VE472 Lab Report

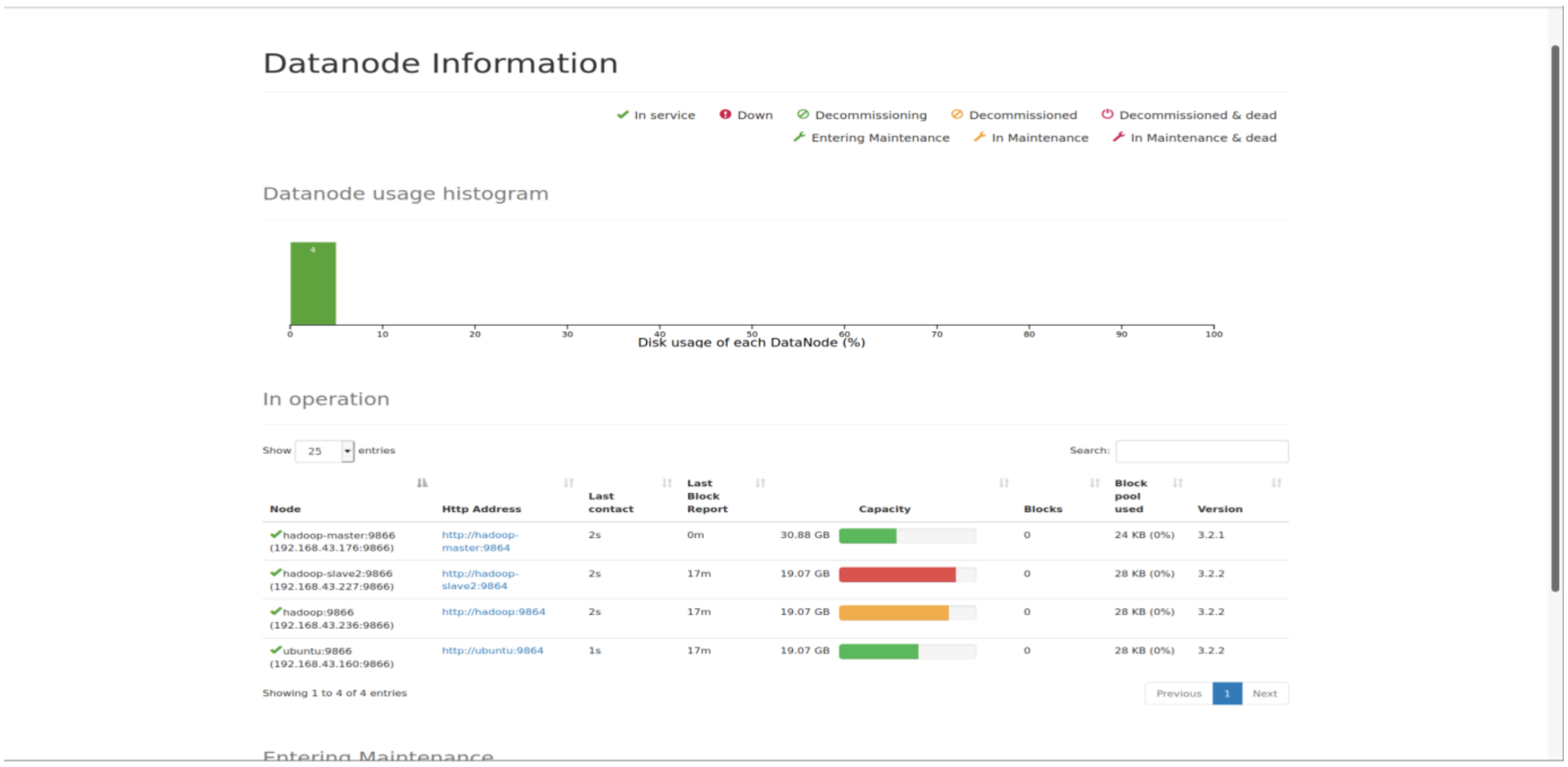
Lab 4 - Summer 2021

Name: Mingxuan Lu
ID: 518021911166
Email: mingxuan.lu@sjtu.edu.cn

Table of Contents

Ex. 1 Drill and Spark Installation

Our group has 4 members. Following instructions, we successfully set up our Hadoop cluster.



