Lab 3 - Assignment

MM16B023[1]

[a]*Indian Institute of Technology Madras*

Keyword: VM, Cloud Functions, DataFlow, pipeline, Pcollections

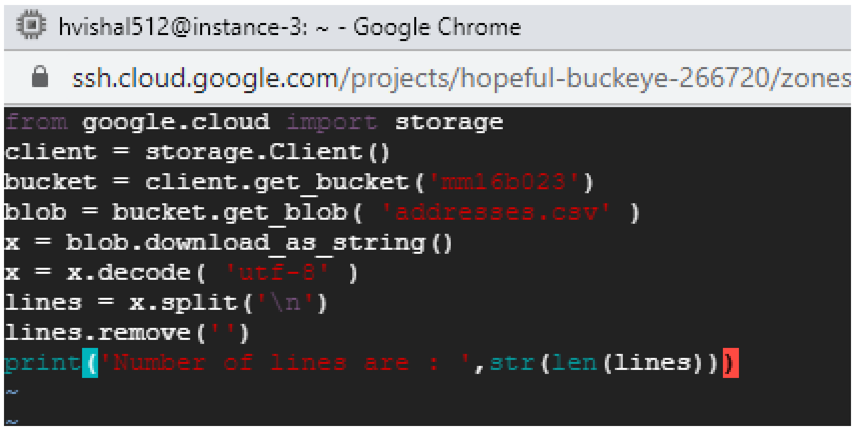
Abstract: This paper presents the solutions to first assignment of the Big Data Laboratory course (CS4830) at *IIT Madras*. All the notations used are as according with the textbook Mining of massive data sets by Anand Rajaraman

Problem 1

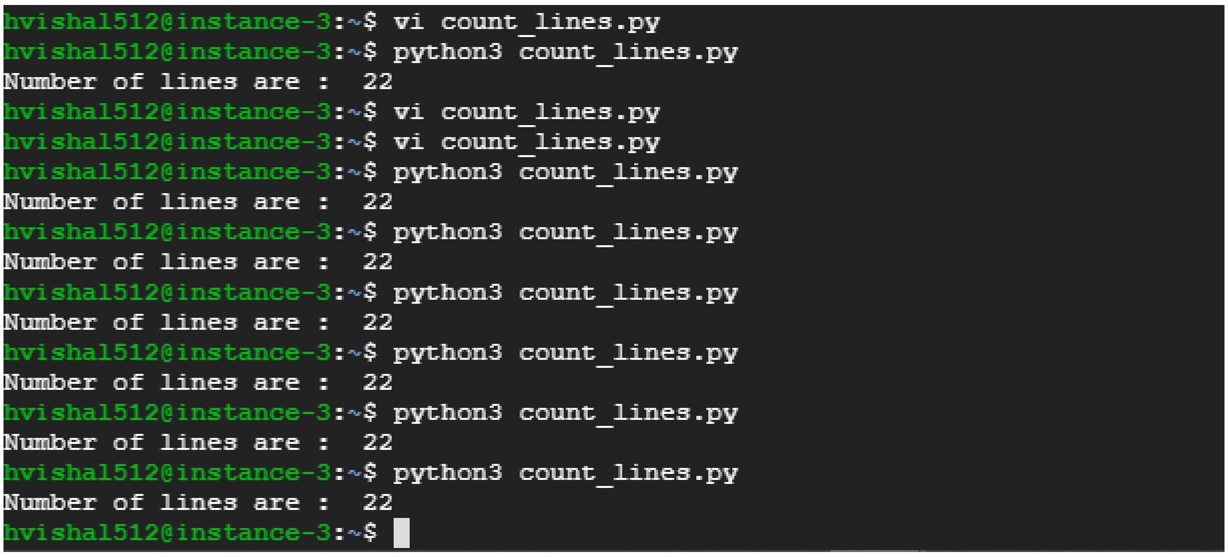
Provide screenshot for logs of all the 3 tasks containing the required result - line count using VM, Cloud Functions and DataFlow, along with the python file for each task.

