Technical Paper

Performance and Tuning Considerations for SAS® Grid® Manager 9.4 on Amazon (AWS) Cloud using Intel Cloud Edition for Lustre File System

Release Information Content Version: 1.0 April 2015.

Contents

SAS® Grid® Manager 9.4 Testing on AWS using Intel Cloud Edition for Lustre .1
Test Bed Description .1
Data and IO Throughput .1
Instance Description .2
Networking .3
Storage .4
Test Results .5
General Considerations and Tuning Recommendations .6
General Notes .6
Configuring Lustre .6
Configuring SAS Grid Manager .6
Configuring the Storage .7
Conclusion .7
Resources .7
initdiskcluster.sh .8
initeph.sh .9
install-lustre-client.sh .11
mountlustre.sh .11
Network Template .12
SAS Instance Template .14
Intel Lustre Template .15

SAS® Grid® Manager 9.4 Testing on AWS using Intel Cloud Edition for Lustre

Testing was conducted with the Intel Cloud Edition for Lustre file system using SAS Grid Manager 9.4 on four i2.8xlarge Amazon EC2 instances. The testing was to determine a relative measure of how well SAS performed with IO heavy workloads compared with a SAS Grid Manager configuration with SAS WORK and UTILLOC (SAS temporary files) on local files system (EXT4) and Lustre. Of particular interest was whether the EC2 instances using Lustre as the clustered file system would yield substantial benefits for SAS large-block, sequential IO patterns. We will describe the performance tests, the hardware used for testing and comparison, and the test results.

In addition, we would like to document all the steps we went through in setting up the EC2 instances and the Lustre file system. This includes guidelines for the number of EC2 instances used, how they were configured using templates to create the Lustre file system and scripts to attach the ephemeral storage to the SAS Grid nodes.

Test Bed Description

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

The actual workload chosen was composed of 19 individual SAS tests: 10 computation, 2 memory, and 7 IO intensive tests. Each test was composed of multiple steps, some relying on existing data stores, with others (primarily computation tests) relying on generated data. The tests were chosen as a matrix of long running and shorter-running tests (ranging in duration from approximately 5 minutes to 1 hour and 53 minutes, depending on hardware provisioning. 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 30 simultaneous streams of heavy IO, computation (fed by significant IO in many cases), and Memory stress. In all, to achieve the 30-concurrent test matrix, 101 tests were launched.

Data and IO Throughput

The IO tests input an aggregate of approximately 300 Gigabytes of data, and the computation over 120 Gigabytes of data – for a single instance of each test. Much more data is generated as a result of test-step activity, and threaded kernel procedures such as SORT (e.g. SORT makes 3 copies of the incoming file to be sorted). As stated, some of the same tests run concurrently using different data, and some of the same tests are run back-to-back, to garnish a total average of 30 tests running concurrently. This raises the total IO throughput of the workload significantly. In its run span, the workload quickly jumps to 900 MB/sec, climbs steadily to 2.0 GB/s, and achieves a peak of 4+ GB/s throughput before declining again. This is a good average “SAS Shop” throughput characteristic for a single-instance OS (e.g. non-grid). This throughput is from all three primary SAS file systems: SASDATA, SASWORK, and UTILLOC.

SAS File Systems Utilized

There are 3 primary SAS libraries involved in the testing:

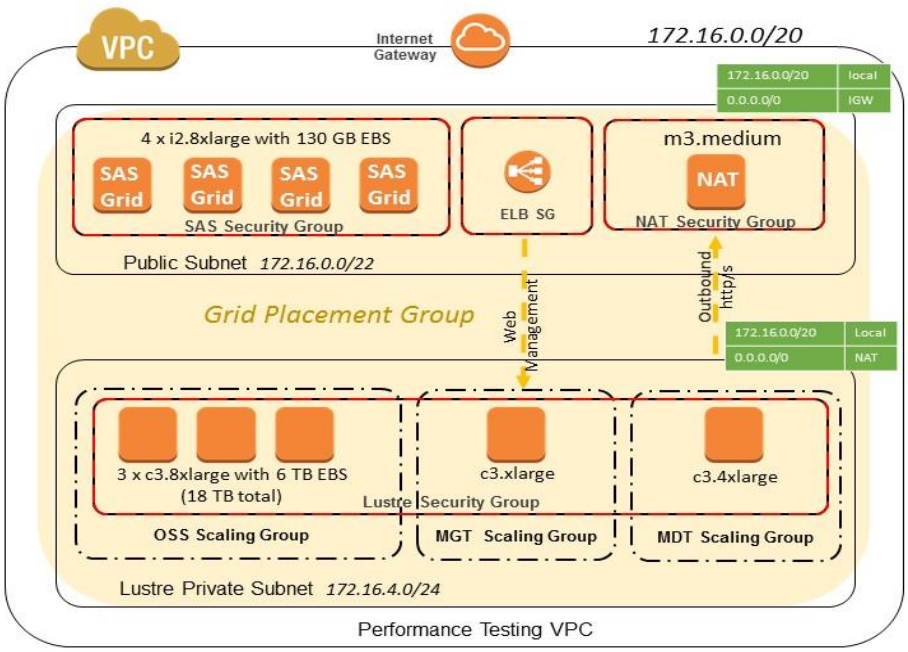
- SAS Permanent Data File Space (Lustre) - SASDATA
- SAS Working Data File Space(ephemeral or Lustre)– SASWORK
- SAS Utility Data File Space(ephemeral or Lustre) – UTILLOC

One test had permanent storage on the Lustre file system while temporary storage (SASWORK and UTILLOC) was on the local file system, called ephemeral storage in AWS. The other test had all three libraries on the Lustre file system.

