Cloud Performance Benchmark Series

Amazon Elastic Block Store (EBS)
Amazon Simple Storage Service (S3)
Amazon EC2 Instance Local Storage

Shripad J Nadgowda
Radu Sion

ver. 0.8

1. Overview

This document describes a benchmark of Amazon's Elastic Block Store (EBS) and Simple Storage Service (S3) mechanisms. EBS volumes provide traditional network attached disk storage for EC2 instances. The Simple Storage Service (S3) is primarily designed for easy-to-use web-scale computing and provides REST/SOAP access interfaces to manipulate data "buckets" (up to 5GBytes) uniquely identified by user-generated keys. Additionally, each Amazon EC2 compute unit ("EC2 instance") comes with default local disk storage. The benchmarking framework used here was Filebench that allows the simulation of different application workloads on file systems and disks. Experiments were conducted repeatedly at different times of the day to capture the variance in performance throughout.

2. Setup

The *Amazon Elastic Block Store* (EBS) is a service that allows EC2 users to attach block storage volumes to EC2 instances. These volumes are network attached off-instance storage. As a result, traffic to these volumes is metered towards that instance's overall network bandwidth. The experimental setup for the EBS service included: (i) an off-line creation of EBS volumes, (ii) at-runtime attachment of the volumes to EC2 instances hosted within the same availability zones (to minimize inter-zone network-related issues), (iii) formatting of the mounted volume with the ext3 file system, (iv) mounting of the volume as a filesystem within a standard EC2 Redhat OS instance, (v) subsequent benchmarking thereof.

The *Amazon Simple Storage Service* (S3) was primarily designed to provide a simple web interface for storing and retrieving data organized in "buckets" uniquely identified by user-generated keys. Different access mechanisms to S3 exist. To eliminate complexity-induced variance, we decided to deploy a clean and simple command line access mechanism allowing for bucket creation and subsequent access (the GNU s3tools' s3cmd client). Separate scripting was deployed to harness s3cmd and record results.

Finally, every EC2 instance, when created, comes with default *local "disk" storage*. Its capacity varies according to the instance type (i.e., small, medium, large). This storage is immediately available as a pre-mounted ext3 file system on most UNIX variants AMIs. This storage was also benchmarked.

To capture variance in performance throughout, for all three types of storage, experiments were conducted repeatedly at different times of day.

The in-cloud EC2 instances chosen as benchmarking client platforms were of type High-CPU Medium (c1.medium) with 5 ECUs, 2 virtual CPU cores and 1.7GB memory, running the ubuntu-9.04-jaunty-base-20100319 AMI. Similarly, for benchmarking S3, High-CPU Medium (c1.medium) AMIs with 5 ECUs were chosen as client platforms.

2.1. The Filebench suite

FileBench (originally by Sun Microsystems) is a “configurable file level workload synthesis and measurement framework. It facilitates easy reproduction of complex applications which are pre-defined by workload descriptions” [1]. The following four profiles were used for generating standard workloads:

a) **seqread.** A workload profile which benchmarks sequential file reads. To eliminate unwanted interference, caching disabled, file sizes were set to be larger than the available host machine RAM (e.g., 3-5 GB), iosize was set to 1 MB and the workload was executed single threaded at different 10-15 minute time intervals.
b) **seqwrite.** This profile generates a workload of sequential file writes. The size of file writes was set to 1 MB, caching was disabled and the sync flag was enabled. A single thread was deployed to eliminate unwanted scheduling-related interference.
c) **rread.** This profile benchmarks random file reads. It was run on 3-5 GB files, with caching disabled, iosize 2K, single threaded, at multiple times throughout. Each run consisted of a total of 128MB read data.
d) **rwrite.** Similarly, this profile benchmarks random file writes. File sizes were set in the range 3-5 GB, writes were synchronized, and caching was disabled.

The execution of each above profile shows different behavior of the block storage device in terms of Total number of operations, Throughput (operations/second), bandwidth (megabytes/seconds) and latency (milliseconds). Graphs below illustrate these results for EC2 local disk storage and EC2 attached EBS. Since Amazon S3’s object store data model is different, the S3 performance results are shown separately.

2.2. Amazon CloudWatch

We thought it would be interesting to observe our experiments also using Amazon’s internal monitoring service, CloudWatch. Using this service, one can monitor Amazon EC2 instances, EBS volumes, RDS database instances. However, since the monitoring is continuous in nature and only aggregated results are provided, it is difficult to use it to compute discrete results for a given workload simulation. None the less, we have captured different aspects of EC2 instance behavior and EBS volume characteristics over the period of performing the filebench experiments. These results provided additional insights, e.g., monitoring the network traffic for the EC2 client instances illustrated the associated network overhead of storage-related operations generated.

3. Results

3.1. Block Storage

To visualize the impact of time of day on results, each workload profile was run at four different times (12 AM, 6AM, 12 PM, 6PM). The results are illustrated below.

As expected (Figure 1), read bandwidth for the EC2 local block storage varies quite significantly throughout. Midnight seems to yield almost 80% more sequential read bandwidth than 6 PM and 43% more than during mid-day values. This is likely the result of multi-tenancy. Interestingly however, EBS read volume bandwidth was more or less constant throughout the day. This suggests a network-related cap.