Solution:

a) Virtual Machine



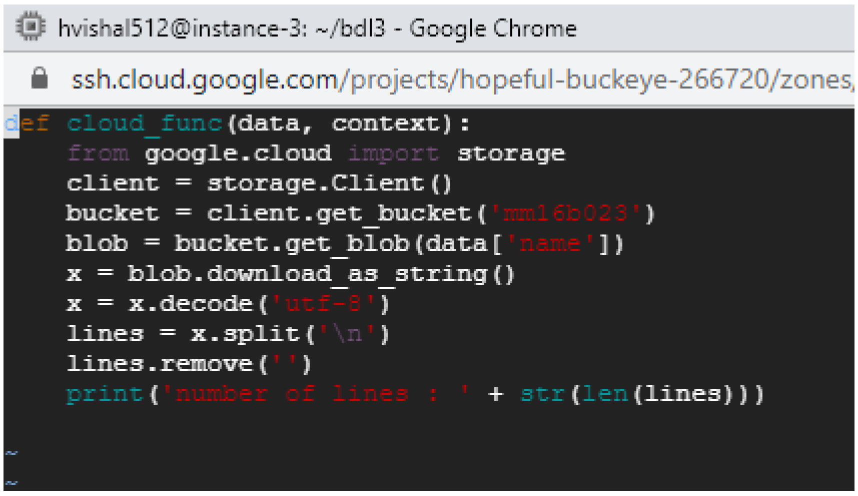
```
hvishal512@instance-3: ~ - Google Chrome
ssh.cloud.google.com/projects/hopeful-buckeye-266720/zones
from google.cloud import storage
client = storage.Client()
bucket = client.get_bucket('mm16b023')
blob = bucket.get_blob( 'addresses.csv' )
x = blob.download_as_string()
x = x.decode( 'utf-8' )
lines = x.split('\n')
lines.remove('')
print('Number of lines are : ',str(len(lines)))
~
~
```



```
hvishal512@instance-3:~$ vi count_lines.py
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ vi count_lines.py
hvishal512@instance-3:~$ vi count_lines.py
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ python3 count_lines.py
Number of lines are :  22
hvishal512@instance-3:~$ 
```

Fig1: Counting number of lines using Virtual Machine

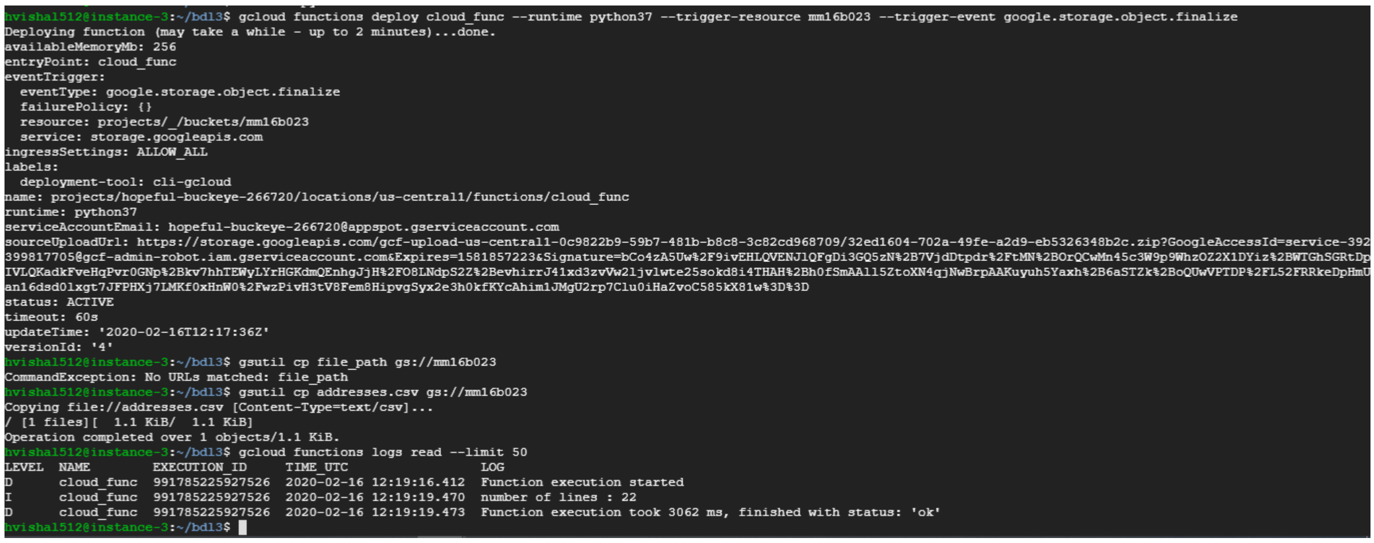
b) Cloud Function



hvishal512@instance-3: ~/bdl3 - Google Chrome

ssh.cloud.google.com/projects/hopeful-buckeye-266720/zones,

```python
def cloud_func(data, context):
    from google.cloud import storage
    client = storage.Client()
    bucket = client.get_bucket('mm16b023')
    blob = bucket.get_blob(data['name'])
    x = blob.download_as_string()
    x = x.decode('utf-8')
    lines = x.split('\n')
    lines.remove('')
    print('number of lines : ' + str(len(lines)))
```

