Technical Paper

Performance and Tuning Considerations on Amazon Web Services with SAS® 9.4 Using IBM Spectrum Scale™

Release Information

Content Version: 1.0 May 2018.

Contents

Introduction

Shared file systems are a necessity when implementing multiple SAS 9.4 compute nodes in an Amazon Web Services (AWS) environment. There are many options and questions about shared file systems. This paper presents recent testing and research performed by SAS and IBM using SAS 9.4 with IBM Spectrum Scale as the shared file system. It covers a range of topics from performance implications to best practices and tunings to ideal AWS infrastructure configurations. This paper is a basis for an informal round-table discussion that will provide some guidance, solicit customer experiences, and create group talk about trends and feedback that will be helpful to SAS, the IBM Spectrum Scale team, and the Amazon Web Services (AWS) offering teams.

This paper represents test results for SAS 9 workloads using AWS Elastic Cloud Computing (EC2) instances. The testing was conducted using SAS 9.4 and the IBM Spectrum Scale version 4.2.3.5 file system. The configuration included four i3.8xlarge AWS EC2 instances for the compute nodes and four r4.8xlarge AWS EC2 instances for the IBM Spectrum Scale Network Shared Disk (NSD) nodes.

Before starting, it should be noted that the decisions made and the conclusions drawn about AWS EC2 design points were determined when this paper was written in the first quarter of 2018. Be aware that AWS EC2 offerings are constantly changing. It is in your best interest to understand the rationale used in the selection process. Consider what was done for these specific results as a point-in-time design. Future improvements in AWS offerings might change the selections that were made here.

Test Bed Description

This effort used a scaled test bed of one and four compute nodes running a SAS mixed analytics workload. Of interest was whether the EC2 instances using IBM Spectrum Scale as the clustered file system (CFS) would yield benefits for SAS large-block sequential IO patterns.

The test bed chosen for SAS 9.4 in AWS with IBM Spectrum Scale testing was a SAS mixed analytics workload. This was a scaled workload of computation and IO-oriented tests to measure concurrent, mixed job performance.

The actual workload was composed of 19 individual SAS tests: 10 computation, two memory, and seven IO-intensive tests. Each test was composed of multiple steps. Some tests relied on existing data stores, and other tests (primarily, computation tests) relied on generated data. The tests were chosen as a matrix of long-running and shorter-running tests, ranging in duration from approximately five minutes to one hour and 20 minutes. Actual test times vary by hardware-provisioning differences. In some instances, the same test (running against replicated data streams) was run concurrently and/or back-to-back in a serial fashion to achieve an average of *20 simultaneous streams of heavy IO, computation (fed by significant IO in many cases), and memory stress. In all, to achieve the 20-concurrent test matrix, 77 tests were launched.

Data and IO Throughput

The IO tests input an aggregate of approximately 300GB of data, and the computation tests input more than 120GB of data for a single instance of each SAS mixed analytics workload within 20 simultaneous tests on each node. Much more data is generated as a result of test-step activity and threaded kernel procedures such as the SAS SORT routines (e.g., SORT makes three copies of the incoming file to be sorted). As stated, some of the same tests run concurrently using different data. Some of the same tests are run back-to-back to produce a total average of 20 tests running concurrently. This raises the total IO throughput of the workload significantly.

In Figure 1, the aggregate SASDATA I/O bandwidth quickly exceeds 2.5GB/sec and achieves a peak of about 4GB/sec with the workload. This is a good, average, SAS-shop-throughput characteristic for a four-node cluster. This throughput is cumulative from all three primary SAS file systems shared by all four nodes. Note that no SASWORK

I/O is reflected in this figure.

