REPORT

제 목 : 하둡설치보고서

과 목 명 : 빅데이터처리

담당교수 : 신현석 교수님

이 름 : 조 정 민

학 번 : 32164420

제 출 일 :

목 차

Ⅰ. 서론

A. 하둡이란?

하둡이란 대용량 데이터를 분산 처리할 수 있는 자바 기반의 오픈소스 프레임워크이다. 핟부은 대규모 데이터 세트를 효율적으로 처리하는 데 사용할 수 있는 오픈 소스 소프트웨어 프로젝트이고, 하나의 대형 컴퓨터를 사용하여 데이터를 처리 및 저장하는 대신, 하둡을 사용하면 상용 하드웨어를 함께 클러스터링하여 대량의 데이터 세트를 병렬로 분석할 수 있다.

이에 사용될 수 있는 기술에는 분산저장(HDFS)기술과 분산처리기술(MapReduce)이 탁월하며, 한 번에 여러 디스크로부터 데이터를 읽을 수 있다.

다시 말해, 하둡은 HDFS(Hadoop Distributed File System)라는 데이터 저장소와 맵리듀스(MapReduce)라는 분석 시스템을 통해 분산 프로그래밍을 수행하는 프레임워크인 것이다.

그렇다면 하둡을 사용하는 이유는 무엇일까? 한마디로 기존 RDBMS는 비싸지만, 하둡은 비용적으로 저렴하다. RDBMS는 빅데이터의 대용량 데이터를 감당하기엔 비용이 너무 크다. 하나의 예시를 들어보자면, 2008년 뉴욕타임즈의 130년 분량의 신문기사 1100만 페이지를 하둡을 이용해 하루 만에 200만원의 비용을 들여 PDF로 변환하였다. 이는 일반서버로 진행했을 경우, 약 14년이 소요되는 작업이었다. 하둡은 분산컴퓨팅 방식을 이용하여 저렴한 구축 비용과 비용 대비 빠른 데이터 처리, 그리고 장애를 대비한 특성을 갖추고 있다.

하둡은 여러 대의 서버에 데이터를 저장하고, 저장된 각 서버에서 동시에 데이터를 처리하는 방식이다. 하둡은 기존의 RDBMS(Oracle, MS-SQL, MySQL 등)을 대치하는 것이 아니다. 즉, 트랜젝션이나 무결성을 보장해야 하는 데이터 처리에는 적합하지 않다. 하둡은 배치성으로 데이터를 저장하고 처리하는데 적합한 시스템이다.

하둡은 두 가지 큰 요소의 결합이다. '처리(계산)'와 '저장'이다. 즉, '분산처리'와 '분산저장'이라고 보면 된다. 여러 개의 저가형 컴퓨터를 마치 하나인 것처럼 묶어주는 기술이라고 보면 된다.

분산저장

하둡 파일시스템(HDFS)을 이용해서 파일을 적당한 블록 사이즈(64MB)로 나누어 각 노드 클러스터(각각의 개별 컴퓨터)에 저장한다. 또한 데이터 유실의 위험이나 사람들이 많이 접근할 대의 부하처리를 위해서 각 블록의 복사본을 만들어 둔다.

분산처리

MapReduce라는 프레임워크를 이용해서 계산하며 프레임워크라는 것이 중요하다. 즉, 분산처리를 위해서 프레임워크를 만들어 둔 것이다. 분산처리를 위해서 프레임워크에 맞추어 코딩을 하고 하둡 시스템에서 그것을 실행하면 자동으로 분산처리를 해준다. 맵리듀스라는 프레임워크는 Map+Reduce라는 두 가지 형식으로 나누어진다. Map 함수에서 데이터를 처리하고 Reduce 함수에서 원하는 결과값을 계산한다.

하둡의 특징은 다음과 같다.

1. Distributed: 수십 만대의 컴퓨터에 자료 분산 저장 및 처리
2. Scalable: 용량이 증대되는 대로 컴퓨터 추가
3. Fault-tolerant: 하나 이상의 컴퓨터가 고장나는 경우에도 시스템이 정상 동작
4. Open source: 공개 소프트웨어

B. 가상 분산 모드란?

하둡은 기본적으로 분산 컴퓨팅 환경을 상정해서 만들어진 시스템이다. 한 대의 컴퓨터를 이용해서 저장 및 분석을 한다면 하둡을 사용할 이유가 없다. 가상 분산 모드란 마치 여러 대의 컴퓨터를 이용하는 것처럼 한 대의 컴퓨터에 하둡을 설치하는 방법이다.

즉, 하나의 장비가 클러스터로 구성이 되어있고, 모든 하둡(HDFS와 MapReduce)과 관련된 데몬이 이 장비에서만 실행되는 환경을 말한다. 주로 맵리듀스 프로그램 개발 및 디버깅에 가장 많이 사용되며 테스트 환경으로 적합한 모드이다.

C. 가상 머신의 개념

가상 머신이란 컴퓨터 안에 또 다른 컴퓨터를 동작시키는 것이다. 컴퓨터 시스템을 에뮬레이션 한다고 말한다. 실제로 물리적인 존재의 컴퓨터는 아니지만 실제 컴퓨터처럼 작동한다. 컴퓨터의 주된 부품들(CPU, RAM, 하드디스크 등)의 기능을 소프트웨어적으로 구현해 가상으로 만들어 구현한다.

