김 원

EX1.

1)

```
%pyspark
gender_grouped_df = df.groupBy(df.gender)
gender_grouped_df.agg({"*": "count"}).collect()
```

```
[Row(gender=1, count(1)=647233), Row(gender=2, count(1)=401342)]
```

Took 13 sec. Last updated by anonymous at November 28 2017, 7:11:27 PM.

2)

```
%pyspark
df.describe(["cardholder",  "balance",  "numTrans"]).show()
```

```
+-------+------------------+------------------+------------------+
|summary|        cardholder|           balance|          numTrans|
+-------+------------------+------------------+------------------+
|  count|           1048575|           1048575|           1048575|
|   mean| 1.0299234675631215| 4111.859637126577|28.930147104403595|
| stddev|0.17037629334940388|3995.9449963844913|26.562409033728674|
|    min|                 1|                 0|                 0|
|    max|                 2|             39725|               100|
+-------+------------------+------------------+------------------+
```

Took 9 sec. Last updated by anonymous at November 28 2017, 7:14:06 PM.

3)

FINISHED

```
%pyspark
for col1 in ["cardholder",  "balance",  "numTrans"]:
    for col2 in ["cardholder",  "balance",  "numTrans"]:
        print "Correlation between {0}&{1} = ".format(col1, col2), df.corr(col1, col2, method
            ='pearson')
```

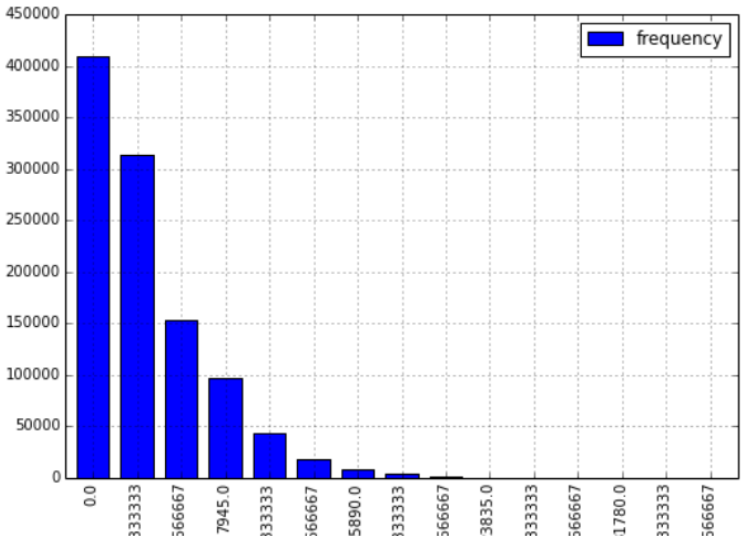
```
Correlation between cardholder&cardholder =  1.0
Correlation between cardholder&balance =  -0.000634395400249
Correlation between cardholder&numTrans =  0.00103231377033
Correlation between balance&cardholder =  -0.000634395400249
Correlation between balance&balance =  1.0
Correlation between balance&numTrans =  0.000311155436117
Correlation between numTrans&cardholder =  0.00103231377033
Correlation between numTrans&balance =  0.000311155436117
Correlation between numTrans&numTrans =  1.0
```

Took 1 min 15 sec. Last updated by anonymous at November 28 2017, 7:43:31 PM.

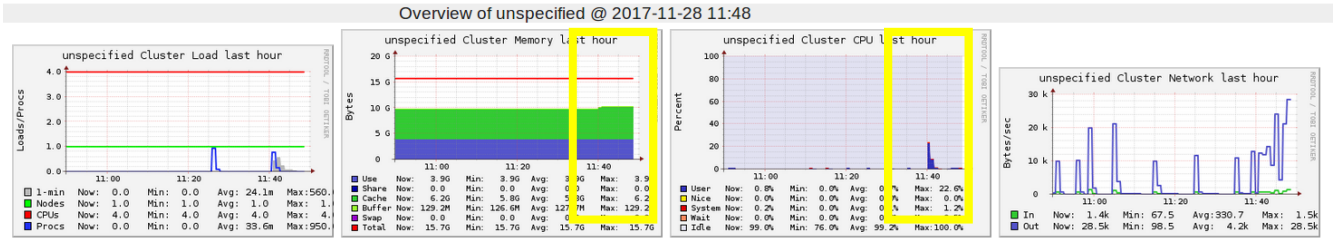
4)

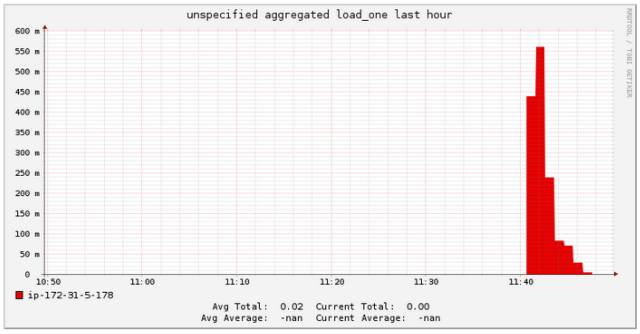
```
%pyspark
pd.DataFrame(zip(list(balance_hist)[0], list(balance_hist)[1]),columns=['bin','frequency']).set_index('bin').plot(kind='bar');
```

```
<matplotlib.axes.AxesSubplot object at 0x7f04e6b12ad0>
```



