빅데이터 마스터과정
(DAM)

하석재
CEO, 2HCUBE
sjha72@gmail.com

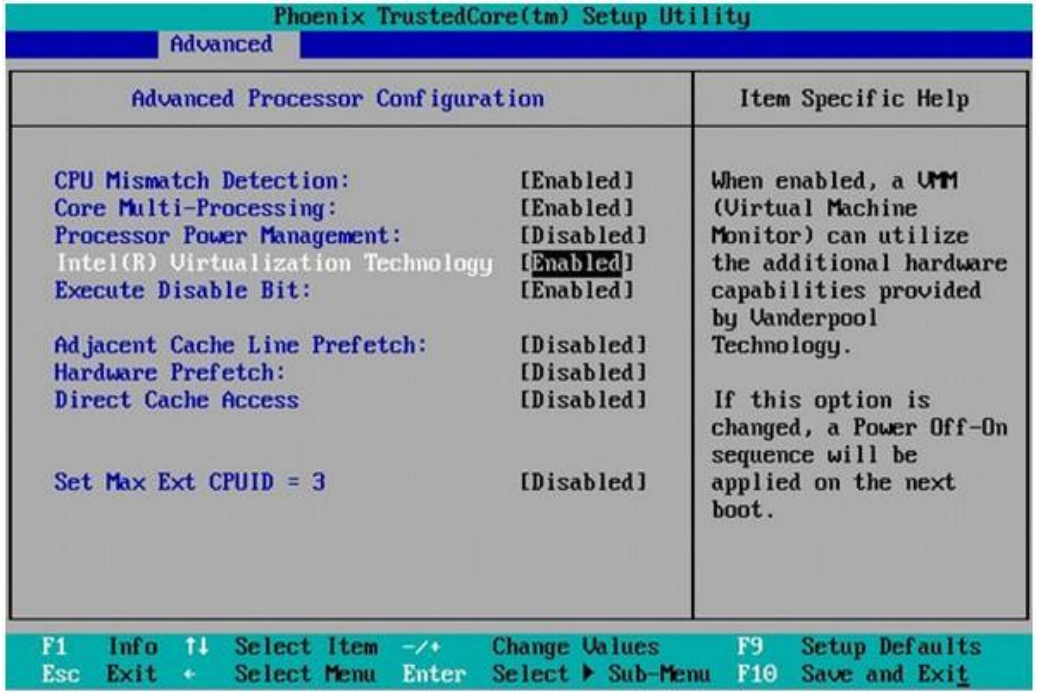
하둡 실습

하둡 설치 준비사항

- 버추얼박스 다운로드 및 설치
  - virtualbox.org
- 우분투**서버** iso다운로드
  - ubuntu.com 에서 다운로드
- 우분투 설치 및 로그인
- 자바설치 및 환경변수 설정