가상 머신은 크게 시스템 가상 머신과 프로세스 가상 머신 두 가지로 나뉜다. 시스템 가상 머신은 실제 기계를 대체해서 제공하며, 전체 운영체제를 실행하기 위한 기능들을 제공한다. 프로세스 가상 머신은 플랫폼에 독립적인 환경에서 컴퓨터 프로그램을 실행하기 위해 고안되었다. 프로그래밍 언어의 하드웨어 추상화를 위해 사용된다.

에뮬레이션(Emulation)

모든 부품의 모든 기능을 소프트웨어적으로 구현하는 방식이다. 속도는 가장 느리나 범용성은 가장 뛰어나다.

가상화(Virtualization)

CPU 등 주요 부품의 기능에서 하드웨어의 기능을 지원받는다. 속도는 빠르나 해당 하드웨어에 종속되므로 범용성이 떨어진다. 예를 들어, CPU를 가상화하면 실제 CPU가 처리하는 기계어 세트에서 크게 벗어날 수 없다.

반가상화(Paravirtualization)

기반이 되는 하드웨어, 소프트웨어와 동일하지는 않지만 비슷한 가상 머신에 대한 소프트웨어 인터페이스를 제공하는 기술이다. 완전한 에뮬레이션/가상화를 포기한 방법이다. 가상 머신에 설치되는 OS에 수정을 가하거나 전용 드라이버를 이용하여 하드웨어에 직접 접근하는 방식을 이용한다. 속도는 가장 빠르나 운영체제와 드라이버에 종속되어서 범용성은 가장 떨어진다. Hypervisor를 이용한다.

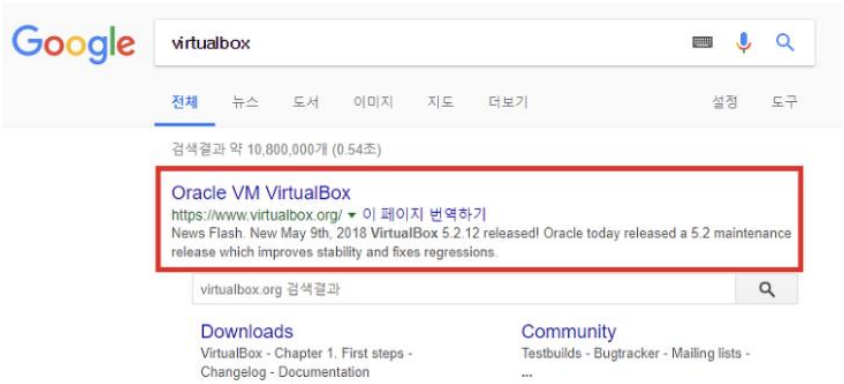
가상 머신을 사용하는 목적과 이유는 다음과 같다.

1. 하나의 컴퓨터로 서로 다른 두 개 이상의 운영체제를 실행하고자 할 때
2. 하나의 컴퓨터 자원을 여러 사용자에게 나누어 주는 상황에서 상호 간섭을 없애고 싶을 때
3. 컴퓨터의 다른 부분에 영향을 주지 않는 독립 환경을 만들고 싶을 때

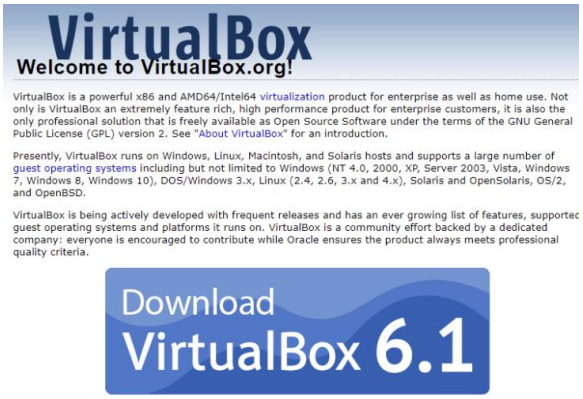
하둡은 리눅스 환경에서 실행이 되고, 본인은 윈도우 환경이기에 Hypervisor(가상 머신)을 설치하고 그 위에 리눅스를 올려서 하둡을 설치할 예정이다.

