

CS3.304 - AOS Assignment 5 - Report

About:

Py-SPARK is a powerful open-source distributed computing system that provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. In this assignment, we utilized Py-SPARK to analyze the House Pricing dataset. The dataset contains information on property sales in England and Wales, including transaction unique identifiers, prices, and countries.

Objective:

In this report, I provide a comparison of the time taken for three distinct tasks using different numbers of CPU cores. The findings are presented in three sections, each corresponding to the performance with 2, 4, and 6 cores. Additionally, visual representations have been included to illustrate the performance trends for each task under varying core configurations. This comparison aims to offer insights into the efficiency of parallel processing across different core counts, facilitating a comprehensive understanding of the computational performance.

How to Run the code:

To execute the Py-SPARK programs on Abacus, follow the instructions provided:

Login to Abacus:

`ssh cs3304.24@abacus.iiit.ac.in`

Create Virtual Environment and Install Py-SPARK:

```
pip install pyspark
```

Copy Files to Abacus:

```
scp <source_file_path>  
<user_id>@abacus.iiit.ac.in:<destination_path>
```

Requesting an Interactive Job:

```
sint3 -c 2/4/6
```

Execute Python Program:

```
python3 2023201024_q1/q2/q3.py
```

Relinquish the Job:

```
exit
```

Comparison:

2 Cores:

Time taken using 2 cores are as follows

- Question 1: 42.02 seconds
- Question 2: 41.23 seconds
- Question 3: 41.81 seconds

```
Workspaces Applications Nov 13 5:27 PM
cs3304.024@abacus:~/venv

[cs3304.024@abacus venv]$ sint3 -c 2
salloc: Granted job allocation 1061281
salloc: Waiting for resource configuration
salloc: Nodes node06 are ready for job
[cs3304.024@node06 venv]$ source bin/activate
(venv) [cs3304.024@node06 venv]$ python3 2023201024_q1.py

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:19:45 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
  FutureWarning

===== OUTPUT OF QUESTION 1 =====

Second highest transacted value is 20000000 which is transacted in GREATER LONDON.
Execution time for Question 1: 42.02441191673279 seconds

=====
(venv) [cs3304.024@node06 venv]$
(venv) [cs3304.024@node06 venv]$ python3 2023201024_q2.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:20:37 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
  FutureWarning

===== OUTPUT OF QUESTION 2 =====

Country with second most transactions is 'GREATER MANCHESTER'.

Execution time for Question 2: 41.230886459350586 seconds

=====
```

```
Workspaces Applications Nov 13 5:31 PM
cs3304.024@node03:~/venv

[cs3304.024@abacus venv]$ python3 2023201024_q3.py
Traceback (most recent call last):
  File "2023201024_q3.py", line 1, in <module>
    from pyspark.sql import SparkSession
ModuleNotFoundError: No module named 'pyspark'
[cs3304.024@abacus venv]$ sint3 -c 2
salloc: Granted job allocation 1061284
salloc: Waiting for resource configuration
salloc: Nodes node03 are ready for job
[cs3304.024@node03 venv]$ source bin/activate
(venv) [cs3304.024@node03 venv]$ python3 2023201024_q3.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:30:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
  FutureWarning

===== OUTPUT OF QUESTION 3 =====

Execution time: 41.815133571624756 seconds

=====
(venv) [cs3304.024@node03 venv]$
```

4 Cores:

Time taken using 2 cores are as follows

- Question 1: 26.53 seconds
- Question 2: 27.64 seconds
- Question 3: 28.29 seconds

```

Workspaces Applications Nov 13 5:34 PM
cs3304.024@node03:~/venv
[cs3304.024@abacus venv]$ sint3 -c 4
salloc: Granted job allocation 1061286
salloc: Waiting for resource configuration
salloc: Nodes node03 are ready for job
[cs3304.024@node03 venv]$ source bin/activate
(venv) [cs3304.024@node03 venv]$ python3 2023201024_q1.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:33:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning

===== OUTPUT OF QUESTION 1 =====

Second highest transacted value is 200000000 which is transacted in GREATER LONDON.
Execution time for Question 1: 26.53835916519165 seconds

=====
(venv) [cs3304.024@node03 venv]$ python3 2023201024_q2.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:34:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning

===== OUTPUT OF QUESTION 2 =====

Country with second most transactions is 'GREATER MANCHESTER'.
Execution time for Question 2: 27.64998304214478 seconds

=====
(venv) [cs3304.024@node03 venv]$

```

```

Workspaces Applications Nov 13 5:35 PM
cs3304.024@node03:~/venv
(venv) [cs3304.024@node03 venv]$ python3 2023201024_q3.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:35:17 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning

===== OUTPUT OF QUESTION 3 =====

Execution time: 28.29346442225952 seconds

=====
(venv) [cs3304.024@node03 venv]$

```

6 Cores:

Time taken using 2 cores are as follows

- Question 1: 23.30 seconds

- Question 2: 22.70 seconds
- Question 3: 22.46 seconds

```
Workspaces Applications Nov 13 5:40 PM
cs3304.024@node01:~/venv

cs3304.024@node01:~/venv
salloc: Granted job allocation 1061291
salloc: Waiting for resource configuration
salloc: Nodes node01 are ready for job
[cs3304.024@node01 ~]$ cd venv/
[cs3304.024@node01 venv]$ source bin/activate
(venv) [cs3304.024@node01 venv]$ python3 2023201024_q1.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:59:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning
===== OUTPUT OF QUESTION 1 <=====
Second highest transacted value is 20000000 which is transacted in GREATER LONDON.
Execution time for Question 1: 23.305224418640137 seconds

(venv) [cs3304.024@node01 venv]$ python3 2023201024_q2.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:59:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning
===== OUTPUT OF QUESTION 2 <=====
Country with second most transactions is 'GREATER MANCHESTER'.
Execution time for Question 2: 22.70239782333374 seconds

(venv) [cs3304.024@node01 venv]$
```

```
Workspaces Applications Nov 13 5:41 PM
cs3304.024@node01:~/venv

cs3304.024@node01:~/venv
(venv) [cs3304.024@node01 venv]$ python3 2023201024_q3.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/13 17:40:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/iiit/cs3304.024/venv/lib64/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning
===== OUTPUT OF QUESTION 3 <=====
Execution time: 22.467137098312378 seconds

(venv) [cs3304.024@node01 venv]$
```

Results:

Table:

Cores	Question 1	Question 2	Question 3
2	42.02s	41.23s	41.81s
4	26.53s	27.64s	28.29s
6	23.30s	22.70s	22.46s

Observations:

- As the number of cores increases from 2 to 6, there is a noticeable decrease in the time taken for all three questions.
- Question 3 consistently shows a faster execution compared to Question 1 and Question 2 across different core configurations.
- The reduction in time is more prominent when moving from 2 to 4 cores than from 4 to 6 cores, indicating diminishing returns with additional cores.

These observations suggest that parallel processing with a higher number of cores contributes to improved efficiency, particularly noticeable in the transition from 2 to 4 cores. However, the gains become less significant when moving from 4 to 6 cores. The choice of the number of cores should be considered in light of achieving an optimal balance between computational power and resource utilization.