Understanding and Taming the Variability of Cloud Storage Latency

Gurneet Kaur Umesh Moghariya John Reed

June 14, 2013

Contents

1 **Introduction** 1

2 **Related Work** 2

3 **Design & Implementation** 2

4 **Experimental Setup** 4

5 **Evaluation** 4

6 **Conclusions** 10

1 Introduction

With the emergence of cloud computing and its advantages over the other systems, organizations of various sizes are increasingly looking to utilize cloud-storage services, especially for storing large quantities of data. Cloud storage services for consumers and companies are currently offered by many different providers. They can roughly be divided into commercial and consumer service providers. Services offered by Amazon (S3) and Microsoft (Windows Azure Blob Storage, abbreviated from now on as WABS) are generally targeted at companies with relatively large storage needs (up to thousands of terabytes per customer), therefore referred to as commercial service providers. Companies such as Dropbox and Google[1] (Google Drive) are generally designed to attract consumers and businesses with relatively small storage needs. While Dropbox and Google Drive do offer plans for business customers, they will be referred to as consumer services in this report for convenience.

Each of the cloud-storage service providers covered in this project offer different packages for their services, and have data centers at various locations. Because the complexities of cloud-storage services are hidden from users, it can be difficult to anticipate or estimate cloud-storage performance. It is essential to understand the latencies involved in uploading or downloading the files in the cloud storage network which will help in making the choice of right service provider depending on the kind of data you want to store.

For this project we measured the variability of latency associated with using cloud-storage services — two commercial services (Amazon S3, and Windows Azure Blob Storage), and two consumer services (Dropbox, and Google Drive). The first part of this project was focused on measuring the dependence of latency on time (*e.g.* over the period of a week), and on file size. The second part of this project explored two different candidate techniques for reducing the latency and variability of cloud-storage services; one candidate technique used file compression to reduce PUT and GET latencies, and the other candidate technique used file replication to reduce the vari-

[1] Google also offers cloud-storage services designed for commercial customers (those that utilize cloud computing), but these services are not considered in this project. Google only opened it's Google Compute Engine for all customers on May 15, 2013 — too late to include in our measurements which started on May 12, 2013.

ability in latency. This report includes a brief summary of related work, details on the design and implementation of our latency measurement scripts, information on our experimental setup, a results of our evaluation, and conclusions.

2 Related Work

Replication is already used extensively by various cloud service provider to improve data integrity and in many systems to reduce latency. This technique is more useful when the size of the file to be transferred is small. As mentioned in [2], human-computer interaction studies show that people react to small differences in the delay of operations. Achieving low-latency consistently in a distributed framework has always been a challenging task. Task replication has always been useful in improving the response-time and alleviating the impact of slowest performing nodes.

There has been an enormous work in reducing latency through replication. This can be accomplished by replicating DHT queries and transmissions to multiple servers [4, 5], using multi-home proxy servers in which DNS queries are replicated to their local DNS server and connection establishment request is sent in parallel to all the servers [3], or by sending multiple copies of TCP SYNs to the same server on the same path and replicating DNS queries to multiple public servers over the same access link. It has been shown [1] that replication is a general technique that can be applied in a variety of common wide-area Internet applications.

There are various file compression techniques that are widely used to compress/decompress the file and send it over the network to reduce latency. Many modems even have compression algorithms built-in. It has been shown that one of the easiest solutions for limited bandwidth connections is to use file compression. It also provides maximum benefit for the resources that are required to provide storage efficiency.

3 Design & Implementation

Internal and public IP addresses are optionally recorded as identification numbers for the client machine in cases where multiple clients are used to evaluate a single cloud-storage service. In the case of each script (the latency measurement script and the scripts for each of the two candidate techniques), the test files were read from, and written to, system memory. Since all of the test clients were either Linux VMs or physical servers running Linux, `/dev/shm` was used to store all the test files in system memory; this prevented local storage from contaminating our cloud-storage latency measurements with fluctuations in hard drive performance. This is especially important for VMs located in public clouds due to the potential for local storage to be utilized by many different customers at once.

The first set of test files were created using pseudo-random data from `/dev/urandom/` with sizes of 1kB, 10kB, 100kB, 1MB, and 10MB; these are our binary data files. In each case, the binary test files were placed in `/dev/shm/`[2] in order to prevent files from being read from, or written to, shared or local storage. Because the binary test files created directly from `/dev/urandom/` did not compress very well, a separate set of text test files with sizes of 1kB, 10kB, 100kB, 1MB, 10MB, 25MB, and 50MB were created by base 64 encoding the binary data from `/dev/urandom/`.