Ⅱ. 본론

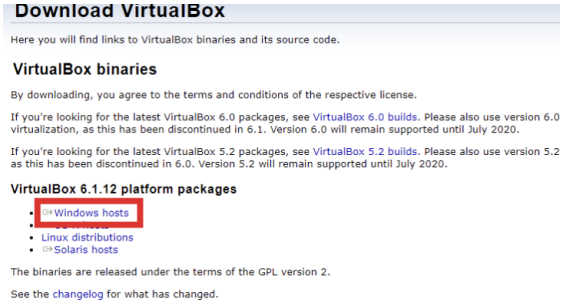
A. VirtualBox 설치



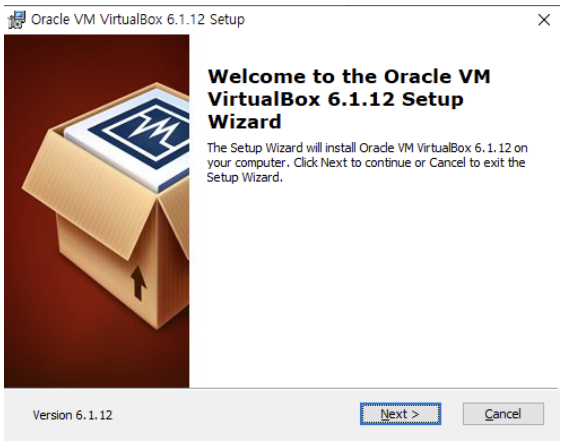
VirtualBox를 다운로드 받기 위해 구글에 검색하여 VirtualBox 공식 사이트에 접속했다.



메인 페이지에 VirtualBox에 대한 간략한 소개와 다운로드 버튼이 있다. 해당 버튼을 눌러 다운로드를 진행한다.



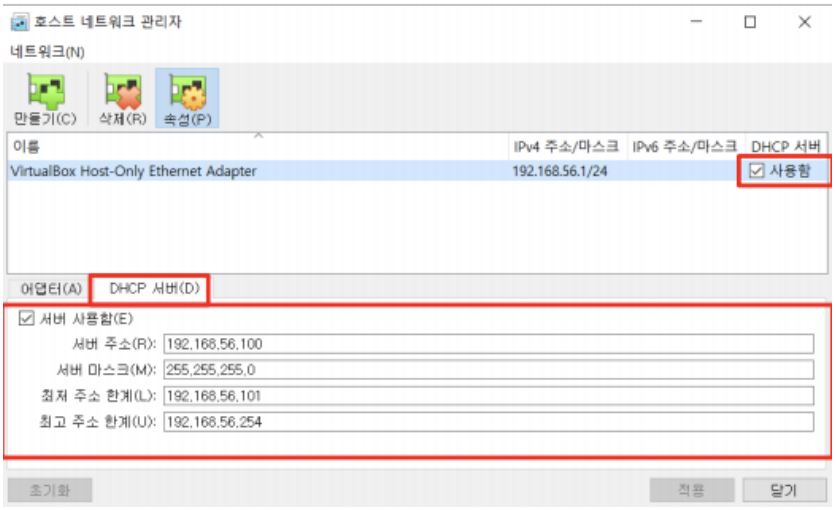
내 운영체제에 맞는 설치파일을 다운로드 받는다. 본인은 윈도우 운영체제이기에 윈도우 버전을 다운로드 받았다.



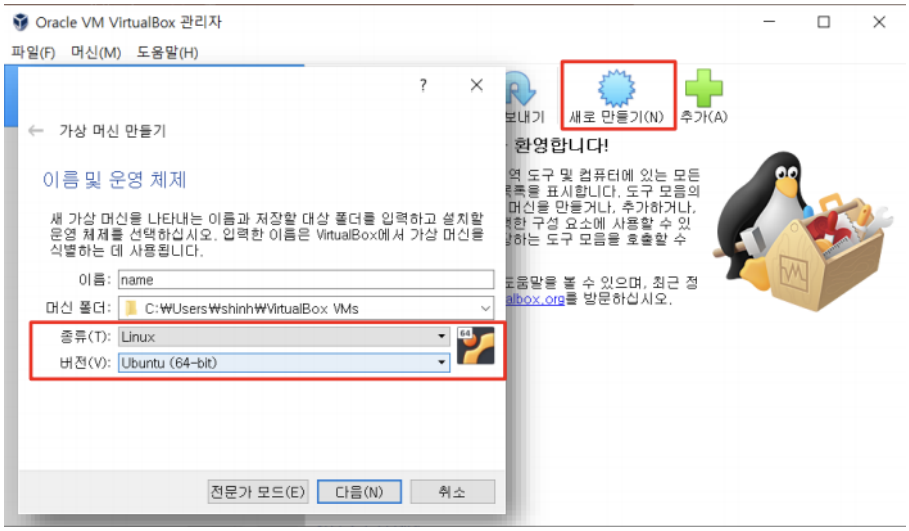
이로써, VirtualBox Setup 파일로 설치를 완료하였다.

B. Ubuntu 설치 및 환경 구축

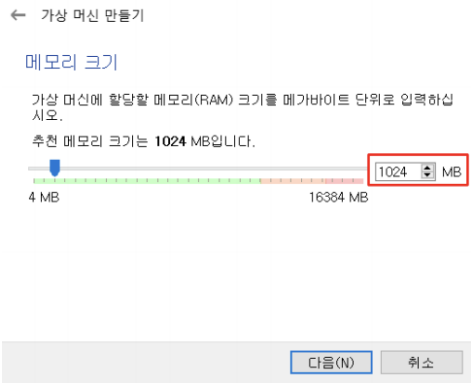
가상 머신을 생성하기 이전에 Ubuntu 20.04 LTS를 설치했다. Ubuntu 20.04 LTS는 Ubuntu 공식 홈페이지에서 다운로드 받을 수 있다.



