Introduction:

This file serves as a guide on how to run/replicate the map-reduce program with all the features which have been performed using gcloud faas, and replicating all the things on how to test and run everything from scratch.

This file acts as a report (answering the questions asked), contains the examples as well as the cases through examples.

Submission Requirements:

Bash script file (just contains the bash code) – bash_final.bash
Test cases – attached in the file below.
Main code – map_final.py, reduce_final.py, master.py, main_final.py, trigger.py, index.html
GCP bucket ssh key - prashul-kumar-fall2023-23ecf366ca39.json

File Structure –

prashul-kumar-fall2023-23ecf366ca39.json – Should place this in praskumainstance2 (my already used VM for all the assignments) This file is mandatory and serves as the connect (has private key)

Architecture –

The system is designed for distributed processing of text data stored in Google Cloud Storage. The workflow involves triggering cloud functions in response to changes in the Cloud Storage bucket. The architecture consists of multiple components, including functions for mapping, reducing, and a master function coordinating the overall workflow.
Note: For my case I am hardcoding as of now 3 mappers and 3 reducers for each run.

Components:

1.) **Master Cloud Function (master.py):**

Orchestrates the entire workflow by dividing tasks among mapping and reducing components. Coordinates synchronization using threading barriers.

Design: Utilizes threading for parallelizing tasks. Implements barriers to manage synchronization between mapping and reducing tasks.

Flow: List all files in the bucket with a specified prefix. Generate map payloads for each source file, including the number of mappers. Flatten the map payloads to cover multiple mappers for each source file. Invoke map functions in parallel using threading for each flattened map payload. Wait for all map operations to complete using a barrier. Print a message indicating the completion of all map operations. Start reduce operations by generating reduce payloads for each reducer. Invoke reduce functions in parallel using threading for each reducer. Wait for all reduce operations to complete using a barrier. This function writes 3 separate files per each reducer. Later the 3 files are combined since the end goal is that the user may enter any query word and it should display in what documents each word's count is how much. Since not all reduced files will have all words hence instead of iterating through each file when user inputs query, I combined the reducer output in a single combined.json Upload the combined results to the output bucket.

Input:
Cloud Storage bucket name.
File name prefix.
Output bucket name.
Number of mappers.
Number of reducers.

Output:
Completion message or an error message if no files are found.

Function Trigger: HTTP request or Cloud Storage event trigger.

2.) Mapping Cloud Function (map.py):

Counts the occurrences of words in a given text chunk and create a word mapping.

The function tokenizes the input text into words using a regular expression. Divide the words into chunks for parallel processing based on the number of mappers. For each word in the chunk, create or update a word mapping dictionary.

The dictionary structure: {word: {source_name: [counts]}}.
This structure is maintained since the end goal is we want to know which word appears in which document in total how many times, thus to maintain the lineage we used a nested structure.
Return the word mapping and mapper name.

Overall Workflow: The Cloud Function is triggered by an HTTP request. For each file in the request, divide the content into chunks. Parallelize the processing of chunks using mappers.

Mapper Processing: Each mapper creates word mappings for its assigned chunk. Intermediate results are saved in temporary files. Upload temporary files to Cloud Storage. Elapsed time and processing messages are aggregated and returned. Use ThreadPoolExecutor to parallelize the processing of multiple files. Wait for all tasks to complete. Return processing results.

Input:
args: Tuple containing file name, output bucket, mapper names, and total number of mappers.
Output:
List of messages indicating successful processing by each mapper, including elapsed time.
Map Word Count Cloud Function:

3.) **Reduce Word Count function (reduce.py):**

ASCII-based Hash Function: Compute a hash value for a given word based on the sum of ASCII values of its characters.

Input:
word: Word to be hashed.
num_reducers: Number of reducer tasks.
Output:
Hash value indicating the assigned reducer index.

Note: Earlier simple hashing was used, but an issue is identified with simple hashing. Since here we are invoking the reduce function based on the no of reducers (3 times), i.e., conceptually the reducer is running parallelly in 3 separate containers. So the hash value of the same word changes on running the hash (fn) multiple times on different computers. Tried this on local system by running hash('abc') and then restarted kernel and again ran hash('abc'). It was resulting in different numbers. Thus it was resulting in some words missing or repeating set of words. As a result ASCII based hashing was used for consistent results.

Reduce Word Count Function: Reduce word counts by aggregating results from multiple mappers based on ASCII-based hash values. Retrieve input parameters from the HTTP request payload. Initialize a dictionary to store the reduced results. Iterate through intermediate results stored in Cloud Storage. For each word, use the ASCII-based hash to determine the assigned reducer. If the reducer index matches the current reducer, aggregate counts for the word. Upload the final reduced 3 files results to Cloud Storage.

4.) Streaming Cloud Function (trigger.py):

Triggered by a change to a Cloud Storage bucket, specifically for new text files with names starting with 'book' and ending with '.txt'.

Extract bucket name and file name from the trigger data. Check if the file name follows the expected pattern. If the pattern matches, initiate a master function to process the new file. Upon successful triggering of the master function, append the results to the combined results file in the bucket.

Input: Trigger data containing information about the bucket and the file that triggered the function.
Output: Log messages indicating the success or failure of triggering the master function and appending to the combined results.

Append to Combined Results Function: Append results from individual reducer outputs to a combined results file. Upload the combined results to a file named 'combined_results.json' in the bucket.

Overall Workflow: The function is triggered by a change in the specified Cloud Storage bucket. Extract bucket name and file name from the trigger data. Check if the file name follows the expected pattern. If the pattern matches, trigger the master function to process the new file. Upon successful processing by the master function, append the results to the combined results file.

5.) Search Documents Function (main.py):

Retrieve documents based on a search query from a combined results file.

Steps: Extract the search query from the request parameters. Validate that a valid search query is provided. Retrieve the combined results file from the specified Cloud Storage bucket. Search for documents based on the provided query in the combined results. Return the search results as a JSON response.

Input: Search query provided as a parameter in the request.
Output: JSON response containing the search results or an error message.

Overall Workflow: Extract the search query from the user input in the front end in the text box. Retrieve the combined results file from the specified Cloud Storage bucket. Search for documents based on the provided query in the combined results. Return the search results as a URL clicking which will show the results.

Initial Setup steps/design –

First of all, the VM's created during previous assignments were only reused. Hence, just needed to start the VM's first using –

gcloud compute instances start praskumainstance2

Now the files have to be uploaded to the VM instance using –

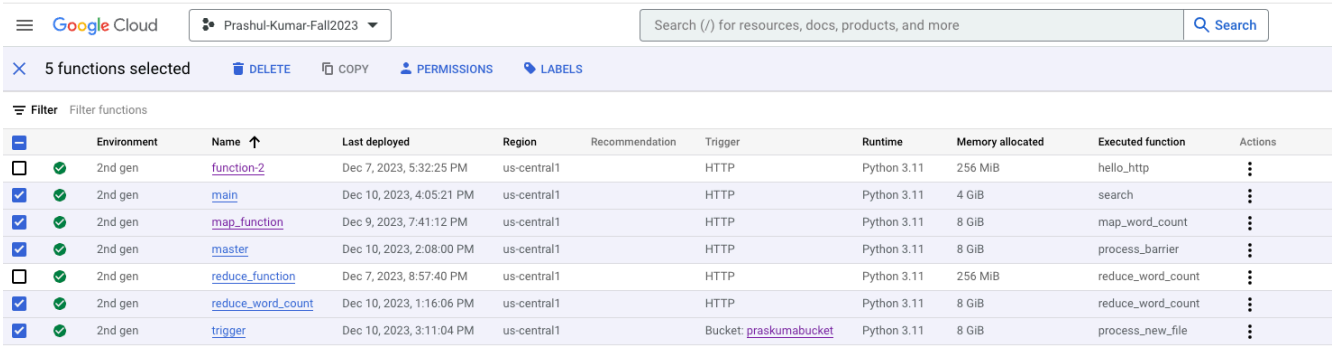
gcloud compute scp /Users/prashulkumar/Documents/SEM-3/ECC/Assignment-4/prashul-kumar-fall2023-23ecf366ca39.json praskumainstance2

Then authorize to write/ modify the praskumabucket my local VM using –

export GOOGLE_APPLICATION_CREDENTIALS=~/prashul-kumar-fall2023-23ecf366ca39.json

Now initial setup is ready for VM. Next we create all the functions in gcloud function and deploy them –

