CLAS_NOTE 2004-003

MySQL database for the monitoring of pass0/1 cooking of photon data

Clarisse Tur, Mark Ito

January 27, 2004

1 Introduction

The pass0 cooking has for main purpose the monitoring of the quality of the calibration throughout a given run period. The photon run groups traditionally cook one or a few files of every single one of their runs during the pass0 phase of their analysis. As soon as a given file is cooked, a certain number of monitoring programs are run on this file to produce root or hbook files and usually a text file with tables of numbers (or tables on the standard output that one can direct to a text file). As an example, one of the monitoring programs is called trk_mon, which when run on a cooked file, will produce an hbook file to precisely visualize the quality of the drift chamber calibration for that particular run as well as a text file containing numbers for this run, such as the drift chamber resolution for each sector and superlayer. The usual approach is then to parse through the text files and store the value of those variables per run number and file number in a database, in order to, later, draw plots representing a given variable as a function of the run number or file number. The deviation from a flat line anywhere on those plots will indicate a "bad calibration" for the corresponding runs and require a close look at the hbook or root files for those runs and eventually a recalibration.

A very simplistic perl database has so far been used to store those monitoring variables. But a MySQL database is also available and has the advantage of having a web interface (at http://clasweb.jlab.org/csql_db/index.html) allowing a quick display and visualization of the stored data per run number and file number. We recently modified the cooking scripts used for the cooking and monitoring of the CLAS data in order give its users the choice of storing their monitoring variables in the perl or the MySQL database. Our modifications are only to the benefit of the photon run groups, as the electron run groups have a completely different approach to cooking and monitoring.

2 How to get an "account" on the MySQL database server for your run group

Mark Ito is the one to contact to get a space on the MySQL database for your run group. Once Mark is done, he will provide your group with a host name, a user name and a database name. Gagik Gavalian is the one in charge of maintaining the web interface for this database. So, you need to contact him in order for him to create a link to your run group's database area on the web interface. Just give him the host name, user name and database name for your run group. No password is necessary.

Once both of these people are done, you can go on the site http://clasweb.jlab.org/csql_db/index.html, select your run group, and view the page allocated to it. There is a CLAS_NOTE by Gagik Gavalian ([1]) if you need information on this web site.

3 Quick reminder on the cooking scripts and how to indicate your choice of using the MySQL database instead of the perl database

The cooking scripts are located in $CLAS_PACK/scripts/cooking_scripts/. Please refer to the file called cooking.readme in this directory in order to know how exactly to use those scripts so as to cook the data files of your choice and monitor them.

The file called ENV_SRC_FILE in the cooking_scripts directory is the one each run group has to modify first. This file contains a set of environment

variables that indicate various paths for your run group such as the location of the raw data files to be cooked, the location where the cooked files or the hbook and root files produced by the monitoring scripts need to be stored, or where your executables are. If you wish to use the MySQL database instead of the perl database you need to indicate 4 additional environment variables: the host name, user name and database name for your run group and the name of the table that you want to create in your run group's area. Here is an example:

CSQL_DBHOST clasdb.jlab.org
CSQL_DB g7_offline
CSQL_USER offline_g7
CSQL_TABLE pass0_v1_test

Each time you want to create a new table, the CSQL_TABLE variable needs to be set to a new value.

The script called NEXT_TAPE.pl is one of the main cooking scripts. NEXT_TAPE.pl also needs to be modified by each run group, as this is the script that tells the farms which executable to use to cook the files (a1c or recsis) and which monitoring programs to run once the cooking is done. Each run group can indicate their choices by changing the value of $options there according to their wishes. For instance, the g7 group used the following options:
$options="\'+d +sp +mysql\'";
The meaning of those options is defined in the script called RunJob: +d means "mark cache file for early deletion", +sp means "Run standard photon processing" (each run group needs to modify the section of RunJob that tells the farms what to do if +sp is used, that is which flags to use for a1c and which monitoring programs to use after the cooking of each file), and finally the +mysql option tells the farms that you want to use the MySQL database instead of the perl database. Please note that the default (case where +mysql is not indicated among the options) is to store the monitoring variables in the perl database.