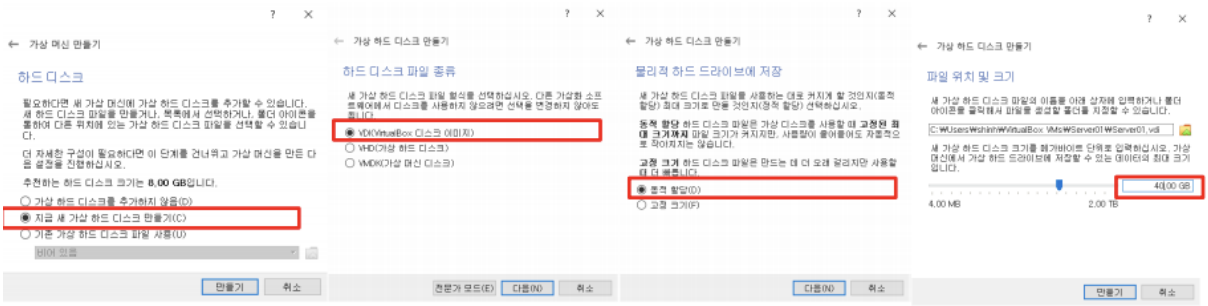
우선 네트워크 설정을 해준다. 파일 -> 호스트 네트워크 관리자로 들어가 DHCP 서버의 정보를 확인한다. VirtualBox Host-Only Ethernet Adapter를 DHCP 서버로 사용함에 체크하고, 서버 주소, 서버 마스크, 최저/최고 주소 한계를 위와 같이 설정해준다.



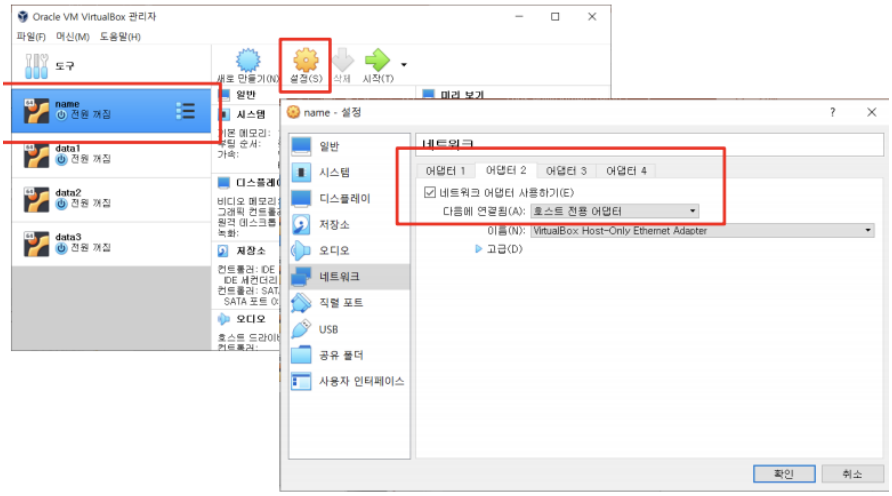
이제 리눅스 환경의 우분투 가상 머신을 생성해준다. 새로 만들기를 통해 이름과 종류, 버전을 선택해준다.



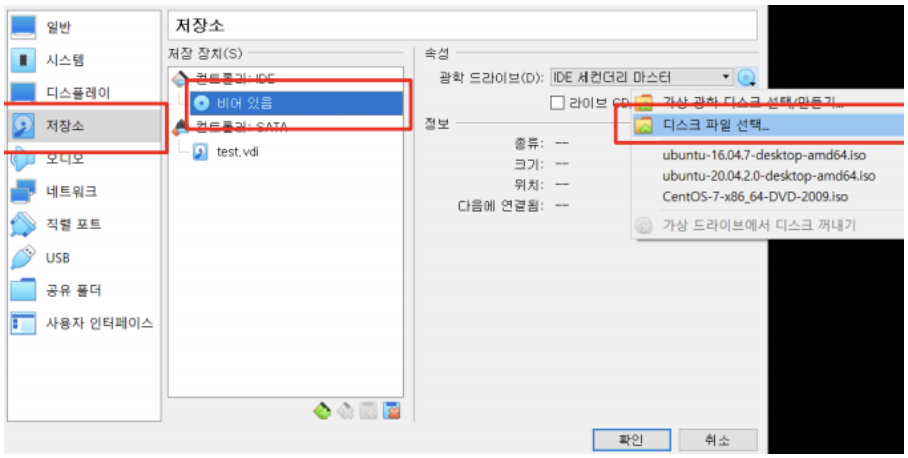
메모리 크기를 할당해준다. 본인은 추천 메모리 크기인 1024MB로 설정해주었다.



새 가상 하드 디스크를 만들어 준다. 하드 디스크 파일 종류는 VDI(VirtualBox 디스크 이미지로 선택한다. 해당 가상 하드 디스크의 크기는 동적 할당으로 선택해준다. 파일 크기는 교수님께서는 40GB로 설정을 하셨지만 본인은 40GB로 설정하고 가상 머신을 생성하니 버벅거림이 생겨 10GB로 설정해주었다.