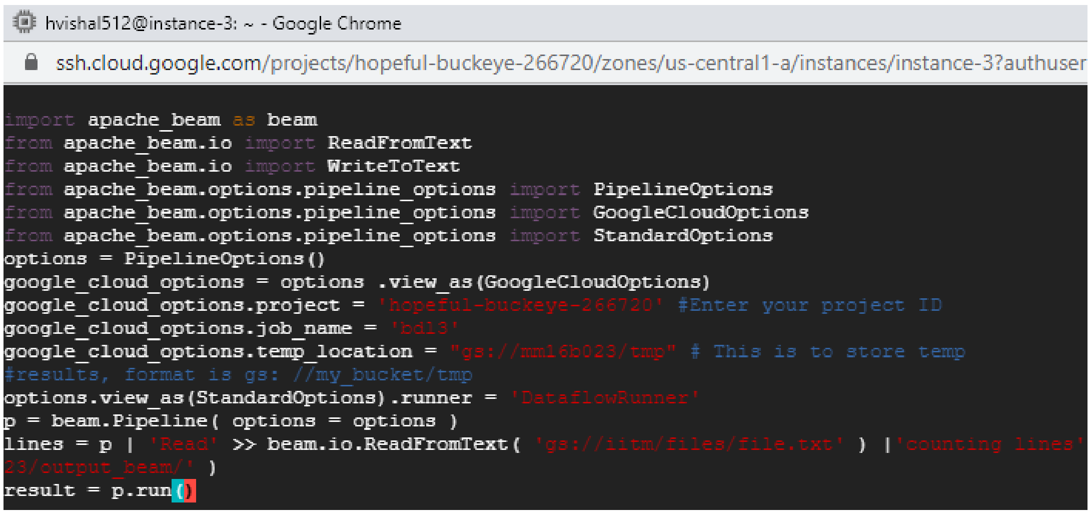


```
hvishal512@instance-3:~/bdl3$ gcloud functions deploy cloud_func --runtime python37 --trigger-resource mm16b023 --trigger-event google.storage.object.finalize
Deploying function (may take a while - up to 2 minutes)...done.
availableMemoryMb: 256
entryPoint: cloud_func
eventTrigger:
  eventType: google.storage.object.finalize
  failurePolicy: {}
  resource: projects/_/buckets/mm16b023
  service: storage.googleapis.com
ingressSettings: ALLOW_ALL
labels:
  deployment-tool: cli-gcloud
name: projects/hopeful-buckeye-266720/locations/us-central1/functions/cloud_func
runtime: python37
serviceAccountEmail: hopeful-buckeye-266720@appspot.gserviceaccount.com
sourceUploadUrl: https://storage.googleapis.com/gcf-upload-us-central1-0c9822b9-59b7-481b-b8c8-3c82cd968709/32ed1604-702a-49fe-a2d9-eb5326348b2c.zip?GoogleAccessId=service-392
399817705@gcf-admin-robot.iam.gserviceaccount.com&Expires=1581857223&Signature=bCo4zA5Uw%2F9ivEHLQVENJlQFgDi3GQ5zN%2B7VjdDtpdr%2FtMN%2BOrQCwMn45c3W9p9WhzOZ2X1DYiz%2BWTGhSGRtDp
IVLQKadkFveHqPvr0GNp%2Bkv7hhTEWyLYrHGKdmQEnhgJjH%2FO8LNdpS2Z%2BevhirrJ41xd3zvVw2ljvlwte25sokd8i4THAH%2Bh0fSmAAl15ZtoXN4qjNwBrpAAKuyuh5Yaxh%2B6aSTZk%2BoQUwVPTDP%2FL52FRRkeDpHmU
an16dsd0lxgt7JFPHXj7LMKf0xHnW0%2FwzPivH3tV8Fem8HipvgSyx2e3h0kfKYcAhim1JMgU2rp7Clu0iHaZvoC585kX81w%3D%3D
status: ACTIVE
timeout: 60s
updateTime: '2020-02-16T12:17:36Z'
versionId: '4'
hvishal512@instance-3:~/bdl3$ gsutil cp file_path gs://mm16b023
CommandException: No URLs matched: file_path
hvishal512@instance-3:~/bdl3$ gsutil cp addresses.csv gs://mm16b023
Copying file://addresses.csv [Content-Type=text/csv]...
/ [1 files][  1.1 KiB/  1.1 KiB]
Operation completed over 1 objects/1.1 KiB.
hvishal512@instance-3:~/bdl3$ gcloud functions logs read --limit 50
LEVEL  NAME        EXECUTION_ID     TIME_UTC                 LOG
D      cloud_func  991785225927526  2020-02-16 12:19:16.412  Function execution started
I      cloud_func  991785225927526  2020-02-16 12:19:19.470  number of lines : 22
D      cloud_func  991785225927526  2020-02-16 12:19:19.473  Function execution took 3062 ms, finished with status: 'ok'
hvishal512@instance-3:~/bdl3$
```

