CLEO-c Monte Carlo Farm at Minnesota

Alexander Scott

28th April 2008

1 Introduction

The University of Minnesota group has been one of the main producers, along with the University of Florida, of large Monte Carlo samples for the CLEO project. The current Minnesota Monte Carlo facility was developed in 2003 to meet the simulations needs of CLEO-c. It now consists of approximately 45 machines with over 160 virtual processors for running jobs in parallel, with Condor as the parallel processing distribution agent. The storage space for Monte Carlo generation and staging is provided by a RAID array with a 5 TB capacity that is serviced by 3 dedicated servers, each with its own share of the disks and a specific function. One is the access point for constants used by the Monte Carlo farm, one handles the libraries and generation scripts, and one handles the output from the farm. Perl code to run Monte Carlo generating jobs on the farm nodes was developed by Alex Smith and modified by Alexander Scott, but the generating code itself, and the constants needed to simulate the CLEO-c data, is provided by Cornell.

There are four people involved in the operation of the Minnesota Monte Carlo farm: Graham Allan (who manages the network connections and Condor), the constants person at Cornell (who generates constants savesets), the software manager at Cornell (specifies Monte Carlo parameters and provides code releases), and the Monte Carlo farmer at Minnesota. The Monte Carlo farmer is responsible for being familiar with the hardware and software components of the farm, generating the Monte Carlo, and maintaining and troubleshooting the farm components. These duties will be outlined below, in the form of a how-to manual.

2 Farm Components

The Minnesota Monte Carlo farm components can be subdivided into the categories of hardware and software. The hardware component, buying, installing, maintaining, and supervising, is principally the responsibility of the farmer. The software component is maintained by others (Graham in the case of Condor, and Cornell in the case of constants and code). However, the farmer is responsible for the specific implementation of the software and may need to debug and send reports to the appropriate authority. The hardware and software are briefly described in this section, with a description of use in Section 3 and a guide for maintaining and updating in Section 4.

2.1 Hardware

Most of the farm hardware resides in the sub-basement room S43. There are 45 machines rack-mounted there, 2 free-standing machines, and the RAID array for data storage. There should be a gigabit connection between all the farm machines. Disks for shipping are traditionally mounted on the farmer's terminal.

2.1.1 Linux boxes

There are three machines that act as servers (twserv1-3), 40 nodes (twins4-44, exclusive of twins10), and two new machines that also serve as nodes.

- twserv1, twserv2, twserv3 are original farm machines: dual Xeon 2660 MHz processors with 1 GB RAM, running SL4 (bought 2003?). twserv1 is the constants server and should be the box used for submitting jobs. It exports `/data/ccon`. twserv2 exports the Monte Carlo collection and storage area (`/data/mcfarm`). twserv3 exports the code and analysis areas (`/data/farmlib` and `/data/farmlib2/`).

- twins4-31: dual Xeon 2.40 GHz processors with 1 GB RAM, 20 GB storage (from INTERPRO, based in CA - bought early 2004). Twins4 is a special node, with SL3 installed in order to compile CLEO-c code. Twins10 has been taken out of the farm to act as the condor server. The other machines are nodes in the farm and are hyperthreaded to serve 4 jobs simultaneously. The average performance of these machines for each virtual processor is 1.3-1.5 sec/event for DDbar type Monte Carlo, 1.5-1.7 sec/event for DDMix (mixture of D, D*, and Ds) type Monte Carlo, 0.8-1.3

sec/event for continuum type Monte Carlo, 1.3-1.8 sec/event for radiative returns type Monte Carlo, 1.2-1.4 sec/event for psi(2S) simulation, and 0.6-0.9 sec/event for tau-tau type Monte Carlo.

- twins32-44: dual Xeon 2.66 GHz processors with 1 GB RAM, 100 GB storage, running SL4 (from NOW Micro Inc in St. Paul - bought late 2005). The performance specifications are the same as above.