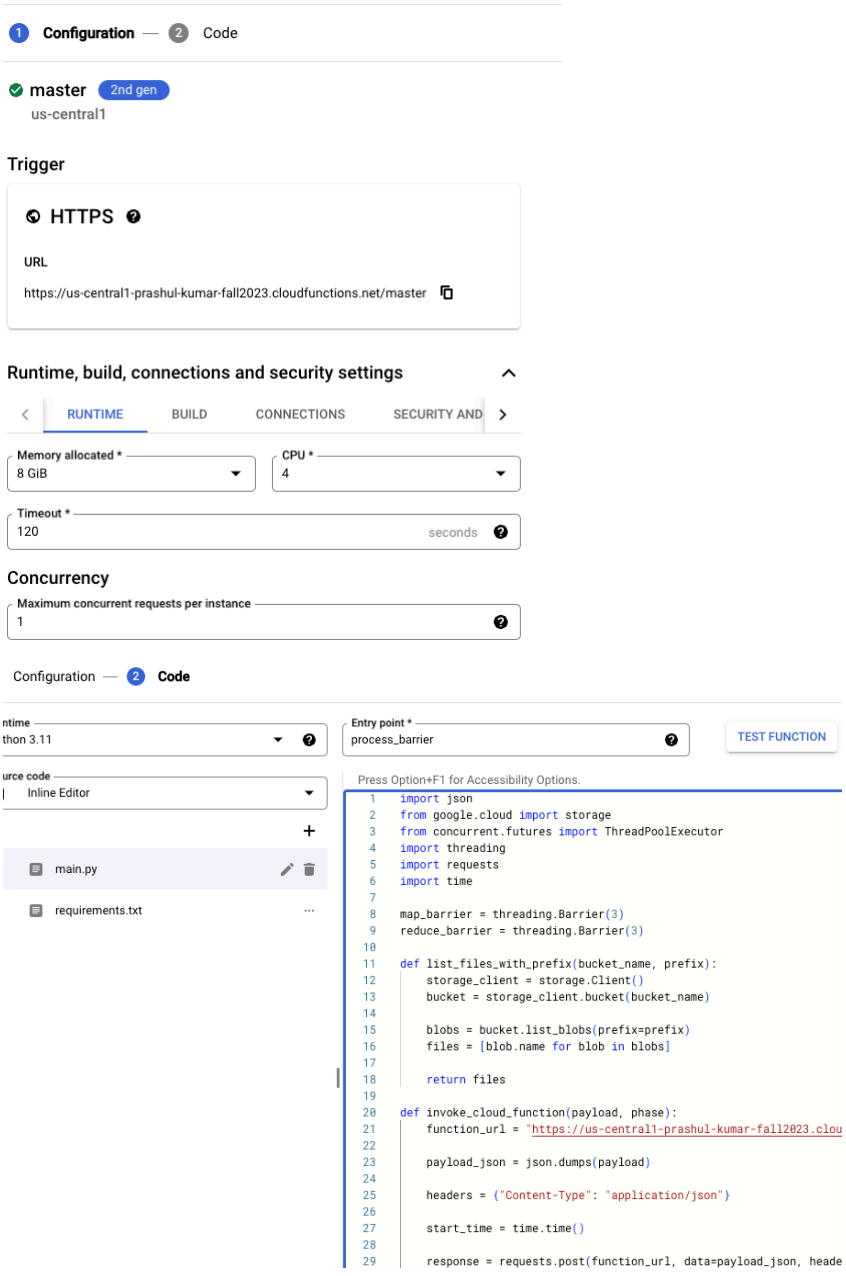
This was done using the UI of google cloud –



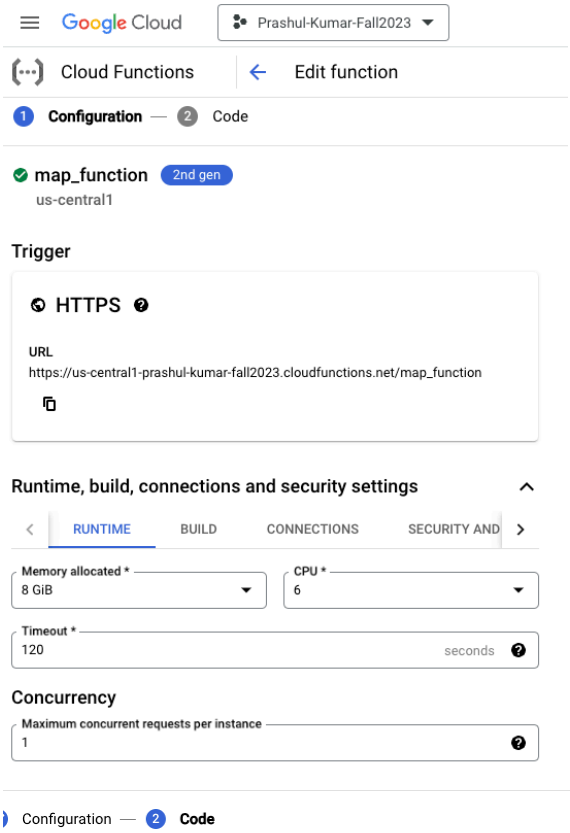
Environment	Name	Last deployed	Region	Recommendation	Trigger	Runtime	Memory allocated	Executed function	Actions
2nd gen	function-2	Dec 7, 2023, 5:32:25 PM	us-central1		HTTP	Python 3.11	256 MiB	hello_http	
2nd gen	main	Dec 10, 2023, 4:05:21 PM	us-central1		HTTP	Python 3.11	4 GiB	search	
2nd gen	map_function	Dec 9, 2023, 7:41:12 PM	us-central1		HTTP	Python 3.11	8 GiB	map_word_count	
2nd gen	master	Dec 10, 2023, 2:08:00 PM	us-central1		HTTP	Python 3.11	8 GiB	process_barrier	
2nd gen	reduce_function	Dec 7, 2023, 8:57:40 PM	us-central1		HTTP	Python 3.11	256 MiB	reduce_word_count	
2nd gen	reduce_word_count	Dec 10, 2023, 1:16:06 PM	us-central1		HTTP	Python 3.11	8 GiB	reduce_word_count	
2nd gen	trigger	Dec 10, 2023, 3:11:04 PM	us-central1		Bucket: praskumabucket	Python 3.11	8 GiB	process_new_file	

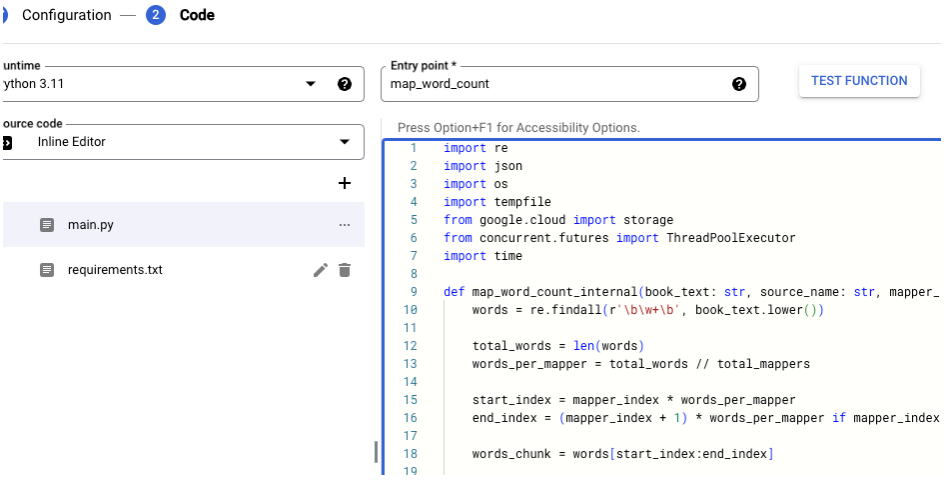
The highlighted functions are the ones which have been deployed with following params –