Fig 2: Counting the number of lines using Cloud function

Using the above two methods, number of lines in the file 'addresses.csv' = 22

c) Data flow

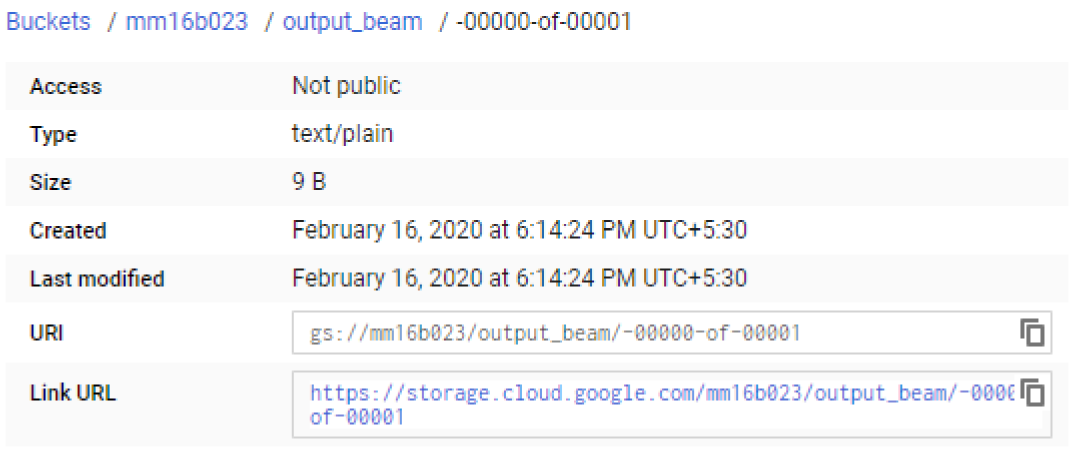


hvishal512@instance-3: ~ - Google Chrome

ssh.cloud.google.com/projects/hopeful-buckeye-266720/zones/us-central1-a/instances/instance-3?authuser

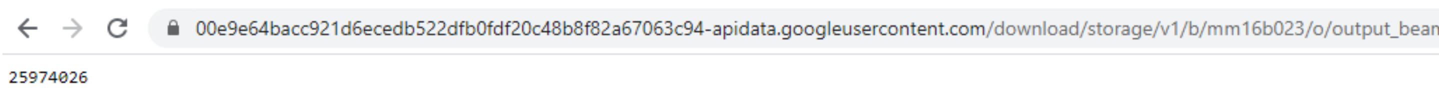
```python
import apache_beam as beam
from apache_beam.io import ReadFromText
from apache_beam.io import WriteToText
from apache_beam.options.pipeline_options import PipelineOptions
from apache_beam.options.pipeline_options import GoogleCloudOptions
from apache_beam.options.pipeline_options import StandardOptions
options = PipelineOptions()
google_cloud_options = options .view_as(GoogleCloudOptions)
google_cloud_options.project = 'hopeful-buckeye-266720' #Enter your project ID
google_cloud_options.job_name = 'bdl3'
google_cloud_options.temp_location = "gs://mm16b023/tmp" # This is to store temp
#results, format is gs: //my_bucket/tmp
options.view_as(StandardOptions).runner = 'DataflowRunner'
p = beam.Pipeline( options = options )
lines = p | 'Read' >> beam.io.ReadFromText( 'gs://iitm/files/file.txt' ) |'counting lines'
23/output_beam/' )
result = p.run()
```

Note: Some portion of the image has been cropped out for aesthetics. Please refer to the code attached.



