Experimental Instructions 2

Target：analyze the performance of Hadoop MapReduce applications by changing the resources reservation based on **Docker** engine.

This experiment is carried out in groups. Write your own part in your experiment report.

- **Step One:**
  - **Install Hadoop Docker**

    Update apt-get:

```
root@VM-16-47-ubuntu:~/cloudz/hadoop-2.7.3/etc/hadoop# apt-get update
Hit:1 http://mirrors.tencentyun.com/ubuntu xenial InRelease
Get:2 http://mirrors.tencentyun.com/ubuntu xenial-security InRelease [107 kB]
Get:3 http://mirrors.tencentyun.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://mirrors.tencentyun.com/ubuntu xenial-updates/main Sources [327 kB]
Get:5 http://mirrors.tencentyun.com/ubuntu xenial-updates/main amd64 Packages [893 kB]
Get:6 http://mirrors.tencentyun.com/ubuntu xenial-updates/main i386 Packages [790 kB]
Get:7 http://mirrors.tencentyun.com/ubuntu xenial-updates/universe amd64 Packages [715 kB]
Get:8 http://mirrors.tencentyun.com/ubuntu xenial-updates/universe i386 Packages [655 kB]
Fetched 3,596 kB in 0s (4,161 kB/s)
Reading package lists... Done
```

    Install curl:

```
root@VM-16-47-ubuntu:~/cloudz/hadoop-2.7.3/etc/hadoop# apt-get install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcurl3-gnutls
The following packages will be upgraded:
  curl libcurl3-gnutls
2 upgraded, 0 newly installed, 0 to remove and 253 not upgraded.
Need to get 323 kB of archives.
After this operation, 2,048 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://mirrors.tencentyun.com/ubuntu xenial-security/main amd64 curl amd64 7.47.0-1ubu
Get:2 http://mirrors.tencentyun.com/ubuntu xenial-security/main amd64 libcurl3-gnutls amd64
Fetched 323 kB in 0s (1,997 kB/s)
(Reading database ... 65978 files and directories currently installed.)
Preparing to unpack .../curl_7.47.0-1ubuntu2.11_amd64.deb ...
Unpacking curl (7.47.0-1ubuntu2.11) over (7.47.0-1ubuntu2.2) ...
```

Install docker:

```
root@VM-16-47-ubuntu:~/cloudz/hadoop-2.7.3/etc/hadoop# curl -fsSL https://get.docker.com/ | sh
# Executing docker install script, commit: 4957679
+ sh -c apt-get update -qq >/dev/null
+ sh -c apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | apt-key add -qq - >/dev/null
+ sh -c echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial edge" > /etc/apt/
+ sh -c apt-get update -qq >/dev/null
+ sh -c apt-get install -y -qq --no-install-recommends docker-ce >/dev/null
+ sh -c docker version
Client:
 Version:           18.09.0
 API version:       1.39
 Go version:        go1.10.4
 Git commit:        4d60db4
 Built:             Wed Nov  7 00:48:57 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
```

Check if succeed:

```
root@VM-16-95-ubuntu:~# docker -v
Docker version 18.09.0, build 4d60db4
```

Copy docker image from another server:

```
root@VM-16-95-ubuntu:~# scp ubuntu@172.21.16.47:~/hadoop-docker.tar .
ubuntu@172.21.16.47's password:
hadoop-docker.tar                                     100% 1722MB 191.3MB/s   00:09
```

Note: Please use **inner IP** to copy image between servers. And copy it **from your group leader**, not the path in the example above.

Load docker image:

```
root@VM-16-95-ubuntu:~# docker load --input hadoop-docker.tar
18493e95e551: Loading layer [==================================================>]  156.2MB/156.2MB
5f70bf18a086: Loading layer [==================================================>]  1.024kB/1.024kB
1f8baaf19ae9: Loading layer [==================================================>]  3.584kB/3.584kB
0e149f479fd4: Loading layer [==================================================>]  290.9MB/290.9MB
5f94e10b6d20: Loading layer [==================================================>]  65.01MB/65.01MB
bd71a02725d5: Loading layer [==================================================>]  413.9MB/413.9MB
5f70bf18a086: Loading layer [==================================================>]  1.024kB/1.024kB
5f70bf18a086: Loading layer [==================================================>]  1.024kB/1.024kB
5f70bf18a086: Loading layer [==================================================>]  1.024kB/1.024kB
5f70bf18a086: Loading layer [==================================================>]  1.024kB/1.024kB
d2bfbb2b0e1e: Loading layer [==================================================>]  2.048kB/2.048kB
971be2b28567: Loading layer [==================================================>]   68.9MB/68.9MB
00b37e8c7fe7: Loading layer [==================================================>]  2.331MB/2.331MB
```