All of the latency measurements were logged to .csv files that included a timestamp, type of request (GET or PUT), file-size (or other file identifier), and latency. A local machine that was not being used to run measurement scripts was used to backup the log files generated by each test script, every two hours. The resource requirements involved with backing up log files is assumed to be negligible.

[2] `/dev/shm/` is a file system mounted in virtual memory instead of persistent storage. It is commonly found on Unix-like operating systems.

3.1 Latency Measurement

Latency measurements for all four services started on 2013-05-12 02:35 UTC, and stopped on 2013-05-19 02:35 UTC. In all, one week of data was collected. Measurements on GET and PUT latency for each service were run approximately every 10 minutes. Each loop through the measurement script uploads and then downloads each test file; on average, this would add about 19kB/s of reads and writes to each service — low enough that we assume the effect of our measurements on the performance of each service is negligible.

The latency data for each of the services were measured and recorded using APIs for each of the services. Measurement scripts written in Python (depending on which language is best-supported by each service's existing API SDKs) were used to automate the process of latency measurement and logging.

The test files have known file sizes, so programmatic determination of file size is not necessary. We define latency associated as the time from issuing a GET or PUT request until the file has been transferred. By running our latency measurement programs at roughly the same time, and letting them run for a significant period of time — long enough to observe daily trends in latency — we were able to make comparisons between the variability in latency associated with each cloud-storage service.

3.2 Candidate Techniques

In order to reduce the latency of cloud storage file transfers, we evaluate two candidate techniques. One technique uses file replication to reduce variability in GET latencies, and the other uses file compression to reduce PUT and GET latencies.

3.2.1 File Replication

Latency measurements for GET requests were started on 2013-05-26 04:48 UTC, and stopped on 2013-05-28 08:53. The file replications latency measurement script logged measurements roughly every 10 minutes. For the redundancy technique, only GET latency is considered. Three copies of a file are uploaded to the cloud storage service. Then, three separate GET requests for the three identical files are issued in parallel. The GET latency when using the multiple GET requests can then be compared with single GET requests to determine whether or not it is beneficial. While only one file would be fully downloaded in practice, the script used to evaluate file replication downloads each of the three copies entirely in order to get overall latency measurements for each of the three GET requests. Using this data, we can make comparisons between the effect of using one, two, and three copies of a file on overall cloud-storage latencies.

3.2.2 File Compression

Latency measurements started on 2013-05-27 06:31 and stopped on 2013-05-27 13:59. Measurements were made roughly every 10 minutes. For the compression technique, files are compressed on the client using Python's `gzip` library before being uploaded to cloud storage; the `gzip` library's default compression level is a 9 — on a scale from 0 to 9, with 9 being the slowest and resulting in the highest data compression ratio. GET requests for this technique are not complete until the file has been decompressed. The time taken to complete the compression and decompression steps are included in the measurement of cloud-storage latency since the files are not assumed to be precompressed. Within the script, the PUT and GET latencies for the uncompressed files were also measured as a baseline for our comparison.

This technique is based on reducing latency by decreasing the quantity of data transmitted across network connections; however, the cost of this technique is that it requires more CPU time to perform the compression and decompression steps. Consumers that use services such as Drop-

box or Google Drive may have extra CPU cycles available to handle file compression and decompression. Companies that use VMs provided by Amazon or Microsoft to use either Amazon S3 or WABS, respectively, are charged per hour for their computing power; transferring large quantities of data that need to be compressed and decompressed may increase costs for these types of customers.

4 Experimental Setup

When possible, we made reasonable efforts to minimize the latencies involved with using each service. For Amazon S3 and Windows Azure Blob Storage (WABS), we ran the latency measurement scripts from virtual machines (VMs) running inside of each company's cloud. For Dropbox and Google Drive, latency measurement scripts were run on a server located in UCR's network in order to utilize UCR's internet connection.[3]

4.1 Amazon S3

Amazon S3 latency measurements are made using a Python-based customer app called `s3cmd`[4] — used to interact with the S3 service. The script was run in an Amazon EC2 instance located in the same region as the S3 container to minimize network latency between the measurement VM and the S3 service. The t1.micro Amazon EC2 VM instance is located in Amazon's us-west-2a region and has a shared virtual core and 613MB of memory. Oregon was chosen as the region for the Amazon S3 bucket. Ubuntu 12.04.2 was used as the OS for the VM.

[3] UCR has a 10Gb internet connection through CENIC (Corporation for Education Network Initiatives in California), see `cenic.org`.

