An Operator's Guide to the Processing and Filtering System

Version 1.0
Erik Blaufuss (blaufuss@icecube.umd.edu)

Introduction

The Processing and Filtering system (PnF) is designed to be the place where online, realtime filtering of events takes place for IceCube. The system consists of a central server (sps-fpmaster), and a farm of reconstruction clients (sps-fpslave0n, currently 2 machines). The reconstruction nodes run an instance of IceTray, the offline software framework (see Additional Information), allowing reconstruction modules written in the North to be directly plugged in.

System Overview

A short overview that is followed by more details for each system.

- All PnF systems can be accessed via ssh through the sps-access node.
- The sps-fpmaster node runs the server software.
  - The JBOSS JMX console is used to control, monitor and configure this system.
  - You must 'su' to the user jboss to control, view log files, or do anything to the system running on sps-fpmaster.
  - Logs, configuration data, etc. can all be found within the pfcommon area:
    - sps-fpserver:/usr/local/icecube/pfcommon (aka ~jboss/pfcommon)
- The sps-fpslave01,02 machines run the reconstruction software.
  - Reconstruction processes are started automatically at boot time and continuously try to connect to the data server.
  - If a reconstruction client dies for some reason, it restarts automatically.
  - Logs, configuration data, etc. can all be found in the pfcommon area (must be user jboss to see this information
    - sps-fpslave0{n}: /usr/local/icecube/pfcommon/

Gaining access

All configuration, control and monitoring of the PnF server is intended to be done via the JBOSS JMX console, although I will provide alternative access and control instructions in this document. The PnF system is

coded in C/C++, and the JMX control is done through a simple JAVA interface that calls control scripts, which can also be called on the command line (information is included here for both methods).

In either case, you need access to the sps-fpmaster node as the user 'jboss'. All scripts, log files, etc are owned by this userid. You can access sps-fpmaster via sps-access and then use 'su – jboss'.

To view the JMX console on a machine not attached to the sps-* subnet, use the ssh -L command to tunnel through:

ssh -L 8090:sps-fpmaster:8080 sps-access

Then, you can view on your local machine:

http://localhost:8090

will be the 8080 JMX console from sps-fpmaster.
**Warning: other systems use JMX, and each port specified as the tunnel port (8090 in this case) must be unique to prevent conflicts.

The sps-fpmaster machine

The "Master" process directs the flow of data through the PnF system (more information is available in the design docs, see Additional Information). It is a multi-threaded process that reads data in from the DaqDispatch interface, sends packets of data out to a reconstruction node for analysis, receives the results of reconstruction and filtering for each packets, formats the output, including reconstruction information and filtering information, for output to file(s) on disk. As of February 2005, no filtering or reconstruction is being performed, and all events are tagged as "keep" and since no new information is being added, these files are not being transferred over the satelitte.

Configuration items, either via file or JMX console are the same and include (defaults in parenthesis):

- DaqIP – IP address of DaqDispatch. As of February 2005, DaqDispatch will likely run on the sps-stringproc01 (10.1.3.11)
- DaqPort – Port of DaqDispath (2468)
- LogServerIP – IP address of experiment wide logging system, as of February 2005, not implemented yet (127.0.0.1)
- LogServerPort – Port of logging system (4445)
- PfSetupDir – Root of configuration and log directory tree (/usr/local/icecube/pfcommon)
- PfBinDir – Location of PnF executables, relative to PfSetupDir (bin)
- PfClient - <<Deprecated- no longer used, will be removed>>
- PfClientPort – Port for reconstruction clients to attach to get data chunks (7646)
- PfCommFIFO – Location of Fifo on disk, for interprocess

communication (/tmp/acqFifo)
- PfConfigDir – location of config scripts, relative to PfSetupDir (config)
- PfDataDir – location of output data (to be registered with Data Movement, absolute or relative to PfSetupDir. (/mnt/data/common/pfdata)
- PfErrorDir – Place for corefiles, debugging data dumps to be collected, relative to PfSetupDir (cores)
- PfLockFile – Name of lock file, prevents two instances of pfserver from running (/tmp/pfLockFile)
- PfLogDir – Location of log files, relative to PfSetupDir (logs)
- PfServer – Name of executable in PfBinDir (Master)
- PfServerIP – IP address of sps- fpmaster node, used by reconstruction processes to connect (10.1.2.50)
- PfServerMemory – Size of memory chunk alloted for data processing in bytes (82880000)
- PfStatusFile – Name of file used by JBOSS/JMX to find server status. (status.txt)
- PfTempDataDir – Directory used to write data as being recorded. Once closed, moved to PfDataDir, absolute or relative to PfSetupDir. (/mnt/data/common/tmpdata)

In general most of these values will never have to be changed and caution should be shown when changing them.

Using JBOSS/JMX to start PnF

(Remember: the reconstruction processes on the fpslave0n machines should already be running automatically.)

Make sure you have established communications with the JMX console via ssh tunneling (see Gaining access above.) Aim your browser to the tunneled port (localhost:8090 in the above example). Choose the JMX console item on the mainpage.

