🌟

DIC - HW3 - PYSPARK

Name: Bhanu Chakra Sai Tarun Reddi **UBIT:** bhanucha

UB Number: 50545060 **Email:** bhanucha@buffalo.edu

In this report I’ll be presenting two programs of word count one is basic and the other is little complex with some cleaning and sorting functions with the use of spark, , the environment where we running this is Jupiter notebook which is installed in the WSL (Windows subsystem for Linux). Let’s now look at the each implementations below.

Here we’re running the apace spark and jupyter notebook on the WSL

Basic Word Count

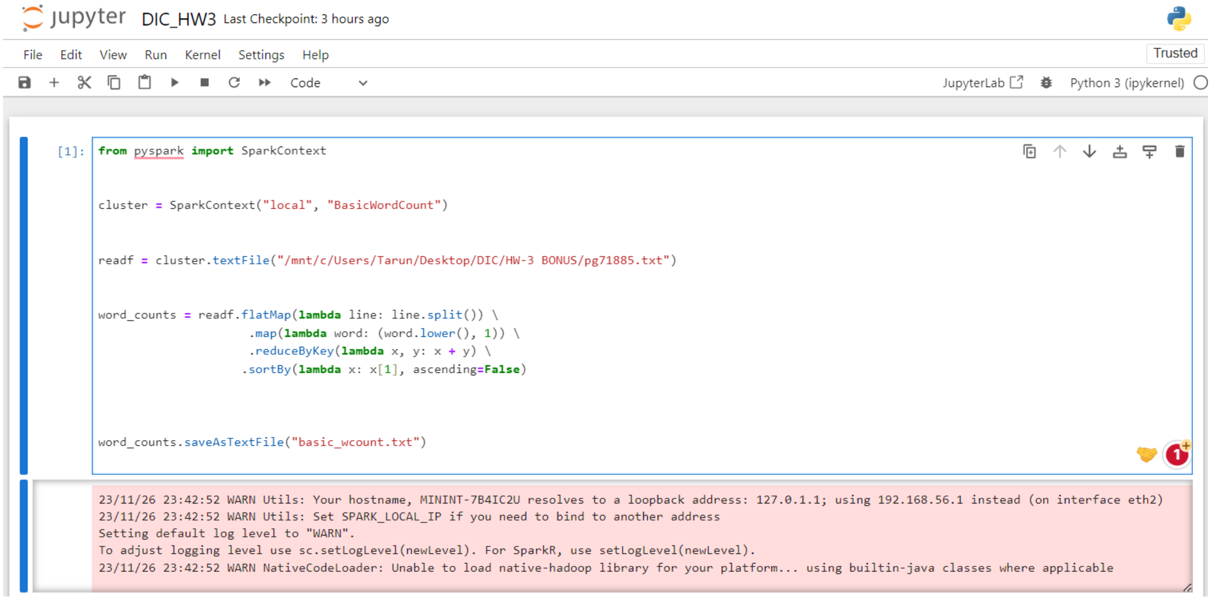
The implemented program here analyzes a text file by counting how many times each word appears. It then produces an output in a text file, displaying a list where each word is paired with the number of times it occurs.

Program to Basic Word Count:

```
from pyspark import SparkContext


cluster = SparkContext("local", "BasicWordCount")


readf = cluster.textFile("/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71885.txt")


word_counts = readf.flatMap(lambda line: line.split()) \
                     .map(lambda word: (word.lower(), 1)) \
                     .reduceByKey(lambda x, y: x + y) \
                    .sortBy(lambda x: x[1], ascending=False)
```

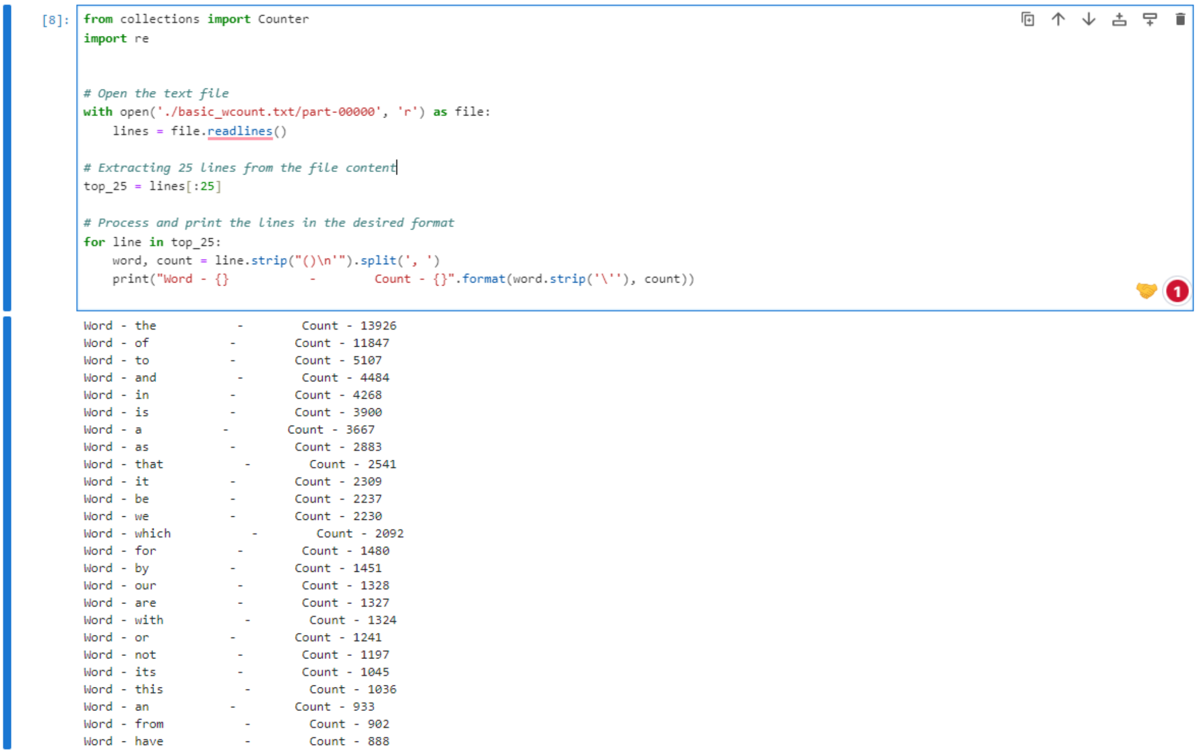
```
word_counts.saveAsTextFile("basic_wcount.txt")
```

Output:


jupyter DIC_HW3 Last Checkpoint: 3 hours ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[1]: from pyspark import SparkContext

cluster = SparkContext("local", "BasicWordCount")

readf = cluster.textFile("/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71885.txt")

word_counts = readf.flatMap(lambda line: line.split()) \
                    .map(lambda word: (word.lower(), 1)) \
                    .reduceByKey(lambda x, y: x + y) \
                   .sortBy(lambda x: x[1], ascending=False)


word_counts.saveAsTextFile("basic_wcount.txt")
```

```
23/11/26 23:42:52 WARN Utils: Your hostname, MININT-7B4IC2U resolves to a loopback address: 127.0.1.1; using 192.168.56.1 instead (on interface eth2)
23/11/26 23:42:52 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/26 23:42:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```



```
[8]: from collections import Counter
import re


# Open the text file
with open('./basic_wcount.txt/part-00000', 'r') as file:
    lines = file.readlines()

# Extracting 25 lines from the file content
top_25 = lines[:25]

# Process and print the lines in the desired format
for line in top_25:
    word, count = line.strip("()\n'").split(', ')
    print("Word - {}            -        Count - {}".format(word.strip('\''), count))
```

```
Word - the            -        Count - 13926
Word - of            -        Count - 11847
Word - to            -        Count - 5107
Word - and            -        Count - 4484
Word - in            -        Count - 4268
Word - is            -        Count - 3900
Word - a            -        Count - 3667
Word - as            -        Count - 2883
Word - that            -        Count - 2541
Word - it            -        Count - 2309
Word - be            -        Count - 2237
Word - we            -        Count - 2230
Word - which            -        Count - 2092
Word - for            -        Count - 1480
Word - by            -        Count - 1451
Word - our            -        Count - 1328
Word - are            -        Count - 1327
Word - with            -        Count - 1324
Word - or            -        Count - 1241
Word - not            -        Count - 1197
Word - its            -        Count - 1045
Word - this            -        Count - 1036
Word - an            -        Count - 933
Word - from            -        Count - 902
Word - have            -        Count - 888
```