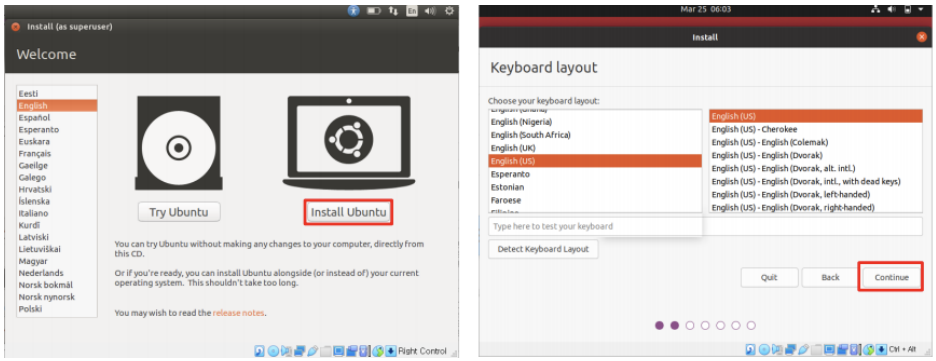


가상 머신을 생성한 이후에는 해당 가상 머신의 네트워크 설정을 해주어야 한다. 설정-네트워크에 어댑터2 탭에 들어가 호스트 전용 어댑터를 선택해준다.

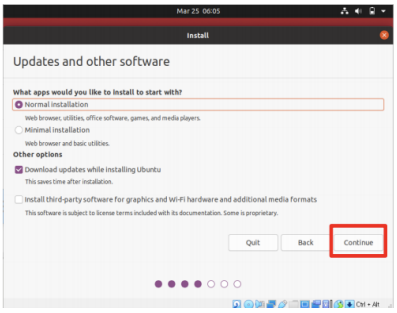


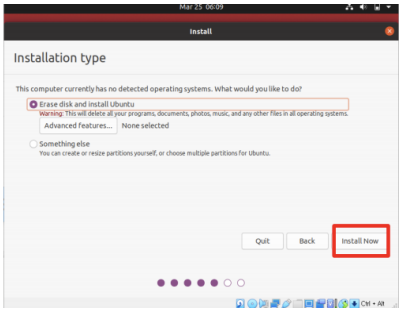
다음은 이미지 마운트다. 설정-저장소에 들어가 비어있는 컨트롤러 IDE에 이전에 설치해준 ubuntu 20.04 iso 디스크 파일을 선택해준다.

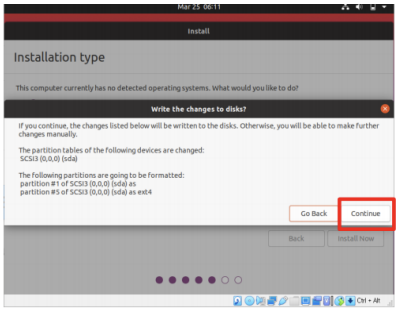
가상 머신 생성을 마무리했으면 우분투를 설치해준다. 생성한 가상 머신을 시작하면 우분투를 설치할 수 있다.



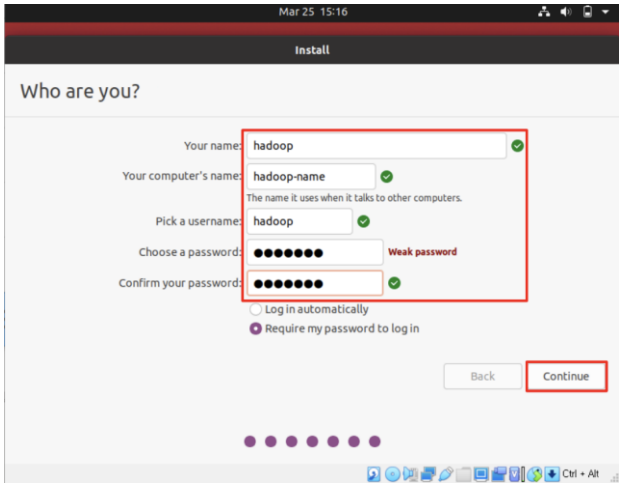
install Ubunu로 설치를 진행한다. 키보드 레이아웃은 영어로 선택하여 진행한다.



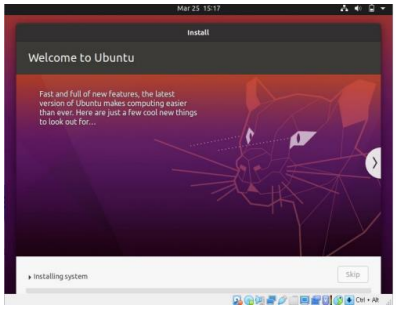


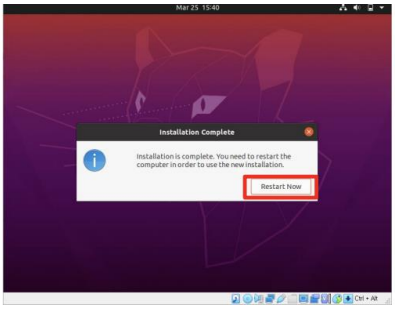


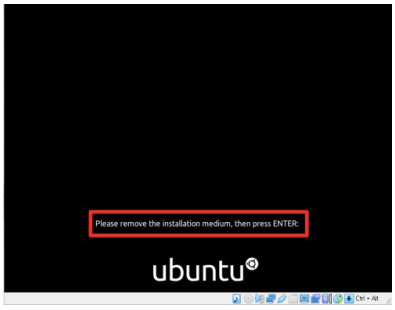
다음 설치 단계는 기본 select 의 옵션(Normal installation/Download updates while installing Ubuntu -> Erase disk and install Ubuntu)으로 설치를 진행한다.



