

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN

Khai thác dữ liệu

ĐỀ TÀI:

**TỐI ƯU HÓA TỐC ĐỘ THỰC THI CỦA THUẬT TOÁN K-MEANS
BẰNG CÔNG NGHỆ HADOOP**

GVHD: Ngô Dương Hà

Sinh viên thực hiện:

Trần Hữu Lợi - 2001181186

TP.Hồ Chí Minh, Ngày 16 Tháng 12 Năm 2020

➤ MỤC LỤC

I. Giới thiệu.....	3
1. Đặt vấn đề.....	3
2. Phạm vi đề tài.....	3
II. Hadoop.....	3
1. Hadoop là gì?.....	3
2. Kiến trúc Hadoop.....	4
3. MapReduce.....	4
4. Hệ thống tệp phân tán Hadoop.....	4
5. Hadoop hoạt động như thế nào?.....	5
6. Ưu điểm.....	5
7. Nhược điểm.....	6
III. Cài đặt Hadoop.....	7
1. Yêu cầu về phần mềm trước khi cài đặt.....	7
2. Tiến hành cài đặt.....	7
3. Một số lệnh thường dùng trong Hadoop.....	20
IV. Thuật toán K-Means.....	21
1. Lý thuyết cơ bản.....	21
2. Áp dụng mô hình MapReduce vào thuật toán K-Means.....	21
3. Thực nghiệm.....	22
V. Kết luận và định hướng phát triển.....	25
VI. Tài liệu tham khảo.....	26

I. Giới thiệu

1. Đặt vấn đề

Với sự phát triển vượt bậc về công nghệ hiện nay, việc xử lý dữ liệu cũng như thực hiện đa tác vụ cùng một lúc đã trở nên vô cùng cần thiết và tính toán và xử lý song song trên thiết bị máy tính đã dần trở thành các tiêu chuẩn tối thiểu cho các nhà phát triển phần mềm cũng như phần cứng về sau.

Tuy nhiên, dữ liệu mà ta thu thập được đã trở nên vô cùng lớn, việc tính toán và xử lý khối lượng lớn dữ liệu như vậy tốn một khối lượng lớn thời gian để hoàn thành. Do đó, các nhà phát triển, doanh nghiệp,... đều muốn tận dụng tối đa nguồn tài nguyên tính toán của mình, mà rõ hơn ở đây, đó chính là máy tính. Việc xây dựng một máy tính có cấu hình mạnh là vô cùng tốn kém chi phí. Vì vậy, một giải pháp được đề ra, đó là kết nối một mạng các máy tính con lại với nhau để tạo ra một khối có sức mạnh tính toán của một siêu máy tính.

Dựa theo ý tưởng trên, có rất nhiều cách và cũng như framework đã được tạo nên để hiện thực hóa ý tưởng đó. Tuy nhiên, cụ thể ta sẽ chỉ đề cập và phân tích Hadoop trong đề tài này.

Việc xử lý dữ liệu hầu hết được diễn ra theo các thuật toán, và như một lẽ đương nhiên, dữ liệu càng lớn, việc tính toán càng lâu và tốn thời gian. K-Means cũng không ngoại lệ. K-Means là một thuật toán phân loại dữ liệu rất phổ biến hiện nay, đây là một thuật toán đơn giản, dễ hiểu, và ta muốn nâng cao hiệu quả tính toán của thuật toán này bằng cách áp dụng khả năng song song hóa và xử lý phân tán của công nghệ Hadoop. Đây cũng chính là mấu chốt của bản báo cáo này.

2. Phạm vi đề tài

- Đề tài tập trung vào việc giới thiệu và phân tích về công nghệ Hadoop cũng như quá trình cài đặt một Hadoop cluster gồm 3 node(1 master và 2 slave/worker đối với quá trình minh họa việc cài đặt).
- Phân tích và thiết kế giải thuật K-Means theo hướng song song hóa.

II. HADOOP

1. Hadoop là gì?

Là một framework mã nguồn mở cho phép lưu trữ và xử lý dữ liệu lớn trong một môi trường phân tán trên các cụm máy tính bằng các mô hình lập trình đơn giản. Nó được thiết kế để mở rộng quy mô từ các máy chủ đơn lẻ đến hàng ngàn máy, mỗi máy cung cấp tính toán và lưu trữ cục bộ.

2. Kiến trúc Hadoop

Hadoop có hai lớp chính là:

- Lớp xử lý / tính toán (MapReduce) và
- Lớp lưu trữ (Hệ thống tệp phân tán hdfs của Hadoop).

3. MapReduce

MapReduce là mô hình lập trình song song để viết các ứng dụng phân tán được phát minh tại Google để xử lý hiệu quả một lượng lớn dữ liệu (bộ dữ liệu nhiều terabyte), trên các cụm lớn (hàng nghìn nút) phân cứng hàng hóa một cách đáng tin cậy, có khả năng chịu lỗi. Chương trình MapReduce chạy trên Hadoop, một khung công tác nguồn mở Apache.

Ý tưởng:

Một mô hình MapReduce gồm 3 giai đoạn chính:

- Map: mỗi worker sẽ áp dụng kỹ thuật phân chia dữ liệu của mình và sau đó trả về giá trị là các cặp (key, values). Máy chủ/master sẽ đảm bảo rằng, không có các cặp (key, values) nào bị trùng lặp.
- Shuffle: Sắp xếp lại trật tự của dữ liệu đầu ra trong quá trình Map và phân tán dữ liệu đi các node khác trong cụm.
- Reduce: Quá trình này sẽ tổng hợp lại kết quả thu được từ 2 quá trình trên dựa trên các key và values tương ứng, quá trình này sẽ được thực hiện song song trên nhiều node khác nhau và cho ra kết quả cuối cùng.

4. Hệ thống tệp phân tán Hadoop

Hệ thống tệp phân tán Hadoop (HDFS) dựa trên Hệ thống tệp của Google (GFS) và cung cấp một hệ thống tệp phân tán được thiết kế để chạy trên phần cứng hàng hóa. Nó có nhiều điểm tương đồng với các hệ thống tệp phân tán hiện có. Tuy nhiên, sự khác biệt từ các hệ thống tệp phân tán khác là đáng kể. Nó có khả năng chịu lỗi cao và được thiết kế để triển khai trên phần cứng giá rẻ.

Nó cung cấp quyền truy cập thông lượng cao vào dữ liệu ứng dụng và phù hợp với các ứng dụng có bộ dữ liệu lớn.

Ngoài hai thành phần cốt lõi nêu trên, khung Hadoop còn bao gồm hai mô-đun sau

- **Hadoop Common** - Đây là các thư viện và tiện ích Java được yêu cầu bởi các mô-đun Hadoop khác.
- **Hadoop YARN** - Đây là dịch vụ lập kế hoạch công việc và quản lý tài nguyên cụm.

5. Hadoop hoạt động như thế nào?

Nó khá tốn kém để xây dựng các máy chủ lớn hơn với cấu hình nặng xử lý quy mô lớn, nhưng thay vào đó, bạn có thể kết hợp nhiều máy tính hàng hóa với CPU đơn, như một hệ thống phân tán chức năng duy nhất và thực tế, các máy phân cụm có thể đọc được tập dữ liệu song song và cung cấp thông lượng cao hơn nhiều. Hơn nữa, nó rẻ hơn một máy chủ cao cấp. Vì vậy, đây là yếu tố thúc đẩy đầu tiên đằng sau việc sử dụng Hadoop mà nó chạy trên các máy phân cụm và chi phí thấp.

Hadoop chạy mã trên một cụm máy tính. Quá trình này bao gồm các nhiệm vụ cốt lõi sau mà Hadoop thực hiện :

- Dữ liệu ban đầu được chia thành các thư mục và tập tin. Các tệp được chia thành các khối có kích thước đồng nhất là 128M(cài đặt mặc định của Hadoop phiên bản 3 trở lên).
- Các tệp này sau đó được phân phối trên các nút cụm khác nhau để xử lý thêm.
- HDFS, nằm trên cùng của hệ thống tệp cục bộ, giám sát quá trình xử lý.
- Các khối được nhân rộng để xử lý lỗi phần cứng.
- Kiểm tra xem mã đã được thực thi thành công.
- Thực hiện sắp xếp dữ liệu đầu ra đã được xử lý qua quá trình map.
- Gửi dữ liệu được sắp xếp đến một node nhất định trong cụm để được thực hiện tiếp quá trình combine và reduce để cho ra kết quả cuối cùng.
- Viết nhật ký gỡ lỗi cho từng công việc.