Buckets / mm16b023 / output_beam / -00000-of-00001

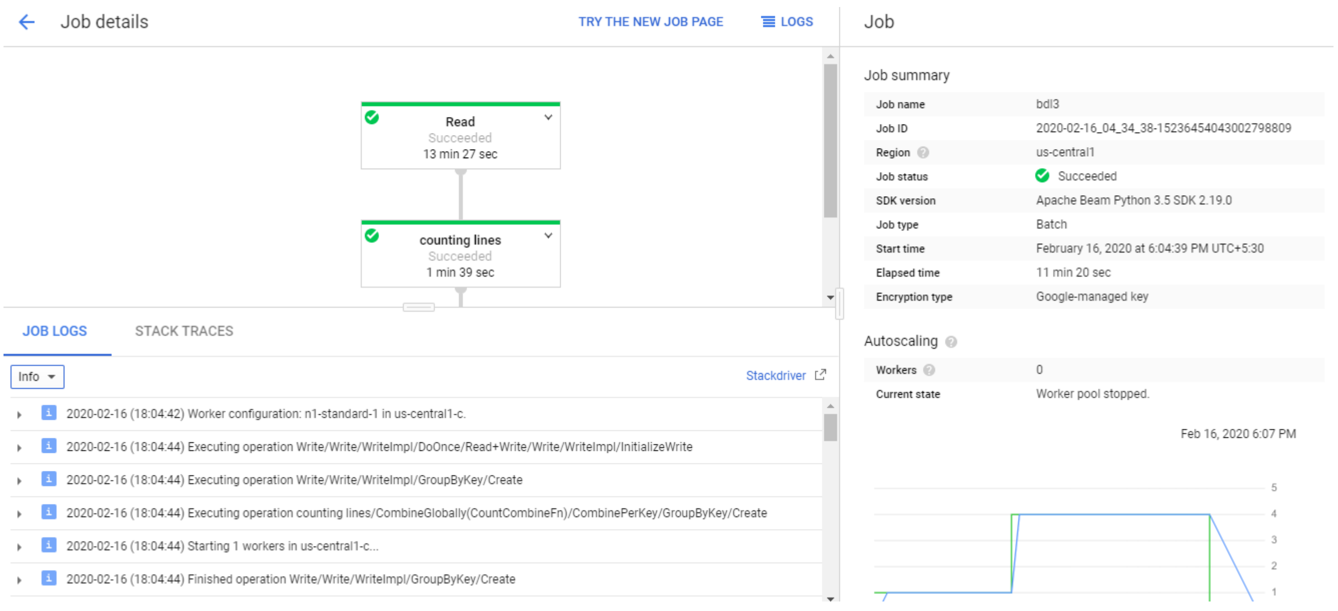
Access	Not public
Type	text/plain
Size	9 B
Created	February 16, 2020 at 6:14:24 PM UTC+5:30
Last modified	February 16, 2020 at 6:14:24 PM UTC+5:30
URI	gs://mm16b023/output_beam/-00000-of-00001
Link URL	https://storage.cloud.google.com/mm16b023/output_beam/-0000 of-00001



00e9e64bacc921d6ecedb522dfb0fdf20c48b8f82a67063c94-apidata.googleusercontent.com/download/storage/v1/b/mm16b023/o/output_bean

```
25974026
```

Logs



```
2020-02-16 (18:04:42) Worker configuration: n1-standard-1 in us-central1-c.
2020-02-16 (18:04:44) Executing operation Write/Write/WriteImpl/DoOnce/Read+Write/Write/WriteImpl/InitializeWrite
2020-02-16 (18:04:44) Executing operation Write/Write/WriteImpl/GroupByKey/Create
2020-02-16 (18:04:44) Executing operation counting lines/CombineGlobally(CountCombineFn)/CombinePerKey/GroupByKey/Create
2020-02-16 (18:04:44) Starting 1 workers in us-central1-c...
2020-02-16 (18:04:44) Finished operation Write/Write/WriteImpl/GroupByKey/Create
```