다음으로 name 과 password 를 설정해준다. 교수님께서는 name 을 hadoop 으로 입력하셨지만 본인은 jeongmin 으로 입력하여 진행하였다.

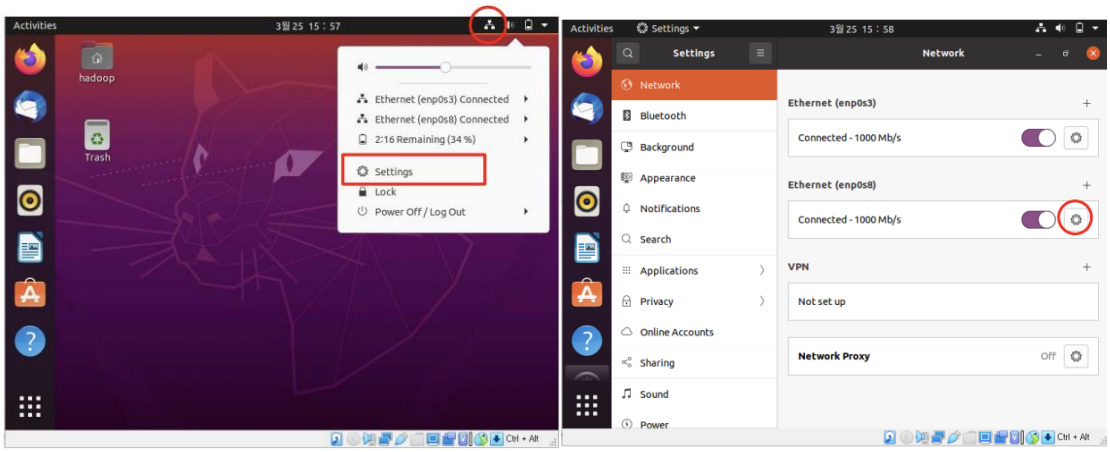


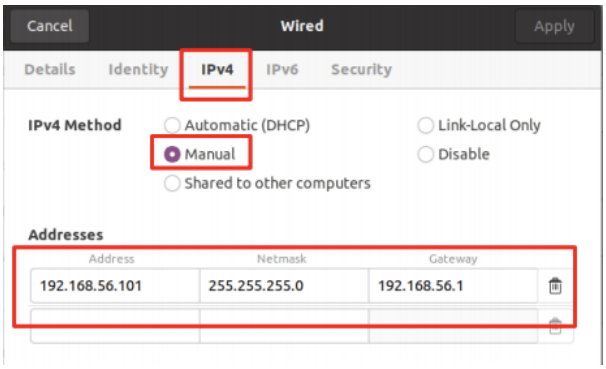




설치 진행이 끝나고 완료되면 가상 머신을 재시작해준다. 위에서 지정한 password 를 입력하면 첫 시작화면이 화면에 띄워진다.

이 과정이 모두 끝나면 가상 환경 상에서 네트워크 설정을 해준다.





Settings – Network – Ethernet(enp0s8)의 설정으로 들어간다. IPv4 탭에서 Method 를 Manual 로 설정하고 Address(192.168.56.101), Netmask(255.255.255.0), Gateway(192.168.56.1)로 설정한 이후, 이전 화면으로 돌아가 재연결을 시켜주면 해당 사항들이 적용된다. 터미널에서 'ifconfig' 명령어를 통해 설정한 네트워크를 확인할 수 있다.

본인은 net-tools 가 설치가 되어있지 않아 'sudo apt install net-tools'로 net-tools 를 설치한 이후 'ifconfig'로 네트워크를 확인하였다.

다음 단계는 ssh server 설치다. Secure Shell Protocol 통신을 위해 'sudo apt install ssh'를 통해 ssh 를 설치해준다.

```
sudo vi /etc/ssh/sshd_config
    #Authentication:
    LoginGraceTime 120
    #PermitRootLogin without-password
    PermitRootLogin yes
    StrictModes yes
sudo service ssh restart
```

'/etc/ssh/sshd_config'를 열어 위와 같이 변경사항을 반영해 저장한 후 재시작을 해준다.

