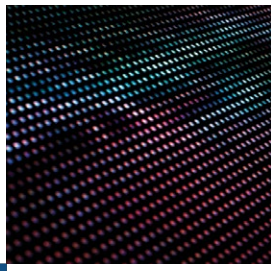
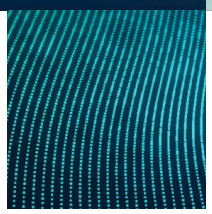
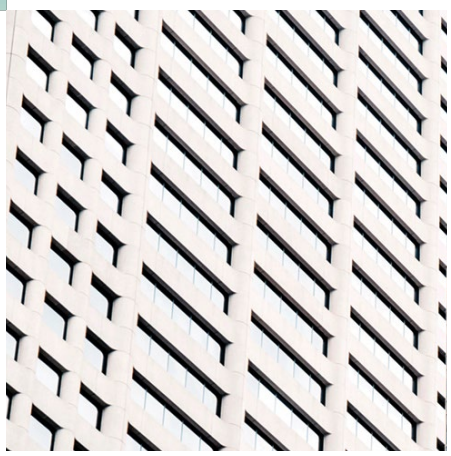






WHITE PAPER

S3 and Analytics
Taming Your Storage Costs

NetApp®

S3 and Analytics – Taming Your Storage Costs

In the cloud, object storage is ubiquitous when it comes to analytics. Whether the use case is map reduce, Hive, Spark, genomics or even artificial intelligence engines, you are sure to find and object storage entity storing its data. The hyperscalers are consumers of their own technology, consider that AWS uses S3 as the storage repository for Elastic Map Reduce — thus they eat their own dog food. After all, object storage is durable, highly available, and the API is a standard. So why even call out potential alternative options to such a popular architecture? As cloud continues to innovate with new tools for analytics, feeding those tools predictably and efficiently can have substantial impact on the ROI and TCO of your project. In other words, think cost and control.

In this white paper we explore analytics from the perspective of S3 and the Cloud Volumes Service alternative. Much of the investigation will focus on the three-fold cost of using object storage, with attention paid to the S3 API. We will walk through a test workload and explore the costs thereof using AWS CloudTrail and AWS Athena and then follow up with a discussion on cost cutting and cost forecasting measures. We will close this architectural review with an investigation into the merits of the Cloud Volumes Service as the solution for in place analytics.

Though this architectural review references S3 specifically, what's true for AWS is equally true for the other big name hyperscaler technologies; the lessons learned are universal.

Three Independent Costs

Three independent costs are associated with S3:

- The cost of capacity
- The cost of egress
- Cost of use

Regarding capacity — The cost to store data in S3 is low at about two cents per gigabyte per month. As for egress [reading from the S3 bucket], if you are using the Standard Access tier and the application is in the same region as the bucket then the egress cost is waved — ingress is always free. More on this later. Of the three costs, use is the most interesting to understand and difficult to tame. So, let's explore it.

Exploring the Cost for Use

All interactions with S3 are done via API request. If you are unfamiliar with what API requests are used for, consider the example wherein a file is being retrieved — think read by a client. GetObject API calls are used to generate the retrieval request.

AWS charges per 1,000 API requests to S3, as with all things Amazon the cost varies by region. See below for the Standard Access cost of use in the Northern Virginia region then follow on to see the prices for accessing from the Standard-Infrequent Access tier in the same region.

STANDARD ACCESS TIER

Data Returned by S3 Select	$0.0007 per GB
Data Scanned by S3 Select	$0.002 per GB
PUT, COPY, POST, or LIST Requests	$0.005 per 1,000 requests
GET, SELECT and all other Requests	$0.0004 per 1,000 requests
Lifecycle Transition Requests into Standard – Infrequent access or One Zone – Infrequent access	$0. 01 per 1,000 requests

STANDARD-INFREQUENT ACCESS TIER

Data Retrievals	$0.01 per GB
Data Returned by S3 Select	$0.01 per GB
Data Scanned by S3 Select	$0.002 per GB
PUT, COPY, or POST Requests	$0.01 per 1,000 requests
GET, SELECT and all other Requests	$0.001 per 1,000 requests

