UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAŞI
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DOMENIUL: Calculatoare şi tehnologia informaţiei
SPECIALIZAREA: Tehnologia informaţiei

Sisteme Distribuite - Laborator 12

Operații Big Data folosind paradigma de programare MapReduce

Iaşi, 2022

Sisteme Distribuite - Laborator 12

Operații Big Data cu Map-Reduce și Hadoop

Introducere

Big Data

Big Data a apărut în contextul societăților informaționale unde se produceau fluxuri enorme de informații de toate tipurile și era necesară dezvoltarea a noi tehnici de analiză control vizualizare. După cum am discutat la curs există la ora actuală zece dimensiuni pentru big data și se asteaptă o crestere rapidă (la un numaăr mult mai mare pe măsură ce se dezvoltă și instrumentele teorietice împreună cu creșterea diversității și complexității aplicațiilor specifice. Tehnicile și aplicațiile specifice big data sunt esențiale în contextul dezvoltării ciberspațiului. Aplicațiile sunt virtual infinite pornind de la sisteme de comandă control (la un nivel global, statal, corporatist, social, organizațional, grup și microgrup) asistate sau chiar automate (bazate pe AI avansat) și ajungând la aplicații industriale (Industria 4.0) sau chiar de divertisment (industria jocurilor și rețelelor sociale - vezi prima încercare de creare a unui spațiu virtual (tipic cyberspațiului) de către Facebook - Meth

MapReduce

MapReduce este o abordare specifică calculului paralel din anii 60 care a fost adoptată pentru calculul în cluster/grid şi mai apoi nor (pornind de la calcul functional). Ea se încadrează din punct de vedere al implementării în sistemele distribuite în categoria master-slave deci centralizată cu un control arborescent din punct de vedere arhitectural. Acest model implică, în general, existenţa unui nod de procesare cu rol de coordonator (master) şi mai multe noduri de procesare (slave ulterior înlocuit cu termenul de muncitor/lucător(worker) din cauza corectitudinii politice). Este inspirat din operațiile de reducere utilizate în mod comun în limbajele de programare funcțională cum ar fi Lisp. Sub acest nume a fost introdus in 2003 de Jeffrey Dean și Sanjay Ghemawat de la Google și publicat în 2004.

În esență procesarea primește la intrare un set de perechi cheie valoare aflate într-un fișier. Apoi urmează o procesare în paralel care presupune mai multe procese de tip Map sau Reduce care la rândul lor produc chei valoare

Funcția de transformare(Map) : **(key1, val1) → (key2, val2)** (procesul de transformare în alt spațiu prin asocierea unui index la una sau mai multe date) (mapare este un termen intrat forțat în limba română doarece referă procesul de transformare între două spații cu ajutorul unei (sau unui set de) funcții).

Funcția de reducere: (key2, [val2 list]) → [val3] (procesul de reducere de fapt de numărare a entităților identice în abordările simplificate)

Din punct de vedere al abordarii Google există mai multe etape în implementarea acestei abordări