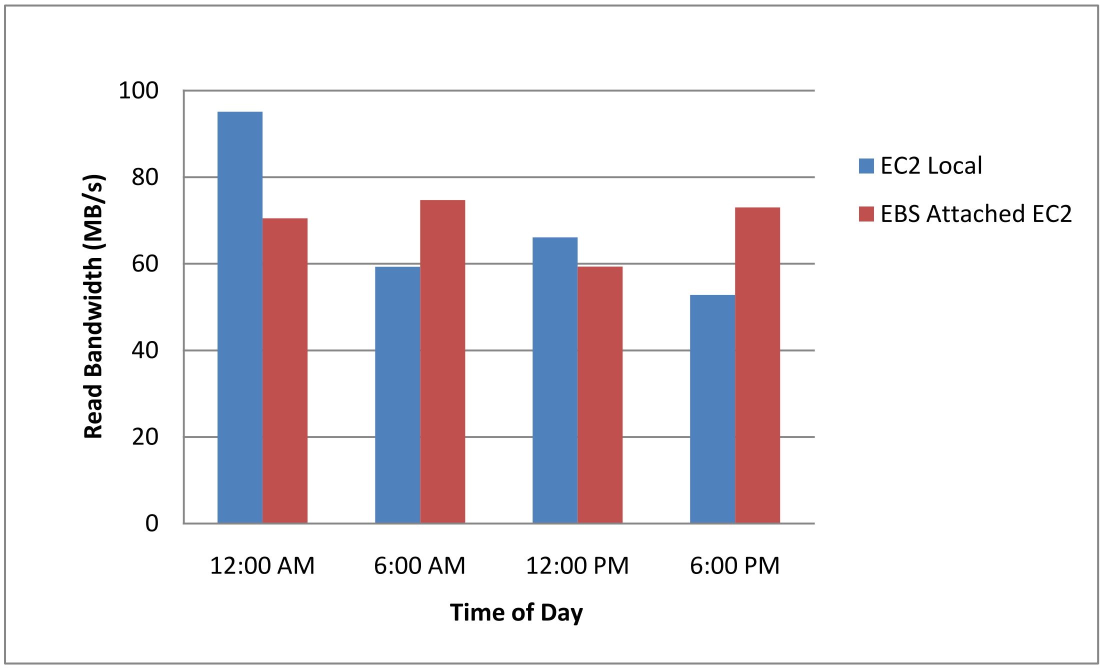


Figure 1: Sequential Read on Block Storage

Similarly the perceived bandwidth for sequential writes on EC2 local storage at midnight is almost double than in the evening. Random reads and writes yield almost the same bandwidth throughout the day. This also suggests a network-related cap.

