**Programming Assignment 3**

**Google App Engine:** **DESIGN DOCUMENTATION**

Benchmarks to be done:

* Transferring or Uploading Dataset to Google Cloud Storage
* Verifying if the file name exists in the cache or distributed Storage system.
* Retrieve and display the content of any given file from cache or distributed Storage system.
* Removing any given file from cache or distributed Storage system.
* Retrieving the list of all files as an array.

Design of Programs is as follows

1. Files Generation:

With the help to **generatefile.java** we can generate 411 files of spanning 311MB and the files are given random file names of 10 characters each. The program also generates 100 files of 1KB size, 100 files of 10KB size, 100 files of 100KB size, 100 files of 1MB size, 10 files of 10MB size and 1 file of 100MB size.

1. Uploading Files:

By using **FileUpload.java** servlet uploading the files to memcache and blobstore using their respective API’s specified by Google. Since Memcache cannot store files if size is more than 1MB, files with size less than 1 MB is uploaded to memcache and all the other files greater then 1MB size are uploaded to Blobstore.

1. Searching Files:

By using **CheckFile.java,** it finds the input file named in cache or distributed storage and displays the appropriate messages on successful search. The flow of program as below,

* + Get the filename using **getFilename** function and iterate over every filename to find the matching file. Once the file is found print the message.

1. Displaying File Contents:

By using **FileContents.java** we will display the content of a filename that is entered by user from the data storage. The flow of program as below,

* + Get the filename from the user using getParameter.
  + Find the location of the file (memcache or blobstore).
  + Use FileServices and AppEngineFile classes to display the content of the file from blob using the command readableFile = fService.getBlobFile(blobToRead.get(i).getBlobKey());.
  + Open the read channel using command,

FileReadChannel readChannel = fService.openReadChannel(readableFile, false); Where false specifies the keylock i.e. file is accessible simultaneously by more than one users. The printing is done using the res.getWriter().println(line); function.

* + Finally close the read channel.

1. Deleting Files:

By using **Remove.java,** deletes the filename entered by user from data storage. The flow of the program as below,

* + Get the filename from the user using getParameter.
  + If the file in blobstore, use command blobstoreService.delete(blobToList.get(i).getBlobKey()); and if in memcache using command

MemcacheService cache = MemcacheServiceFactory.getMemcacheService(); and cache.delete(name);

1. List All Files:

By using **ListAllFiles.java,** it displays the list of all files in data storage. The flow of program as below,

* + First it gets the size of Blob and then it will iterate over every file and gets its respective file name using the function **getFilename** and prints the filename.

1. Extra Credits:
2. Find File in Storage:

By using **CheckFileInStorage.java,** finds whether the file is in Distributed Storage And the method used for this is fileNameInStorage =dataBlobList.get(i).getFilename();.

1. Find File in Memcache:

By using **CheckFileInCache.java,** finds whether the file is in Memcahe or not.

And the method used for this is cacheFile = (AppEngineFile) memCacheService.get(searchFileInCache);

1. Remove Files from Cache:

By using **RemoveCache.java,** remove all the files in Cache. And the method we used here is memCacheService.clearAll();

1. Remove Files from Cache and Distributed System:

By using **RemoveAll.java,** remove all the files from Cache and Distributed Storage. To delete from Cache the method used is memCacheService.clearAll(); And to delete from Distributed System method used is blobStoreService.delete(dataBlobList.get(i).getBlobKey()).

1. Check MemCache Space Used:

By using **MemCacheUsedSize.java,** we can check the size of Cache used.

The method used is sizeinBytes=memCacheService.getStatistics().getTotalItemBytes();

1. Elements in Cache:

By using **MemCacheElements.java,** we can check the number of elements in Cache. The method used is TotalElements=memCacheService.getStatistics().getItemCount();

1. Check Distributed Storage Space Used:

By using **DistributedStorageSize.java,** we can check the total size used to store the data in distributed storage. For this we find the size of each file and then iterate and sum of all the values. Here the method used is TotalBytes += dataBlobList.get(i).getSize();

1. Total Elements in Distributed Storage:

By using **DistributedStorageSize.java,** we can check the number of elements in Storage.

The method used is dataBlobList.size()); to get the value for it.