Datanode Information

✓ In service ● Down ⊘ Decommissioning ⊘ Decommissioned ⏻ Decommissioned & dead
✗ Entering Maintenance ✗ In Maintenance ✗ In Maintenance & dead

Datanode usage histogram

In operation

Show 25 entries Search:

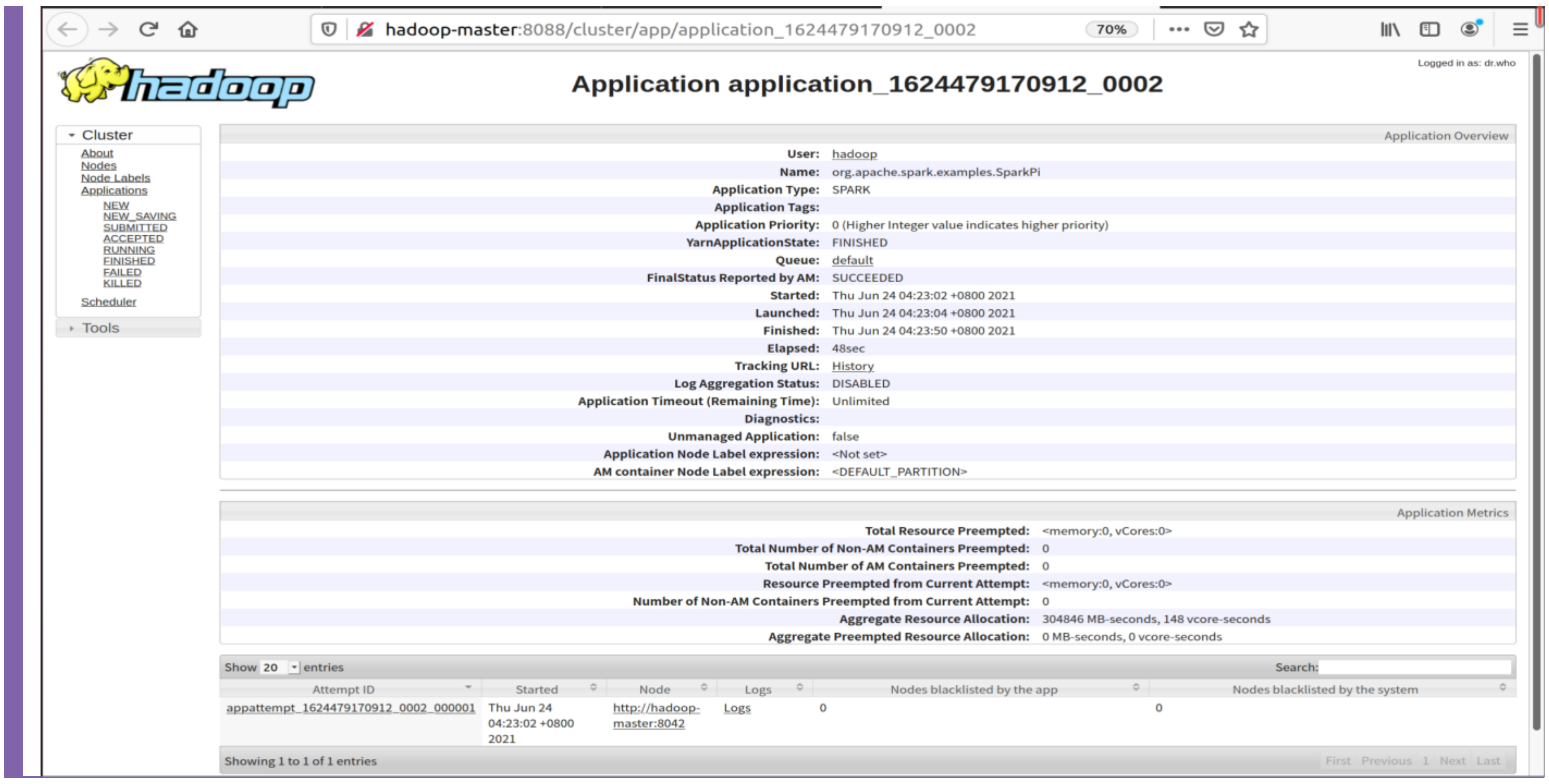
Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓hadoop-master:9866 (192.168.43.176:9866)	http://hadoop-master:9864	2s	0m	30.88 GB	0	24 KB (0%)	3.2.1
✓hadoop-slave2:9866 (192.168.43.227:9866)	http://hadoop-slave2:9864	2s	17m	19.07 GB	0	28 KB (0%)	3.2.2
✓hadoop:9866 (192.168.43.236:9866)	http://hadoop:9864	2s	17m	19.07 GB	0	28 KB (0%)	3.2.2
✓ubuntu:9866 (192.168.43.160:9866)	http://ubuntu:9864	1s	17m	19.07 GB	0	28 KB (0%)	3.2.2

Showing 1 to 4 of 4 entries Previous 1 Next

Entering Maintenance

Then we successfully set up Yarn, Drill and Spark in our cluster.

