Design and implementation of a parallel algorithm

0/1 Knapsack problem

Aude Laurent , Shah Rishabh
Department of Computer Science,
Illinois Institute of Technology,
Chicago, IL
alaurent1@hawk.iit.edu, rshah92@hawk.iit.edu

ABSTRACT

The CS546 course of the Computer Science Master in the Illinois Institute of Technology aims to give students a throughout knowledge about parallel and distributed systems. It covers general issues of parallel and distributed processing from a user's point of view which includes system architectures, programming, performance evaluation, applications, and the influence of communication and parallelism on algorithm design. A final project concludes the course by making students work about their choice (turned on implementation or research) and put into practice the knowledge gained during the course. Our project consists on designing and implementing a parallel algorithm to solve the knapsack problem. Based on the sequential algorithm, 3 programs have been implemented using MPI, openMP and Hadoop. Finally, test performance has been performed to compare the solutions.

Keywords: Hadoop, OpenMP, MPI, Map Reduce

INTRODUCTION

This project aims to put into practice the knowledge gained during the semester in the Parallel and Distributed Processing course. We choose to work on the 0/1 Knapsack problem, and to implement two solutions using parallelism to solve this problem.

This subject has been chosen since we wanted to work on a programming project to gain skills and practice on parallel implementation methods learnt during the semester. The 0/1 knapsack problem seemed to be a good subject as it was a problem on which neither of us had worked before, so we both were discovering it, and seemed to be understandable and interesting. Finally, we choose two work as a 2-student's team to be able to produce 3 programs and compare their efficiency.

Our work has been divided into several parts. The first one has consisted on analyzing and understanding the 0/1 knapsack problem and find a sequential algorithm. Then, we have worked simultaneously, to design and implement a MPI based solution, openMP based solution and a Hadoop based solution. Finally, the last part has been to create a benchmark to compare and analysis the efficiency of both programs.

This report relates this work and completes the presentation made in class.

RELATED WORK

A significant amount of research has been done on the 0-1 knapsack problem, which has numerous applications in business. First scientists and mathematicians, like Balas or Zemel developed a sequential algorithm. More recently, parallel algorithms have been discussed by a number of people. Several approaches has been studied by mathematicians like large data set problem or CPU based solution. Martello pointed out the impracticality of an FPTAS for the knapsack problem due to this increase in space requirement

Our objective is not to challenge the results of previous researches but to familiarize ourselves with the openMP, MPI and Hadoop design and implementation, and to observe and analyze the consistency of our results.

The sequential algorithm for the problem has been reuse from prior work. Our part of the work has been to implement the sequential program, and to design and implement the 3 solutions we propose.

PROBLEM STATEMENT

1. Knapsack Problem

The knapsack problem or rucksack problem is a problem in combinatorial optimization.

A knapsack can carry a fixed weight. Given a set of items, each with a weight and a value, the objective is to determine the number of item to include in the knapsack so that the total weight is less than or equal to the limit of the knapsack and the total value is as large as possible.

More precisely, we worked on the 0/1 version of the knapsack problem, in which an item is either put is the knapsack or not and, in contrast to the bounded and unbounded variants, cannot be multiplied or divided.

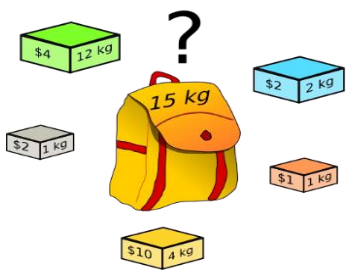


Figure 1: Bag Problem

Source: Wikipedia

2. Sequential algorithm

To understand the sequential algorithm, we need to find a representation of the computations' evolution. To do so, we use a 2-dimension matrix as follow:

	1	2	3	...	Max weight
Item 0					
Item 1					
...					
Item n-1					

Figure 2: Problem representation

Each cell contains the best value achievable:

- using the item of its row and the items above it
- not exceeding the weight of its column

Thus, at the end of the computations, the best value achievable for the input data is found is the cell [item n-1][max weight].