Even a quick comparison of the costs associated with the two tiers is telling. First you should notice that the costs for using the Standard-Infrequent Access tier are orders of magnitude higher than the cost for using the Standard Access tier. With Standard Access tier you pay one cent per 25,000 GetObject calls while the GetObjects issued against the Standard-Infrequent Access (SIA) tier come at a price tag of one cent per 1,000. Thus, we see that the Standard Access tier is 25X less expensive for use than the SIA tier. Second you should notice that the Standard-Infrequent Access tier builds in a one cent cost per GB of data retrieved.

Many see the value of tiering less frequently used content as an effective cost cutting measure — at roughly a penny per GB, moving data to the Infrequent Access layer cuts storage capacity costs in half. We refer to this as "control" in the introduction to this architectural review. Never move working set (a.k.a. hot data) to the Standard-Infrequent Access tier, the 2X cost savings pales in comparison to the 25X cost increase that comes with accessing the data from this tier. This eliminates the best cost cutting/cost control mechanism available to S3.

So far, we have spoken to theory only, so let's look at pricing in practice to see if our theory regarding cost is correct. Towards that end we called upon our good friend and Hadoop expert Karthik Nagalingam. We set up a Hadoop Spark cluster on EC2 and ran a the HiBench workload to see just how work is done, to see just how many API calls are issued.

The Lab Setup

For these tests, we created an environment made up of a fifteen node [c5.9xlarge] Hortonworks Hadoop Data Platform 3.0.0.0 cluster running Spark 2.2. To test the capabilities, we ran the HiBench Wordcount workload generator using an 74TB bigdata dataset. The dataset was made up of 420 maps contained within a single S3 bucket. Each file was roughly 165GB in size and made up of random ASCII formatted words.

Sample Map File Content

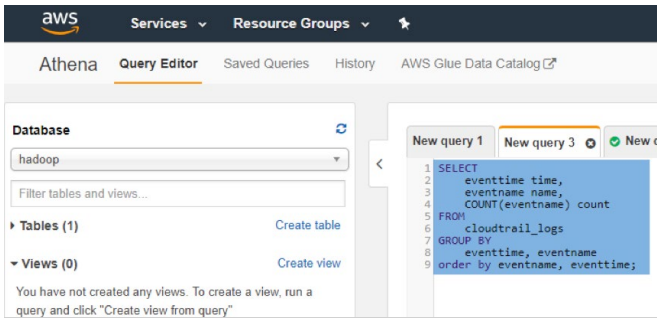
The files were accessed via 420 Mappers and Reducers or in other words — 420 containers spread across the fifteen EC2 instances. Each container had access to 1 VCPU and anywhere from 2GB to 64GB of RAM. The HiBench Wordcount workload generators' goal was to parse the entire data set returning the number of unique words per file, as shown below.

In order to track S3 API calls, we configured the S3 bucket to support object-level logging. With object-level logging in place, AWS CloudTrail records in JSON files all API data events that relate to objects in the bucket. These JSON files can be queried by using the AWS Athena Hive analytics engine to answer the question about the quantity of API calls, thus allowing us to calculate the use cost for the workload. By following these steps, you can determine the cost of running analytics in S3 for your environment.

Athena for CloudTrail Analytics

If you're new to Athena, the following steps will get you started.

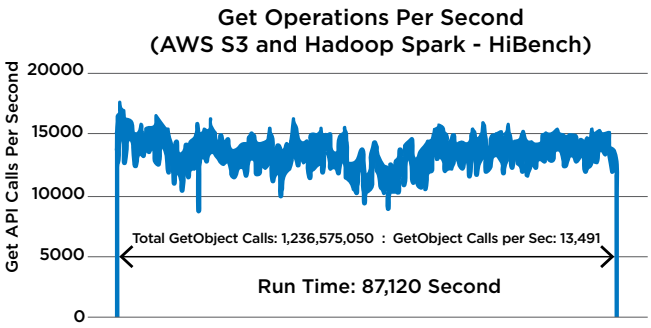
1. From the Athena service, create a new database.
2. Follow the instructions to create a table in the Athena User Guide in the section "Manually Creating the Table for CloudTrail Logs in Athena."
3. Use the following query to determine the number distinct API calls per second in your S3 bucket.