1. Regular Expression in File:

By using **FindStringInFile.java,** it searches for regular expressions in given filename by user.

**Possible Improvement and extensions to the Systems:**

The performance of the programs script could be improved by including:

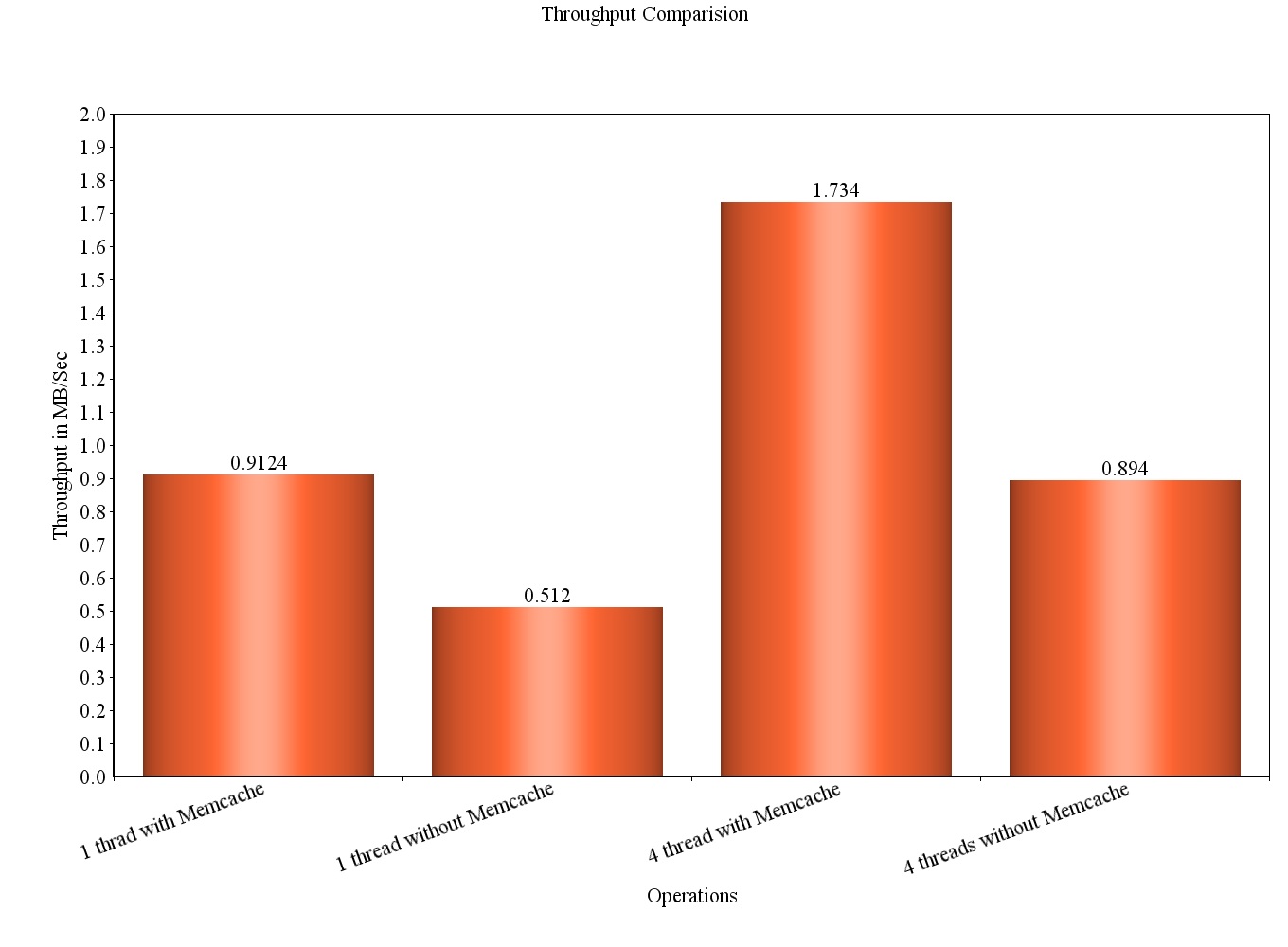
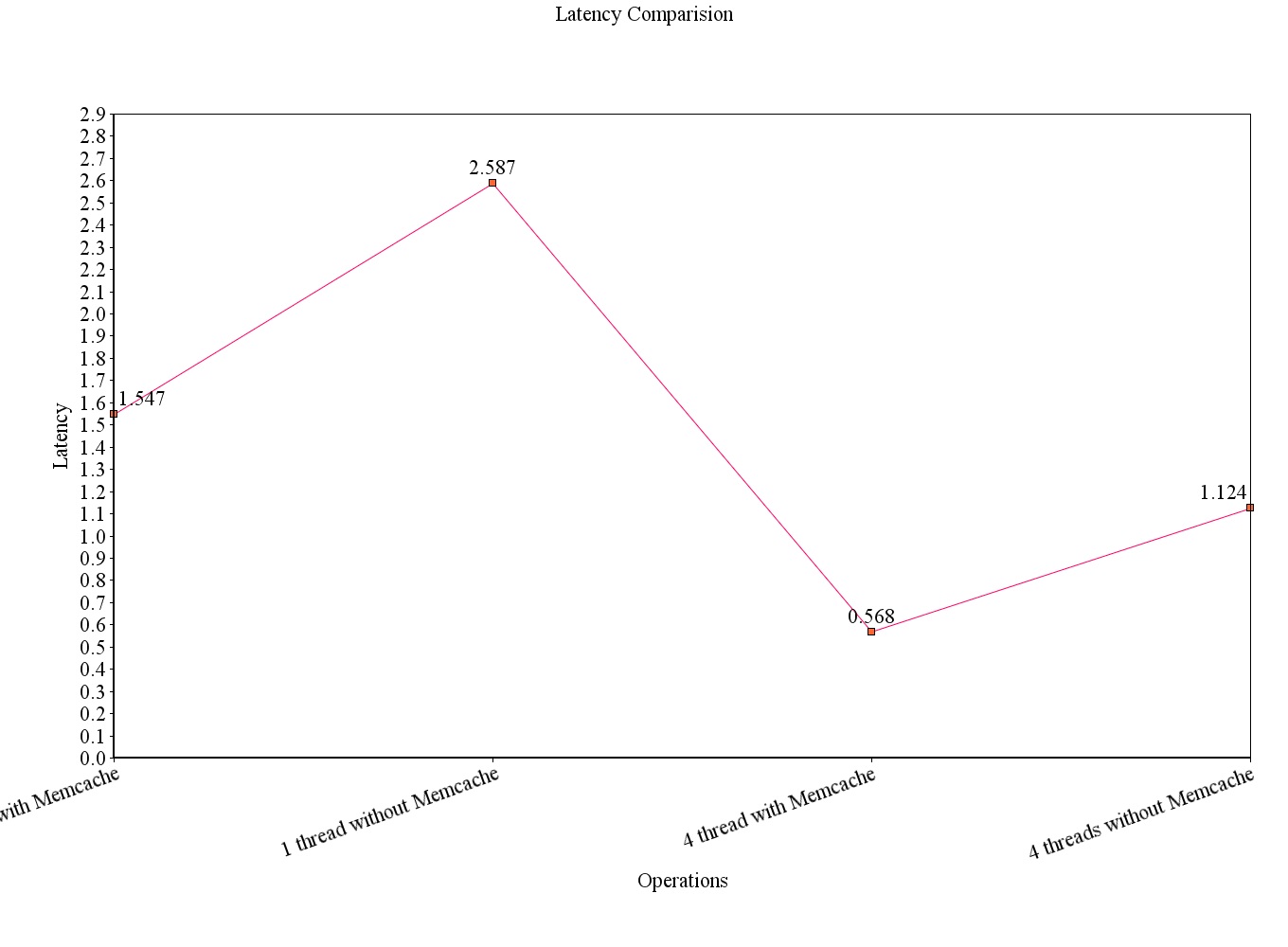
1. More number of threads, this would lead in increase in speed of execution.
2. Increasing the internet bandwidth which may enhance the speed of data upload and download and would incur less elapsed time.