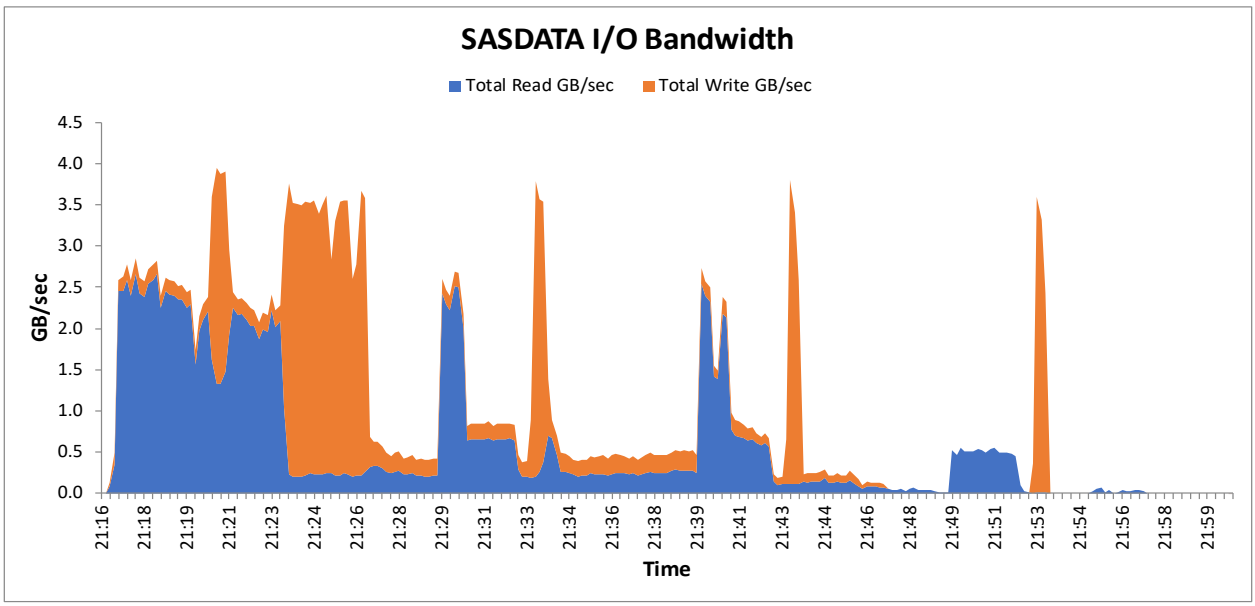

Figure 1. I/O Bandwidth of SASDATA in the AWS Cloud

SAS File Systems

There are three primary file systems involved in the testing.

- SAS permanent data file space—SASDATA
- SAS working data file space—SASWORK
- SAS utility data file space—UTILLOC

The SASDATA file system resides on IBM Spectrum Scale. The SASWORK and UTILLOC file systems reside on Red Hat Enterprise Linux XFS.

The XFS file systems were configured on each compute node using local ephemeral storage with multiple drives. Ephemeral storage is unique in that storage is deleted and all data is lost every time an AWS EC2 instance is stopped. Therefore, it requires the file system to be re-created and restriped on each reboot or launch of the instance.

For this workload's code set, data, result space, working space, and utility space, the following space allocations were made:

- SASDATA—100TB of EBS st1 storage
- SASWORK—7.6TB of local ephemeral storage (NVMe)
- UTILLOC is combined with SASWORK space

This gives you a general size of the workload's on-storage footprint. It is important to note that throughput, not capacity, is the key factor in configuring storage for SAS performance.

With regard to the enormous amount of file space (100TB) reserved for SASDATA, the input data for tests is only a few TBs in size. However, the over-provisioning of storage is required to achieve better throughput. To achieve a sustained 500MB/sec per EBS volume, we had to create volumes that were greater in size than 12TB. For more information about why this was done, see AWS EBS documentation at https://aws.amazon.com/ebs/.

Hardware Description

This test bed was run against one and four compute nodes using the SAS mixed analytics workload with 20 simultaneous tests. The storage nodes were configured as four IBM Spectrum Scale NSD nodes. All compute nodes were identical in configuration. All storage nodes were identical in configuration.