[4] `http://s3tools.org/s3cmd`

4.2 Windows Azure Blob Storage

Windows Azure Blob Storage latency measurements were made using a script that was written in Python and is based on the official Python Azure SDK[5] from Microsoft. WABS latency measurements were made using a Small (A1) Windows Azure VM instance located in Microsoft's West US region; the Small instance has 1 virtual core and 1.75GB of memory. Ubuntu 12.10 was used as the OS for the VM. The WABS storage container is also located in the West US with geo-replication disabled.

4.3 Dropbox & Google Drive

The latency measurement scripts for Dropbox and Google Drive were each written in Python, with each using the Python API SDK provided by Dropbox[6] and Google,[7] respectively. The server used to run the scripts was `storm.engr.ucr.edu`. The `storm` server is a machine with Dual Intel Xeon (E5-2630 @ 2.30GHz) CPUs, 32GB RAM, a 1GbE connection to UCR's network, and runs CentOS release 6.3 as its OS.

5 Evaluation

Using the results we collected with our scripts (discussed in Section 3), we evaluated each of the services included in our week-long latency measurement tests, and the potential benefits of using compression, and file replication to reduce latency and variability.

[5] `http://www.windowsazure.com/en-us/develop/python/common-tasks/install-python/`

[6] `https://www.dropbox.com/developers/core/sdk`

[7] `https://developers.google.com/api-client-library/python/`

5.1 File Size Variability

5.1.1 PUT Latency

Median PUT latency measurements for several file types for each service are shown in Figure 1. Tabulated data for Figure 1 is shown in Table 1.

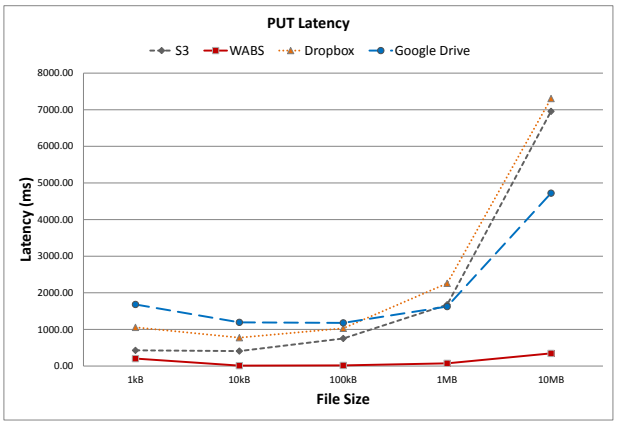

Figure 1: Median PUT latency measurements for a variety of file sizes on each service. The measurements were made over a period of one week and have been adjusted for each setup's RTT.

File Size	S3	WABS	Dropbox	Google Drive
1kB	428.40	203.21	1055.74	1682.09
10kB	409.40	10.86	774.23	1193.59
100kB	753.40	16.29	1030.69	1179.25
1MB	1676.40	73.65	2259.33	1624.51
10MB	6956.40	344.98	7306.86	4721.82

Table 1: Median PUT latency measurements for a variety of file sizes on each service. The measurements were made over a period of one week and have been adjusted for each setup's RTT. Units are in milliseconds.

Of all four services, WABS has the lowest PUT latency for all file types. This is likely due to a combination of low network contention with other VMs and possibly being in the same datacenter as the Azure storage servers. While the EC2 instance used to make latency measurements for S3 may have also potentially shared the benefit of being physically located close to the S3 servers, the EC2 instance had less available bandwidth. The Azure VM had an observed upload bandwidth of at least 248Mbps whereas the EC2 VM had an observed average upload bandwidth of roughly 96Mbps (spiking briefly at a maximum of 200Mbps).[8] This is likely due to the increased VM density on the server that the EC2 VM was located on. Because the Azure VM and EC2 VM were not created using comparable instance types (the Azure VM is larger), it is expected for the Azure VM to have less contention for network resources and more available bandwidth; this might account for Azure's relatively low latency measurements.

The latency for Dropbox and Google Drive vary with file size. For example, for the 1kB and 10kB files, the latency measurement for Dropbox is higher than that of the Google Drive; alternatively, Google Drive shows has a higher latency when issuing PUT requests for larger files. Also, it can be noted that the latency measurements for all the service providers are higher for uploading a 1kB file than a 10kB file. For file sizes larger than 10kB, the latency measurements for all tested storage services show an increasing trend.