Then, the sequential algorithm is:

Figure 3: Pseudo code for sequential knapsack problem

Source: Wikipedia

```
 1 // Input:
 2 // Values (stored in array v)
 3 // Weights (stored in array w)
 4 // Number of distinct items (n)
 5 // Knapsack capacity (W)
 6
 7 for j from 0 to W do:
 8     m[0, j] := 0
 9
10 for i from 1 to n do:
11     for j from 0 to W do:
12         if w[i] > j then:
13             m[i, j] := m[i-1, j]
14         else:
15             m[i, j] := max(m[i-1, j], m[i-1, j-w[i]] + v[i])
```

We use a 2 for-loops structure to parse the matrix with a row orientation (the loop handling rows is the outer loop).As described above, each cell is filled with the maximum achievable value for its column weight and row (and below) items. So at each iteration, the value is the maximum value between:

- The value computed for the same weight without using the new item. This value is found in [i-1][j] (same column, upper row)

- The value using this item and the best value achievable for the weight: 'column's weight minus item's weight'. This value found in [i-1][j- item's weight].

That is the computation made in the line 15. The computation line 13 is executed if the weight of the considered item is higher than the column weight. In that case, the best value achievable is the value of the row above. The first loop is used to fill the first row with 0 as a reference for the next computations.

At the end of the computations, the best value is found in the cell [item n-1] [max weight]. To know witch items is used, a back sparse should be done.

METHODOLOGY

1. Dependencies analysis

First, our project consists on software parallelism so we didn’t consider hardware parallelism.

While analyzing the above algorithm, we can first conclude that there is no task parallelization realizable. Indeed, all the task depend of each other and we cannot extract some computation that can be done simultaneously.

We can first conclude that we will parallelize our solution using data parallelism. Since, the same computation is done on every cell of the data, this seems to be the more obvious solution.

Then, we need to figure out which data is parallelizable. As we analyze the sequential algorithm, we saw that the value of a cell can either be the value of the cell [i-1] [j] or use the data in the cell [i-1] [j- item’s weight]. We can then conclude that there are dependencies between the rows but not on the columns. Indeed, a cell need to know the value of the rows before itself, but do not need the value of the column of its rows.

So, the code is organizing with a column parallelization. That means that for each row, all the column is performed simultaneously.

2. MPI optimization

The MPI implementation use the column parallelization explained above. Each column of the matrix is mapped to a processor. The mapping is made to evenly divide the work, so each process works on the columns (process number + k*number of process).

For each row, the rank computes the column it is mapped to. At each iteration the execution contains several steps:

1. Compute the maximum value achievable using the item of the row
    a. If the weight of the item is bigger than the column’s weight, the value is 0
    b. If it’s the first item and the If the weight of the item is smaller than the column’s weight, the value is the item’s value
    c. Else, the value id the item value plus the value of the cell [i-1][j-weight[i]]. The rank gets this value thanks to a MPI_Recv from the rank that had compute this value.
2. Compute the value without the new item. This value is the value just above in the matrix (same maximum weight withut the new item) or 0 if it is the first item.
3. Save in the cell the maximum value achievable using or not the new item
4. Send to all the processers that could need it in future iteration the new value.

This implementation is quite optimized since, it does require any “waiting time”. Indeed, there is no need for MPI_Barrier, sending are non-synchronous and receiving is fast since the data must have been send before.

3. OpenMP Optimization

OpenMP optimization uses the column parallelization. Each column of the matrix is mapped to a processor. The mapping is made to evenly divide the work, so each process works on the columns (process number + k*number of process).

There are several steps in execution of OpenMP version.

- First, a new instance is generated for the knapsack problem. System ask for the user to enter the four parameters `N`: Total number of items. `max_`: The maximum weight and value of an item. `w`: Array of weights. v`: Array of values.
- As shown above in the code here we tried to parallelize the calculation of the every column. So once all the weights are calculated column wise then only thing left is to compare all the weights and find out the best profit.
- The main improvement over the sequential version is that it only uses an array of two rows because for every element to be computed, we only need two elements from the previous row.
- Second method is use to get the total weight and calculate the total profit.

4. Hadoop Optimization

The first step has been to set up the cluster. Please refer to the annexes for the described steps of the cluster set up.

Unfortunaly, we didn't achieve to run our code on the cluster. We thus are not able to have result from our Hadoop implementation.

EXPERIMENTATIONS

After implementing the programs, the second part of the project was to stress the program in order to see if it was more efficient or not and try to explain why.

This section analyzes all the experimental results including evaluation performed on Sequential, MPI, OpenMP in terms of their throughput and latency to show how our system can improve performance of OpenMP and Hadoop.

1. Testbed

All the tests have been performed on a system with I5 intel 3rd generation 4-core computer and 4GB of RAM.

For the Hadoop, we created instance on Amazon EC2. We have used t2.micro instance on Amazon EC2. It had High-Frequency Intel Xeon E5-2680 v2 (Ivy Bridge) Processors with 2-vCPU and the instances had Linux operation system.