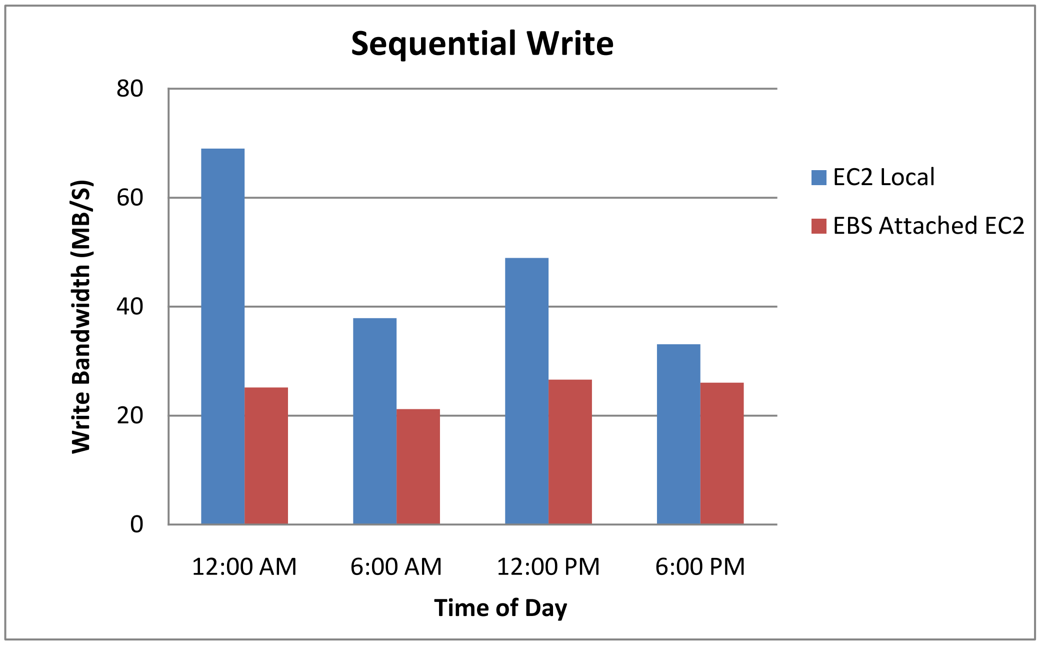


Figure 2: Sequential Write on Block Storage

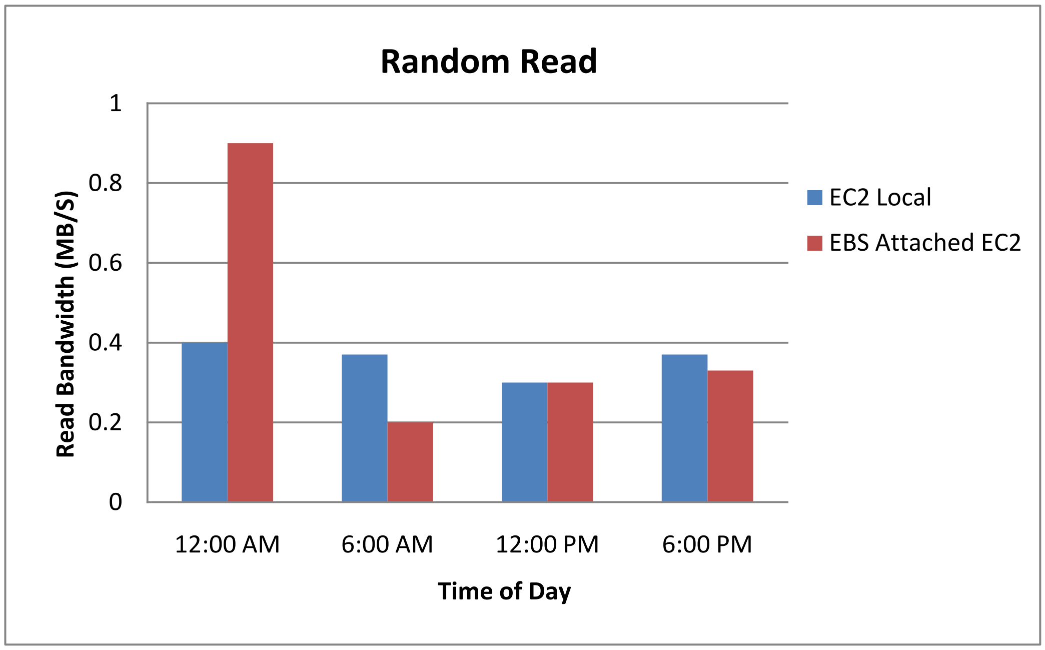


Figure 3: Random Read on Block Storage

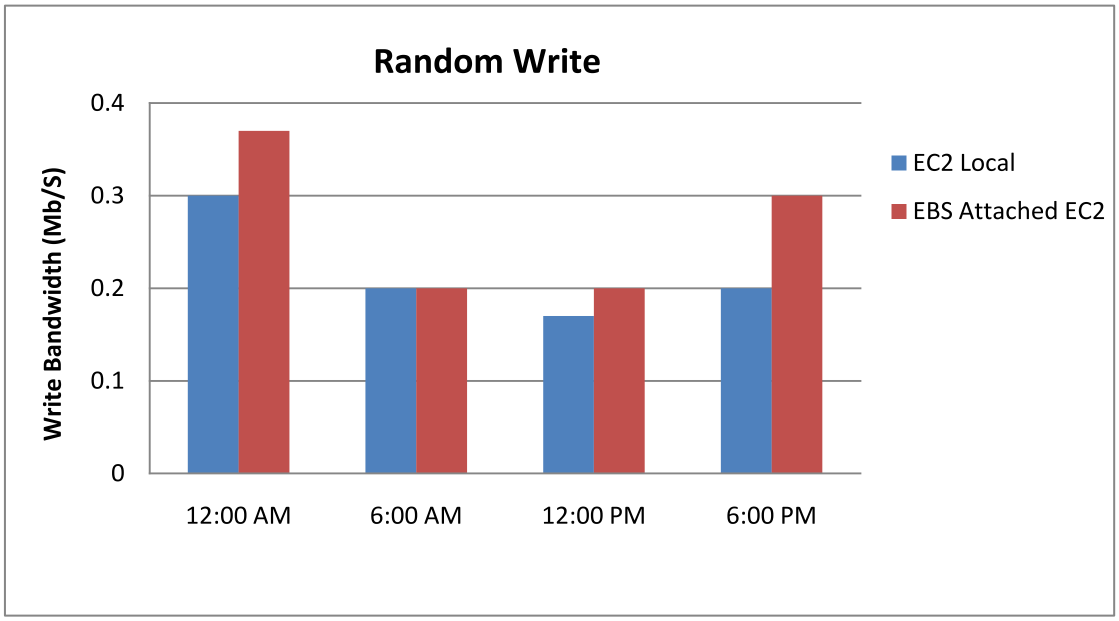


Figure 4: Random Write on Block Storage

Monitoring EC2 instances and EBS volumes with Amazon CloudWatch served a validation purpose. Figures 5-8 show monitoring results for one set of experiments carried out around noon-time. In all figures, the x-axis represents time.

