PHY4902: Undergraduate Research

Research Report

Samantha Wohlstadter

Table of Contents

List of Figures

Overview and Goals

The main goal of this project is to set up CMS software locally on the Cluster in order to simulate Mehdi's event samples. To accomplish this goal, the work was divided into multiple steps:

- Ensure that the Cluster is functional, including the compute nodes, the Storage Element (SE), the Compute Element (CE), the NAS units, and the UPS units.
- Reconfigure MySQL after installing the nodes, so that the website is accessible.
- Set up Condor, the job submission software, to automatically run data between the different steps of the simulation process.
- Install and set up the CMS simulation software, including Root.

Before this project began, the cluster was still in the first step. The new CE had just been installed and set up, including: the ROCKS OS installation, the zpool for the OS drives, and most of the node integration. The remaining tasks to complete this step included verifying the node integration, though there was also ongoing general maintenance of the cluster such as hard drive and battery replacements.

Figure 1: The Cluster

CE and Nodes

As part of ensuring the functionality of the Cluster, the new CE had to be set up as well as the compute nodes. It was planned to move the sfp+ port from the old CE to the new CE, though unfortunately the computer used as the new CE was too small for the sfp+ port to fit, so this task has been postponed.

At the beginning of this project, the setup of the new CE was almost complete. ROCKS, the CentOS-based Operating System that was to be utilized, had been installed, and a ZFS mirrored 'zpool' array was created to mirror the OS drive. This was done in order to provide redundancy in case of drive failure. However, there were still some issues with the zpool. Upon attempting to reboot the CE after shutting it down for a tropical storm, it could not generate the initramfs images properly and entered dracut emergency mode. This was presumably due to an incompatibility with the zpool created on both drives that hosted the OS. There was no native functionality to ZFS to create a 'root pool', and it appeared that the workaround (using a third drive to create the zpool) was not completely successful. To resolve this issue, the initramfs images needed to be regenerated using the following command:

```
dracut-cmdline --regenerate-all -f
```

After the execution of this command, the initramfs issue was resolved, and the CE could boot properly. It was discovered that the above command needed to be run every time the CE was turned on, and no permanent solution has yet been found. No other issues were discovered with the CE, so the compute nodes were worked on next.

In the beginning of this project, most of the compute nodes had already been installed and integrated into the Cluster using insert-ethers. The first step to verify the node integration was to check that they were present in ROCKs' list of known hosts, and the second step was to test that the nodes could be reached by the CE. To check the list of known hosts, the following command was run:

```
rocks list host
```

All nodes except for compute-1-3 were present, and were verified to be reachable by the CE using ssh.

NAS Maintenance

As another part of ensuring Cluster functionality, the NAS units had to be maintained throughout the project. This mainly involved replacing hard drives as they failed and checking that the drive arrays were healthy. It also included restarting NAS-1 if its RAID card suddenly failed to be detected, which would resolve that error. However, an issue arose when attempting to replace hard drives in NAS-0: the drive names and order in the ZFS zpool did not match the physical disk order.

During the project, a total of three hard drives in NAS-0 failed. NAS-0's zpool is set up into a 'RAID 6+0' array, meaning that there is a mirrored group of six drives, with each group having a redundancy of two. One of the drives failures was a hot spare (of which there are two), and the other two drive failures happened in different groups. One of those failed drives was successfully replaced by the functional hot spare. Thus, there was no risk to the data stored in the zpool; the drives just needed to be replaced.

```
[root@nas-0-0 ~]# zpool status
  pool: nas0
 state: ONLINE
  scan: resilvered 1.12T in 48h30m with 0 errors on Wed Sep 30 15:16:07 2020
config:

        NAME                                    STATE     READ WRITE CKSUM
        nas0                                    ONLINE       0     0     0
          raidz2-0                              ONLINE       0     0     0
            scsi-1AMCC_5QD42KX0000000000000     ONLINE       0     0     0
            scsi-1AMCC_W0331585000000000000     ONLINE       0     0     0
            sdo                                 ONLINE       0     0     0
            scsi-1AMCC_9QK2C2T8000000000000     ONLINE       0     0     0
            scsi-1AMCC_W0146976000000000000     ONLINE       0     0     0
            scsi-1AMCC_9QK291KV000000000000     ONLINE       0     0     0
          raidz2-1                              ONLINE       0     0     0
            scsi-1AMCC_5QD42KX6000000000000     ONLINE       0     0     0
            sdl                                 ONLINE       0     0     0
            scsi-1AMCC_9QK2AXYP000000000000     ONLINE       0     0     0
            scsi-1AMCC_W0135708000000000000     ONLINE       0     0     0
            scsi-1AMCC_3QD1GLJK000000000000     ONLINE       0     0     0
            sdn                                 ONLINE       0     0     0
        spares
          sdm                                   AVAIL

errors: No known data errors
```

