MINOS Beam Data Process

Brett Viren

Physics Department
Brookhaven National Lab

2011/01

Overview

MINOS Beam Data Process Suite

- bdp-server
- Big Green Button

Database Filling

Maintenance Requirements

Suggestions for Minerva

The MINOS Beam Data Process (BDP) suite

Online:

bdp-server	marshals data from Beam Division source to MINOS "*rotorooter*" sink (output files)
"Big Green Button"	output file monitor for shift to stare at
GUI monitor	internal-state monitor, for experts or bored shifters
poll-xmlrpc	CLI script for polling XML-RPC for device info

Offline:

archiver	loads output files to tape
BDP DBU	**d**ata**b**ase **u**pdater, fills `BEAMMON*` tables
BeamMon* Packages	MINOS packages, classes to work with the data

The Beam Data Process server (bdp-server)

`bdp-server`

- Implemented in Python using the asynchronous network framework, Twisted[1].
- Upstream communication is via XML-RPC, downstream is RotoTalk
- Runs on `minos-beamdata` in FCC and, in back up mode, on `minos-beamdata2` in MINOS FNAL control room.
- Configured with ACNET trigger/delay, device list and server location.

[1] `http://twistedmatrix.com/`

bdp-server Operation Overview

1. Start up as client, connect to BD server to register location
2. As server, wait for and accept data from BD DAE client
3. Format data, pass along to rotorooter server which writes to file.

bdp-server: Upstream Communications (Charlie King)

Registration request (`startList` command) made via XML-RPC to http://www-bd.fnal.gov/minos with arguments:

callback URL the URL on which the BDP will soon listen (http://minos-beamdata.fnal.gov:19870/RPC2)

callback function the name of the function to call on the BDP (`accept_callback`)

device list `I:NUTARZ`, `E:TORTGT`, `E:HPTGT[]`, etc.[2]

trigger and delay in arcane ACNET encoding ("`e,A9,e,500`" = 500 ms after 0xA9 trigger)

Request returns idstr, eg "`xmlrpc_12ccc9c6b9b`".
Then, client and server swap and data is pushed to BDP each spill. Each push must include a matching idstr.

[2] Definitive list in MINOS CVS and available via `http://minos.phy.bnl.gov/software/bd/BeamData/doc/acnet-devices-to-readout.txt`

bdp-server: Downstream Communication (Robert Hatcher)

The *rotorooter*:

- accepts raw data from MINOS DAQ, DCS and BeamMon (BDP).
- uses "RotoTalk" protocol completely hidden behind a C-API.
- BDP wraps this for Python with SWIG[3].

RotoTalk is simple:

- Open/close connection
- Close a file, open a new one based on a time stamp
- Send block of packed data

Although RotoTalk is a network protocol, bdp-server still must be able to access the output disks to perform some bookkeeping.

[3] `http://www.swig.org/`

Big Green Button

The Big Green Button[4]

- Monitors output file and polls ACNET via XML-RPC
- Complains if ACNET has new data but file is not updating
- Implemented in Python, simple X11 window.
- Runs on `minos-beamdata`, displays via SSH-X11 tunnel
- Recent call (Art Kreymer) to replace with MINOS DCS-style web page.
  - Remove need for `minos-beamdata` access.
  - Avoid rare, undiagnosed hang.

API documentation for XML-RPC polling at `http://www-bd.fnal.gov/xmlrpc/Accelerator`.

[4] Author: Rustem Ospanov.

Beam Data Process Database Updater (DBU)

The BDP DBU fills BEAMMON* tables:

- runs from cron on `minos-beamdata`
- watches for new output files to be ready
- one pass for profile/hadron/muon monitor pedestals
- one pass for everything else
- fits profile monitors, extrapolates BPMs to target

Periodically (few times a year) fails when fitting the profile monitors. File needs recovery by-hand.

Maintenance

On-going maintenance of Beam Data Process things

- Periodically purge fully processed/archived output files from `minos-beamdata:/data`. Enough disk for ∼1 year (∼10GB/month).
- Recover failed DBU files.
- Start backup on `minos-beamdata2` to cover known downtime.
- Restart bdp-server after system upgrade (shift can also do this)
- Respond to FNAL networking attacks.

Suggestions for Minerva

Based on 1/13 infrastructure meeting I understand you need three things:

1. General purpose, prompt display of beam quantities
   $\Rightarrow$ NuMIMon/JAS3.
   - already in use by multiple MINOS and Minerva sites.
2. Beam/detector correlation for production processing
   $\Rightarrow$ MINOS BDP/DBU/DBI.
   - default file latency: 8 hours
   - minimum latency $\sim$20 minutes (guess, untested)
3. Prompt ($\sim$minute) beam/detector correlation
   $\Rightarrow$ Something New^{TM}

#3 is a "mini-production" so best to keep as much code as possible the same with #2. This leads to a trade-off between desired latency and the amount of new coding required.

Suggestions for Minerva, how to do #3

Some possible solutions to consider for the "mini-production":

(A) Develop novel XML-RPC service populating a local cache
- Latency to fill cache is ∼1 second,
- XML-RPC multi-lingual, easy to code to
- requires novel code for beam/detector correlation, may require an async/threaded implementation

(B) Push on BDP/DBU/DBI latency, maybe ∼20min is enough?
- requires second BDP and DB installation (not a big deal) and someone to do minor code tweaks and test.
- use same DBI backend for beam/detector correlation

(C) Merge XML-RPC → DB into single application
- Latency of few seconds
- requires significant code development, but mostly porting of existing MINOS code
- may want a separate process filling same data into production DB (better optimization with longer validity ranges)
- use same DBI backend for beam/detector correlation