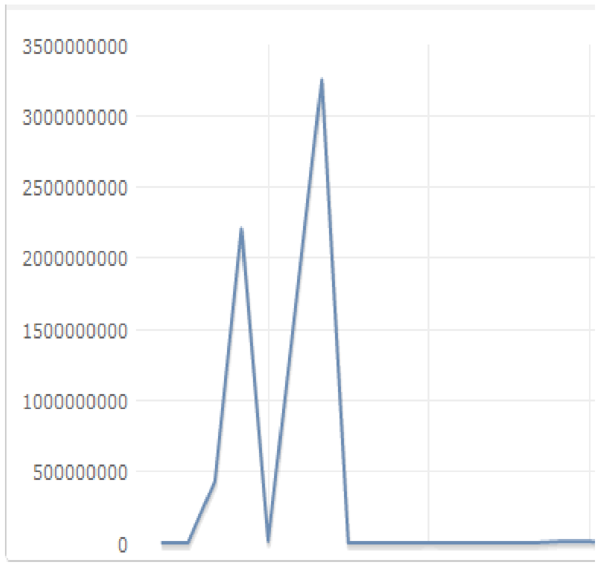


Figure 5: Disk Reads (bytes)

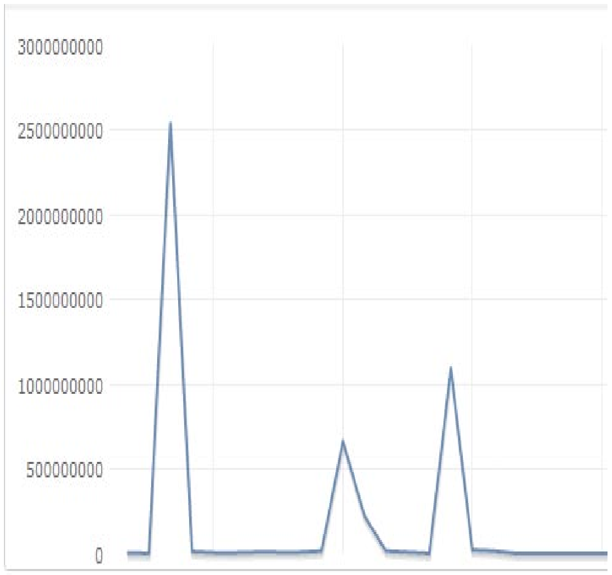


Figure 6: Disk Writes (bytes)

Figure 5 shows the EC2 disk storage monitoring results during sequential and random read profile runs. It displays the average of number of bytes read. The peaks correspond to data read during sequential and random read runs respectively. Note that these numbers represent aggregates. It can be seen that, for same filesize, the random read profile generated much more traffic than the sequential reads. Figure 6 shows uniform behavior during the sequential data writes (first peak in the graph), followed by the variable data write operations during the random writes.

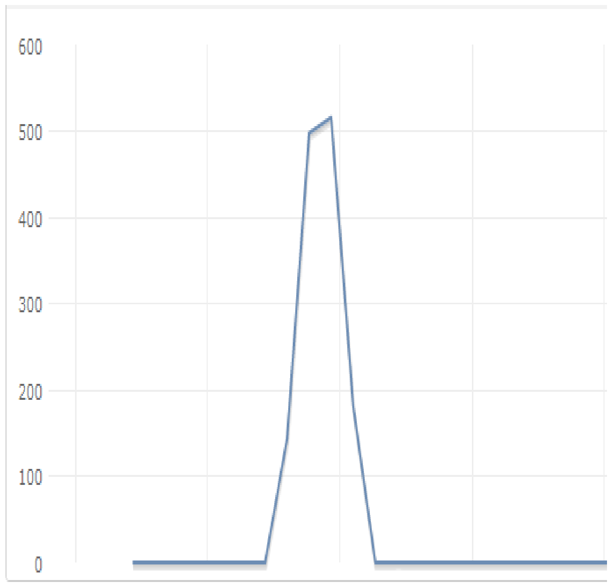


Figure 7: EBS Read throughput (ops/s)

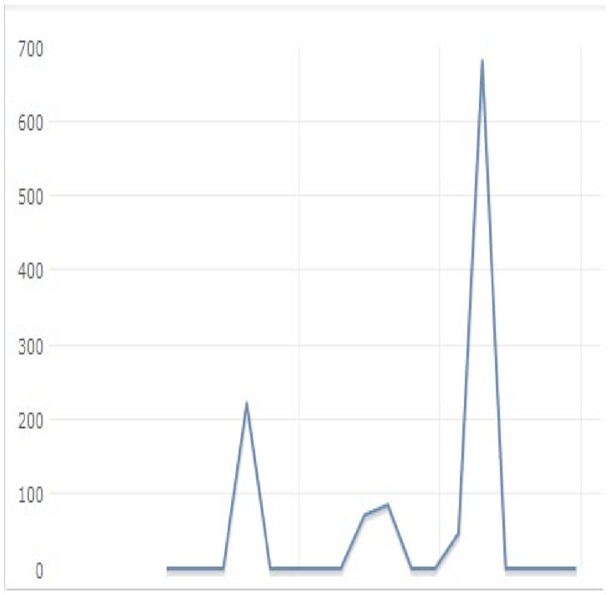


Figure 8: EBS Write throughput (ops/s)

Figures 7 and 8 show aggregated EBS throughputs (ops/sec) during the (sequential and random) read and write phases. Figure 8 suggests that write throughput for random writes is higher when compared to sequential writes. On the other hand, a write throughput peak was observed for sequential writes on EBS.