Master –

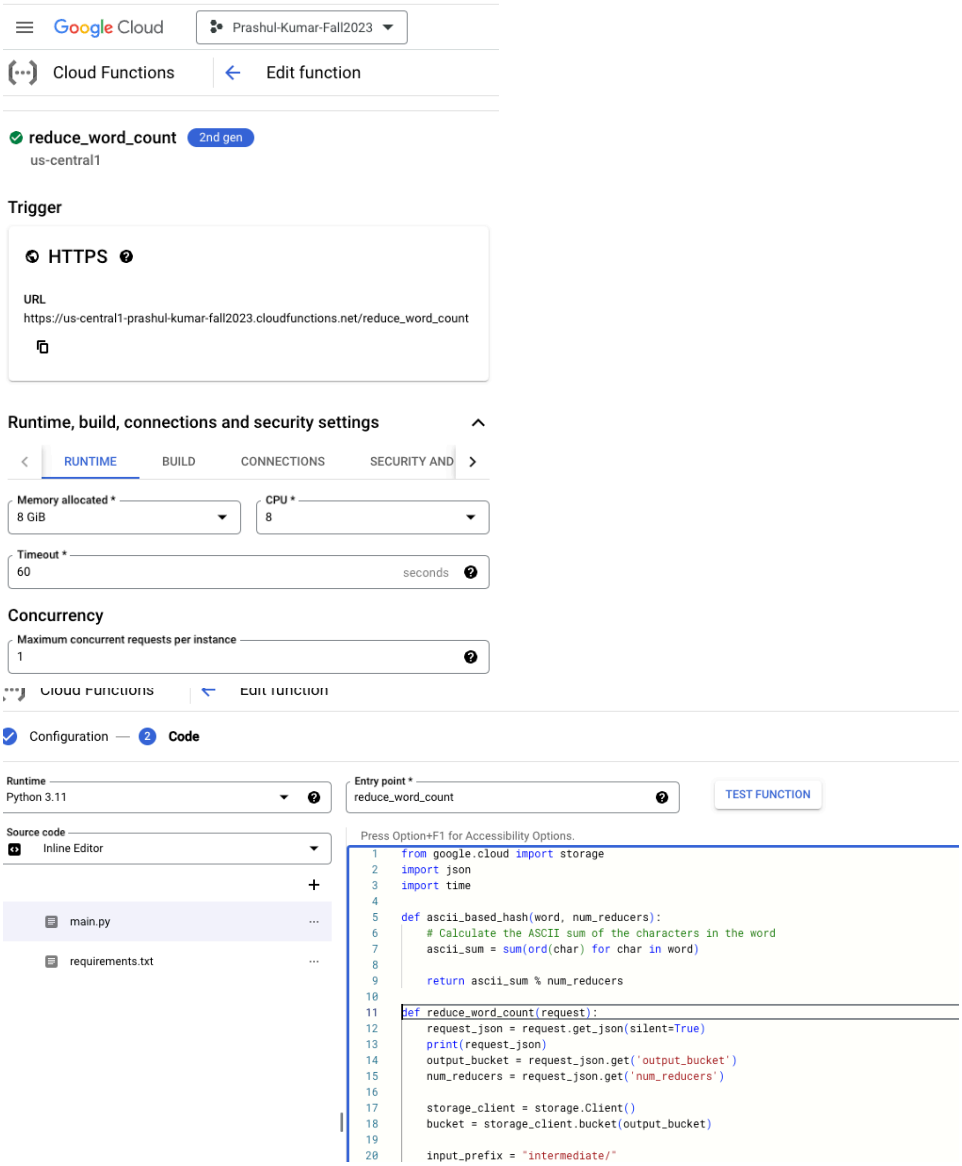

1 Configuration — 2 Code
master 2nd gen
us-central1
Trigger
HTTPS
URL
https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/master
Runtime, build, connections and security settings
RUNTIME
BUILD
CONNECTIONS
SECURITY AND
Memory allocated *
8 GiB
CPU *
4
Timeout *
120
seconds
Concurrency
Maximum concurrent requests per instance
1
Configuration — 2 Code
ntime
thon 3.11
Entry point *
process_barrier
TEST FUNCTION
urce code
Inline Editor
main.py
requirements.txt
Press Option+F1 for Accessibility Options.
import json
from google.cloud import storage
from concurrent.futures import ThreadPoolExecutor
import threading
import requests
import time
map_barrier = threading.Barrier(3)
reduce_barrier = threading.Barrier(3)
def list_files_with_prefix(bucket_name, prefix):
storage_client = storage.Client()
bucket = storage_client.bucket(bucket_name)
blobs = bucket.list_blobs(prefix=prefix)
files = [blob.name for blob in blobs]
return files
def invoke_cloud_function(payload, phase):
function_url = "https://us-central1-prashul-kumar-fall2023.clou
payload_json = json.dumps(payload)
headers = {"Content-Type": "application/json"}
start_time = time.time()
response = requests.post(function_url, data=payload_json, heade

Map.py –

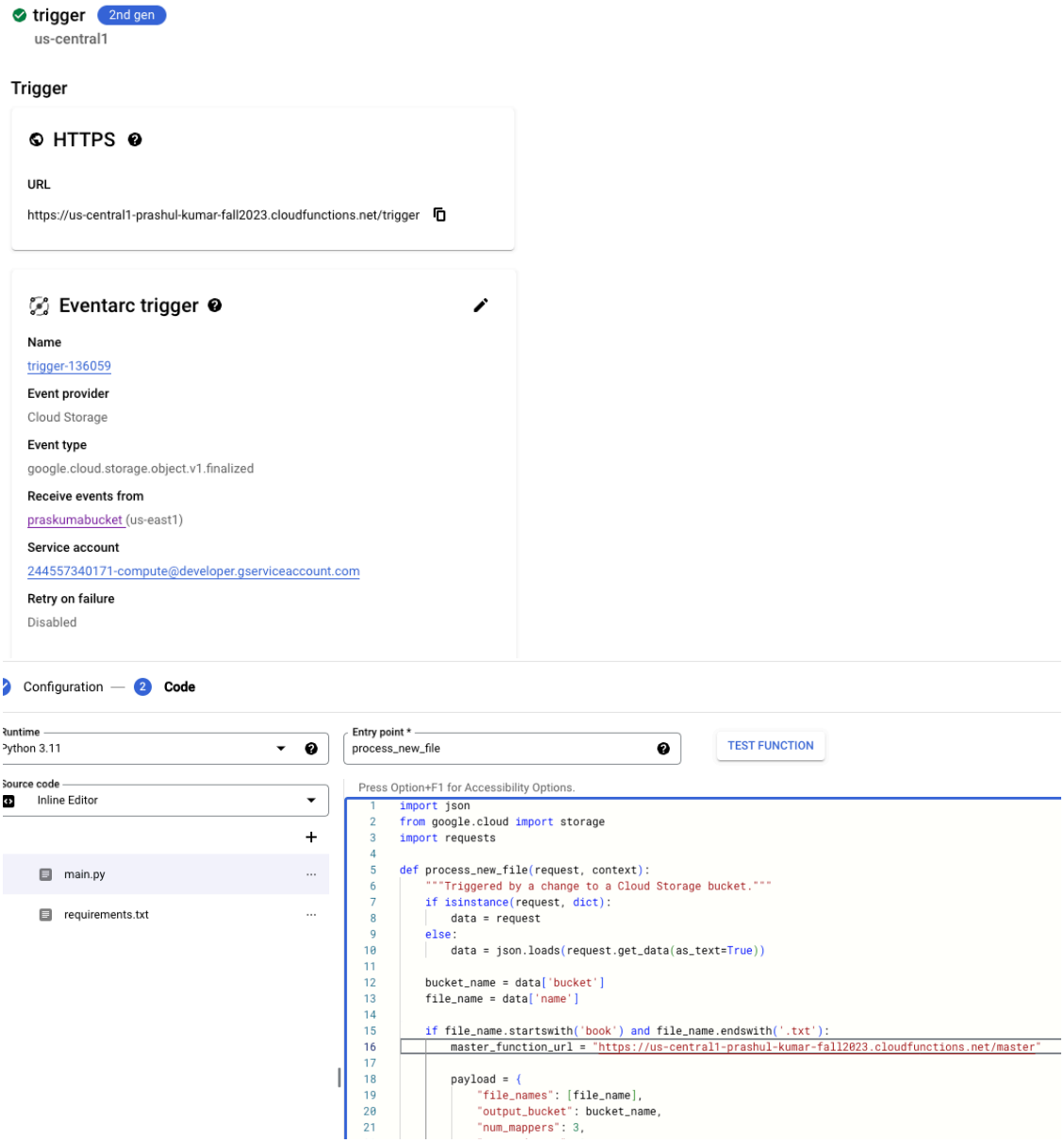

Google Cloud
Prashul-Kumar-Fall2023
Cloud Functions
Edit function
1 Configuration — 2 Code
map_function
2nd gen
us-central1
Trigger
HTTPS
URL
https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
Runtime, build, connections and security settings
RUNTIME
BUILD
CONNECTIONS
SECURITY AND
Memory allocated *
8 GiB
CPU *
6
Timeout *
120
seconds
Concurrency
Maximum concurrent requests per instance
1


Configuration — 2 Code
untime
ython 3.11
Entry point *
map_word_count
TEST FUNCTION
ource code
Inline Editor
main.py
requirements.txt
Press Option+F1 for Accessibility Options.
import re
import json
import os
import tempfile
from google.cloud import storage
from concurrent.futures import ThreadPoolExecutor
import time
def map_word_count_internal(book_text: str, source_name: str, mapper_
words = re.findall(r'\b\w+\b', book_text.lower())
total_words = len(words)
words_per_mapper = total_words // total_mappers
start_index = mapper_index * words_per_mapper
end_index = (mapper_index + 1) * words_per_mapper if mapper_index
words_chunk = words[start_index:end_index]