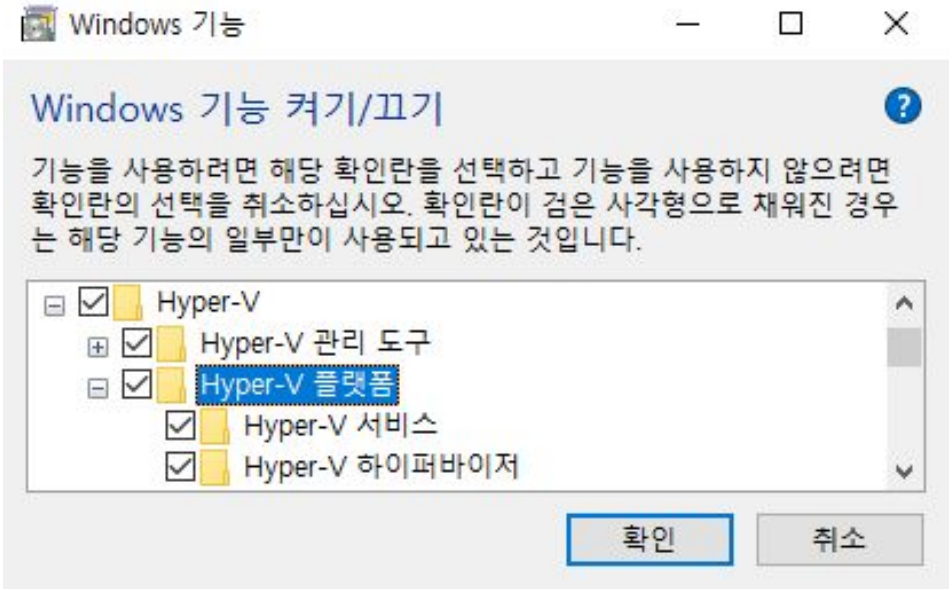
주의사항

- 가상화기술(CPU / 충돌문제)
  - pentium/celeron 이름을 가진 CPU는 불가능 -> vt-x/amd-v
  - BIOS에서 기능을 enable(advanced - intel virtualization technology 메뉴)
  - 다른 하이퍼바이저(VMWare / Hyper-V)와 충돌가능성 cf. WSL2


Phoenix TrustedCore(tm) Setup Utility
Advanced
Advanced Processor Configuration
Item Specific Help
CPU Mismatch Detection: [Enabled]
Core Multi-Processing: [Enabled]
Processor Power Management: [Disabled]
Intel(R) Virtualization Technology [Enabled]
Execute Disable Bit: [Enabled]
Adjacent Cache Line Prefetch: [Disabled]
Hardware Prefetch: [Disabled]
Direct Cache Access [Disabled]
Set Max Ext CPUID = 3 [Disabled]
When enabled, a VMM (Virtual Machine Monitor) can utilize the additional hardware capabilities provided by Vanderpool Technology.
If this option is changed, a Power Off-On sequence will be applied on the next boot.
F1 Info ↑↓ Select Item -/+ Change Values F9 Setup Defaults
Esc Exit ← Select Menu Enter Select ▶ Sub-Menu F10 Save and Exit

Hyper-V 설정

- 윈도우-설정 - 앱 - 프로그램 및 기능 - Windows 기능 켜기/끄기



https://blog.gaerae.com/2019/04/hyper-v-troubleshooting.html

하둡 설치 준비사항

- VirtualBox설치(virtualbox.org)
- Ubuntu Server 20.04(LTS) 설치(ubuntu.com)
- VM 추가

  RAM **2GB이상**, Disk **200GB**, **Linux/Ubuntu(64bit)**

  설정-시스템-프로세서 개수지정(**2이상**)

  설정-저장소-비어있슴(CD아이콘클릭)-오른쪽 CD롬 아이콘 클릭하고 choose a disk file - 다운받은 이미지파일 선택

하둡 설치 준비사항

- '시작'버튼 클릭
  계속 엔터치다가 탭으로 done 선택 - 엔터 -
  디스크 삭제 경고시 No-> Continue
  전부 ubuntu로 통일(서버명, 아이디, 비밀번호 등)
  나머지 옵션은 모두 탭쳐서 Done으로 진행
  실제 설치진행 -> 설치후 리부팅
  로그인(ubuntu / ubuntu) 진행

하둡 설치 준비사항

- 최신 버그패치
  $ sudo apt update

- ssh서버 설치
  $ sudo apt install openssh-server

하둡 설치

- 버추얼박스 포트 포워딩
  - 버추얼박스 설정-네트워크-고급-
    포트포워딩- +(추가) - **22/22** - 확인

- putty(MS Windows)
  - 구글에서 'putty' 검색 후 다운로드(64비트, install버전 아닌 것) 및 실행
  - 127.0.0.1(또는 localhost)에 22번으로 접속

- MacOS / 리눅스에서는 ssh 명령어로 접속
  $ ssh ubuntu@localhost 또는 $ ssh ubuntu@127.0.0.1

하둡 설치 준비사항

- 환경변수(Environment Variable)

$ env 또는 $ echo $PATH
cf. set(윈도우) / echo %PATH%
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
$ pwd (현재 위치/폴더확인)

$ export PATH=$PATH:/test
cf. set PATH=%PATH%; (윈도우)
현재 폴더(.)에 없는 파일을 $PATH의 값을 따라 뒤짐
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/test

하둡 설치 준비사항

- 자바설치
  $ sudo apt install openjdk-11-jdk

- 자바 환경변수 설정
  JAVA_HOME(JDK설치된 폴더)
  PATH(실행프로그램 대상 검색)
  $JAVA_HOME 아래의 bin 폴더 (javac/java)
  CLASSPATH
  .java -> .class(실행파일)
  $JAVA_HOME/lib/*