We ran our test on sequential, mpi (with 2, 3 and 4 nodes), openMP (with 2, 3 and 4 nodes), and Hadoop.

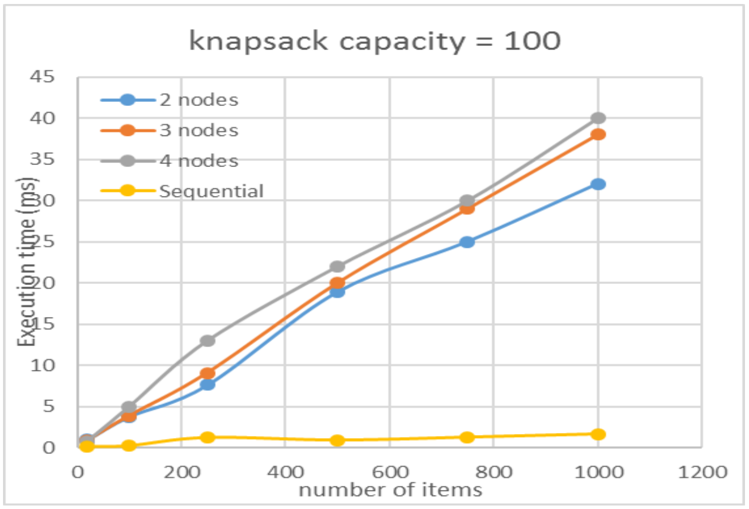
The data set we used was:

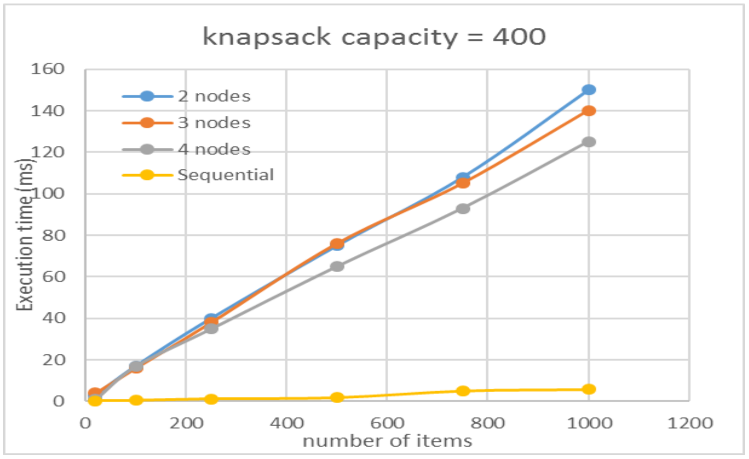
- Number of item: 20, 100, 250, 500, 750, 1000
- Maximum knapsack capacity: 10, 100, 400, 700, 1000

2. Results and performances

a) MPI Results

First, here are the results for 3 knapsack capacities, of the time execution according to the number of items.





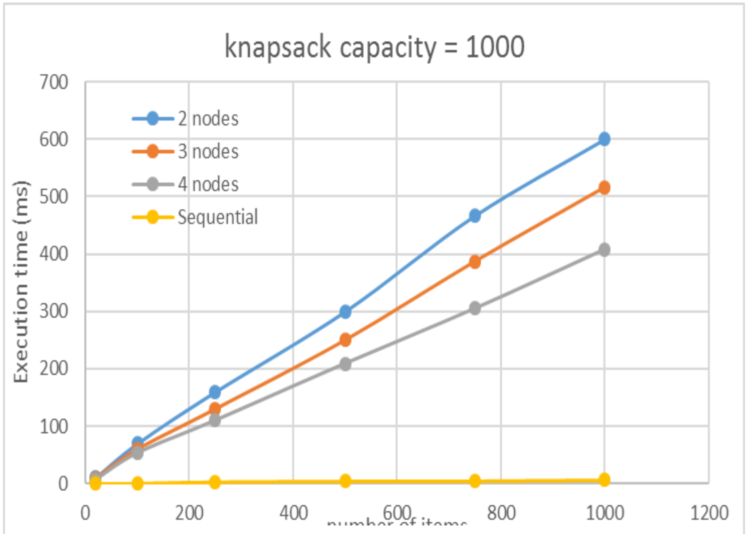


Figure-3: MPI execution results

So, we can first see that our MPI program does not optimize the execution time at all. Even on big matrix like 1000*1000, sequential algorithm is hundreds of times better than MPI. We can think to several points that could explain this:

- The messages passing is not efficient. We can think that a lot of message are send at each iteration (ranks use tag in order to identify the message). "Finding" the good one when MPI_Recv is reach could be a reason of unexpected delay. This is the most likely reason, since our code, excepted for the large number of message sent, is quite simple.

