Sisteme Distribuite - Laborator 12

Operații Big Data cu Map-Reduce și Hadoop

Introducere

Big Data

Big Data[1] sunt date care conțin o **varietate** mai mare (tipurile de date care sunt disponibile, structurate, semi-structurate și nestructurate), cu un **volum** în continuă creștere (poate ajunge la terabytes de date sau chiar petabytes pentru unele companii) și cu o **viteză** tot mai mare (rata cu care datele sunt primite și poate prelucrate). Aceasta mai este cunoscută și ca „cei trei V”. Mai târziu, au mai fost incluși doi V: **valoare** și **veridicitate**. Valoarea intrinsecă a datelor e irelevantă dacă nu este descoperită. De asemenea, veridicitatea datelor este crucială pentru a putea lua decizii în baza lor.

Cazuri de utilizare:

- Dezvoltarea produsului: anticiparea cererilor clienților
- Mentenanță predictivă
- Experiența clientului
- Învățare automată (machine learning)
- Fraudă și conformitate

Ce presupune Big Data?

- Integrare: obținerea și prelucrarea datelor într-un format pe care analistul îl recunoaște;
- Gestionarea spațiului de stocare: datele pot fi stocate în cloud, local (în sedii), sau în ambele;
- Analiza datelor cu scopul de a lua decizii.

MapReduce

MapReduce[2] este o paradigmă de programare pentru calculul distribuit. Acest model implică, în general, existența unui nod de procesare cu rol de coordonator (sau master sau inițiator) și mai multe noduri de procesare cu rol de worker.

Modelul **MapReduce** cuprinde două etape: Map (task-ul de mapare) și Reduce (task-ul de reducere):

1. Etapa de **mapare** – primește un set de date de intrare și îl convertește în alt set de date, unde elementele individuale sunt „sparte” în perechi cheie-valoare
    - nodul cu rol de *coordonator* împarte problema „originală” în subprobleme și le distribuie către *workeri* pentru procesare;
    - divizarea problemei de lucru (a datelor de procesat) se realizează într-o manieră similară divide-et-impera – în unele cazuri nodurile worker pot diviza la rândul lor subproblema primită și pot trimite aceste subdiviziuni către alți workeri, rezultând o arhitectură arborescentă;
    - divizarea caracteristică acestei etape nu trebuie să coreleze efectiv dimensiunea datelor de intrare cu numărul de workeri din sistem; un worker poate primi mai multe subprobleme de rezolvat;
2. Etapa de **reducere** – primește ca date de intrare ieșirea de la etapa de mapare și combină perechile cheie-valoare astfel încât câmpul cheie va fi cuvântul, iar câmpul valoare va fi frecvența de apariție a cuvântului în text. Așadar, rezultă un set de date mai mic.

---

[1] https://www.oracle.com/big-data/guide/what-is-big-data.html

[2] https://www.ibm.com/analytics/hadoop/mapreduce

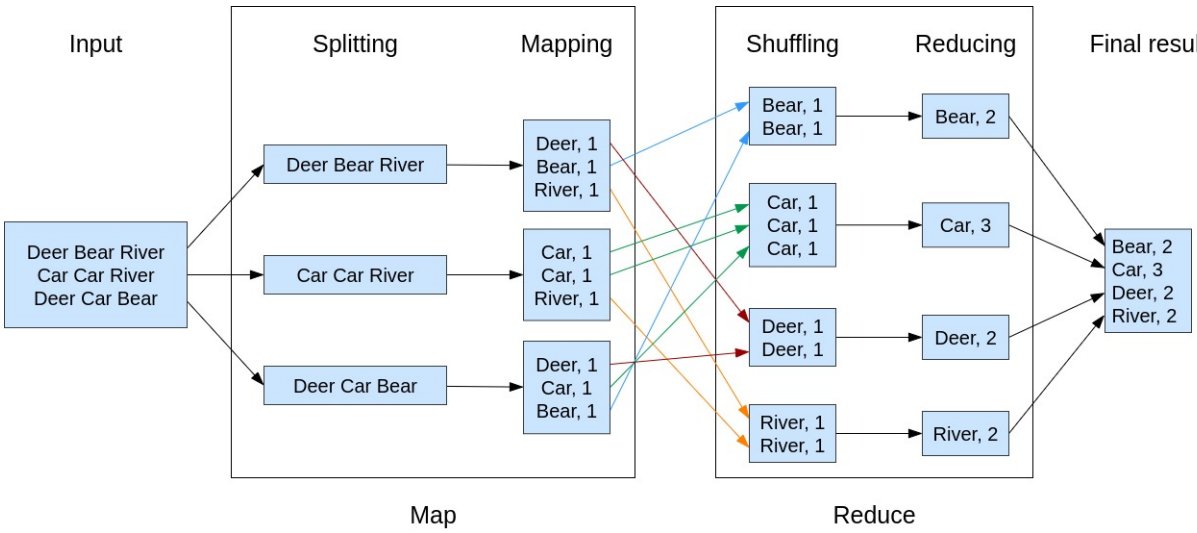
- nodul cu rol de *coordonator* (sau un set de noduri cu rol de worker „desemnat” de *coordonator*) colectează soluțiile subproblemelor și le combină pentru a obține rezultatul final al procesării dorite.