Question 1

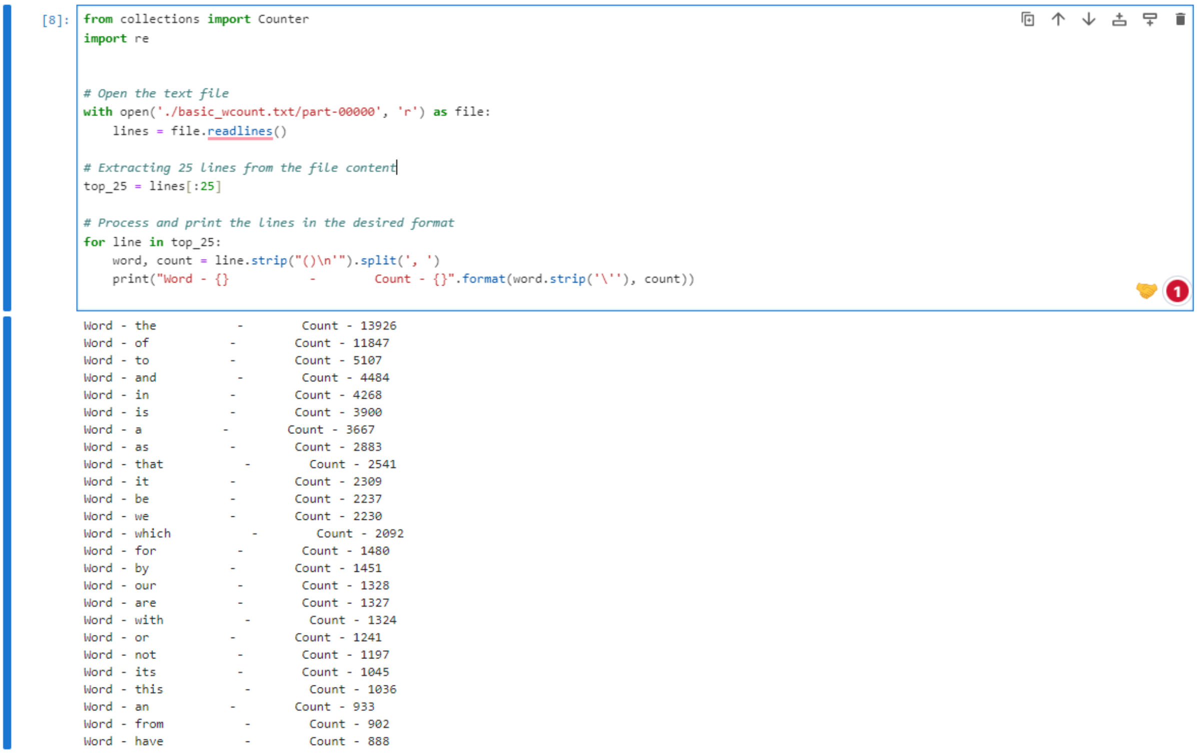
**In the PySpark REPL/ Jupyter notebook , run your basic word count program on a single
text file.**

1. **What are the 25 most common words? Include a screenshot of program output to
back-up your claim.**
2. **How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim**

Answer for part 1:

My Top 25 Words:

```
Word - the           -        Count - 13926
Word - of           -        Count - 11847
Word - to           -        Count - 5107
Word - and           -        Count - 4484
Word - in           -        Count - 4268
Word - is           -        Count - 3900
Word - a           -        Count - 3667
Word - as           -        Count - 2883
Word - that           -        Count - 2541
Word - it           -        Count - 2309
Word - be           -        Count - 2237
Word - we           -        Count - 2230
Word - which           -        Count - 2092
Word - for           -        Count - 1480
Word - by           -        Count - 1451
Word - our           -        Count - 1328
Word - are           -        Count - 1327
Word - with           -        Count - 1324
Word - or           -        Count - 1241
Word - not           -        Count - 1197
Word - its           -        Count - 1045
Word - this           -        Count - 1036
Word - an           -        Count - 933
Word - from           -        Count - 902
Word - have           -        Count - 888
```



```
[8]: from collections import Counter
     import re


     # Open the text file
     with open('./basic_wcount.txt/part-00000', 'r') as file:
         lines = file.readlines()

     # Extracting 25 lines from the file content
     top_25 = lines[:25]

     # Process and print the lines in the desired format
     for line in top_25:
         word, count = line.strip("()\n'").split(', ')
         print("Word - {}           -        Count - {}".format(word.strip('\''), count))
```

```
Word - the           -        Count - 13926
Word - of           -        Count - 11847
Word - to           -        Count - 5107
Word - and           -        Count - 4484
Word - in           -        Count - 4268
Word - is           -        Count - 3900
Word - a           -        Count - 3667
Word - as           -        Count - 2883
Word - that           -        Count - 2541
Word - it           -        Count - 2309
Word - be           -        Count - 2237
Word - we           -        Count - 2230
Word - which           -        Count - 2092
Word - for           -        Count - 1480
Word - by           -        Count - 1451
Word - our           -        Count - 1328
Word - are           -        Count - 1327
Word - with           -        Count - 1324
Word - or           -        Count - 1241
Word - not           -        Count - 1197
Word - its           -        Count - 1045
Word - this           -        Count - 1036
Word - an           -        Count - 933
Word - from           -        Count - 902
Word - have           -        Count - 888
```

Answer for part 2:

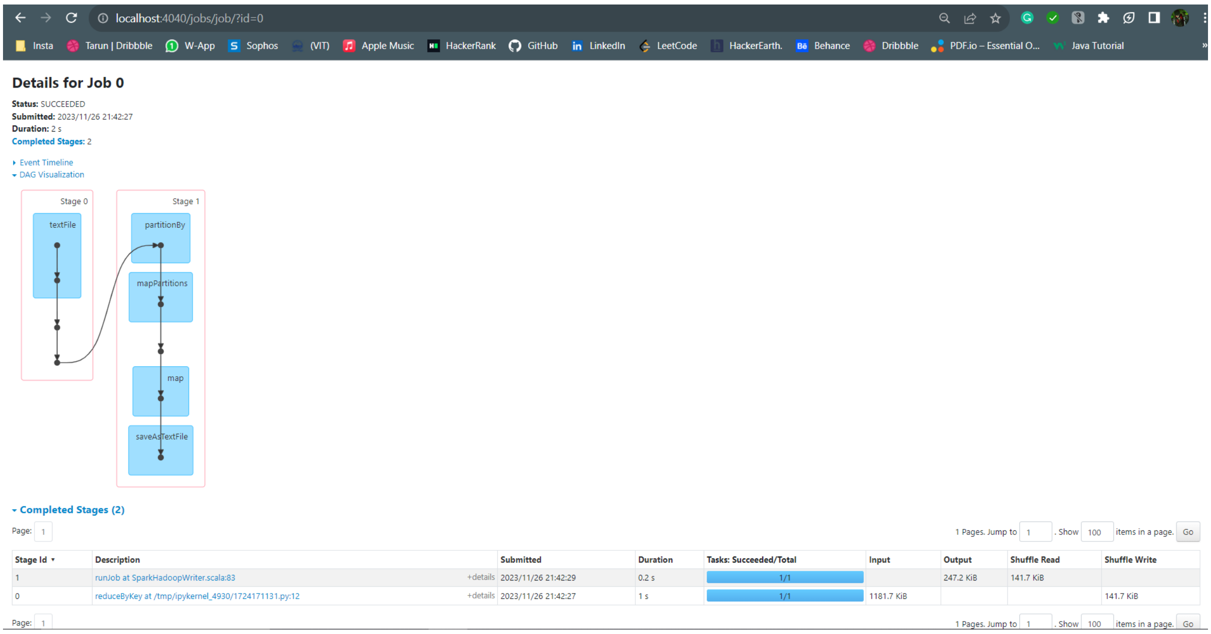
The execution is broken down into 2 stages