- mnhep13 and mnhep20: dual quad-core 2.33 GHz processors with 8 GB RAM, 2-3 TB storage, running SL4 (from Dell - bought mid 2007). These machines are desktops, not rack-mounted, and are also used for analysis. We have found that the production ratio of these machines to the twins machines is lower than expected (based on analysis job speeds). The rate of generation falls off as the total number of jobs in the farm increases (the mnhep# machines can be up to 100% idle when running at full job capacity). We believe that this is due to increased demands on the constants server. The mnhep machines are approximately 3x as fast as the twins machines, and can run twice as many jobs.

There is a significant hardware problem that crops up in the twins machines, in that the machine will quickly (1-3 hours) crash with a kernel panic when running a suez job from Condor. This problem has been found in the past to track the motherboard, and replacing the motherboard has stopped the crashes. This is discussed in more detail in a later section.

2.1.2 USB drives

We maintain a number of USB external hard drives to ship Monte Carlo to Cornell. The cost of transferring files to Cornell over the Internet is prohibitive, so we copy the simulations to a USB drive and ship via FedEx. Three-day shipping costs approximately $15, so a full 250 GB disk shipped with three-day delivery costs $0.12/GB (including return shipping) with a bandwidth of 1 MB/s. These disks are currently mounted on mnhep1 and will probably be mounted in the future on the new farmer's workstation.

2.1.3 RAID array

The Minnesota RAID array is a JetStor SATA 316F, which hosts 16 disks with a capacity of 5 TB. The storage space is split between the three servers, with 1056

GB allocated to `/data/mcfarm`, 500 GB allocated to `/data/ccon`, and 2 TB to each of `/data/farmlib` and `/data/farmlib2`. Graham Allan is responsible for maintaining the RAID array. Technical details on this model should be available at `http://www.jetstor.com`.

2.2 Software

Most of the software used by the Minnesota Monte Carlo farm is built and maintained by others. Graham Allan manages the Condor releases now, as well as the installation of Scientific Linux operating systems and updates. Pete Zweber coordinates from Cornell the code releases to be compiled at Minnesota, while Debabrata Mohapatra creates the constants savesets that Minnesota needs. The farmer in most cases only needs to be competent to recognize errors and provide detailed feedback on those errors to the experts, although code installation in some cases is up to the farmer.

2.2.1 Condor

Condor is the parallel processing agent used by the Monte Carlo farm. According to the Condor website,

> Condor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.
>
> -http://www.cs.wisc.edu/condor/description.html

Condor is currently maintained at Minnesota by Graham Allan. The Minnesota Monte Carlo farmer should also have a folder containing documentation on Condor and Condor commands. More information on using Condor is available in later sections.

2.2.2 Constants/servers

The constants for CLEO-c Monte Carlo generation are held in an Objectivity database with a license from Cornell (it has only had to be renewed at Minnesota

once in the past five years). The constants databases, archives, and servers are located in the `/data/ccon` partition of the RAID array, with twserv1 as the constants server. There are databases for every class of constants, with two federated databases: Constants and RunStatistics. There are three servers to fetch from Cornell to serve constants: AllConstantsServer, RunStatisticsDBServer, and VersionManagerServer. These are maintained separately from releases and savesets, but very rarely need updating.

2.2.3 Releases

The Monte Carlo and analysis code is built into stable releases at Cornell and then installed at Minnesota. The software release for physics generation of a Monte Carlo job should be the most recent version, but the release for reconstruction should correspond to the release used to process the corresponding data set. The releases currently installed at Minnesota can be found in the directory `/nfs/cleo3/cleo3/Offline/rel`; many older releases previously built at Minesota have been deleted because they were built using the generator from SL4. As a rule of thumb, files at Minnesota have the same directory location as at Cornell, except for the second "cleo3" in the address. There is a release installer developed by Valentin to automatically handle the release installations.