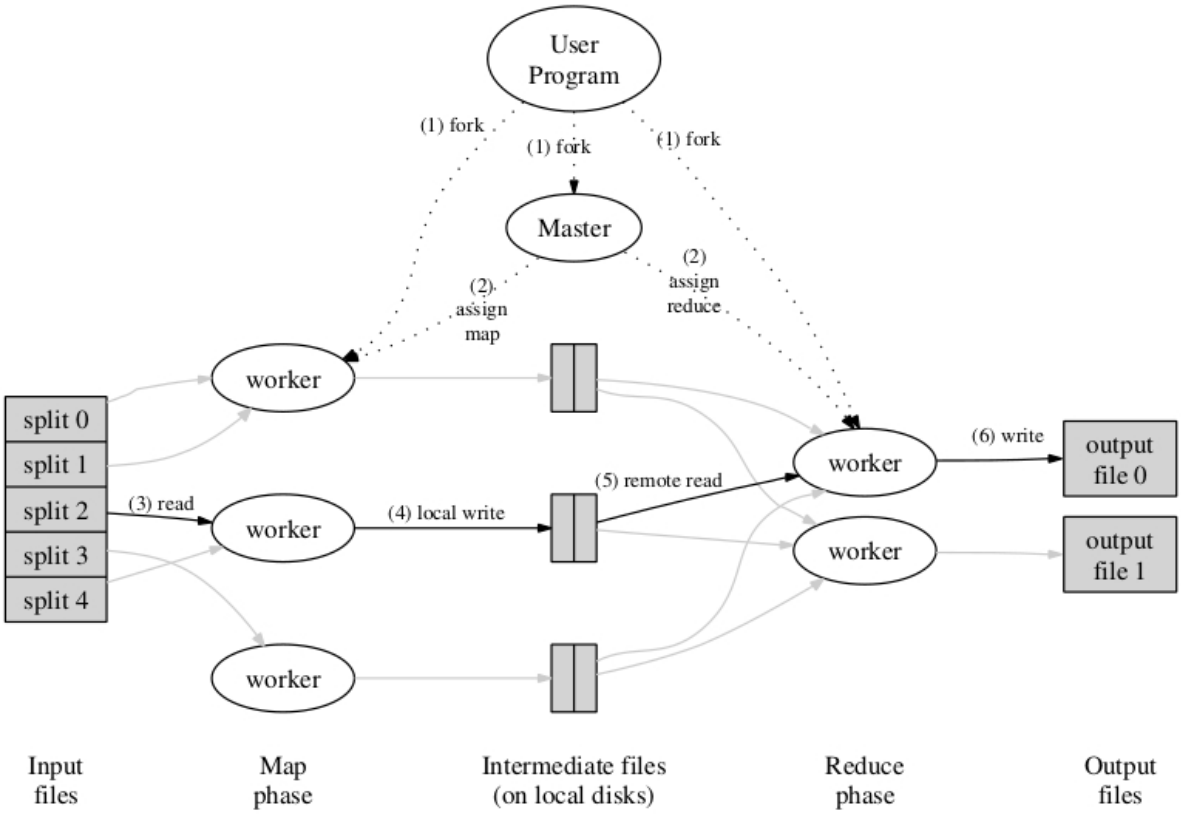
1. se începe prin împărțirea fișierului de intrare în M unități de dimensiuni egale (bătrânul divide și stăpâneste într-o versiune o idee mai avansată)

2. Se asociază programul de procesare la noduri de calcul de tip coordonator și la executanți (fiecare primește o bucată din cele M)

3. Vor exista sarcini de tip transformare (map) și reducere (reduce). Echilibrarea dinamică a încărcării va fi realizată de coordonator

Dacă ne uităm în figura de mai jos observăm că pentru a evita timpii morți care pot apărea datorită unor abordări wave (de exemplu bazați pe barieră) se utilizează acele zone

temporare (pe disc în abordarea Google) care odată ce un nod care efectua o transformare a terminat anunță coordonatorul care asociază imediat un executant care face reducerea. Pentru a nu pierde timp până se termină toate procesele de reducere Google recomandă ca aceste fișiere parțiale să intre mai departe tot separat în alte procesări și de aceea în unele cazuri nodurile lucători (worker) pot împărți la rândul lor subproblema primită și pot trimite aceste subdiviziuni către alți workeri, rezultând o arhitectură arborescentă de procesare. T**otuși la un moment dat se va astepta ca sa se termine toate operațiile pentru a face centralizarea și interpetarea tuturor rezultatelor. Din această cauză multe implementări simpliste prezintă acest algoritm** cu următoarea variație și a nume că nodul cu rol de *coordonator* (sau chiar un set de noduri cu rol de muncitor care este „desemnat" de *coordonator*) colectează soluțiile subproblemelor și le combină pentru a obține rezultatul final al procesării dorite.