- The matrix is too small for the MPI program to be better than the sequential one. We were not able to test ou program on bigger data set since our computers was not powerful enough. It is possible that sequential execution time increases exponentially since MPI execution time regulates.

Nevertheless, we can still analyze that for smaller matrix (when the knapsack capacity is under 200), the execution time is bigger with 4 nodes than with 2 nodes. This can be explain since "starting" the program by initializing MPI take times. And this time don't make it profitable if the amount of data to compute is too small.

Given the results, we didn't compute the speed up.

b) OpenMP Results

As before, here are the results for 3 knapsack capacities, of the time execution according to the number of items.

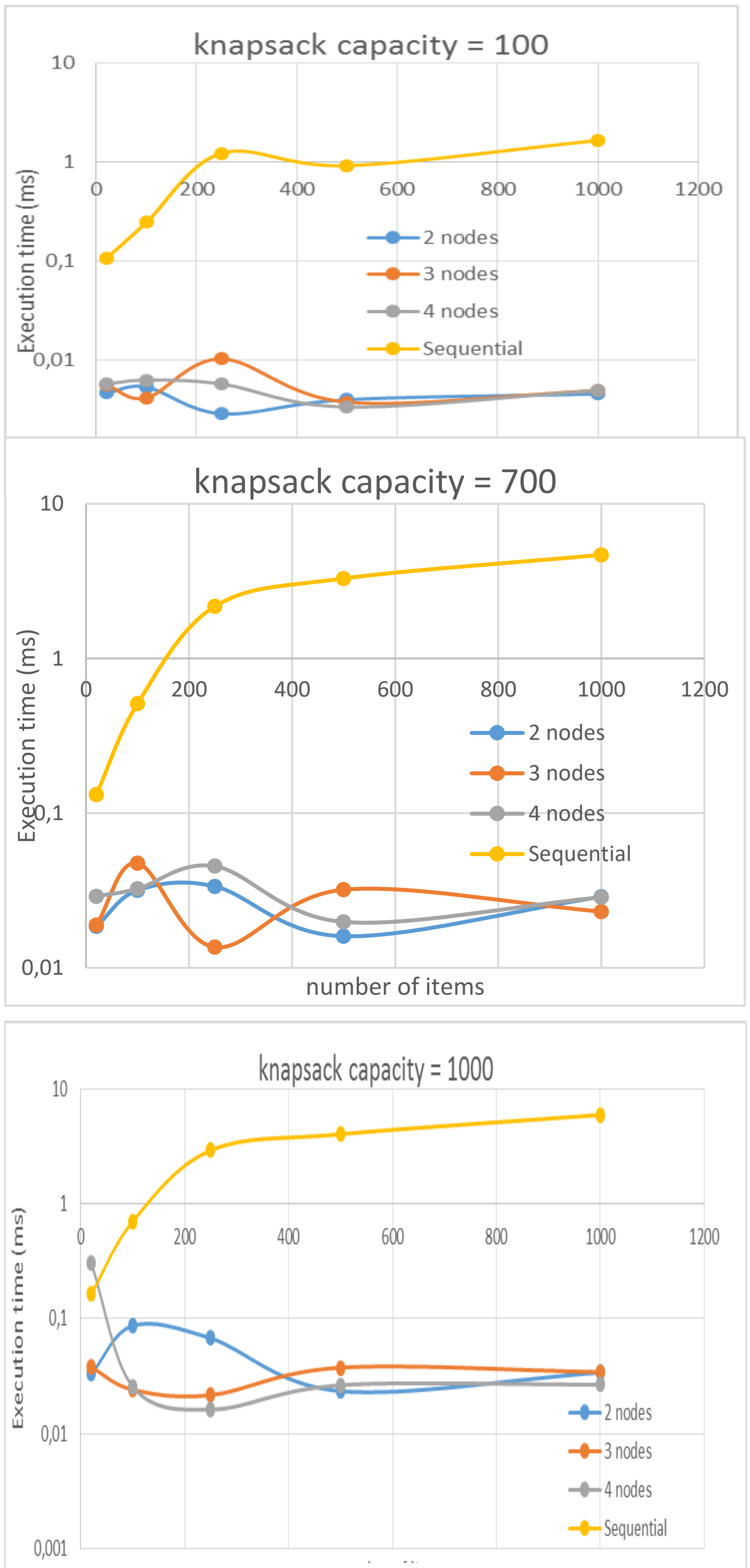


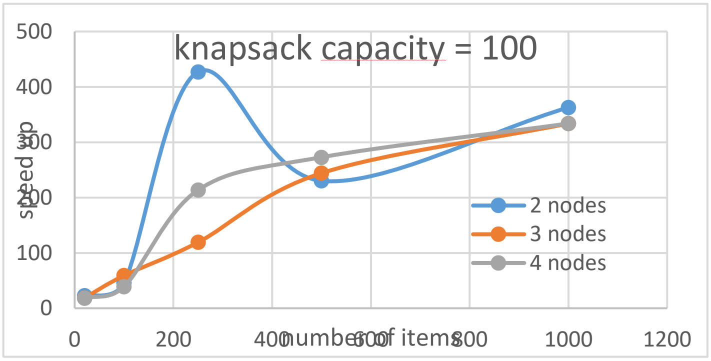
Figure-4: openMPI execution results

So, we can first see that our OpenMP program is fully optimize the execution time at all. Even on big matrix like 1000*1000, OpenMP algorithm is hundreds of times better than sequential. We can think to several points that could explain this:

Column indices can be solved independently in the knapsack 0/1 version. So applying pragma for directory to the column loop. So that each column execution is done parallelly.

We can also notice that some measures doesn't seem coherent. These measure are not to take into account on our analysis, since we noticed irregularities during our tests, and some measures may not be representative.

Then, we computed the speedup:



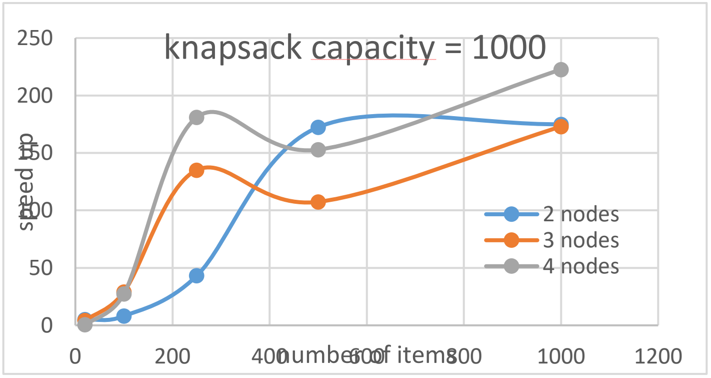


Figure5: Speed up for openMP

So we observe a very good speed up (between 100 and 400) that was expected regarding the results. Moreover, we observe that the speed up is not constant, thus, it gets better when the matrix size increases. We can think that it is because, compute a small amount of data with a sequential program is efficient but the efficiently decrease exponentially with the matrix expansion. As it takes some time to initialize the openMP library, more the matrix increases, more this initialization time is make profitable.