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

- SASDATA – 3 Terabytes
- SASWORK – 3 Terabytes
- UTILLOC – 2 Terabytes

This gives you a general "size" of the application's on-storage footprint. It is important to note that throughput, not capacity, is the key factor in configuring storage for SAS performance. Fallow space is generally left on the file systems to facilitate write performance and avoid write issues due to garbage collection when running short on cell space.



Instance Description

A list of the AWS EC2 instances, each running Read Hat Enterprise Linux 6.6, that were used for the testing are listed below.

SAS Servers		
SAS Grid node1	i2.8xlarge	us-east-1e
SAS Grid node2	i2.8xlarge	us-east-1e
SAS Grid node3	i2.8xlarge	us-east-1e
SAS Grid node4	i2.8xlarge	us-east-1e
Lustre Nodes:		
Mgt	c3.xlarge	us-east-1e
Metadata	c3.4xlarge	us-east-1e
ost00	c3.8xlarge	us-east-1e
ost01	c3.8xlarge	us-east-1e
ost02	c3.8xlarge	us-east-1e
NAT	m3.medium	us-east-1e

The i2 instances are Storage Optimized instances designed for high I/O workloads. The c3 instances are Compute Optimized instances. All the instances are run on Intel E5-2670 v2 (Ivy Bridge) or E5-2680 v2 (Ivy Bridge) processors. For more details on the many types of EC2 instances, please refer to this web site https://aws.amazon.com/ec2/instance-types/.

Networking

The SAS Grid Manager testing used an Amazon Virtual Private Cloud (VPC) configured with a public and private subnet. The Amazon Virtual Private Cloud provides an isolation boundary fully controlled by the customer. The above diagram shows the system components in the testing environment.

The test setup deployed all instances within a placement group. Placement groups are used for workloads that have demanding network requirements and ensures close proximity from a network viewpoint. A placement group can span subnets within a VPC that are in the same availability zone (consider an availability zone similar to a single data center within an AWS region). See Placement Groups for more information. The 8xlarge instance types have full bisection bandwidth of 10 gigabits per second placed within the placement group. Network bandwidth is determined by instance size, therefore the c3.xlarge and c3.4xlarge will have lower bandwidth. The Lustre cluster has a management (MGT), a metadata (MDT) and several Object Storage Servers (OSS) nodes. The Lustre cluster also uses a Network Address Translation (NAT) node further described below. The OSS instances serve data and cluster capacity is primarily determined by the number of OSS nodes, and the amount of storage attached to each OSS node. The OSS nodes are sized as c3.8xlarge nodes to take advantage of the 10g network since those nodes serve the actual data and are the primary driver of cluster throughput performance. Similarly the SAS instances require high throughput to the shared file system over the 10g network and are sized as i2.8xlarge. The i2.8xlarge provides fast ephemeral (local) SSD storage for temporary space (saswork and utilloc) in addition to 10g network bandwidth to shared storage. The Lustre management, metadata, and NAT nodes do not have as demanding network throughput demands, and are appropriately sized to smaller instance types.

The VPC has one subnet with a route defined the internet (Public Subnet) and a second subnet that only routes within the internal network (Lustre Private Subnet). Some of the instances in the Lustre Private Subnet require outbound public access in order to interact with AWS web service APIs. These AWS services are used in extensions developed by Intel that provide additional high availability capabilities for Lustre running that are not available in the standard Lustre version. The outbound http/https traffic is routed through the NAT instance in the public subnet. This configuration provides isolation for the subnet from any public internet traffic while allowing outbound http/https traffic to route through the NAT instance from the Lustre Private Subnet. The MGT provides a web interface for monitoring the Lustre environment. The management interface is exposed externally through an Amazon Elastic Load Balancer (ELB) instance. The SAS client instances are deployed to the public subnet, although in a production setting they could also be deployed to a subnet that is not accessible from the internet.

Security groups act as stateful firewalls. Security groups have the ability to restrict outbound traffic and inbound traffic, although in the tested configuration no outbound rules were defined. The system has four security groups defined:

SAS SG	
ALL	SASSG
22	0.0.0.0/0

ELB SG	
80	0.0.0.0/0

NAT SG	
22	0.0.0.0/0
80	172.16.4.0/24
443	172.16.4.0/24

Lustre SG	
ALL	LustreSG
22	0.0.0.0/0
80	0.0.0.0/0
988	0.0.0.0/0

- The SAS security group – This security group provides access to the SAS instances and has two rules. Secure shell access through Port 22 is permitted from any endpoint. The second rule allows any traffic between nodes that are in the SASSG to allow SAS Grid nodes to communicate with each other. This could be further locked down to specific ports by removing the ALL rule, and only allowing the ports identified in the Configuring SAS Grid Manager section. Often in a production environment these servers will be placed in a private subnet that is only reachable over a virtual private network (VPN) connection back to your corporate network. In that case you would restrict the IP address range to your internal corporate IP address range. Alternatively if you do not have a VPN connection, you can either use a jump host (in this case the NAT instance serves as a jump host for Lustre instances) and place the SAS instances in the private subnet,

or leave the SAS instances in the public subnet, and restrict the IP address range to your corporate public IP address range.