1. **Stage 0:** this is the initial stage where reading the text file and performing operations like reducing it by key.
2. **Stage 1:** After the stage 0, the stage 1 is performed, in this stage the operations like partitioning the data, mapping the partitioned data, and fnally saving the all processed and tranformed data takes place.

Why:

The division of stage occurs as Apache Spark is to optimize the overall computation process whenever theres a data shuffling occurs and also helps in fault tolerance, as tasks are distributed across the stages this helps in the control flow of the operations and optimize the resources utilized.

DAG visualization from Spark's WebUI:



Details for Job 0

Status: SUCCEEDED
Submitted: 2023/11/26 21:42:27
Duration: 2 s
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization

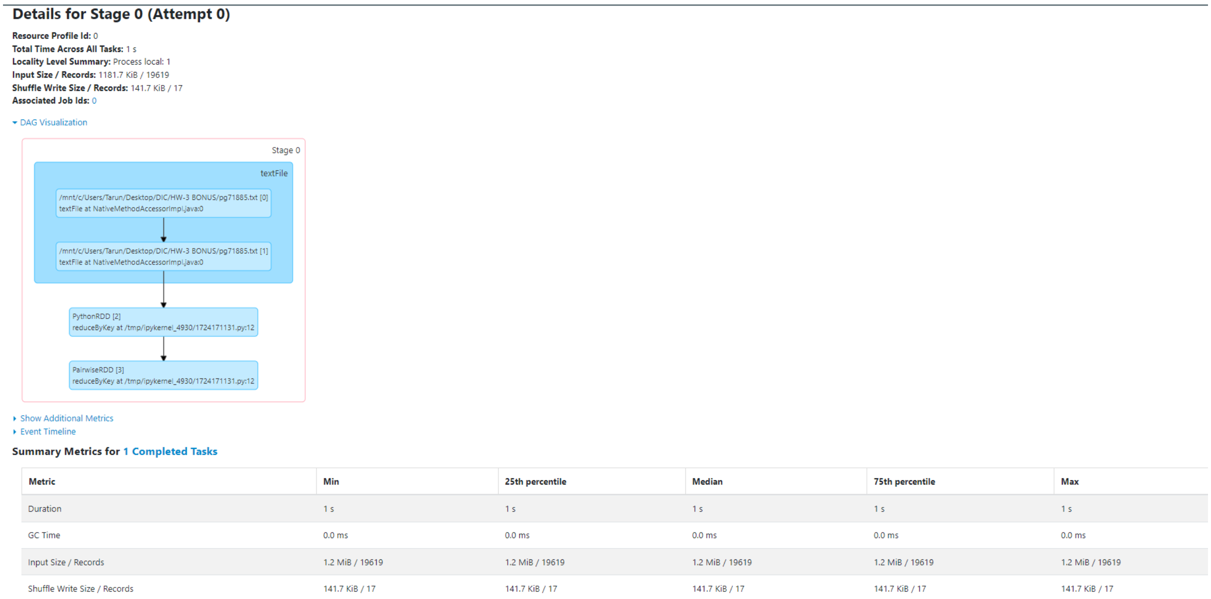
▾ Completed Stages (2)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	runJob at SparkHadoopWriter.scala:83 +details	2023/11/26 21:42:29	0.2 s	1/1		247.2 KiB	141.7 KiB	
0	reduceByKey at /tmp/ipykernel_4930/1724171131.py:12 +details	2023/11/26 21:42:27	1 s	1/1	1181.7 KiB			141.7 KiB

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Details for the Job


Stage 0
textFile
/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71885.txt [0]
textFile at NativeMethodAccessorImpl.java:0
/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71885.txt [1]
textFile at NativeMethodAccessorImpl.java:0
PythonRDD [2]
reduceByKey at /tmp/ipykernel_4930/1724171131.py:12
PairwiseRDD [3]
reduceByKey at /tmp/ipykernel_4930/1724171131.py:12

Details for Stage 0 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 1 s
Locality Level Summary: Process local: 1
Input Size / Records: 1181.7 KiB / 19619
Shuffle Write Size / Records: 141.7 KiB / 17
Associated Job Ids: 0

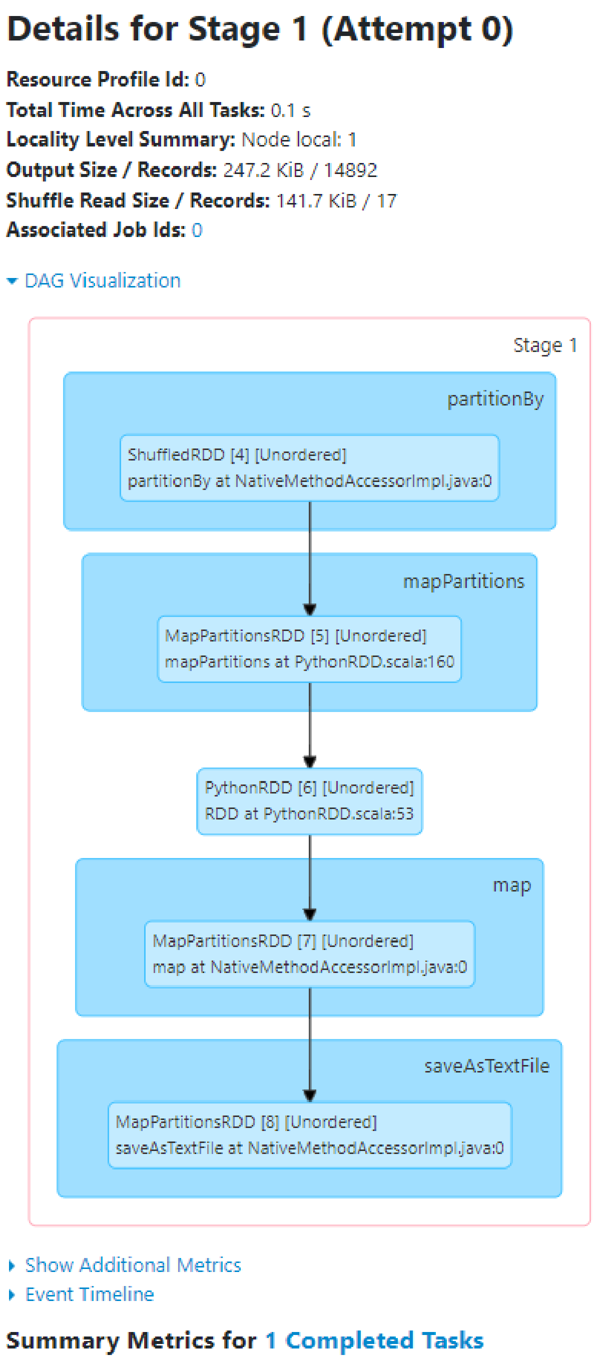
▾ DAG Visualization

▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 1 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	1 s	1 s	1 s	1 s	1 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Input Size / Records	1.2 MiB / 19619	1.2 MiB / 19619	1.2 MiB / 19619	1.2 MiB / 19619	1.2 MiB / 19619
Shuffle Write Size / Records	141.7 KiB / 17	141.7 KiB / 17	141.7 KiB / 17	141.7 KiB / 17	141.7 KiB / 17