4 How to create a table in the MySQL database

We added a new perl script to the cooking_scripts directory called CreateDBtable.pl. This very simple script uses functions defined in a Perl API for the CSQL system (see reference [2] for detailed information and the section 6 of this CLAS_NOTE for a quick description) to connect to the MySQL database, and then disconnect after creating a table there with columns that bear the names specified in an array of hashes defined in it. Each column is defined in CreateDBtable.pl as a hash with 2 elements: a name and a type. Thus the table specification goes like:

```
@table_spec = (
              {
                  name => "startTime",
                  type => "datetime"
                  },
              {
                  name => "beamEnergy",
                  type => "float"
                  },
```

... and so forth.

CreateDBtable.pl takes for only argument the name (with full path) of the environment variable file ENV_SRC_FILE. Note that this is needed in order for the script to know which database to use, on which host, for which user and what the name of the table to be created is. So, after you have defined your environment variables, in order to create your table specified by CreateDBtable.pl, just type:
CreateDBtable.pl +env ENV_SRC_FILE
If you are unsatisfied with your table and want to delete it from the database and create a new one with different/more/less columns, use the following command line to erase a previously created table:

echo drop table <table_name> | mysql -h<host_name> -u<user_name> <database_name>
Example: echo drop table pass0_v1_test | mysql -hclasdb -uoffline_g7 g7_offline

5 Which cooking scripts have already been modified to accommodate the storage of monitoring variables into the MySQL database

We modified the following scripts as of today:

RunA1c.pl : for a given file, the standard output of a1c is parsed and the following information is extracted from there and written in the MySQL database if the +mysql switch is used, in the perl database otherwise: the beam energy, the torus current, the mini torus current, the tagger current, the time when a1c starts running, the number of events it reads, the number of events it writes and the time it finishes.

RunTrk_mon.pl: all the tables produced by trk_mon for a given file on the standard output are parsed through and all the numbers that they contain are written in the MySQL database. There are too many numbers there to be cited one by one here, so, please run trk_mon once for yourself and see the tables it produces.

RunPid_mon.pl: again, all the useful tables produced by pid_mon on the standard output are parsed through and inserted into the MySQL database.

6 How to further modify the cooking scripts in order to extract additional variables for your particular run period

Further modifying the cooking scripts to extract more variables and insert their values into the MySQL database is extremely simple. You need to make use of the subroutines already written by us in a Perl API called Csql.pm and located in the directory $CLAS_TOOLS/perl. There are 5 subroutines there, with very explicit names: ConnectToServer, DisconnectFromServer, CreateTable, InsertRow and UpdateRow. A more detailed description of this module is given in the reference [2]. If you want to extract variables from the text output of a program for which we did not modify the corresponding cooking script (anything else than RunA1c.pl, RunTrk_mon.pl

and RunPid_mon.pl), you will need to do the necessary modification yourself. Edit the corresponding cooking script (for instance RunRF_mon.pl) and copy the following 2 lines at the beginning of that script:

```
use lib ("\ENV{CLAS_TOOLS}/perl");
use Csql;
```

Then you will need to define each additional column as an additional hash in CreateDBtable.pl (just look at how all the other columns are defined there), and insert the corresponding values by parsing through the text output and calling the subroutine UpdateRow; this is done in the cooking script that runs the monitoring program that you want to modify (for instance RunRF_mon.pl). Just have a look at the cooking scripts that we already mentioned as modified in the previous section to give yourself some examples on how to do that. Once again, this is very easy, so do not be scared.

7 Conclusion

We recommend the use of the MySQL database for the monitoring of the calibration quality during the pass0 stage of photon run analyses. The web interface that this database has, makes it very easy to view the information that you are storing there anytime during or after the pass0 cooking. The cooking scripts have already been extensively modified to extract many useful variables and it is extremely simple to further modify the scripts in order to extract more variables thanks to the Perl API described in section 6.

References

[1] Gagik Gavalian, 2002. CLAS_NOTE 2002-011.

[2] Mark Ito and Clarisse Tur, 2004. CLAS_NOTE 2004-002.