

CS553 – Cloud Computing

Programming Assignment 3

Performance Evaluation

Karan Jeet Singh; Kanwal Preet Singh; Alaa Ayach
11-8-2014

1 INTRO

In this assignment we have built a distributed storage system on top of Google App Engine, using the Google Cloud Storage Client Library.

2 DESIGN

The system supports the storage of arbitrary file using the key/value concept where the key will be the filename and the value will be the size of the file. The filename will be unique and the size of the file will be varying. We have used a payload generator for generating a payload consisting of varying file sizes from 1KB to 100 MB to be stored in the implemented non-trivial distributed system. Since it is a persistent storage system there is high latency overhead involved in the retrieval of the small files and in order to improve it we have used the memcache to cache the smaller files whereas the medium and large sized files will not be cached.

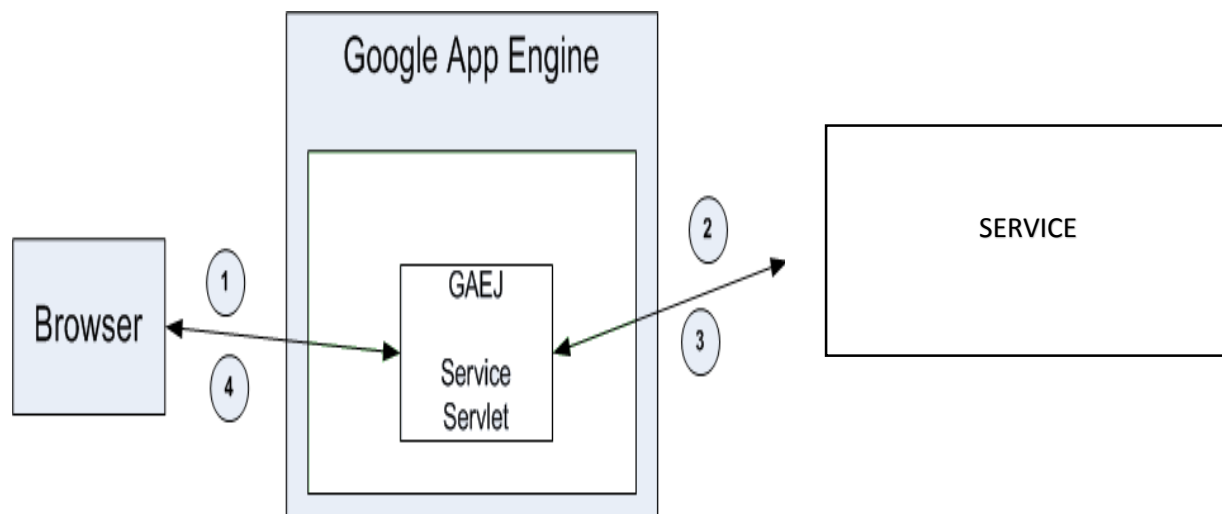
2.1 ABSTRACT

The files are uploaded using the blob to a temporary cache and from there we move the files either to the memcache or the Google Cloud Storage (GCS). File size is used to determine whether that file should be moved to the GCS or memcache.

Storing small files without utilizing the memcache.

