Automatic analysis for $\mathcal{TWIST}$
Technical note #73

Andrei Gaponenko

November 2, 2002

Abstract

To facilitate an early detection of possible problems affecting $\mathcal{TWIST}$ data it has been decided to analyze a fraction of data automatically soon after they are taken. That should make the experiment more efficient in taking good quality data. This note describes an implementation of a system automatically running the analysis.

Introduction

The system to automatically process $\mathcal{TWIST}$ data in a "semi-online" mode has been implemented as a set of Perl scripts. It receives input from DAQ about termination of a run, then proceeds with analysis of those data files from the run which can be found on disk. It can run MOFIA and/or QOD. It can be configured to handle any number of "data streams" and "analysis queues".

Data streams are kinds of data files written by DAQ. Presently we have two data streams: the main one, and sample files. For every data stream one can set up an arbitrary number of analysis queues. A queue defines a particular kind of analysis used to process data files.

For example, we may want to run both MOFIA and QOD on sample files, but run QOD only on full data files. To do this we can set up two queues for the sample run data stream, and the third one for the full run data stream.

Each queue is a LIFO, so that the newest[1] file gets analyzed first. If new data arrive faster than analysis of a particular queue goes, the queue grows, but we still have hbook files for fresh runs available. When there are no new data, older data are processed, and the queue shrinks again. A maximum size limit is configurable per queue. When a queue is bigger than the limit, a number of oldest entries is discarded to bring the size to it's allowed value.

There can be an arbitrary number of "analysis daemons" processing a queue. They can run on multiple hosts. So the amount of analysis we want to do automatically is pretty much our choice. It's limited by the total CPU power

[1] Everywhere in the note I say "newer" as a synonym to "having a bigger run number". The system uses run numbers, not time stamps of any kind.

available to $\mathcal{TWIST}$. (There may be other limits, like a limit on the number of files simultaneously open on a single disk as well.)

It's possible to safely update analysis software without stopping the system following a procedure described below. For each version there is a directory tree which keeps all related files: the binary executable, the kcm file, and the output files produced. Thus the complete history is kept.

The technicalities

To set up a data processing queue one needs to create its directory structure. See Fig. 1. Then DAQ should be told to fill the queue with new entries (`endrun.pl`), and one or several processing daemons (`ad.pl`) should be started on the queue.

Requests for processing are made by creating symbolic links in the `queued/` directory (Fig. 1). The links point to data files. Normally the links are created by script `endrun.pl` which is called by DAQ at the end of each run. One can also make such symlinks manually; e.g. to request processing of older runs after installing a new MOFIA version. While a queue entry `runXXX.ybs` is being processed, there is also a lock `runXXX.ybs.lock` in the `queued/` directory. After the processing is finished, both the item and the lock are removed.

endrun.pl

This script provides interface between DAQ and the automatic processing system. Its configuration section consists of two hashes, `%datastreams` and `%queues`. The former names DAQ data streams and gives pathname formats for their data files. The latter defines data processing queues for the data streams.

An example of configuration section of the script is shown on Fig 2. It defines three queues (`split_mofia`, `full_mofia`, `full_qod`) for two data streams (`split` and `full`).

Obviously all the "queued" directories above should be writable by the script. If DAQ and the automatic jobs are run by two different users, the `endrun.pl` script should be suid to the latter one. In the present set up the analysis is run by e614, while DAQ is run by twistonl. Remember to set the suid bit again if you modify the script.

ad.pl

This script is the "analysis daemon" watching a queue. When it finds a queue entry it runs a job on it. There can be several `ad.pl` processes watching the same queue. To avoid conflicts they use a locking technique utilizing symlinks, which works well on NFS.

The `ad.pl` script requires one argument: instance id number. It's an arbitrary integer which should be unique among daemons simultaneously running on the same host and watching the same queue. There is a mandatory environment variable `TOPDIR` defining the queue the daemon should process. An

split_mofia/			⇐	TOPDIR
	control/			
		log/	⇐	`ad.pl` log file(s).
		queued/	⇐	The queue. Items are symlinks to ybs files to be processed.
		locks/	⇐	`ad.pl` locks.
	current@		⇐	Symbolic link to one of directories below.
	20021006/		⇐	Version for October 6, 2002.
	20021006.v2/		⇐	Another version.
	....			
	20021025/		⇐	October 25, 2002.
		VERSION	⇐	Explains how the binary was made.
		config	⇐	See text.
		mofia	⇐	The executable.
		sample_runs.kcm	⇐	The kcm file.
		data/	⇐	Output: hbook, mofia logs, etc.