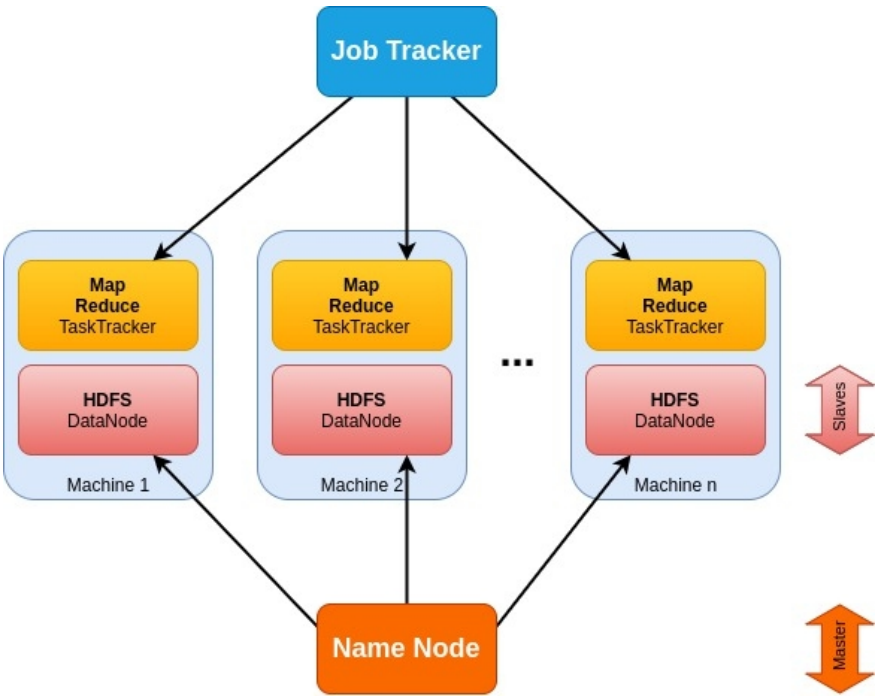


Hadoop

Hadoop este un sistem de fișiere distribuit și totodată un sistem de procesare distribuită a unor seturi imense de date pe clustere folosind abordări paralele și distribuite (ați avut un curs pe subiect) din care face parte și 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.

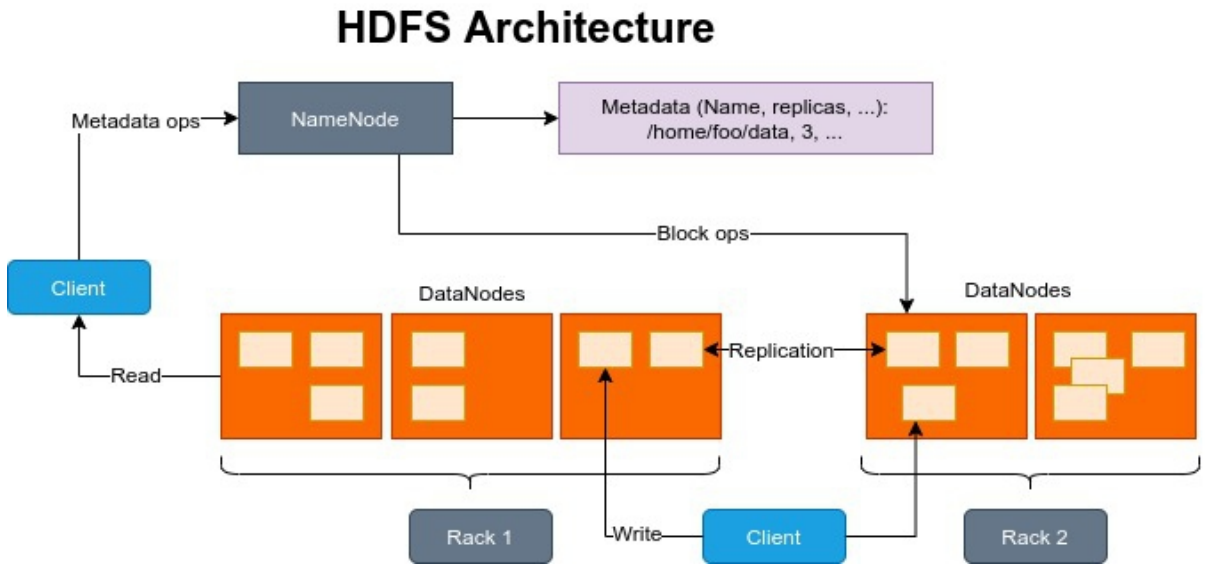
Hadoop are o arhitectură de tip master-slave, reprezentată schematic în figura de mai jos:



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)[1] 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).



