1 Setup

We've elected to create the brand new system on one of Stefano's new computers. That way, the current system is left untouched, and we are able to play around freely. The current system is running Scientific Linux 5, so we are installing the latest Scientific Linux, SL7, on the new testing computer.

SL7 has been cleanly installed, now to enable ssh. Turns out ssh is installed and turned on by default in SL7, but it is unreachable via the WiFi; I have to first ssh into NAS-1, then into the new computer.

1.1 ROOT Installation

One of the most important pieces of software we're going to need is ROOT. I installed the latest CentOS7 binary from root.cern.ch, and unpacked it. The CentOS7 binary may be used on SL7 because they are both closely related.

1.2 Installing Stefano's Image

Stefano created a disk image for us before he left with DATE and SCRIBE preinstalled and ready to go! It's at `/nas1/cmsgem/software/srs_centos7_2.vdi`. Unfortunately, it's a `vdi` file rather than an `iso`, so some conversions need to be done. I'm performing everything on the MTS development machine.

First we need to install some packages: `qemu` and `tklpatch`. Since `vdi` files are VirtualBox images, we need to download VirtualBox to get the conversion tools:

- `cd /etc/yum.repos.d`
- `wget http://download.virtualbox.org/virtualbox/rpm/rhel/virtualbox.repo`
- `yum --enablerepo=epel install dkms`
- `yum install VirtualBox-5.1`

Now that VirtualBox is installed, we can start converting! First, the `vdi` needs to be converted into a `vmdk`, `VBoxManage clonemedium srs_centos7_2.vdi srs_centos7_2.vmdk --format VMDK`. Then, to a raw file, `qemu-img convert -f vmdk srs_centos7_2.vmdk -O raw srs_centos7_2.raw`. Next, the raw

disk must be mounted as a loop device, but, since the image has multiple partitions, some extra steps must be performed.

Since the image has partitions, it must be specially prepared for mounting, `loopdev=$(sudo losetup --show -f srs_centos7_2.raw)`. The the command was assigned to a variable, because its output will be used more than once. The tool `kpartx` needs to be installed, `sudo yum install kpartx`, and used to add mappings to the image's partitions, `sudo kpartx -a $loopdev`. Now, the image is ready to be mounted. Create a mount point, `mkdir srs_centos7_2.mount`, and mount the image `sudo mount /dev/mapper/$(basename $loopdev)p1 srs_centos7_2.mount`. The image should be visible from `df -h`, and accessible via the route specified by `df`.

Now a copy may be made of the root file system of the image. Create a directory for the file system, `mkdir srs_centos7_2.rootfs`, then copy the contents of the mounted file system to the new directory, `sudo rsync -a -t -r -S -I srs_centos7_2.mount/ srs_centos7_2.rootfs`. Now that the mount point has been copied over, the original may be unmounted.

Scratch that, we're just gonna put DATE on it. Stefano linked me to the CERN forum thread where he asked Brian Dorney for help to do exactly what we're trying to do.

The first thing we need to do is follow the setup instructions from `http://linux.web.cern.ch/linux/centos7/docs/install.shtml`. The first thing that must be done is the configuration of the AFS client. Unfortunately, the necessary command, `locmap` is not installed on our system, nor is it available from any of the installed repositories. I found it and its dependencies from the CERN's Linux website where all of its packages are hosted, `http://linuxsoft.cern.ch/cern/centos/7/cern/x86_64/Packages/`.

Once those RPMs are downloaded, they may be directly installed with `rpm`, but that command will not automatically install dependencies. To ease the installation process, a yum repository containing the RPMs can be created. A new file, `/etc/yum.repos.d/localCERN.repo` was created, and in it were placed the lines:

```
[localCERN] # the name of the new repository
baseurl = file://<full path to directory containing RPMs>
enabled=1 # enable the repo be default
gpgcheck=0 # disable package verification
```

While disabling package verification is generally bad practice, we do not have the keys necessary to properly install them. We also know where we got them from, and they are locally installed, so, in this particular case, it should not

be a big deal.