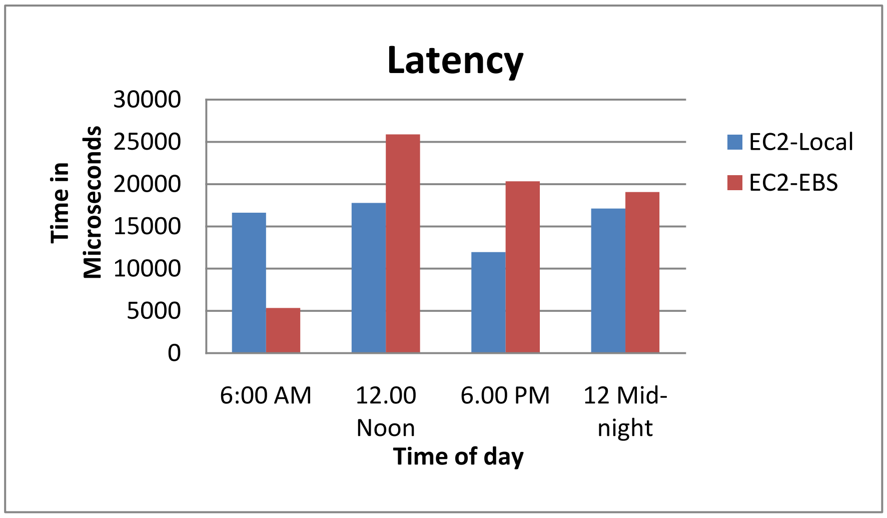


Figure 9: Write Latency for Amazon Block storage

The latency for the Amazon Block Storage (Local and EBS-attached) was computed by issuing synchronous write requests of 4096 bytes of data from within a custom-written application. Synchronous I/O is achieved by deploying the O_SYNC flag in each write call. The flag aims to minimize the effect of caching. Read latencies cannot be benchmarked properly as it is not possible to bypass the inherent caching infrastructure. Results are illustrated in Figure 9.

As can be seen**,** although the write latency for EC2-Local storage is almost constant throghout the day, there is a noticable variation in the latency for EC2-attached EBS storage. This is likely due to the variation in network bandwidth as EBS is attached over the network to the EC2 instance.

3.2. Amazon S3

The *Amazon Simple Storage Service* (S3) was primarily designed to provide a simple web interface for storing and retrieving data organized in "buckets" uniquely identified by user-generated keys. Benchmarks were performed from two different vantage points: Stony Brook and an instance within the same EC2 region. Results are shown in Figures 9 and 10. An average throughput of about 15.5 MB/sec was observed from within EC2. On the other hand, the Stony Brook vantage point yielded only 1.12 MB/sec. – which suggests a network-related cap.

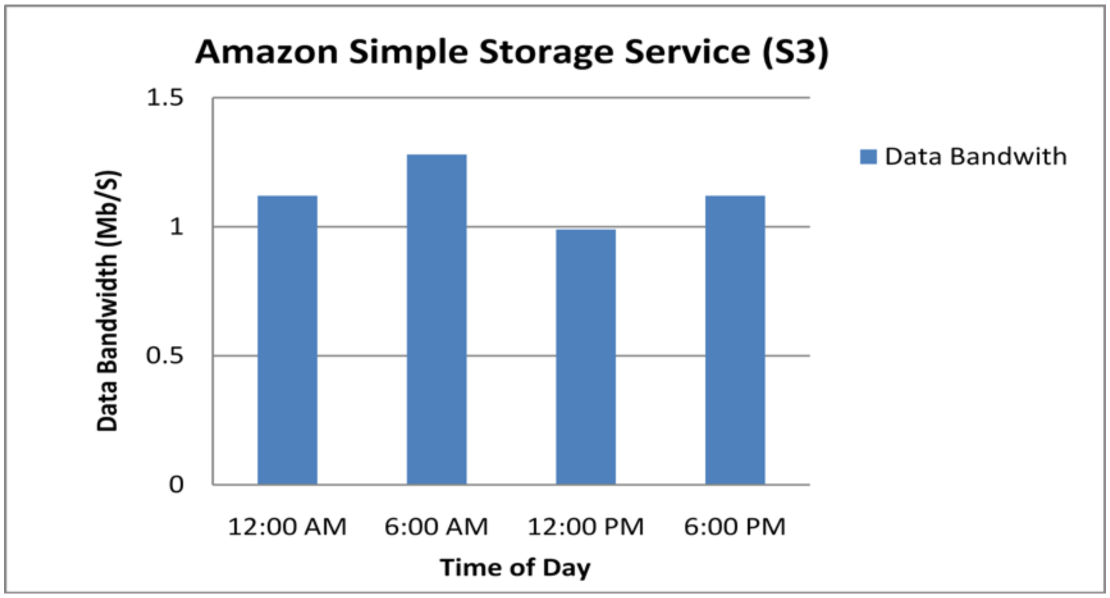


Figure 10: Data bandwidth on Amazon S3 (Stony Brook vantage point)