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

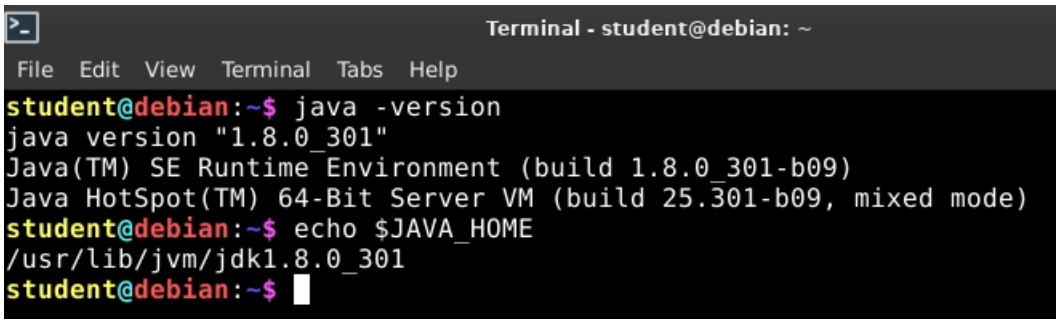
Atenţie! Cei care lucrează de pe staţiile din laborator, săriţi direct la Secțiunea "Configurare Apache Hadoop, **subpunctul 6** (modificarea fişierului /etc/hosts)".

Configurarea mediului de lucru

Instalare Oracle Java versiunea 1.8.0_301

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
```

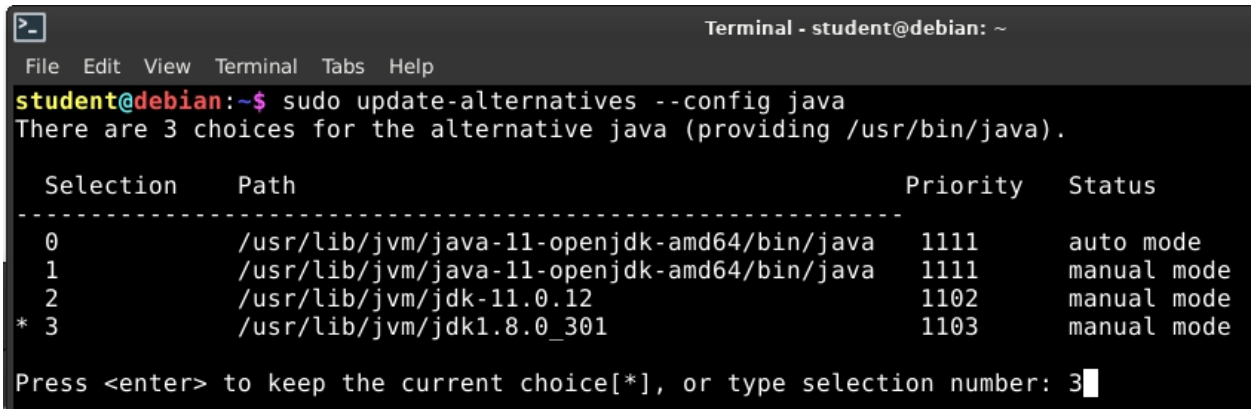


Verificarea versiunii de java

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.



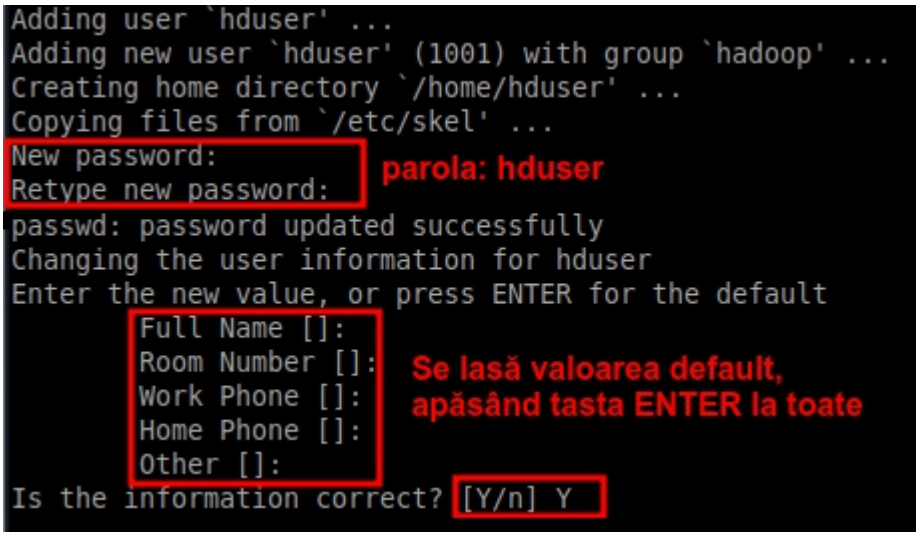
Selectarea versiunii de java

Pentru instalarea jdk1.8 și configurarea JAVA_HOME consultați laboratorul 1.

Instalare Apache Hadoop versiunea 3.3.1

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-
3.3.1.tar.gz # download hadoop
tar -xvf hadoop-3.3.1.tar.gz # extract from archive
sudo chown -R hduser:hadoop hadoop-3.3.1 # change owner and group to
hduser and hadoop
sudo mv hadoop-3.3.1 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
```