c) Influence of knapsack capacitiy and number of items

Finally, we "cross the data" to try to determine what have the bigger influence on the execution time: the knapsack capacity or the number of item.

To test that we compared the execution time for matrix of the same rough size, but with high number of items and small knapsack capacity and vice versa.

The results are:

Matrix rough size (nb of items * capacity)	Sequential	OpenMP(4)	MPI(4)
1000*10	1,24	0,00	8,52
10*1000	0,16	0,31	10,35

Figure6: Influence of capacity and number of items

So we can observe:

- For the sequential program: compute a high number of items takes more time
- For the openMP and MPI program: compute a high capacity takes more time

This observations are consistent with the way we paralyzed our system. Indeed, the columns (which are the capacity values) are parallelized and computed simultaneously. So a matrix with a small number of column will be compute fast since each thread will have few work at each iteration. On the other side, as the capacity increases, the amount of work per thread at each iterations also increases and the total execution time become longer

FUTURE ENHANCEMENTS

There are several way this work could be extended in the future.

- The first objective would be to achieve to run the Hadoop program.
- Then, the MPI program should be optimized, and think over to determine where delays appears and how we could reduce them.
- In order to stress ours program, bigger data set could be used to determine the efficiency of our programs on extra-large matrix.

- Our programs could also be run and tested on more powerful computers with an higher number of nodes.
- Finally, another approach could be to compare develop another solution to this problem and to compare the efficiency. For example, a CPU based solution, using CUDA.

.

CONCLUSION

To conclude this project was very interesting since it let us reflect and work on a practical parallelism problem. We have put into practice the knowledge learnt during this course, both technically with the program implementations and theoretically with the analyses of results.