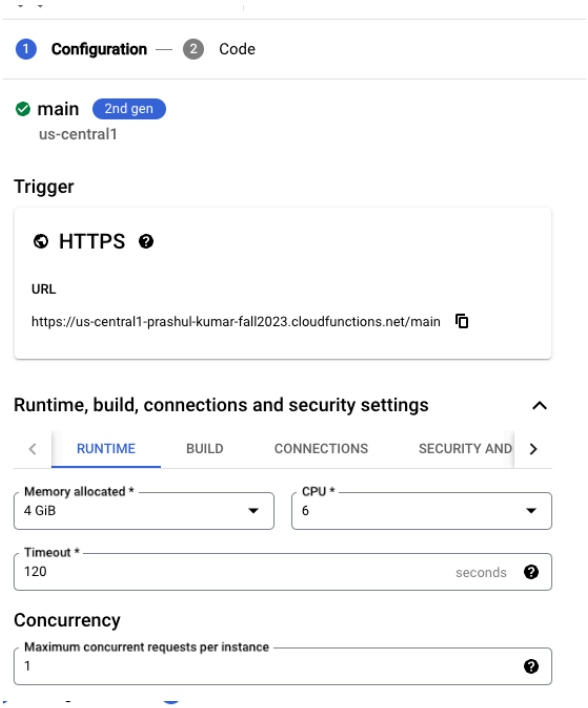
Reduce.py –

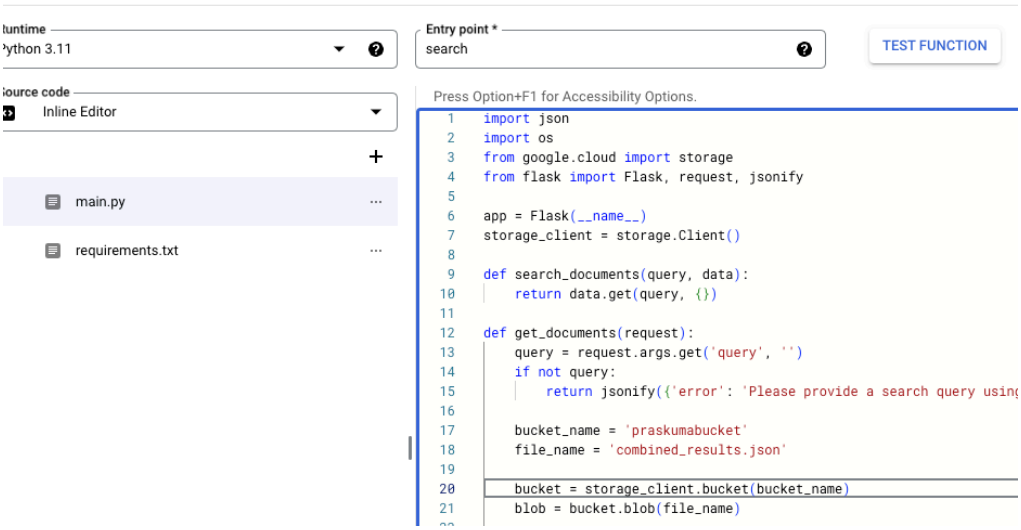

Google Cloud
Prashul-Kumar-Fall2023
Cloud Functions
Edit function
reduce_word_count
2nd gen
us-central1
Trigger
HTTPS
URL
https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
Runtime, build, connections and security settings
RUNTIME
BUILD
CONNECTIONS
SECURITY AND
Memory allocated *
8 GiB
CPU *
8
Timeout *
60
seconds
Concurrency
Maximum concurrent requests per instance
1
Configuration
2
Code
Runtime
Python 3.11
Entry point *
reduce_word_count
TEST FUNCTION
Source code
Inline Editor
main.py
requirements.txt
Press Option+F1 for Accessibility Options.
from google.cloud import storage
import json
import time
def ascii_based_hash(word, num_reducers):
# Calculate the ASCII sum of the characters in the word
ascii_sum = sum(ord(char) for char in word)
return ascii_sum % num_reducers
def reduce_word_count(request):
request_json = request.get_json(silent=True)
print(request_json)
output_bucket = request_json.get('output_bucket')
num_reducers = request_json.get('num_reducers')
storage_client = storage.Client()
bucket = storage_client.bucket(output_bucket)
input_prefix = "intermediate/"

Trigger.py –


trigger
2nd gen
us-central1
Trigger
HTTPS
URL
https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/trigger
Eventarc trigger
Name
trigger-136059
Event provider
Cloud Storage
Event type
google.cloud.storage.object.v1.finalized
Receive events from
praskumabucket (us-east1)
Service account
244557340171-compute@developer.gserviceaccount.com
Retry on failure
Disabled
Configuration — 2 Code
Runtime
Python 3.11
Entry point *
process_new_file
TEST FUNCTION
Source code
Inline Editor
main.py
requirements.txt
Press Option+F1 for Accessibility Options.
import json
from google.cloud import storage
import requests
def process_new_file(request, context):
"""Triggered by a change to a Cloud Storage bucket."""
if isinstance(request, dict):
data = request
else:
data = json.loads(request.get_data(as_text=True))
bucket_name = data['bucket']
file_name = data['name']
if file_name.startswith('book') and file_name.endswith('.txt'):
master_function_url = "https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/master"
payload = {
"file_names": [file_name],
"output_bucket": bucket_name,
"num_mappers": 3,

Main.py –





Now for loading the data to cloud storage, instead of script I downloaded the data from project Gutenberg (in this case book1, book2, book3, book4.txt) and uploaded it manually from the UI in my bucket

praskumabucket

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Subject to object ACLs	None

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE | OBSERVABILITY | INVENTORY REPORTS

Buckets > praskumabucket

UPLOAD FILES | UPLOAD FOLDER | CREATE FOLDER | TRANSFER DATA | MANAGE HOLDS | DOWNLOAD | DELETE

Filter by name prefix only | Filter | Filter objects and folders

Name	Size	Type	Created	Storage class	Last modified
book1.txt	38.7 KB	text/plain	Dec 1, 2023, 3:27:52 PM	Standard	Dec 1, 2023, 3:27:52 PM
book2.txt	138.8 KB	text/plain	Dec 1, 2023, 3:27:52 PM	Standard	Dec 1, 2023, 3:27:52 PM
book3.txt	788.3 KB	text/plain	Dec 1, 2023, 3:27:52 PM	Standard	Dec 1, 2023, 3:27:52 PM
book4.txt	221.1 KB	text/plain	Dec 10, 2023, 1:08:30 PM	Standard	Dec 10, 2023, 1:08:30 PM

What each book looks like –

ff1c40d5295a9a4eb5de33147258cfb5ed1871b063fa7ae9e9b7761-apidata.googleusercontent.com/download/sto

```
The Project Gutenberg eBook of A Modest Proposal

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this ebook or online
at www.gutenberg.org. If you are not located in the United States,
you will have to check the laws of the country where you are located
before using this eBook.

Title: A Modest Proposal


Author: Jonathan Swift

Release date: October 1, 1997 [eBook #1080]
                Most recently updated: March 10, 2021

Language: English


*** START OF THE PROJECT GUTENBERG EBOOK A MODEST PROPOSAL ***


A Modest Proposal

For preventing the children of poor people in Ireland,
from being a burden on their parents or country,
and for making them beneficial to the publick.

by Dr. Jonathan Swift

1729



It is a melancholy object to those, who walk through this great town,
or travel in the country, when they see the streets, the roads, and
cabbin-doors crowded with beggars of the female sex, followed by three,
four, or six children, all in rags, and importuning every passenger for
an alms. These mothers, instead of being able to work for their honest
livelihood, are forced to employ all their time in stroling to beg
sustenance for their helpless infants who, as they grow up, either turn
thieves for want of work, or leave their dear native country, to fight
for the Pretender in Spain, or sell themselves to the Barbadoes.
```

Test cases and Examples –

Test Case 1: Show 1 complete flow of mapreduce program with all the steps involved on how to run:

Now how to execute the test –
IN VM execute -

curl -X POST https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/master

```
Processing completed.prashulkumar@praskumainstance2:~$ curl -X POST https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/master
Processing completed.prashulkumar@praskumainstance2:~$ ls
```

This starts the master function which inturn invokes the map and reduce and then writes to the final. Let's look at images below to show all the results, time taken and final output. Remember the source has 4 files – book1,2,3,4.txt

Master function -

Now let's go step by step, first look at the logs of master function –

2023-12-10 18:58:59.977 EST
POST200676 B10.4 scurl/7.74.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/master
2023-12-10 18:59:02.922 EST
book2.txt processed by map. Elapsed time: 2.7635631561279297 secondsbook1.txt processed by map. Elapsed time: 2.7699153423309326 seconds
2023-12-10 18:59:02.923 EST
book1.txt processed by map. Elapsed time: 2.7763655185699463 seconds
2023-12-10 18:59:03.766 EST
book3.txt processed by map. Elapsed time: 3.602792739868164 seconds
2023-12-10 18:59:03.766 EST
book2.txt processed by map. Elapsed time: 3.6052310466766357 seconds
2023-12-10 18:59:03.769 EST
book1.txt processed by map. Elapsed time: 3.619551420211792 seconds
2023-12-10 18:59:04.363 EST
book2.txt processed by map. Elapsed time: 4.206693172454834 secondsbook3.txt processed by map. Elapsed time: 4.197648763656616 seconds
2023-12-10 18:59:04.365 EST
book4.txt processed by map. Elapsed time: 4.194602012634277 seconds
2023-12-10 18:59:05.054 EST
book3.txt processed by map. Elapsed time: 4.886235237121582 seconds
2023-12-10 18:59:05.054 EST
book4.txt processed by map. Elapsed time: 4.882280588150024 seconds
2023-12-10 18:59:05.057 EST
book4.txt processed by map. Elapsed time: 4.88309907913208 seconds
2023-12-10 18:59:05.061 EST
All map operations completed. Starting reduce operations...
2023-12-10 18:59:08.962 EST
Reducer 2 completed. Elapsed time: 3.8954060077667236Reducer 0 completed. Elapsed time: 3.9007718563079834

2023-12-10 18:59:08.962 EST
Reducer 1 completed. Elapsed time: 3.898232936859131
2023-12-10 18:59:08.967 EST
Reducer tasks completed.
2023-12-10 18:59:10.352 EST
Combined results saved to 'combined_results.json' in the bucket.
2023-12-10 18:59:11.517 EST

Screenshot depicting the same of the logs-

>	2023-12-10 18:58:59.977 EST	POST 200 676 B 10.4 s curl/7.74.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/master
>	2023-12-10 18:59:02.922 EST	book2.txt processed by map. Elapsed time: 2.7635631561279297 secondsbook1.txt processed by map. Elapsed time: 2.7699153423309326 seconds
>	2023-12-10 18:59:02.923 EST	book1.txt processed by map. Elapsed time: 2.7763655185699463 seconds
>	2023-12-10 18:59:03.766 EST	book3.txt processed by map. Elapsed time: 3.602792739868164 seconds
>	2023-12-10 18:59:03.766 EST	book2.txt processed by map. Elapsed time: 3.6052310466766357 seconds
>	2023-12-10 18:59:03.769 EST	book1.txt processed by map. Elapsed time: 3.619551420211792 seconds
>	2023-12-10 18:59:04.363 EST	book2.txt processed by map. Elapsed time: 4.206693172454834 secondsbook3.txt processed by map. Elapsed time: 4.197648763656616 seconds
>	2023-12-10 18:59:04.365 EST	book4.txt processed by map. Elapsed time: 4.194602012634277 seconds
>	2023-12-10 18:59:05.054 EST	book3.txt processed by map. Elapsed time: 4.886235237121582 seconds
>	2023-12-10 18:59:05.054 EST	book4.txt processed by map. Elapsed time: 4.882280588150024 seconds
>	2023-12-10 18:59:05.057 EST	book4.txt processed by map. Elapsed time: 4.88309907913208 seconds
>	2023-12-10 18:59:05.061 EST	All map operations completed. Starting reduce operations...
>	2023-12-10 18:59:08.962 EST	Reducer 2 completed. Elapsed time: 3.8954060077667236Reducer 0 completed. Elapsed time: 3.9007718563079834
>	2023-12-10 18:59:08.962 EST	Reducer 1 completed. Elapsed time: 3.898232936859131
>	2023-12-10 18:59:08.967 EST	Reducer tasks completed.
>	2023-12-10 18:59:10.352 EST	Combined results saved to 'combined_results.json' in the bucket.