This screen presents the details of the stage 0



Details for Stage 1 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.1 s
Locality Level Summary: Node local: 1
Output Size / Records: 247.2 KiB / 14892
Shuffle Read Size / Records: 141.7 KiB / 17
Associated Job Ids: 0

▾ DAG Visualization

Stage 1
partitionBy
ShuffledRDD [4] [Unordered]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [5] [Unordered]
mapPartitions at PythonRDD.scala:160
PythonRDD [6] [Unordered]
RDD at PythonRDD.scala:53
map
MapPartitionsRDD [7] [Unordered]
map at NativeMethodAccessorImpl.java:0
saveAsTextFile
MapPartitionsRDD [8] [Unordered]
saveAsTextFile at NativeMethodAccessorImpl.java:0

▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 1 Completed Tasks

This screen presents the details of the stage 0

Extended word count

Here Iam modifying the basic word count program to ensure the code should be case insentive so we’re taking all the words to lower format, ignoring all the punctuations with the use of rejex, also included several stop words and making sure not to count those during the word count program, Now let’s have a look into the modifies program below.

```
from pyspark import SparkContext
import string

sc = SparkContext("local", "CountWord")


readf1 = sc.textFile("/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71913.txt")
readf2 = sc.textFile("/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS//pg71927.txt")


combinedfil = readf1.union(readf2)

stop_words = ["on", "or", "such","a", "about", "above", "after", "again", "against",
"all", "am", "an", "and", "any", "are", "aren't",
              "as", "at", "be", "because", "been", "before", "being", "below", "betwee
n", "both", "but",
              "by", "can't", "cannot", "could", "couldn't", "did", "didn't", "do", "do
es", "doesn't", "doing",
              "don't", "down", "during", "each", "few", "for", "from", "further", "ha
d", "hadn't", "has", "hasn't",
              "have", "haven't", "having", "he", "he'd", "he'll", "he's", "her", "her
e", "here's", "hers", "herself",
              "him", "himself", "his", "how", "how's", "i", "i'd", "i'll", "i'm", "i'v
e", "if", "in", "into", "is", "isn't",
              "it", "it's", "its", "itself", "let's", "me", "more", "most", "mustn't",
"my", "myself", "no", "nor", "not", "of",
              "off", "on", "once", "only", "or", "other", "ought", "our", "ours", "our
selves", "out", "over", "own", "same", "shan't",
              "she", "she'd", "she'll", "she's", "should", "shouldn't", "so", "some",
"such", "than", "that", "that's", "the", "their",
              "theirs", "them", "themselves", "then", "there", "there's", "these", "th
ey", "they'd", "they'll", "they're", "they've", "this",
              "those", "through", "to", "too", "under", "until", "up", "very", "was",
"wasn't", "we", "we'd", "we'll", "we're", "we've",
              "were", "weren't", "what", "what's", "when", "when's", "where", "wher
e's", "which", "while", "who", "who's", "whom",
              "why", "why's", "with", "won't", "would", "wouldn't", "you", "you'd", "y
ou'll", "you're", "you've", "your",
              "yours", "yourself", "yourselves"]


def remove_punctuation(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)

word_counts = combinedfil.flatMap(lambda line: remove_punctuation(line).lower().split
()) \
                         .map(lambda word: (word.strip(), 1)) \
                         .filter(lambda x: x[0] not in stop_words) \
                         .reduceByKey(lambda a, b: a + b) \
                         .sortBy(lambda x: x[1], ascending=False)
```

```
word_counts.saveAsTextFile("extended_wcount.txt")
```

Question 2

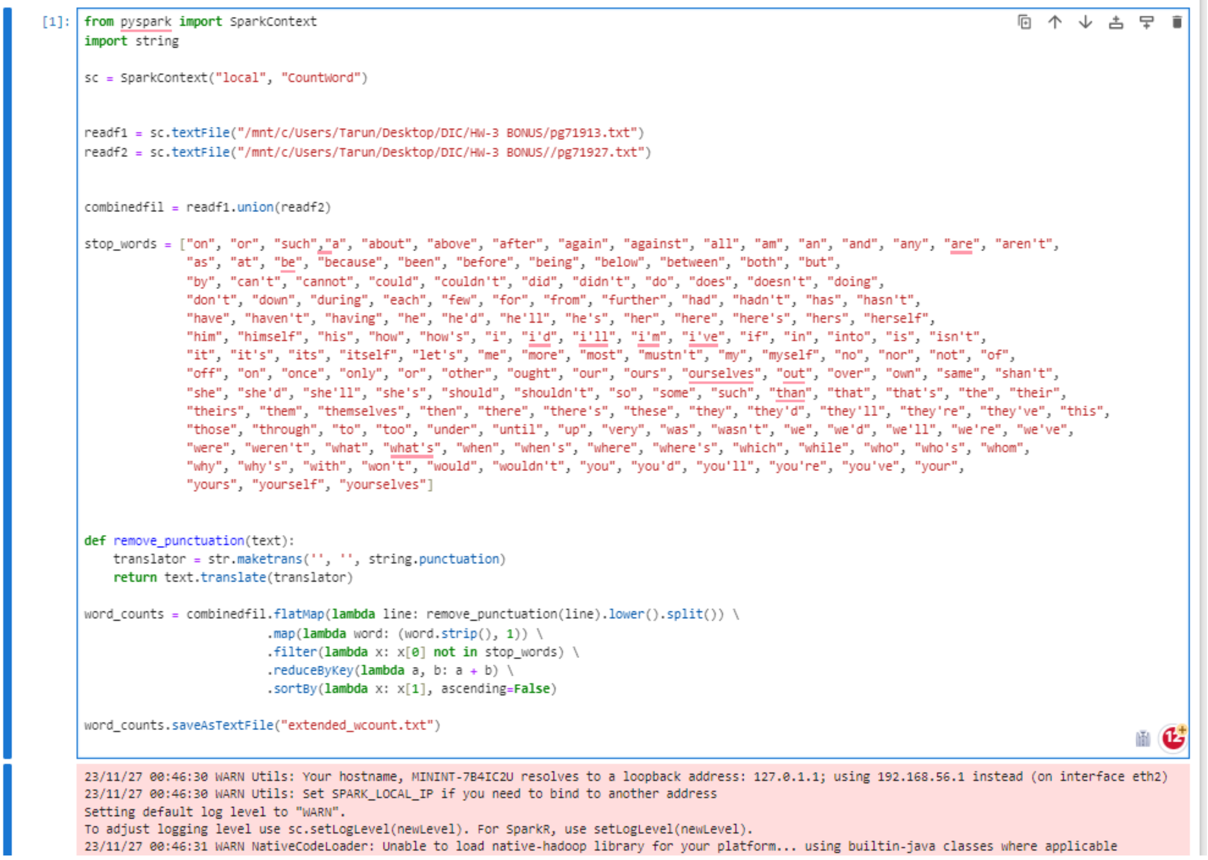
In the PySpark REPL/Jupyter notebook, run your word count extended program on all 2
text files.