AWS EC2 Instance Compute Node Description

AWS offers a variety of systems to choose from for the compute nodes. The i3.8xlarge system was chosen. The selection process considered the number of CPU cores, memory per core, onboard disk drives, network speed, maximum throughput, and cost per hour.

Here are the specifications for the four AWS EC2 compute nodes:

Host: AWS EC2 i3.8xlarge instances
Kernel: Linux 3.10.0-693.el7.x86_64 (RHEL 7.4 HVM)
CPU: 16 cores, Genuine Intel® Xeon® CPU E5-2686 v4, Family 6 Model 79, Stepping 1, 2 Socket, x86_64, 2.3GHz
AWS Available: 1 socket, 16 cores
Memory: 244GB
Disks: 1x10GB EBS virtual disk for OS and system usage, 4x1.9TB NVMe SSD onboard (ephemeral storage)– striped together
Network: 10Gigabit
Maximum Throughput: 875MB/sec from system NIC to EBS storage

As of this writing, two metrics worth noting are the 16-core system and the 875MB/sec throughput. This implies a throughput-per-core ratio of 875/16=54.6MB/sec/core. SAS applications can operate at higher ratios typically at or above 125MB/sec/core. Because of the throughput limitations, the performance scores will be limited in comparison to similar systems without this restriction.

Host Tuning

Host tuning is based on the Red Hat throughput-performance profile with modifications performed and saved as sas-performance. See the **Appendix** for information about the actual modifications and other pertinent information.

Additional Settings Used

The four NVMe disks were formatted and striped as an XFS file system and used for SASWORK and UTILLOC. The file systems were created as a logical volume consisting of four LUNs striped at a 64KB-block size to match SASBUFSIZE. See the **Appendix** for additional instructions on the procedures used to create the XFS file system and provisions for the directories.

AWS EC2 Instance Storage Node Description

AWS offers a variety of systems to choose from for the storage nodes. The r4.8xlarge system was chosen. The selection process started with knowing the compute node was the i3.8xlarge and trying to match the network speed and throughput performance. The selection process was further narrowed by considering the number of CPU cores, memory per core, and cost per hour. Note that a storage node with a large amount of memory can allow for a larger

IBM Spectrum Scale pagepool size setting. Pagepool can help the effective throughput of the system, but care must be taken when determining this setting because too much can have a negative performance impact.

Here are the specifications for the four AWS EC2 storage nodes:

Host: AWS EC2 r4.8xlarge instance
Kernel: Linux 3.10.0-693.el7.x86_64 (RHEL 7.4 HVM)
CPU: 16 cores, Genuine Intel® Xeon® CPU E5-2686 v4, Family 6 Model 79, Stepping 1, 2 Socket, x86_64, 2.3GHz
AWS Available: 1 socket, 16 cores
Memory: 244GiB
Disks: 1x10GiB EBS virtual disk for OS and system usage
Network: 10Gigabit
Maximum Throughput: 875MB/sec from system NIC to EBS storage

Host Tuning

Host tuning is identical to the compute node. It is based on the Red Hat throughput-performance profile with modifications performed and saved as sas-performance. See the **Appendix** for information about the actual modifications and other pertinent information.

Additional Settings Used

Storage is configured as IBM Spectrum Scale NSD nodes. See the **Appendix** for additional information about IBM Spectrum Scale and its settings.

AWS EC2 Storage Test Description

AWS provides several types of storage. Amazon EBS was chosen due to its highly available, consistent, low-latency block storage attributes. st1 large-block IO was chosen from the EBS storage sub-categories for its large-block IO capabilities. Within EBS, there is general-purpose SSD (gp2), sustained IOPS SSD (io1), and cold storage (sc1).