Once we find out that one of our nodes is unhealthy, but the whole cluster is still running, which proves the fault capacity of the hadoop cluster. Later we find that it is because the node has reached its capacity(since we are running the applications on virtual machines, the capacity is very limited). After we give more storage to the node, it turns back to healthy.



hadoop-master:8088/cluster/app/application_1624479170912_0002 70%

Logged in as: dr.who

Application application_1624479170912_0002

Application Overview

User:	hadoop
Name:	org.apache.spark.examples.SparkPi
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Thu Jun 24 04:23:02 +0800 2021
Launched:	Thu Jun 24 04:23:04 +0800 2021
Finished:	Thu Jun 24 04:23:50 +0800 2021
Elapsed:	48sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

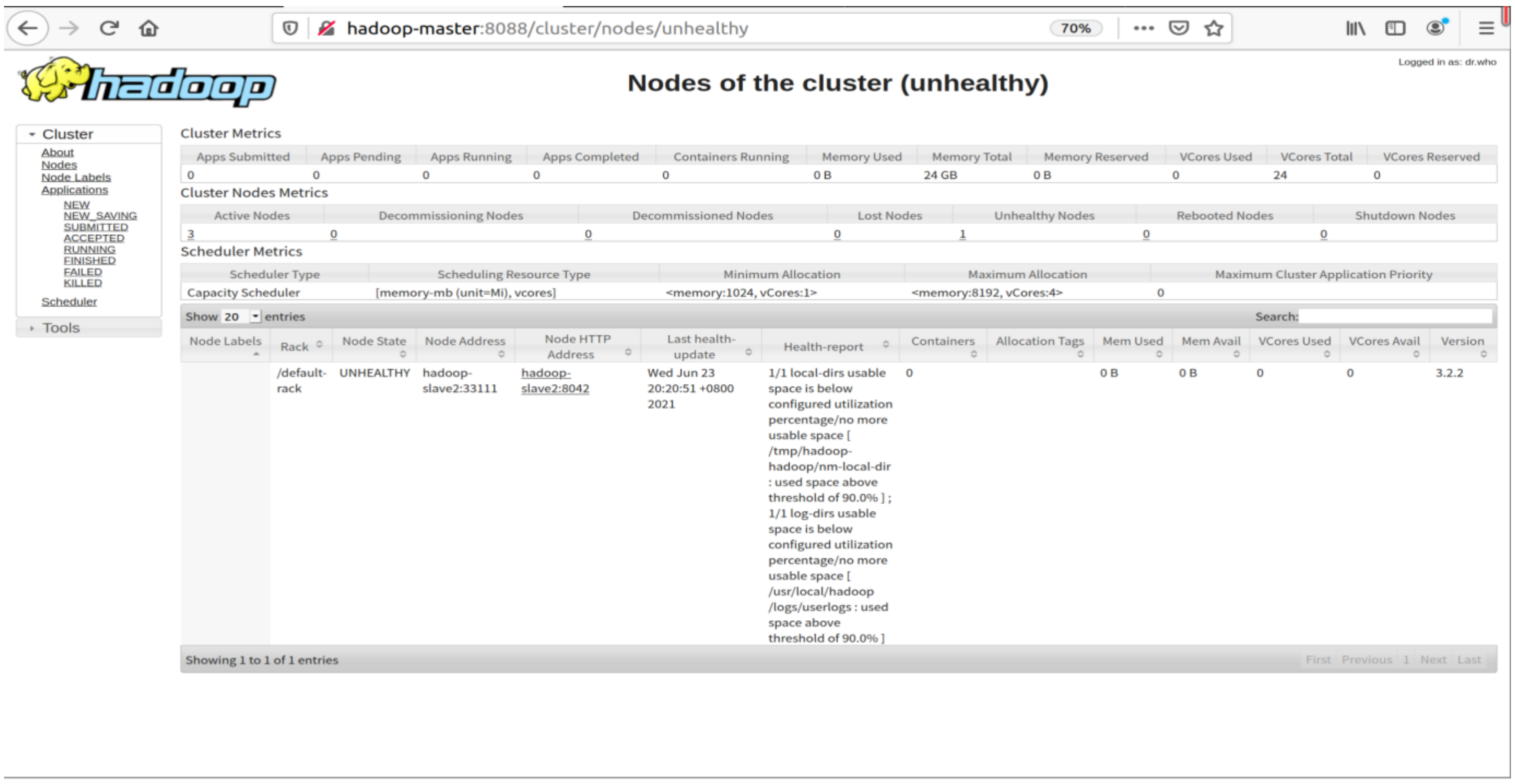
Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	304846 MB-seconds, 148 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