6. Ưu điểm

- Framework của Hadoop cho phép người dùng nhanh chóng viết và kiểm tra các hệ thống phân tán. Nó là hiệu quả, và nó tự động phân phối dữ liệu và làm việc trên các máy và lần lượt, sử dụng sự song song cơ bản của các lõi CPU.
- Hadoop không dựa vào phần cứng để cung cấp khả năng chịu lỗi và tính sẵn sàng cao (FTHA), thay vào đó, thư viện Hadoop đã được thiết kế để phát hiện và xử lý các lỗi ở lớp ứng dụng. Khả năng chịu lỗi cao.
- Máy chủ có thể được thêm hoặc xóa khỏi cụm một cách linh hoạt và Hadoop tiếp tục hoạt động mà không bị gián đoạn.
- Một ưu điểm lớn khác của Hadoop là ngoài việc là nguồn mở, nó tương thích trên tất cả các nền tảng vì nó dựa trên Java.
- Hadoop được thiết kế để có thể xử lý mọi loại dữ liệu hiện hành(MySQL Data, JSON, XML,...)
- Tốc độ xử lý cao do có thể tính toán phân tán trên nhiều máy cùng một lúc
- Thông lượng cao
- Không gây sức ép lớn đối với đường truyền mạng của hệ thống

7. Nhược điểm:

- Quá trình cài đặt phức tạp, đòi hỏi người dùng phải có kiến thức rõ ràng về máy tính
- Xử lý trên các file dữ liệu nhỏ không hiệu quả
- Không bảo mật trong quá trình thực thi do các thư viện java và java là một trong những ngôn ngữ được sử dụng phổ biến nhất
- Phụ thuộc nhiều vào tốc độ đường truyền mạng của cả cụm
- Khả năng đọc/viết dữ liệu của Hadoop cũng bị hạn chế nhiều bởi phần cứng của các node

III. Cài đặt Hadoop

1. Yêu cầu về phần mềm trước khi cài đặt

- Thiết bị chạy hệ điều hành ubuntu 18.04 trở lên
- Đã cài sẵn các giao thức, dịch vụ cần thiết như: ssh, openssh-server, pdsh, jdk8

2. Tiến hành cài đặt¹

Lưu ý: việc cài đặt sẽ được diễn ra chủ yếu bằng lệnh và trên terminal của ubuntu

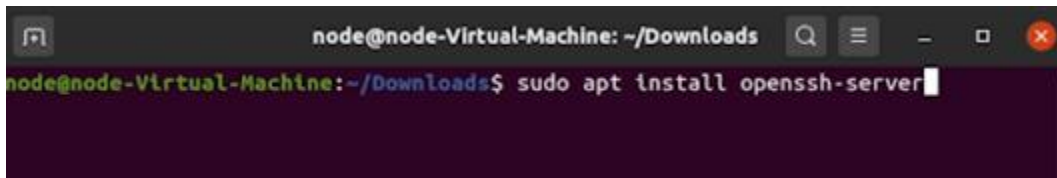
- Cài đặt ssh:

Giao thức ssh đã được cài đặt sẵn trong hệ điều hành ubuntu 18.04 trở lên nên ta không cần thực hiện lại việc cài đặt cho giao thức này.

- Cài đặt openssh-server:

Mở terminal trong ubuntu lên và chạy dòng lệnh:

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



Cài đặt openssh-server

- Cài đặt pdsh:

```
sudo apt install pdsh
```

```
node@node-Virtual-Machine:~/Downloads$ sudo apt install pdsh
```

Cài đặt pdsh

- Chỉnh lại giao thức mặc định trong pdsh thành ssh:

Trở về directory của Home bằng lệnh:

```
cd ~
```

```
node@node-Virtual-Machine:/$ cd ~
```

Mở để chỉnh .bashrc:

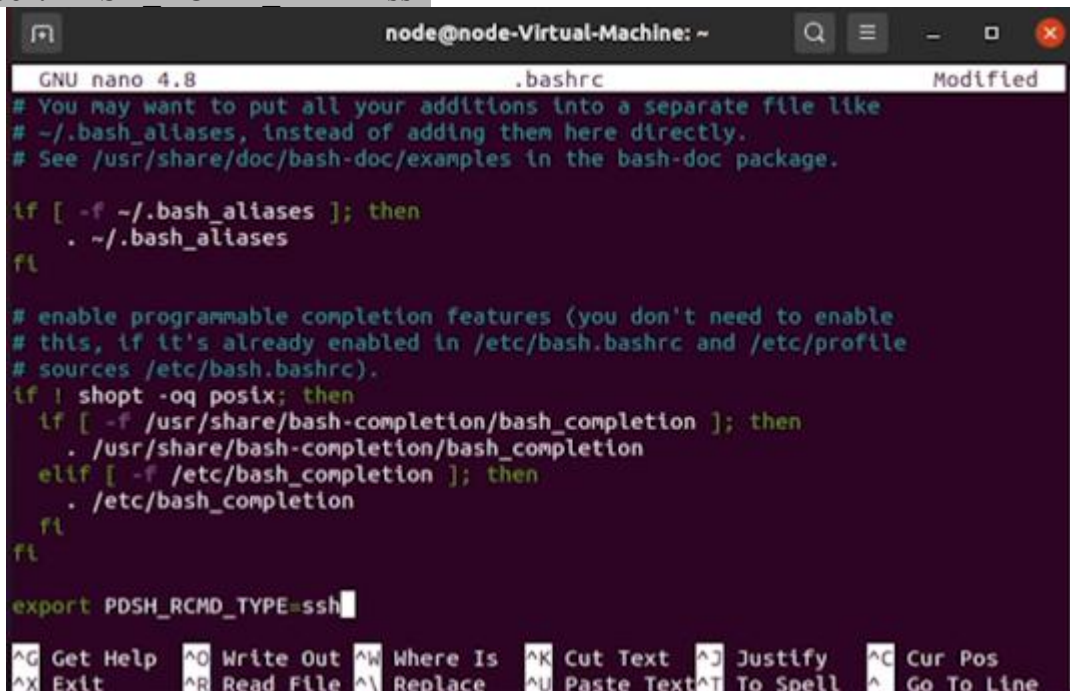
```
sudo nano .bashrc
```

```
node@node-Virtual-Machine:~$ sudo nano .bashrc
```

¹ Link hướng demo quá trình cài đặt: <https://youtu.be/S8FaBW30prg>

Lưu ý: đây là video ở chế độ không công khai, chỉ có những ai có đường link này mới có thể truy cập vào video, và video thuộc quyền sở hữu của chính tác giả(Trần Hữu Lợi) của bản báo cáo này.

Đi xuống cuối file bashrc và chèn thêm dòng sau rồi lưu thay đổi:
`export PDSH_RCMD_TYPE=ssh`



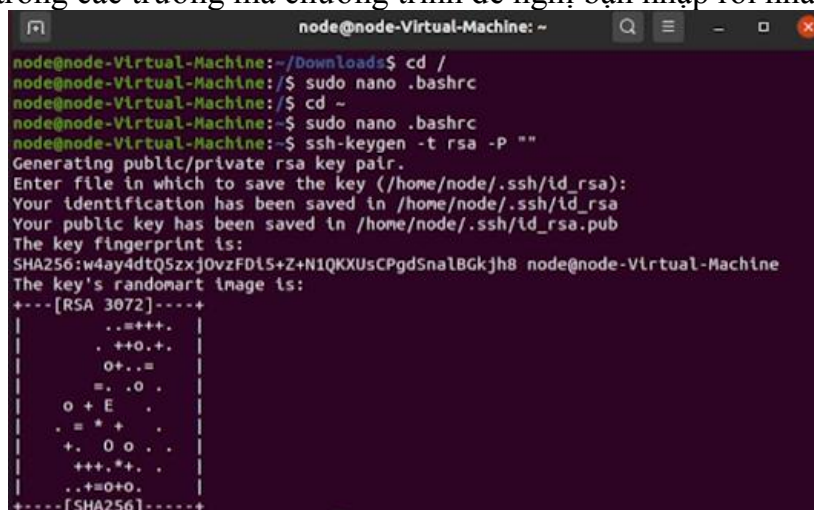
```
node@node-Virtual-Machine: ~  
GNU nano 4.8 .bashrc Modified  
# You may want to put all your additions into a separate file like  
# ~/.bash_aliases, instead of adding them here directly.  
# See /usr/share/doc/bash-doc/examples in the bash-doc package.  
  
if [ -f ~/.bash_aliases ]; then  
    . ~/.bash_aliases  
fi  
  
# enable programmable completion features (you don't need to enable  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
if ! shopt -oq posix; then  
    if [ -f /usr/share/bash-completion/bash_completion ]; then  
        . /usr/share/bash-completion/bash_completion  
    elif [ -f /etc/bash_completion ]; then  
        . /etc/bash_completion  
    fi  
fi  
  
export PDSH_RCMD_TYPE=ssh  
  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Chỉnh lại .bashrc

- Tạo key cho ssh:

`ssh-keygen -t rsa -P ""`

Ta tiếp tục bỏ trống các trường mà chương trình đề nghị bạn nhập rồi nhấn enter.