Now that the repo has been created, `locmap` can be installed, `sudo yum install locmap`. When the required command, `sudo locmap --configure afs`, is run, it complains that it cannot connect to `xldap.cern.ch` and that the afs module is disabled. I've enabled the module with `sudo locmap --enable afs` and tried again to see what would happen, and, while it failed once again to connect to `xldap.cern.ch`, it says that it configured everything!

The next step is to preconfigure the AFS client for CERN by configuring NTP, which must be enabled, `sudo locmap --enable ntp`. Now, it can be configured with `sudo locmap --configure ntp`. Next, the following modules must be enabled using the same command as above: cvmfs, eosclient, kerberos, lpadmin, nscd, sendmail, ssh, and sudo. A list of every available module and their statuses can be viewed with, `sudo locmap --list`. Now, they must all be configured with, `sudo locmap --configure all`.

Well, turns out the AFS stuff doesn't work, probably because we couldn't connect to the server. Fortunately, we only needed AFS to get the CERN repositories, `yum-alice-daq.cc7_64.repo` and `yum-alice-daq.slc6_64.repo`. I had grabbed `yum-alice-daq.cc7_64.repo` from another computer, but the SLC6 repo was nowhere to be found. This problem was solved by creating the repo file myself by copying the CC7 file, and changing the URLs to point to where the SLC6 RPMs ought to be located. The contents of `yum-alice-daq.cc7_64.repo` is:

```
[main]
[alice-daq-cc7]
name=ALICE DAQ software and dependencies - CC7/64/
baseurl=http://cern.ch/alice-daq/yum-alice-daq.cc7_64/
baseurl=https://yum:daqsoftrpm@alice-daq-yum.web.cern.ch/alice-daq-yum/cc7_64/
enabled=1
protect=1
gpgcheck=0
```

The contents of the newly created `yum-alice-daq.slc6_64.repo` is:

```
[main]
[alice-daq-slc6]
name=ALICE DAQ software and dependencies - slc6/64/
baseurl=http://cern.ch/alice-daq/yum-alice-daq.slc6_64/
baseurl=https://yum:daqsoftrpm@alice-daq-yum.web.cern.ch/alice-daq-yum/slc6_64/
enabled=1
```

```
protect=1
gpgcheck=0
```

Running a quick `sudo yum update` is helpful here, just double check to make sure you're not accidentally installing any SLC6 packages.

Now to install a bunch of things from these repositories! The installation instructions from here, `https://alice-daq.web.cern.ch/products/date-installation-and-configuration` were followed.

The first command `rpm -e mysql-libs mysql mysql-devel postfix` produced some errors, but they are excusable. They appeared because those packages simply are not installed on our machine; the `postfix` one, however, is. While simply `rpm -e postfix` could have been executed with equal affect, it is safe to include the other three packages in case they happened to have been mysteriously installed onto the system.

The next command is `rm -rf /var/lib/mysql/`, but that directory should not exist in the first place, since MySQL is not installed. Now, we must install a bunch of packages (including DATE!), `sudo yum install BWidget MySQL-shared MySQL-client MySQL-devel dim smi tcl-devel tk-devel curl-devel libxml2-devel pciutils-devel mysqltcl xinetd ksh tcsh date`. With that completed, MySQL server must be installed, `yum install MySQL-server`. This time, a conflict with `mariadb-libs` was encountered, but we can just erase this package, since we don't need it, `yum erase mariadb-libs`.

The next step is to setup MySQL. First the service must be started, `service mysql start`. Since the root account must be setup first, and the temporary root password is stored in root's home directory, we must switch to root to complete setting up the initial account, `sudo su`. The temporary password is viewed with `cat /root/.mysql_secret`. Copy that password, then paste it when prompted after running, as root, `mysqladmin -u root -h localhost -p password`.

