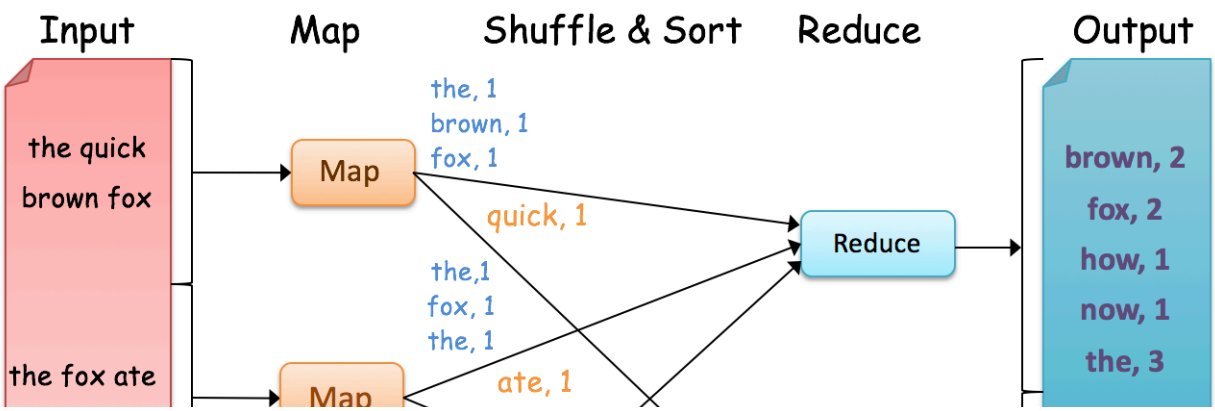
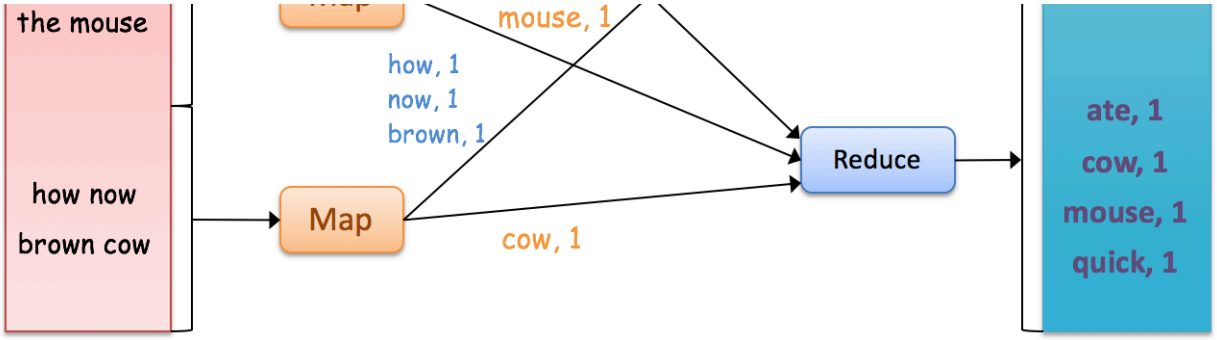
Rec Hadoop MapReduce