Something that is not captured in the plots of PUT latency is the number of failed requests. In our experiments, Google Drive had a significantly higher failure rate of PUT requests for 1kB files than any other service — roughly 20%.[9] The failure rate of PUT requests for the other services was either nonexistent or negligible.

5.1.2 GET Latency

Median GET latency measurements for several file types for each service are shown in Figure 3. Tabulated data for Figure 3 is shown in Table 2.

Unlike PUT latency measurements, the GET latency measurements do not show a decrease in performance for the 1kB file; as file size increases, latency increases. The only exception to this is Google Drive, which has a marginally

[8] Upload bandwidth was estimated by copying an 800MB linux .iso from the test VM to another instance in the same region using `scp`.

[9] This high failure rate was only observed when issuing PUT requests for 1kB files.

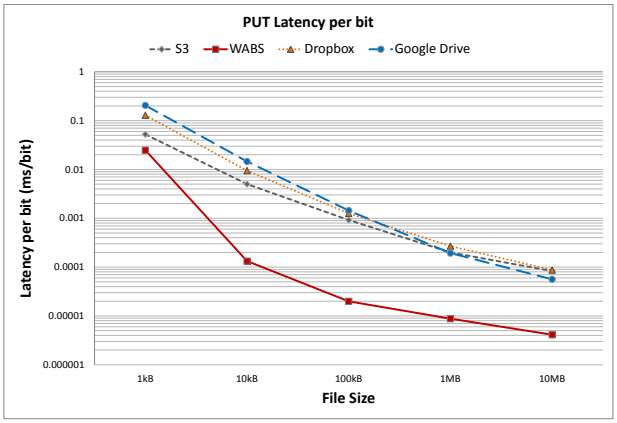

Figure 2: Median PUT latency per bit for each test file. The latency measurements were made over a period of one week and have been adjusted for each setup's RTT.

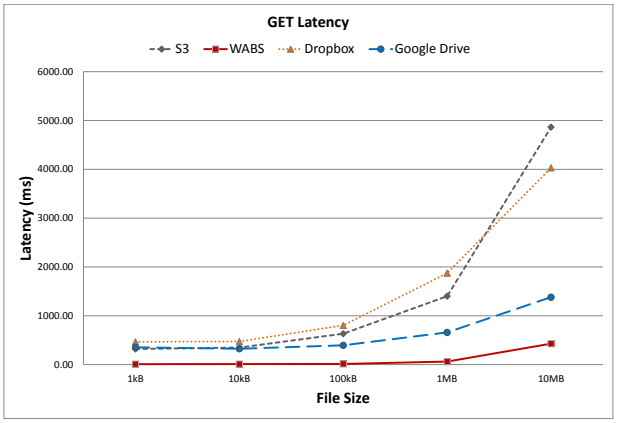

Figure 3: Median GET latency measurements for a variety of file sizes on each service. The measurements were made over a period of one week and have been adjusted for each setup's RTT.

File Size	S3	WABS	Dropbox	Google Drive
1kB	319.46	8.87	465.71	353.26
10kB	349.32	10.11	471.98	324.37
100kB	631.81	14.11	803.09	394.77
1MB	1401.66	61.58	1874.34	658.44
10MB	4863.93	427.90	4029.66	1380.19

Table 2: Median GET latency measurements for a variety of file sizes on each service. The measurements were made over a period of one week and have been adjusted for each setup's RTT. Units are in milliseconds.

smaller latency for the 10kB file relative to the 1kB file. This may have been influenced by the relatively high failure rate of PUT requests for the 1kB file; in the Google Drive latency script, if the the test file fails to upload after two requests, then the upload and the GET (since there is no file in cloud-storage to download) are abandoned for that file until the next test. With the smaller dataset for Google Drive, the calculated median for the 1kB GET latency might be more sensitive to occasional spikes in latency.

The latency of the GET requests for each service are lower, on average, than the latency of the PUT requests. If we only compare the consumer services, the measurements for Dropbox are better than Google Drive for the GET requests on all the file sizes. In this case, the latency of Dropbox is lower than the latency of Amazon S3, one of the commercial cloud-storage services. As the file size increases to 10MB, Amazon S3 latency increases above Dropbox latency. This is likely due to limited download bandwidth available to the EC2 VM. For all other file sizes where bandwidth limitations are not as significant, S3 has consistently lower latencies than Dropbox.