Show 20 entries Search:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1624479170912_0002_000001	Thu Jun 24 04:23:02 +0800 2021	http://hadoop-master:8042	Logs	0	0

Showing 1 to 1 of 1 entries First Previous 1 Next Last



Nodes of the cluster (unhealthy)

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	24 GB	0 B	0	24	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	0	1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	UNHEALTHY	hadoop-slave2:33111	hadoop-slave2:8042	Wed Jun 23 20:20:51 +0800 2021	1/1 local-dirs usable space is below configured utilization percentage/no more usable space [/tmp/hadoop-hadoop/nm-local-dir : used space above threshold of 90.0%] ; 1/1 log-dirs usable space is below configured utilization percentage/no more usable space [/usr/local/hadoop /logs/userlogs : used space above threshold of 90.0%]	0		0 B	0 B	0	0	3.2.2

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Ex.2 Simple Drill Queries

We first generate a large file with students and their grades.

```python
import random

firstnames = []
for line in open("firstnames.txt"):
    firstnames.append(line.strip())

lastnames = []
for line in open("lastnames.txt"):
    lastnames.append(line.strip())

for fileNum in range(0, 1):
    fileName = "mapper" + str(fileNum) + ".csv"
    names = []
    IDs = []
    f = open(fileName, "w")
    for i in range(0, 100000):  # total number of students is 100000
        ID = ""
        for x in range(0, 10):
            digit = random.randint(0, 9)
            ID = ID + (str(digit))

        names.append(random.choice(firstnames) + " " + random.choice(lastnames))
        IDs.append(ID)

    for j in range(0, 200000000):  # total number of lines is 200000000
        grade = random.randint(0, 100)
        index = random.randint(0, 99999)
        result = names[index] + "," + IDs[index] + "," + str(grade) + "\n"
        f.write(result)
```

Then we get a file `mapper0.csv`, which exceeds 5GB.

```
-rw-rw-r-- 1 hadoop hadoop 5741434141 Jun 28 00:07 mapper0.csv
```

By running

```
hdfs dfs -mkdir l4
hdfs dfs -put ./mapper0.csv l4
```

we put the file to hdfs.

We can check `http://10.119.6.238:8047/storage` for storage plugin configuration

2. 1)student who had the lowest grade:

```
apache drill> select name, min(score) as min_score from (select columns[0] as name, INTEGER columns[2] as score from hdfs.`/user/pgroup1/l4/mapper0.csv`) group by name order by min_score asc limit 1;
+----------------+-----------+
|      name      | min_score |
+----------------+-----------+
| Dwana Harradon | 0         |
+----------------+-----------+
1 row selected (31.429 seconds)
```

2)student who had the highest average grade:

```
apache drill> select name, AVG(score) as avg_score from (select columns[0] as name, cast(columns[2] as int) as score from hdfs.`/user/pgroup1/l4/mapper0.csv` ) group by name order by avg_score desc limit 1;
+----------------+------------------+
|      name      |    avg_score     |
+----------------+------------------+
| Domonique Mess | 53.2076209086468 |
+----------------+------------------+
1 row selected (48.804 seconds)
```

3. median over all the scores:

```
apache drill> SELECT COUNT(*) FROM hdfs.`/user/pgroup1/l4/mapper0.csv`;
+-----------+
|  EXPR$0   |
+-----------+
| 200000000 |
+-----------+
1 row selected (22.093 seconds)
apache drill> SELECT AVG(score)
2..semicolon> FROM (SELECT cast(columns[2] as int) as score
3.........)>       FROM hdfs.`/user/pgroup1/l4/mapper0.csv`
4.........)>       ORDER BY score
5.........)>       LIMIT 2    -- odd 1, even 2
6.........)>       OFFSET 99999999)
7..semicolon> ;
+--------+
| EXPR$0 |
+--------+
| 50.0   |
+--------+
1 row selected (85.175 seconds)
```

Ex.3 Simple Spark

(The contents below is identical to the `ex3.ipynb` file.)

```
"""
We first run `hdfs dfs -put ./grade.csv /` to put the local file "grade.csv" into hdfs.
"""
from pyspark import SparkConf, SparkContext
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))    # create an instance
```

```
# create an RDD from file
content = sc.textFile("grade.csv")

# .collect() serialize the RDD into a <list>
content.collect()

def f(x):
    a = x.split(',')
    return (a[1], int(a[2]))

mp = content.map(f)
test = mp.reduceByKey(lambda x, y: y if y>x else x)
test.collect()
"""
Check ex3_result for the result.
We don't show the result here because it's too long.
"""
```

```
"""
flatmap() function return a serialized result without Key-Value pair structure.
"""
fm = content.flatMap(f)
fm.collect()
```