Remember each book is processed thrice since we have 3 chunks of each book (source) which the map runs in parallel. Also the reduce has started only after map has completed all.

Map function -
Now let's look at the logs of map function –

Map logs:

2023-12-10 18:59:02.001 EST	POST	200	872 B	855 ms	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.065 EST	POST	200	875 B	693 ms	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.126 EST	POST	200	875 B	781 ms	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.193 EST	POST	200	875 B	713 ms	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.198 EST	POST	200	875 B	957 ms	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.206 EST	POST	200	876 B	1.2 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.245 EST	POST	200	876 B	1.1 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.262 EST	POST	200	873 B	1.6 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.269 EST	POST	200	876 B	1.8 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.274 EST	POST	200	876 B	1.5 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.279 EST	POST	200	875 B	1.1 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:02.279 EST	POST	200	876 B	1.6 s	python-requests/2.31.0	https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
2023-12-10 18:59:03.258 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "map--function-1" on port 8080.					
2023-12-10 18:59:03.281 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "map--function-1" on port 8080.					
2023-12-10 18:59:03.446 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "map--function-1" on port 8080.					
2023-12-10 18:59:03.457 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "map--function-1" on port 8080.					
2023-12-10 18:59:03.471 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "map--function-1" on port 8080.					

From the logs above and instance count graph we can see that POST has been called 12 times, thus invoking the map function that many times across different instances which confirm that all the map operations are running in parallel across different instances and not on the same instance 12 times showing parallelism.

To confirm the same we can see the different instance id for some sample log below –

2023-12-10 18:59:02.001 EST

POST200872 B855 mspython-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function

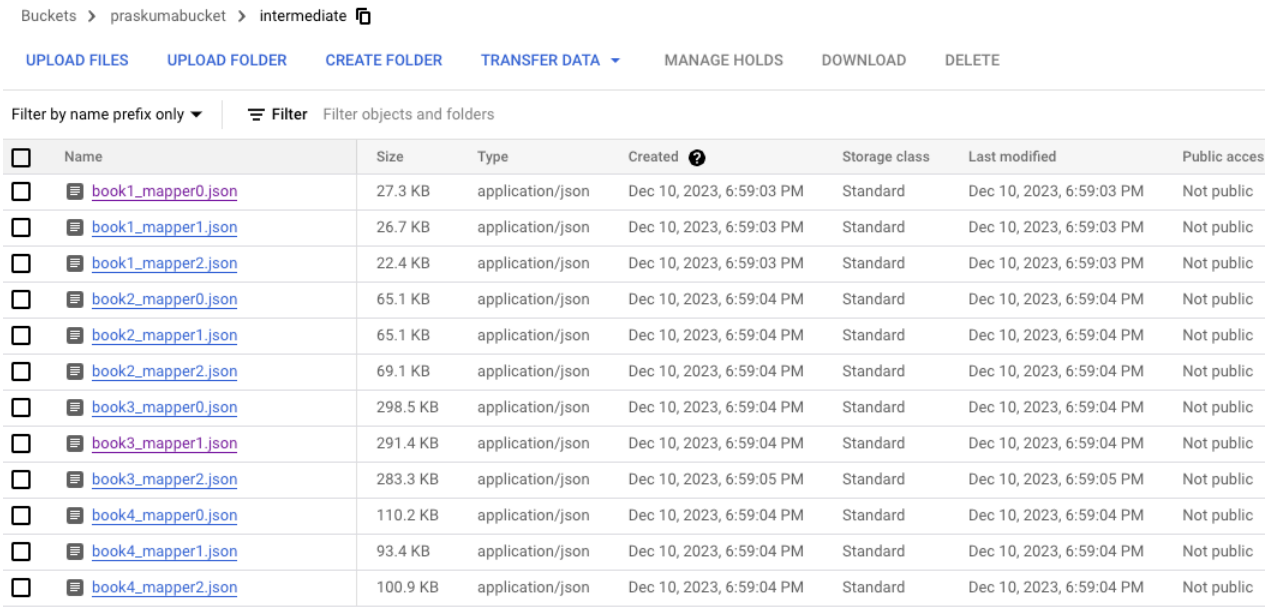
Open in Logs Explorer

{
httpRequest: {10}
insertId: "657650c6000d350ed1238f60"
labels: {
goog-managed-by: "cloudfunctions"
instanceId: "0087599d423e970122601a8ae466b7f756bcdd0857036817daee867b1817b9777393b6b3cf528769da66e32cb0075c87a301aae490328bd4b2b98a576c84cb5139"
}
logName: "projects/prashul-kumar-fall2023/logs/run.googleapis.com%2Frequests"
receiveTimestamp: "2023-12-10T23:59:02.871231856Z"
resource: {2}
severity: "INFO"

spanId: "3283334573801074309"
timestamp: "2023-12-10T23:59:02.001769Z"
trace: "projects/prashul-kumar-fall2023/traces/dcdb95651ec7e1b2a653fabbe7c6eae3"
traceSampled: true
}
2023-12-10 18:59:02.065 EST
POST200875 B693 mspython-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/map_function
Open in Logs Explorer
{
httpRequest: {10}
insertId: "657650c6000bc2a1d667c9fc"
labels: {
instanceId: "0087599d420837ab00569e37492d1c2438551c1d016a3b1073c0635617448f0a1497498fab7b3b2a11ce49622e0cb12831a364e8a71641e02b430b64e027595da665"
goog-managed-by: "cloudfunctions"
}
logName: "projects/prashul-kumar-fall2023/logs/run.googleapis.com%2Frequests"
receiveTimestamp: "2023-12-10T23:59:02.778554835Z"
resource: {2}
severity: "INFO"
spanId: "10514560909600777760"
timestamp: "2023-12-10T23:59:02.065206Z"
trace: "projects/prashul-kumar-fall2023/traces/a47eaccb5db3b5ca35b8fc84699cd38b"
traceSampled: true

Intermediate results:

Now let's look at the intermediate results stored in GCS bucket –



Name	Size	Type	Created	Storage class	Last modified	Public acces
book1_mapper0.json	27.3 KB	application/json	Dec 10, 2023, 6:59:03 PM	Standard	Dec 10, 2023, 6:59:03 PM	Not public
book1_mapper1.json	26.7 KB	application/json	Dec 10, 2023, 6:59:03 PM	Standard	Dec 10, 2023, 6:59:03 PM	Not public
book1_mapper2.json	22.4 KB	application/json	Dec 10, 2023, 6:59:03 PM	Standard	Dec 10, 2023, 6:59:03 PM	Not public
book2_mapper0.json	65.1 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book2_mapper1.json	65.1 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book2_mapper2.json	69.1 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book3_mapper0.json	298.5 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book3_mapper1.json	291.4 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book3_mapper2.json	283.3 KB	application/json	Dec 10, 2023, 6:59:05 PM	Standard	Dec 10, 2023, 6:59:05 PM	Not public
book4_mapper0.json	110.2 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book4_mapper1.json	93.4 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public
book4_mapper2.json	100.9 KB	application/json	Dec 10, 2023, 6:59:04 PM	Standard	Dec 10, 2023, 6:59:04 PM	Not public

This shows that each book-mapper combination has written to intermediate storage in GCS bucket. Let's look at the sample file to see data for one of them –