1. **What are the 25 most common words? Include a screenshot of program output to**
   back-up your claim.
2. **How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim.**

Answer for part 1:

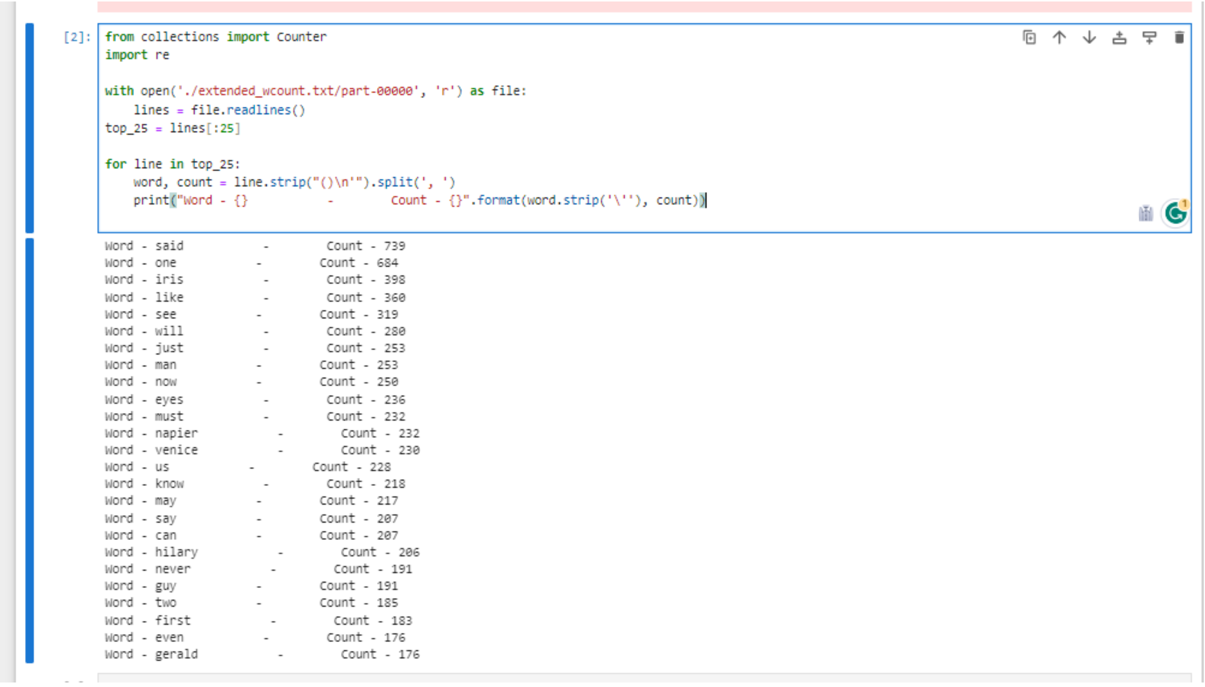
My Top 25 Words:

```
Word - said           -        Count - 739
Word - one           -        Count - 684
Word - iris           -        Count - 398
Word - like           -        Count - 360
Word - see           -        Count - 319
Word - will           -        Count - 280
Word - just           -        Count - 253
Word - man           -        Count - 253
Word - now           -        Count - 250
Word - eyes           -        Count - 236
Word - must           -        Count - 232
Word - napier           -        Count - 232
Word - venice           -        Count - 230
Word - us           -        Count - 228
Word - know           -        Count - 218
Word - may           -        Count - 217
Word - say           -        Count - 207
Word - can           -        Count - 207
Word - hilary           -        Count - 206
Word - never           -        Count - 191
Word - guy           -        Count - 191
Word - two           -        Count - 185
Word - first           -        Count - 183
Word - even           -        Count - 176
Word - gerald           -        Count - 176
```



```
[1]: from pyspark import SparkContext
import string

sc = SparkContext("local", "CountWord")


readf1 = sc.textFile("/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71913.txt")
readf2 = sc.textFile("/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS//pg71927.txt")


combinedfil = readf1.union(readf2)

stop_words = ["on", "or", "such","a", "about", "above", "after", "again", "against", "all", "am", "an", "and", "any", "are", "aren't",
              "as", "at", "be", "because", "been", "before", "being", "below", "between", "both", "but",
              "by", "can't", "cannot", "could", "couldn't", "did", "didn't", "do", "does", "doesn't", "doing",
              "don't", "down", "during", "each", "few", "for", "from", "further", "had", "hadn't", "has", "hasn't",
              "have", "haven't", "having", "he", "he'd", "he'll", "he's", "her", "here", "here's", "hers", "herself",
              "him", "himself", "his", "how", "how's", "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", "isn't",
              "it", "it's", "its", "itself", "let's", "me", "more", "most", "mustn't", "my", "myself", "no", "nor", "not", "of",
              "off", "on", "once", "only", "or", "other", "ought", "our", "ours", "ourselves", "out", "over", "own", "same", "shan't",
              "she", "she'd", "she'll", "she's", "should", "shouldn't", "so", "some", "such", "than", "that", "that's", "the", "their",
              "theirs", "them", "themselves", "then", "there", "there's", "these", "they", "they'd", "they'll", "they're", "they've", "this",
              "those", "through", "to", "too", "under", "until", "up", "very", "was", "wasn't", "we", "we'd", "we'll", "we're", "we've",
              "were", "weren't", "what", "what's", "when", "when's", "where", "where's", "which", "while", "who", "who's", "whom",
              "why", "why's", "with", "won't", "would", "wouldn't", "you", "you'd", "you'll", "you're", "you've", "your",
              "yours", "yourself", "yourselves"]


def remove_punctuation(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)

word_counts = combinedfil.flatMap(lambda line: remove_punctuation(line).lower().split()) \
                         .map(lambda word: (word.strip(), 1)) \
                         .filter(lambda x: x[0] not in stop_words) \
                         .reduceByKey(lambda a, b: a + b) \
                         .sortBy(lambda x: x[1], ascending=False)

word_counts.saveAsTextFile("extended_wcount.txt")
```

```
23/11/27 00:46:30 WARN Utils: Your hostname, MININT-7B4IC2U resolves to a loopback address: 127.0.1.1; using 192.168.56.1 instead (on interface eth2)
23/11/27 00:46:30 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/11/27 00:46:31 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```



```
[2]: from collections import Counter
import re

with open('./extended_wcount.txt/part-00000', 'r') as file:
    lines = file.readlines()
top_25 = lines[:25]

for line in top_25:
    word, count = line.strip("()\n'").split(', ')
    print("Word - {}           -        Count - {}".format(word.strip('\''), count))
```