Calculating the S3 Cost

Having described the HiBench workload, the cluster configuration, and how to run queries within Athena, let's look at the test results and use them to study the possible API costs. Athena tells us that for the HiBench Wordcount workload, 1,236,575,050 GetObjects calls were required to complete the job. No other API request type was seen during test any quantity worth discussing so we will ignore them. Those 1.2 billionish calls occurred and average rate of 13,491 GetObject calls per second as the graph below shows — this is important for the overall analyses as it will help see costs at the lowest possible granularity of time. The graph below shows these details, this graph is the visual result of the data taken from Athena.

S3 Graph of GetObjects



To get the API cost for the job, we used the following formula:

$$\text{API cost} = \$\,\frac{0.0004 * \text{TotalGetObject calls}}{1{,}000}$$

To find out the cost per second, the formula becomes:

$$\text{API cost} = \frac{\left(\$\,\frac{0.0004 * \text{TotalGetObject calls}}{1{,}000}\right)}{\text{TotalRunTime in seconds}}$$

We have created the table below to show how the costs scale across time using the calculated per second cost of half a penny. We've included per minute, job, hour, day, month and yearly cost. That $0.0004 cost per 1,000 GetObject calls add up quickly as you can, and as you may have already experienced in your own environment. If we were using S3 as the storage backend for a primary analytics cluster and as such work like this were always running, the API costs alone add up to about $170,000 per year. That is irrespective of EC2 instance count or use of Elastic Map Reduce, storage capacity used, license costs and personnel costs.

S3 GetObject API Cost by Time

S3	
A	Avg Gets/s
	13,491
Job duration (Minutes)	1,452
Job cost	**$470.14**
Second cost	$0.005
Minute cost	$0.32
Hour cost	$19.43
Day cost	$466.25
Month cost	$14,181.85
Year cost	$170,182.16

Options to Control Costs

The general suggestion to tier cold data to lower tiers of storage will reduce overall cost, but it has no positive effect upon working set. As we've shown in a previous section, there is potential for massive — as great as 25X — spending increase if working set is tiered to even the Standard Infrequent-Access storage tier.

Speeding up job runs by optimizing S3 data layout, for example by spreading your files across multiple prefixes may potentially increase your costs. Slowing down the run impacts business negatively without positively impacting job cost whatsoever. The former is true if by increasing the speed of a run you end up adding more runs thus generating more API calls. The latter is true as the number of API calls a job requires is dataset dependent and not time dependent. The HiBench Wordcount workload shown above generates ~1.2 billion GetObject calls no more how long the job takes.

To accurately forecast the cost of object storage for analytics workloads requires an intimate knowledge of not just your data including working set size, but access patterns and number of GetObject requests as well. Do you have this level of knowledge — have you tamed your data? Do you even want to do amount of work required to get this level of knowledge?

What about a caching layer? Some of our customers tell us that the best way to control the cost of object storage is to divide and conquer. They keep their working set on primary storage while retaining the overall data set in object storage. This technique not only decreases the cost of operations and enables forecasting but provides a level of control as well — you get to determine how fast the jobs can run and can lean towards best outcome for business or best outcome for financials (cost). A caveat for this scenario is that it's hard to do — i.e. you the customer need to know what is working set and what is not to accurately place the data.

Then there is the Goldilocks zone — a simple scenario wherein it all just works; in place analytics sits squarely in this space. In place analytics allows you to derive value from your data where it sits. Perhaps the data set size is not enormous, fitting nicely within 100TB filesystems — the size today of a NetApp Cloud Volumes Service volume. Take for example the scenario of application logging for business-critical infrastructure. Applications logs may contain information valuable to the overall business — but this value needs to be derived. Avoid the complexity of a caching system and analyze your content where it sits.