Book1mapper0.json -

```
{"are": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "yearly": {"book1.txt": [1]}, "overrun": {"book1.txt": [1]}, "being": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1]}, "the": {"book1.txt": [1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "principal": {"book1.txt": [1, 1]}, "breeders": {"book1.txt": [1, 1]}, "of": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
[1, 1, 1]}, "as": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "well": {"book1.txt": [1, 1]}, "our": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "most": {"book1.
{"book1.txt": [1]}, "enemies": {"book1.txt": [1]}, "and": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1]}, "who": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "stay": {"book1.txt": [1, 1]}, "at": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "home": {"book1.txt":
1]}, "purpose": {"book1.txt": [1, 1]}, "with": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "a": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "design": {"book1.txt": [1, 1]}, "to": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "deliver": {"book1.txt": [1]}, "kingdom": {"book1.txt": [1, 1, 1, 1, 1, 1]}, "pretender": {"book1.txt": [1]}, "hoping": {"book1.txt": [1]}, "take": {"book1.txt": [
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "advantage": {"book1.txt": [1]}, "by": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "absence": {"book1.tx
1, 1, 1, 1, 1]}, "many": {"book1.txt": [1, 1, 1, 1]}, "good": {"book1.txt": [1, 1, 1, 1]}, "protestants": {"book1.txt": [1]}, "have": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "chosen": {"
{"book1.txt": [1]}, "leave": {"book1.txt": [1, 1]}, "country": {"book1.txt": [1, 1, 1, 1, 1, 1, 1]}, "than": {"book1.txt": [1, 1, 1, 1]}, "pay": {"book1.txt": [1, 1]}, "tithes": {"book1.txt": [1]}, "agains
"conscience": {"book1.txt": [1]}, "an": {"book1.txt": [1, 1, 1, 1, 1, 1]}, "episcopal": {"book1.txt": [1]}, "curate": {"book1.txt": [1]}, "secondly": {"book1.txt": [1, 1]}, "poorer": {"book1.txt": [1]}, "
"will": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "something": {"book1.txt": [1, 1, 1]}, "valuable": {"book1.txt": [1]}, "own": {"book1.txt": [1, 1, 1, 1, 1, 1]}, "which":
1]}, "law": {"book1.txt": [1, 1, 1, 1]}, "may": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1]}, "be": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "made":
{"book1.txt": [1]}, "distress": {"book1.txt": [1]}, "help": {"book1.txt": [1, 1]}, "landlord": {"book1.txt": [1]}, "s": {"book1.txt": [1, 1, 1, 1, 1, 1]}, "rent": {"book1.txt": [1, 1]}, "corn": {"book1.txt
"already": {"book1.txt": [1]}, "seized": {"book1.txt": [1]}, "money": {"book1.txt": [1, 1, 1]}, "thing": {"book1.txt": [1]}, "unknown": {"book1.txt": [1]}, "thirdly": {"book1.txt": [1]}, "whereas": {"book1
{"book1.txt": [1]}, "hundred": {"book1.txt": [1, 1]}, "thousand": {"book1.txt": [1, 1, 1, 1, 1, 1]}, "children": {"book1.txt": [1, 1, 1, 1, 1]}, "from": {"book1.txt": [1, 1, 1, 1, 1, 1]}, "two": {"book1
{"book1.txt": [1, 1, 1]}, "old": {"book1.txt": [1, 1, 1, 1]}, "upwards": {"book1.txt": [1]}, "cannot": {"book1.txt": [1]}, "computed": {"book1.txt": [1]}, "less": {"book1.txt": [1]}, "ten": {"book1.txt":
1]}, "piece": {"book1.txt": [1]}, "per": {"book1.txt": [1, 1, 1]}, "annum": {"book1.txt": [1, 1, 1]}, "stock": {"book1.txt": [1, 1]}, "thereby": {"book1.txt": [1, 1, 1]}, "encreased": {"book1.txt": [1]}, "
{"book1.txt": [1, 1]}, "besides": {"book1.txt": [1, 1, 1]}, "profit": {"book1.txt": [1, 1]}, "new": {"book1.txt": [1, 1]}, "dish": {"book1.txt": [1]}, "introduced": {"book1.txt": [1]}, "tables": {"book1.tx
1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "gentlemen": {"book1.txt": [1, 1]}, "fortune": {"book1.txt": [1]}, "in": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1]}, "any": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "refinement": {"book1.txt": [1]}, "taste": {"book1.txt": [1, 1]}, "circulate": {"book1.txt": [1]}, "among": {"book1.txt": [1, 1, 1]},
{"book1.txt": [1, 1]}, "entirely": {"book1.txt": [1]}, "growth": {"book1.txt": [1, 1]}, "manufacture": {"book1.txt": [1, 1]}, "fourthly": {"book1.txt": [1]}, "constant": {"book1.txt": [1, 1, 1]}, "gain":
{"book1.txt": [1]}, "sterling": {"book1.txt": [1, 1]}, "sale": {"book1.txt": [1]}, "rid": {"book1.txt": [1]}, "charge": {"book1.txt": [1, 1, 1, 1]}, "maintaining": {"book1.txt": [1]}, "them": {"book1.txt":
{"book1.txt": [1, 1]}, "first": {"book1.txt": [1, 1, 1]}, "year": {"book1.txt": [1, 1]}, "fifthly": {"book1.txt": [1]}, "this": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
"food": {"book1.txt": [1, 1, 1]}, "would": {"book1.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "likewise": {"book1.txt": [1]}, "bring": {"book1.txt": [1, 1]}, "great": {"book1.txt": [1, 1, 1, 1]}, "custom": {"
{"book1.txt": [1]}, "where": {"book1.txt": [1, 1, 1]}, "vintners": {"book1.txt": [1]}, "certainly": {"book1.txt": [1]}, "prudent": {"book1.txt": [1]}, "procure": {"book1.txt": [1]}, "best": {"book1.txt":
```


Book3mapper2.json -


```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "depends":
{"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "say": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "action": {"book3.txt": [1, 1, 1, 1, 1, 1, 1]}, "cannot": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "possibly": {"book3.txt": [1, 1,
"because": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "must": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1]}, "s": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "sake
1, 1, 1, 1, 1, 1, 1]}, "hardest": {"book3.txt": [1]}, "instantly": {"book3.txt": [1, 1]}, "obedient": {"book3.txt": [1]}, "still": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
{"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "put": {"book3.txt": [1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "leave": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "alone": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
"minutes": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "death": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "world": {"book3.txt": [1
1]}, "delay": {"book3.txt": [1, 1, 1, 1, 1, 1]}, "submissive": {"book3.txt": [1]}, "see": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "nothing": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "else": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1]}, "kissed": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1]}, "key": {"book3.txt": [1, 1, 1, 1, 1]}, "hurrying": {"book3.txt": [1]}, "partly": {"book3.txt": [1, 1, 1, 1]}, "looked": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "throng": {"book3.txt": [1, 1, 1]}, "men": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
"women": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "enough": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "number": {"book3.txt": [1, 1, 1, 1
{"book3.txt": [1, 1, 1, 1]}, "forty": {"book3.txt": [1, 1, 1, 1, 1]}, "fifty": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "possession": {"book3.txt": [1, 1, 1, 1]}, "them": {"book3.txt": [1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "work": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "evidently": {"book3.txt": [1, 1, 1, 1]}, "set": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "purpose": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1]}, "convenient": {"book3.txt": [1, 1]}, "spot": {"bo
"awful": {"book3.txt": [1, 1, 1, 1, 1]}, "workers": {"book3.txt": [1]}, "double": {"book3.txt": [1]}, "handle": {"book3.txt": [1, 1]}, "madly": {"book3.txt": [1]}, "whose": {"book3.txt": [1, 1, 1, 1, 1, 1,
"faces": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "long": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1]}, "hair": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "flapped": {"book3.txt": [1]}, "whirlings": {"book3.txt": [1]}, "horrible": {"book3.txt": [1, 1]}, "cruel": {
"visages": {"book3.txt": [1]}, "wildest": {"book3.txt": [1, 1]}, "savages": {"book3.txt": [1]}, "most": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
"disguise": {"book3.txt": [1]}, "false": {"book3.txt": [1, 1, 1, 1]}, "eyebrows": {"book3.txt": [1, 1, 1]}, "moustaches": {"book3.txt": [1]}, "stuck": {"book3.txt": [1, 1]}, "hideous": {"book3.txt": [1, 1]
1]}, "bloody": {"book3.txt": [1, 1]}, "sweaty": {"book3.txt": [1]}, "awry": {"book3.txt": [1]}, "howling": {"book3.txt": [1]}, "staring": {"book3.txt": [1, 1, 1]}, "glaring": {"book3.txt": [1, 1, 1]}, "bea
"excitement": {"book3.txt": [1, 1]}, "want": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "sleep": {"book3.txt": [1, 1, 1, 1, 1, 1]}, "ruffians": {"book3.txt": [1]}, "matted": {"book3.txt": [1]}, "
{"book3.txt": [1, 1]}, "forward": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "over": {"book3.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```


Here we see all the mappers have the same structure of {“word”: “bookn.txt”:[1,1,1,1]}
The ones are the times the word is seen in that part of mapper and thus that many times 1 is emitted.

Reduce function-

Now let’s look at the reduce function logs. Remember I have 3 reduce functions –

> i	2023-12-10 18:59:05.493 EST	POST 200 766 B 2.3 s python-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
> *	2023-12-10 18:59:05.498 EST	POST 200 766 B 2.3 s python-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
> *	2023-12-10 18:59:05.500 EST	{'output_bucket': 'praskumabucket', 'num_reducers': 3, 'reducer_index': 1}
> i	2023-12-10 18:59:05.501 EST	POST 200 766 B 2.4 s python-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
> *	2023-12-10 18:59:05.505 EST	{'output_bucket': 'praskumabucket', 'num_reducers': 3, 'reducer_index': 0}
> i	2023-12-10 18:59:06.543 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "reduce--word--count-1" on port 8080.
> *	2023-12-10 18:59:06.609 EST	{'output_bucket': 'praskumabucket', 'num_reducers': 3, 'reducer_index': 2}

Here again we can see that the POST function was invoked 3 times, proving that the reduce function was invoked 3 times in parallel.
The same is confirmed from the instance count also –

Now let's look at the output of the reduce function, i.e. the 3 files –
First the files generated in gcs bucket –

☐	intermediate_0_reduced_final.json	158.3 KB	application/json	Dec 10, 2023, 6:59:07 PM	Standard	Dec 10, 2023, 6:59:07 PM	Not public
☐	intermediate_1_reduced_final.json	153.5 KB	application/json	Dec 10, 2023, 6:59:07 PM	Standard	Dec 10, 2023, 6:59:07 PM	Not public
☐	intermediate_2_reduced_final.json	153.6 KB	application/json	Dec 10, 2023, 6:59:08 PM	Standard	Dec 10, 2023, 6:59:08 PM	Not public

Output of the intermediate reducers –