하둡 설치

- conf/hadoop-env.sh 수정
  export JAVA_HOME =

- 자바 홈 디렉토리 설정
  우분투의 경우 /usr/lib/jvm 밑에 있슴
  실제 디렉토리 확인 필요

- 오픈자바(JDK)의 경우
  /usr/lib/jvm/java-11-openjdk-amd64

- 오라클 자바의 경우
  /usr/lib/jvm/java-11-oracle

하둡 설치 준비사항

- .bashrc 또는 .profile(자동실행파일)
  -> export 내용을 저장하면 계속 유지됨

$ cd / $ pwd / $ ls -al
$ nano $HOME/.bashrc
마지막에 추가

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export CLASSPATH=$JAVA_HOME/lib/*:.
export PATH=$PATH:$JAVA_HOME/bin
```

$ source .bashrc / $ echo $JAVA_HOME
$ echo $CLASSPATH / $ echo $PATH

하둡 설치 준비사항

- 환경변수 할당단위
  - **프로세스 마다 하나씩 할당**(환경변수 영역)
    자식 프로세스는 부모프로세스의 환경변수 복사(반대는 안 됨)
    부모가 자식을 실행(띄운다)

- 리눅스 부팅과정
  swapper(0) -> init(1) -> kthreadd(2) -> ...
    cf. ssh localhost (putty)-> ssh localhost

- 죽였다가 다시 띄우면 환경변수 리셋
  -> 죽였다가 다시 띄워도 안지워지게 하려면?

하둡 설치

- 하둡 다운로드

$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz

- 압축해제

$ tar xvfz hadoop-3.3.1.tar.gz

$ cd hadoop-3.3.1

$ ls -al

하둡 설치

- nano 에디터 수행
  $ nano ~/.profile 또는 **$ nano ~/.bashrc**
- 환경변수 설정 및 저장(마지막에 추가)
  export HADOOP_HOME=/home/ubuntu/hadoop-3.3.1
  export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
- 환경설정 반영
  $ source $HOME/.profile 또는 **source $HOME/.bashrc**
- 환경변수 설정 확인
  $ echo $HADOOP_HOME / $ echo $PATH

하둡 실행모드

- 스탠드얼론(**Standalone**)
  - 기본수행모드
  - **HDFS 사용하지 않음**
  - **서버없이** 간단한 프로그램 테스트용
- 의사분산모드(**Pseudo Distributed Mode**)
  - **HDFS** 사용(입출력 모두)
  - **한 대**의 서버에 **모든 서버** 수행
    - 네임노드/데이터노드/잡트래커/데이터노드
- 완전분산모드(**Fully Distributed Mode**)
  - **HDFS**사용
  - 각 서버를 **여러 대**의 시스템에 나눠 수행
  - 하둡을 제대로 사용하는 모드

하둡 - Standalone

- 별도의 HDFS없이 하나의 서버, 하나의 JVM에서 모두 수행하는 모드

- $HADOOP_HOME/etc/hadoop/mapred-site.xml
- $HADOOP_HOME/etc/hadoop/hdfs-site.xml
- $HADOOP_HOME/etc/hadoop/core-site.xml

  세 파일 모두
  <configuration>
  </configuration>
  만 있는것 확인

하둡 - Standalone

- 하둡 수행

$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar wordcount $HADOOP_HOME/README.txt $HOME/output

- 결과 확인

$ nano ~/output/part-r-00000 또는 **$ cat ~/output/part-r-00000**

하둡 1.x와 2.x 폴더구조/사용법 차이점

- 하둡 1.x
  - **conf 폴더 안에 설정 파일들이 있슴**
  - 예제가 압축 푼 폴더($HADOOP_HOME)에 있슴
  - 1.x 초기 버전은 $HADOOP_HOME/bin에 실행파일 및 스크립트파일 존재
  - 1.x 후기 버전부터 2.x 부터는 **$HADOOP_HOME/bin**과 $HADOOP_HOME/sbin 에 나눠 저장됨
    - sbin에는 .sh와 같은 스크립트가 저장

하둡 **1.x**와 **2.x** 폴더구조/사용법 차이점

- 하둡 1.x
  - 1.x 초기버전은
    - **start-all.sh/stop-all.sh**에 HDFS/MapReduce 모두 수행/정지
  - 1.x 후기 버전부터 hdfs의 수행/정지는 **start-dfs.sh/stop-dfs.sh**로,
  - 맵리듀스 수행/정지는 **start-mapred.sh/stop-mapred.sh**로 나눠짐

하둡 1.x와 2.x 폴더구조/사용법 차이점

- 하둡 2.x 이후
  - 설정파일위치
    - **$HADOOP_HOME/etc/hadoop** 폴더
    - **$HADOOP_CONF_DIR**
  - 예제의 위치
    - **$HADOOP_HOME/share/hadoop/mapreduce**
  - 패키지명 변경
    - **hadoop-core(1.x) -> hadoop-common(2.x)**
  - 맵리듀스 수행/정지는 **start-yarn.sh/stop-yarn.sh**로 나눠짐

워드카운트 예제코드

- 예제위치
  - $HADOOP_HOME/share/hadoop/mapreduce/sources

$ cd $HADOOP_HOME/share/hadoop/mapreduce/sources
$ jar xvf hadoop-mapreduce-examples-3.3.1-sources.jar
$ nano org/apache/hadoop/examples/WordCount.java

하둡 - Standalone

$ hadoop jar hadoop-examples-3.3.1.jar wordcount README.txt ~/wordcount-output

Warning: $HADOOP_HOME is deprecated.

13/11/20 11:48:42 INFO util.NativeCodeLoader: Loaded the native-hadoop library
13/11/20 11:48:42 INFO input.FileInputFormat: Total input paths to process : 1
13/11/20 11:48:42 WARN snappy.LoadSnappy: Snappy native library not loaded
13/11/20 11:48:42 INFO mapred.JobClient: Running job: job_local_0001
13/11/20 11:48:42 INFO util.ProcessTree: setsid exited with exit code 0
13/11/20 11:48:42 INFO mapred.Task: Using ResourceCalculatorPlugin :
org.apache.hadoop.util.LinuxResourceCalculatorPlugin@44e03e45
13/11/20 11:48:42 INFO mapred.MapTask: io.sort.mb = 100

하둡 - Standalone

- 결과확인

  $ cat ~/wordcount-output/part-r-00000

하둡 - Pseudo Distributed Mode

- 하둡 서버(Daemon)
  - NameNode, Secondary NameNode, DataNode
    NodeManager, ResourceManager
- 설정
  - hadoop-env.sh 수정
  - ssh 인증서기반 로그인(자동로그인) 설정
  - 3개의 환경설정 파일 수정
    hdfs-site.xml, core-site.xml, mapred-site.xml, yarn-site.xml
- HDFS 포맷
- 5개의 데몬 수행(start-dfs.sh / start-yarn.sh)

하둡 - Pseudo Distributed Mode

$HADOOP_HOME/etc/hadoop/core-site.xml:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

$HADOOP_HOME/etc/hadoop/hdfs-site.xml:

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

하둡 - Pseudo Distributed Mode

- 우분투 임시폴더(/temp)에 HDFS폴더 생성하는 것을 변경

core-site.xml