```
node@node-Virtual-Machine: ~/Downloads$ cd /  
node@node-Virtual-Machine:/$ sudo nano .bashrc  
node@node-Virtual-Machine:/$ cd ~  
node@node-Virtual-Machine:/$ sudo nano .bashrc  
node@node-Virtual-Machine:~$ ssh-keygen -t rsa -P ""  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/node/.ssh/id_rsa):  
Your identification has been saved in /home/node/.ssh/id_rsa  
Your public key has been saved in /home/node/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:w4ay4dtQ5zxj0vzFDl5+Z+NlQKXUsCPgdSna1BGkjH8 node@node-Virtual-Machine  
The key's randomart image is:  
+---[RSA 3072]-----+  
| ..=+++ |  
| .++0.+ |  
| O+..= |  
| =.O. |  
| o+E. |  
| .=*+. |  
| +.Oo.. |  
| +++*+. |  
| ..+=0+0. |  
+---[SHA256]-----+
```

Tạo key cho SSH

- Lưu lại key ssh vừa tạo:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

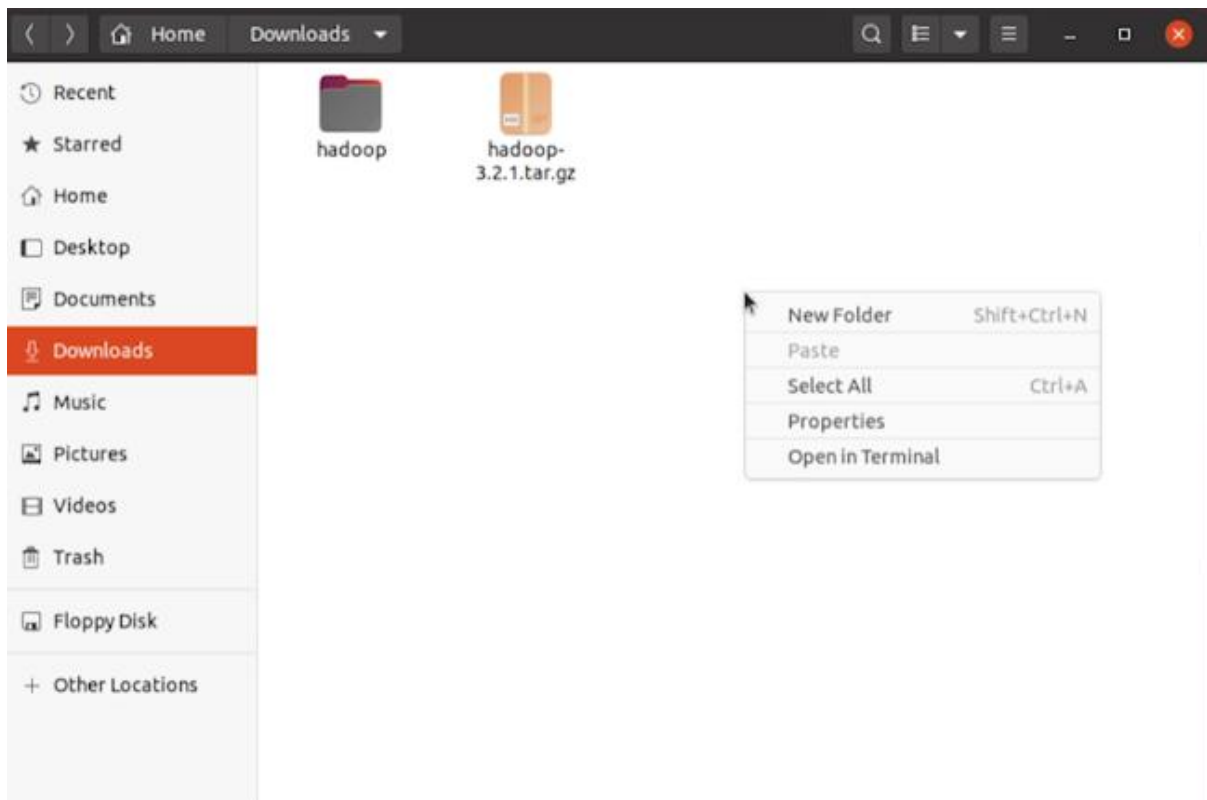
```
node@node-Virtual-Machine:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- Cài đặt JDK 8 của Java:

```
sudo apt install openjdk-8-jdk
```

```
node@node-Virtual-Machine:~$ sudo apt install openjdk-8-jdk
```

- Tải hadoop phiên bản 3.2.1 về từ <https://mirrors.sonic.net/apache/hadoop/common/> và giải nén.
- Tại thư mục chứa folder hadoop vừa giải nén, ta nhấn chuột phải rồi chọn "Open in Terminal".



Mở terminal tại thư mục lưu hadoop vừa tải về

- Ta mở file hadoop-env.sh:

```
nano hadoop/etc/hadoop/hadoop-env.sh
```

```
node@node-Virtual-Machine:~/Downloads$ nano hadoop/etc/hadoop/hadoop-env.sh
```

- Tìm đến dòng "export JAVA_HOME=" và gán giá trị của đường dẫn thư viện jdk8 ta vừa tải về vào sau dấu bằng.

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

```
# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
```

Chỉnh sửa hadoop-env.sh

- Mở file environment và chỉnh sửa như sau:

```
sudo nano /etc/environment
```

```
node@node-Virtual-Machine:~/Downloads$ sudo nano /etc/environment
```

Trong biến PATH, trước khi kết thúc dấu nhảy đôi, ta thêm các đường dẫn sau:

```
:/usr/local/hadoop/bin:/usr/local/hadoop/sbin
```

```
GNU nano 4.8 /etc/environment Modified
games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin
```

Thêm đường dẫn vào environment

Thoát khỏi dấu nhảy, ta xuống dòng và tiếp tục thêm biến sau vào:

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

```
node@node-Virtual-Machine: ~/Downloads
GNU nano 4.8 /etc/environment Modified
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin"
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre"
```

Thêm đường dẫn vào environment

Lưu và thoát.

