MINOS Beam Data Process

B. Viren

November 23, 2010

Abstract

This note describes how the MINOS Beam Data Process works.

1 Overview

The MINOS Beam Data Process (BDP) is responsible for reading ACNET device data and producing ROOT files and filling a subset of devices into the Offline database. Minimal data analysis is performed.

The BDP consists of these major parts:

BDP Server initiates communication with and accepts data from Beam Division Data Acquisition Engine (DAE) and marshals it to an instance of the rotorooter.

Big Green Button a monitor[1] that alerts about problems that cause files to stop being written while spills are still being delivered.

BDP GUI a graphical user interface for monitoring the server.

BD DBU a database updater job[2] that does some minimal analysis on the data and uploads a subset to the Offlne database.

These three parts are described in more detail.

2 BDP Server

The server is found in the `BeamData` package's `python/` sub directory. The "server" works by first acting as a client and contacting the DAE through an XML-RPC call. It announces a URL. It then turns around and becomes a server waiting for XML-RPC callbacks to that URL. Each callback delivers one spill's worth of data which is marshalled to the rotorooter instance which writes it to disk.

2.1 Installation

The BDP server is intended to run as `minos@minos-beamdata.fnal.gov` although with proper configuration can run on any system that can talk to the DAE.

See section 9 for details on installing on a green-field system. In particular a stand-alone copy of a MinosSoft base release (R1.26 for most running), Python and some Python-related packages are needed. Finally, a private release is made to hold `BeamData` and `OnlineUtil` packages, The former is should be brought to the HEAD to pick up any rare fixes or changes in the device list.

To install type `make` and `make install` from `Beamdata/python/`. After the install the following directories will be populated.

[1] Rustem Ospanov.

[2] Mark Dierckxsens, Mary Bishai.

$HOME/python
$HOME/bin
$HOME/etc
$HOME/share

Files in these areas are typically make non-writable to avoid shifters playing with them. Any non-writable file should be modified in the BeamData package and re-installed.

2.2 Contents of BeamData/python/

Some important parts of the package:

cfg/ holds bdp.cfg which defines how the server communicates. This **must** be changed in order to safely run on another system than minos-beamdata. It also holds devices.cfg which lists the devices to store, the trigger and delay.

bdp/ supporting Python, C++ and SWIG wrapper code.

main/ the main bdp-server.py, bdp-gui.py and some utility programs.

scripts/ scripts to start/stop the server, gui and BGB.

2.3 Starting the BDP Server

There should be an icon on the minos-acnet display that will start it. To manually start it:

```
$ minos-kinit    # if needed
$ ssh minos-beamdata
$ start_bdp
```

It starts both the BDP server and a rotorooter. It is safe to run this if it has already been run. If either the server or rotorooter has gotten in a bad state they needed to be killed manually before restarting.

To gently stop the BDP server and rotorooter run:

```
[minos@minos-beamdata ~]$ shutdown_bdp
```

2.4 Output

Logs go to $HOME/run/bdp/ and data files go to /data/bdpdcp[3] where they are picked up by the archiver and the BD DBU cron jobs.

2.5 Basic Monitoring

See next section for official shift monitoring but some basic monitoring is also available. Log into minos@minos-beamdata and run:

```
[minos@minos-beamdata ~]$ check_bdp.sh
-rw-r--r-- 1 minos e875 210M Nov 21 10:00 B101121_080001.mbeam.root
-rw-r--r-- 1 minos e875 196M Nov 21 18:00 B101121_160001.mbeam.root
-rw-r--r-- 1 minos e875 223M Nov 22 02:00 B101122_000001.mbeam.root
-rw-r--r-- 1 minos e875 210M Nov 22 10:00 B101122_080001.mbeam.root
-rw-r--r-- 1 minos e875 195M Nov 22 18:00 B101122_160001.mbeam.root
-rw-r--r-- 1 minos e875 128M Nov 23 02:00 B101123_000001.mbeam.root
-rw-r--r-- 1 minos e875 164M Nov 23 10:00 B101123_080002.mbeam.root
-rw-r--r-- 1 minos e875 160M Nov 23 17:35 B101123_160001.mbeam.root
lrwxrwxrwx 1 minos e875   49 Nov 23 10:00 currentfile -> /data/bdpdcp/data-files/B101123_160001.mbeam.root
-rw-r--r-- 1 minos e875    0 May 15  2006 empty

minos     5058  0.1  1.6  66544 33824 ?       S    Nov19  11:45 rotorooter -l /data/bdpdcp/data-files/currentfile -p beammon -e
```