Michael Kleber[3] (Google Inc.) definește următoarele etape implicate în paradigma MapReduce:

1. pre-procesare – datele sunt pregătite pentru funcția de mapare
2. mapare – stabilirea datelor de interes
3. amestecare și sortare – datele pot fi organizate astfel încât să fie optimizată etapa de reducere
4. reducere – determinarea rezultatului
5. stocare rezultat

Avantaje:

- Scalabilitate masivă pe sute sau mii de servere într-un cluster;
- Flexibilitate (accesarea mai ușoară a mai multor surse și tipuri de date)
- Paralelizare automată;
- Echilibrarea încărcării (load balancing)
- Gestionarea defecțiunilor (machine failures) prin reasignarea task-ului unui alt woker;



Exemplu de aplicare a Map-Reduce pentru Word Counter

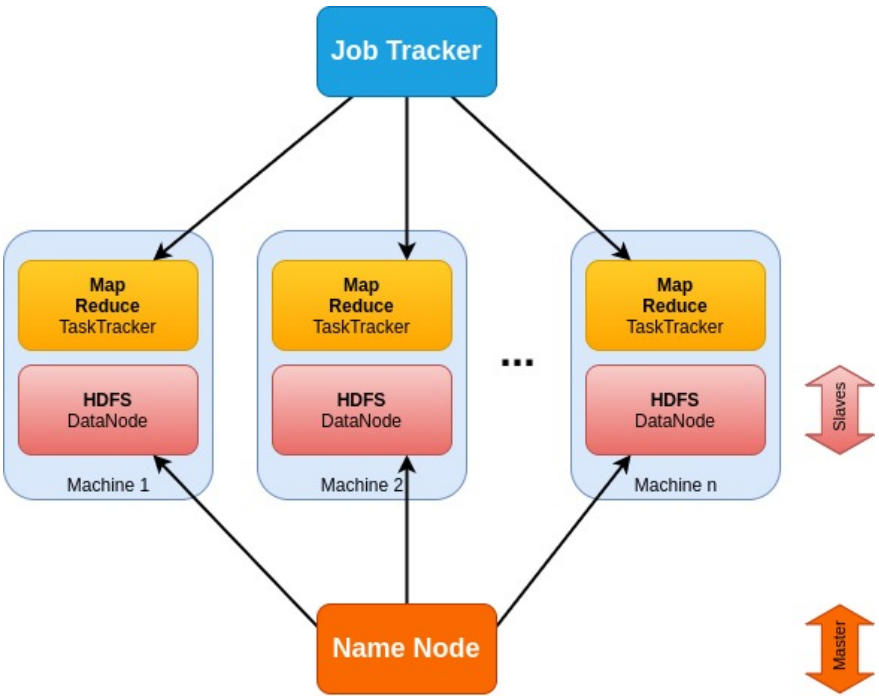
Hadoop

Hadoop este un sistem de fișiere distribuit și totodată un sistem de procesare distribuită a unor seturi imense de date pe cluster-e folosind paradigme de programare precum MapReduce. Acesta a fost introdus de Apache, fiind un framework open-source scris în java. Hadoop este proiectat să mărească spațiul de stocare și puterea de calcul de la un singur server până la mii de calculatoare. De asemenea, acesta a fost proiectat să fie „tolerant la defecte” (fault tolerant), adică în cazul în care un nod din cluster se defectează, datele nu se vor pierde.

Deși framework-ul este scris în java, el conține un utilitar numit Hadoop Streaming (utilizat în cadrul exemplului din laborator) ce permite utilizatorilor să creeze și să execute task-uri cu orice tip de executabil (ex.: python, java) ca Mapper și/sau Reducer.