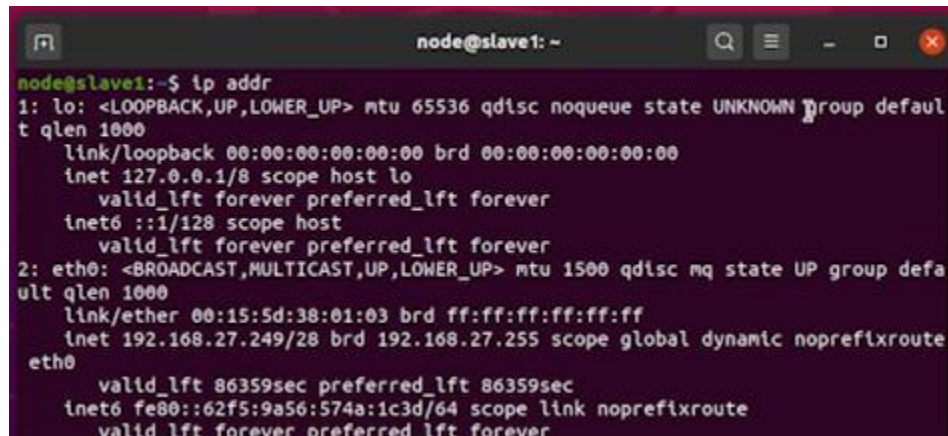
Ta dời thư mục hadoop về đường dẫn mới:

```
sudo mv hadoop /usr/local/hadoop
```

```
node@node-Virtual-Machine:~/Downloads$ sudo mv hadoop /usr/local/hadoop
```

- Lặp lại các bước cài đặt trên cho tất cả các node trong cluster, cụ thể ở đây, ta sẽ có một cluster gồm 3 node(1 master, 2 slave)
- **Tại mỗi node**, ta chỉnh lại file hosts, điền thêm vào địa chỉ IP của tất cả các máy trong cluster cũng như tên của node đó, sau đó ta thay đổi hostname của từng node thành với tên tương ứng ta đã đặt trong hosts:

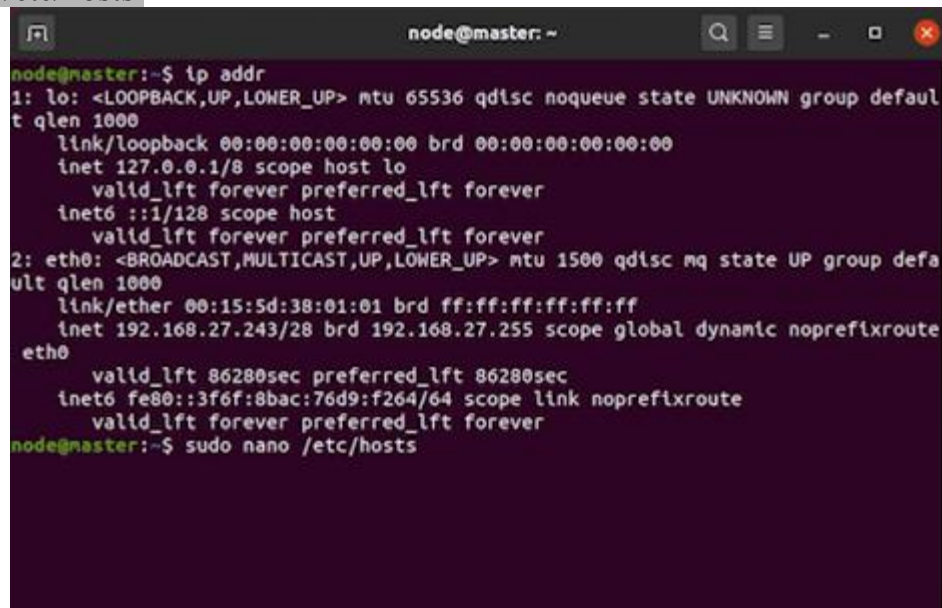
Kiểm tra địa chỉ IP của mỗi node:
ip addr



```
node@slave1: ~  
node@slave1:~$ ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default  
    link/ether 00:15:5d:38:01:03 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.27.249/28 brd 192.168.27.255 scope global dynamic noprefixroute eth0  
        valid_lft 86359sec preferred_lft 86359sec  
    inet6 fe80::62f5:9a56:574a:1c3d/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever
```

Kết quả của lệnh ip addr

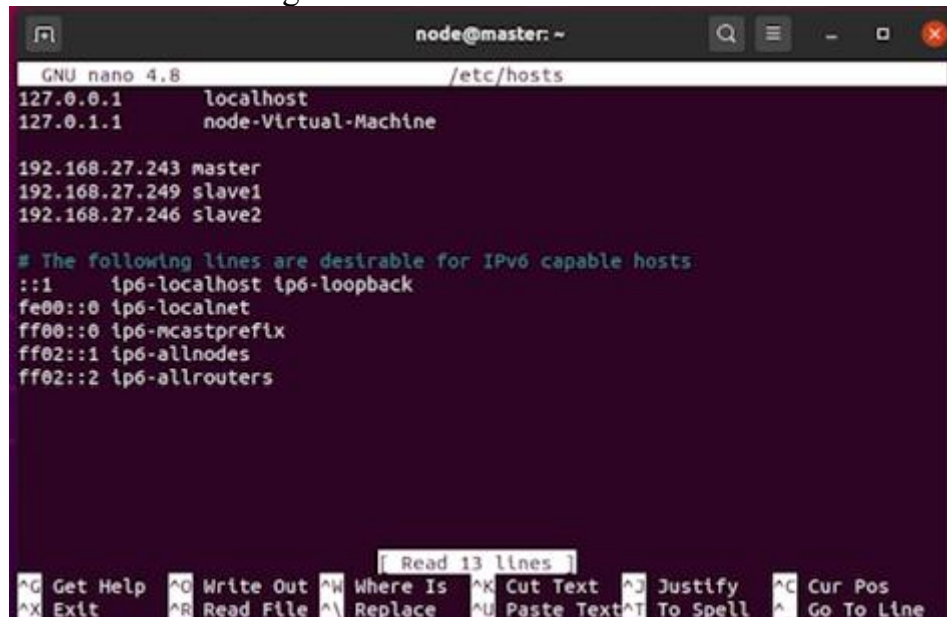
Mở file hosts:
sudo nano /etc/hosts



```
node@master: ~  
node@master:~$ ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default  
    link/ether 00:15:5d:38:01:01 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.27.243/28 brd 192.168.27.255 scope global dynamic noprefixroute eth0  
        valid_lft 86280sec preferred_lft 86280sec  
    inet6 fe80::3f6f:8bac:76d9:f264/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
node@master:~$ sudo nano /etc/hosts
```

Gõ lệnh vào terminal

Thêm danh sách các node trong cluster vào hosts:



```
node@master: ~  
GNU nano 4.8 /etc/hosts  
127.0.0.1 localhost  
127.0.1.1 node-Virtual-Machine  
  
192.168.27.243 master  
192.168.27.249 slave1  
192.168.27.246 slave2  
  
# The following lines are desirable for IPv6 capable hosts  
::1 ip6-localhost ip6-loopback  
fe00::0 ip6-localnet  
ff00::0 ip6-mcastprefix  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
  
Read 13 lines  
^G Get Help ^O Write Out ^W Where Is ^X Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^I Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Thêm địa chỉ IP của tất cả các node trong cụm Hadoop và hostname tương ứng

Mở file hostname:

```
sudo nano /etc/hostname
```

Tại mỗi node với các IP tương ứng thì ta đặt hostname tương ứng rồi lưu lại.

Ví dụ: tại node mà ta chọn làm master



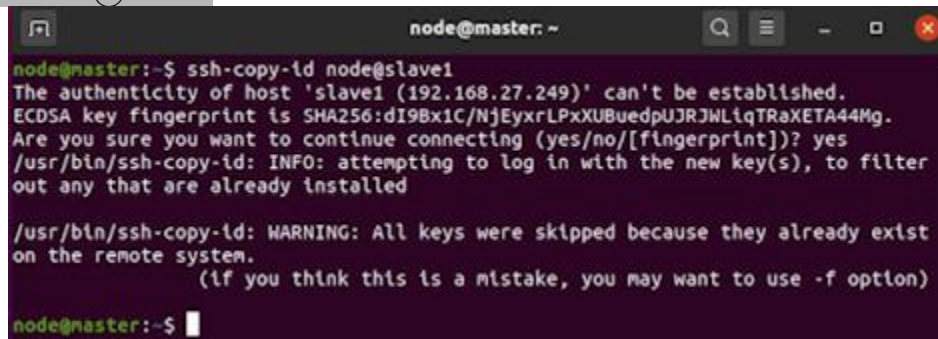
```
GNU nano 4.8 /etc/hostname Modified  
master
```

Chỉnh lại hostname của máy

Sau khi chỉnh lại hosts và hostname, ta có cluster như sau:

1. master
2. slave1
3. slave2

Tại node master, ta dùng lệnh để copy các ssh key của mình sang các node slave:
`ssh-copy-id node@slave1`



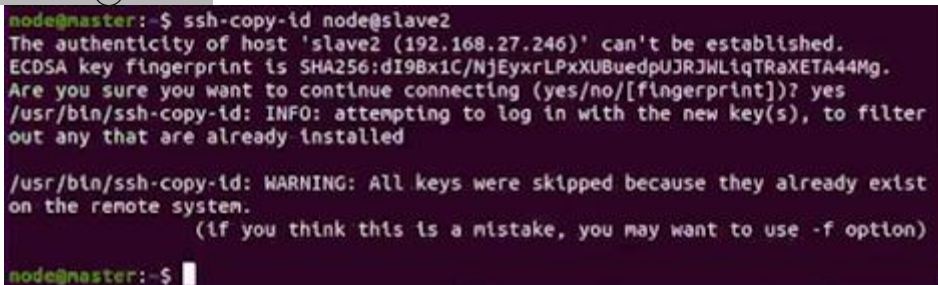
```
node@master:~$ ssh-copy-id node@slave1
The authenticity of host 'slave1 (192.168.27.249)' can't be established.
ECDSA key fingerprint is SHA256:dI9Bx1C/NjEyxrLPxXUBuedpUJRJWLiQTRaXETA44Mg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist
on the remote system.
(if you think this is a mistake, you may want to use -f option)