2.2.4 Generating scripts

The Monte Carlo perl scripts were developed by Alex Smith to automate the process of generating Monte Carlo. The current version of the farm scripts are located in the directory `/home/hep/cleo3/farm_scripts_20040819`. The main function is `mc_start`, which initializes the generating process and calls the other functions. The scripts `mc_processor_cleog.pl` and `mc_processor_mcpass2.pl` create the `.tcl` scripts and execute the suez jobs that actually generate the Monte Carlo. The script `mc_congealer.pl` compiles generated files into volumes in the directory "/data/mcfarm/output/<FARMNAME>/" (where <FARMNAME> is the farm name parameter, such as Test_DDBarD42_01), keeps track of what stage of generation the files are in ("queued," "cleog," or "mcpass2"), and resubmits failed jobs. The script `mc_archiver.pl` moves volumes from the network to either a USB drive or to a remote location. The function `mc_webpage.pl` starts a webpage that displays information about the jobs.

2.2.5 Webpage

The CLEO Monte Carlo farm webpage is located at `http://webusers.physics.umn.edu/~cleo3/`. It links to the most current farm page, which may vary if more than one farm is active, through which one can view the log area of each run and the associated output logs. The main webpage also has a link to a list of old farm webpages (although the log areas are probably deleted and so the run links will be broken), a link to a page that keeps statistics on the processors in the farm, and a page that specifies how to set up a request file.

3 Farm Management

Managing the Monte Carlo farm consists of generating Monte Carlo, shipping it, and fixing any hardware problems. There are also issues of maintenance which will be described in the following section.

3.1 Generating Monte Carlo

The Monte Carlo generation process is largely automated, but there are a few things that must still be done by hand (or are easier to do so). There are pre-generation manual steps to take and post-generation checks to be done by hand.

3.1.1 Pre-generating

The first step in generating Monte Carlo is to make a request file. This can be done in an automated fashion (by using the request form off the main farm page) or by hand. The automated form is currently not functional; it doesn't record the input values into a .request file. It can also be tedious, as many values have to be looked up, but it may be worth fixing. The directory `/home/hep/cleo3/farm_scripts_20040819/requests` holds the old request files for generating Monte Carlo, which can be copied to make a new request file that has the same physics. The parameters determining the farm name, the cleog and mcpass2 `.tcl` files, and the location of the user-defined `decay.dec` may need to be updated, but minor edits to an old request file are usually all that is necessary.

The cleog and mcpass2 code releases need to be set in the `cleo3rc` file in order to run properly. The file is `~cleo3/.cleo3rc_bash`; edit the values of the `CLEOG_REL` and `PASS2_REL` parameters and then source the file. It is a good idea to reboot all of the machines before generating a new sample; although it isn't required to run, the nodes develop errors over time and may freeze up or have kernel panics, so pre-generation is as good a time as any to reboot. The script for rebooting all the nodes is `/utilities/mc_reboot.pm` in the farm scripts area, but it requires super-user permissions on all machines (which cleo3 doesn't have, so I use my own account) and your password has to be entered for every machine to be rebooted, which can be tedious. After rebooting, the constants will obviously need restarting (use the command `mc_start_constants`). The USB drive that will hold the Monte Carlo archives should be mounted at this time, if it is not already. The machine on which the drive is mounted is a parameter in the request file and normally corresponds to the terminal that the Monte Carlo farmer uses. You can mount the disk with the command "mount /media/usbdisk" (probably done under your personal account).

3.1.2 Generation

The rest of the farm process is automated. Executing the command "mc_start <FARMNAME>" will begin the process. One common mistake is to write the farm name parameter as <FARMNAME.request> which will cause the farm to fail (it will append the ".request" and find no request file that matches). The `mc_start` function will create condor job files and submit them to the condor queue. Condor will pick up the jobs and assign them to nodes as CPU becomes available. The `processor.bash` script calls the scripts to handle cleog generation and mcpass2 reconstruction and output is created in the `/export/local/condor` area and then moved to `/data/mcfarm/output`. The congealer volumes the output files and the archiver moves them to the USB disk (or sends over the network to a remote location).

A typical Monte Carlo farm will take 24-72 hours to complete. During this time, the Monte Carlo farmer should monitor production to minimize dead time if an error occurs.