In general, SAS Foundation requires large-block sequential IO for best performance. We considered whether throughput was more important than IOPS for our workload, and it was. The other storage possibilities were ruled out for the following primary reasons: cold storage (sc1) is not intended for high performance, sustained IOPS SSD (io1) is cost prohibitive, and low-latency SSD (gp2) is not designed for large-block IO.

st1 storage is based on hard disk drive (HDD) technology. Historically, HDD systems that are designed for high throughput require over-provisioning of drives to attain high rates of sustained IO performance. st1 storage is no exception. st1 is designed so that, based on capacity, there is a base IO rate and a burst IO rate with flexibility in performance, capacity, and cost. For more information, see AWS documentation on EBS storage at https://aws.amazon.com/ebs/.

Because of the test environment, it was necessary to select a capacity that insured the burst IO rate equaled the base IO rate so that the storage would not run out of burst credits. When burst credits are depleted, the result is throttling or reduced IO. Exploring burst budgets was not in the scope of this testing. Documentation shows st1 storage has a maximum throughput of 500MB/sec. The minimum capacity needed to ensure that burst IO=base IO was 12.5TiB at the time of this writing.

The compute nodes and storage nodes were selected and matched for throughput at 875MB/sec per node. Therefore, each storage node required two volumes of 12.5TiB to supply the necessary IO for the 875MB/sec maximum throughput of the storage nodes. Doing the math reveals each storage node was provisioned with two 12.5TiB of st1 volumes, for a total of 25TiB per node. The four storage nodes had a total capacity of 100TiB.

This might seem to be an extreme amount of storage. However, it was necessary to ensure that long-sustained

periods of large-block sequential IO not be affected by the expiration of the burst IO budget credits. Preliminary testing of smaller volumes revealed the IO rates did reduce after long periods of use. See the AWS website for more information.

AWS EC2 Networking Description

SAS mixed analytics workload testing was performed in the AWS cloud using an AWS virtual private cloud (VPC). The VPC provides a logically isolated virtual network in the AWS cloud. It was configured in the us-east-1a availability zone. Some of the essential components within the VPC include a public subnet, network address translation (NAT) gateway, and a private subnet. Figure 2 shows (in AWS notation) the system components used in the testing environment.