The Goldilocks zone is the focus of the rest of this paper.

Cloud Volumes Service for In-Place Analytics Needs

When performing analytics work, it's generally best to select the option that offers the most overall bandwidth, even though the higher-bandwidth options come at a higher advertised storage cost. Higher bandwidth enables higher throughput, which results in faster completion times. Faster completion equates to improved resource utilization and greater cost efficiency. In other words, the higher-priced option results in lower overall cost.

Though there are technically three service levels (Standard, Premium, Extreme) the amount of bandwidth accessible to the volume converges at higher capacity points — beyond 56TB. Therefore, choose the Premium rather than Extreme service level to satisfy your bandwidth needs.

For more information on service levels, check out the service level reference document. Please note, as the capacity tested in the lab exceeds the 56TB quoted above, the remainder of this architectural review references the Standard and Premium service levels only.

STANDARD	PREMIUM	EXTREME
$0.10 GB per month	$0.20 GB per month	$0.30 GB per month

As a primer on service levels, the following information is good to know.

- Volumes provisioned through the NetApp Cloud Volumes Service have a maximum size of 100TB; allocate additional volumes to satisfy additional space demands
- Standard service level: This service level is the best fit when lots of capacity is needed and lower amounts of bandwidth are acceptable. The Standard service level comes with sixteen MB of bandwidth per allotted TB of capacity up to a maximum of 1536MB at 100TB.
- Premium service level: In general, this service level is the best fit when a balance of capacity and bandwidth is needed. At 56TB of allotted capacity, the premium service level becomes the most cost effective for bandwidth hungry workloads. Our tests have revealed that given a single cloud volume is capable of roughly 3,500MB/s of throughput — your numbers of course will vary.

Within the next few weeks NetApp will add metered pricing with adjustable performance levels. Park your data at the Standard service level when not in use, and shift into gear during the times of actual analysis as peak performance is your goal.

How far can your analytics drive a volume? We ran the same tests described in the lab section above and achieved an average throughput of 3,100MB/s against a single Cloud Volumes Service volume. However, you are entirely free to throttle back the bandwidth to get a lower cost.

As graphs are only meaningful when comparing multiple points, the following is a comparison of each of the tests we ran in the lab. Only the storage backends were changed between tests. Each test was an evaluation of single entity, i.e. a single volume or bucket spread across 15 EC2 instances.

The HiBench workload generator had responsibility for data layout, no attempts were made to tune for any specific platform.

S3	
A	Avg Gets/s
	13,491
Job duration (Minutes)	1,452
Job cost	$570.73
Second cost	$0.006
Minute cost	$0.36
Hour cost	$21.51
Day cost	$516.14
Month cost	$15,699.12
Year cost	$188,389.40

CLOUD VOLUMES SERVICES	
PREMIUM SERVICE LEVEL	
B	CVS/s
	3,500
Job duration (Minutes)	414
Job cost	$143.41
Second cost	$0.006
Minute cost	$0.35
Hour cost	$20.78
Day cost	$498.83
Month cost	$15,172.60
Year cost	$182,071.20

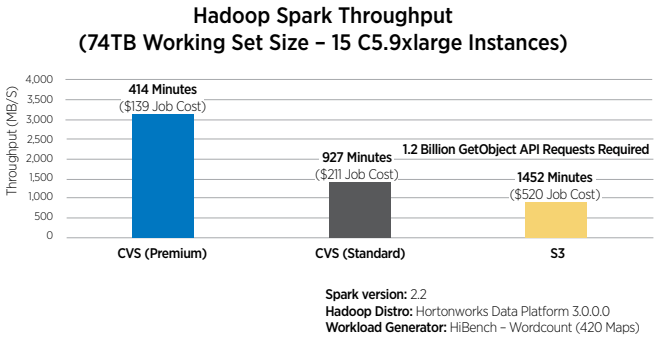
STANDARD SERVICE LEVEL	
C	CVS/s
	1,562
Job duration (Minutes)	927
Job cost	$211.64
Second cost	$0.004
Minute cost	$0.23
Hour cost	$13.70
Day cost	$328.77
Month cost	$10,000.00
Year cost	$120,000.00

Cloud Volumes Service is fast and efficient: Get jobs done up to 3.5 time faster with as little as ¼ the cost with Cloud Volumes Service. Service Levels are flexible: Reduce costs between analytics jobs by toggling the Service Level when not in use.