- The ELB security group lets users access the monitoring web site that is served from the MGT (management) server. The web requests go to the ELB instance, and then are forwarded to the MGT server on its private IP address so the MGT server is not directly exposed to the internet. Access should also be locked down to your internal network or public corporate IP address range for production. This also can be configured to only be accessible from a VPN connection.
- The NAT security group permits inbound traffic on port 80 and 443 from the private Lustre Private Subnet IP address range, and port 22 from any address. Port 80 and 443 allow instances in the private subnet to get out to the internet on http and https. Port 22 access allows the instance to be used as a jump host to reach the instances in the Lustre Private Subnet. In production environments port 22 should be locked down to your internal network over VPN or to your public corporate IP address range.
- The Lustre security group – Protects all servers in the Lustre cluster. This security group allows traffic on ports 80 (http), 443(https), and 22(ssh) to all addresses as tested. The security group also allows traffic on all ports between all members within the Lustre Security Group (unless explicitly defined access is not permitted between instances within the same security group). This security group could be constrained to only allow the IP address range of the VPC for http and https. The subnet is unreachable from the internet for inbound traffic like http or https because of the routing configuration. This security group configuration is recommended for a production environment. You will need to modify the Cloud Formation template from Intel discussed in the following section to apply these settings.

The Lustre configuration uses three autoscaling groups. Autoscaling is a capability of AWS to automatically scale the number of instances up or down in a group depending upon some metric. It also automatically replaces any instances that are determined unhealthy. The Lustre system uses autoscale groups to create the instances it needs, and to ensure that if an instance fails, it is automatically replaced. This is achieved by defining the minimum and maximum as the same value (for example in this case the OSS autoscale group has a minimum of 3 and maximum of 3). The autoscale group replaces the node on failure, and a feature built by Intel for AWS automatically reattaches storage for the failed node to the replacement in order to automatically recover the cluster for you on a failure. While not shown in the diagram, the Lustre configuration also uses a small table in DynamoDB (a no SQL database as a service available in AWS) in order to track assignments of volumes to OSS instances as part of its auto-recovery HA feature.

Storage

The storage file system is comprised of attached local storage (ephemeral) and permanent storage on the Lustre file system (Version 2.5.3).

- Each of the four SAS Grid nodes has (8) 800 GB SSD volumes of ephemeral storage.
- The Lustre file system is comprised of a management node, a metadata node, and 3 OSSs (the nodes that serve the data for the file system). The OSSs each have 6 TB of EBS general purpose SSD storage (also called gp2) for a total of 18TB of storage.

The Lustre storage throughput capacity is gated by the network throughput of the OSSs that serve file data (10 Gb/per/sec network) and the throughput of the EBS volumes attached to the OSSs (3 IOPS/GB, with a maximum output of 128 MB/per/sec /volume). The general purpose SSDs perform well for large sequential workloads at a lower price point than provisioned IOP (PIOP) volumes. The IOPs rate of a general purpose SSD volume is determined by size therefore storage is overprovisioned in order to get the throughput capacity. The same IOPs rate can be achieved at a lower storage volume using PIOP volumes, although the overall cost will be higher. Amazon replicates EBS storage on the backend for fault tolerance. Ephemeral storage is not recommended for permanent storage by systems that don't have a built-in replication scheme, like HDFS. See Amazon EBS Volume Performance on Linux Instances for more information.

Since the 10 Gb/per/sec network is used for the network traffic to EBS and the network traffic to the SAS clients, the realized file system throughput per Lustre OSS is expected to be in the 500 MB/per/sec range and will scale relatively linearly with the number of OSSs in the cluster. If the file operations often hit the read cache then throughput may be higher since the network traffic to EBS is reduced. Instance size in AWS dictates the network performance and 8xlarge instance types are recommended in order to get 10 Gbps network performance.

Test Results

The table below shows an aggregate of the SAS Real Time in minutes, summed for all the SAS jobs submitted. It also shows summed Memory utilization, summed User CPU Time, and summed System CPU Time in Minutes.

Location of SAS WORK and UTILLOC	Elapsed Real Time in Minutes – Workload Aggregate	Memory Used in GB - Workload Aggregate	User CPU Time in Minutes – Workload Aggregate	System CPU Time in Minutes - Workload Aggregate
Lustre File System	1493	192.6	1094	434
Local EXT4 File Systems – Single SMP Host	1481	92.42	1750	275

The AWS construction using LUSTRE required less CPU Minutes utilization to perform the Mixed Analytics Workload than the non-virtualized, direct attached storage system– a difference of 497 Total Workload CPU Minutes. In addition the workload experienced a very slightly longer aggregate jobs Real Time of 12 minutes across all 101 jobs.