node@master:~$
```

Copy key của SSH vào slave1

`ssh-copy-id node@slave2`



```
node@master:~$ ssh-copy-id node@slave2
The authenticity of host 'slave2 (192.168.27.246)' can't be established.
ECDSA key fingerprint is SHA256:dI9Bx1C/NjEyxrLPxXUBuedpUJRJWLiQTRaXETA44Mg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist
on the remote system.
(if you think this is a mistake, you may want to use -f option)

node@master:~$
```

Copy key của SSH vào slave2

Để chắc chắn ta cũng có thể thực hiện việc tương tự đối với node master
`ssh-copy-id node@master`

- Vẫn tại node master, ta đi đến thư mục `/usr/local` rồi mở bằng Terminal tại thư mục đó

Mở file `core-site.xml` của hadoop

`sudo nano hadoop/etc/hadoop/core-site.xml`

với nội dung như sau:

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




```
node@master: /usr/local
GNU nano 4.8      hadoop/etc/hadoop/core-site.xml      Modified
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

Chỉnh sửa core-site.xml

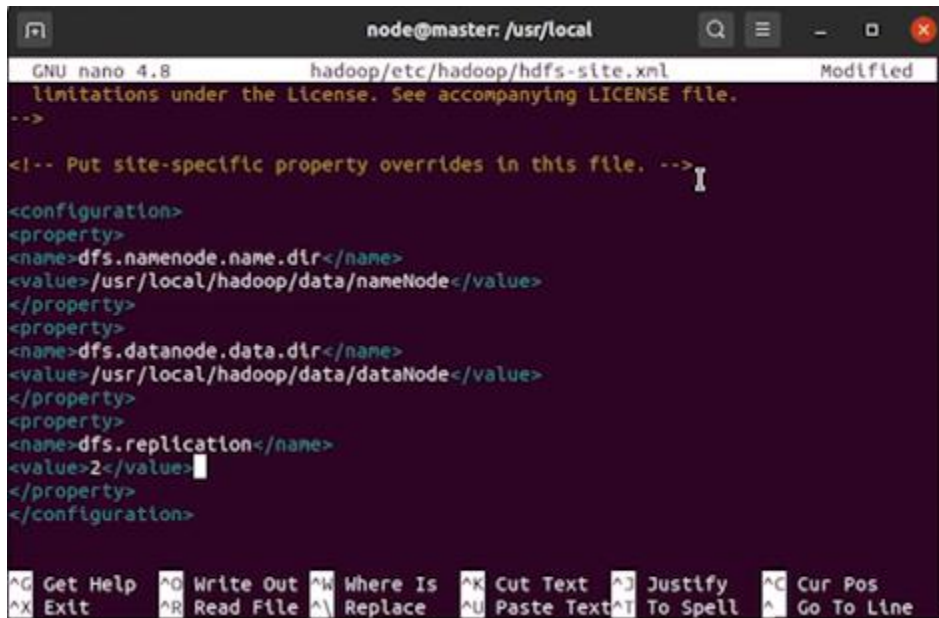
Lưu lại và thoát ra.

Mở file hdfs-site.xml:

```
sudo nano hadoop/etc/hadoop/hdfs-site.xml
```

với nội dung sau:

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>/usr/local/hadoop/data/nameNode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/usr/local/hadoop/data/dataNode</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

A terminal window showing the nano 4.8 text editor editing the file /usr/local/hadoop/etc/hadoop/hdfs-site.xml. The file content includes XML configuration for HDFS, with properties for name node and data node directories, and replication factor. The cursor is positioned at the end of the line containing the replication factor value '2'.

```
node@master: /usr/local
GNU nano 4.8 hadoop/etc/hadoop/hdfs-site.xml Modified
#configuration
#hadoop.tmp.dir: set to the local temporary directory on each node.
# by default hadoop.tmp.dir is the local temporary directory on
# the master, and the local temporary directory on each slave.
# It can be set to the local temporary directory on each node
# by adding the following lines to the configuration file:
#
# hadoop.tmp.dir: /usr/local/hadoop/tmp
#
# Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>/usr/local/hadoop/data/nameNode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/usr/local/hadoop/data/dataNode</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Chỉnh hdfs-site.xml

Lưu và thoát.

Mở file workers và chỉnh:

`sudo nano hadoop/etc/hadoop/workers`

với nội dung là tên của các node slave trên các dòng

A terminal window showing the nano 4.8 text editor editing the file /usr/local/hadoop/etc/hadoop/workers. The file content is empty, and the cursor is positioned at the end of the line containing the text 'slave2'.

```
node@master: /usr/local
GNU nano 4.8 hadoop/etc/hadoop/workers Modified
slave1
slave2
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Thêm các node đóng vai trò là slave vào workers

Lưu và thoát.

- Copy những thiết lập ta vừa chỉnh sửa đến các node slave:

```
scp /usr/local/hadoop/etc/hadoop/* slave1:/usr/local/hadoop/etc/hadoop/
node@master:/usr/local$ scp hadoop/etc/hadoop/* slave1:/usr/local/hadoop/etc/hadoop/
```

```
scp /usr/local/hadoop/etc/hadoop/* slave2:/usr/local/hadoop/etc/hadoop/
node@master:/usr/local$ scp hadoop/etc/hadoop/* slave2:/usr/local/hadoop/etc/hadoop/
```

- Chỉnh lại source để lấy đường dẫn các chương trình và chức năng của hadoop:

```
source /etc/environment
```

```
node@master:/usr/local$ source /etc/environment
```

- Đến bước này, về cơ bản chúng ta đã set up xong cho nơi lưu trữ ảo hdfs của cluster này, bây giờ ta cần phải format lại hdfs trước khi sử dụng:

```
hdfs namenode -format
```

```
node@master:/usr/local$ hdfs namenode -format
WARNING: /usr/local/hadoop/logs does not exist. Creating.
```

- Khởi chạy hdfs bằng lệnh:

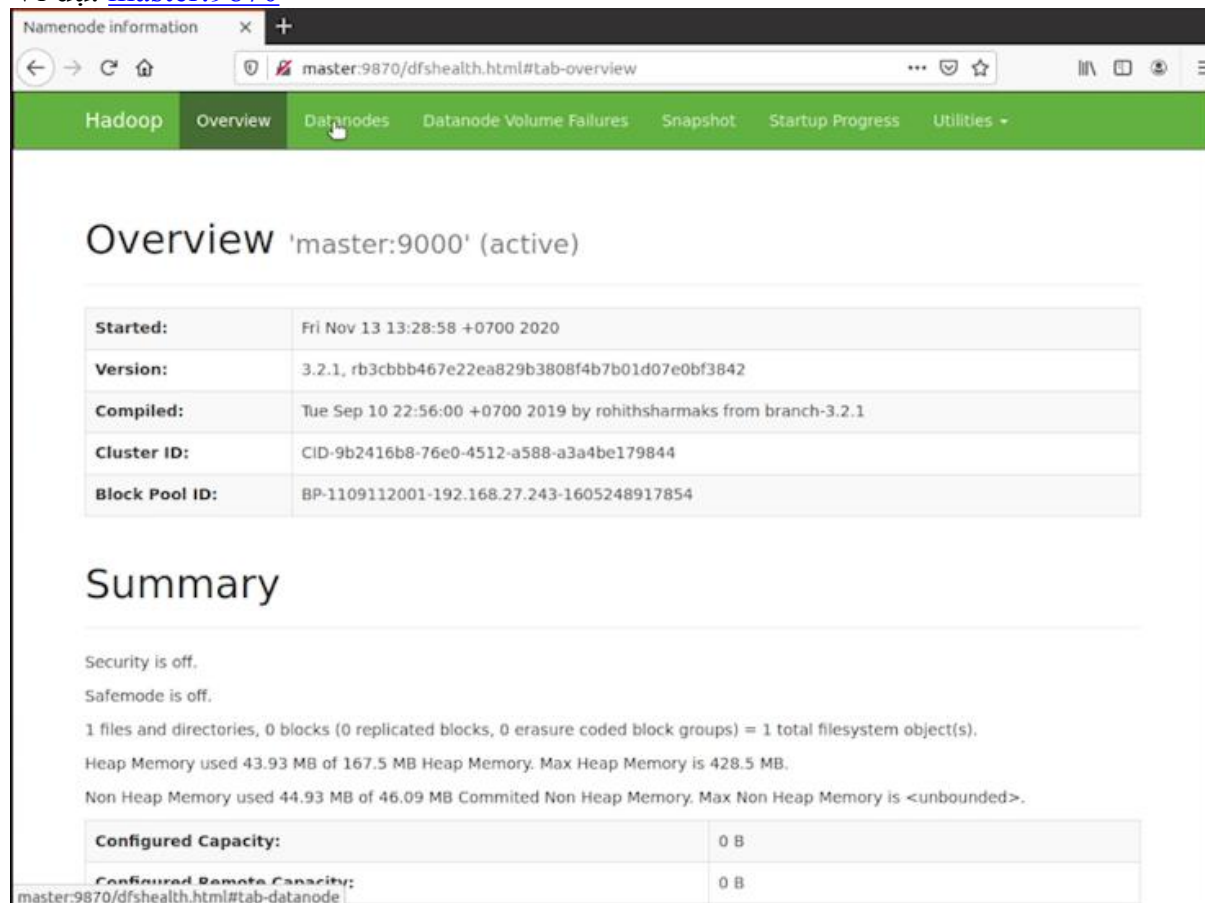
```
start-dfs.sh
```

```
node@master:/usr/local$ start-dfs.sh
Starting namenodes on [master]
Starting datanodes
slave1: WARNING: /usr/local/hadoop/logs does not exist. Creating.
slave2: WARNING: /usr/local/hadoop/logs does not exist. Creating.
Starting secondary namenodes [master]
node@master:/usr/local$
```

Khởi chạy hdfs

Để kiểm tra thông tin các node, ta vào trang web được host tại <tên của node master>:9870

Ví dụ: master:9870



Started:	Fri Nov 13 13:28:58 +0700 2020
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 22:56:00 +0700 2019 by rohitsharmaks from branch-3.2.1
Cluster ID:	CID-9b2416b8-76e0-4512-a588-a3a4be179844
Block Pool ID:	BP-1109112001-192.168.27.243-1605248917854

Summary

Security is off.
Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 43.93 MB of 167.5 MB Heap Memory. Max Heap Memory is 428.5 MB.

Non Heap Memory used 44.93 MB of 46.09 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	0 B
Configured Ramdisk Capacity:	0 B

Kiểm tra trên giao diện web của hadoop

- Dừng hdfs bằng lệnh:

`stop-dfs.sh`

```
node@master:/usr/local$ stop-dfs.sh
Stopping namenodes on [master]
Stopping datanodes
Stopping secondary namenodes [master]
```

Dừng hdfs

- Vẫn tại node master, ta thực hiện những bước cuối cùng của quá trình cài đặt

```
export HADOOP_HOME="/usr/local/hadoop"
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
export HADOOP_YARN_HOME=$HADOOP_HOME
```

```
node@master:/usr/local$ export HADOOP_HOME="/usr/local/hadoop"
node@master:/usr/local$ export HADOOP_COMMON_HOME=@HADOOP_HOME
node@master:/usr/local$ export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
node@master:/usr/local$ export HADOOP_HDFS_HOME=@HADOOP_HOME
node@master:/usr/local$ export HADOOP_MAPRED_HOME=$HADOOP_HOME
node@master:/usr/local$ export HADOOP_YARN_HOME=$HADOOP_HOME
node@master:/usr/local$ export HADOOP_COMMON_HOME=$HADOOP_HOME
node@master:/usr/local$ export HADOOP_HDFS_HOME=$HADOOP_HOME
```

Cài đặt các biến môi trường cuối cùng cho hadoop

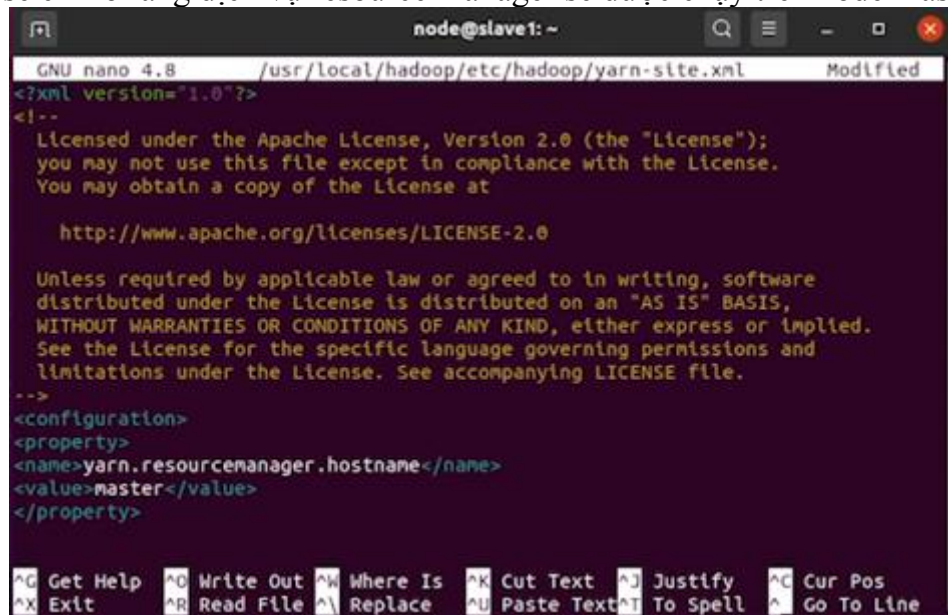
- Lần lượt đi đến các slave, ta chỉnh lại file yarn-site.xml:

```
sudo nano /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

với nội dung như sau:

```
<property>
<name>yarn.resourcemanager.hostname</name>
<value>hadoop-master</value>
</property>
```

Bước này sẽ chỉ rõ rằng dịch vụ resource manager sẽ được chạy trên node master



```
node@slave1: ~
GNU nano 4.8 /usr/local/hadoop/etc/hadoop/yarn-site.xml Modified
<?xml version="1.0"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<configuration>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>master</value>
</property>
```

Chỉnh lại thiết lập của yarn trên các slave

Quay lại node master, ta bây giờ có thể khởi chạy dịch vụ Resource manager bằng lệnh:

start-yarn.sh

```
node@master: /usr/local
node@master:/usr/local$ start-dfs.sh
Starting namenodes on [master]
Starting datanodes
Starting secondary namenodes [master]
node@master:/usr/local$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
node@master:/usr/local$
```

Khởi động hdfs và resource manager

Ta có thể xem thông tin của Resource manager qua giao diện web nằm trên port 8088, ví dụ: master:8088

Dừng Resource manager:

stop-yarn.sh

```
node@master:/usr/local$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
node@master:/usr/local$ stop-dfs.sh
Stopping namenodes on [master]
Stopping datanodes
Stopping secondary namenodes [master]
node@master:/usr/local$
```

Dừng resource manager và hdfs

Trong lúc hdfs và resource manager được chạy, ta có thể kiểm tra xem các node con có chạy đúng tiến trình hay không bằng cách gọi lệnh `jps`

