Monitoring the EVN via Field System logfiles

Des Small, JIVE

2009-04-09

Introduction

We describe the prototype of a database-backed client-server logfile analyser/plotter that uses "live" logfiles from station field systems. The system uses the correlation control computer's database management system, shell scripts at the stations (to copy logs to Mark 5s), Python scripts at JIVE to copy log files to a local machine, analyse them and populate the database, and a C++/Qt client for operators and support scientists to use to examine the logs.

The prototype is limited only in so far as few dataseries are currently transferred from the logs to the database; work to extend its range is ongoing.

Outline of infrastructure

The current arrangement is that stations run a shell script (based on tail -f) to copy files to their local Mark 5 units. Much of the rest of the work, and all of the communication between the logfile processors and the monitoring clients is done through a database on the correlation control computer. (This shares the MySql DBMS with the correlation control system database, but is a separate database.)

A relational database may seem like an odd choice as a real time client-server protocol but it does have advantages: in particular the clients can query the latest results only (for live displays) or the history of a variable (to create plots), and the server and clients are strongly decoupled.

Details of infrastructure

The program getMark5IP.py consults the ccs database and a local configuration file to construct the station_info table, which maps stations to Mark5 IP addresses and stores the path on the Mark 5s where log files are kept.

Table 1: The station_info table

station	mark5_ip	basepath
Tr	158.75.6.207	/home/oper
Wb	192.168.2.56	/home/oper
On	129.16.208.28	/home/oper/
Mc	193.204.89.195	/home/oper/log/
Ys	193.146.252.31	/home/oper

The program availableLogs.py checks the Mark 5s for log files and updates the job info table. Querying this table is the preferred method for clients to find the most recent job.

Table 2: The job_info table

job	station	filename	local_filename	datetime
apr5c	Mc	apr5cmc.log	Logs/apr5cmc.log |	2009-04-05 19:17
apr5c	On	apr5con.log	Logs/apr5con.log	2009-04-05 18:01

Other individual tables are populated by other Python programs running on services.jive.nl; at the time of writing the log file “data_valid” diagnostic is used to track whether stations field systems are currently “on source” (see note below), and the formatter table is used to store the formatter time offset calculated by the station's GPS systems. The formatter table is of interest because there was a jump in the Mc formatter in a recent job (RF006A) which went undetected at the time and caused problems in the correlation; a few rows are shown in Table 3 below.

Table 3: The formatter table

job	station	datetime	gps_fmount
rf006a	mc	2009-03-24 14:44:36	7.4179e-05
rf006a	mc	2009-03-24 14:45:08	7.4178e-05

On-source indications in log files

Note that the log files contain various fields to indicate tracking and changes of source, some of which are excepted below (from the log file for Mc in the 100-hours demo APR3C):

```
2009.093.15:29:20.97#antcn#Commanding to a new source
2009.093.15:30:01.83/onsource/TRACKING
2009.093.15:30:08.19/onsource/TRACKING
```

However. the `/onsource/` directive only ever occurs with TRACKING or SLEWING for any station, and the `#antcn#` directive also doesn't say when a new source is acquired (rather than sought). Another candidate might be lines of the form:

```
2009.093.09:29:50.00&preob/onsource
2009.093.09:30:02.97&midob/onsource
2009.093.09:44:20.02&postob/onsource
```

But these occur only once in the whole Mc log for APR3C, even though two distinct sources are observed. As a result, I have fallen back on the data-valid flag. This has the disadvantage that it flags a period on the order of 40 seconds as invalid immediately before a new scan starts. However, it has the advantage of flagging scan boundaries. Since each scan has a source (stated in the log) it would be possible to have the log monitoring software edit out indications of scan-boundaries where the source does not change, and this would presumably be closer to the Platonic form of “onsource” indicators.

Suggestions from operators and support scientists on how this can best be handled are of course welcome.

Client software

The client software has been developed by Bob Eldering, using C++ and the QT widget set. (C++ was chosen largely because of incompatibilities between Python, the MySQL library and the installed C compiler on ccsops.jive.nl.)

A screenshot of the client in use is shown in Illustration 1. The main window (top left) shows the available data sets, and clicking within it causes plots of a given variable for a given station to be shown. Note that this display is for the job RF006A, and the jump in the formatter offset is clearly visible.

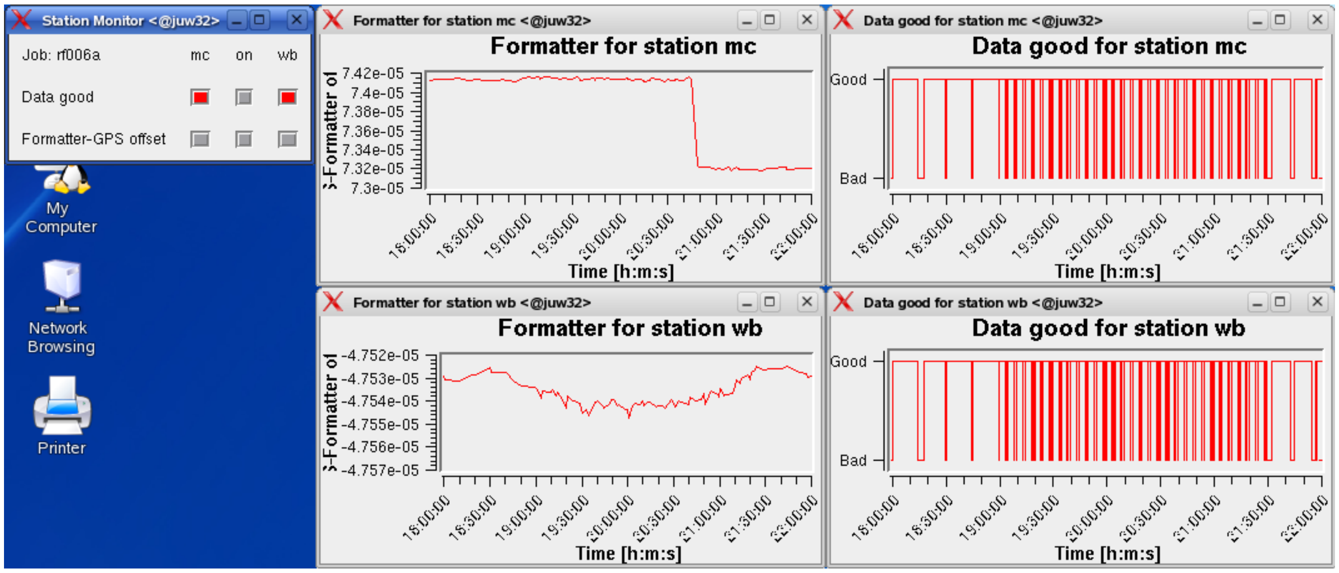

Illustration 1: Screenshot of the log-monitor client

Conclusions and directions for further work

We have developed a client-server database backed log-monitoring tool that is capable of analysing and plotting station log files either in real time or from archive data. The amount of log data currently processed and made available for viewing and graphing is currently not extensive, but now that the architecture has been proven we anticipate developing this much further.