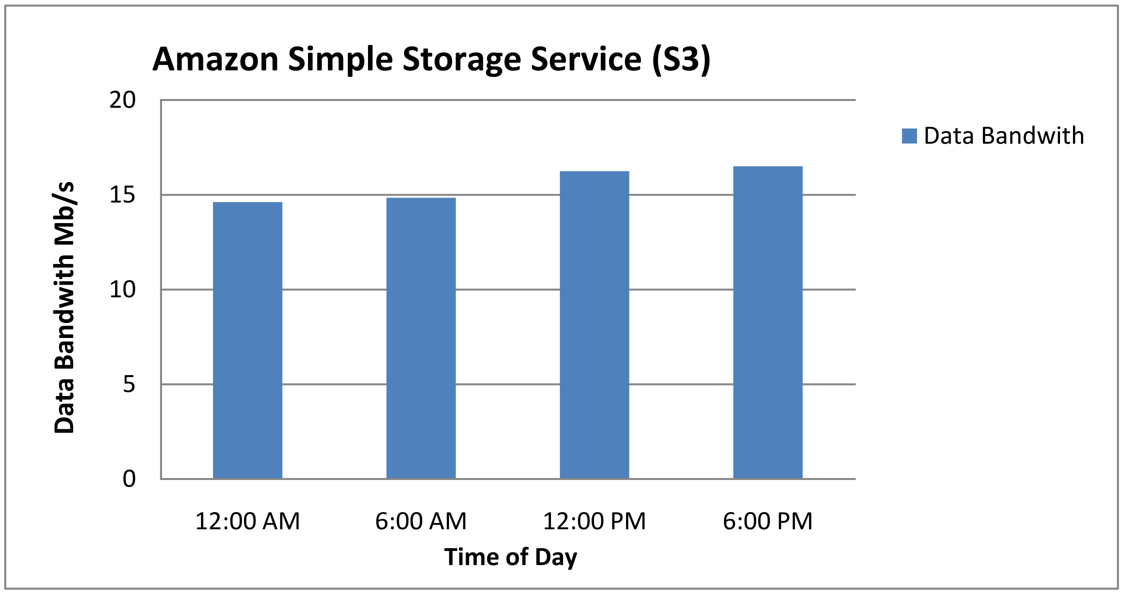


Figure 11: Data bandwidth on Amazon S3 (EC2 vantage point)

The latency of PUT and GET requests of object-storage on Amazon S3 was also computed for 4096 bytes object data. Results are depicted in Figure 12. As can be seen again the variation in the observed latency for GET and PUT requests on Amazon S3 is due to variation in network latencies, especially when considering the numbers depicting the local machine - S3 connection.

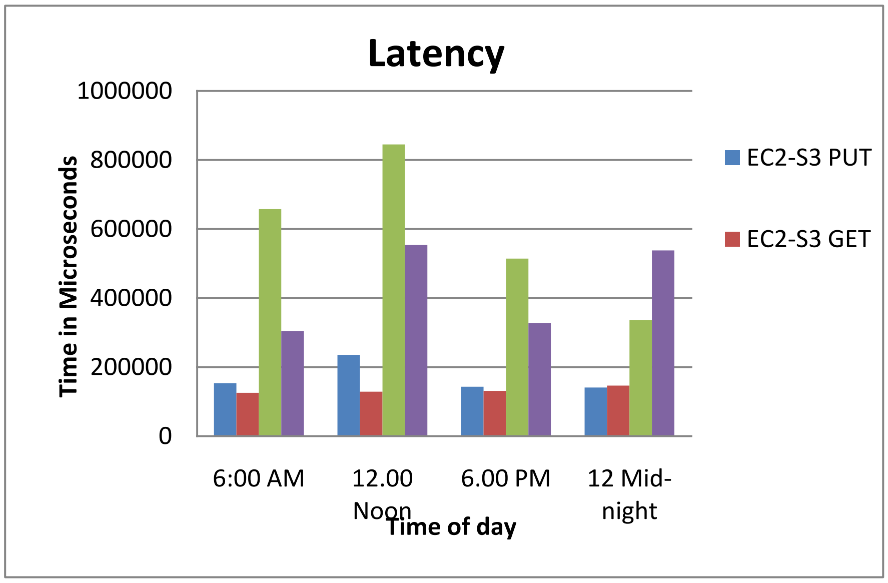
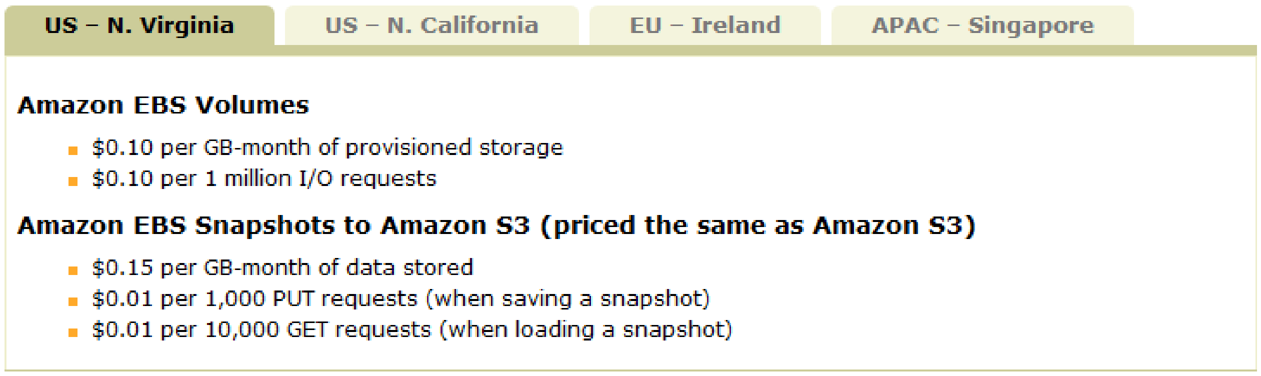