[3]Beam Data Process Data Copy Process

```
minos     5067 27.4  0.4  37736 10180 ?       Sl   Nov19 1743:01 python2.3 /home/minos/bin/bdp-server /home/minos/etc/bdp.cfg /home/minos/etc/devices.cfg

G:EA9SNC 8.918000 secs
E:TORTGT 35.622426 E12
G:E23SNC 1.695000 secs
-rw-r--r-- 1 minos e875 166892429 Nov 23 17:35 /data/bdpdcp/data-files/B101123_160001.mbeam.root

G:EA9SNC 1.594000 secs
E:TORTGT 0.005624 E12
G:E23SNC 3.462000 secs
-rw-r--r-- 1 minos e875 166973475 Nov 23 17:35 /data/bdpdcp/data-files/B101123_160001.mbeam.root

G:EA9SNC 0.971000 secs
E:TORTGT 35.715253 E12
G:E23SNC 0.277000 secs
-rw-r--r-- 1 minos e875 166973475 Nov 23 17:35 /data/bdpdcp/data-files/B101123_160001.mbeam.root
```

This shows the recently written files, looks for the BDP server and rootrooter running. It then lists the most recent file with a sleep in between. Unless the beam is down one should see the file change size.

Basic querying of ACNET can be performed. In particular to see when the last `$A9` trigger fired and how many protons were delievered, run:

```
~/BD/opt/bin/python ~/BD/R1.18/BeamData/python/main/poll-acnet.py G:EA9SNC E:TORTGT
G:EA9SNC 0.356000 secs
E:TORTGT 35.808079 E12
```

3 Big Green Button

The, so called, "Big Green Button" is an X11 window that does much the same thing as the basic monitoring. It should normally be green and displaying a recent time stamp. It tells the shifter the time since the last beam spill and the time since the last update to the file. The difference should be close to the nominal spill period.

4 BDP GUI

The BDP GUI is optional and not described.

5 BD DBU

The BD DBU runs from cron through a script:

```
[minos@minos-beamdata ~]$ crontab -l
...
# Run the DBU job 10 minutes after every hour.  It will just exit if there is nothing to do.
20 * * * * /bin/bash /home/minos/BD/dbu/BeamDataDbi/scripts/run_bdbu_fnal_cron.sh
```

It looks for files under `/data/bdpdcp/dbu`:

bdbu-logs/ holds logs

data-to-dbu/ where it looks for symlinks to new files

data-dbued/ where it moves completed symlinks

dbu-crashed/ where crashed files can be copied for later debugging

6 Purging Old Data Files

The disk nominally fills at about 15 GBytes/month and periodically old data files must be purged. It is important to only purge files that have been archived.

A helper script `$HOME/share/data-files.sh` exists to check which files are safe to remove. Just running it will make a list of files that are safe to remove. However, it is safest to remove only the oldest. This can accomplished like:

```
# list a specific month's files
[minos@minos-beamdata ~]$ data-files.sh|grep B0911
# delete them
[minos@minos-beamdata ~]$ cd /data/bdpdcp/data-files
[minos@minos-beamdata data-files]$ rm $(data-files.sh|grep B0911)
```

It can take a few 10s of seconds to remove a month's worth of files.

7 Recovering From DBU Failures

Convert the failed symlinks to files in dbu-crashed/.

```
cd data-crashed/
cp $(readlink ../data-to-dbu/BYYMMDD_HHMMSS.mbeam.root) .
```

Reproduce the problem:

```
source $HOME/.bashrc
setup_dbu
export USER=minos
loon -bq $SRT_PRIVATE_CONTEXT/BeamDataDbi/scripts/dbu_peds.C failed.root
loon -bq $SRT_PRIVATE_CONTEXT/BeamDataDbi/scripts/dbu.C failed.root
```

The first produces pedestal entries which are needed for the second.

