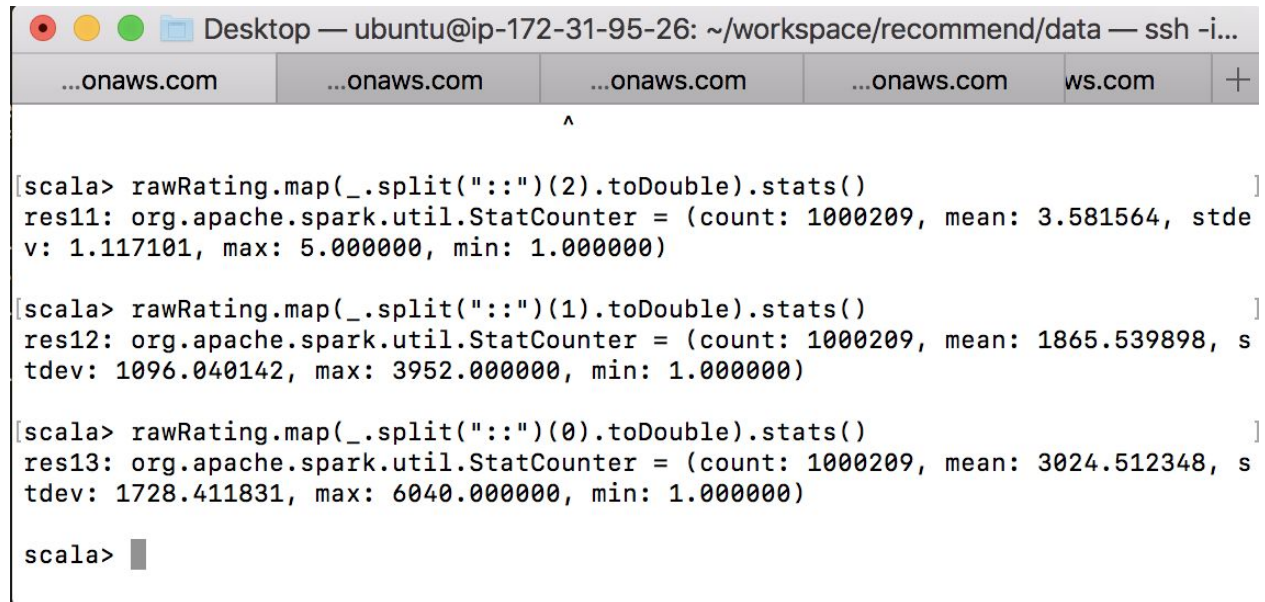
```
version 2.0.0  
  
Using Python version 2.7.14 (default, Oct 16 2017 17:29:19)  
SparkSession available as 'spark'.  
>>>
```

3. Achieving movie recommendation with Spark Mlib ALS algorithm

Here I'd download data from MovieLens, move it to HDFS and see some data stats.

- The first column would be the 'user_id', as we can see, there are 6040 unique users
- The second column would be the 'movie_id' for like unique 3952 movies
- The third column would be 'rating_score', there are like 1 million ratings here
- Other columns are ignored

For the recommendation process, I will put the code in the code file



```
Desktop — ubuntu@ip-172-31-95-26: ~/workspace/recommend/data — ssh -i...
...onaws.com ...onaws.com ...onaws.com ...onaws.com ws.com +
^
[scala> rawRating.map(_.split("::")(2).toDouble).stats()]
res11: org.apache.spark.util.StatCounter = (count: 1000209, mean: 3.581564, stdev: 1.117101, max: 5.000000, min: 1.000000)

[scala> rawRating.map(_.split("::")(1).toDouble).stats()]
res12: org.apache.spark.util.StatCounter = (count: 1000209, mean: 1865.539898, stdev: 1096.040142, max: 3952.000000, min: 1.000000)

[scala> rawRating.map(_.split("::")(0).toDouble).stats()]
res13: org.apache.spark.util.StatCounter = (count: 1000209, mean: 3024.512348, stdev: 1728.411831, max: 6040.000000, min: 1.000000)

scala> █
```