Figure 1: An example of a queue directory tree. Items in bold font should be in place before starting the queue.

```
my %datastreams = (
    split => '/twist/data7/current_split/srun%05d.ybs',
    full => '/twist/data_onl/current/run%05d.ybs',
);

my %queues = (
    split => ['/twist/data7/auto/split_mofia/control/queued',
      ],
    full => ['/twist/data7/auto/full_mofia/control/queued',
     '/twist/data7/auto/full_qod/control/queued',
      ],
);
```

Figure 2: Configuration section of `mbatch.pl`

optional environment variable `MAX_QUEUE_SIZE` can be used to override a default limit on the number of entries in the queue. If the number of entries exceeds the limit, `ad.pl` will truncate the queue before running the next job. You have to use an absolute path name of the `ad.pl` script to start it, so that it can find other scripts it calls.

Each `ad.pl` process gets a unique id string composed of the script name, the instance number, hostname, pid, and the start time of the process. The string is used as a target for all "lock" symbolic links made by the script. It is also used as a prefix for all log messages.

On start up the script tries to create a lock in the $TOPDIR/control/locks/ directory by making a symlink named like "ad.pl-1@m13beam.triumf.ca", where "1" is the instance number. If a read back link value is not the same as the script unique id, the lock is not acquired. The script checks for stale locks and removes them. If the lock still can not be obtained, `ad.pl` exits quietly. This feature can be used to run the daemons from crontab, so that they restart automatically after a system reboot.

Before running each new job, `ad.pl` checks that the lock is still valid. If not, it terminates. This gives the convenience of stopping a queue processing without logging on to all hosts where the daemons are running.

The `ad.pl` script call `mbatch.pl` which actually runs the analysis. But before doing so it makes sure the data file is not removed from disk.

keep_file.pl

Since the data file is usually read by the analysis program via NFS, deletion of the file (e.g. by lazylogger) would crash the analysis. A way to secure data on disk is to hard link the data file to a temporary name.

The `ad.pl` process itself may not have a write permission to the online

```
# Config file for MOFIA

PROGNAME=mofia
MOFIA_INIT=sample_runs.kcm
CAL_DB=/home/e614/e614soft/caldb_ascii
CFM_DB=/home/e614/e614soft/cfm/db_noswap
```

Figure 3: An example of config file for `mbatch.pl`

data directory, that is why it makes the hard link by calling an auxiliary script `keep_file.pl`. The `keep_file.pl` script is suid to a user who can write to online data directories. (twistonl in the present set up.) The temporary name is removed by another call to `keep_file.pl` after the analysis is done.

mbatch.pl

This script is called by `ad.pl` with a single argument: a name of the file to be analyzed. It parses config file in the "current" directory (see Fig. 1). The config file is self-describing. Figure 3 shows an example.

The `TOPDIR` environment variable should be set before calling `mbatch.pl`. Normally it is inherited from `ad.pl`.

How to...

Update MOFIA version used for automatic jobs

There is no need to stop processing.

First of all, prepare the MOFIA binary and the kcm file you want to use. The best way to make sure the results can be reproduced is to do a cvs checkout (with a tag or date option!) into a new directory and compile everything from scratch. Write a `VERSION` file explaining how to reproduce the binary.

Become e614.

Suppose you want to update software for the "split_mofia" queue. Create a new directory in `TOPDIR/split_mofia`. For example `20021103`. Copy the binary, kcm and VERSION files there. Copy `config` file from the "current" directory to the new one and modify it if necessary. After all the preparations are done:

```
e614[~]>cd /twist/data7/auto/split_mofia/
e614[~]>ls -l
```

Note where "current" is pointing to.

```
e614[data7/auto/split_mofia]>rm current; ln -s 20021103 current
```

When the current analysis job is completed, the next one will be started in the newly set up directory. A group writable `data/` subdir is automatically created. You can also create the `data/` subdir manually. Don't forget to make it group writable in the latter case, so that `paw++` users can easily save plots. Also, it makes sense to copy kumac and related files from the previous "current/data/" to the new one.

Stop an analysis queue gracefully

Remove lock file of the daemon you want to stop from `TOPDIR/control/locks/`. As soon as the current analysis job is completed, `ad.pl` will terminate.

Stop an analysis immediately

Remove the lock file as for the graceful stop. Then log as e614 onto the machine where the daemon is running (the host name is obvious from the lock file name) and kill mofia or qod. Don't kill the perl scripts. You may need to use

```
ps ux --forest
```

to get pid of the "right" mofia process.

Start a queue analysis

Become e614 on the machine there you want to run the analysis. Then start the `ad.pl` script with proper environment and arguments. For the `split_mofia` queue there is no need to specify `MAX_QUEUE_SIZE`. For `full_mofia` and `full_qod` it's better to do so, though not absolutely necessary. Example for the `full_qod` queue:

```
e614[/twist/data7/auto]>cd /twist/data7/auto
e614[/twist/data7/auto]>bash
e614@m13beam /twist/data7/auto$ TOPDIR=$PWD/full_qod \
             MAX_QUEUE_SIZE=300 $PWD/online_scripts/ad.pl 1 &
```

Repeat on other hosts or with other instance numbers to start more daemons for the same queue.