If the failure is confirmed then debugging is needed. Known failures include floating point errors, failure of profile monitor fitter to converge or corrupted files. The latter can happen if the rotorooter was not shutdown cleanly.

The former can be handled by skipping the offending spill. Given a time stamp after the offending spill this can be done using BeamDataUtil/BDSpliceModule.

In the latter case, it may be possible to recover the file using ROOT[4].

8 Running a Backup BDP Server

A backup system can be run when there are known times that FNAL will conspire to bring down the primary server. The official backup server is minos-beamdata2 in the MINOS control room. It is set up just like the primary system except one must modify the bdp.cfg file to adjust these settings:

```
# required
server_url = http://minos-beamdata2.fnal.gov:19870/RPC2

# only if the gui is used
monitor_url = http://minos-beamdata2.fnal.gov:19879/RPC2
```

Everything else should be the same except that there is no DBU cron job and any files that need to be recovered need to be run through DBU by-hand.

[4] `Seethisthreadonroot-talk:http://root.cern.ch/root/roottalk/roottalk10/0062.html`

8.1 Recovering using Backup Data

It is possible to "splice" the backup files so they directly span the gap of missing data. This can be done with code in BeamDataUtil/BDSpliceModule and the scripts/splice.C macro. This will produce a new file from the input which spans the configured time range.

It is also possible to load in backups files even if they overlap with data from the primary server. The overlaps should contain identical entries and the DBI will handle the overlaps.

9 Appendix: Installation Details

This includes some notes on installing the required software on minos-beamdata or minos-beamdata2.

9.1 Install Externals

Essentially, the standard method is followed[5]. In addition follow the conventions for locations.

```
# Base area
mkdir -p $HOME/minossoft-sl5/{src,opt}
# libsigc++
cd $HOME/minossoft-sl5/src
wget http://www-numi.fnal.gov/offline_software/external_pkgs/tar_files/RECOMMENDED/libsigc++-1.2.5.tar.gz
tar -xzvf libsigc++-1.2.5.tar.gz
cd libsigc++-1.2.5
./configure --prefix=$HOME/minossoft-sl5/opt
make
make install
# libsigc++ Environment needed
export SIGC_DIR=$HOME/minossoft-sl5/opt
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SIGC_DIR/lib
# ROOT
mkdir -p $HOME/minossoft-sl5/src/
cd $HOME/minossoft-sl5/src/
svn checkout http://root.cern.ch/svn/root/tags/v5-16-00 root-v5-16-00
cd root-v5-16-00/
./configure linux --enable-mysql --enable-xml --enable-asimage --enable-minuit2 --enable-roofit > configure.log 2>&1 &
make
make install
# ROOT environment needed
PATH=$PATH:$HOME/minossoft-sl5/src/root-v5-16-00/bin
```

In addition Python and Twisted are needed. Normally, Python is not an explicit dependency on MinosSoft but when BDP Server was first written the Python that came with FermiLinux was archaic. Only the versions shown have been used.

- Python-2.3.4.tgz
- Twisted_NoDocs-1.3.0.tar.gz

Download these files and install them into `prefix=$HOME/opt`. Beware of poluted MAKE-related environment variables due to SRT lameness. Start with a clean shell before going on.

```
mkdir -p $HOME/BD/opt/src
cd $HOME/BD/opt/src
wget ...
tar -xzvf Python-2.3.4.tgz
cd Python-2.3.4
./configure --prefix=$HOME/BD/opt
make
make install
tar -xzvf Twisted_NoDocs-1.3.0.tar.gz
cd Twisted-1.3.0/
~/BD/opt/bin/python setup.py install --prefix=~/BD/opt
```

[5] `http://www-numi.fnal.gov/offline_software/srt_public_context/WebDocs/external_products.html`

9.2 Base Release

Follow the usual 12 steps of msrt. Production has almost always ran R1.26.

9.3 Private Release

Follow the usual method for a private release and include OnlineUtils and BeamData.