Hadoop are o arhitectură de tip master-slave, reprezentată schematic în figura de mai jos:

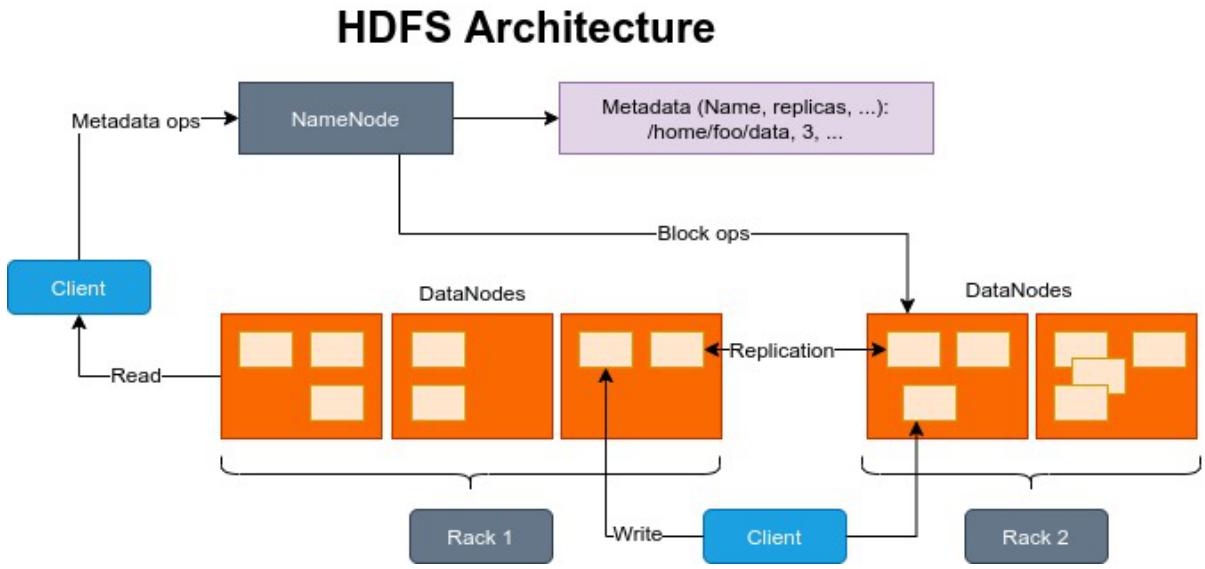
[3] http://www.csce.uark.edu/~mqhuang/courses/5013/s2018/lecture/5_intro_to_mapreduce.pdf



Principalele componente ale Apache Hadoop sunt:

- **NameNode-ul** – controlează funcționarea job-urilor de date;
- **DataNode-ul** – scrie datele în blocuri în sistemul local de stocare și replică blocuri de date către alte DataNode-uri;
- **JobTracker-ul** – trimite job-uri de tip MapReduce către nodurile din cluster;
- **TaskTracker-ul** – acceptă task-uri de la JobTracker;
- **Yarn (Yet Another Resource Manager)** – pornește componentele ResourceManager și NodeManager; Yarn-ul este un framework pentru programarea job-urilor (job scheduling) și gestionarea resurselor din cluster.

Apache HDFS (Hadoop Distributed File System)[4] este un sistem de fișiere structurat pe blocuri în care fiecare fișier este împărțit în blocuri de dimensiune pre-determinată, acestea fiind stocate într-un cluster cu una sau mai multe mașini de calcul. HDFS are o arhitectură master/slave ce poate fi observată în figura de mai jos, unde cluster-ul este format dintr-un singur NameNode (nodul master) și toate celelalte noduri sunt DataNode-uri (noduri slave).



[4] https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/

Configurarea mediului de lucru

Instalare Oracle Java versiunea 1.8.0_221

Atenţie! Hadoop NU funcţionează cu versiunile de Java > 8 , aşadar trebuie să folosiţi o versiune de **Java SDK cel mult egală cu 8**. Verificaţi versiunea de Java cu următoarea comandă:

```
java -version
```

În cazul în care sunt disponibile mai multe versiuni de Java pe staţia de lucru (cu sistemul de operare Debian), iar acestea sunt înregistrate ca alternative, se poate schimba versiunea de Java la cerere, folosind comanda:

```
sudo update-alternatives --config java
```

La executarea ultimei comenzi, este necesară introducerea indexului versiunii java ca în figura de mai jos. Se alege java 8.

```
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      manual mode
  2            /usr/lib/jvm/oracle-java8-jdk-amd64/bin/java     1081      manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
```

Selectarea versiunii de java

Similar, se va configura şi javac (java compiler):

```
sudo update-alternatives --config javac
```

```
There are 2 choices for the alternative javac (providing /usr/bin/javac).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-11-openjdk-amd64/bin/javac     1111      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/javac     1111      manual mode
  2            /usr/lib/jvm/oracle-java8-jdk-amd64/bin/javac    1071      manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
```