Another way to review the results is to look at the ratio of total CPU time (User + System CPU) against the total Real time. If the ratio is less than 1, then the CPU is spending time waiting on resources, usually IO. For the testing with all three SAS libraries on the Lustre file system, the ratio is 0.95. And for the testing with SAS WORK and UTILLOC on a local XFS file system, the ratio is 1.15.

The 0.95 Mean Ratio associated with the AWS EC2 GRID instances indicates a good efficiency in getting IO to the CPU for service. For the IO intensive SAS Application set, this is a good outcome, but not quite as good as a direct attached storage system. The question arises, “How can I get above a ratio of 1.0 for the spinning storage?” Because some SAS procedures are threaded, you can actually use more CPU Cycles than wall-clock, or Real time.

Storage System	Mean Ratio of CPU/Real-time - Ratio	Mean Ratio of CPU/Real-time - Standard Deviation	Mean Ratio of CPU/Real-time - Range of Values
Local EXT4 File Systems – Single SMP Host	1.15	0.369	2.107
Amazon Lustre File System	0.95	0.354	2.79

In short our test results were good. It showed that the Amazon i2.8xlarge System, can provide good performance for SAS workloads, in this case with a commensurate run time as a large, high throughput spinning array, utilizing less CPU resources. The workload utilized was a mixed representation of what an average SAS shop may be executing at any given time. Due to workload differences your mileage may vary.

General Considerations and Tuning Recommendations

General Notes

It is very helpful to utilize the SAS tuning guides for your operating system host to optimize server-side performance and additional host tuning is performed as noted below. See: http://support.sas.com/kb/42/197.html

Configuring Lustre

The file system was formatted using default options. Several file system settings were modified for best performance.

On the OSS the option readcache_max_filesize controls the maximum size of a file that both the read cache and writethrough cache will try to keep in memory. Files larger than readcache_max_filesize will not be kept in cache for either reads or writes. This was set to 2M on each OSS via the command:

lctl set_param obdfilter..readcache_max_filesize=2M*

On the client systems, the maximum for dirty data that can be written and queued was increased from the default to 256 MB. Also, the maximum number of in-flight RPC's was increased from the default to 16. These changes were made using the following commands:

lctl set_param osc..max_dirty_mb=256*

lctl set_param osc..max_rpcs_in_flight=16*

In addition to the above commands, you must mount the Lustre file systems with the *–o* flock option on all the application client systems. An example of how to do this is in the lustremount.sh script in Appendix 1.

Configuring SAS Grid Manager

In addition to the above information regarding setting up the Lustre file systems, there are some items that need to be done to the EC2 instances being used as the SAS Grid nodes so that they SAS Grid Manager functions correctly.

For SAS Grid Manager to work the following ports need to be opened on the network connecting the SAS Grid nodes:

7869 / TCP + UDP
6881 / TCP
6878 / TCP
6882 / TCP
6884 / TCP
7870 / TCP
7871 / TCP
7872 / TCP
9090 / TCP
53 / TCP + UDP (if using DNS – if only using host files and not utilizing EGO for High Availability then this is not required)

+ all standard SAS ports such as SAS metadata server port as assigned or defaulted in the SAS Configuration.

Install LSF (which is used by SAS Grid Manager to schedule SAS tasks) in a persistent shared file system.
By default, the hostname of each EC2 instance is its IP address. We recommend that customers change this for each node in the SAS Grid Manager environment to a "user friendly name" including editing */etc/hosts* as well as

/etc/sysconfig/network to specify the desired hostname for each node. These nodes will be used in the LSF configuration when LSF is installed and this way all the nodes will be found should they be restarted.

And we would also recommend installing all SAS binaries and configuration files to a shared file system to minimize administrative tasks such as installation, SETINIT maintenance, etc. Also, this ensures that the path to the SAS launch script in the config directory is the same and is in synch across all nodes. Having a single copy of the SAS binaries and configuration files will make growing the SAS Grid nodes much easier.

Once again, the best location for this will be the Lustre file system. Unlike the LSF installation, this is a best practice, but not an absolute requirement.

Configuring the Storage

To expedite the setup of the storage used for the local file systems and Lustre, our colleagues at Amazon have supplied us with several scripts and templates to use. These are available for your review in Appendix 1 of this document. Please note that these are unsupported scripts and templates for your guidance.

Conclusion

The testing of the above workload shows that a properly configured Amazon AWS infrastructure with robust IO and a Lustre file system can work nicely with SAS Grid Manager.

We have provided some guidance on how to create the AWS infrastructure, but it is crucial to work with your Amazon engineer to plan, install, and tune your AWS EC2 instances to get maximum performance. It is very important to fully understand what all needs to be setup and your intended usage – leave the instances up at all times or only use them when you need them (this will require extra work to make sure that when they are restarted all the file systems are attached properly and all the SAS servers are started properly).

One other guidance is for you to determine the level of support you will need for your EC2 instances and Red Hat software. Basic support comes with your EC2 license, but additional support can be obtained from Amazon Support (https://aws.amazon.com/premiumsupport/). This support does not include support for the Lustre file system. However, you can obtain support for Lustre via the Intel Cloud Edition for Lustre – Global Support service (https://aws.amazon.com/marketplace/pp/B00GBFUO9A). As always, support for SAS issues are available as part of your SAS license.