We managed to design and implement 3 programs using MPI, openMP and Haddop. Even if some results were not what was expected, it let us reflected about the reasons of this and some possible enhancements. OpenMP provided good results and thus we were able to conclude that the way we parallelized the problem was correct. Finally, analyzing the influence of capacity and number of items was very interesting since we were able to report consistent results for parallelized programs.

ADDITIONAL RESOURCES

1. Presentation feedback

During the presentation, we have been ask to explicit what was our work and what comes from sources.

2. Work partition:

We first both worked on the understanding of the problem and the sequential algorithm. Then the partition work has been as follow:

- Aude : Sequential, MPI, analyses of results, project report, PowerPoint
- Rishabh: Hadoop, openMP, Sequential, openmp and Hadoop parts of report

3. References

[1] The Journal of "Supercomputing archive", Dan E. Tamir, Natan T. Shaked, Shlomi Dolev, Volume 62 Issue 2, November 2012 Pages 633-655

[2] 2. "Parallel metaheuristics for combinatorial optimization", Duni eks¸ioglu, mauricio g.c. resende, panos m. Pardalos, and sandra

[3] 3. "Space-Efficient Parallel Algorithms for Combinatorial Search Problems?" A. Pietracaprina1, G. Pucci1, F. Silvestri1, and F. Vandin2,3

[4] 4. "An efficient parallel algorithm for solving the knapsack problem on the hypercube" A. Goldmany and D. Trystram

[5] 5. "Approximation algorithms for minimum knapsack problem" Mohammad Tauhidul islam

[6] 6. "Parallelization of the Knapsack Problem as an Introductory Experience in Parallel Computing" Michael Crawford, David Toth

[7] 7. "0-1 Knapsack Problem in paralle"l Progetto del corso di Calcolo Parallelo, Salvatore Orlando

ANNEXE

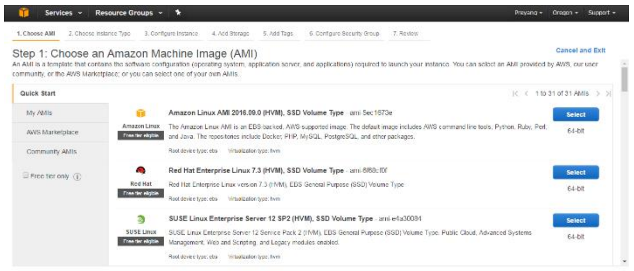
Haddop Cluster set up

Part1: Virtual Cluster

- These steps tell how to setup a virtual cluster on amazon server. Follow the below steps.
- To create an account: http://aws.amazon.com/ec2/
- Click on: Sign in to Console
- It will ask for the details and set up for the card options and it will cost you $1.
- Once the account is activated you need to create an Instance. For that you need to click on the compute under the service tab.
- we are only going to use EC2 (Virtual Servers in the Cloud).Click there, Now you can create AMIs (Amazon Machine Images). Click on Launch Instance to launch a new AMI.

Step 1: Choose an Amazon Machine Image (AMI)

- After selecting the Launch Instance, you need to select the AMI. I selected the **Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-a9d276c9 (64 bit)



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

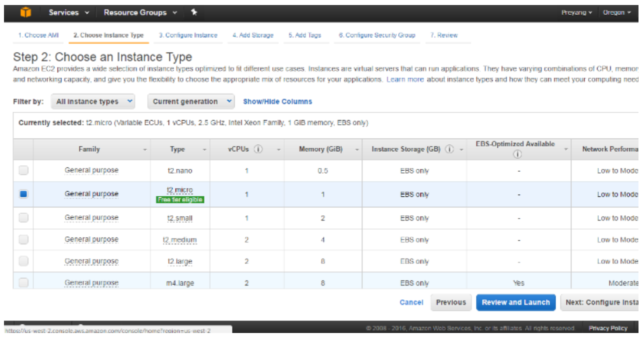
Amazon Linux — Free tier eligible — **Amazon Linux AMI 2016.09.0 (HVM), SSD Volume Type** - ami-5ec1673e — 64-bit
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm

Red Hat — Free tier eligible — **Red Hat Enterprise Linux 7.3 (HVM), SSD Volume Type** - ami-6f68cf0f — 64-bit
Red Hat Enterprise Linux version 7.3 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm

SUSE Linux — Free tier eligible — **SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type** - ami-e4a30084 — 64-bit
SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm

Step 2: Choose an Instance Type

- Click on the t2.micro because it should be enough for our needs. As it is free and it is the updated version after m3.medium.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memor and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing need