Selectarea versiunii de javac (Java Compiler)

Dacă versiunea Java 8 nu este instalată, urmați pașii de mai jos:

1. Se va descărca de pe https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html ***jdk-8u251-linux-x64.tar.gz.***
2. Se va descărca de pe https://www.oracle.com/java/technologies/javase-jre8-downloads.html ***jre-8u251-linux-x64.tar.gz***

Se vor executa următoarele comenzi:

```
tar -xvf jdk-8u251-linux-x64.tar.gz
sudo mv jdk1.8.0_251 /usr/lib/jvm
tar -xvf jre-8u251-linux-x64.tar.gz
sudo mv jre1.8.0_251 /usr/lib/jvm

sudo update-alternatives --install /usr/bin/java java
/usr/lib/jvm/jdk1.8.0_251 1103
sudo update-alternatives --config java
```

La ultima comandă executată de mai sus, se introduce numărul opțiunii */usr/lib/jvm/jdk1.8.0_251.*

Dacă este setată variabila *JAVA_HOME* și în /etc/environment, va trebui actualizată calea:

```
sudo vim /etc/environment
```

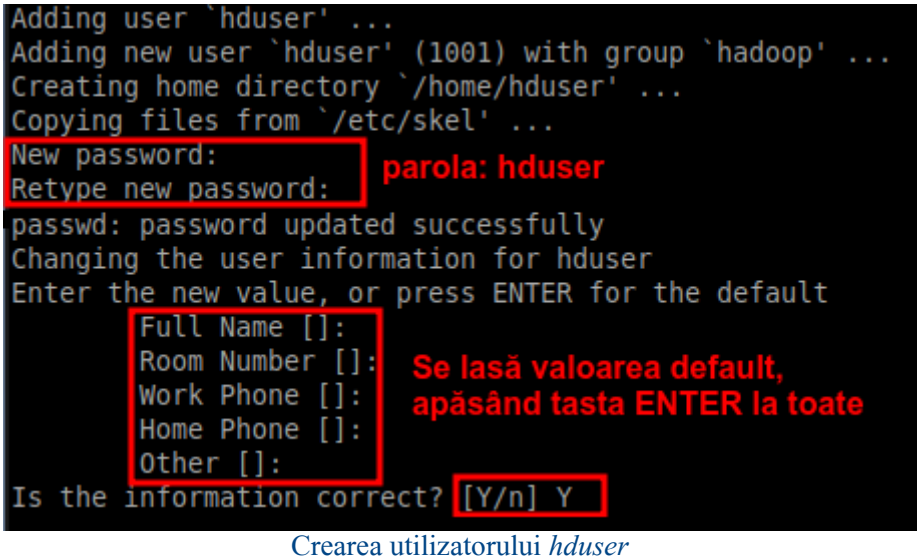
JAVA_HOME=/usr/lib/jvm/oracle-java8-jdk-amd64

```
. /etc/environment # reload environment file
java -version # check java version
```

Instalare Apache Hadoop versiunea 3.2.1

Se adaugă un utilizator de sistem pentru Hadoop

```
sudo addgroup hadoop  # create group hadoop
sudo adduser --ingroup hadoop hduser  # create user hduser inside the hadoop group
```



Crearea utilizatorului *hduser*

Se vor executa într-un terminal următoarele comenzi:

```
cd ~/Downloads  # change directory
wget https://www-eu.apache.org/dist/hadoop/common/stable/hadoop-3.2.1.tar.gz # download hadoop
tar -xvf hadoop-3.2.1.tar.gz # extract from archive
sudo chown -R hduser:hadoop hadoop-3.2.1 # change owner and group to hduser and hadoop
sudo mv hadoop-3.2.1 hadoop  # rename to hadoop
sudo mv hadoop /opt # move hadoop to /opt directory
```

Instalare și configurare SSH

Pentru instalare, se execută comenzile de mai jos:

```
sudo apt update
sudo apt install openssh-server
```