Figure 12: Write/Read Latency for Amazon S3

3.3. Cost


US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

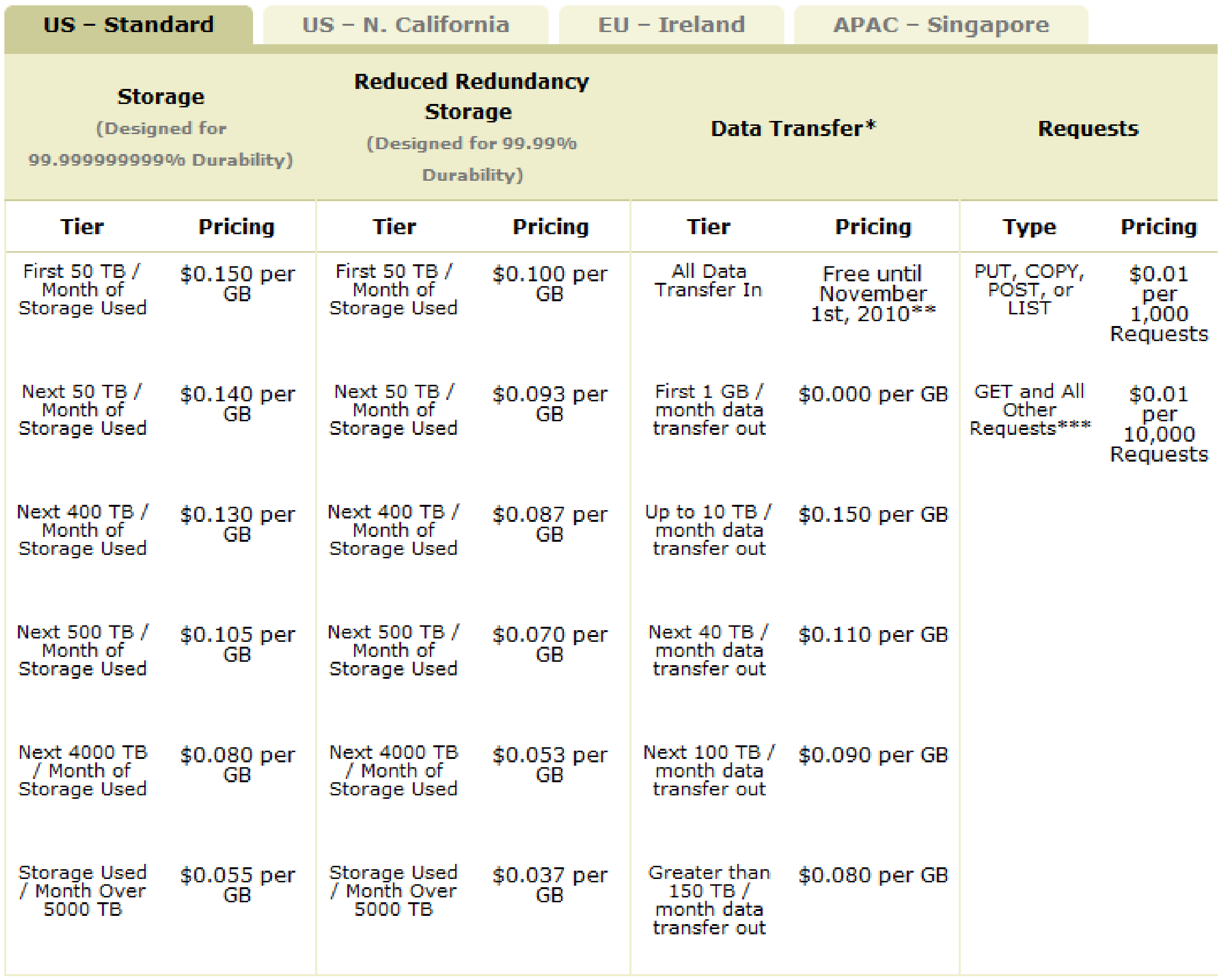
Amazon EBS Volumes

- $0.10 per GB-month of provisioned storage
- $0.10 per 1 million I/O requests

Amazon EBS Snapshots to Amazon S3 (priced the same as Amazon S3)

- $0.15 per GB-month of data stored
- $0.01 per 1,000 PUT requests (when saving a snapshot)
- $0.01 per 10,000 GET requests (when loading a snapshot)

Figure 12: Monthly price of EBS Volume (amazon.com)