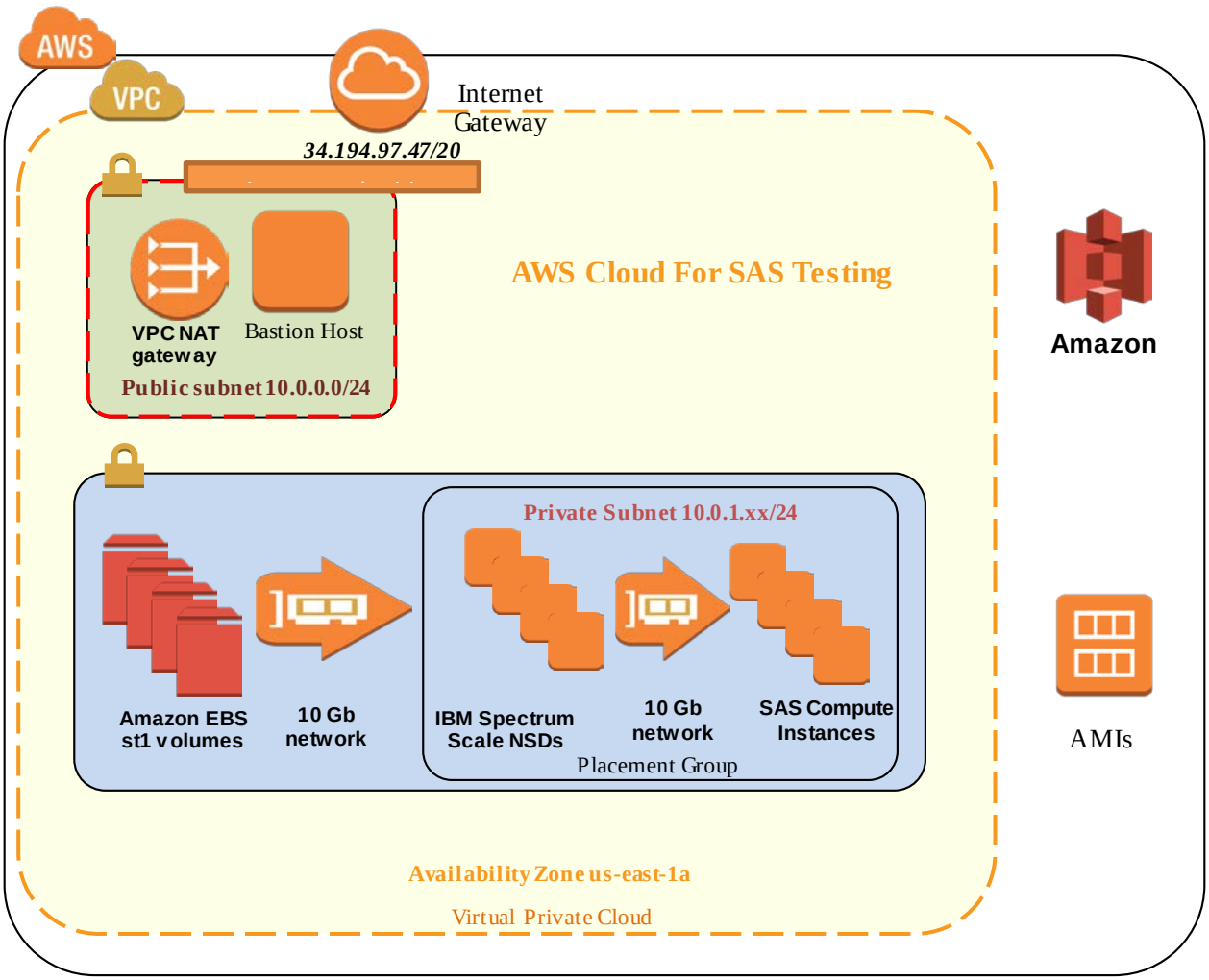

Figure 2. AWS VPC Configuration for SAS Testing

The test setup was initially deployed using an AWS Quick Start Script developed by the IBM Spectrum Scale team. Working in collaboration with this team, the script was improved during testing. At the time of this writing, the Quick Start Script can be accessed at https://aws.amazon.com/quickstart/architecture/ibm-spectrum-scale/. The definitions of the VPC created using the Quick Start Script are listed in the **AWS Quick Start Script Using IBM Spectrum Scale** section.

The compute nodes were i3.8xlarge systems with a network bandwidth of 10Gbit/sec. The storage nodes (NSDs) were sized as r4.8xlarge and had a 10Gbit/sec network bandwidth. It is important to note that when the VPC was created, the instances were defined using an AWS placement group in one availability zone only. This is important for two performance reasons:

1. Placement groups ensure that all nodes are in close proximity to each other. In the case where instances are not close to each other, the 10Gbit instances could operate at a lower network bandwidth of 5Gbit/sec.

2. A placement group can span availability zones within a VPC. AWS recommends doing so for high availability (HA). However, it was decided not to span availability zones for this testing because doing so automatically limits the network performance from 10Gbit/sec to 5Gbit/sec. More information about placement groups can be found at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html.

The AWS VPC had a public subnet that used an AWS elastic IP of 34.194.97.47 and was the route to the internet. In this type of setup, outbound traffic is routed through the NAT instance, which provides isolation from public internet traffic while allowing outbound traffic from the public subnet (10.0.0.0/20).

The private subnet (10.0.2.xx/24) is used to route traffic within the internal network. The compute nodes and storage nodes are in the private subnet.

Security groups are a part of the network fabric and are important for privacy and security. They act as firewalls and can restrict outbound and inbound traffic. For the purpose of this testing, the security groups were minimally configured and are not covered in this paper.

Test Results

The SAS mixed analytics workload was run in a quiet setting. There was no competing activity on the server or storage. Multiple runs were performed to standardize results.