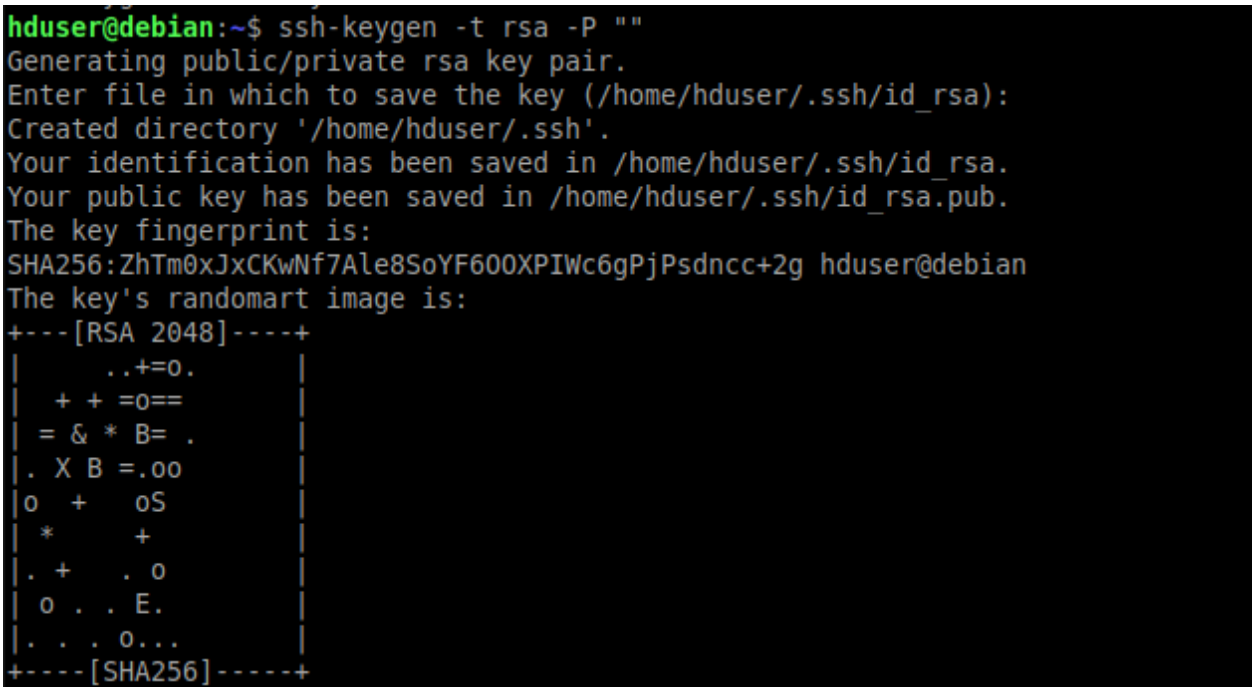
Pentru a verifica instalarea corectă, se poate utiliza comanda:

```
sudo systemctl status sshd
```

Este necesară generarea unei chei SSH. La întrebarea cu locația cheii, se apasă ENTER pentru locația default.

```
su - hduser # switch user to hduser
# <password for hduser> # enter the password for hduser

ssh-keygen -t rsa -P "" # generate SSH key
```



Generarea unei chei SSH

Este necesară autorizarea cheii SSH pentru a putea realiza și testa o conexiune pe localhost.

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys # authorize SSH key
ssh localhost # check connection
```

Conexiunea SSH este stabilită cu setările default.
[OPȚIONAL]: Dacă se dorește securizarea accesului la server, se recomandă modificarea configurărilor SSH (a se vedea fișierul de configurare *sshd_config* anexat și comentariile aferente, introduse prin caracterul #). Pentru modificarea setărilor SSH, se deschide fișierul */etc/ssh/sshd_config* cu editorul preferat (sunt necesare drepturi de admin, deci se execută cu *sudo*).

```
sudo featherpad /etc/ssh/sshd_config
```

Dacă s-au realizat modificări în fișierul de configurare al server-ului SSH, trebuie forțată reîncărcarea configurărilor cu comanda:

```
sudo service ssh reload
```

Configurarea variabilelor HADOOP_HOME și JAVA_HOME

Se actualizează fișierul /home/hduser/*.bashrc*.

```
su - hduser
# <password for hduser>
vim ~/.bashrc # deschidere fisier pentru editare
```

```
# sau
nano ~/.bashrc
# sau
mousepad ~/.bashrc  # pentru un editor grafic
```

Se adaugă la finalul fișierului următoarele linii:

```
# Set HADOOP HOME
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

# Set JAVA HOME
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_251 # check path
export PATH=$JAVA_HOME/bin:$PATH

# Mofify pdsh's default rcmd to ssh
export PDSH_RCMD_TYPE=ssh
# export HADOOP_SSH_OPTS="-p 4567"  # if you modified the port
```

Se reîncarcă configurările anterioare executând comanda:

```
source ~/.bashrc
```

Configurare Apache Hadoop

Pentru configurarea Apache Hadoop, trebuie editate mai multe fișiere. Într-un **terminal nou**, se execută următoarele comenzi:

```
sudo vim <nume_fisier>
# exemplu:
sudo mousepad /opt/hadoop/etc/hadoop/hadoop-env.sh
```

1. /opt/hadoop/etc/hadoop/hadoop-env.sh
   Se adaugă următoarele linii:

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_251  # check path
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

2. /opt/hadoop/etc/hadoop/core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
   <property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:9000</value>
   </property>
</configuration>
```

3. /opt/hadoop/etc/hadoop/mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
   <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
   </property>
   <property>
       <name>yarn.app.mapreduce.am.resource.mb</name>
       <value>1024</value>
   </property>
   <property>
       <name>yarn.app.mapreduce.am.command-opts</name>
       <value>-Xmx983m</value>
   </property>
   <property>
       <name>mapreduce.map.memory.mb</name>
       <value>1024</value>
   </property>
   <property>
       <name>mapreduce.reduce.memory.mb</name>
       <value>1024</value>
   </property>
   <property>
       <name>mapreduce.map.java.opts</name>
       <value>-Xmx983m</value>
   </property>
   <property>
       <name>mapreduce.reduce.java.opts</name>
       <value>-Xmx983m</value>
   </property>
   <property>
     <name>yarn.scheduler.capacity.maximum-am-resource-percent</name>
     <value>100</value>
   </property>
   <property>
     <name>yarn.app.mapreduce.am.env</name>
     <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
   </property>
   <property>
     <name>mapreduce.map.env</name>
     <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
   </property>
   <property>
     <name>mapreduce.reduce.env</name>
     <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
   </property>
</configuration>
```

4. /opt/hadoop/etc/hadoop/hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
```

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

5. /opt/hadoop/etc/hadoop/yarn-site.xml

```
<?xml version="1.0"?>
<configuration>
   <property>
       <name>yarn.scheduler.minimum-allocation-mb</name>
       <value>1228</value>
   </property>
   <property>
       <name>yarn.scheduler.maximum-allocation-mb</name>
       <value>9830</value>
   </property>
   <property>
       <name>yarn.nodemanager.resource.memory-mb</name>
       <value>9830</value>
   </property>
   <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>
   <property>
      <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
      <value>org.apache.hadoop.mapred.ShuffleHandler</value>
   </property>
   <property>
     <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-
per-disk-percentage</name>
     <value>100</value>
   </property>
</configuration>
```

6. */etc/hosts*

Pentru a obține adresa IP privată, se execută în terminal comanda:

```
ip a | grep "inet 192" | awk '{print $2}' | cut -d/ -f1
```

Pentru a obține hostname-ul, rulați în terminal comanda:

```
hostname
```

Se deschide cu drepturi de administrator fișierul */etc/hosts*:

```
sudo featherpad /etc/hosts
```

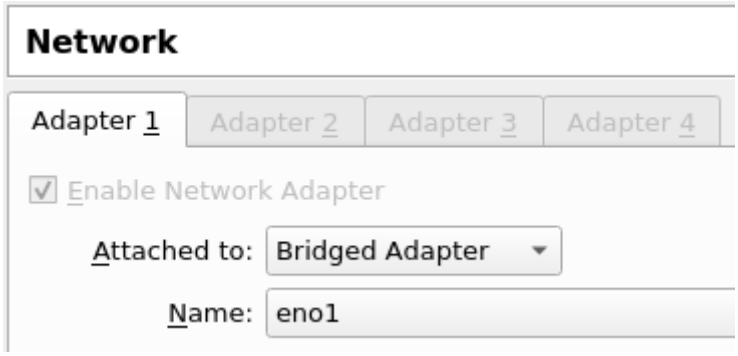
Se modifică cu conținutul:

```
#127.0.0.1 localhost
#127.0.1.1 debian

# în loc de 192.168.0.100 pune i adresa IP privată ob inută anterior
# în loc de debian pune i hostname-ul ob inut anterior
192.168.0.100 debian localhost

# The following lines are desirable for IPv6 capable hosts
#::1     localhost ip6-localhost ip6-loopback
#ff02::1 ip6-allnodes
#ff02::2 ip6-allrouters
```

Dacă se utilizează o mașină virtuală, se vor modifica setările pentru Network ca în imaginea de mai jos:



Configurarea setărilor de rețea pentru VirtualBox

Observație: În loc de *eno1* este posibil să fie afișat *eth0* sau alte denumiri de rețea. Nu se selectează rețelele wireless (ex.: wlp2s0).