Figure 2: An example of the output of 'zpool status'

Once the NAS-0 zpool was reimported after rebooting, it was discovered that the disk order and naming did not correspond to the order of the physical disks. Only a couple were out of order, but the specific disks that were out of order could not be readily determined, so none of the names in the zpool array could confidently be associated with a physical hard drive. The first step in atempting to resolve this issue was to check the output of the RAID card tool, tw_cli. Normally, the output looks similar to the following:

```
[root@nas-0-0 ~]# tw_cli /c0 show

Unit  UnitType  Status         %RCmpl  %V/I/M  Stripe  Size(GB)  Cache  AVrfy
------------------------------------------------------------------------------
u0    RAID-1    DEGRADED       -       -       -       698.481   Ri     ON
u1    JBOD      OK             -       -       -       931.513   Ri     OFF
u2    JBOD      OK             -       -       -       698.638   Ri     OFF
u3    JBOD      OK             -       -       -       698.638   Ri     OFF
u4    JBOD      OK             -       -       -       698.638   Ri     OFF
u5    JBOD      OK             -       -       -       698.638   Ri     OFF
u6    JBOD      OK             -       -       -       698.638   Ri     OFF
u7    JBOD      OK             -       -       -       698.638   Ri     OFF
u8    JBOD      OK             -       -       -       698.638   Ri     OFF
u9    JBOD      OK             -       -       -       698.638   Ri     OFF
u10   JBOD      OK             -       -       -       698.638   Ri     OFF
u11   JBOD      OK             -       -       -       698.638   Ri     OFF
u12   JBOD      OK             -       -       -       698.638   Ri     OFF
u13   JBOD      OK             -       -       -       698.638   Ri     OFF
u14   JBOD      OK             -       -       -       698.638   Ri     OFF

VPort Status         Unit Size      Type  Phy Encl-Slot    Model
------------------------------------------------------------------------------
p0    OK             u0   698.63 GB SATA  0   -            ST3750640NS
p2    OK             u1   931.51 GB SATA  2   -            ST1000NM0033-9ZM173
p3    OK             u2   698.63 GB SATA  3   -            ST3750640NS
p4    OK             u7   698.63 GB SATA  4   -            WL750GSA6472
p5    OK             u3   698.63 GB SATA  5   -            WDC WD7502ABYS-18W8
p6    OK             u4   698.63 GB SATA  6   -            GB0750EAMYB
p7    OK             u5   698.63 GB SATA  7   -            ST3750330NS
p8    OK             u6   698.63 GB SATA  8   -            ST3750330NS
p9    OK             u13  698.63 GB SATA  9   -            WL750GSA6472
p10   OK             u8   698.63 GB SATA  10  -            ST3750330NS
p11   OK             u9   698.63 GB SATA  11  -            ST3750330NS
p12   OK             u10  698.63 GB SATA  12  -            WDC WD7502ABYS-02A6
p13   OK             u14  698.63 GB SATA  13  -            WL750GSA6472
p14   SMART-FAILURE  u11  698.63 GB SATA  14  -            ST3750640NS
p15   OK             u12  698.63 GB SATA  15  -            ST3750330NS

Name  OnlineState  BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   On           Yes       OK        OK       OK       0      xx-xxx-xxxx
```

Figure 3: An example of the output of tw_cli, showing the hard drives that the RAID card can detect.