Now that the root account has been created, the user accounts can be added. You may log out of root. First the `DATE_SITE` environment variable must be cleared, `unset DATE_SITE`, then the date setup script must be executed, `. /date/setup.sh`. Afterwards, the MySQL initialization script must be executed to configure the new database, `newMysql.sh`. While the settings may be changed to whatever best fits the situation, the defaults are sufficient.

Now that the MySQL database is initialized, it must be configured, for DATE, `newDateSite.sh`. NOTE: To create the required `/dateSite` direc-

tory, it will ask for the root password. When prompted, confirm that a minimal configuration is to be created; this is not the default setting. Also confirm that all detectors are to be added and that all trigger class masks ought to be added; these are not the default settings.

With the base DATE configuration complete, the local configuration can begin. These next few steps must be completed as root (`sudo su`). The path of the DATE_SITE must be set, `export DATE_SITE=/dateSite`, the setup script must be executed, `. /date/setup.sh`, and the local configuration command must be run, `dateLocalConfig -s`.

Running `dateLocalConfig -s` will result in an error because, in CentOS7, `iptables` has fallen away in favor of `firewalld`. These instructions form StackOverflow, `https://stackoverflow.com/questions/24756240/how-can-i-use-iptables-on-centos-7` describe how to switch back from `firewalld` to `iptables` so that DATE, built for SLC6, will be happy.

First, `firewalld` needs to be stopped,`systemctl stop firewalld`, and hidden away, `systemctl mask firewalld`. Next, `iptables` must be installed, `yum install iptables-services`, and enabled, `systemctl enable iptables` followed by `systemctl start iptables`. Running `dateLocalConfig -s` will, again, produce an error, but running it again immediately will work, somehow.

Next, the DIM (Distributed Information Management) name server must be launched (and must be launched after each boot; this will be automated), `/date/runControl/start_dim.sh &`. This returned a "no process found" error, but running it again seemed to work fine. This error comes about because of slightly different paths to `do_start_dim.sh`, whose location can be found with `locate do_start_dim.sh`.

The infoLoggerServer must also be started (these commands must be run after each boot; this will be automated):

```
export DATE_SITE=/dateSite
.  /date/setup.sh
/date/infoLogger/infoLoggerServer.sh start
```

The DAQ ought to be ready to be launched:

```
export DATE_SITE=/dateSite
cd /date
.  ./setup.sh
infoBrowser
runControl/DAQCONTROL.sh
```

Now, it's time to install AMORE as root (`sudo su`), `yum install amore`. While all of the AMORE tools are installed, they are not part of the `PATH` environment variable, so they cannot be run by basename on the normal prompt. Their install location can be found with `locate amoreMysqlSetup`, which will display the full path to one of the tools. The remainder of the tools are in the same directory. In our case, that directory is `/opt/amore/bin/`, so that is what we will add to the `PATH` variable, `PATH=$PATH:/opt/amore/bin`. Be sure to add `export PATH=$PATH:/opt/amore/bin` to both the user's and root's `/.bash_profile`. Now all the AMORE tools are accessible by their basenames.

The next step is to set up the MySQL database for AMORE, `amoreMysqlSetup`. The defaults are sufficient. Now the AMORE site itself needs to be created, `newAmoreSite`, and the defaults are fine. The problem with AMORE now is that the `amore` command, itself, does not work.

The next step is to install the hardware drivers for the MTS hardware. Unfortunately, that is proving difficult. The source for `date-114` does not contain many of the required directories. `date-100`, however, does, so we've uninstalled the latest version of DATE, and installed the old version instead. We've followed all the instructions for configuring DATE again, including rebuilding the MySQL database.

We've come across our first issue: `dateLocalConfig -s` is throwing some `xinetd` errors.