```
<configuration>
  <property>
   <name>fs.default.name</name>
   <value>hdfs://localhost:9000</value>
  </property>
  <property>
      <name>hadoop.tmp.dir</name>
      <value>/home/ubuntu/temp</value>
  </property>
</configuration>
```

하둡 - Pseudo Distributed Mode

- 우분투 임시폴더에 HDFS폴더 생성하는 것을 변경

- 폴더생성

$ mkdir $HOME/temp

하둡 - Pseudo Distributed Mode

- hadoop-env.sh 수정

export JAVA_HOME = ... 찾아 # 삭제 후 실제 자바 홈 디렉토리 설정
export HADOOP_HOME= 실제 하둡 홈 디렉토리 설정

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/home/ubuntu/hadoop-3.3.1

하둡 - Pseudo Distributed Mode

- ssh 자동로그인 설정

```
$ ssh-keygen -t dsa -P "" -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
$ cat ~/.ssh/authorized_keys
```

- 테스트

```
$ ssh localhost      비밀번호없이 로그인되는 것을 확인
$ exit
```

하둡 - Pseudo Distributed Mode

mapred-site.xml

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
    </property>
</configuration>
```

하둡 - Pseudo Distributed Mode

yarn-site.xml

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSP
ATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
    </property>
</configuration>
```

하둡 - Pseudo Distributed Mode

- DataNode, Secondary NameNode, Node manager, Resource manager
  localhost로 기본 설정됨

$ cat masters
localhost

$ cat slaves (workers)
localhost

하둡 - Pseudo Distributed Mode

- HDFS 포맷
  $ hdfs namenode -format 또는 $ hadoop namenode -format

하둡 - Pseudo Distributed Mode

- 5개 데몬 실행

  $ start-all.sh (start-dfs.sh / start-yarn.sh)

  $ jps