Filter by: All instance types | Current generation

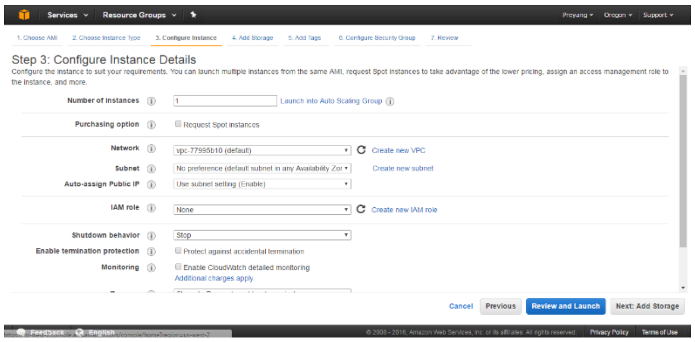
Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performa
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Mode
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Mode
☐	General purpose	t2.small	1	2	EBS only	-	Low to Mode
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Mode
☐	General purpose	t2.large	2	8	EBS only	-	Low to Mode
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Step 3: Configure Instance Details

- You need to fill all the details as shown in the below figure. For now, and since we are going to first show how to configure a single node, let’s just choose Number of instances=1.



Step 3: Configure Instance Details

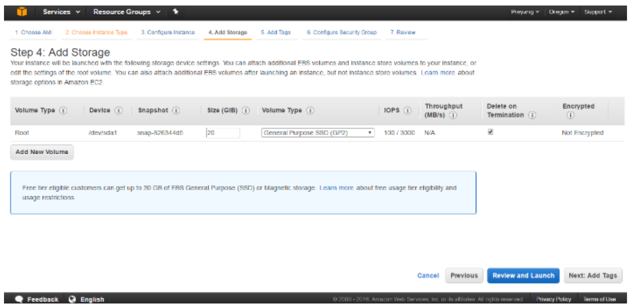
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot instances
Network	vpc-77995b10 (default)
Subnet	No preference (default subnet in any Availability Zor
Auto-assign Public IP	Use subnet setting (Enable)
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Step 4: Add Storage

- One disk should be enough. You should remove /dev/sdb. I believe using a small disk (<30GB) will get you no extra cost. Let's increase the size of /dev/sda1 to 20gb.



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

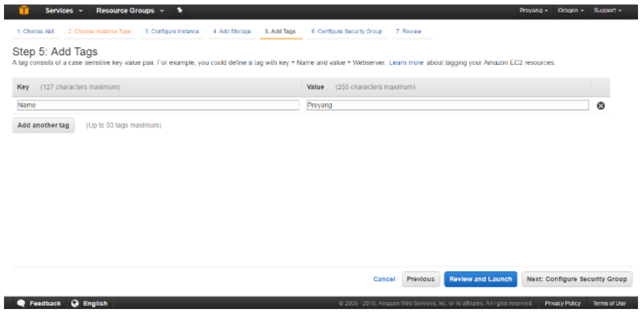
Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-826344d5	20	General Purpose SSD (GP2)	100 / 3000	N/A	✓	Not Encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Step 5: Add Tags

- This allows you to tag this instance with multiple names (for identification purposes). Let's give this node a name: Preyang(master node). For now, we only have one node but later we will add more as slaves, and we will keep this one as the master.



Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

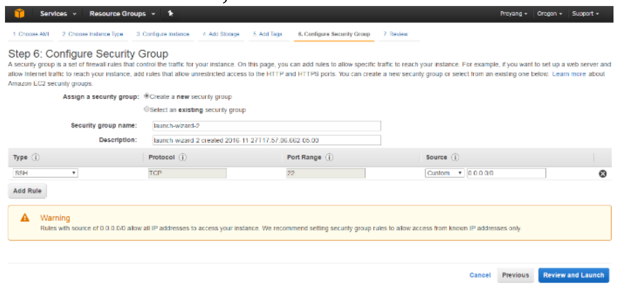
Key (127 characters maximum)	Value (255 characters maximum)
Name	Preyang

Add another tag (Up to 50 tags maximum)

Cancel
Previous
Review and Launch
Next: Configure Security Group
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Step 6: Configure Security Group

- This part is very important, because it defines the network rules that will be applied to all the AMIs that use a public security group. By default, Amazon blocks all ports to the instances but SSH, which is needed for us to access our instances and if not available, we won't be able to even use them.



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-2

Description: launch-wizard-2 created 2016-11-27T17:57:06.662-05:00

Type	Protocol	Port Range	Source
SSH	TCP	22	Custom 0.0.0.0/0

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.