Check if load succeed:

```
e9b9ff52f1f5: Loading layer [==================================================>]  2.552MB/2.552MB
Loaded image: sequenceiq/hadoop-docker:2.7.0
root@VM-16-95-ubuntu:~# docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
sequenceiq/hadoop-docker   2.7.0               789fa0a3b911        3 years ago         1.76GB
root@VM-16-95-ubuntu:~#
```

Start container:

```
root@VM-16-47-ubuntu:~# docker run -it sequenceiq/hadoop-docker:2.7.0 /etc/bootstrap.sh -bash
/
Starting sshd:                                             [  OK  ]
Starting namenodes on [90dac4cbc9dc]
90dac4cbc9dc: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-90dac4cbc9dc.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-90dac4cbc9dc.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-90dac4
ut
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-90dac4cbc9dc.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-90dac4cbc9dc.out
bash-4.1#
```

Hadoop environment has been installed:

```
bash-4.1# cd $HADOOP_PREFIX
bash-4.1# bin/hadoop
hadoop      hadoop.cmd
bash-4.1# bin/hadoop fs -ls /
Found 1 items
drwxr-xr-x   - root supergroup          0 2015-05-16 05:42 /user
```

Checking running containers list in a new session:

```
root@VM-16-47-ubuntu:~# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
    PORTS
                    NAMES
4c7b88b02f4f        sequenceiq/hadoop-docker:2.7.0   "/etc/bootstrap.sh -…"   About a minute ago   Up About a minut
e   2122/tcp, 8030-8033/tcp, 8040/tcp, 8042/tcp, 8088/tcp, 19888/tcp, 49707/tcp, 50010/tcp, 50020/tcp, 50070/tcp, 5
0075/tcp, 50090/tcp   amazing_shtern
1db4f13f7dcd        sequenceiq/hadoop-docker:2.7.0   "/bin/bash"              About an hour ago    Up About an hour
    2122/tcp, 8030-8033/tcp, 8040/tcp, 8042/tcp, 8088/tcp, 19888/tcp, 49707/tcp, 50010/tcp, 50020/tcp, 50070/tcp, 5
0075/tcp, 50090/tcp   Hadoop
```

- **Step Two:**
  - **Run Hadoop programs & Collect runtime information**

To finish this exercise, you need to run Hadoop MapReduce programs, and collect the resource consumption information of running container.

You can check docker resource like:

```
root@VM-16-47-ubuntu:~# docker stats

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
4c7b88b02f4f        amazing_shtern      0.01%               903.3MiB / 7.671GiB   11.50%              2.45kB / 1.44kB     81.9kB / 3.26MB     382
1db4f13f7dcd        Hadoop              0.00%               312KiB / 7.671GiB     0.00%               210B / 0B           0B / 0B             1
```

Note: **Hadoop** is the name of our running container.

Analyze these results and show your findings (e.g., text, tables, figures).

You can run any programs you want. Or you can choose and run Hadoop MapReduce example programs that you are interested in.

```
bash-4.1# cd $HADOOP_PREFIX
bash-4.1# bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.0.jar
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input files.
  wordmedian: A map/reduce program that counts the median length of the words in the input files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.
```

Example jar location in the container:

/usr/local/hadoop/share/hadoop/mapreduce/Hadoop-mapreduce-examples-2.7.0.jar

Note: Careful on your disk capacity because some example applications can generate extremely huge dataset. You can modify the source code (https://github.com/apache/hadoop/tree/release-2.7.0/hadoop-mapreduce-project/hadoop-mapreduce-examples/src/main/java/org/apache/hadoop/examples) to run applications with your own parameters.

- **Step Three:**
  - **Update container configuration**

We can limit the resource that can be used by a container. For example, you can limit container memory:

```
root@VM-16-47-ubuntu:~# docker update Hadoop --memory=1500m
Hadoop
```

Check if succeed:

```
root@VM-16-47-ubuntu:~# docker stats

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
4c7b88b02f4f        amazing_shtern      0.01%               1.033GiB / 7.671GiB   13.46%              2.45kB / 1.44kB     147kB / 4.34MB      382
1db4f13f7dcd        Hadoop              0.00%               312KiB / 1.465GiB     0.02%               210B / 0B           0B / 0B             1
```

In addition, you can limit CPU cores to 2:

```
root@VM-16-47-ubuntu:~# docker update Hadoop --cpu-period=100000 --cpu-quota=200000
Hadoop
```

You can also limit device IO rate when start a container through parameters like: --device-read-bps, --device-write-bps.

For detailed docker usage and parameters, you can refer to https://docs.docker.com/engine/reference/run/.

- **Analyze changes of applications**

  Re-run Hadoop MapReduce applications in container with limit resource. Record the changes of applications under various restrictions. And compare the performance (duration) of different applications.

  Note: Everyone in a group should **take charge of one part** of this experiment, for example, test performance changes of a kind of application. Show the analysis and result of your part in your report.