  ```
  5225 NodeManager
  5134 SecondaryNameNode
  4636 NameNode
  4878 DataNode
  5535 Jps
  5473 ResourceManager
  ```

  $ stop-all.sh (stop-dfs.sh / stop-yarn.sh)

하둡 - Pseudo Distributed Mode

- HDFS에 파일 업로드

```
$ hdfs dfs -mkdir /input
$ hdfs dfs -put $HADOOP_HOME/README.txt /input
$ hdfs dfs -ls /input
-rw-r--r--   1 sjha supergroup       1366 2013-11-20 13:36 /input/README.txt
```

하둡 - Pseudo Distributed Mode

- Wordcount 실행

  $ hadoop jar
  $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
  wordcount /input/README.txt /output

- 결과보기

  $ hdfs dfs -cat /output/part-r-00000 | more

워드카운트 예제코드

- 예제위치
  - $HADOOP_HOME/share/hadoop/mapreduce/sources

$ cd $HADOOP_HOME/share/hadoop/mapreduce/sources
$ jar xvf hadoop-mapreduce-examples-3.3.1-sources.jar
$ nano org/apache/hadoop/examples/WordCount.java

맵 태스크 수의 결정방식

- getSplits 메소드
  InputSplit들의 리스트 리턴
- InputSplit마다 맵 태스크가 할당됨

- **입력 파일의 수**
- **입력 파일의 크기**
  데이터 블록(기본 64MB/128MB)마다 맵태스크 할당 cf. 2.4GB(38/19)
  데이터 블럭이 InputSplit
- 입력 포맷의 변수
  압축파일의 경우 전체를 하나의 맵 태스크에 할당
  (SequenceFileInputFormat의 경우 예외)
  isSplitable

Reduce Class

- run, setup, cleanup

- 리듀스 입력
  - 맵단의 출력에 따라 결정
- 리듀스 출력
  - TextOutputFormat/SequenceFileOutputFormat
  - Customization -> Job.setOutputFormatClass
  - FileOutputFormat.setOutputPath
  - 출력타입
    - setOutputKeyClass/setOutputValueClass

리듀스 태스크의 수 : Job.setNumReduceTasks(2)
- cf. 기본값 1

위키피디어 파일 검색

- 2M.TITLE.ID
  문서의 제목과 ID를 가지는 파일
- **2M.ID.CONTENTS** (2.4GB)
  문서 ID와 파일내용을 가지는 파일
  209만건
- 2M.SRCID.DSTID
  문서들의 링크를 표현한 파일
- RandowmString.txt
  임의의 문자열, 320만 줄로 구성

Wordcount 실행(wiki)

```
$ wget --load-cookies /tmp/cookies.txt
"https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies
/tmp/cookies.txt --keep-session-cookies --no-check-certificate
'https://docs.google.com/uc?export=download&id=1oaq1XvmPSyLBrtZkpncuC7YpG10t
Qi8U' -O- | sed -rn
's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=1oaq1XvmPSyLBrtZkpncuC7YpG10tQi8U" -O
data.tar.gz && rm -rf /tmp/cookies.txt

$ tar xvfz data.tar.gz
$ cd data
$ hdfs dfs -put 2M.ID.CONTENTS  /input
$ hdfs dfs -ls /input
```

Wordcount 실행(wiki)

$ hadoop jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar
wordcount /input/2M.ID.CONTENTS /output_wiki

$ hdfs dfs -ls /output_wiki
출력 결과가 900만 줄 (단어의 종류 900만 단어)
$ hdfs dfs -tail /output_wiki/part-r-00000
$ hdfs dfs -cat /output_wiki/part-r-00000 | grep world

Wordcount 결과(wiki)

- 특정 단어가 몇 번 나온 것은 확인가능
- 하지만 빈도수로 많이 나온 단어는?

- R이나 엑셀에서 워드카운트 결과 열리지 않음(900만 줄)
  - 알고 싶다면 빅데이터에서 직접해야 함
  - 하지만 **빅데이터에서는 전체정렬은 어려운 과제 -> TopN으로 대체**

- 결국 분석툴에서 처리하기 어려운 문제 발생

감사합니다