- The Monte Carlo webpage displays the status of the congealer and archiver, the status of each job (queued, in cleog, or mcpass2, or symbol "10" if the output is being moved), the machine it is running on, and the average generating time/event (if the job has finished). Clicking on the run num-

ber follows a link to the log area, showing the files used by the job and allows the logfiles to be read. The `processor*.out` file, `cleog*.log` file, and the `pass2*.log` file are the most informative when checking for errors. Sometimes the error is systemic, such as if all the jobs fail at the CORBA step (which indicates that the constants servers are not running or up-to-date).

- The command "condor_q" allows the farmer to see the list of jobs in the queue, while "condor_status" shows the status of the machines in the condor pool. The command "condor_status <HOSTNAME>" shows the status of the virtual processors of just one machine, like twins5, while the command "condor_status <HOSTNAME> -v" shows parameter values on those virtual processors. If jobs are not being picked up, it can be useful to check machine stats with this command to see if a node is not meeting job preconditions. This can be because the CPU is busy or there is not enough disk space available for the job (sometimes space has to be cleared manually). The status webpage linked to from the main farm webpage tracks some of this information, but it updates too slowly to be of much use (and it hangs if one of the machines hangs, which could be fixed in the future).

- The farmer can set, as one of the environment variables, how many times a job will be retried if it fails. Some runs are bad, and they will not ever successfully generate. Some generators have buggy code (like the Lund Area Law for continuum), so that there are random failures but trying again with different random numbers may be successful. Trying three times before marking a job as failed seems to be the most efficient number for producing despite random failures while not wasting too much time on dysfunctional runs. If a job hangs so that no failure message is returned, the farmer needs to run "mc_abort_job <FARMNAME> <RUNNUM>" to create an abort.status file in that log area. The congealer will then mark that job as aborted and will not resubmit it. This can be necessary to handle runs that were picked up by a machine that later entered a hung state since no job failure message can be returned.

- If a machine that is not a known problem machine hangs while running, you should go downstairs and reboot it. It takes up to twenty minutes for Condor to reassign jobs to a rebooted machine. If a known trouble

machine hangs (like twins35 or twins42), it is usually more productive to leave it out of the condor pool because it will likely fail again in 1-3 hours, requirig more maintenance.

- Sometimes a lot of jobs fail at once and then it is worth resubmitting failed jobs (a total of 2-3 failed jobs are usually not worth resubmitting). This can be due to a power glitch taking down all machines, or an `nfs` error preventing access to the code (this is relatively common), or sometimes one machine will develop an error and serially pick up jobs and fail them (if the error occurs quickly enough, one machine can empty the entire queue before another machine becomes free). In this case, use "mc_resubmit -all <FARMNAME>" to resubmit just the failed jobs ("mc_start <FARMNAME>" will redo the successful ones too). If there is insufficient space in `/data/mcfarm` for additional files, the congealer will halt the farm. If the farmer clear space quickly, the condor command "condor_release" will release the hold on cleo3 jobs in the queue. If too much time goes by, the jobs cannot be restarted by releasing the hold, and the farm will have to be killed using `mc_kill` and the jobs resubmitted using "mc_resubmit -all <FARMNAME>."

3.1.3 Post-generation

There are some final checks to do before sending archives. Verify that each Monte Carlo farm job has all its volumes by comparing the numbers on the USB disk and the farm job's output area, which is `/data/mcfarm/output/<FARMNAME>`. Sometimes the archiver doesn't find the last volume, and the webpage reporting is incorrect as well. If a volume is missing, restarting the archiver will fetch the remaining volumes.

After a farm has been entered into Eventstore at Cornell, then it can be deleted here. Use "mc_clean <FARMNAME>" to completely remove the old farm. This may also be necessary if you decide to wipe out a farm that is running and start over.

3.2 Shipping Monte Carlo

We currently use a USB disk to ship Monte Carlo from Minnesota to Cornell; this disk is traditionally mounted on the farmer's terminal and is either 250 or 1000 GB. The archiver copies volumed Monte Carlo in 5000 MB chunks to the

USB disk. If more than one Monte Carlo farm is copied to the disk, make a list of the contents. Be sure to unmount the external drive correctly before flipping the power; a failure to do so can make the contents unreadable. If the USB disk won't unmount, ususally it is because someone is logged into the mounted directory.