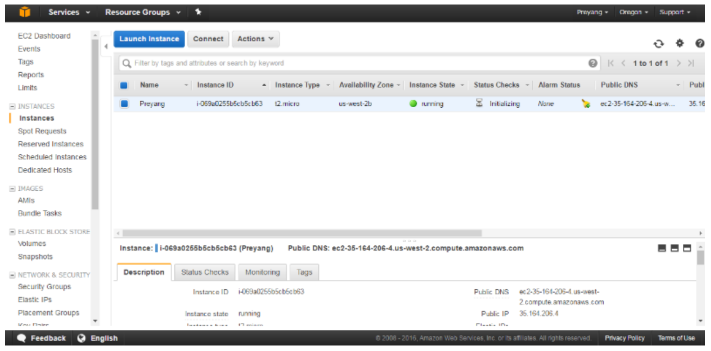
Step 7: Review Instance Launch

- Once its done you can review your instance and launch it.

public/private key: you need to create a pair of public/private keys to access your instance through ssh. Amazon is not using password for this (which is a great idea). Give any name you want to the key pair and click Download key pair. Download the file and name it as MapReduce.pem file and don't delete it (or your will be unable to access your instances). You can leave it on your home directory, and you will need to use it every time you run through ssh to the instances (either using unix ssh or putty on windows).

Linux: ssh -i key.pem ubuntu@public-ip

$ ssh i MapReduce.pem ubuntu@ ec2-35-164-206-4.us-west-2.compute.amazonaws.com



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Publ
Preyang	i-069a0255b5cb5cb63	t2.micro	us-west-2b	running	Initializing	None	ec2-35-164-206-4.us-w...	35.16

Instance: i-069a0255b5cb5cb63 (Preyang) Public DNS: ec2-35-164-206-4.us-west-2.compute.amazonaws.com

Instance ID	i-069a0255b5cb5cb63	Public DNS	ec2-35-164-206-4.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	35.164.206.4

Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

In my case (Ubuntu), the user name is ubuntu.

Before Installing Hadoop:

Hadoop is a framework written in Java for running applications on large clusters of commodity hardware.

PreRequisites to install hadoop:

```
# Update the source list
$ sudo apt-get update

$ sudo apt-get install default-jdk.
```

user@ubuntu: java –version //To check the version of java.

Adding a dedicated Hadoop system user.

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
```

This adds *hduser* and the *hadoop* group is added to the local machine.

To set up single mode of Hadoop, we need to configure SSH access to localhost for the above created *hduser.*

Configuring SSH

To generate an SSH key for the *hduser*, run the commands below:

```
$ su - hduser
$ ssh-keygen -t rsa -P ""
```

The final step is to test the SSH setup by connecting to your local machine with the hduser user.

```
$ ssh localhost
```

Download Hadoop:

Once the prerequisites are checked and properly installed you can now download the hadoop:

For that you need to write the following command:

```
$ sudo wget http://www.gtlib.gatech.edu/pub/apache/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

to change the owner of all the files to the hduser user and hadoop group, for example:

```
$ cd /usr/local
$ sudo tar xzf hadoop-2.7.3.tar.gz
$ sudo mv hadoop-2.7.3.3 hadoop
$ sudo chown -R hduser:hadoop hadoop
```

Configuring Hadoop:

Now we need to set Hadoop related environment variables as well as add bin directory to PATH, set the JAVA_HOME environment variable and related terms. Hence, we update the following files of the user.

1. .bashrc:

```
export PATH=~$HADOOP_HOME/bin
export JAVA_HOME=~/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=~usr/local/hadoop
```

2. hadoop-env.sh: /usr/local/hadoop/conf/hadoop-env.sh we do following changes in the file:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Note: we will configure the directory where Hadoop will store its data files, the network ports it listens to, etc.

```
$ sudo mkdir -p /app/hadoop/tmp

$ sudo chown hduser:hadoop /app/hadoop/tmp
$ sudo chmod 600 /app/hadoop/tmp
```

We need to set the permissions or else it will throw an java.io. IOException

3. In file core-site.xml:

```
<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs.  If "local", then jobs are run in-process as a single map and reduce task.

</description>
</property>
```

4. In file mapred-site.xml:

```
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
```

5. In File hdfs-site.xml:

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
```

Running Hadoop:

The first step to starting up your Hadoop installation is formatting the Hadoop filesystem which is implemented on top of the local filesystem of your “cluster”

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode –format
```

Starting your single-node cluster:

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

Stop your single-node cluster:

```
duser@ubuntu:~$ /usr/local/hadoop/bin/stop-all.s
```