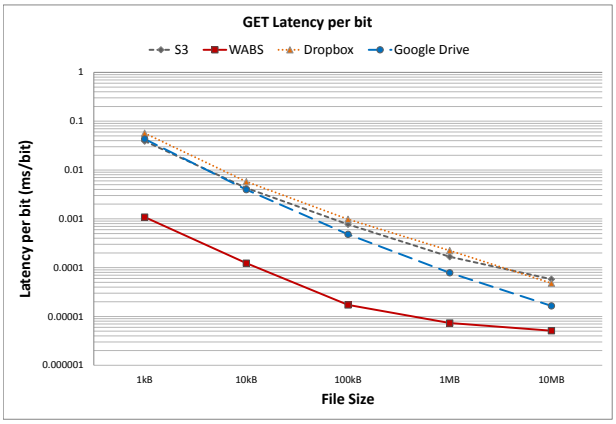

Figure 4: Median GET latency per bit for each test file. The latency measurements were made over a period of one week and have been adjusted for each setup's RTT.

5.2 Latency Variability Over Time

In order to evaluate how cloud-storage performance varies over the period of a week, we cal-

culated the median latency for 24 hour periods. Daily medians were used because the latency measurements for each service included in this project are highly variable; by using median values instead of mean values, the daily calculated averages are less sensitive to extreme outliers in the latency measurements.

It should be noted that the script for our dropbox measurements crashed at approximately 2013-05-13 08:44 UTC and was not resumed until approximately 18:23 UTC on the same day. This temporary failure of the script does not seem to have significantly affected the results of the latency calculations for that day; however, the daily medians for that day might potentially be higher than they actually were due to the failure occuring during the early to mid morning in the United States — a time during which usage of the service is expected to be relatively low.

Because of the large number of permutations for file size and type of transactions, not all of the time-variant latency plots are reproduced in this paper. A plot of the daily median PUT latency for the 1MB binary test file is shown in Figure 6. A plot of the daily median GET latency for the 1MB binary test file is shown in Figure 5. Plots for the 1MB file size were included in this paper — rather than plots for the other file sizes — primarily because we assume that 1MB is approximately the average size of files stored in cloud-storage services.

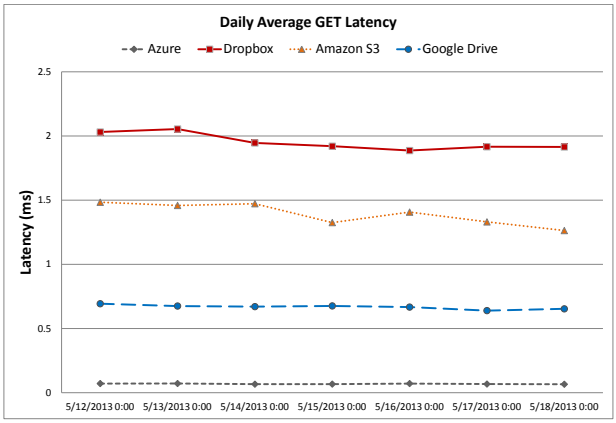

Figure 5: Median GET latency for the 1MB file over the period of one week.

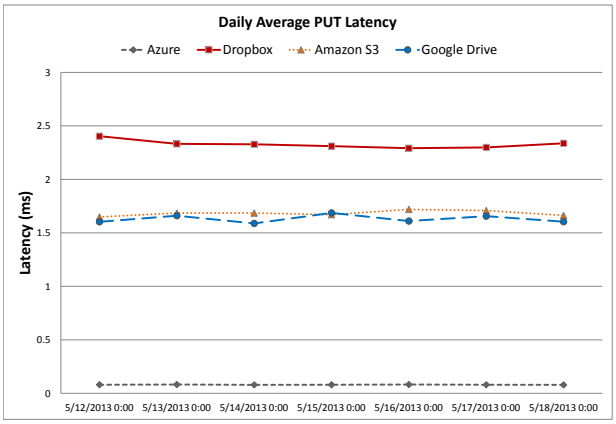

Figure 6: Median PUT latency for the 1MB file over the period of one week.

As can be seen in Figure 5 and Figure 6, WABS has the lowest latency of all of the services, followed by Google Drive, then Amazon S3, and finally by Dropbox. This is consistent with what is shown in Figure 1 and Figure 3. In general, the trends observed for the latencies observed when transferring the 1MB file are consistent with the trends in latency observed for the other file sizes.

While WABS and Google Drive have relatively consistent performance over the course of an entire week, both Dropbox and Amazon S3 show clear drops in latency of the course of a week for GET latency measurements. While the causes of these trends is unknown, it is clear that there are long-term (longer than one week) trends in latency for both Amazon S3 and Google Drive.