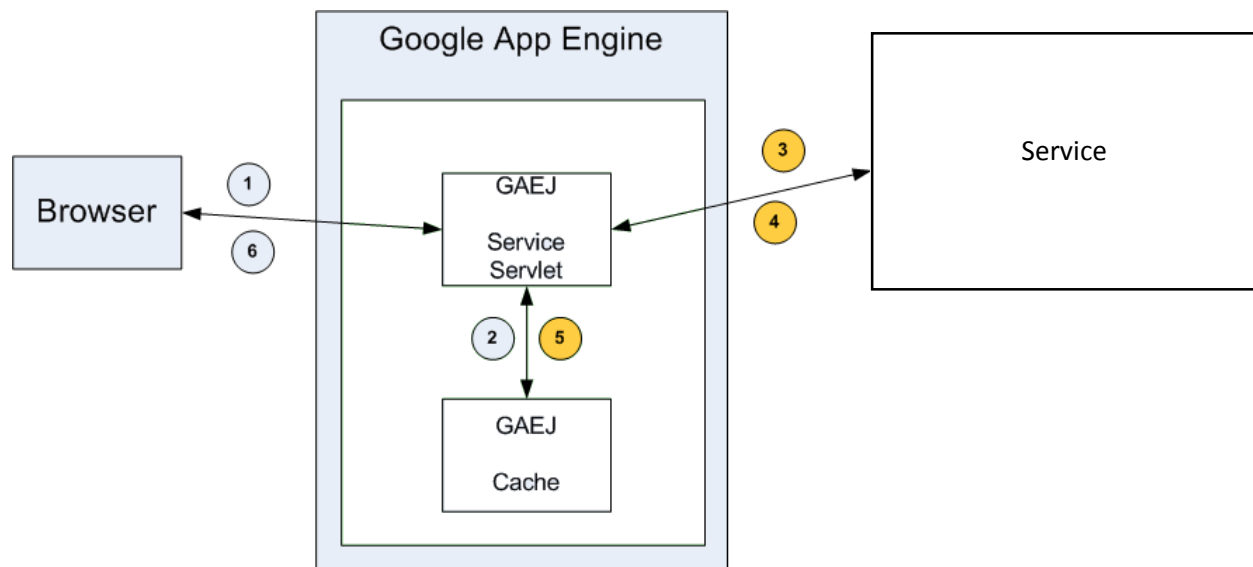
The first figure shows it **without Memcache**

- 1) The request
- 2) GCS access
- 3) GCS response
- 4) Result



Using the memcache for small file if

- i) Storing a new file
 - 1) The request
 - 2) Fetch Memcache
 - 3) If not in Memcache, GCS access
 - 4) GCS response
 - 5) Store result in Memcache if small file
 - 6) Result
- ii) Fetching a file
 - 1) The request
 - 2) fetch Memcache and send response
 - 6) Result



2.2 ASSUMPTIONS WHILE DESIGN

- Since the Memcache is not permanent we have to save the file both in the Memcache and the GCS so that the file is available for retrieval at a later stage.
- We have assumed that the user has initiated only a single session i.e. files will be uploaded from a single browser.
- The user will either use Mozilla Firefox, Google Chrome or Microsoft Internet Explorer to access the services as they support the jsp and HTML5

2.3 DESIGN TRADEOFFS

- Because of the assumptions mentioned above we have to perform the delete operation on both Memcache and GCS.
- Since uploading is handled at the client side rather than the server side, uploading the files is a single threaded process.
- To implement the multithreading upload process, the main process is put to sleep for 20 seconds in order to let the other threads complete the processing.
- Once the remove process has been initiated we cannot use the page as the page is unavailable till all the files are removed.
- File greater than 32MB cannot be uploaded using the blob store and if that needs to be done than the original file is divided into chunks with each chunk lesser than 32 MB.

Java Restrictions

There are many restrictions using Java in GAE some of these are:

- **Threads**
You can't use multithreading in GAE in the same way it is used in Java, we used instead **Background threads** as a way to implement the multithreading part of the assignment
- **File System**
A Java application cannot use any classes used to write to the file system, such as `java.io.FileWriter`. An application can read its own files from the file system using classes such as `java.io.FileReader`.

2.4 IMPROVEMENT AND EXTENSIONS TO THE CURRENT SYSTEM

- Using CROM jobs to handle removeAll

3 MANUAL

3.1 DESIGN DETAILS

1) Insert(key, value)

We implemented that by considering the key representing the file and the value will represent the content of this file. We used here Memcache as mentioned before for small files to **be stored in Memcache** and GCS for bigger files.

2) Check(key)

This will check both Memcache and GCS if the file mentioned is there. We made an interface for it as well.

3) Find(key)

This will check both Memcache and GCS if the file mentioned is there and then retrieve the content according to where it is. We made an interface for it as well. We retrieved the value and showed it in a new page in the website

4) Remove(key)

This will check both Memcache and GCS if the file mentioned is there and remove it from both. We made an interface for it as well.

5) Listing()

This will iterate the key value storage and retrieve all the file names. We made an interface for it as well.

6) checkCache(key)

This will check both Memcache only if the file mentioned is there. We made an interface for it as well.

7) RemoveAllCache()

This will check Memcache if the file mentioned is there and remove it. We made an interface for it as well.