```
node@slave1:~$ jps
3714 NodeManager
3555 DataNode
3828 Jps
```

Sử dụng jps

3. Một số lệnh thường dùng trong Hadoop

Lệnh dùng để thực thi một mapreduce job với bất kỳ file nào có thể thực thi và được hỗ trợ về ngôn ngữ lập trình(Đây cũng là lệnh được dùng trong thực nghiệm để chạy chương trình):

```
mapred streaming \  
-input myInputFile.txt \  
-output myOutputFile.txt \  
-mapper myMapper.py \  
-reducer myReducer.py
```

Lệnh copy file vào hdfs từ bộ nhớ local trên thiết bị:

```
hadoop dfs -copyFromLocal ~/MyFolder/myFile.txt /myFileInHDFS.txt
```

lệnh để xuất kết quả:

```
hadoop dfs -cat /myOutput.txt
```

Tạo đường dẫn trong hdfs:

```
hadoop dfs -mkdir /my/path
```

Xóa đường dẫn trong hdfs:

```
hadoop dfs -rm -r /my/path
```

Liệt kê ra các item trong đường dẫn:

```
hadoop dfs -ls /my/path
```

Các file dữ liệu sẽ được chia nhỏ thành các block khác nhau để lưu trữ trên hdfs. Để check vị trí và tình trạng của các block ta sử dụng lệnh:

```
hdfs fsck /myFile.txt -files -blocks -locations
```

IV. Thuật toán K-Means

1. Lý thuyết cơ bản

Phân cụm k-means là 1 phương pháp lượng tử hóa vector dùng để phân các điểm dữ liệu cho trước vào các cụm khác nhau. Phân cụm k-means có nhiều ứng dụng, nhưng được sử dụng nhiều nhất trong Trí tuệ nhân tạo và Học máy (cụ thể là Học không có giám sát).

Thuật toán 1

1. *Tạo ngẫu nhiên k tâm cụm.*
2. *Gán các điểm dữ liệu vào các tâm cụm mà có khoảng cách gần với chúng nhất.*
3. *Tính toán lại tâm cụm bằng cách tính giá trị trung bình của các điểm dữ liệu đối với các tâm cụm tương ứng.*
4. *Lặp lại bước 2, 3 cho đến khi tâm cụm không đổi thì dừng.*

Để tổng quát lại, ta có thể nói như sau: Thuật toán k-means sử dụng phương pháp tạo và cập nhật trung tâm để phân nhóm các điểm dữ liệu cho trước vào các nhóm khác nhau. Đầu tiên chúng sẽ tạo ra các điểm trung tâm ngẫu nhiên. Sau đó gán mỗi điểm trong tập dữ liệu vào trung tâm gần nó nhất. Sau đó chúng sẽ cập nhật lại trung tâm và tiếp tục lặp lại các bước đã kể trên. Điều kiện dừng của thuật toán: Khi các trung tâm không thay đổi trong 2 vòng lặp kế tiếp nhau. Tuy nhiên, việc đạt được 1 kết quả hoàn hảo là rất khó và rất tốn thời gian, vậy nên thường người ta sẽ cho dừng thuật toán khi đạt được 1 kết quả gần đúng và chấp nhận được.

2. Áp dụng mô hình MapReduce vào thuật toán K-Means

Như đã đề cập ở trên, việc tính toán của thuật toán K-Means là rất tốn thời gian mà cụ thể hơn, chính là bước 2 và bước 3 của *Thuật toán 1*, điều này càng được thể hiện rõ khi số lượng dữ liệu đầu vào càng lớn.

Do đó ta có thể tận dụng khả năng tính toán phân tán để tối ưu hóa hiệu suất của bước 2 và bước 3 này.

Thuật toán 2

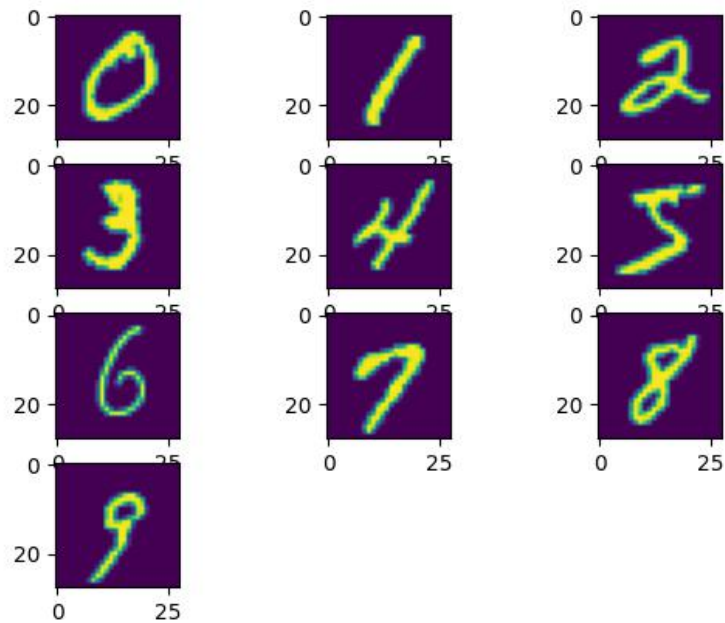
1. Tạo ngẫu nhiên k cụm.
 2. Thực hiện map các dữ liệu đầu vào, tạo ra output là cặp (key, values) với key là số hiệu của tâm cụm gần với dữ liệu đó nhất và values là các giá trị của dữ liệu đó.
 3. Thực hiện reduce các cặp (key, values) bằng cách tính trung bình các values lại theo từng key tương ứng, sau quá trình này ta sẽ ra được output là k cặp (key, values) tương ứng với k tâm cụm mới của thuật toán.
 4. Cập nhật lại dữ liệu của tâm cụm dựa theo kết quả vừa tính được từ bước 3.
 5. Lặp lại bước 2, 3, 4 cho đến khi tâm cụm không đổi.
- Bằng việc áp dụng mô hình MapReduce, ta có thể giảm đáng kể thời gian cần thiết để hoàn thành việc tính toán.

3. Thực nghiệm

Giả sử số lần lặp để tìm được tâm cụm cần thiết sau cùng là hữu hạn N (với N thuộc tập số tự nhiên lớn hơn 0), tổng số lượng phép tính cần tính toán ở bước 2 và 3 là M (với M thuộc tập số tự nhiên lớn hơn 0) thì ta có độ phức tạp của thuật toán K-Means tổng quát là $O(N*M)$.

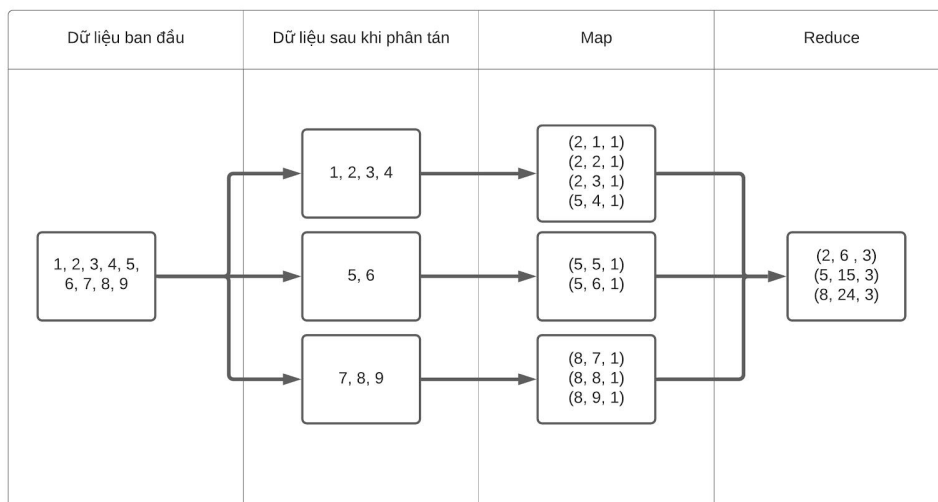
Để tối ưu tốc độ chạy thuật toán, ta chỉ cần giảm N hoặc M . Tuy nhiên, N là số lần lặp không biết trước, ta không thể tác động trực tiếp đến N . Ngược lại, M là số lần tính toán hữu hạn mà ta có thể biết rõ và dự đoán nên việc giảm M để giảm thiểu độ phức tạp tổng thể và một bước khả thi hơn. Ta giảm M bằng cách chia nhỏ M ra để xử lý phân tán cùng lúc trên nhiều node.

Thực nghiệm được tiến hành để đánh giá tốc độ tính toán của bước 2 và 3 của **Thuật toán 2** so với **Thuật toán 1** với dữ liệu đầu vào là tập dữ liệu ảnh chữ số viết tay được thu thập bởi MNIST với độ lớn của từng ảnh là 28×28 pixel, tương đương với 784 trường dữ liệu cho một vector ảnh. Thuật toán được chạy bằng cách tuần tự (bước 2, 3 của **Thuật toán 1**) và song song (bước 2, 3 của **Thuật toán 2**) và được cài đặt trên ngôn ngữ Python.

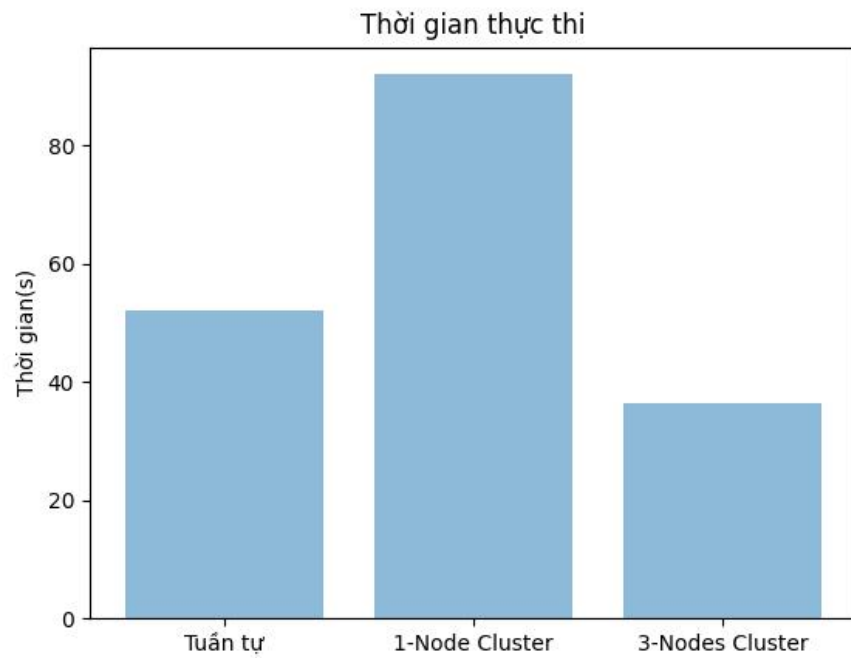


Hình minh họa bộ ảnh dữ liệu đầu vào của MNIST

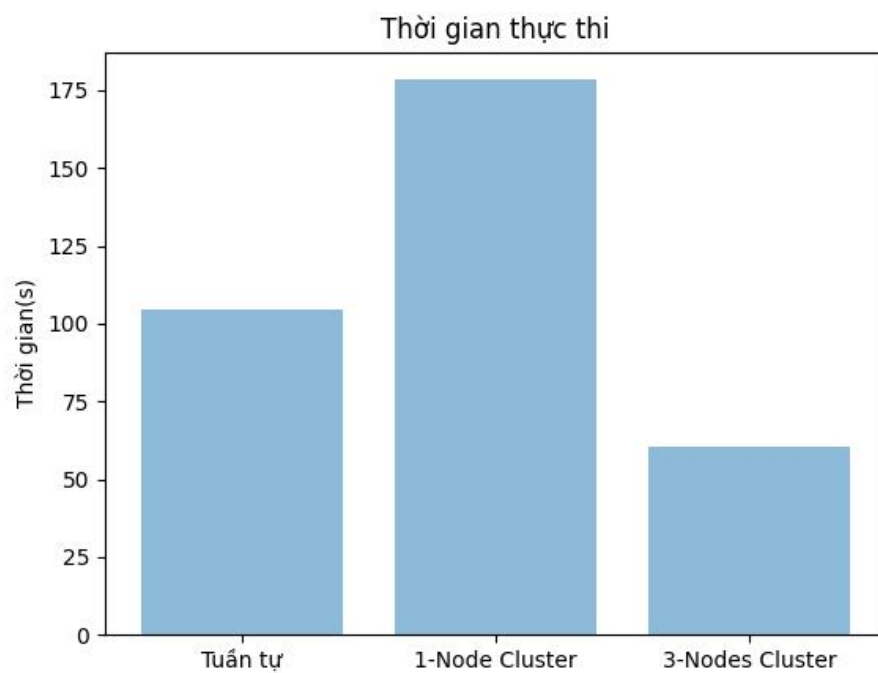
Dữ liệu đầu ra của thực nghiệm là gồm k dòng, đại diện cho k vector ảnh của tâm cụm với giá trị là tổng của các ảnh trong thuộc một tâm cụm gần nhất với chúng và số lượng ảnh thuộc tâm cụm đó.



Minh họa cho MapReduce của K-Means khi chọn $k=3$, với các giá trị khởi tạo là tương ứng của tâm cụm 1, 2, 3 là 2, 5, 8



Biểu đồ 1. Thời gian thực thi trên bộ dữ liệu với 30000 ảnh



Biểu đồ 2. Thời gian thực thi trên bộ dữ liệu 60000 ảnh

Qua 2 biểu đồ, ta nhận thấy rõ rằng đối với cluster chỉ có 1 node, tốc độ chạy chậm hơn hẳn so với tuần tự. Điều này là dễ hiểu bởi vì khi chạy MapReduce trên một node, điều này cũng có nghĩa là chạy MapReduce một cách tuần tự trên node ấy, ta không tận dụng được khả năng phân tán của mô hình MapReduce, mà ngược lại, ta lại tốn kém hơn vì có thêm chi phí truyền dữ liệu của mô hình MapReduce. Tuy nhiên, có một sự khác biệt rõ rệt về hiệu năng khi chạy thuật toán theo mô hình MapReduce trên cụm Hadoop 3 node. Thời gian thực thi của cụm Hadoop gồm 3 node thể hiện rõ ràng rằng, việc cho thêm node vào cụm đã giúp việc tính toán đồng thời trên 3 máy của cả hệ thống Hadoop về tổng thể chiến thắng được hao phí về mặt truyền thông của mô hình MapReduce và tận dụng được tối đa nguồn tài nguyên CPU có sẵn từ các node khác nhau.

V. Kết luận và định hướng phát triển

Việc xử lý một khối lượng lớn thông tin trong tin học đã trở thành một vấn đề cấp thiết. Các thuật toán nền tảng và cơ bản như K-Means giờ đã có thể được cải thiện hiệu năng một cách rõ rệt bằng cách áp dụng công nghệ phân tán của Hadoop. Điều này đã từ lâu được các nhà nghiên cứu, tổ chức đưa vào thực tiễn áp dụng. Áp dụng ưu điểm của Hadoop để tạo ra các siêu máy tính cho riêng mình. Từ đó, ta thấy được lợi ích của việc song song hóa là vô cùng to lớn. Song song hóa nói chung và Hadoop nói riêng đều đã giúp ta tận dụng được tối đa nguồn tài nguyên tính toán của mình.

Có thể thấy là thực nghiệm của đề án này chỉ dừng lại ở quá trình Map và Reduce của K-Means. Điều này là do thời lượng nghiên cứu cả đề tài này có giới hạn nên đề án xin được dừng lại tại đây. Tuy nhiên đối với những người có mong muốn thực hiện tiếp đề tài này, họ hoàn toàn có thể dựa vào bản báo cáo này *như một nguồn dữ liệu tham khảo* để hoàn thành thuật toán K-Means khi tính toán phân tán trên hệ thống của Hadoop một cách hoàn chỉnh và áp dụng và các vấn đề lớn hơn, xử lý đồng thời trên các bộ dữ liệu lớn hơn.

VI. Tài liệu tham khảo

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Fully-Distributed_Operation

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html>

http://it.husc.edu.vn/Media/ChuyenMuc/KhoaHoc/Hoithao-Hoinghi/SAICT_2015_submission_5.pdf

https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/bodoia.pdf

https://github.com/marko-mih/kmeans_mapreduce

HẾT