Unlike the gradual decrease in GET latency observed for Amazon S3 and Dropbox, PUT latencies for all services are relatively stable for the entire week. PUT latencies for Dropbox decrease slightly during the week, and increase slightly towards the weekends; this may be due to increased utilization of the dropbox service on the weekends. Amazon S3 and WABS do not show significant variation in latency over the measurement period. Google Drive latency measurements demonstrate a slight increase in latency during the middle of the week; latencies also drop slightly on before, and slightly after,

the middle of the week.

5.3 Reducing Latency: File Compression

Graphical comparisons of PUT and GET latencies with compression and without using compression are shown in Figure 7 and Figure 8, respectively. Compression is ideal for reducing latency for cloud-storage customers who have extra CPU time, but limited network bandwidth. `trickle` was used to simulate a client with a low-bandwidth, 1Mbps symmetric internet connection; this is representative of a cloud-storage customer who uses a standard T1 internet connection. Customers whose cloud-storage transactions are not limited by bandwidth are unlikely to benefit from using file compression; the use of file compression in scenarios where there is abundant bandwidth are much more likely to increase cloud-storage latencies due to the additional CPU time required to compress files. In some cases it may be ideal to use file compression to reduce bandwidth usage — and data transfer costs — at the cost of increased latency; however, reduction in latency is our goal.

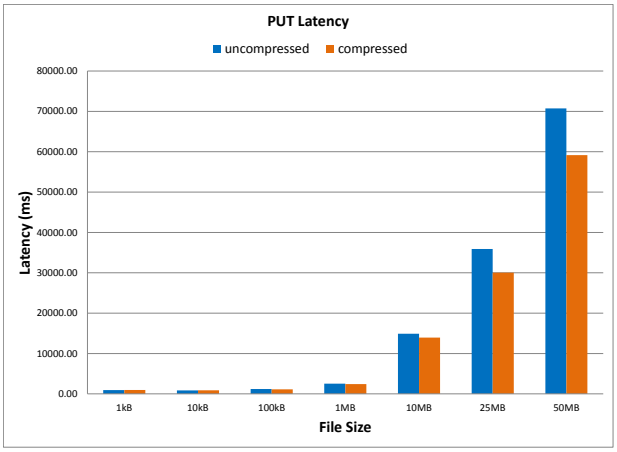

Figure 7: Comparison between latencies for various file sizes when using compression and not using compression.

As seen in Figure 7, the reduction in file transfer times due to file compression more than offsets the increase in time taken to compress or decompress the file.

The effect of file compression is either not observed at all or is minimal for smaller file sizes less than equal to 1MB. But on increasing the file sizes beyond 10 MB, a significant reduction in the latencies is observed. Also, it can be observed that the reduction in latency is more while uploading (PUT requests) the files than it is for downloading (GET requests) the same files.

In cases where the connection between the client and cloud service has limited bandwidth, we have demonstrated that compression is effective at reducing both GET and PUT latencies. In order to verify our assumption that high bandwidth clients would not benefit from compression, we also ran our latency measurement scripts using a high-bandwidth Azure VM and a WABS container; using this setup, we found that compression significantly increased both GET and PUT latencies for each of our test files. In some cases, the use of compression increased the latency measurements by almost 100%.

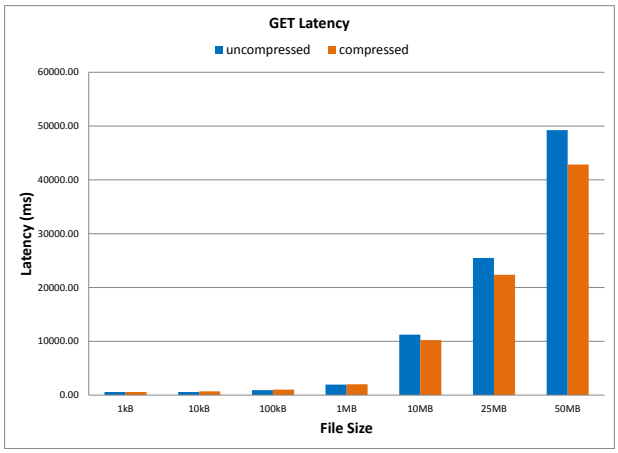

Figure 8: Comparison between latencies for various file sizes when using compression and not using compression.