**Performance Document**

**Performance/Benchmarking for Google App Engine**

The performances for different experiments are as follows:

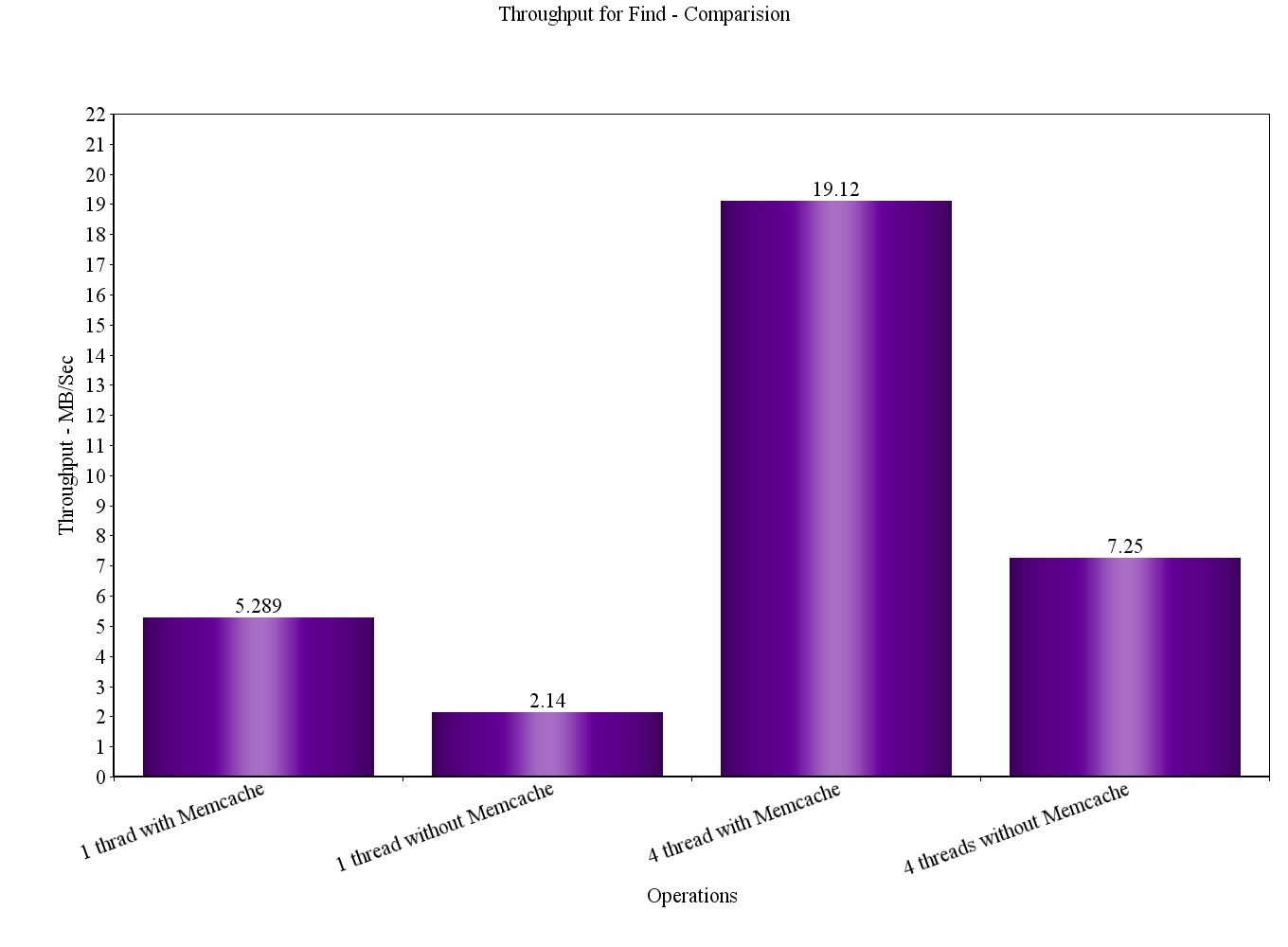
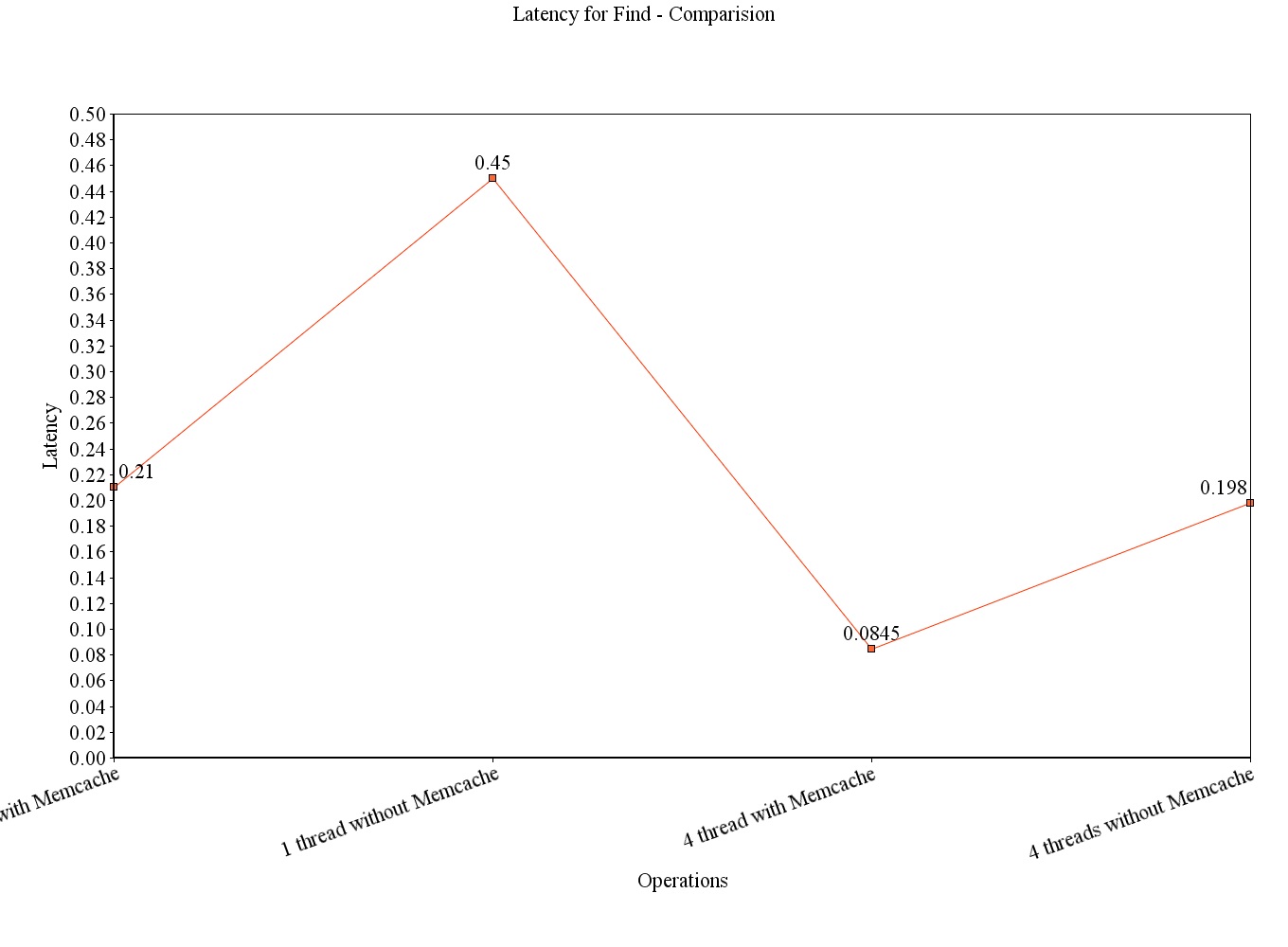
**Insert Operations:**