8) RemoveAll()

This will iterate the key value storage and retrieve all the filenames and remove them from both. We made an interface for it as well.

9) CacheSizeMB()

This will sum up all sizes for the files stored in Memcache. We made an interface for it as well.

10) CacheSizeElem()

This will sum up the number for the files stored in Memcache. We made an interface for it as well.

11) StorageSizeMB()

This will sum up all sizes for the all files stored. We made an interface for it as well.

12) StorageSizeElem()

This will sum up the number for the files stored. We made an interface for it as well.

13) FindInFile(key, string)

This will check both Memcache and GCS if the file mentioned is there using a regular expression. We made an interface for it as well.

14) Listing(Sring)

This will iterate the key value storage and retrieve all the file names using a regular expression. We made an interface for it as well.

3.2 SETUP INSTRUCTIONS

3.2.1 Client Side

- Generation of the payload using the FileGenerator.jar file
- From the command line run the following command
java -jar FileGenerator.jar "Path to the output folder"
(the path to the output folder should be followed by a '\' symbol to dump the generated files into the specified folder)

3.2.2 Server Side

- Create a project (with any name)
- Then create a bucket name ass3gs

Use following link to access the website.

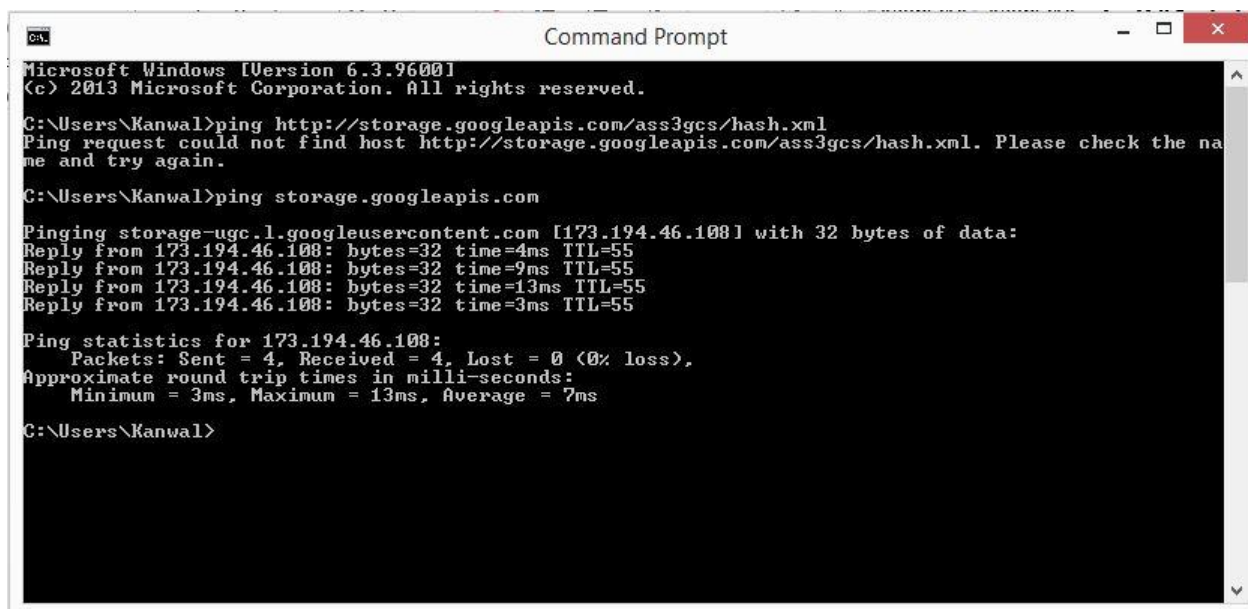
<http://1-dot-singular-object-754.appspot.com/>

4 PERFORMANCE EVALUATION

4.1 GOOGLE APP ENGINE

- **Latency**

The average latency in from client machine to the GCS is 7ms.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Kanwal>ping http://storage.googleapis.com/ass3gs/hash.xml
Ping request could not find host http://storage.googleapis.com/ass3gs/hash.xml. Please check the name and try again.