Notice that the Premium service level has the fastest run time as well as the lowest cost per job as we called out in the service level descriptions above. Though the Standard service level has a higher per job cost than the Premium service level, the overall price of Standard is lower. If you notice that the job cost shown below for S3 is higher than that which was quoted earlier in this architectural review, know that it is not an error. The 74TB data set is priced at just about $50 per day at $0.02 per GB. The job price in the graph below includes the cost of capacity, above only the API costs were included.

The following graph shows that the Premium service level results in the fastest run time of the Spark workload. Although the advertised price of the Premium service level is higher than both the Standard service level and the upfront costs of S3 (capacity + egress), the value add of increased bandwidth resulted in both an overall cost reduction and an improved run time over both.

Working Set Throughput Comparisons



Spark version: 2.2
Hadoop Distro: Hortonworks Data Platform 3.0.0.0
Workload Generator: HiBench – Wordcount (420 Maps)

Cloud Volumes Service Costs

The boxes B|C below demonstrate the two separate pricing scenarios and the performance you can expect with each. As above, we have included the S3 box as well for completeness. You may notice that the annual price for the Premium service level is based on the 74TB data set while the Standard service level pricing is based on 100TB of allocated capacity. This is intentional. Allocating the Standard service level 100TB of capacity gives the workload the ability to go as fast as possible at this tier. On the other hand, the Premium service level maxes out on bandwidth at 56TB of allocated capacity and as such allocating capacity based on data set used here provides full access to bandwidth.

Some additional things you'll notice above:

- Both sets of service levels enable getting the job done quickly and at low cost.
- The Premium service level (Box B) has the lowest overall job cost (75% lower than object storage).
- The Standard service level (Box C) has the overall lowest hourly rate (37% lower than object storage.
- Object storage (Box A) has the highest job cost, of which only $50.00 is for capacity. API calls account for the remainder.

EC2 Instance Cost

Decreased compute instance costs are a value add to getting your work done more quickly (and less expensively, and more predictably). As CentOS qualifies for Amazon's per second EC2 billing, and as the C5.9xlarge instance costs $1.53 per hour, the math looks like this:

$$\text{compute cost} = \frac{\text{instance cost per hour} * \text{instance count}}{60 \text{ minutes}} * \text{duration in minutes}$$

Storage Backend	Job Duration	Compute Cost	Storage Cost	Total Cost
Object Storage Backend	1,452	$555	$520	$1,076
Cloud Volumens Service (Premium Service Level)	414	$158	$143	$302
Cloud Volumens Service (Standard Service Level)	927	$355	$212	$566

The result is $397 compute savings and an overall job cost reduction (compute + storage) of $774.

For the highest overall savings, use the Standard service level when performance is not a concern and shift to the most cost-effective service to get your work done. If your dataset is less than 28TB, the Extreme service level is the most cost effective; beyond 56TB, Premium is the clear winner. Between 28TB and 56TB, calculating the cost gets more complex. But that's a blog for another day — for now, the math indicates that the sweet spot for shifting from Extreme to Premium occurs at 47TB.

Summary and Call To Action

The total cost of ownership involved in running analytics includes much more than calculating capacity costs. Available bandwidth effects run time which lengthens not just the amount of time to complete an analytics job but the cost of the job overall — more compute uptime means higher computation costs which results in higher bills — faster storage means lower spend on compute. Object storage capacity is inexpensive and while egress is often free, API costs are potentially significant and hard forecast and difficult to tame. So what options do you have?