The insert operation with memCache for thread is more when we compare with without MemCache for 1 thread as we are not storing large files (i.e > 1MB) in memCache. Only memCache will accept less than 100KB file sizes. The Distributed storage will store large number of files and the throughput for that is high. When we implement the threads the performance will increase for both with MemCache and without MemCache but the throughput is less for without memcache when compared with MemCache option.

The Latency also calculated for all experiments. The latency is less when we use with MemCache compared to without memCache in both 1 thread and 4 threads.

**Find Operation:**

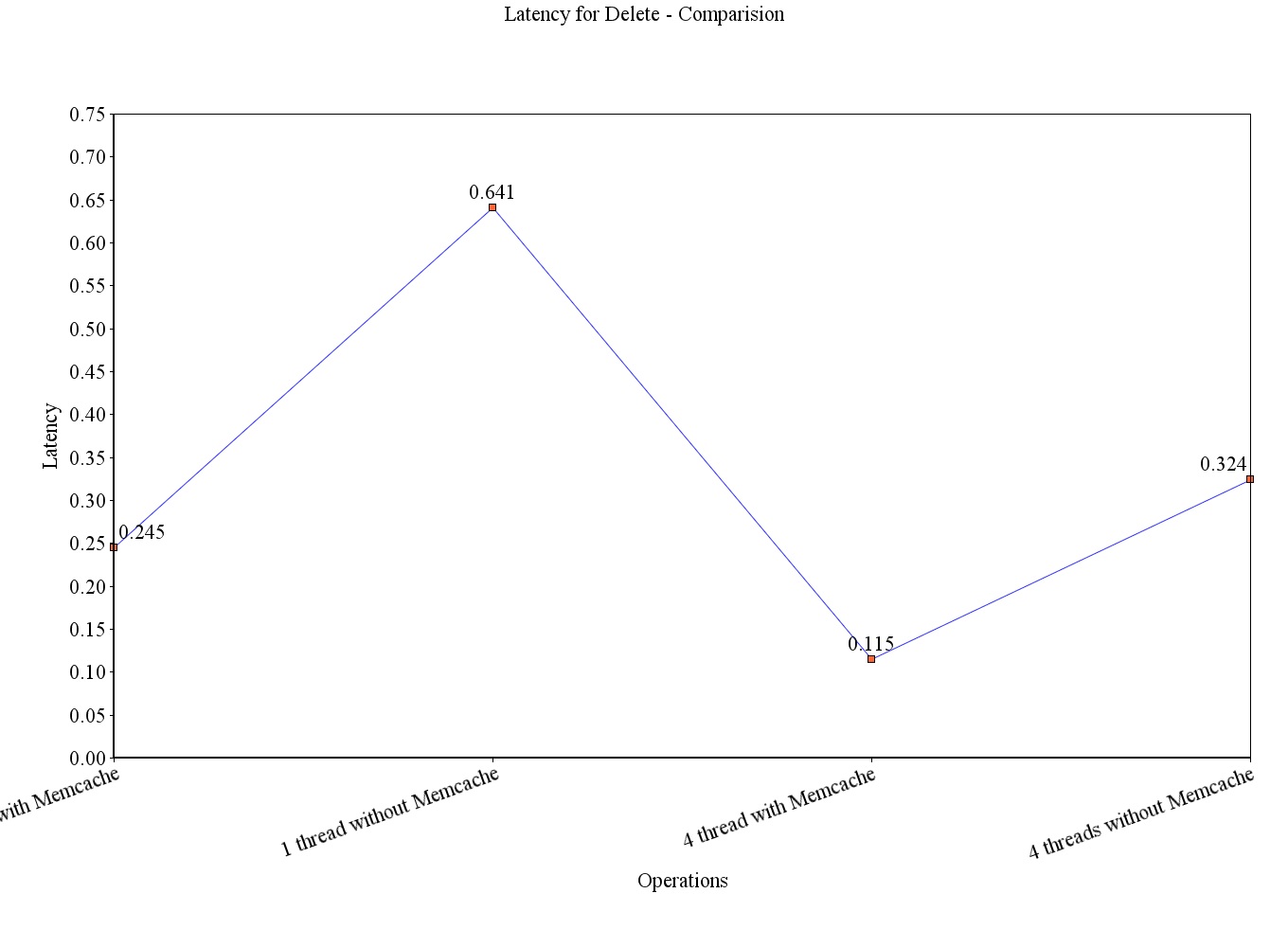
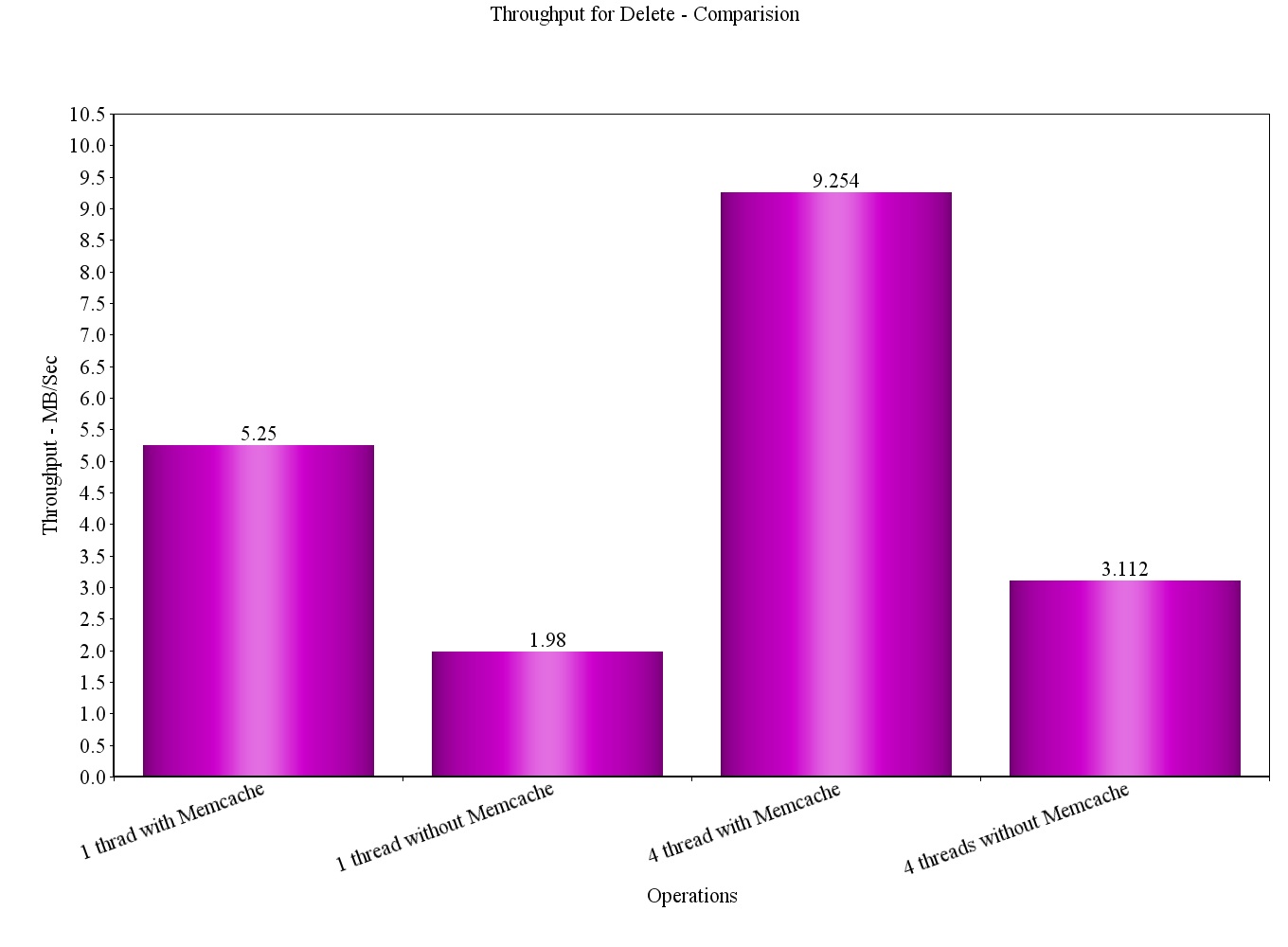