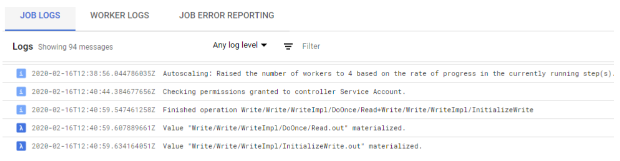
Job

Job summary

Job name	bdl3
Job ID	2020-02-16_04_34_38-15236454043002798809
Region	us-central1
Job status	✓ Succeeded
SDK version	Apache Beam Python 3.5 SDK 2.19.0
Job type	Batch
Start time	February 16, 2020 at 6:04:39 PM UTC+5:30
Elapsed time	11 min 20 sec
Encryption type	Google-managed key

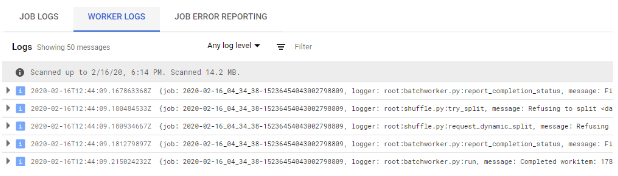
Autoscaling

Workers	0
Current state	Worker pool stopped.



Logs Showing 94 messages

```
2020-02-16T12:38:56.044786035Z  Autoscaling: Raised the number of workers to 4 based on the rate of progress in the currently running step(s).
2020-02-16T12:40:44.384677656Z  Checking permissions granted to controller Service Account.
2020-02-16T12:40:59.547461258Z  Finished operation Write/Write/WriteImpl/DoOnce/Read+Write/Write/WriteImpl/InitializeWrite
2020-02-16T12:40:59.607889661Z  Value "Write/Write/WriteImpl/DoOnce/Read.out" materialized.
2020-02-16T12:40:59.634164051Z  Value "Write/Write/WriteImpl/InitializeWrite.out" materialized.
```



Logs Showing 50 messages

```
Scanned up to 2/16/20, 6:14 PM. Scanned 14.2 MB.
2020-02-16T12:44:09.167863368Z  {job: 2020-02-16_04_34_38-15236454043002798809, logger: root:batchworker.py:report_completion_status, message: Fi
2020-02-16T12:44:09.180484533Z  {job: 2020-02-16_04_34_38-15236454043002798809, logger: root:shuffle.py:try_split, message: Refusing to split <da
2020-02-16T12:44:09.180934667Z  {job: 2020-02-16_04_34_38-15236454043002798809, logger: root:shuffle.py:request_dynamic_split, message: Refusing
2020-02-16T12:44:09.181279897Z  {job: 2020-02-16_04_34_38-15236454043002798809, logger: root:batchworker.py:report_completion_status, message: Fi
2020-02-16T12:44:09.215024232Z  {job: 2020-02-16_04_34_38-15236454043002798809, logger: root:batchworker.py:run, message: Completed workitem: 178
```