Running your analytics jobs from the Cloud Volumes Service tames the complexity involved with both forecasting and controlling the costs associated with object storage backend. With the Cloud Volumes Service, you are in control of what you spend and how. Spare yourself the overhead of API costs, control your storage spend through service levels, decrease your compute costs by maximizing the compute resources – run the them hot then turn them off. Analytics is complex, simplify where you can, use Cloud Volume Services and run your analytics in place.

What's coming regarding analytics and Cloud Volumes Service? Be on the lookout for HDFS support for Cloud Volumes Service via HDFS connector — soon!! EMR Cloud Volumes Service connectivity is upcoming as well.

Come check us out at cloud.netapp.com

Read more from our author Chad Morgenstern @NetApp

Configuration Files

#/usr/src/HiBench-master/conf/hadoop.conf

hibench.hadoop.home	/usr/hdp/3.0.0.0-1634/hadoop
hibench.hadoop.executable	${hibench.hadoop.home}/bin/hadoop
hibench.hadoop.configure.dir	${hibench.hadoop.home}/etc/hadoop
hibench.hdfs.master	s3a://spark.cloudvolume.netapp.com/
#hibench.hdfs.master	nfs://<ip>:2049/hadoop-cluster-volume-one
hibench.hadoop.release	hdp

#/usr/src/HiBench-master/conf/spark.conf

hibench.spark.home	/usr/hdp/3.0.0.0-1634/spark2
hibench.spark.master	spark://10.11.0.198:7077
hibench.yarn.executor.num	2
hibench.yarn.executor.cores	4
spark.executor.memory	16g
spark.driver.memory	16g
spark.default.parallelism	${hibench.default.map.parallelism}
spark.sql.shuffle.partitions	${hibench.default.shuffle.parallelism}

#/etc/hadoop/3.0.0.0-1634/0/core-site.xml

```
fs.s3a.fast.upload = true
fs.s3a.fast.upload.buffer = bytebuffer
fs.s3a.multipart.size = 100M
fs.s3a.user.agent.prefix  = User-Agent: APN/1.0 Hortonworks/1.0 HDP/
{{version}}
fs.s3a.connection.maximum = 20
fs.s3a.connection.ssl.enabled = true
fs.s3a.endpoint = s3.us-east-1.amazonaws.com
fs.s3a.fast.upload.active.block = 20
fs.s3a.impl = org.apache.hadoop.fs.s3a.S3AFileSystem
fs.s3a.max.total.tasks = 5
fs.s3a.threads.keepalivetime = 60
```

#/etc/hadoop/3.0.0.0-1634/0/slaves

```
ip-10-11-0-198.ec2.internal
ip-10-11-0-244.ec2.internal
ip-10-11-0-120.ec2.internal
ip-10-11-0-133.ec2.internal
ip-10-11-0-83.ec2.internal
ip-10-11-0-65.ec2.internal
ip-10-11-0-177.ec2.internal
ip-10-11-0-193.ec2.internal
ip-10-11-0-19.ec2.internal
ip-10-11-0-29.ec2.internal
ip-10-11-0-40.ec2.internal
ip-10-11-0-95.ec2.internal
ip-10-11-0-6.ec2.internal
ip-10-11-0-213.ec2.internal
ip-10-11-0-222.ec2.internal
```

#/usr/src/HiBench-master/conf/hibench.conf

hibench.scale.profile	tremendous
hibench.default.map.parallelism	420
hibench.default.shuffle.parallelism	420

#/usr/src/HiBench-master/conf/workloads/micro/wordcount.conf

#datagen

hibench.wordcount.tiny.datasize	32000
hibench.wordcount.small.datasize	320000000
hibench.wordcount.large.datasize	3200000000
hibench.wordcount.huge.datasize	32000000000
hibench.wordcount.gigantic.datasize	320000000000
hibench.wordcount.bigdata.datasize	1600000000000
hibench.wordcount.tremendous.datasize	80000000000000

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

WP-7289-1118

NetApp®