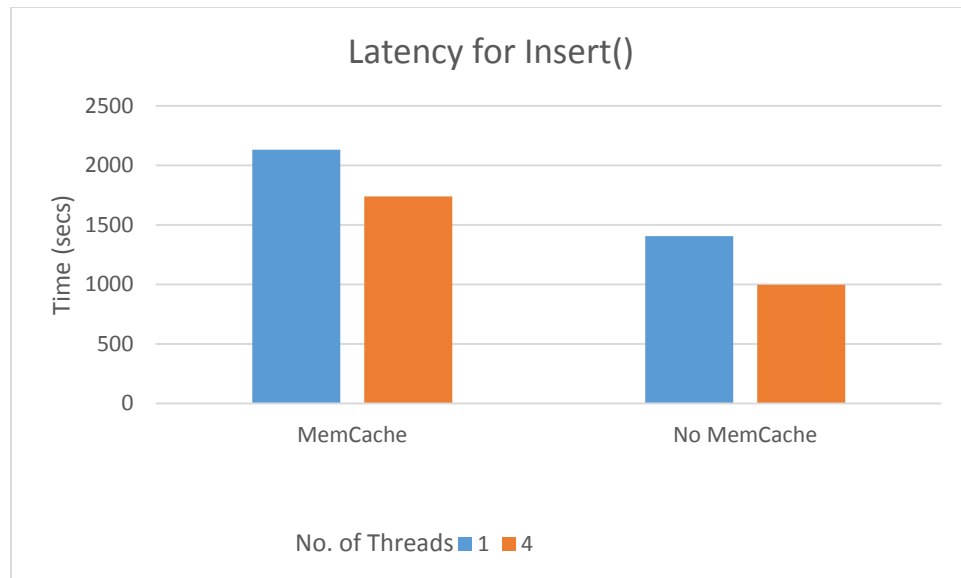
C:\Users\Kanwal>ping storage.googleapis.com

Pinging storage-ugc1.googleusercontent.com [173.194.46.108] with 32 bytes of data:
Reply from 173.194.46.108: bytes=32 time=4ms TTL=55
Reply from 173.194.46.108: bytes=32 time=9ms TTL=55
Reply from 173.194.46.108: bytes=32 time=13ms TTL=55
Reply from 173.194.46.108: bytes=32 time=3ms TTL=55

Ping statistics for 173.194.46.108:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 13ms, Average = 7ms