```
{"the": {"book1.txt": 357, "book2.txt": 1348, "book3.txt": 8241, "book4.txt": 3579}, "project": {"book1.txt": 89, "book2.txt": 90, "book3.txt
97}, "ebook": {"book1.txt": 13, "book2.txt": 14, "book3.txt": 13, "book4.txt": 13}, "of": {"book1.txt": 259, "book2.txt": 557, "book3.txt":
204}, "use": {"book1.txt": 15, "book2.txt": 24, "book3.txt": 32, "book4.txt": 24}, "anywhere": {"book1.txt": 2, "book2.txt": 4, "book3.txt":
"other": {"book1.txt": 26, "book2.txt": 62, "book3.txt": 201, "book4.txt": 92}, "world": {"book1.txt": 3, "book2.txt": 4, "book3.txt": 46, "
{"book1.txt": 3, "book2.txt": 3, "book3.txt": 6, "book4.txt": 3}, "with": {"book1.txt": 59, "book2.txt": 247, "book3.txt": 1358, "book4.txt":
{"book1.txt": 111, "book2.txt": 119, "book3.txt": 441, "book4.txt": 222}, "terms": {"book1.txt": 22, "book2.txt": 23, "book3.txt": 27, "book
{"book1.txt": 4, "book2.txt": 4, "book3.txt": 4, "book4.txt": 5}, "www": {"book1.txt": 9, "book2.txt": 9, "book3.txt": 9, "book4.txt": 10},
"book2.txt": 30, "book3.txt": 337, "book4.txt": 192}, "located": {"book1.txt": 7, "book2.txt": 7, "book3.txt": 7, "book4.txt": 7}, "have": {
```

```
{"this": {"book1.txt": 71, "book2.txt": 143, "book3.txt": 599, "book4.txt": 243}, "anyone": {"book1.txt": 5, "book2.txt": 13, "book3.txt": 5,
"parts": {"book1.txt": 4, "book2.txt": 2, "book3.txt": 5, "book4.txt": 17}, "no": {"book1.txt": 22, "book2.txt": 84, "book3.txt": 553, "book
{"book1.txt": 12, "book2.txt": 13, "book3.txt": 12, "book4.txt": 13}, "it": {"book1.txt": 41, "book2.txt": 385, "book3.txt": 2082, "book4.tx
{"book1.txt": 2, "book2.txt": 14, "book3.txt": 13, "book4.txt": 2}, "under": {"book1.txt": 10, "book2.txt": 25, "book3.txt": 148, "book4.txt
{"book1.txt": 189, "book2.txt": 835, "book3.txt": 3653, "book4.txt": 860}, "country": {"book1.txt": 16, "book2.txt": 5, "book3.txt": 41, "boo
{"book1.txt": 3, "book2.txt": 1, "book3.txt": 1, "book4.txt": 11}, "jonathan": {"book1.txt": 2}, "swift": {"book1.txt": 2, "book3.txt": 6},
"updated": {"book1.txt": 2, "book2.txt": 2, "book3.txt": 2, "book4.txt": 1}, "2021": {"book1.txt": 1, "book2.txt": 1, "book3.txt": 1}, "langu
"book2.txt": 1, "book3.txt": 47, "book4.txt": 4}, "on": {"book1.txt": 18, "book2.txt": 165, "book3.txt": 938, "book4.txt": 219}, "publick":
```


Now after this we combine the final 3 reducer outputs in a single file, so that when user queries the word, instead of iterating through3 files it just looks at the combined file –
So let's look how the final combined output json looks like –

☐	book4.txt	221.1 KB	text/plain	Dec 10, 2023, 1:08:30 PM	Standard	Dec 10, 2023, 1:08:30 PM	Not public
☐	combined_results.json	613.7 KB	text/plain	Dec 10, 2023, 6:59:10 PM	Standard	Dec 10, 2023, 6:59:10 PM	Not public

Screenshot showing that the final combined json is created.


```
{
  "the": {
    "book1.txt": 357,
    "book2.txt": 1348,
    "book3.txt": 8241,
    "book4.txt": 3579
  },
  "project": {
    "book1.txt": 89,
    "book2.txt": 90,
    "book3.txt": 88,
    "book4.txt": 88
  },
  "gutenberg": {
    "book1.txt": 97,
    "book2.txt": 99,
    "book3.txt": 97,
    "book4.txt": 97
  },
  "ebook": {
    "book1.txt": 13,
    "book2.txt": 14,
    "book3.txt": 13,
    "book4.txt": 13
  },
  "of": {
    "book1.txt": 259,
    "book2.txt": 557,
    "book3.txt": 4140,
    "book4.txt": 1996
  },
  "for": {
    "book1.txt": 72,
    "book2.txt": 193,
    "book3.txt": 987,
    "book4.txt": 204
  },
  "use": {
    "book1.txt": 15,
    "book2.txt": 24,
    "book3.txt": 32,
    "book4.txt": 24
  },
  "anywhere": {
    "book1.txt": 2,
    "book2.txt": 4,
    "book3.txt": 6,
    "book4.txt": 2
```


Notice the final output wherein for each word, all the book's data is combined as compared to earlier where not all words were present in each reducer output.

Front end:

To access the front-end I have the index.html in stored in the gcs bucket itself which is the front end for the main function. (Remember main function is the one which takes the input as combined file and sends to the user for the queried word) –

So we open the index.html –

Search Documents

Enter Search Term: Search

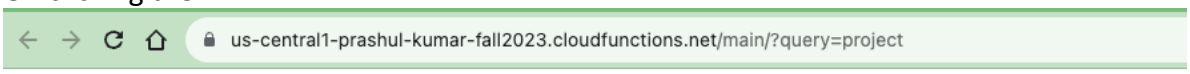
Now suppose user enters the word ‘project’ then the link is generated for that –

Search Documents

Enter Search Term: project Search

Results for query: https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/main/?query=project

On clicking the link –



{"book1.txt":89,"book2.txt":90,"book3.txt":88,"book4.txt":88}

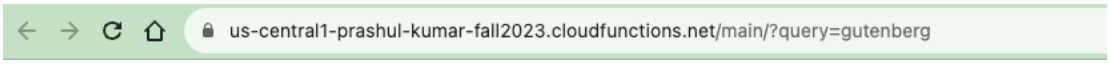
Thus for each word, the link gets generated and on clicking the link, we are able to see the results for that, how many times that word appears in that document.

Eg2:

Search Documents

Enter Search Term: gutenberg Search

Results for query: https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/main/?query=gutenberg



{"book1.txt":97,"book2.txt":99,"book3.txt":97,"book4.txt":97}

Now let’s look at the logs to verify the same of main function –

We can see the same https GET request created in the logs of main function which I am showing in the front-end.

Hence, this case, shows just by running the curl command in VM, the entire map-reduce function happens, and we finally get the result in the front end for the specific query/word which the user inputs.

Test Case 2: Streaming Function

For this, automatically trigger the function as soon as a file is uploaded in the GCS bucket. Hence lets upload file in gcs bucket let's say book5 through the UI –

☐	book4.txt	221.1 KB	text/plain	Dec 10, 2023, 1:08:30 PM	Standard	Dec 10, 2023, 1:08:30 PM	Not public
☐	book5.txt	714.4 KB	text/plain	Dec 10, 2023, 7:35:28 PM	Standard	Dec 10, 2023, 7:35:28 PM	Not public
☐	combined_results.json	613.7 KB	text/plain	Dec 10, 2023, 6:59:10 PM	Standard	Dec 10, 2023, 6:59:10 PM	Not public

Now we see that book5 is uploaded in GCS bucket which should result in triggering of trigger function in faas –

✓	2nd gen	trigger	Dec 10, 2023, 3:11:04 PM	us-central1	Bucket: praskumabucket	Python 3.11	8 GiB

Let's look at trigger function logs, after uploading the book5 –

❯	2023-12-10 19:35:48.213 EST	POST 200 653 B 7 ms APIs-Google; (+https://developers.goo…	https://trigger-ohwrtazuka-uc.a.run.app/?__GCP_CloudEventsMode=GCS_NOTIFICATION
❯	2023-12-10 19:35:48.222 EST	Master function triggered successfully for file: book5.txt	
❯	2023-12-10 19:35:49.753 EST	Combined results saved to 'combined_results.json' in the bucket.	
❯	2023-12-10 19:35:49.774 EST	POST 200 653 B 7 ms APIs-Google; (+https://developers.goo…	https://trigger-ohwrtazuka-uc.a.run.app/?__GCP_CloudEventsMode=GCS_NOTIFICATION

2023-12-10 19:35:48.213 EST
POST200653 B7 msAPIs-Google; (+https://developers.google.com/webmasters/APIs-Google.html) https://trigger-ohwrtazuka-uc.a.run.app/?__GCP_CloudEventsMode=GCS_NOTIFICATION
2023-12-10 19:35:48.222 EST
Master function triggered successfully for file: book5.txt
2023-12-10 19:35:49.753 EST
Combined results saved to 'combined_results.json' in the bucket.
2023-12-10 19:35:49.774 EST
POST200653 B7 msAPIs-Google; (+https://developers.google.com/webmasters/APIs-Google.html) https://trigger-ohwrtazuka-uc.a.run.app/?__GCP_CloudEventsMode=GCS_NOTIFICATION

We can see that the master function was successfully invoked and book5 uploading resulted in finally going through the whole cycle to upload the results for the same.

Let's look at the logs to confirm the same –

Master function –

2023-12-10 19:35:39.207 EST	book5.txt processed by map. Elapsed time: 6.377906084060669 secondsbook4.txt processed by map. Elapsed time
2023-12-10 19:35:39.214 EST	All map operations completed. Starting reduce operations...
2023-12-10 19:35:46.708 EST	Reducer 2 completed. Elapsed time: 7.489595651626587Reducer 1 completed. Elapsed time: 7.491865158081055
2023-12-10 19:35:46.710 EST	Reducer 0 completed. Elapsed time: 7.495272397994995
2023-12-10 19:35:46.712 EST	Reducer tasks completed.
2023-12-10 19:35:48.196 EST	Combined results saved to 'combined_results.json' in the bucket.

Reduce function –

2023-12-10 19:35:39.598 EST	POST 200 823 B 4 s python-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
2023-12-10 19:35:39.624 EST	POST 200 766 B 3 s python-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
2023-12-10 19:35:39.626 EST	POST 200 764 B 2.9 s python-requests/2.31.0 https://us-central1-prashul-kumar-fall2023.cloudfunctions.net/reduce_word_count
2023-12-10 19:35:40.689 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "reduce--word--count-1" on port 8080.
2023-12-10 19:35:40.705 EST	Default STARTUP TCP probe succeeded after 1 attempt for container "reduce--word--count-1" on port 8080.
2023-12-10 19:35:40.739 EST	{'output_bucket': 'praskumabucket', 'num_reducers': 3, 'reducer_index': 0}
2023-12-10 19:35:40.766 EST	{'output_bucket': 'praskumabucket', 'num_reducers': 3, 'reducer_index': 1}
2023-12-10 19:35:43.764 EST	{'output_bucket': 'praskumabucket', 'num_reducers': 3, 'reducer_index': 2}

Intermediate map results –

book5_mapper0.json	274.8 KB	application/json	Dec 10, 2023, 7:35:38 PM	Standard	Dec 10, 2023, 7:35:38 PM	Not
book5_mapper1.json	284 KB	application/json	Dec 10, 2023, 7:35:38 PM	Standard	Dec 10, 2023, 7:35:38 PM	Not
book5_mapper2.json	307.9 KB	application/json	Dec 10, 2023, 7:35:39 PM	Standard	Dec 10, 2023, 7:35:39 PM	Not

Intermediate map json output for book5 –

, 1, 1]}, "country": {"book5.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "parts": {"book5.txt": [1,
": {"book5.txt": [1, 1]}, "darling": {"book5.txt": [1, 1]}, "frenchmen": {"book5.txt": [1]}, "kind": {"book5.txt": [1, 1, 1, 1, 1,
, 1]}, "phrase": {"book5.txt": [1]}, "suit": {"book5.txt": [1]}, "yourself": {"book5.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "boulanger": {"book5.txt": [1]}, "names": {"book5.txt": [1, 1, 1, 1, 1, 1, 1, 1]}, "those"
1, 1, 1, 1, 1, 1, 1]}, "boys": {"book5.txt": [1, 1, 1, 1, 1, 1]}, "dear": {"book5.txt": [1, 1, 1, 1, 1, 1, 1]}, "fellows": {"book5.
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "laughed": {"book5.txt": [1, 1, 1, 1, 1, 1, 1]}, "poverty
, "hard": {"book5.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "aground": {"book5.txt": [1]}, "breton": {"book5.txt": [1]}, "village
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}, "us": {"book5.txt": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

Reducer file output –

Intermediate reducer output –

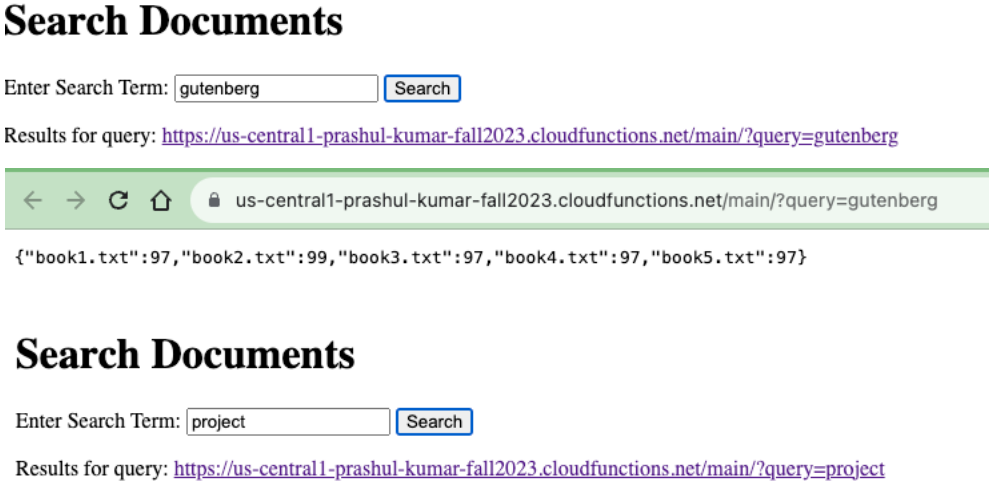
{"a": {"book1.txt": 152, "book2.txt": 344, "book3.txt": 3015, "book4.txt": 762, "book5.txt": 3167}, "modest": {"book1.txt": 5, "bool
{"book1.txt": 44, "book2.txt": 51, "book3.txt": 844, "book4.txt": 604, "book5.txt": 1419}, "united": {"book1.txt": 15, "book2.txt":
"book3.txt": 5065, "book4.txt": 840, "book5.txt": 5245}, "most": {"book1.txt": 14, "book2.txt": 18, "book3.txt": 78, "book4.txt": 2
"book5.txt": 2}, "whatsoever": {"book1.txt": 2, "book2.txt": 3, "book3.txt": 3, "book4.txt": 2, "book5.txt": 3}, "you": {"book1.txt"
"book2.txt": 15, "book3.txt": 50, "book4.txt": 8, "book5.txt": 94}, "license": {"book1.txt": 18, "book2.txt": 18, "book3.txt": 19, "
"book5.txt": 9}, "not": {"book1.txt": 44, "book2.txt": 204, "book3.txt": 883, "book4.txt": 125, "book5.txt": 936}, "laws": {"book1.
"book2.txt": 11, "book3.txt": 10, "book4.txt": 8, "book5.txt": 12}, "1": {"book1.txt": 47, "book2.txt": 46, "book3.txt": 47, "book4.
"book5.txt": 10}, "10": {"book1.txt": 1, "book4.txt": 22, "book5.txt": 10}, "preventing": {"book1.txt": 1, "book5.txt": 1}, "childr

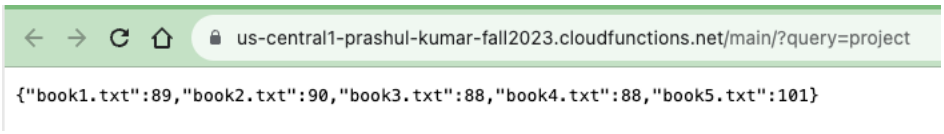
Final Combined_json output –