Inițializare HDFS

Într-un **terminal deschis pe user-ul *hduser***, pentru formatarea componentei NameNode, se execută comanda de mai jos.

```
hdfs namenode -format
```

Atenție: aceasta va șterge toate fișierele pe care stocate anterior în hdfs. Dacă se reformatează, se vor relua pașii prin care se copiază datele în hdfs.
Dacă apare întrebarea „Re-format filesystem in Storage Directory root= /tmp/hadoop-hduser/dfs/name; location= null ? (Y or N)” se răspunde Y și se apasă tasta ENTER.

Pentru a porni/opri cluster-ul cu un singur nod, există două posibilități

1. Se pornesc toate componentele:

```
start-all.sh
# respectiv
stop-all.sh
```

2. Se pornesc componentele individual:

```
start-dfs.sh
start-yarn.sh
jps # afiseaza componentele active
# respectiv
stop-yarn.sh
```

```
stop-dfs.sh
```

Pentru crearea directorului */user/hduser/input* în hdfs, se execută comanda:

```
hdfs dfs -mkdir -p /user/hduser/input
```

Exemplu MapReduce

Exemplul propus este un Word Counter care folosește două scripturi python: mapper.py și reducer.py. Mapper-ul sparge efectiv fișierele text în perechi de forma:

```
<cuvânt_0, 1>
<cuvânt_1, 1>
<cuvânt_0, 1>
<cuvânt_0, 1>
```

Reducer-ul reunește acele perechi, obținând astfel numărul de apariții ale cuvintelor. Rezultatul final este:

```
<cuvânt_0, 3>
<cuvânt_1, 1>
```

Codul sursă

- *~/exemplu_mapreduce/mapper.py*

```
#!/usr/bin/env python
"""mapper.py"""

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        # write the results to STDOUT
        # this output will be the input for the Reduce step
        print('%s\t%s' % (word, 1))
```

- *~/exemplu_mapreduce/reducer.py*

```
#!/usr/bin/env python
"""reducer.py"""

import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()
    word, count = line.split()

    try:
```

```
        count = int(count)
    except ValueError:
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word != word:
        if current_word:
            print('%s\t%s' % (current_word, current_count - 1))
        current_count = count
        current_word = word
    current_count += count

if current_count > 0:
    print('%s\t%s' % (current_word, current_count - 1))
```

Se deschide **un nou terminal** (pe user-ul curent, nu *hduser*) în folder-ul *$HOME* și se execută comenzile:

```
sudo chown -R hduser:hadoop exemplu_mapreduce
sudo mv exemplu_mapreduce/ /home/hduser
```

Se poate închide terminalul curent și **se revine la cel anterior (în care este logat *hduser*)**. Acum se copiază fișierele text în hdfs (Hadoop Distributed File System) executând comanda:

```
start-all.sh
hdfs dfs -mkdir -p /user/hduser/input
hdfs dfs -put /home/hduser/exemplu_mapreduce/input/*
/user/hduser/input
```

Dacă apare eroarea „put: File /user/hduser/input/pg20417.txt._COPYING_ could only be written to 0 of the 1 minReplication nodes. There are 0 datanode(s) running and 0 node(s) are excluded in this operation.”, executați următoarele comenzi:

```
stop-all.sh
rm -rf /tmp/hadoop-*
hdfs namenode -format
start-all.sh
hdfs dfs -mkdir -p /user/hduser/input
hdfs dfs -put /home/hduser/exemplu_mapreduce/input/*
/user/hduser/input
```

Dacă se observă un mesaj de eroare *„ipc.Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 0 time(s)”*, Resource Manager-ul nu este activ. Executați comanda:

```
start-yarn.sh
```

Pentru testarea exemplului, executați comanda de mai jos:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming*.jar
-input /user/hduser/input -output /user/hduser/output -mapper
/home/hduser/exemplu_mapreduce/mapper.py -reducer
/home/hduser/exemplu_mapreduce/reducer.py -file
```

```
/home/hduser/exemplu_mapreduce/mapper.py -file
/home/hduser/exemplu_mapreduce/reducer.py
```

Dacă au fost urmați toți pașii, output-ul obținut este următorul:

```
2020-02-07 03:13:47,564 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1581063088186_0001/
2020-02-07 03:13:47,568 INFO mapreduce.Job: Running job: job_1581063088186_0001
2020-02-07 03:13:59,894 INFO mapreduce.Job: Job job_1581063088186_0001 running in uber mode : false
2020-02-07 03:13:59,899 INFO mapreduce.Job:  map 0% reduce 0%
2020-02-07 03:14:17,190 INFO mapreduce.Job:  map 100% reduce 0%
2020-02-07 03:14:25,293 INFO mapreduce.Job:  map 100% reduce 100%
2020-02-07 03:14:26,321 INFO mapreduce.Job: Job job_1581063088186_0001 completed successfully
2020-02-07 03:14:26,466 INFO mapreduce.Job: Counters: 55
```

Pentru verificarea rezultatului, se listează conținutul directorului *output*, apoi se copiază fișierul de ieșire în exteriorul hdfs.