As always, your mileage may vary based on your planned use of SAS and the size of your data file. Please do not skimp on the size of the AWS EC2 instances as you will need the faster storage and networking associated with them to optimal performance of SAS Grid Manager on the Amazon cloud.

Resources

SAS papers on Performance Best Practices and Tuning Guides: http://support.sas.com/kb/42/197.html

Developing High-Performance, Scalable, cost effective storage solutions with Intel Cloud Edition Lustre and Amazon Web Services. http://www.intel.com/content/dam/www/public/us/en/documents/reference-architectures/ICEL-RA.pdf

Contact Information:

Name: Chris Keyer
Enterprise: Amazon Web Services
Work Phone: +1 (425) 736-4174
E-mail: ckeyser@amazon.com

Name: Margaret Crevar
Enterprise: SAS Institute Inc.
Work Phone: +1 (919) 531-7095
E-mail: margaret.crevar@sas.com

Appendix 1

As mentioned earlier in the paper, Amazon engineers shared with us some scripts and templates to use during the setup of our Amazon configuration. We have listed these scripts below along with notes on what they do and when to use them. These scripts as specific to configuration within Amazon Web Services, so please contact Amazon support for any issues you encounter (https://aws.amazon.com/premiumsupport/).

initdiskcluster.sh

This script is ran on the SAS servers (Lustre clients) to setup the temporary work locations using ephemeral storage and should be ran first on the SAS instances. The script sets configuration for the volumes to SAS recommendations for scheduler, readahead, and transparent huge page. It then creates two RAID 0 volumes for SASWORK and SASUTIL using mdadm from ephemeral storage, formats the RAID volumes using ext4, and mounts the RAID devices. The script also updates the RAID configuration (mdadm.conf) to automatically reassemble the RAID on reboot, updates fstab to automatically mount the RAIDs, and registers a script, initeph.sh, to execute on startup. That script is described next. Please note that the script expects the ephemeral volumes to be located at specific devices (xvd[hijklmno]), and these locations will need to change if you choose to put the volumes on different devices. The settings that need to change are highlighted below. The cloud formation example provided in the following appendix shows how to place the ephemeral volumes at xvd[hijklmno].

```
#!/bin/bash
# NOTE: must run as su

READAHEAD=16384

yum install mdadm -y  -q

#setup noop for scheduler and set read ahead for all volumes.
for disk in `ls -1 /sys/block |egrep '^xvd[b-z]'`;do echo "noop" > /sys/block/$disk/queue/scheduler;
done
for disk in `ls -1 /dev |egrep '^xvd[b-z]'`;do blockdev --setra $READAHEAD /dev/$disk; done


#disable transparent_hugepage
echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled

# create raid0 volumes for saswork, and utilloc
mdadm -Cv -c256 -N saswork /dev/md/saswork -l0 -n5 /dev/xvd[hijkl]
mdadm -Cv -c256 -N utilloc /dev/md/utilloc -l0 -n3 /dev/xvd[mno]

#set readahead for RAID volumes
blockdev --setra $READAHEAD /dev/md/saswork
blockdev --setra $READAHEAD /dev/md/utilloc

#create file system (note xfs perferred on RHEL7)
mkfs.ext4 -b 4096 /dev/md/saswork
mkfs.ext4 -b 4096 /dev/md/utilloc

#create mount points and mount volumes.  sasdata used later to mount lustre.
mkdir /saswork
mkdir /utilloc
mkdir /sasdata
mount /dev/md/saswork /saswork
mount /dev/md/utilloc /utilloc

# setup mdadm.conf so volumes setup on reboot automatically.
echo 'DEVICE /dev/xvd[h-o]' | sudo tee /etc/mdadm.conf
mdadm --detail --scan | sudo tee -a /etc/mdadm.conf

#setup fstab to remount the raid volumes automatically.
workdev=$(cat /dev/md/md-device-map | grep saswork | awk '{print $1}')
workuuid=$(blkid | grep ${workdev} | awk '{print $2}' | sed 's/\"//g')
utildev=$(cat /dev/md/md-device-map | grep utilloc | awk '{print $1}')
utiluuid=$(blkid | grep ${utildev} | awk '{print $2}' | sed 's/\"//g')
datadev=$(cat /dev/md/md-device-map | grep sasdata | awk '{print $1}')
```

```
datauuid=$(blkid | grep ${datadev} | awk '{print $2}' | sed 's/\"//g')
echo "${workuuid} /saswork ext4 noatime,nodiratime 0 0" >> /etc/fstab
echo "${utiluuid} /utilloc ext4 noatime,nodiratime 0 0" >> /etc/fstab

# setup start script so that ephemeral is recreated on stop/start.
# the script initeph.sh detects if the volumes have been configured, and then
# and if not (as is the case with an instance stop then start), it will
# recreate the RAID volumes from scratch.
cp ./initeph.sh /etc/init.d/initeph.sh
chmod +x /etc/init.d/initeph.sh
chkconfig --list initeph.sh
chkconfig --add initeph.sh
```

initeph.sh

This script does not need to be explicitly ran. Rather it is used by initdiskcluster.sh to setup automatic rebuilding of RAIDs configured using ephemeral storage on a stop/start. In AWS ephemeral storage is reset on stop/start (technically a stop then a start brings the system back up on a different virtual instance). As a result you lose ephemeral storage, and therefore the ephemeral RAIDs will no longer be configured. This script first attempt to reassemble a raid volume (in our case SAS WORK and UTILLOC), and then re-creates if it can't. As with the previous script, this depends upon ephemeral volumes being mapped to devices xvd[hijklmno]).. For smaller instances, or different mappings, this script must be updated.