However, in this case, there were missing units and VPorts. Those missing drives could reasonably be assumed to be the failed drives, however here too the order of the units and the Vports did not match. (For example, unit 13 is associated with VPort 9 in the figure above). After reading through the documentation for ZFS and tw_cli, as well as research into similar issues, it was determined that the best way to correlate the drives in the zpool to the physical drives would be to sequentially turn each individual drive's activity light on, and note the physical location. This was accomplished by reading from each drive using the "dd" command as follows:

```
dd if=/dev/disk/by-id/scsi-1AMCC_9QK2C2T8000000000000 of=/dev/null
```

This copies the data from the specified input 'if' to the specified output 'of', which in this cas eis null. If the drive is not faulted, the activity light will activate. In this manner, it was discovered that the physical drives that had failed were in slots 0:4, 0:9, and 0:13. These drives could then be replaced. The process to replace a hard drive is as follows:

```
zpool offline nas0 [drive name]
tw_cli maint remove c0 p[x]
```

(physically remove the hard drive and replace with new drive)

```
tw_cli /c0 rescan
zpool replace nas0 [drive name]
```

In this way, the two hard drives in the zpool groups that had failed were successfully replaced. The process for replacing a faulted hot spare, however, was a bit more tricky. ZFS does not have native functionality to replace a drive that is marked as a spare; instead it must be removed from the zpool entirely, and once the drive is replaced, it can be manually added back into the zpool. Currently, the third drive is failing to appear in the kernel message output dmesg, so its name cannot be found to add it back to the zpool. once this name is found, it can be added back to the zpool as a hot spare using the following command:

```
zpool add nas0 spare [drive name]
```

Before ordering new drives, the old drives were checked to see if they could be repaired and reused. CrystalDriveChecker and Seatools, two (Windows exclusive) programs to analyze and repair hard drives, were installed and used to check the health of the three faulted drives. Both programs were used to analyze each drive, and Seatools was used to attempt to repair the drives. One of the hrad drives would not even spin up when it was connected, and the other two failed their SMART tests. This means that they were not repairable by writing '0"s to them and resilvering them. (One of these two had audible scratching sounds while it was in use by the software.) As none of the drives could be repaired, three new hard drives were ordered to replace the failed drives in the zpool.

UPS Maintenance

As another part of ensuring Cluster functionality, the USP units had to be maintained throughout the project. This, for the most part, included balancing the UPS loads and replacing batteries with low voltages. A side task was to set up the Tripplite monitoring software, and installation CDs were found for the Tripplite UPS units, though this step is still in progress.

In the beginning of the project, one of the UPS units would not turn on after having been shut down due to a tropical storm over the summer. As the other UPS units were turned on, the compute nodes and other cluster components also turned on, indicating that they were not shut down before or after the UPS units were unplugged and shut down. This may have been the cause of the UPS unit that would not turn on, though it may also have been the age of the batteries.

The first step to diagnose the issue with the UPS was to check the breakers. They were untripped, so these were not the issue. Next, the battery voltages were checked. It was discovered that the voltages on every single battery inside that unit were well below the acceptable level. The batteries are supposed to be 12V, but the batteries in the case averaged 3V. New batteries were ordered, and the other two UPS units were checked as well, though their batteries were at the proper voltages. Replacements were ordered, however a couple of bad F2 connectors inside the unit were found. They would not connect properly to the terminals of the batteries, and they needed to be replaced. This task remains to be completed, so currently half of the nodes of the Cluster are unpowered and offline.

Software

once the cluster is functional, the next step is to set up the required software to run the simulations. This includes Condor, Root, and the CMS software. The website also required setting up, although this was not strictly necessary to run the simulation software.

MySQL and Website

http://uscms1.fltech-grid3.fit.edu/diagnostics/

After installing ROCKS onto the CE, MySQL needs to be configured to set up the website and allow it to access the wiki database. First, iptables must be modified to allow incoming connections to port 80, so that the website can be viewed. This is done with the following command:

```
iptables -I INPUT 5 -p tcp -m state --state NEW,ESTABLISHED
         -m tcp --dport 80 -j ACCEPT
service iptables save
```

After this, the website backup was copied from where it was stored on nas1 to the appropriate directory on the CE (/var/www/html). At this point MySQL must be configured, otherwise the follwing error will be visible at the website address:

```
Error: cannot contact database server
```