```
hdfs dfs -ls /user/hduser/output
```

```
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2020-02-07 03:14 /user/hduser/output/_SUCCESS
-rw-r--r--   1 hduser supergroup     888040 2020-02-07 03:14 /user/hduser/output/part-00000
```

```
hdfs dfs -get /user/hduser/output/part-00000 /home/hduser/
xdg-open /home/hduser/part-00000
```

Rezultatul va fi un fișier text în care pe fiecare linie va fi un cuvânt și numărul de apariții ale cuvântului respectiv în fișierele de intrare.

```
"Italians,"     2
"Je,    2
"La     2
"Le     4
"Leonardo       2
"Les    2
"Libricciuolo   2
"Libro  2
"Lionardo       2
"Lionardo"      2
"Lipsia,        2
"Locho  2
"Magnifico",    2
"Majestati      2
"Man    2
"Many   2
```

Rezultatul obținut

Aplicaţii şi teme

Teme de laborator

1. Să se implementeze un algoritm de sortare disitribuit (se va utiliza sistemul de fișiere distribuit Hadoop), bazat pe algoritmul Map-Reduce. Funcția de mapare va extrage cuvintele dintr-un set de fișiere text și va returna perechi de forma: `<prima_literă_din_cuvânt, cuvântul>`, iar funcția de reducere va reuni perechile generate de funcția de mapare în perechi de forma: `<literă, [cuvânt0, cuvânt1, ...]>`, valoarea fiind practic o listă de cuvinte care încep cu acea literă.

2. Să se implementeze o funcție GREP distribuită (se va utiliza Hadoop) pe baza algoritmului Map-Reduce. Funcția de mapare va returna liniile din output-ul unei comenzi linux (obținut cu ajutorul modulului *subprocess*) care corespund unui regex dat ca parametru, iar funcția de reducere va reuni acele linii într-un singur string.

Teme pentru acasă

1. Să se implementeze o aplicație care generează utilizând algoritmul Map-Reduce (și Hadoop) un Site-Map pentru fiecare pagină WEB dintr-un set de URL-uri. Funcția de mapare va face un request HTTP de tip GET pentru conținutul unei pagini, va parsa HTML-ul pentru a extrage toate ancorele (link-urile) și va genera perechi de forma `<URL_pagină, URL_intern>`. Funcția de reducere va reuni acele perechi, returnând elemente de forma `<URL_pagină, [URL_intern0, URL_intern1, ...]>` în care valoarea este o listă ce conține toate URL-urile interne din acea pagină (practic, Site-Map-ul).

2. Să se implementeze o aplicație care extrage istoricul URL-urilor accesate dintr-un browser (spre exemplu Firefox) și calculează pe baza algoritmului Map-Reduce frecvența de accesare a acestora. Funcția de mapare va genera perechi de forma `<URL, 1>`, iar funcția de reducere va aduna toate valorile comune pentru aceeași pagină, generând o pereche de forma: `<URL, frecventa_de_accesare>`. Observație: se pot elimina fragmentele din URL (`#fragment`).
   HINT: Pentru browser-ul Firefox, baza de date SQLite3 cu istoricul URL-urilor accesate se află în locația: *~/.mozilla/firefox/abcd12ef.default/places.sqlite* (partea cu roșu variază), tabela de interes fiind *moz_places*. Întrucât *places.sqlite* oferă deja frecvența de accesare, funcția map va genera perechi de forma <host, frecvență_URL> (exemplu: <google.com, 12345>, <google.com, 3120>)

Bibliografie

[1] Jeffrey Dean, Sanjay Ghemawat, *„MapReduce: Simplified data processing on large clusters”*, OSDI, 2004
[2] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, *„An Introduction to Information Retrieval”*, Cambridge University Press , 2009.
[3] Garry Turkington, Tanmay Deshpande, Sandeep Karanth, *„Hadoop: Data processing and modelling”*
[4] Srinath Perera, Thilina Gunarathne, *„Hadoop MapReduce Cookbook”*
[5] Documentația Apache Hadoop: https://hadoop.apache.org/docs/current/
[6] Documentația SQLite3: https://docs.python.org/3/library/sqlite3.html