Table 1 shows the performance of a single node run with SASDATA on XFS, a single node run with SASDATA on IBM Spectrum Scale, and the per-node average of a four-node run with SASDATA on IBM Spectrum Scale. SASWORK/UTILLOC resided on local NVMe SSDs (ephemeral storage) for all of these runs. This table shows a frequency mean value of the CPU/real-time ratio, summed from all of the 77 tests submitted. It shows the summed User CPU Time and summed System CPU Time in minutes.

Number of Nodes	SASDATA File System Type	Mean Value of CPU/Real Time Ratio	Elapsed Real Time in Minutes—Workload Aggregate	User CPU Time in Minutes—Workload Aggregate	System CPU Time in Minutes—Workload Aggregate
1-Node	XFS	1.11	804	850	87
1-Node	Spectrum Scale	1.10	817	866	83
4-Node (avg)	Spectrum Scale	1.08	824	868	83

Table 1. Performance

The third column shows the ratio of total CPU Time (User + System CPU) to total Real Time. If the ratio is less than 1, then the CPU is spending time waiting on resources, usually IO. The AWS test configurations delivered excellent ratios ranging from 1.08 to 1.11 of Real Time to CPU. The natural question is, "How can I get above a ratio of 1.0?" Because some SAS procedures are threaded, you can actually use more CPU cycles than wall-clock time (Real

Time).

The fourth column shows the total elapsed Real Time in minutes, summed from each of the jobs in the workload. The AWS-configured single node run with SASDATA on XFS executes the aggregate run time of the workload in approximately 804 minutes of total execution time. The one- and four-node runs with SASDATA on IBM Spectrum Scale executes the aggregate run time of the workload in 817 minutes and 824 minutes of real time per node, respectively.

The primary take-away from these tests is that the one- and four-node AWS configurations using IBM Spectrum Scale NSDs easily provided enough throughput to fully exploit this host environment. The four-node AWS configuration using IBM Spectrum Scale could meet the accelerated and scaled throughput demand, while providing a very healthy CPU/Real Time ratio per node.

Conclusion

These conclusions come with a caveat. The results are good relative to the environment and the envelope of performance.

The governance of the 10Gbit network fabric in AWS constrains the overall IO potential. This is due to the NIC-to-EBS limit of 875MB/sec throughput of the best performing systems today in 10Gbit network fabric. This equates to a potential IO of 54.6MB/sec/core of the 16 core systems tested.

As of this writing, SAS recommends at least 125MB/sec/core of throughput, or more than 2.28 times the current AWS limits. Understanding this point is key and essential to reading the conclusions—the results are good relative to the potential performance.

You might have questions about the 25Gbit systems offered in AWS that are 2.5 times faster in fabric and have full duplex (bi-directional) capability in the network. These systems were tested and found to have little improvement in performance and were more expensive. This is due to the 25Gbit systems having double the cores (i3.16xlarge-32 cores) and double the throughput (1,750MB/sec) to equal the same throughput-per-core ratio of 54.6MB/sec/core (1750MB/sec/32 cores). Note that the i3.8xlarge was not available in 25Gbit fabric as of this writing. Had it been available, there might have been significantly higher scores.

This information intends to condition you to take the proverbial “grain of salt” with the conclusions. Given the limits of the environment and having an understanding of the performance impacts, the systems were designed and tuned to their best potential. That being considered, the results are good.

In summary, the results are good. The test demonstrates that the AWS i3.8xlarge server instance has a capability of providing reasonable performance for a SAS mixed analytics workload in this performance environment. Also, the r4.8xlarge server instance was a good choice for the storage nodes because it matched the performance metrics of the compute nodes for throughput, network bandwidth, and memory.

As AWS is constantly evolving, if the throughput limits per core were to increase, the resulting scores would likely have a corresponding benefit.