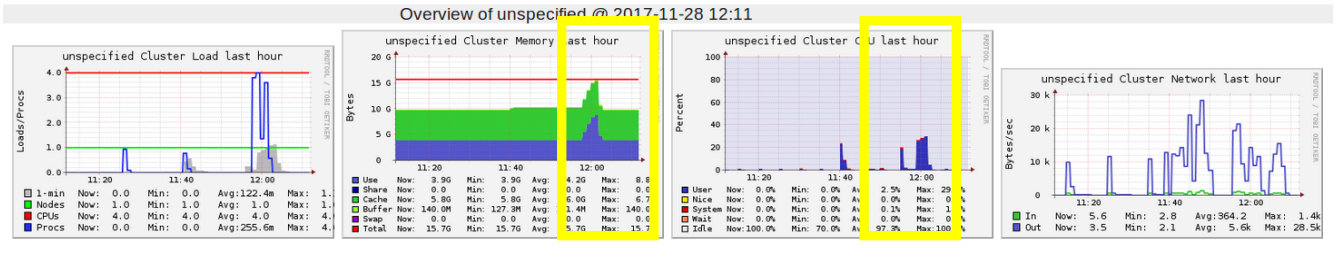
EX2.
1) wordcount

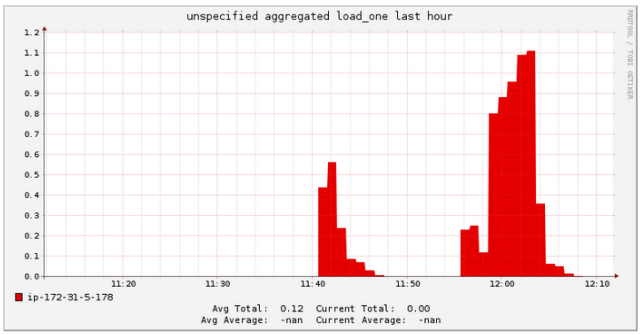




→ wordcount task의 경우, CPU-bound이다. Memory와 CPU 사용 추세를 각각 살펴보면, memory 사용은 매우 조금 늘어난 반면, CPU는 활발하게 사용되는 것으로 나타난다. cpu의 task를 끝마칠 것을 memory가 기다리지 않는 것으로 보아, bottleneck은 없는 것으로 판단된다.

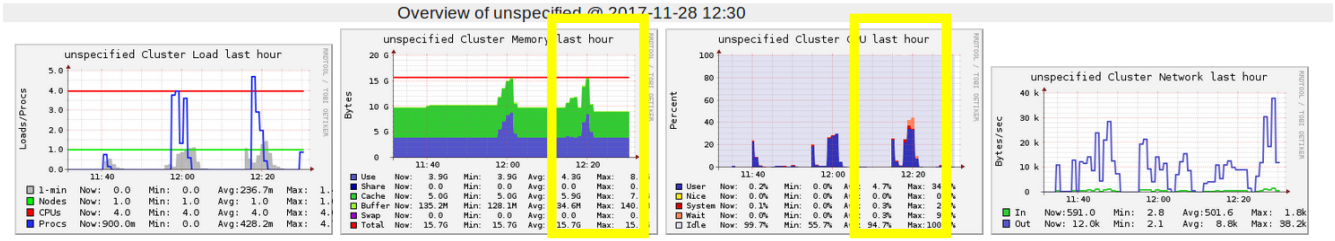
2) pagerank

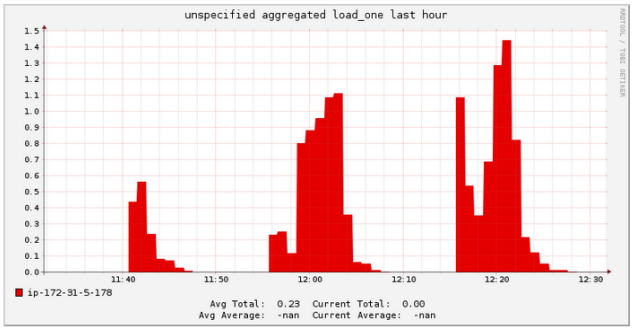




→ pagerank task의 경우, memory-bound이다. Memory와 CPU 사용 추세는 둘 다 급격하게 상승한 것을 볼 수 있다. Memory는 CPU가 작업을 마치고 memory에 넘겨주기를 기다리지 않는다. 따라서 bottleneck은 없다고 볼 수 있다.

3) terasort





→ terasort task의 경우, CPU-bound이다. Memory와 CPU 사용 추세는 둘 다 급격하게 상승한 것을 볼 수 있다. 하지만 CPU가 해당 task의 bottleneck이라고 판단할 수 있다. 왜냐하면 memory는 CPU가 작업을 마치고 자신에게 넘겨주기를 기다리기 때문이다.