Another issue is that AMORE doesn't actually work. All of the commands exist, but trying to run `amore` results in a error complaining about being unable to load the shared library `libCore.so.5.34`. Running a `locate libCore.so` reveals that it's a ROOT library, but that only three versions of the file exists: `/usr/lib64/root/libCore.so`, `/usr/lib64/root/libCore.so.6.14`, and `/usr/lib64/root/libCore.so.6.14.04`. Huh, it doesn't look like we have the required file. The `LD_LIBRARY_PATH` environment variable, however, does contain the appropriate paths of `/opt/amore/lib` and `/usr/lib64/root`. It looks like AMORE depends on ROOT 5 rather than ROOT 6, so we've got to downgrade ROOT.

To downgrade ROOT, the current version must be first uninstalled, `sudo yum remove root`. `yum` will verify that ROOT 6 is to be removed, and it will warn that AMORE will no longer work, since it depends on ROOT. The available versions of ROOT can be checked with, `yum --showduplicates list root`. ROOT 6 is only in the EPEL repository, but ROOT 5 is in the

alice-daq-slc6 repository, so we're going to tell yum to install ROOT only using the SLC6 repo, yum --disablerepo=* --enablerepo=alice-daq-slc6 install root. yum should report that it will install ROOT 5.

Since AMORE was removed when we removed ROOT, it must be re-installed, yum install amore. Now, when amore is run, we get the same error, but with a different file, libAmoreUI.so, that has only one version. I checked the LD_LIBRARY_PATH variable, and it was empty! I put in the path to the AMORE library, export LD_LIBRARY_PATH='/opt/amore/lib'. Now we just need to set up AMORE as before.

The first step is to re-add the AMORE binary file to the PATH environment variable by adding the following to the /.bashrc, export PATH=$PATH:/opt/amore/bin. Now, amoreMysqlSetup can be run.

ASIDE: It may complain that an AMORE database already exists. That's fine, it can be removed by using MySQL. The MySQL prompt can be accessed with mysql -u daq -p, where the username and password are both daq by default. Once at the prompt, the available databases can be listed with show databases;, the old AMORE database can be removed with drop database AMORE;, and the prompt can be exited with quit.

The issue now is that amoreUpdateDB.tcl is in the wrong place. It complains that /opt/amore/bin/amoreMysqlSetup could not find /bin/amoreUpdateDB.tcl, which is fine because it's in /opt/amore/bin/amoreMysqlSetup. In /opt/amore/bin/amoreMysqlS the path to amoreUpdateDB.tcl is described as $AMORE/bin/amoreUpdateDB.tcl. Since $AMORE is blank, I've run export AMORE=/opt/amore. After dropping the old AMORE database from MySQL, amoreMysqlSetup will run successfully.

When we downgraded ROOT, we also had to reinstall all of its dependencies and rebuild it. The rebuild process was giving us some issues because it cannot be built in the installation directory; it needs to be first placed in the builddir directory, then built. Now that ROOT is installed, AMORE may be set up again as normal with amoreMysqlSetup and newAmoreSite (NOTE: if not running newAmoreSite as root, it must be run with sudo env ''PATH=$PATH'' newAmoreSite in order for the environment to be properly preserved (sudo -E newAmoreSite was ineffective)).

When amore is run, however, after several ROOT warnings about a bunch of classes already existing in TClassTable, it says terminate called after throwing an instance of 'std::runtime_error', then what(), the AMORE function that is supposed to print out what caused an error, shows a "Usage" dialogue for some command; it just prints out "Usage:" followed by

some flags. Evidently, the issue is that some command is not being executed as expected, or something isn't installed/configured properly.

Let's first try to find which command this "Usage" belongs to. I navigated to `/opt/amore` and grepped for a keyword in one of the flag definitions, `grep -R ''<LIBSUFFIX>'' *`, and some binary files were matched, `lib/libAmoreCore.so` and `lib/libAmoreDA.a`. To tell grep to treat the binary files as text files, the `-a` flay may be used, `grep -aR ''<LIBSUFFIX>'' *`. Digging into those binaries didn't help a whole lot; I just found the "Usage" text on its own without any context.