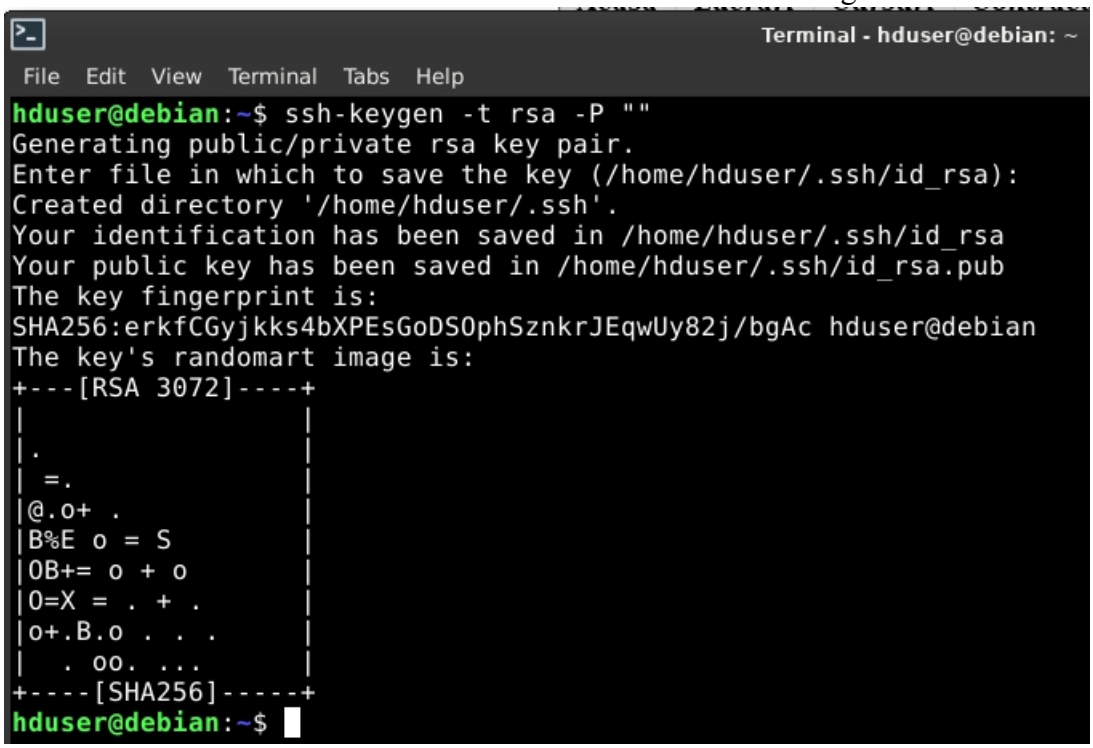
Pentru a verifica instalarea corectă, se poate utiliza comanda:

```
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 mousepad /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_301 # 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_301  # 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 mousepad /etc/hosts
```