Wednesday, October 9, 2024 9:09 AM

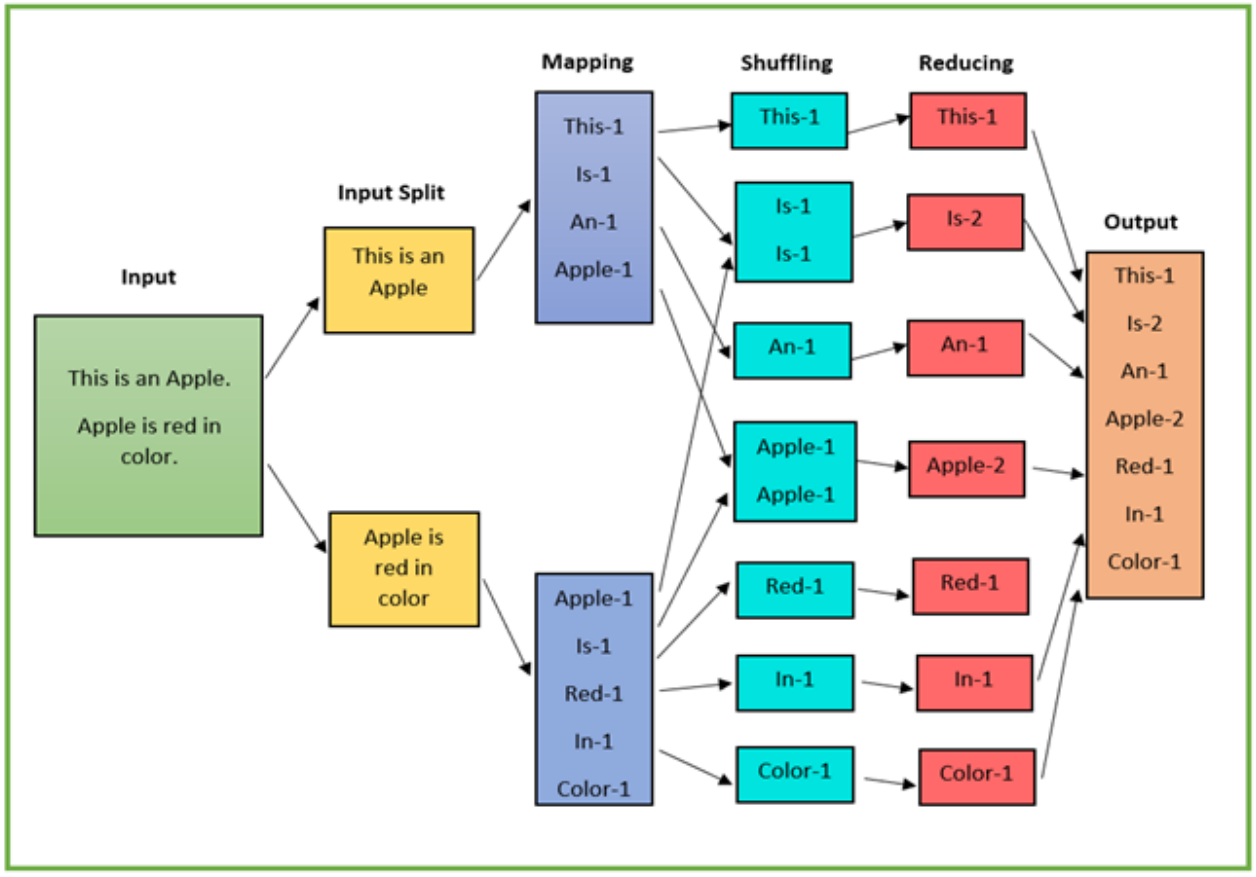
https://www.databricks.com/glossary/hadoop

- **HDFS** - Hadoop Distributed File System. HDFS is a Java-based system that allows large data sets to be stored across nodes in a cluster in a fault-tolerant manner.
- **YARN** - Yet Another Resource Negotiator. YARN is used for cluster resource management, planning tasks, and scheduling jobs that are running on Hadoop.
- **MapReduce** - MapReduce is both a programming model and big data processing engine used for the parallel processing of large data sets. Originally, MapReduce was the only execution engine available in Hadoop. But, later on Hadoop added support for others, including Apache Tez and Apache Spark.
- **Hadoop Common** - Hadoop Common provides a set of services across libraries and utilities to support the other Hadoop modules.





https://admicloud.github.io/www/_images/wordcount.png



https://miro.medium.com/v2/resize:fit:1314/1*OVea8gXNO3lP5-z4bqOhLA.png

Data
https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt

Step 1:
Prepare *mapper.py* and *reducer.py* function and data *corpus.txt*
Step 2:
Run in terminal the following command to test the code

```
cat corpus.txt | python mapper.py | sort | python reducer.py
```

Step 3:

Prepare *docker-compose.yml* through podman to connect to hadoop

- Note: assume this yml file has created a mnt folder
- exit the existing containers: `podman compose down`
- setup container: `podman compose up --build`
- connect to hadoop server: `podman exec -it namenode bash`

Step 4:

When you connect to hadoop server

- Prepare hdf
  - check files in current directory and mnt folder by `ls` and `ls /mnt`
  - Go to mnt folder by `cd /mnt`
  - check files in hdf by `hdfs dfs -ls /`
  - create a input file in there's not one by `hdfs dfs -mkdir /input`
  - put data into input folder by `hdfs dfs -put /mnt/corpus.txt /input/`
- Provide permission to the files
  - `chmod +x /mnt/mapper.py`
  - `chmod +x /mnt/reducer.py`
- Remember to remove output folder if it exist
  - `hdfs dfs -rm -r /output`
- Check if *source.list* file is up-to-date and replace it with the prepared file
  - assume we have prepare *sources.list* in your original folder
  - `cat /etc/apt/sources.list`
  - `cp /mnt/sources.list /etc/apt/sources.list`
- Make sure packages are up-to-date and have python3 installed
  - `apt-get update`
  - `apt-get install python3`
- Check .jar file path and run MapReduce code with found .jar file
  - `find / -name "hadoop-streaming*.jar"`
  - `hadoop jar /opt/hadoop-2.7.4/share/hadoop/tools/lib/hadoop-streaming-2.7.4.jar -input /input/corpus.txt -output /output -mapper /mnt/mapper.py -reducer /mnt/reducer.py`
- Save the output to a file named *part-00000*
  - `hdfs dfs -get /output/part-00000 /mnt/`

The following the log of running the steps above

hdf only takes data, you don't need to put into hdfs

```
(runapp) (base) yukezhang@Yukes-MacBook-Pro hadoop_mapreduce % podman
exec -it namenode bash
root@6c1b2a25bb57:/# ls
bin   dev            etc     hadoop-data  lib    media  opt   root
run.sh  srv  tmp  var
boot  entrypoint.sh  hadoop  home         lib64  mnt    proc  run
sbin    sys  usr
root@6c1b2a25bb57:/# cd /mnt
root@6c1b2a25bb57:/mnt# ls
corpus.txt  docker-compose.yml  mapper.py  reducer.py  sources.list
root@6c1b2a25bb57:/mnt# cat mapper.py
#!usr/bin/env python3
import sys

for line in sys.stdin:

    line = line.strip()
    words = line.split()

    for word in words:
        print("{}\t{}".format(word, 1))

### cat corpus.txt | python mapper.py | sort | python reducer.pyroot@
6c1b2a25bb57:/mnt# hdfs dfs -ls
ls: `.': No such file or directory
root@6c1b2a25bb57:/mnt# hdfs dfs -mkdir /input
root@6c1b2a25bb57:/mnt# hdfs dfs -ls
ls: `.': No such file or directory
root@6c1b2a25bb57:/mnt# hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - root supergroup          0 2024-10-09 13:31 /input
root@6c1b2a25bb57:/mnt# hdfs dfs -put /mnt/corpus.txt /input/
root@6c1b2a25bb57:/mnt# hdfs dfs -ls /input
Found 1 items
-rw-r--r--   3 root supergroup    5458199 2024-10-09 13:32
/input/corpus.txt
```

Get permission first and remove output folder if it exists

```
root@6c1b2a25bb57:/mnt# chmod +x /mnt/mapper.py
root@6c1b2a25bb57:/mnt# chmod +x /mnt/reducer.py
root@6c1b2a25bb57:/mnt# hdfs dfs -rm -r /output
```

check current source.list file, we should update it

```
root@6c1b2a25bb57:/mnt# cat /etc/apt/sources.list
```

```
deb http://deb.debian.org/debian jessie main
deb http://deb.debian.org/debian jessie-updates main
deb http://security.debian.org jessie/updates main
deb http://ftp.debian.org/debian jessie-backports main
root@6c1b2a25bb57:/mnt# cp /mnt/sources.list /etc/apt/sources.list
root@6c1b2a25bb57:/mnt# cat /etc/apt/sources.list
```

After update change source.list

```
apt-get update
apt-get install python3
```

Run the code

```
root@6c1b2a25bb57:/mnt# find / -name "hadoop-streaming*.jar"
/opt/hadoop-2.7.4/share/hadoop/tools/lib/hadoop-streaming-2.7.4.jar
/opt/hadoop-2.7.4/share/hadoop/tools/sources/hadoop-streaming-2.7.4-
sources.jar
/opt/hadoop-2.7.4/share/hadoop/tools/sources/hadoop-streaming-2.7.4-
test-sources.jar
root@6c1b2a25bb57:/mnt# hadoop jar
/opt/hadoop-2.7.4/share/hadoop/tools/lib/hadoop-streaming-2.7.4.jar -
input /input/corpus.txt -output /output -mapper /mnt/mapper.py -
reducer /mnt/reducer.py
```

Save the output file

```
hdfs dfs -get /output/part-00000 /mnt/
```