# Performance Comparison of PySpark Programs on Different Cores

2023201012

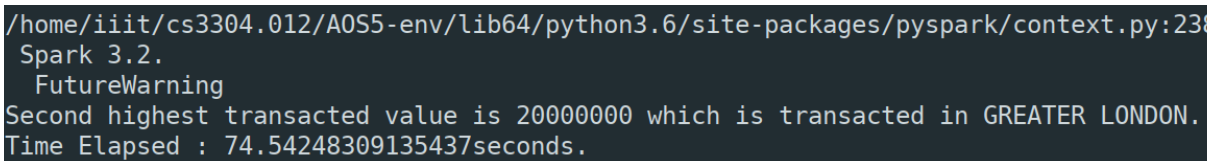
## 1 Introduction

This report presents a performance comparison of PySpark programs designed to analyze transaction data on an Abacus system. The programs were executed with different numbers of cores (2, 4, and 6) to assess their scalability.

## 2 Program 1: Second Highest Value Transaction ('Price') in Selected Countries

### 2.1 Results

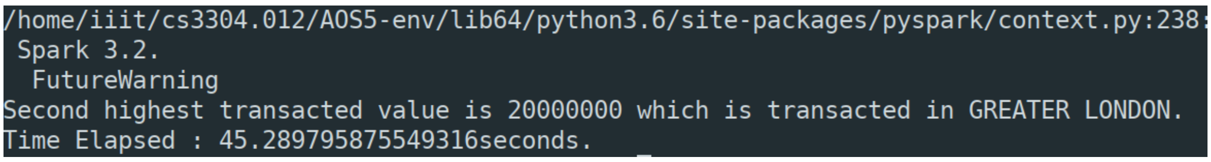
#### 2.1.1 2 Cores



```
/home/iiit/cs3304.012/AOS5-env/lib64/python3.6/site-packages/pyspark/context.py:23
 Spark 3.2.
  FutureWarning
Second highest transacted value is 20000000 which is transacted in GREATER LONDON.
Time Elapsed : 74.54248309135437seconds.
```

Figure 1: Screenshot of Program 1 Output with 2 Cores

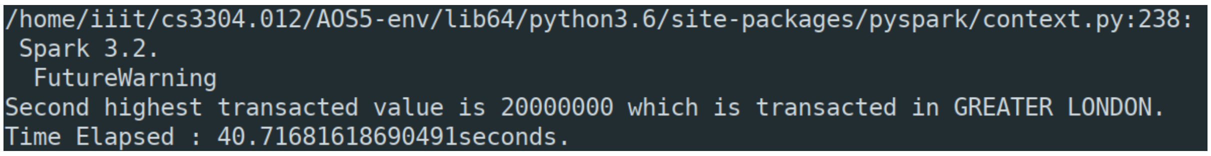
#### 2.1.2 4 Cores



```
/home/iiit/cs3304.012/AOS5-env/lib64/python3.6/site-packages/pyspark/context.py:238
 Spark 3.2.
  FutureWarning
Second highest transacted value is 20000000 which is transacted in GREATER LONDON.
Time Elapsed : 45.289795875549316seconds.
```

Figure 2: Screenshot of Program 1 Output with 4 Cores

### 2.1.3 6 Cores



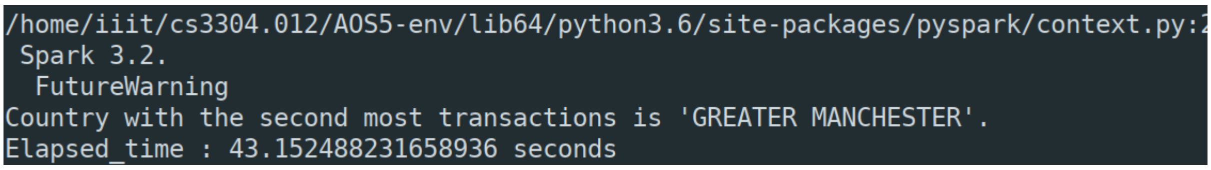
```
/home/iiit/cs3304.012/AOS5-env/lib64/python3.6/site-packages/pyspark/context.py:238:
 Spark 3.2.
  FutureWarning
Second highest transacted value is 20000000 which is transacted in GREATER LONDON.
Time Elapsed : 40.71681618690491seconds.
```

Figure 3: Screenshot of Program 1 Output with 6 Cores

## 3 Program 2: Country with Second Most Transactions

### 3.1 Results

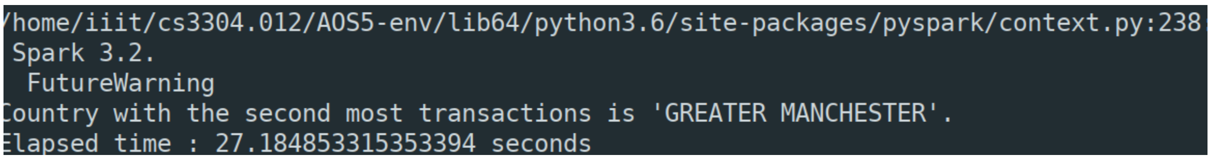
#### 3.1.1 2 Cores



```
/home/iiit/cs3304.012/AOS5-env/lib64/python3.6/site-packages/pyspark/context.py:2
 Spark 3.2.
  FutureWarning
Country with the second most transactions is 'GREATER MANCHESTER'.
Elapsed_time : 43.152488231658936 seconds
```