We have a number of cardboard boxes for shipping either 1 or 2 disks at a time. Pack the USB enclosure securely (no need to include the power or USB cable) in the shipping box and take it to room 80. Weigh the package and have Ron Huhn FedEx it to:

> Curtis jastremsky
>
> Wilson lab
>
> Dryden Road
>
> Ithaca, NY 14853-8001

using the expense account number 533-XXXX. The first $100 of insurance is free, so only the 1000 GB drives need additional insurance. Ron Huhn will handle the rest. Post a message on the hypernews forum when a disk is shipped and list its contents.

3.3 Fixing hardware

The Monte Carlo farm manager should get key card access from Graham to room S43 for off-hours computer maintenance. There are a few common problems that can be fixed by the farm manager:

- Power failures: sometimes the power flickers and all the farm machines need rebooting. Currently, Graham Allan maintains mnhep13 and mnhep20, but the twins machines should be rebooted by the farm manager
- Unresponsive machine: sometimes a job will hang a machine and a reboot is necessary. There is a terminal in S43, connected to the farm KVM switch, that can be used to read the last output from the stuck machine. I suggest keeping a spreadsheet of machines, date of crashes, and reason for crash (frozen, kernel panic, write error, etc.). Reboot the machine and make sure that it comes back; sometimes additional maintenance or the "root" password is needed.

- Hardware problems: the Maxtor 80 GB drives are a known problem, and it may be that not all of them have cycled out of usage in the farm. Memory chips may also need changing. This can be verified either by running a diagnostic program on the machine or asking Alex Schumann to check his online diagnostic. More commonly, a machine will not be stable in the farm for more than a few hours. This has been traced to the motherboard, which we confirmed by swapping that motherboard with another machine's and keeping all other components the same. When you experience this problem, follow the same procedure. Graham Allan will need to change some settings for the swapped machines to access the network correctly. If the problem tracks the motherboard, toss the board and buy a new one. We last bought boards from NOW Micro Inc. There are currently 3 machines which exhibit this problem: twins17, twins35, and twins42. Only twins35 has been crashing often enough to warrant a motherboard replacement. As a final test, the motherboards of these three machines have been swapped with a machine one number higher (e.g., the board in twins17 has been put in twins18 and vv.) If the problem tracks, then a new board should be purchased.

4 Farm Updates

It is possible that, by the time that a new farmer takes over the Minnesota Monte Carlo farm, there will be no need for futher code updates for CLEO-c. In case there is a need for future updates, the procedure for installing new code and constants is outlined below.

4.1 Updating Constants

After a dataset is finished, CLEO experts begin fixing the constants for that dataset. After all values are in, the Cornell constants manager should create a saveset that is a snapshot of the Cornell constants set. The farm script `mc_maintainer.pm` contains a routine called `mc_update_constants` to fetch and install the new constants saveset at Minnesota. The constants should be updated from twserv1 (because it has a direct connection to the constants disk space and the user "cleo3" can has a public key to Cornell from twserv1) using the command "mc_update_constants". The objectivity lock server (ools) must be running to successfully update constants (if it is not, execute "ools &").

There are a number of parameters that the routine `mc_update_constants` requests (defaults are set by `.mcfarmrc`):

- The Cornell location where the archive is stored [`/mnt/ccon3/backup/dated/`]
- The name of the Cornell computer on which the archive is stored [`lnxcon.lns.cornell.edu`]
- The local destination of the incoming archive [`/data/ccon/ccon3/archives`]
- The Cornell base directory for the database [`/mnt/ccon/ccon3`]
- The local directory in which to build the database [`/data/ccon/ccon3/constants`]

The default values are correct unless you hear otherwise from Cornell, so pressing <enter> to accept the defaults is correct. The routine then connects to Cornell and checks the archive area for savesets, which are labeled by year and month. The user should select a saveset from the list, which will almost always be the most recent one (for example, "2008-01"). There will then be a list of gzipped files containing the constants information (e.g., "/mnt/ccon3/backup/dated//2008-01/Constants-23_0_9..gz"). The user should select the first two sets of numbers in the filename (in the example above, the suffix is "23_0"), which will signify the version number but not the individual file number. The routine will then display a list of steps for installing the constants.