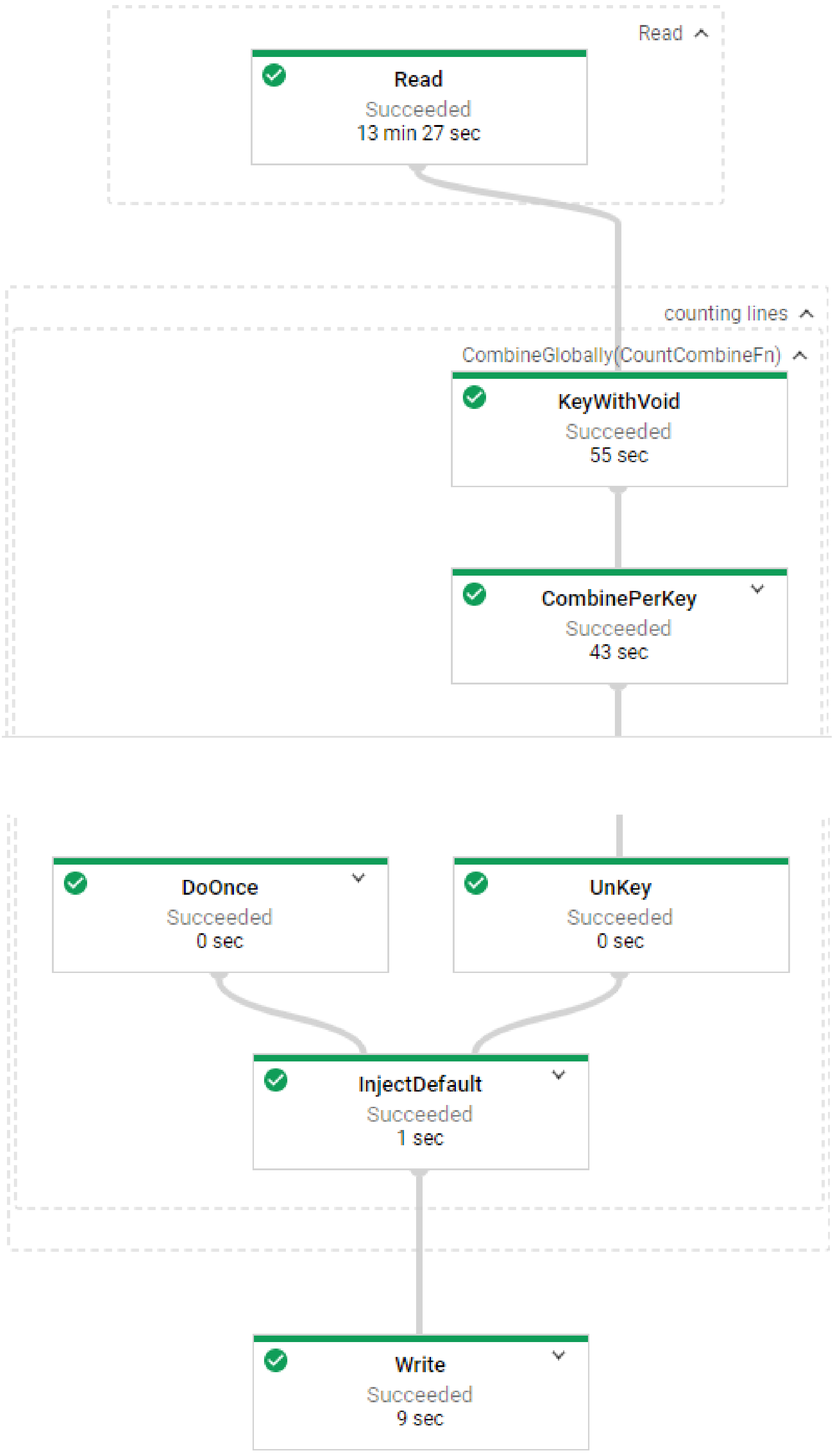
Fig3 : Counting the number of lines using DataFlow

$$Number\ of\ lines\ in\ the\ data\ set = 25974026$$

Problem2

Provide the screenshot for the execution graph created by Dataflow in the background for the pipeline object created in task 3

Solution:



Problem 3

Explain the pipeline used in task 3. What issues did you face while trying to make the code work for task 3 and how did you resolve them?

Solution: The pipeline used has the following transforms embedded in it

- **ReadFromText** - Navigates through the file reading each line of it
- **CombineGlobally** - Stores each line as a string into a bigger dataset called PCollections
- **WriteToText** - Writes the output (no. of lines in this case) to the specified directory

Issues faced

- There was an exception thrown in the worker code when multiple lines of the code were joined by a vertical line. Surprisingly, no error came when it was written as a big single line.
- Not an issue per se, but picking the appropriate transforms and designing the right pipeline structure took some time as there are many functions and hence combinations possible.

Problem4

PCollections can handle unbounded data. What is meant by unbounded data and how do you think Pcollections can handle it? (Hint: Think on the line of triggers)

Solution:

- Data streamed from a constantly updating source is known as unbounded data. Since the data source continuously adds new elements, it has unlimited size in some sense and hence the name unbounded data.
- PCollections can handle it by modifying triggers in such a way that it maps to each collection in a streaming pipeline. This can be done using Apache Beam SDK which uses windowing (segregate unbounded data into windows) to operate the data on a combination of event time (indicated by time stamps), time of processing and the number of data elements. This modified trigger mechanism outputs whenever the watermark passes the edge of the window.