Figure 4: Screenshot of Program 2 Output with 2 Cores

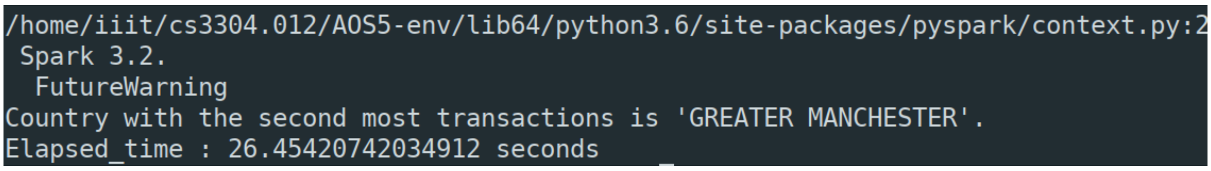
#### 3.1.2 4 Cores



```
/home/iiit/cs3304.012/AOS5-env/lib64/python3.6/site-packages/pyspark/context.py:238
 Spark 3.2.
  FutureWarning
Country with the second most transactions is 'GREATER MANCHESTER'.
Elapsed_time : 27.184853315353394 seconds
```

Figure 5: Screenshot of Program 2 Output with 4 Cores

#### 3.1.3 6 Cores



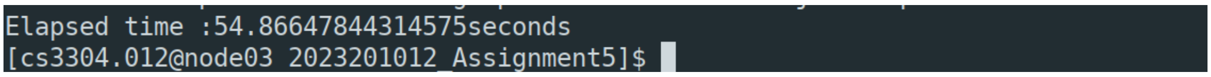
```
/home/iiit/cs3304.012/AOS5-env/lib64/python3.6/site-packages/pyspark/context.py:2
 Spark 3.2.
  FutureWarning
Country with the second most transactions is 'GREATER MANCHESTER'.
Elapsed_time : 26.45420742034912 seconds
```

Figure 6: Screenshot of Program 2 Output with 6 Cores

## 4 Program 3: Counting Transactions per Country

### 4.1 Results

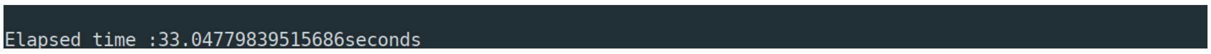
#### 4.1.1 2 Cores



```
Elapsed time :54.86647844314575seconds
[cs3304.012@node03 2023201012_Assignment5]$
```

Figure 7: Screenshot of Program 3 Output with 2 Cores

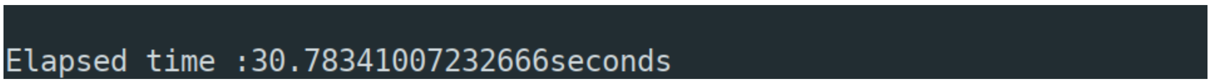
#### 4.1.2 4 Cores



```
Elapsed time :33.04779839515686seconds
```

Figure 8: Screenshot of Program 3 Output with 4 Cores

#### 4.1.3 6 Cores



```
Elapsed time :30.78341007232666seconds
```

Figure 9: Screenshot of Program 3 Output with 6 Cores

## 5 Time Comparison Table

Table 1: Time Comparison for Each Program in Different Core Configurations

| **Program** | **Execution Time** | | |
|---|---|---|---|
| | **2 Cores** | **4 Cores** | **6 Cores** |
| Program 1 | *74 seconds* | *45 seconds* | *40 seconds* |
| Program 2 | *43 seconds* | *27 seconds* | *26 seconds* |
| Program 3 | *54 seconds* | *33 seconds* | *30 seconds* |

## 6 Observations

### 6.1 Program 1

The execution time decreases as the number of cores increases. Noticeable improvement from 2 to 4 cores, with further decrease from 4 to 6 cores. Suggests significant benefits from parallel processing in Program 1.

### 6.2 Program 2

Consistent decrease in execution time with increasing core count. Evident improvement in performance, with good scalability from 2 to 6 cores.

### 6.3 Program 3

Consistent trend of reduced execution time with more cores. Evident improvement in performance, with good scalability from 2 to 6 cores.

### 6.4 Overall Observations

All programs show improved performance with more cores. Varying degrees of improvement, but generally, more cores lead to faster execution times. Programs are well-suited for parallel processing, utilizing available computational resources.

## 7 Conclusion

For optimal performance, we can consider running programs on a higher number of cores. Increasing processor cores initially significantly speeds up program execution, but beyond a certain point, adding more cores provides diminishing returns—limited additional speedup for each extra core.