```
{
  "the": {
    "book1.txt": 357,
    "book2.txt": 1348,
    "book3.txt": 8241,
    "book4.txt": 3579,
    "book5.txt": 7002
  },
  "project": {
    "book1.txt": 89,
    "book2.txt": 90,
    "book3.txt": 88,
    "book4.txt": 88,
    "book5.txt": 101
  },
  "gutenberg": {
    "book1.txt": 97,
    "book2.txt": 99,
    "book3.txt": 97,
    "book4.txt": 97,
    "book5.txt": 97
  },
  "ebook": {
    "book1.txt": 13,
    "book2.txt": 14,
    "book3.txt": 13,
    "book4.txt": 13,
    "book5.txt": 13
  },
  "of": {
    "book1.txt": 259,
    "book2.txt": 557,
    "book3.txt": 4140,
    "book4.txt": 1996,
    "book5.txt": 3418
  },
  "for": {
    "book1.txt": 72,
    "book2.txt": 193,
    "book3.txt": 987,
    "book4.txt": 204,
    "book5.txt": 1054
  },
```


Now we can see in the final combined output that book5 is attached to all the words.
Thus, on just uploading the book5.txt the trigger function was automatically invoked resulting in running the whole pipeline again without any manual code.

Front-end:





Cost Estimate –

Cloud Function Invocation:
Invoking Cloud Functions incurs costs based on the number of invocations and execution time.

Cloud Storage Operations:
Reading, writing, and storing data in Cloud Storage involve costs.
Utilizing GCS in built file compression to reduce the file size can reduce the cloud storage costs for big data sets.

Cloud Storage Bucket Costs:
The number and size of objects stored in Cloud Storage impact costs. This can be improved by removing unnecessary files and only keeping the bare minimum and keeping a constant check on this.

Data Transfer Costs: Transferring data between Cloud Storage, Cloud Functions, and external services incur costs. The aim is to minimize unnecessary data transfers across different regions and utilize the same region.

Threading and Parallelization: It leads to more efficient resource utilization but can increase concurrent execution costs. Thus, the idea is to optimize the number of threads and parallel task to fund the right balance. In this assignment I have implemented parallelization, but infact implementing a sequential operation would not have made that significant difference as the amount of data is not that big.

Cloud Function Memory and CPU: The amount of memory and CPU allocated to Cloud Functions affects costs. As of now the cloud functions memory and CPU allocation had been maxed out for the important parts such as map function and reduce function. But for non-important ones, such as trigger and master, the minimal resources have been used for cost saving.

Data Processing Costs: Data processing costs are incurred when executing complex operations.

Idle Resources: Unused resources, such as idle virtual machines or storage, also contribute to unnecessary costs. The idea here was to cleanup the bucket storage and stop the VM instance and remove storage from that as well after completing the assignment or at the end so that those can be minimized.

Below shows the average cost for last 3 days distributed across different functions which I used as part of this assignment –

Date › Service	Cost
December 10, 2023	$0.13
VM Manager	$0.01
Compute Engine	$0.10
Networking	$0.01
Cloud Functions	$0.02
December 9, 2023	$1.68
VM Manager	$0.05
Compute Engine	$1.31
Networking	$0.05
Cloud Functions	$0.26
December 8, 2023	$1.55
VM Manager	$0.05
Compute Engine	$1.37
Networking	$0.05
Cloud Functions	$0.07

Thus we can see the major costs are due to compute engine and cloud functions.