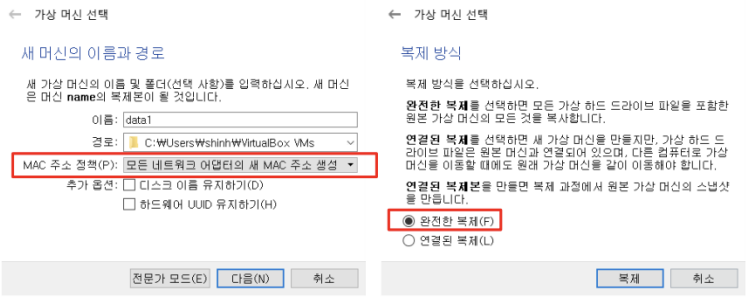
다음으로는 java 를 설치해준다.

```
sudo apt-get install openjdk-8-jdk      # java 설치
sudo vi /etc/profile      # 환경변수 설정
    export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
    export PATH=$PATH:$JAVA_HOME/bin
    export PATH
source /etc/profile
java -version # java 설치 확인
```

host 파일 수정을 해준다.

```
sudo vi /etc/hosts
    192.168.56.101 master
    192.168.56.102 slave1
    192.168.56.103 slave2
    192.168.56.104 slave3
    # 위 내용(nameNode, dataNode)를 입력 후 저장
```

위 과정이 완료되면 가상머신을 종료하고 해당 가상머신을 복제한다.



MAC 주소 정책을 '모든 네트워크 어댑터의 새 MAC 주소 생성'으로 선택하고 완전한 복제본으로 복제한다.

복제된 가상머신들의 hostname 을 변경해준다. hostname 은 다음과 같이 변경할 수 있다.

```
sudo hostnamectl set-hostname hadoop-data1

#sudo hostnamectl set-hostname hadoop-data2

#sudo hostnamectl set-hostname hadoop-data3
```

복제된 가상머신들의 hostname 을 변경했으면 위에서 설명했던 네트워크 설정방법을 참고하여 복제 가상머신들의 네트워크를 설정해준다. IP 주소는 host 파일의 수정내용과 같이 설정(102, 103, 104)하면 된다.

C. Hadoop 설치

여기까지의 모든 과정이 끝나면 이제 hadoop 을 설치한다. 우선 모든 서버의 home 디렉토리에 bigdata 폴더를 생성한다.

```
cd ~

mkdir bigdata
```

이후, name(Jeongmin) 서버에만 bigdata 디렉토리에 다음과 같은 명령어를 통해 하둡을 다운로드 받는다.

```
cd bigdata wget https://archive.apache.org/dist/hadoop/core/hadoop2.5.2/hadoop-2.5.2.tar.gz

tar -xvf hadoop-2.5.2.tar.gz

mv hadoop-2.5.2 hadoop
```

하둡을 다운로드 받고 압축을 해제했으면 환경설정 파일을 수정해주어야 한다. 환경설정 파일 수정은 name(Jeongmin) 서버에만 해당이 된다.

```
cd hadoop/etc/hadoop/      # Hadoop 설정파일이 있는 곳으로 이동

cp mapred-site.xml.template mapred-site.xml
```

hadoop-env.sh / yarn-env.sh 파일을 열어 다음 문구를 추가해준다.

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

yarn-site.xml 파일을 열어 configuration 태그 안에 다음 내용들을 추가해준다.

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

slaves 파일을 열어 localhost 는 삭제하고 slave1, slave2, slave3 를 추가해준다.

```
sudo vi slaves
    slave1
    slave2
    slave3
```

core-stie.xml 파일을 열어 configuration 태그 안에 다음 내용들을 추가해준다.

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/home/jeongmin/bigdata/hadoop_tmp/tmp/</value>
</property>
```

hdfs-site.xml 파일을 열어 configuration 태그 안에 다음 내용들을 추가해준다.

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/jeongmin/bigdata/hadoop_tmp/hdfs/namenode</value>
</property>
```

mapred-stie.xml 파일을 열어 configuration 태그 안에 다음 내용들을 추가해준다.

```
<property>
    <name>mapreduce.job.tracker</name>
    <value>master:5431</value>
</property>
<property>
    <name>mapred.framwork.name</name>
    <value>yarn</value>
</property>
```

다음으로 name(jeongmin)서버에 ssh 키 설정을 해준다. 다음 명령어를 통해 키 설정을 할 수 있다.

```
ssh-keygen -t rsa -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

name(jeongmin)서버에서 공개키를 배포해준다.

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub jeongmin@slave2

ssh-copy-id -i $HOME/.ssh/id_rsa.pub jeongmin@slave2

ssh-copy-id -i $HOME/.ssh/id_rsa.pub jeongmin@slave3
```

Hadoop 디렉토리를 각 노드에 배포해준다. 이 작업 또한 name(jeongmin)서버에서만 실행한다.

```
sudo rsync -avxP /home/jeongmin/bigdata/hadoop/

 jeongmin@slave1:/home/jeongmin/bigdata/hadoop

sudo rsync -avxP /home/jeongmin/bigdata/hadoop/

 jeongmin@slave2:/home/jeongmin/bigdata/hadoop

sudo rsync -avxP /home/jeongmin/bigdata/hadoop/

 jeongmin@slave3:/home/jeongmin/bigdata/hadoop
```

모든 서버에서 환경변수 등록을 해준다.

```
sudo vi ~/.bashrc
 ## bashrc 파일 제일 하단에 추가
 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
 export HADOOP_HOME=/home/hadoop/bigdata/hadoop
 export HADOOP_MAPRED_HOME=$HADOOP_HOME
 export HADOOP_COMMON_HOME=$HADOOP_HOME
 export HADOOP_HDFS_HOME=$HADOOP_HOME
 export YARN_HOME=$HADOOP_HOME
 export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
 export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
 export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
 export PATH=$PATH:$HADOOP_HOME/bin
 export PATH=$PATH:$HADOOP_HOME/sbin
 export PATH=$PATH:$JAVA_HOME/bin
 export PATH
source ~/.bashrc
```

name(jeongmin) 서버에서 네임노드 초기화를 시켜준다. 네임노드 초기화 명령어는 'hadoop namenode – foramt'이다.