C:\Users\Kanwal>
```

Insert operation with Memcache ON and OFF on 1 thread and 4 threads



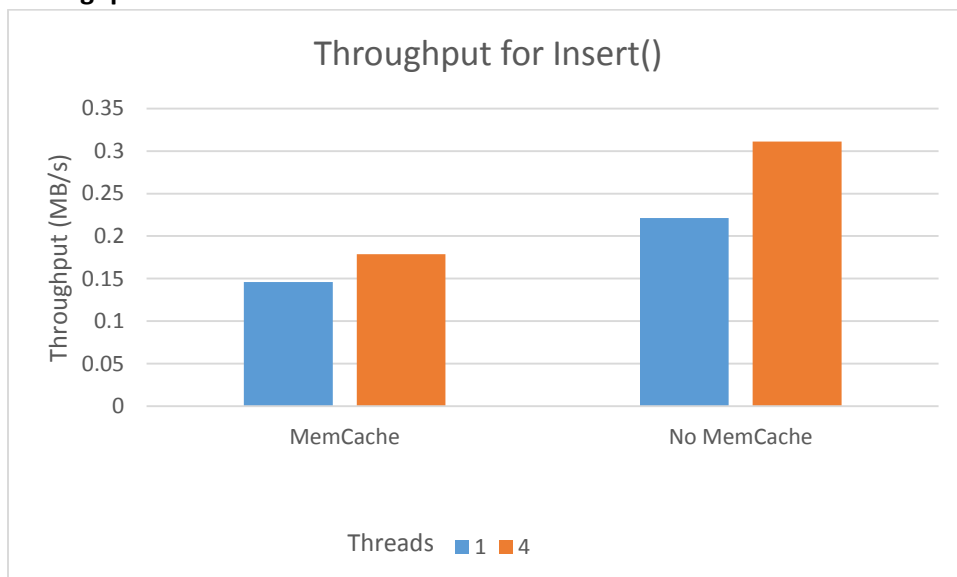
The graph above shows the total insertion time when the memcache is enable and when it is disabled.

The insert operation is done on single thread as well as the four threads.

As we can see that the performance is better in case of without Memcache when we compare the results of the insert operation with memcache and without Memcache. This is due to the system design , we are storing the data in both GCS and Memcache. As a result the performance is lower in case of the insertion operation with Memcache enabled.

As it can be seen in the graph the performance is better in case of multithreaded input operation.

- Throughput



The graph above shows the throughput during the insert operation when the memcache is enable and when it is disabled.

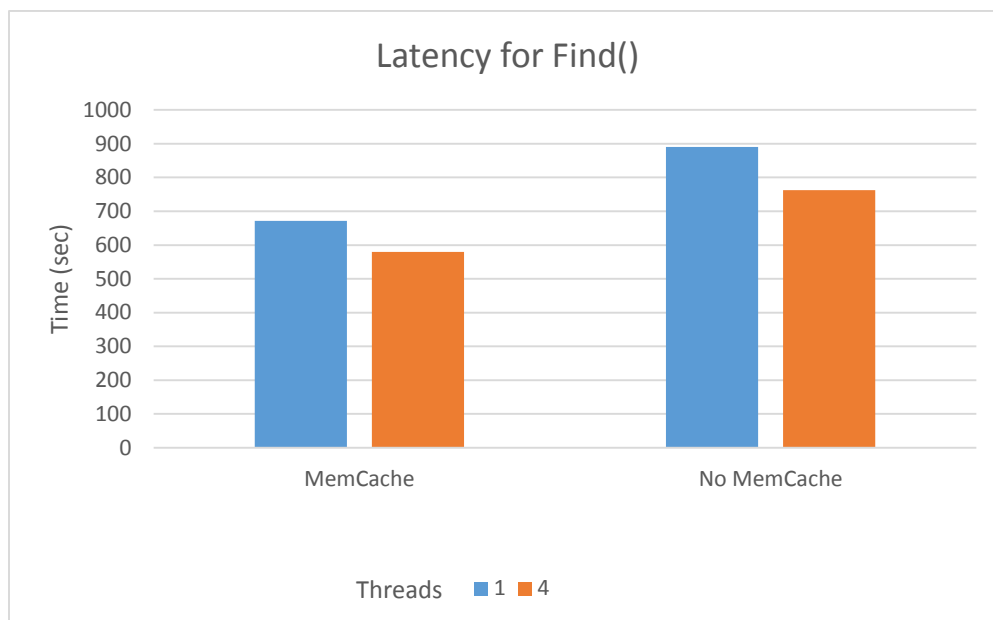
The insert operation is done on single thread as well as the four threads.

As we can see that the performance is better in case of without Memcache when we compare the results of the insert operation with memcache and without Memcache. This is due to the system design , we are storing the data in both GCS and Memcache. As a result the performance is lower in case of the insertion operation with Memcache enabled.

As it can be seen in the graph the throughput is higher in case of multithreaded input operation.

- **Find Operation**

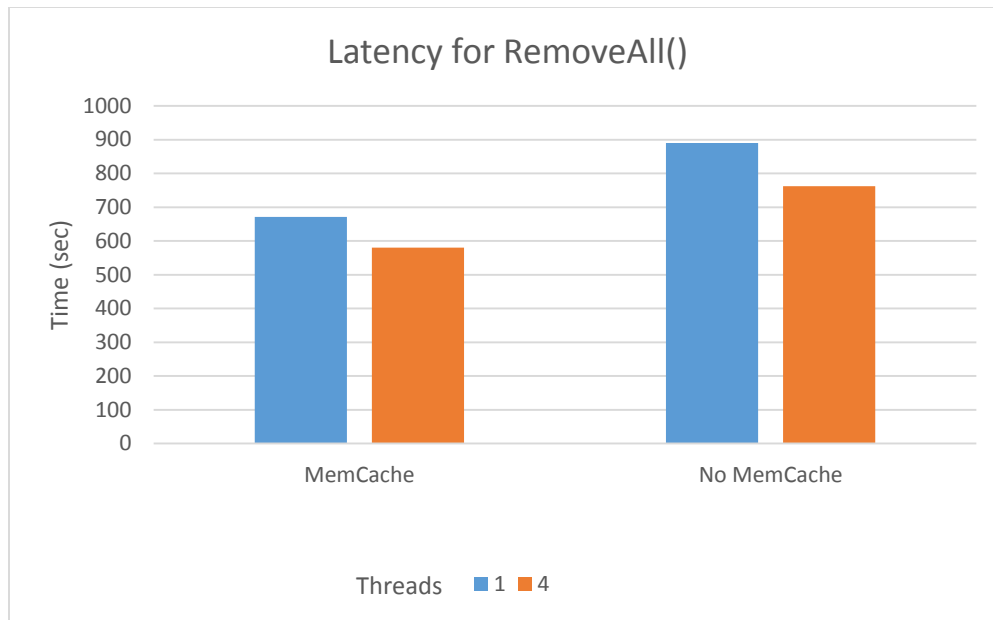
As it can be seen from the graph the performance is better in case of Memcache enabled as the files are first searched in the MemCache and then in the GCS.



- **Remove Operation**

As it can be seen from the graph the performance is better in case of MemCache disabled as in case of MemCache enabled the files are removed from the GCS as well as the MemCache.

Where as in case of MemCache disabled we need to remove files only from the GCS.



4.2 AMAZON S3

Assuming that the work load is increased to 311TB. We need to make a distributed storage system of storage capacity if 311TB. Assuming we make a cloud cluster consisting of 311 c3.large instances with an additional 311TB of space. The cost of such a setup would be \$173.86 for a month for all the c3.large instances and the storage space. For every 1 TB of space we get 3000 IOPS free. Assuming that the IOPS remain in the free quota of the 311TB limit there will be no extra cost incurred of the same.