This script can be run manually after a restart of your EC2 instance, or placed in /etc/init.d so it will be restarted during the boot process of the EC2 instance. The initdiskcluster.sh sets up the script to be automatically ran on restart.

```
#!/bin/bash
# chkconfig: 345 10 10
# description: auto create/start ephemeral raid.
#

# This script will attempt to reassemble a raid volume, and re-create if it can't, for
# ephemeral storage.  It depends upon ephemeral being mapped on an i2.8xlarge from h-o.  For
# smaller instances, or different mappings, this script must be updated.

READAHEAD=16384

start() {
  #setup noop for scheduler and set read ahead for all volumes.
  for disk in `ls -1 /sys/block |egrep '^xvd[h-o]'`;do echo "noop" >
/sys/block/$disk/queue/scheduler; done
  for disk in `ls -1 /dev |egrep '^xvd[h-o]'`;do blockdev --setra $READAHEAD /dev/$disk; done

  #disable transparent_hugepage
  echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled

  # create raid0 volumes for saswork, utilloc and sasdata
  mdadm --misc --test /dev/md/saswork

  ec=$?
  rc=0

  if [ ${ec} -ne 0 ]; then
    rc=1
    echo 'attempting to assemble /saswork'
    mdadm --assemble /dev/md/saswork /dev/xvd[hijkl]
    ec=$?

    if [ ${ec} -ne 0 ]; then
      echo 'attempting to create /saswork'
      mdadm --zero-superblock --force /dev/xvd[hijkl]
      mdadm -Cv -c256 -N saswork /dev/md/saswork -l0 -n5 /dev/xvd[hijkl]
      mkfs.ext4 -b 4096 /dev/md/saswork
      blockdev --setra $READAHEAD /dev/md/saswork
    fi

    echo 'mounting saswork'
```

```
    if [ ! -d /saswork/ ]; then
       mkdir /saswork
    fi

    mount /dev/md/saswork /saswork
  fi

  mdadm --misc --test /dev/md/utilloc

  ec=$?

  if [ ${ec} -ne 0 ]; then
    rc=1
    echo 'attempting to assemble /utilloc'
    mdadm --assemble /dev/md/utilloc /dev/xvd[mno]

    ec=$?
    if [ ${ec} -ne 0 ]; then
       echo 'attempting to create /utilloc'
       mdadm --zero-superblock --force /dev/xvd[mno]
       mdadm -Cv -c256 -N utilloc /dev/md/utilloc -l0 -n3 /dev/xvd[mno]
       blockdev --setra $READAHEAD /dev/md/utilloc
       mkfs.ext4 -b 4096 /dev/md/utilloc
    fi

    if [ ! -d /utilloc ]; then
      mkdir /utilloc
    fi
    mount /dev/md/utilloc /utilloc
  fi

  if [ ${rc} -eq 1 ]; then
    #need to cleanup/recreate fstab and mdadm.conf
    sed -i '/saswork/d' /etc/mdadm.conf
    sed -i '/utilloc/d' /etc/mdadm.conf

    echo 'DEVICE /dev/xvd[h-y]' > /etc/mdadm.conf
    mdadm --detail --scan >> /etc/mdadm.conf
    mdadm --detail --scan >> /tmp/initep.txt

    sed -i '/saswork/d' /etc/fstab
    sed -i '/utilloc/d' /etc/fstab
    workdev=$(cat /dev/md/md-device-map | grep saswork | awk '{print $1}')
    workuuid=$(blkid | grep ${workdev} | awk '{print $2}' | sed 's/\"//g')
    utildev=$(cat /dev/md/md-device-map | grep utilloc | awk '{print $1}')
    utiluuid=$(blkid | grep ${utildev} | awk '{print $2}' | sed 's/\"//g')
    echo "${workuuid} /saswork ext4 noatime,nodiratime 0 0" >> /etc/fstab
    echo "${utiluuid} /utilloc ext4 noatime,nodiratime 0 0" >> /etc/fstab
  fi
}

stop() {
  umount /saswork
  umount /utilloc
}

case "$1" in
    start)
       start
       ;;
    stop)
       stop
       ;;
    restart)
       stop
       start
       ;;
    *)
    echo $"Usage: $0 {start|stop|restart}"
```

```
    exit 2
esac
```

install-lustre-client.sh

This script does not have a dependency on any other scripts. This scripts installs the Lustre client on the SAS Grid nodes, or any other node that needs to access the Lustre file system. This script will install the latest Lustre client from the Intel repository. It also sets selinux to permissive since Lustre is not compatible with selinux.

```
#!/bin/bash
# must run as su

cd /etc/yum.repos.d
printf '[lustre-client] \nname=Intel HPPD EL6 Lustre Client - Maintainence Release
\nbaseurl=https://downloads.hpdd.intel.com/public/lustre/latest-maintenance-release/el6/client/
\nenabled=1\ngpgcheck=0\n' > lustre.repo

yum install -y lustre-client

echo '### if lustre information not listed then an error occurred'
modinfo lustre

#set selinux to permissive
setenforce 0
getenforce
```