If the website does not show error messages, they can be enabled by I enabled by changing the value in `/etc/php/ini` and in `LocalSettings.php` to true, though this should be set to false once the issue is resolved. First, the root password for MySQL was reset using the following procedure:

```
/opt/rocks/mysql/support-files/mysql.server stop
mkdir -p /var/run/mysqld/
mkdir /var/run/mariadb
touch /var/run/mysqld.pid
useradd mysql
chown mysql:mysql -R /var/run/mysqld/
chmod -R 777 /var/run/mysqld
```

```
/opt/rocks/mysql/bin/mysqld_safe --basedir=/opt/rocks/mysql
    --datadir=/var/opt/rocks/mysql
    --log-error=/var/opt/rocks/mysql/uscms1.fltech-grid3.fit.edu.err
    --skip-grant-tables
```

This starts MySQL so that it will not check the password for users. In another window, the following was run to log into MySQL as root and change the password ('password' was not the actual password):

```
mysql --socket=/var/opt/rocks/mysql/mysql.sock -u root
UPDATE mysql.user SET Password=PASSWORD('password') WHERE User='root';
```

After this is done, the configuration file needs to be edited to set the correct MySQL paths. Namely, the correct paths for the socket file and the database files need to be set in the file /etc/my.cnf. The following may be used instead, but is not recommended because it is long and tedious to type.

```
mysql --socket=/var/opt/rocks/mysql/mysql.sock -u (user) -p
```

The correct paths are specified in the file /opt/rocks/mysql/my.cnf, but they need to be set in /etc/my.cnf to match. They need to be set under both the [mysqld] and the [client] headings! The values should be:

- socket=/var/opt/rocks/mysql/mysql.sock
- datadir=/var/opt/rocks/mysql/

After these changes are made, the mysql server and apachectl were restarted. Finally, in `/ect/php.in`, the value of `$wgDBserver` was changed from `localhost` to `localhost:/var/opt/rocks/mysql/mysql.sock`. Then, the old databases could be reimported to the new CE's mysql server using `mysql -u root -p (database) < (database).sql`. Permissions were granted by running `GRANT ALL PRIVILEGES ON (database)* TO (user)@localhost` from within MySQL. Privileges to the database '`wikidatabase`' were granted to the user 'wiki' (which needed to be created). After this, the website was functional and could be accessed in a browser via its URL.

Condor

Condor is the job submission software to be used to automate passing the data between different steps in the simulation chain. There are six steps in the simulation chain:

1. Madgraph MC generator
2. Gen-Sim
3. Digitization step 1
4. Digitization step 2
5. Mini AOD
6. Ntuples

For each step in the chain, input files need to be specified and output files are generated. Condor will run jobs to automatically submit the required files to each step. At the beginning of this project, Condor had already been installed on the CE, yet it had not been tested. To verify that it was functioning correctly, a test user 'sam' was created, and a test job was created after reviewing documentation on the format of the job and the submit files. The test job and associated submit file are included below:

Test Job:

```
#!/bin/bash

TIMETOWAIT="6"

echo "sleeping for $TIMETOWAIT seconds"
/bin/sleep $TIMETOWAIT
```

Submit File:

```
# submit description file for HTCondor

executable              = testjob.sh
log                     = testjob.log
output                  = outfile.txt
error                   = errors.txt
should_transfer_files   = yes
when_to_transfer_output = ON_EXIT
queue
```

Then, the job was submitted using `condor_submit`, and its status could be monitored using `condor_status`. The job ran successfully, verifying the Condor installation and indicating that the next step in setting up the simulation software could begin.

Root

installed latest version after consulting mehdi, needed cmake cmake - yum only had version 2.8, need 3.9 of cmake - yum search –showduplicates cmake only has 2.8. downloaded 3.9 from cmake website repo and followed README to build. root then had some dependency errors, but these were quickly resolved by yum installing the required software. (root/README.md)

CMS Software

Before installing the software, the documentation located at https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBook was reviewed, as well as https://uscms.org/uscms_at_work/physics/computing/setup. The main sections that are of importance are 1.3 Computing Environment, Installing CMSSW using cmspkg, and 6.1 Thirty-Minute Introduction to Generation and Simulation. The FIU Paper on deploying a CMS Tier 3 Computing Cluster was also utilized.