```
Word - said           -        Count - 739
Word - one           -        Count - 684
Word - iris           -        Count - 398
Word - like           -        Count - 360
Word - see           -        Count - 319
Word - will           -        Count - 280
Word - just           -        Count - 253
Word - man           -        Count - 253
Word - now           -        Count - 250
Word - eyes           -        Count - 236
Word - must           -        Count - 232
Word - napier           -        Count - 232
Word - venice           -        Count - 230
Word - us           -        Count - 228
Word - know           -        Count - 218
Word - may           -        Count - 217
Word - say           -        Count - 207
Word - can           -        Count - 207
Word - hilary           -        Count - 206
Word - never           -        Count - 191
Word - guy           -        Count - 191
Word - two           -        Count - 185
Word - first           -        Count - 183
Word - even           -        Count - 176
Word - gerald           -        Count - 176
```

Answer for part 2:

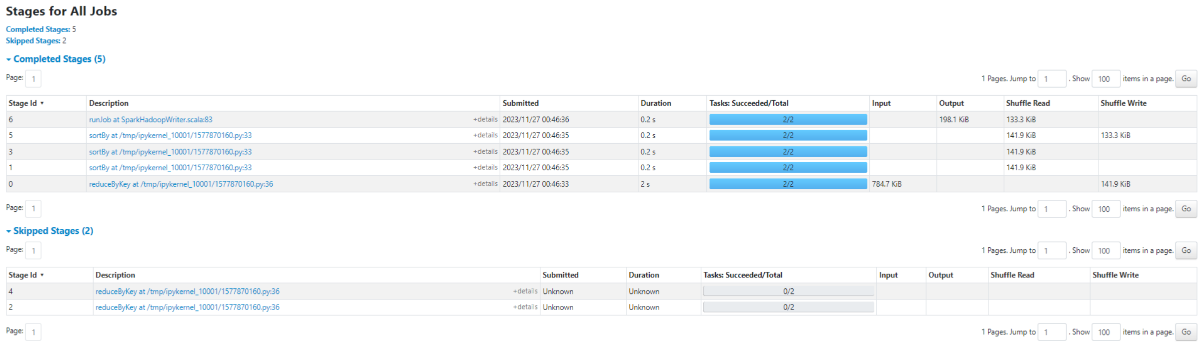
The execution is broken down into 7 stages, where operations are distributed, let’s have a deeper look into each stage.

1. **Stage 0:** THis is the initial stage where basci operations like reading two text files is being performed and after reading they both are being combined, which is union operation.
2. **Stage 1:** here operations involving partitioning and mapping.
3. **Stage 2:** skipped as already performed in teh stage 0.
4. **Stage 3:** sorting the data and also some mapping operations.
5. **Stage 4:** skipped
6. **Stage 5:** partitioning and mapping the data.
7. **Stage 6:** in this final stage the data gets mapped and the word count results to a text file.

Here as you can see each stage is processing a separate computation step like mapping, or partitioning etc, as in the each computation there is shuffle of the data occurs. SO whenever there's a cmputation spark braks down the execution into stages when there is shuffling of data across the network.

Visualization from Spark's WebUI

As you can see there are 7 stages in total and two are skipped.



Stages for All Jobs

Completed Stages: 5
Skipped Stages: 2

▾ Completed Stages (5)

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
6	runJob at SparkHadoopWriter.scala:83	+details	2023/11/27 00:46:36	0.2 s	2/2		198.1 KiB	133.3 KiB	
5	sortBy at /tmp/ipykernel_10001/1577870160.py:33	+details	2023/11/27 00:46:35	0.2 s	2/2			141.9 KiB	133.3 KiB
3	sortBy at /tmp/ipykernel_10001/1577870160.py:33	+details	2023/11/27 00:46:35	0.2 s	2/2			141.9 KiB	
1	sortBy at /tmp/ipykernel_10001/1577870160.py:33	+details	2023/11/27 00:46:35	0.2 s	2/2			141.9 KiB	
0	reduceByKey at /tmp/ipykernel_10001/1577870160.py:36	+details	2023/11/27 00:46:33	2 s	2/2	784.7 KiB			141.9 KiB

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

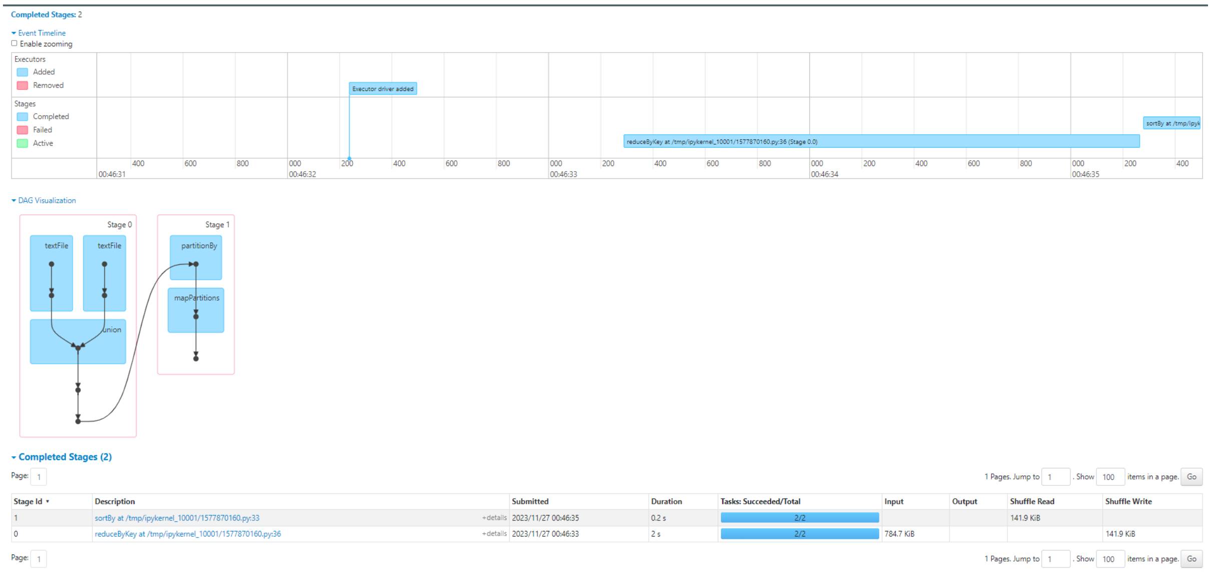
▾ Skipped Stages (2)

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
4	reduceByKey at /tmp/ipykernel_10001/1577870160.py:36	+details	Unknown	Unknown	0/2				