The IBM Spectrum Scale shared file system proved to be of value in using the large, high-throughput EBS st1 spinning-array block storage supporting four compute nodes. Using the XFS file system for comparison and considering the purpose of the shared file system, the scores demonstrate the performance benefits of IBM Spectrum Scale.

A final note: You should consider that the workload used in this testing attempts to model a mixed representation of what an average SAS shop might be executing at any given time. Because your results might vary from those presented in this paper, it is crucial to plan and perform your own testing to confirm that this is a viable solution for your particular needs. AWS might be able to satisfy a business need for your company in this domain of high-

performance analytics. Be sure to consider all of the factors involved with regard to performance and cost so that you can set the proper expectations.

Acknowledgments

This project was a collaboration between IBM and SAS. Special thanks go to the extended IBM Spectrum Scale AWS team (Gautam Shah, John Lewars, and Dheren Singh), the SAS Technical Solutions support team, and the AWS support team, with a special nod to Joe Baron from Amazon for his architecture-enablement training and support during this project.

Contact Information

Your comments and questions are valued and encouraged. Contact the authors at:

Ben Smith
IBM Software Defined Computing
+1 (919) 254-2675
smithbe1@us.ibm.com

Jim Kuell
SAS Institute Inc.
+1 (919) 531-3784
Jim.Kuell@sas.com

Brian Porter
IBM ISV Enablement/Spectrum Scale
+1 (720) 430-7674
bporter1@us.ibm.com

Appendix

IBM Spectrum Scale

IBM Spectrum Scale is a powerful shared file system that provides world-class reliability, scalability, and availability for cloud, big data analytics, and high-performance computing environments. IBM Spectrum Scale simplifies data management with integrated tools designed to help manage from gigabytes to petabytes of data and thousands to billions of files. Key features of IBM Spectrum Scale include:

- Unified block, file, and object storage.
- Massively parallel data access for high performance.
- True software-defined storage deployment using either cloud storage on-prem commodity hardware or the powerful Elastic Storage Server grid architecture.
- Transparent cloud-tiering for hybrid cloud storage in minutes.
- Integrated user interface with the entire IBM Spectrum storage family for simplified administration.
- Comprehensive data lifecycle management tools, including transparent policy-driven data migration, high-speed metadata scanning, and native data compression and encryption.

The SAS testing team described IBM Spectrum Scale as "*a mature and scalable clustering product that has been tested and proven with SAS workloads to have specific advantages compared to competing products*."

AWS Quick Start Script Using IBM Spectrum Scale

This web link provides the AWS Quick Start Script used to create the VPC, security groups, placement groups, NAT, bastion box, and other components of the test environment: https://aws.amazon.com/quickstart/architecture/ibm-spectrum-scale/.

The AWS VPC cluster was built with the following parameters:

Big5-Redux-ClusterStack-B5LW	
AmiID	ami-de8aeea4
BastionSecurityGroupID	sg-c11d68b4
BlockSize	2M
ComputeInstanceType	i3.8xlarge
ComputeNodeCount	4
CreatePlacementGroup	TRUE
DataReplica	1
DiskPerNode	2
DiskSize	12500
EBSType	st1
GpfsMountPoint	/gpfs/fs1
KeyPairName	sasaws-va

Big5-Redux-ClusterStack-B5LW	
LicenseAgreementTerms	Accept
OperatorEmail	smithbe1@us.ibm.com
PrivateSubnet1ID	subnet-1b197550
PrivateSubnet2ID	subnet-1b197550
RootVolume	100
ServerInstanceType	r4.8xlarge
ServerNodeCount	4
SpectrumS3Bucket	spectrum-scale-bucket
VpcId	vpc-bfa0d5c7

IBM Spectrum Scale Tunables

The IBM Spectrum Scale clustered parallel file system has many tunable parameters. There are typically only a few of these that are changed to benefit the performance for SAS. The following commands were used to configure the file system for AWS testing:

```
mmchconfig workerThreads=1024
mmchconfig prefetchPct=40
mmchconfig maxFilesToCache=1M
mmchconfig maxMBpS=2048
mmchconfig pagepool=64G
```

Red Hat Tuning of the AWS Instances

The following tunings that were used for the i3.8xlarge AMI compute nodes are based on the Red Hat Enterprise Linux 7.4 operating system. Any other version of the operating system might require different settings. See the following document for more information about configuring Red Hat systems for SAS workloads: http://support.sas.com/resources/papers/proceedings11/342794_OptimizingSASonRHEL6and7.pdf.