From the above graph we can notice that throughput with memcache is more than through put without memcache, this is because first it will search in memcache and if the file is not there then only it will again redirects to Distributed Storage to retrieve the file contents.

Since Latency is inversely proportional to throughput we have high latency for threads without MemCache.

So user enters the file to be searched, which is searched in entire cloud storage. The searching is relatively faster with the Memcache (Since it searches first here)

**Delete Operations**



From the above graph we can notice that throughput for delete operation with memcache is more than throughput without memcache, The reason is, we have to remove files from both blobstore and memcache. Generally blobstore is bit slower when it comes to deletion using APIs when compared with the deletion from Memcache operations.

**Comparison with Amazon S3**

Q. Simple analysis of which cloud platform is the most cost effective to run a distributed storage system on, Amazon S3 or Google App Engine?

Given: Initial number of Dataset file 411, now increasing the workload to 1 million times, we have

* 411 million dataset file.
* 311 TB size of data which is equivalent to 311\*1024 = 318,464 GB of data.
* And the workload is spread out over 1 month time (30 days on average).

**Let us First Consider Amazon S3 Storage System:**

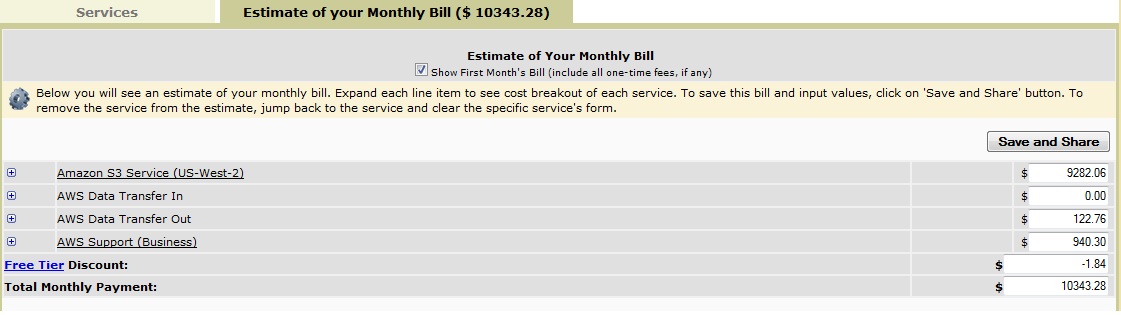
The US Standard pricing list for Amazon S3 can be found at http://aws.amazon.com/s3/pricing/ **,** we have calculated the costs in dollar.

* Data Transfer To Amazon S3:

So we have $0.000 per GB

For 311 TB = **$0 (No Charges)**

* Data Storage (Standard) in Amazon S3 rates are as below:



Now let us calculate using Amazon Simple Monthly Calculator from the link: http://calculator.s3.amazonaws.com/index.html

Amazon S3 Services (Excluding AWS Support Services charges) = $ **10343.28**

* Retrieve the Data from Amazon S3

Total size of data set to retrieve = 622 TB

Now let us calculate using Amazon Simple Monthly Calculator from the link: http://calculator.s3.amazonaws.com/index.html

Amazon S3 Services (Excluding AWS Support Services charges) = **$ 42354.7**

Now for Google App Engine Storage System:

The US Standard pricing list for Google App Engine Storage System is found in https://developers.google.com/storage/docs/pricingandterms#pricing, we have computed the costs in dollars.

* Like Amazon Data Transfer IN to Google Cloud Storage is also Cost **$0**

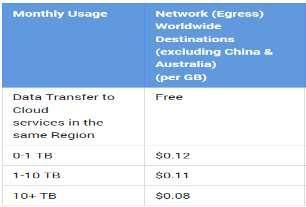
So we have $0.000 per GB

For 311 TB = **$0 (No Charges)**

* Data Storage in Google Cloud Storage Since 311 TB = 311 \* 1024 = 318,464 GB So 318,464\*.026= **8280.064**

Total Standard Storage Cost = **$8280.064**

* Cost to Retrieve 622TB Data from Google Cloud Storage:



Americas and EMEA

1TB (1024GB) at $0.12/per GB: $122.88 9TB (9216GB) at $0.11/per GB: $1,013.76

612TB (626688GB) at $.08/per GB: $50135.04 Therefore total cost is **$51271.72**

So if we take into all three factors like Uploading 311 TB of data, storing it for 30 days and Retrieve the 622TB of Data,

If we use Amazon S3 it will take: **$52,697.98** in total.

If we use Google Storage it will take: **$59,551.78** in total.

**Conclusion:** By so observing all the statics we can say that Amazon S3 will offer good rateswhen compared to Google Cloud Storage. And based on above numbers one can say that Amazon is 12% cheaper than Google Storage.