mountlustre.sh

This script remounts the Lustre file system on each of the SAS Grid nodes after a reboot to the EC2 instance. The ipaddrshare value is determined by running the `lctl list_nids` command and using the IP address that the command echoes back on the management (MGT) or metadata (MDT) nodes.

```
#!/bin/bash
if [ "$3" == ""  ]; then
    echo "usage: mountlustre ipaddr:share mount point"
    echo "example: mountlustre 172.0.0.3 /sasdata /sasdata"
    exit -1
fi

mount -t lustre -o flock tcp@$1:$2 $3

# performance settings recommended by SAS for Lustre.
lctl set_param osc.\*.max_dirty_mb=256
lctl set_param osc.\*.max_rpcs_in_flight=16
```

Appendix 2

Amazon engineers also shared automation templates to help create the environments quickly using Cloud Formation templates. Cloud Formation uses a JSON syntax for specifying resources to create. Intel also provides a Cloud Formation script for launching the Lustre cluster. Please contact AWS support or Intel support if you have any issues. As a precursor to running these scripts you should create a placement group. See Placement Groups.

Network Template

This template defining the network environment (VPC, subnet, security group, etc) to be created.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Creates a VPC network for SAS GRID.",
  "Parameters": {
                "SSHCIDR": {
            "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
            "ConstraintDescription": "must be a valid CIDR range of the form x.x.x.x/x.",
            "Default": "0.0.0.0/0",
            "Description": "CIDR for new private subnet i.e. 10.0.2.0/24",
            "Type": "String"
        }
  },
  "Mappings": {
    "SubnetConfig": {
      "VPC": { "CIDR": "172.16.0.0/20" },
      "PublicSubnet": { "CIDR": "172.16.0.0/22" }
       }
  },
  "Resources": {
    "VPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": { "Fn::FindInMap": [ "SubnetConfig", "VPC", "CIDR" ]},
        "Tags": [
          { "Key": "Name", "Value": "SAS-GRID" },
                { "Key": "Application", "Value": { "Ref": "AWS::StackName" } }
        ]
      }
    },
    "PublicSubnet": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "CidrBlock": { "Fn::FindInMap": [ "SubnetConfig", "PublicSubnet", "CIDR" ]},
        "Tags": [
           { "Key": "Name", "Value": "SAS-GRID-PublicSubnet" },
         { "Key": "Application", "Value": { "Ref": "AWS::StackName" } }
        ]
      }
    },
    "InternetGateway": {
      "Type": "AWS::EC2::InternetGateway",
      "Properties": {
        "Tags": [
          { "Key": "Name", "Value": "SAS-GRID-IGW" },
          { "Key": "Application", "Value": { "Ref": "AWS::StackName" } }
        ]
      }
    },
    "GatewayToInternet": {
       "Type": "AWS::EC2::VPCGatewayAttachment",
       "Properties": {
         "VpcId": { "Ref": "VPC" },
         "InternetGatewayId": { "Ref": "InternetGateway" }
       }
    },
```