1. Fetch database archives - the routine will download the gzipped files from Cornell to the directory `/data/ccon/ccon3/archives` and unzip them. This download will take a few hours and will use about 40 GB of space. If the constants expert at Cornell doesn't change the suffix for each saveset, the user at Minnesota will have to manually agree to overwrite each old file with the new one when the gzunzip command runs.
2. Unpack the database archives - the routine will expand the saveset into the Constants and RunStatistics databases. It will have to delete the existing databases to make the new ones. The unpacked databases take up about 50 GB of space.

3. Install the database archives - the routine builds the federated databases Constants and RunStatistics. The parameters here are sensitive to what was used at Cornell to build the federated databases, so it is possible that the command will have to be adjusted on the fly. If there is a problem, manually execute the command listed by the routine and you will get an output statement of which parameters are incorrect and what they should be.

4. Update constants server executables - this is rarely necessary, and should be indicated by the Monte Carlo coordinator. It is usually done when a new constants class is added. The three servers, AllConstantsServer, RunStatisticsDBServer, and VersionManagerServer, take up 77 MB of space and require only about a minute to download.

It is probably a good idea to reboot twserv1 after finishing the routine `mc_update_constants` so that no old versions of constants are retained. Executing the routine `mc_start_constants` will kill currently running constants servers and start new ones, as well as killing and restarting the objectivity lock server and mico nsd. The log files are kept in `/data/ccon/ccon3/EXE`, but it is usually sufficient to verify that the servers were restarted (the logs can be checked in case of errors).

4.2 Updating Releases

Release updates are handled in two ways; either they are a new code release or a patch of a release already installed. New releases are handled using the release installer `CleoReleaseManager.py` from Valentin. The main difference in release installation between Minnesota and Cornell is the root directory , which is `/nfs/cleo3` at MN and `/nfs` for Cornell. The procedure for installing a new release (which takes approximately one working day) is as follows:

- Login to an SL3 machine (i.e., twins4). There are significant differences in observables in Monte Carlo generated on SL3 and SL4 platforms, so it is important to only compile on an SL3 machine.
- export "CXX=g++", "C3CXX=g++", and "C3CXXTYPE=g++"
- Change directories to `/nfs/cleo3/cleo3_rpm/`.
- Execute "cvs update scripts/" to update the installer and associated scripts

- Change directories to `/nfs/cleo3/cleo3_rpm/scripts/CleoReleaseManager/`
- Execute "CleoReleaseManager.py -cpp-version 3.2.3 -install -release <RELNAME> -c3Dir /nfs/cleo3 -scp 2>&1 1>& install_<RELNAME>.log < /dev/null &" where <RELNAME> is the release name, and the output is redirected to the log file for later diagnosis if there is a problem.

To patch an existing release (which usually takes a few minutes),

- Login to an SL3 machine
- Change directories to `/nfs/cleo3/cleo3_rpm/scripts/`
- Execute "cvs update manage_releases" to update the patching script
- Execute "./manage_releases patch <RELNAME> -u <PKGNAME_1> <VNUMBER_1> -u <PKGNAME_2> <VNUMBER_2> ... 2>&1 1>& patch_<RELNAME>.log < /dev/null &", where <RELNAME> is the release name, <PKGNAME_#> is a package needing an update and <VNUMBER_#> is the version number to update to (e.g., SuezScripts v16_02_00), and the output goes into the log file for later diagnosis if there is a problem.
- `manage_releases` builds the patch with a dead link for cern. Correct this by entering the directory `/nfs/cleo3/cleo3/Offline/rel/<RELNAME>/`, deleting the current soft link cern, and making a new link by the same name to `../../../cern/rel/current`.