이후, name(jeongmin)서버에서 데몬실행을 한다. 데몬실행 명령어는 'start-all.sh'이다. 이제 name 노드와 data 노드에서 'jps' 명령어를 통해 실행확인을 할 수 있다.

```
jeongmin@jeongmin:~$ jps
3074 Jps
2426 NameNode
2813 ResourceManager
2670 SecondaryNameNode
```

```
jeongmin@hadoop-data1:~$ jps
2198 DataNode
2409 NodeManager
2589 Jps
```

성공적으로 실행이 되었음을 확인하였다. hadoop 종료는 'stop-all.sh' 명령어를 사용한다. 실행확인을 마쳤으면 해당 명령어로 hadoop 을 종료한다.

Ⅲ. 결론

A. 설치 시 어려웠던 점

첫 번째로 새 가상 하드디스크 크기를 설정할 때, 교수님과 같이 40GB 로 설정하여 진행했는데 어떤 문제인지는 모르겠지만 우분투 설치 과정에서 더 진행이 되지 않았다. 크기를 좀 더 낮춰 10GB 로 설치를 진행했을 때는 무리 없이 우분투 설치가 완료되었다.

두 번째로는 이더넷 설정에서 Gateway 를 192.168.56.1 로 입력해야 했는데 data2, data3 가상머신에서 192.168.56.0 으로 설정해 이더넷 연결이 불완전하여 하둡 설치 진행이 원활하지 않았다. 처음에는 와이파이 문제인 줄 알았으나 이더넷 설정에서의 문제라는 것을 인지하여 수정 후 문제를 해결하였다.

세 번째로는 xml 파일의 configuration 태그 안 내용들을 입력할 때 발생했다. 본인은 name 을 hadoop 이 아닌 jeongmin 으로 입력하여 진행했다. 때문에, xml 파일에서 경로를 입력할 때도 home/hadoop/bigdata 가 아닌 home/Jeongmin/bigdata 로 입력했어야 했다. 하지만 이 부분을 인지하지 못하고 전자의 경로로 입력을 해줬기에 jps 명령어를 사용할 때 NameNode 와 DataNode 가 출력되지 못했다. name 서버에서 해당 하둡 디렉토리를 data 서버에 배포해줬기 때문에 이 각 서버별로 해당 내용을 하나하나 수정해줬다. NameNode 와 DataNode 의 로그 파일을 확인하여 해당 문제를 발견했고 해결할 수 있었다.

B. 이해하기 어려웠던 부분

가상 서버를 만들고 복제하는 부분까지는 이해가 쉬웠는데 네트워크 설정이나 하둡 설치에 전반적인 부분에 대한 이해를 하기는 어려웠다. 교수님의 설명이나 PPT 파일이 없었다면 백지의 상태로 혼자 하둡을 설치하기에는 무리가 있을 것이라 생각했다. xml 파일 수정이나 ssh 키 설정, 공개키 배포, 하둡 디렉토리 배포 등 명령어나 변경/추가 사항 부분을 적용하는 데에 있어 좀 더 익숙해질 필요가 있다고 느꼈다.

C. 문제 해결

문제 해결은 위 'A. 설치 시 어려웠던 점'을 기반으로 설명하겠다.

1. 새 가상 하드디스크 크기

40GB 로 설정하여 설치를 진행했을 때 특정 부분에서 설치가 더 진행되지 않아 10GB 로 설정하여 문제를 해결하고 설치를 마쳤다.

2. data2, data3 의 이더넷 연결 불안정 상태 해결

ethernet 설정에서 gateway 의 주소 입력이 잘못되어 문제가 발생했다. 192.168.56.1 로 입력해야 하는데 192.168.56.0 으로 입력이 되어있었다. 해당 gateway 주소를 수정하여 문제를 해결하였다.

3. jps 명령어 수행 시 NameNode, SecondaryNameNode, DataNode 미출력 해결

각 노드의 로그파일을 확인한 결과 에러가 발생했고 해당 에러 문구를 검색해보니 xml 파일을 수정할 때 경로지정이 잘못되었다는 것을 알게 되었다. name 을 hadoop 이라 설정하지 않고 jeongmin 으로 설정하여 진행했다. xml 파일에 내용을 추가할 때도 경로 지정을 home/Jeongmin/bigdata 로 입력했어야 하는데 home/Hadoop/bigdata 로 입력해 설치를 진행해 문제가 생긴 것이다. 각 서버의 xml 파일의 경로를 수정하고 jps 명령어를 실행했을 때 원하는 결과가 출력될 수 있었다.