US – Standard | US – N. California | EU – Ireland | APAC – Singapore

Storage (Designed for 99.999999999% Durability)		Reduced Redundancy Storage (Designed for 99.99% Durability)		Data Transfer*		Requests	
Tier	**Pricing**	**Tier**	**Pricing**	**Tier**	**Pricing**	**Type**	**Pricing**
First 50 TB / Month of Storage Used	$0.150 per GB	First 50 TB / Month of Storage Used	$0.100 per GB	All Data Transfer In	Free until November 1st, 2010**	PUT, COPY, POST, or LIST	$0.01 per 1,000 Requests
Next 50 TB / Month of Storage Used	$0.140 per GB	Next 50 TB / Month of Storage Used	$0.093 per GB	First 1 GB / month data transfer out	$0.000 per GB	GET and All Other Requests***	$0.01 per 10,000 Requests
Next 400 TB / Month of Storage Used	$0.130 per GB	Next 400 TB / Month of Storage Used	$0.087 per GB	Up to 10 TB / month data transfer out	$0.150 per GB		
Next 500 TB / Month of Storage Used	$0.105 per GB	Next 500 TB / Month of Storage Used	$0.070 per GB	Next 40 TB / month data transfer out	$0.110 per GB		
Next 4000 TB / Month of Storage Used	$0.080 per GB	Next 4000 TB / Month of Storage Used	$0.053 per GB	Next 100 TB / month data transfer out	$0.090 per GB		
Storage Used / Month Over 5000 TB	$0.055 per GB	Storage Used / Month Over 5000 TB	$0.037 per GB	Greater than 150 TB / month data transfer out	$0.080 per GB		

Figure 13: Amazon S3 Store / GB (amazon.com)

Computing the pricing model for cloud storage offerings in general is necessarily an application specific endeavor and subject to various factors including workload, sizes, usage, number of transactions, add-on services etc.

Nevertheless, we thought it would be interesting to compute the cost of storing bits with Amazon. We considered a file system of 500GBs and an average workload of 150 I/O operations per second. Given the current pricing scheme, we have the following yearly numbers for EBS:
Cost of I/O requests: ~2.6 million seconds/month * 12 * 150 I/O per second *$ 0.10 per million I/O = $468. Cost of EBS volume: 500 GB * $0.10/month *12 = $600. The total storage cost: **$1068/year**. Per-bit this comes down to 248 US picocents (1 picocent = 10^{-14} USD). Cost of only storing the data without I/O: 140 picocents. This is surprisingly close to our predicted numbers in [2] where we had shown that the yearly cost of powering up and storing a bit would hover around 100 picocents.
Similarly, for the Amazon S3 object store, the cost of storing a bit for one year is shown below. Naturally, with increasing storage capacities better deals can be had. We suspect that the 50 picocents cost for capacities over 5000TBytes is possible only while taking into account an expected relatively low utilization factor (e.g., < 50%).

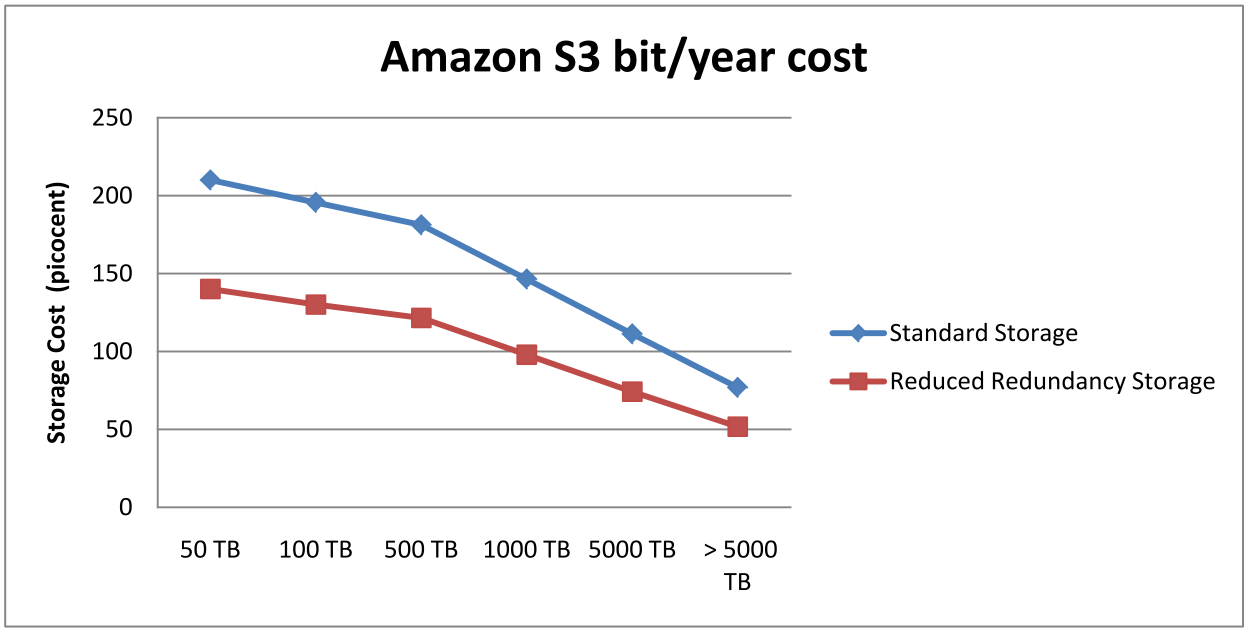


Figure 14: Per-bit yearly costs for Amazon S3 storage

[1] Filebench, http://www.solarisinternals.com/wiki/index.php/FileBench
[2] Yao Chen, Radu Sion, "On Securing Untrusted Clouds with Cryptography", in the ACM Workshop on Privacy in the Electronic Society WPES 2010, at CCS