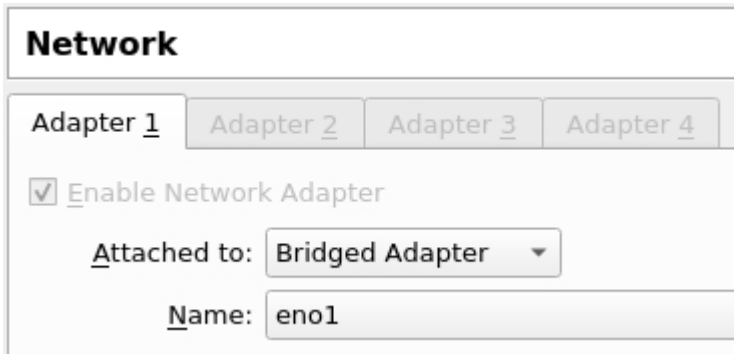
Se modifică cu conținutul:

```
#127.0.0.1 localhost
#127.0.1.1 debian

# în loc de 192.168.0.100 puneți adresa IP privată obținută anterior
# în loc de debian puneți hostname-ul obținut 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 python3
"""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 python3
"""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:

```
2021-11-30 15:02:02,186 INFO mapreduce.Job: The url to track the job: http://debian:8088/proxy/application_1638270526761_0030/
2021-11-30 15:02:02,187 INFO mapreduce.Job: Running job: job_1638270526761_0030
2021-11-30 15:02:06,236 INFO mapreduce.Job: Job job_1638270526761_0030 running in uber mode : false
2021-11-30 15:02:06,239 INFO mapreduce.Job:  map 0% reduce 0%
2021-11-30 15:02:12,361 INFO mapreduce.Job:  map 67% reduce 0%
2021-11-30 15:02:13,367 INFO mapreduce.Job:  map 100% reduce 0%
2021-11-30 15:02:16,382 INFO mapreduce.Job:  map 100% reduce 100%
2021-11-30 15:02:17,411 INFO mapreduce.Job: Job job_1638270526761_0030 completed successfully
2021-11-30 15:02:17,473 INFO mapreduce.Job: Counters: 55
```

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

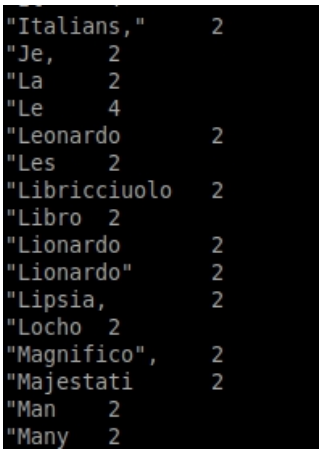
```
hdfs dfs -ls /user/hduser/output
```

```
hduser@debian:~/exemplu_mapreduce/input$ hdfs dfs -ls /user/hduser/output
2021-11-30 15:03:24,150 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2021-11-30 15:02 /user/hduser/output/_SUCCESS
-rw-r--r--   1 hduser supergroup     887486 2021-11-30 15:02 /user/hduser/output/part-00000
hduser@debian:~/exemplu_mapreduce/input$
```

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

xdg-open /home/hduser/part-00000
# sau
mousepad /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.



Rezultatul obținut

Pentru execuția repetată a comenzii de map-reduce este nevoie de ștergerea directorului de output:

```
hdfs dfs -rm -r /user/hduser/output
```

Pentru eventuale erori care mai pot apărea, analizați fișierele de eroare *stderr* care se află la calea */opt/hadoop/logs/userlogs/application_*/container_**.

```
sudo find /opt/hadoop/logs/userlogs -name stderr
```

Aplicaţii şi teme

Teme de laborator

1. Să se implementeze un algoritm de sortare distribuit (se va utiliza sistemul de fişiere 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). Apoi să se analizeze o analiză cu privire la frecvența de apariție a unor termeni în fiecare pagină și să se afișeze primii cinci pentru fiecare pagină dar și primii cinci care apar în tot setul de pagini utilizate
2. Sa se reia teme 1 utilizand accesare https.

3. 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`). `Apoi se va calcula si afisa functia de distributie a resepctivelor frecvente si se vor afisa explicit primele 5 din lista celor mai cautate cuvinte.`
   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>). Să se afișeze primile 5 cele mai vizitate site-uri.

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