2	reduceByKey at /tmp/ipykernel_10001/1577870160.py:36	+details	Unknown	Unknown	0/2				

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

Let's have a deeper look into each stages.



Completed Stages: 2

▾ Event Timeline

□ Enable zooming

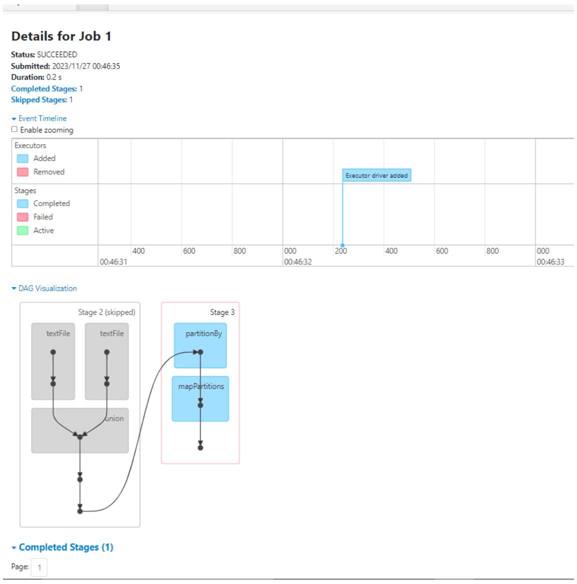
▾ DAG Visualization

▾ **Completed Stages (2)**

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	sortBy at /tmp/ipykernel_10001/1577870160.py:33 +details	2023/11/27 00:46:35	0.2 s	2/2			141.9 KiB	
0	reduceByKey at /tmp/ipykernel_10001/1577870160.py:36 +details	2023/11/27 00:46:33	2 s	2/2	784.7 KiB			141.9 KiB

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go



Details for Job 1

Status: SUCCEEDED
Submitted: 2023/11/27 00:46:35
Duration: 0.2 s
Completed Stages: 1
Skipped Stages: 1

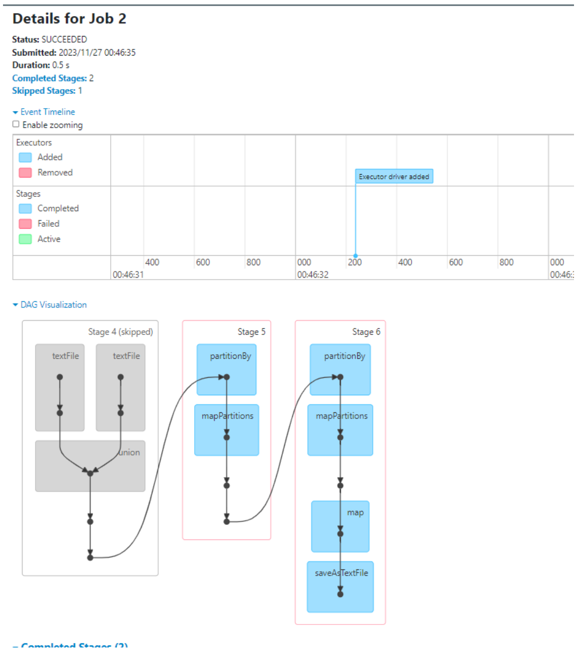
▾ Event Timeline

□ Enable zooming

▾ DAG Visualization

▾ **Completed Stages (1)**

Page: 1



Details for Job 2

Status: SUCCEEDED
Submitted: 2023/11/27 00:46:35
Duration: 0.5 s
Completed Stages: 2
Skipped Stages: 1

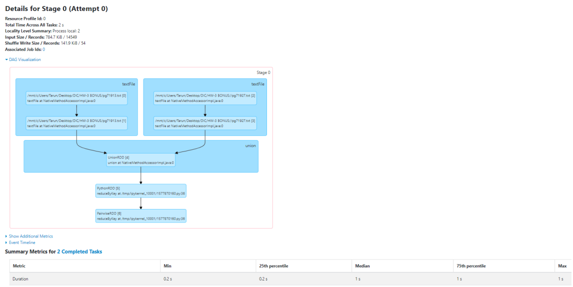
▾ Event Timeline

□ Enable zooming

▾ DAG Visualization

▾ **Completed Stages (2)**

For each stage



Details for Stage 0 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 2 s
Locality Level Summary: Process local: 2
Input Size / Records: 784.7 KiB / 14549
Shuffle Write Size / Records: 141.9 KiB / 54
Associated Job Ids: 0

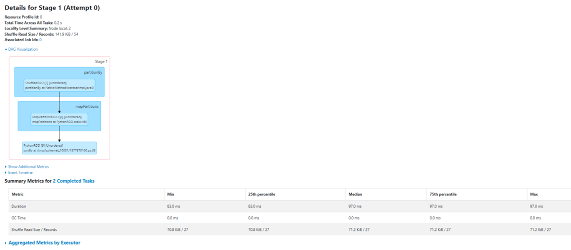
▾ DAG Visualization

Stage 0
textFile
/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71913.txt [0]
textFile at NativeMethodAccessorImpl.java:0
/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS/pg71913.txt [1]
textFile at NativeMethodAccessorImpl.java:0
textFile
/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS//pg71927.txt [2]
textFile at NativeMethodAccessorImpl.java:0
/mnt/c/Users/Tarun/Desktop/DIC/HW-3 BONUS//pg71927.txt [3]
textFile at NativeMethodAccessorImpl.java:0
union
UnionRDD [4]
union at NativeMethodAccessorImpl.java:0
PythonRDD [5]
reduceByKey at /tmp/ipykernel_10001/1577870160.py:36
PairwiseRDD [6]
reduceByKey at /tmp/ipykernel_10001/1577870160.py:36

▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.2 s	0.2 s	1 s	1 s	1 s



Details for Stage 1 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.2 s
Locality Level Summary: Node local: 2
Shuffle Read Size / Records: 141.9 KiB / 54
Associated Job Ids: 0

▾ DAG Visualization

Stage 1
partitionBy
ShuffledRDD [7] [Unordered]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [8] [Unordered]
mapPartitions at PythonRDD.scala:160
PythonRDD [9] [Unordered]
sortBy at /tmp/ipykernel_10001/1577870160.py:33

▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	83.0 ms	83.0 ms	97.0 ms	97.0 ms	97.0 ms
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Shuffle Read Size / Records	70.8 KiB / 27	70.8 KiB / 27	71.2 KiB / 27	71.2 KiB / 27	71.2 KiB / 27

▸ **Aggregated Metrics by Executor**

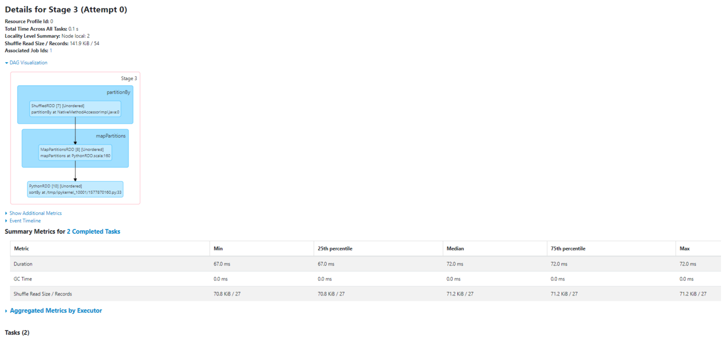
Details for Stage 2 (Attempt 0)

Summary Metrics

No tasks have started yet

Tasks

No tasks have started yet



Details for Stage 3 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.1 s
Locality Level Summary: Node local: 2