Here are the steps for tuning:

1. Navigate to the tuned directory and create a sas-performance profile as a copy of the throughput-performance profile. Edit its tuned.conf file.

   ```
   cd /usr/lib/tuned/
   cp -r throughput-performance/ sas-performance
   vi sas-performance/tuned.conf
   ```

2. Edit the tuned.conf file to appear as follows:

   ```
   [cpu]
   force_latency=1
   governor=performance
   energy_perf_bias=performance
   min_perf_pct=100
   [vm]
   ```

```
transparent_huge_pages=never
[sysctl]
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
vm.dirty_ratio = 30
vm.dirty_background_ratio = 10
vm.swappiness=30
```

3. Set the new tuned profile as the active profile. This can be validated with `tuned-adm active`.

```
tuned-adm profile sas-performance
```

4. Change the ec2-user's max open files to 500,000 and max user processes to 131,072:

```
vi /etc/security/limits.conf
```

5. Add the following two lines:

```
ec2-user          -       nofile            500000
ec2-user          -       nproc           131072
```

6. Log off and back on after performing these steps. Verify changes with `ulimit -a`.

Provisioning the Ephemeral Storage

The SAS compute nodes are i3.8xlarge systems. They each include four 1.9TiB NVMe disk drives suitable for use as a combined SASWORK/UTILLOC directory. The NVMe drives are called “ephemeral” because all current information is lost every time the instance is started or rebooted. They are temporary by design.

To use these ephemeral drives to their best advantage, the following commands were executed every time the instance was started or rebooted. These commands created the physical volumes, volume group, logical volume, and XFS file system and mounted the drives for use in the testing performed in this paper.

Use cases and environments vary, so consider this as just a reference that can be used to model similar tasks in creating local file systems for use with SAS. In this specific case, the four drives were striped as a RAID 0 system with a 64KB block size to match the setting of SAS BUFSIZE and a 16MB read ahead.

```
# ================================================================
# ---- Commands to Re-Create Volumes for Striped /saswork Directory
#
# ================================================================
#
# ---- 1st Time setup:
#      Need to run as root
#      Need lvm2 package installed on a New Instance
#      Need to create /saswork directory
#
# yum install lvm2
# mkdir /saswork
#----------------------------------------------------------------------
# ---- Rebuild the Physical Volumes, Volume Group, Logical Volume, xfs
```

```
#      filesystem and mount /saswork
#
# List devices
 blockdev –report
# Create physical volumes from four ephemeral devices
 pvcreate /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
# View physical volumes
 pvdisplay
# Create a volume group with the four physical volumes
 vgcreate vg_sas_ephemeral /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 dev/nvme3n1
# View volume groups
 vgdisplay
# Create a logical volume from the volume group with a 64 KB block size
#   and a 16 MB read ahead
 lvcreate -l100%FREE vg_sas_ephemeral -n lv_sas_ephemeral -i4 -I64 -r32768
# View detailed info about logical volumes
 lvs -o name,vg_name,size,attr,lv_size,stripes,stripesize,lv_read_ahead
# Create the xfs file system
 mkfs.xfs /dev/vg_sas_ephemeral/lv_sas_ephemeral
# Mount the saswork file system
 mount /dev/vg_sas_ephemeral/lv_sas_ephemeral /saswork
# View file systems
 df –hT
```

To contact your local SAS office, please visit: sas.com/offices