For the 1kB to 50kB files, compression of our random text test files using `gzip` resulted in reductions in file size between 19% (for the 1kB file) and 24% (for the 50MB file). For PUT requests, the reduction in latency ranged between $-.98\%$ and 16.36%. For GET requests, the reduction in latency ranged between -1.38% and 12.99%. In each of these cases, the worst change

in latency occurs for the 1kB file, and the best for the 50MB file. For this specific set of test files, *the reduction in file size does not correspond to an equally large decrease in latency.*

Because the compression technique involves a balance of decreased bandwidth utilization and increased CPU utilization, customers who might consider using this technique should evaluate whether or not it is cost-effective. An expression to evaluate whether file compression is a useful technique for reducing costs is given by

$$t_c C_{\text{CPU}} < \Delta S(C_b + C_s) \tag{1}$$

where t_c is time spent on compression, C_{CPU} is the cost of CPU time, S is the total reduction in data size due to compression, C_b is the cost of data transfer, and C_s is the cost of data storage. This relationship assumes unlimited bandwidth. In cases where bandwidth is restrictive, the reduction in file transfer times would need to be considered. In cases where the workload is specific and well-understood, it is straightforward to estimate all these variables, making it a trivial task to determine whether or not compression would be beneficial.

5.4 Reducing Variability: File Replication

Figure 9 shows the impact of issuing a different number of GET requests to the same file; the single file is replicated three times on Amazon S3. It can be seen that for the smaller file sizes of 1kB and 10kB, issuing more than one GET requests to the same file has negligible impact on their variability. However, for file sizes greater than 100kB, it is clear from the graph that issuing 3 GET requests is better than 2 or less GET requests. It is likely that variability could be reduced further by increasing the number of file copies; however, that would be a topic for future study.

It can be observed from Figure 9 that replication is effective in reducing variability in GET latency. When evaluating the file replication candidate technique, it is important to note that it

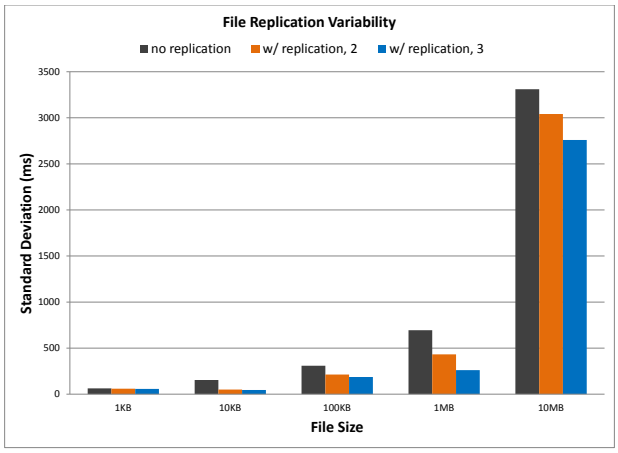

Figure 9: Comparison between variability in latency when using various numbers of file copies. For the tested degrees of replication, standard deviation decreases as the number of file copies increases.

uses three times as much disk space, and requires three times as many GET requests. Given that cloud-storage providers generally charge customers based on storage consumption and the number of GET requests, the needs of the customer and the costs of the cloud-storage service need to be considered. An expression for evaluating whether file replication is a useful technique for reducing costs is given by

$$N_G C_{\mathbb{V}} \Delta\overline{\mathbb{V}} > F\bar{S}(C_s + C_G) \tag{2}$$

where $\bar{S}$ is average file size, F is the replication factor (number of file copies), $\Delta\overline{\mathbb{V}}$ is the average decrease in variability, N_G is the number of GET requests, C_G is the cost of GET requests, C_s is the cost of data storage, and $C_{\mathbb{V}}$ is the cost of variability. Unlike the expression for evaluating the benefit of file compression, not all of these variables have tangible costs associated with them. Because of this, individual customers will need to determine how much they value decreases in variability of their cloud-storage performance.

Although we initially considered file replication solely for reducing latency variability, this technique also shows a clear decrease in 90th percentile latencies as the number of replicas increases. These results are shown in Figure 10. As with the reduction in variability demonstrated by

this technique, latency reduction become more significant as file size increases. While decreases in 90th percentile latencies are almost negligible for the 1kB and 10kB files, 100kB and larger files show more significant decreases.