```
    "PublicRouteTable": {
      "Type": "AWS::EC2::RouteTable",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "Tags": [
         { "Key": "Name", "Value": "SAS-GRID-RouteTable" },
          { "Key": "Application", "Value": { "Ref": "AWS::StackName" } }
        ]
      }
    },
   "PublicRoute": {
      "Type": "AWS::EC2::Route",
      "Properties": {
        "RouteTableId": { "Ref": "PublicRouteTable" },
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": { "Ref": "InternetGateway" }
      }
    },
    "PublicDMZRouteTableAssociation": {
       "Type": "AWS::EC2::SubnetRouteTableAssociation",
       "Properties": {
               "SubnetId": { "Ref": "PublicSubnet" },
               "RouteTableId": { "Ref": "PublicRouteTable" }
       }
    },
    "SASSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Security group for SAS Grid instances",
            "SecurityGroupIngress": [
                {"IpProtocol":"tcp", "FromPort":"22",  "ToPort":"22",  "CidrIp":{"Ref": "SSHCIDR"}}
            ],
            "Tags": [ {"Key": "Env", "Value": "SAS-GRID"} ],
            "VpcId": {"Ref": "VPC"}
        }
     },
     "SASSecurityGroupAllIngress": {
         "Properties": {
                "FromPort": "0",
                "GroupId": {
                    "Ref": "SASSecurityGroup"
                },
                "IpProtocol": "-1",
                "SourceSecurityGroupId": {
                    "Ref": "SASSecurityGroup"
                },
                "ToPort": "65535"
            },
            "Type": "AWS::EC2::SecurityGroupIngress"
     }
  },
  "Outputs": {
    "VPC": {
      "Description": "VPCId of the newly created VPC",
      "Value":  { "Ref": "VPC" }
    },
    "PublicSubnet": {
      "Description": "SubnetId of the public subnet",
      "Value":  { "Ref": "PublicSubnet" }
    },
       "SASSecurityGroup": {
      "Description": "Security group for SAS Instances",
      "Value":  { "Ref": "SASSecurityGroup" }
    }
  }
}
```

SAS Instance Template

You will need to run a template like this once for each SAS Grid instance you want to create in your cluster. You can write a script in a language like Python or Powershell as well to automatically execute the script for you multiple times or use cfncluster. This shows how to launch a plain RHEL6 instance. You will likely want to pre-install SAS Grid onto an instance, create an Amazon Machine Image (AMI), and replace the mapping for the AWSRHEL6AMI with the AMI id of your AMI. This way you only have to install SAS and reuse for all of your instances.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test instance - load testing SAS",
    "Parameters": {
        "SubnetID":{
            "Description": "SUBNET ID to launch into",
            "Type": "AWS::EC2::Subnet::Id",
            "AllowedPattern": "subnet-[a-zA-Z0-9]+"
        },
        "SecurityGroupId": {
            "Description": "SUBNET ID to launch into",
            "Type": "String",
            "AllowedPattern": "sg-[a-zA-Z0-9]+"
        },
        "SSHKeyName": {
            "Description": "SSH Key name for instance",
            "Type": "AWS::EC2::KeyPair::KeyName"
        },
        "PlacementGroupName": {
            "Description": "Name of an existing placement group",
            "Type": "String",
            "MinLength": "1",
            "MaxLength": "32"
        }
    },
    "Mappings": {
        "AWSRHEL6AMI": {
          "us-west-2": { "AMI": "ami-5fbcf36f" },
          "us-east-1": { "AMI": "ami-aed06ac6" }
        }
    },
    "Resources": {
        "InstanceIPAddress": {
          "Type": "AWS::EC2::EIP",
          "Properties": {
            "Domain": "vpc",
            "InstanceId": { "Ref": "Ec2Instance" }
          }
        },
        "Ec2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "KeyName": {"Ref": "SSHKeyName"},
                "ImageId": { "Fn::FindInMap": [ "AWSRHEL6AMI", { "Ref": "AWS::Region" }, "AMI" ]},
                "InstanceType": "i2.8xlarge",
                "SubnetId": {"Ref": "SubnetID"},
                "PlacementGroupName": {"Ref":"PlacementGroupName"},
                "SecurityGroupIds": [ {"Ref": "SecurityGroupId"} ],
                "DisableApiTermination": true,
                "BlockDeviceMappings": [
                    {"DeviceName": "/dev/sdh","VirtualName": "ephemeral0"},
                    {"DeviceName": "/dev/sdi","VirtualName": "ephemeral1"},
                    {"DeviceName": "/dev/sdj","VirtualName": "ephemeral2"},
                    {"DeviceName": "/dev/sdk","VirtualName": "ephemeral3"},
                    {"DeviceName": "/dev/sdl","VirtualName": "ephemeral4"},
                    {"DeviceName": "/dev/sdm","VirtualName": "ephemeral5"},
                    {"DeviceName": "/dev/sdn","VirtualName": "ephemeral6"},
                    {"DeviceName": "/dev/sdo","VirtualName": "ephemeral7"}
                ],
                "Tags": [{"Key": "Name","Value": "SAS-PERF-CLUSTER"}]
```

```
            }
        }
    },
    "Outputs": {
        "IPAddress": {
            "Description": "The public IP Address of instance",
            "Value": { "Fn::GetAtt": ["Ec2Instance", "PublicIp"] }
        }
    }
}
```

Intel Lustre Template

Intel delivers a supported version of Lustre through the AWS marketplace (see Intel Cloud Edition for Lustre* Software - Global Support (HVM)). Please refer to the Intel Cloud Edition for Lustre wiki for additional information. Intel provides different options for launching Lustre using Cloud Formation. You should use the templates at ICEL - Global Support HVM. The templates are region specific since the AMI identifiers are different by region. You will need to accept terms on Intel Cloud Edition for Lustre* Software - Global Support (HVM) AWS Marketplace page once before you run the template, otherwise the creation will fail. For a similar configuration to that used in this testing, make the following edits:

1) The clients launched are unnecessary. The SAS client instances are used instead. Under resources, remove `ClientInstanceProfile`, `ClientLaunchConfig`, `ClientNodes`, `ClientPolicy`, and `ClientRole`. Under `Resources->BasePolicy->Properties->Roles` remove

```
,
{
  "Ref": "ClientRole"
}
```

2) Add a parameter for a placement group at the end of the `Parameters` section:

```
"PlacementGroup" : {
"Description": "Name of an existing placement group",
        "Type": "String",
        "MinLength": "1",
        "MaxLength": "32"
}
```

3) For `MDSNodes`, `MGSNodes`, and `OSSNodes` under resources add the following to the Properties section:

```
"PlacementGroup": {"Ref": "PlacementGroup"},
```

4) Make the following changes to `OSSLaunchConfig`

```
Update Properties->InstanceType to c3.8xlarge
Remove Properties->EbsOptmized (does not apply to c3.8xlarge)
```

5) When you launch the CloudFormation template, specify 3 OSS nodes with 750 GB volumes (the template will launch (8) 750 GB volumes per node, which is an equivalent capacity used in this testing)

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