After the documentation was reviewed, the isntructions in the documentation section “Installing CMSSW using cmspkg “ were followed to install

CMSSW. First, the user 'sam' was given sudo privileges, as the software cannot be downloaded as the root user. The following commands were run:

```
export VO_CMS_SW_DIR=/export/home/sam/CMS/
mkdir -p $VO_CMS_SW_DIR
wget -O $VO_CMS_SW_DIR/bootstrap.sh http://cmsrep.cern.ch/cmssw
     /repos/bootstrap.sh
export SCRAM_ARCH=slc6_amd64_gcc530
sh -x $VO_CMS_SW_DIR/bootstrap.sh setup -path $VO_CMS_SW_DIR
   -arch $SCRAM_ARCH >& $VO_CMS_SW_DIR/bootstrap_$SCRAM_ARCH.log
```

This revealed some dependency errors. The following additional software needed to be installed: libglvnd-opengl, mesa-libGLU-devel, perl-ExtUtils-Embed, perl-Switch, and zsh. All were able to be installed using `yum install [package]` except for libglvnd-opengl, which was not found by yum. After some research into this missing package, the official CentOS repository for this package was found, and libglvnd-opengl attempted to install using `wget` and `yum localinstall`. This, however, also had dependencies that were not able to be found by the yum installer, namely libGLdispatch.so.0()(64bit) and libglvnd(x86-64). Fortunately, the official CentOS repository also had these packages, and libglvnd-opengl could finally be installed. With the dependencies resolved, the above commands to install CMSSW could be finished successfully.

The next step to install CMSSW was to install the release. The following commands were listed in the next step:

```
$VO_CMS_SW_DIR/common/cmspkg -a $SCRAM_ARCH update
$VO_CMS_SW_DIR/common/cmspkg -a $SCRAM_ARCH install
    cms+cmssw+CMSSW_10_2_5
```

However, the last command failed to complete and gave an "unknown package" error. The guide listed CMSSW_8 as the example, so in the off chance that the command syntax had changed slightly, slight variations were also attempted, such as `cms cmssw CMSSW_10_2_5`, though these resulted in the same error messages. After reviewing the documentation again to check that the previous steps were executed correctly, searching for similar issues in the documentation's FAQ, and searching the Cern Twiki site, the official

CMS github was found, and `CMSSW_10_2_5_patch1` was downloaded from that repositiory.

The next step was to set up the computing environment as in the documentation section 1.3 (Your Computing Environment). First, the work area needed to be built:

```
source $VO_CMS_SW_DIR/cmsset_default.csh
```

Then, the CMSSW release needed to be verified:

```
scram list -a | grep CMSSW_10_2_5
```

However, this command resulted in an error: scramv1 not installed. At this point, the current installation of CMS software was backed up to a different folder so that the process could be restarted from scratch. While reviewing the documentation and comparing the twiki documentation to the https://uscms.org docuentation, it was discovered that the Scram arch given in the twiki was out of date. So, the installation proceeded with the following change:

```
export SCRAM_ARCH=slc7_amd64_gcc820
```

The installation continued as it had the first time, though without the dependency errors. However, `$VO_CMS_SW_DIR/common/cmspkg -a $SCRAM_ARCH install cms+cmssw+CMSSW_10_2_5` still failed. The CMSSW_10_2_5_patch1 package from the CMS repository was used instead (as above), and installed successfully. However, the "scramv1 not installed" error persisted. Thus, any further commands such as `cmsrel CMSSW_10_2_5` failed.

The Cern Twiki Documentaion has a couple of pages dedicated to working with SCRAM, though all start witht the assumption that it is already installed and configured. Additionally, the FAQ does not contain any similar issues. Even on other sites, no similar errors could be found. After searching through the CMS github repository, SCRAM was found and downloaded. It appeared to successfully install, though the "scramv1 not installed" error perstisted still when attempting to verify that the CMSSW release had installed correctly.

It was discovered that scram commands were functional within the SCRAM directory, but not outside of it. Then, by running `./export/home/sam/SCRAM/SCRAM-master/bin/scram.in list -a` instead of `scram list -a`, the output would appear outside of the SCRAM directory. It also appeared that the scram installation did not have full functionality, as some commands did not work as expected. This might be due to an incorrect configuration setting for the SCRAM installation, or an incorrect installation attempt, though the documentation for setting up SCRAM is sparse. The new error that appears when trying to list the current scram projects is: `There are no SCRAM project yet installed.`. This could indicate that CMSSW has also not been installed correctly, or that SCRAM is failing to detect a successfull installation because it is incorrectly installed or configured.

The next step to resolve these issues is to try and get in touch with somebody who may know more about SCRAM and installing CMSSW.