Hence in case of the Amazon S3 implementation the cost per month will \$173.86 with the above assumptions in mind.

amazon web services

SIMPLE MONTHLY CALCULATOR

Language: English

Need Help? [Watch the Videos](#) or [Read 'How AWS Pricing Works' Whitepaper](#)

AWS can help you reduce your overall IT costs in multiple ways. [Learn more about our Pricing Philosophy](#)

FREE USAGE TIER: New Customers get free usage tier for first 12 months

Reset All

Services

Estimate of your Monthly Bill (\$ 173.86)

Choose region: US-West-2 (Oregon)

Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances.

Compute: Amazon EC2 Instances:

Description	Instances	Usage	Type	Billing Option	Monthly Cost
Linux on c3.large	1	100 % Utilized/Mor	Linux on c3.large	On-Demand (No Co	\$ 76.86

Add New Row

Storage: Amazon EBS Volumes:

Description	Volumes	Volume Type	Storage	IOPS	Snapshot Storage
General Purpose (SSD)	1	General Purpose (SSD)	1000 GB	3000	0 GB-month of Storage

Add New Row

Elastic IP:

Number of Additional Elastic IPs: 0

Elastic IP Non-attached Time: 0 Hours/Month

Number of Elastic IP Remaps: 0 Per Month

Data Transfer:

Inter-Region Data Transfer Out: 0 GB/Month

Common Customer Samples

Free Website on AWS

AWS Elastic Beanstalk Default

Marketing Web Site

Large Web Application (All On-Demand)

Media Application

European Web Application

Disaster Recovery and Backup

4.3 COMPARISON WITH GOOGLE APP ENGINE

Pricing

Resource billing rates

Resource	Unit	Unit cost (in US \$)
Instances*	Instance hours	\$0.05
Outgoing Network Traffic	Gigabytes	\$0.12
Incoming Network Traffic	Gigabytes	Free
Datastore Storage	Gigabytes per month	\$0.18
Blobstore, Logs, and Task Queue Stored Data	Gigabytes per month	\$0.026
Dedicated Memcache	Gigabytes per hour	\$0.06
Logs API	Gigabytes	\$0.12
SSL Virtual IPs** (VIPs)	Virtual IP per month	\$39.00
Sending Email, Shared Memcache, Pagespeed, Cron, APIs (URLFetch, Task Queues, Image, Sockets, Files, and Users)		No Additional Charge

Since the incoming of data is free there will be no cost incurred for that. While the cost of retrieval of data will be \$37,320. Since there is cost included for the dedicated MemCache, Blobstore and Datastore Storage included which also needs to be considered.

CONCLUSION

We can see from the above figures that the Amazon S3 will be a much less expensive storage option as compared to the GCS.