Shuffle Read Size / Records: 141.9 KiB / 54
Associated Job Ids: 1

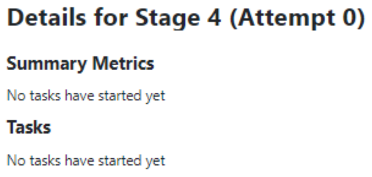
▾ DAG Visualization
Stage 3
partitionBy
ShuffledRDD [7] [Unordered]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [8] [Unordered]
mapPartitions at PythonRDD.scala:160
PythonRDD [10] [Unordered]
sortBy at /tmp/ipykernel_10001/1577870160.py:33
▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	67.0 ms	67.0 ms	72.0 ms	72.0 ms	72.0 ms
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Shuffle Read Size / Records	70.8 KiB / 27	70.8 KiB / 27	71.2 KiB / 27	71.2 KiB / 27	71.2 KiB / 27

▸ **Aggregated Metrics by Executor**

Tasks (2)



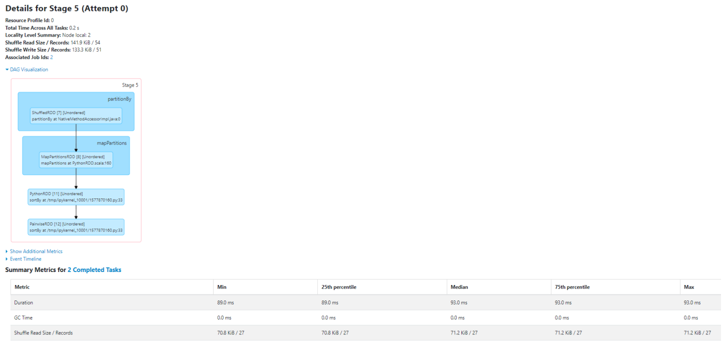
Details for Stage 4 (Attempt 0)

Summary Metrics

No tasks have started yet

Tasks

No tasks have started yet



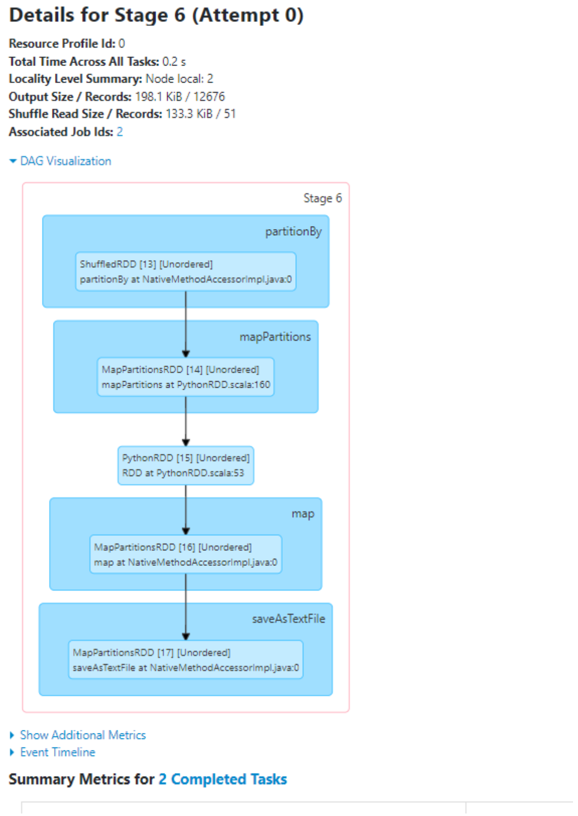
Details for Stage 5 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.2 s
Locality Level Summary: Node local: 2
Shuffle Read Size / Records: 141.9 KiB / 54
Shuffle Write Size / Records: 133.3 KiB / 51
Associated Job Ids: 2

▾ DAG Visualization
Stage 5
partitionBy
ShuffledRDD [7] [Unordered]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [8] [Unordered]
mapPartitions at PythonRDD.scala:160
PythonRDD [11] [Unordered]
sortBy at /tmp/ipykernel_10001/1577870160.py:33
PairwiseRDD [12] [Unordered]
sortBy at /tmp/ipykernel_10001/1577870160.py:33
▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	89.0 ms	89.0 ms	93.0 ms	93.0 ms	93.0 ms
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Shuffle Read Size / Records	70.8 KiB / 27	70.8 KiB / 27	71.2 KiB / 27	71.2 KiB / 27	71.2 KiB / 27



Details for Stage 6 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.2 s
Locality Level Summary: Node local: 2
Output Size / Records: 198.1 KiB / 12676
Shuffle Read Size / Records: 133.3 KiB / 51
Associated Job Ids: 2

▾ DAG Visualization
Stage 6
partitionBy
ShuffledRDD [13] [Unordered]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [14] [Unordered]
mapPartitions at PythonRDD.scala:160
PythonRDD [15] [Unordered]
RDD at PythonRDD.scala:53
map
MapPartitionsRDD [16] [Unordered]
map at NativeMethodAccessorImpl.java:0
saveAsTextFile
MapPartitionsRDD [17] [Unordered]
saveAsTextFile at NativeMethodAccessorImpl.java:0
▸ Show Additional Metrics
▸ Event Timeline

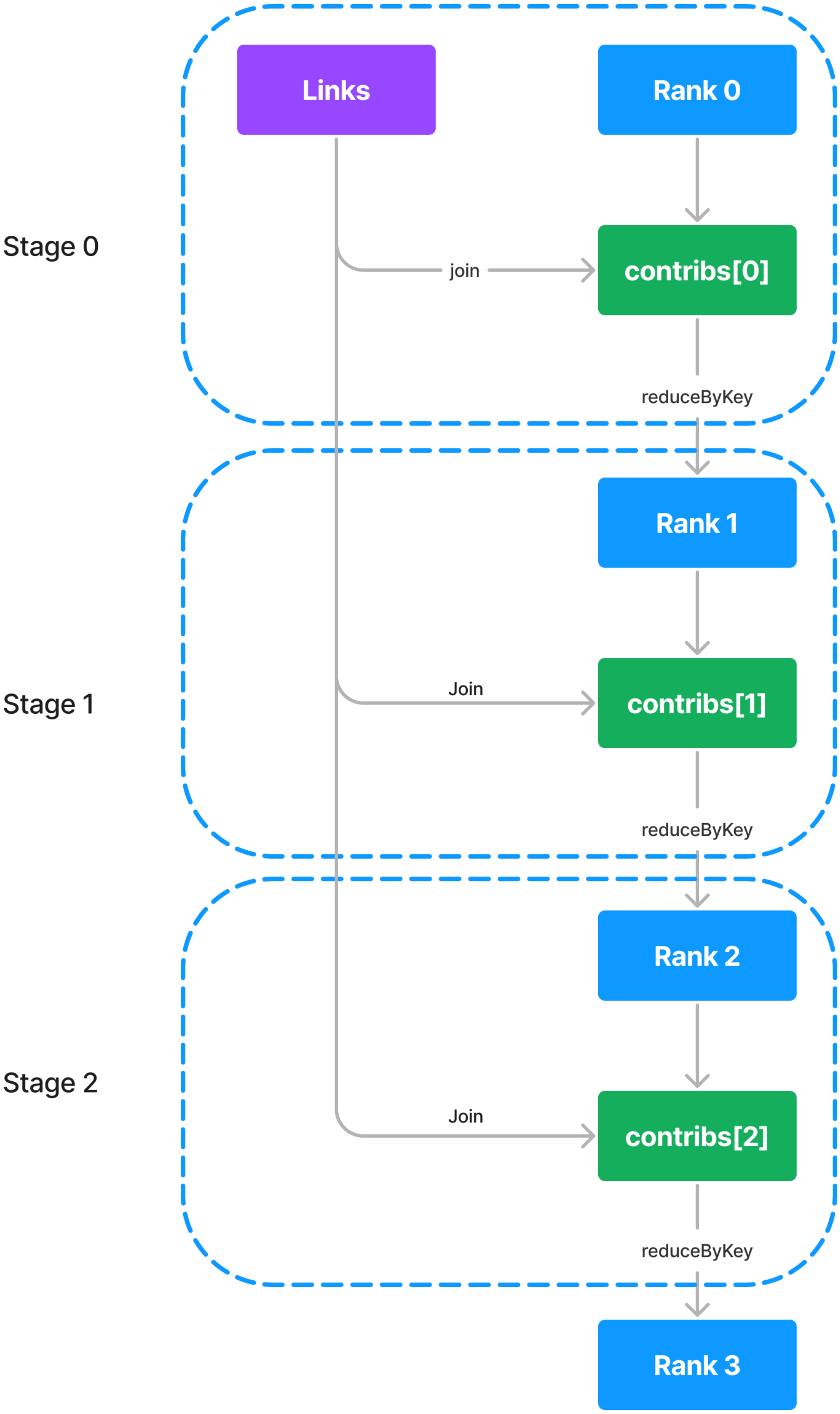
Summary Metrics for 2 Completed Tasks

Question 3

Given the above spark application, draw the lineage graph DAG for the RDD ranks on
line 12 when the iteration variable i has a value of 2. Include nodes for all intermediate
RDDs, even if they are unnamed.

```
lines = sc.textFile(file)
links = lines.map(lambda urls: parseNeighbors(urls)) \
 .groupByKey()
 .cache()
N = links.count()
ranks = links.map(lambda u: (u[0], 1.0/N))

for i in range(iters):
 contribs = links.join(ranks) \
 .flatMap(lambda u: computeContribs(u[1][0], u[1][1]))
 ranks = contribs.reduceByKey(lambda a,b: a+b) \
 .mapValues(lambda rank: rank * 0.85 + 0.15*(1.0/N))
return ranks
```

Answer:


Links
Rank 0
Stage 0
join
contribs[0]
reduceByKey
Rank 1
Stage 1
Join
contribs[1]
reduceByKey
Rank 2
Stage 2
Join
contribs[2]
reduceByKey
Rank 3

Ref: ppt