Some progress has been made! Joseph has reinstalled a slightly newer version of ROOT, v5.34/38, and the strange ROOT errors from before are all gone! ROOT starts normally! Now let's see how AMORE'll play with the new ROOT.

When redoing the `amoreMysqlSetup` stuff, we ran into some issues. `newAmoreSite` returns a syntax error when run, `/opt/amore/bin/newAmoreSite: line 57: syntax error: unexpected end of file`. I'm going to reinstall AMORE.

After reinstalling ROOT, we tried to run AMORE, and it failed because it couldn't find `libCore.so.5.34`. The file that contains the path to the ROOT libraries is `/etc/ld.so.conf.d/root-x86_64.conf`; it contained `/opt/root/lib`. Unfortunately, no libraries are there. We created another file in that same directory, `/etc/ld/so/conf.d/root.conf`, and put the new path to the libraries (the old library paths were messed up after reinstall): `/opt/root/builddir/lib`. To save this change, run `ldconfig`.

Turns out we still had to build ROOT! We navigated to the ROOT build directory and followed the README for building. ROOT's all good to go, now let's see about AMORE.

I'm reinstalling it. To uninstall AMORE, `yum remove amore.x86_64`, and to install it again, `yum install amore`. Hmm, when it tried to get the package from the SLC6 repository, it encountered an HTTP 403 response. `yum` also gave us an error code of 14, and an article providing some troubleshooting steps. The article suggests clearing the cache with `yum clean all` followed by `rm -rf /var/cache/yum*`.

Now `yum` just straight up fails; it had been relying on that cached data to go through in the first place. It's complaining about repository configuration, so let's see what's going on with that.

AMORE has been decommissioned by ALICE, but Stefano sent us a copy of it's source on github (`https://github.com/stefanocolafranceschi/`

amoreSRS). I cloned the repo, so now we've got it. Time to build it on the MTS dev machine!

There's a Makefile in the repository's root directory, so I ran make to use it. It's complaining that there's no /usr/local/etc/Makefile.arch, so let's investigate that. When the make file in the root directory of the repo is run, it immediately travels to the src directory and runs the make file there. The first line in src/Makefile includes Makefile.inc in the same directory, whose first action is to include $(shell root-config --prefix)/etc/Makefile.arch. root-config --prefix outputs /usr/local, which is where it's getting /usr/local/etc/Makefile.arch from. The entire /usr/local/etc directory, however, is empty! It's also empty on the old MTS machine, so that's interesting.

There ARE Makefile.archs scattered around the MTS development machine, though! The one that appears to be the most interesting is /etc/root/Makefile.arch, a self-described "Makefile containing platform dependencies for ROOT based projects.". Since AMORE depends on ROOT, this looks like what we need. /etc/root appears to contain all the ROOT configuration rather than /usr/local, so I'm gonna set the "root-config" prefix to /etc/root. The current prefix configuration can be viewed with root-config --config.

Because root-config wasn't cooperating when trying to change the prefix via root-config --prefix=/etc/root, I found the source code, /usr/local/bin/root-config and changed the prefix setting found in the configargs variable. That alone, however, does not solve the problem; the configuration is not saved.

I checked to see how often root-config --prefix was used in building AMORE. It's only used once in that one file, so I just manually entered the correct path. The make went much further this time, but it still failed. This time it's complaining that TDATEEventParser.h doesn't exist. That file is included by [root of AMORE repo]/src/publisher/SRSPublisher.h. A locate TDATEEventParser.h didn't turn up anything.

The original MTS does, however, have /opt/amore/include/amore/TDATEEventParser.h. I used netcat to throw the MTS's file into the same directory on the MTS development machine and tried make again (NOTE: All TCP connections other than at port 22 are blocked by default, so iptables must be used to free up a port). Now we have different errors in SRSPublisher.h; they are all complaining that "Session" does not have a type.