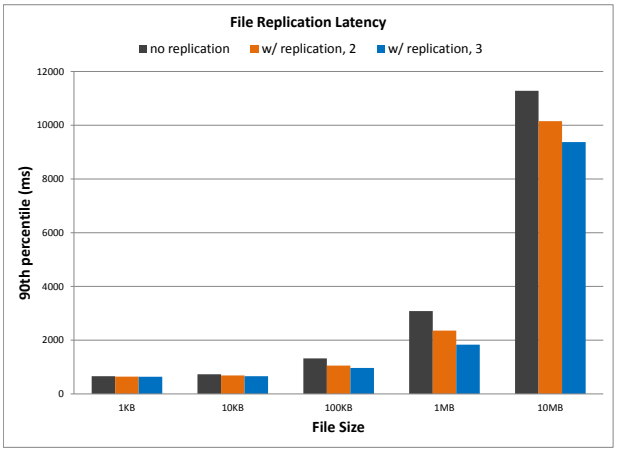

Figure 10: Comparison between the 90th percentile latencies when using various numbers of file copies. For the tested degrees of replication, 90th percentile latency decreases as the number of file copies increase.

6 Conclusions

In this report, we compared the cloud-storage services offered by various commercial and consumer service providers. We ran the latency measurement scripts for each of the cloud services offered by these service providers and noted the variability in latencies over the entire week. We plotted and evaluated the latency measurements for the GET and PUT requests for different file sizes using each of the cloud services by taking the median values of the latencies (to avoid the noise and spikes in latency). We also plotted the graph to show the variability in latency over time for the GET and PUT requests by choosing the most common file size of 1MB. We then evaluated the effect of two candidate techniques to improve latency and variability of cloud-storage requests: file compression, and file replication. We evaluated our results for file compression in both high-bandwidth as well as by limited-bandwidth setups, to simulate a potential scenario for customers who do not have high speed internet connection. The results of this test determined that compression is effective at reducing latency in limited-bandwidth scenarios, but increases latency in high-bandwidth scenarios. In our evaluation of file replication as a technique for reducing latency variability, we compared the variability in latency and the 90th percentile latencies using a different number of file copies. In each case, as the number of file copies increases, both variability and latency are reduced.

The few important observations that were made by analyzing the data gathered from our different experiments are as follows.

- Uploading the 1KB file took more time than 10KB file for all the cloud storage service providers.
- When uploading the files, Dropbox had higher latency than Google Drive for the smaller file sizes while the latter showed the high latencies than the former for the larger file sizes.
- To upload the file size of 1KB, Google Drive showed a significant failure rate of around 20%.
- On an average, downloading the same file takes lesser time than uploading, for each of the service.
- When downloading the files, Google Drive took relatively more time to download the 1KB file than 10KB file.
- Dropbox shows better results than Google Drive for downloading the different file sizes. It was observed to be even better than one of the commercial service providers, Amazon S3, for downloading a larger file size of 10MB.
- Amazon S3 showed consistently lower latencies than Dropbox where bandwidth limitation was not significant

- Of all the four services being compared, the WABS had the lowest median GET and PUT latencies for each of the file types.
- Daily average latencies for uploading 1MB file size is highest for Dropbox and lowest for WABS.
- Compression is ideal for reducing latency for cloud-storage customers who have extra CPU time, but limited network bandwidth.
- With the bandwidth limitation, the file compression showed either a minimal or no effect at all for file sizes smaller than 1MB. For larger file sizes it showed a significant reduction in latencies for both GET and PUT requests, as expected.
- The reduction in latency was observed to be more for the PUT requests than for the GET requests for the same file sizes, in case of file compression.
- In case of file replication, issuing 3 GET requests is optimum than 2 or less for all the file sizes.
- The 90th percentile latency decreases as we increase the level of replication.
- File replication works best to reduce variability in larger file sizes, when issuing a GET request.

The above observations/conclusions will be helpful for an organization to choose the cloud service provider based on their storage needs.

References

[1] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Rohit Rao, *Improving Web availability for clients with MONET*, Proc. 2nd USENIX NSDI (Boston, MA), May 2005.

[2] Jeffrey Dean and Luiz André Barroso, *The tail at scale*, Communications of the ACM **56** (2013), no. 2, 74–80.

[3] Sushant Jain, Michael Demmer, Rabin Patra, and Kevin Fall, *Using redundancy to cope with failures in a delay tolerant network*, SIGCOMM Comput. Commun. Rev. **35** (2005), no. 4, 109–120.

[4] E. Soljanin, *Reducing delay with coding in (mobile) multi-agent information transfer*, Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on, 2010, pp. 1428–1433.

[5] Ashish Vulimiri, Oliver Michel, P. Brighten Godfrey, and Scott Shenker, *More is less: reducing latency via redundancy*, Proceedings of the 11th ACM Workshop on Hot Topics in Networks (New York, NY, USA), HotNets-XI, ACM, 2012, pp. 13–18.