Now you will find three items under the **icecube.pnf** heading:
- configure – Used to change the configuration items listed above.
  - After setting all configuration items on this page, please make sure that you “Apply Changes”, then invoke the configurePnF item near the bottom. This creates the necessary files to run the PnF server, and must be done before using the control Mbean.
- control – Used to control the server process (startup/shutdown)
- monitor – Used to monitor the running process, including:
  - PfBuffersAnalyzed – Number of data chunks analyzed by workers ready to be written to disk, usually near zero. If this value grows too large, there might be a problem with writing to disk.
  - PfBuffersRead – Number of data chunks waiting to be reconstructed

by fpslave machines, usually near zero. If this value grows too large, there might be a problem with the pfclient machines. Check how many are connected and the contents of their logs.
- PfClients – Number of pfclient processes connected to the server. Usually two per operational fpslave machine.
- PfFracDiskSpaceAvail – Fraction of PfDataDir disk space available. If too small, check output disk area and processes that empty it.
- PfMemorayAvailable – Fraction of server memory buffer currently filled with data chunks. As the number of PfBuffersAnalyzed and fBuffersAnalyzed grows, this will decrease.
- PfQueues – Total number of chunks in server memory.
- PfServerStatus - Observed status of Master server process. Either "running" or "pfserver not responding". The later indicates that the server is not running, or is hung, please check with ps -e (see below).
- PfThroughput - Rate of data processed by PnF system (kB/sec)
- PfTime – Date and time when this monitoring data was collected. Please make sure it is current. Use the "Refresh Mbean View" at link at top to reload this information.

Using the command line to start PnF

As the JMX/JBOSS interface simply spawns a program using files on disk, it is possible (but not recommended) to bypass the JBOSS/JMX stuff to control the PnF system. In future years, the JBOSS/JMX interface will capture configurations on a run/run basis for logging to databases, so this method will be deprecated at that time, but given the newness of JBOSS, it is provided now.

Running as jboss@sps-fpmaster, all items are located in the:
/usr/local/icecube/pfcommon/ directory.

- Configuration items (see above) can be set in the config/pfserver.config file.
- Control/monitoring of the process is possible via:
  - ./bin/Master -c <config> -l <logconfig> start/stop/status

for example, to start:
./bin/Master -c ./config/pfserver.config -l ./config/Master_log4cxx.config start

Note: the Master_log4cxx.config specifies the location of the name of the current logfile in the ./logs directory.

The sps-fpslave0n machines

The pfclient-main process, which runs on the fpslave0n machines, is

an instance of the IceTray reconstruction framework. As of February 2005, this is an empty reconstruction chain, and the system is running simply to exercise the system and work out bugs. As reconstruction modules become ready, they can be added to the recontruction chain. This system will eventually decide which events are passed via satellite and which are taped.

The pfclient-main processes running on these machines are started automatically at boot time. They should not require any intervention. If the server disconnects or they find an error, they are automatically restarted and attempt to reconnect to the server.

You can check the status of these processes on each fpslave node:

```
ps -e |grep pfclient
```

should give something like:

```
22485 pts/0    00:00:00 pfclient-main
22487 pts/0    00:00:00 pfclient-main
```

Showing two instances of the client software running.

If you suspect trouble with the client software, you can view the log on each node:

```
sps-fpslave0n:/usr/local/icecube/pfcommon/logs/pfclient_log
```

will contain log messages from each running instance.

If you find that reconstruction clients are not running, there is the script started at boot time:

```
sps-fpslave0n:/usr/local/icecube/pfcommon/bin/Client_loop.sh
```

will be able to restart the worker infinite loop. Each node should have two running.

Sensible use of DaqDispatch (Pointers from PfServer)

As of February 2005, DaqDispatch is a little picky about the specification of it's input files via it's JMX/JBOSS console (The only way to configure and control Daq Dispatch). Here are some helpful hints (from Chris Day) on correctly specifying the input file. This should supplement Simon's documentation:

```
The structure of the directory tree is critical. It must be:

<basedir>/<yr>/TestDAQ/<MM>/<dd>/<ProducingServer>_runNNNNNNN_<SteeringF
ile>/

Where
        1) <basedir> is the root of the tree and can be anything. It's
"/data/exp/IceCube" in the example.
        2) <yr> is four digits for the year the data were taken.
        3) TestDAQ must be literally the string "TestDAQ".
```

4) <MM> is two digits for the month number.
5) <dd> is two digits for the day of the month.
6) <ProducingServer> is a string naming the node that produced the data. This is "SPS-DAQ-01" in the example. DaqDispatch does not appear to interpret this, but for parsing reasons, it must exist and obey the rule that no '_' or ' ' is allowed.
7) "_runNNNNNNN_" is the string "_run" followed by seven padded digits for the run number followed by '_'. For parsing reasons, the '_'s are critical.
8) <SteeringFile> is a string naming something called the Steering File. I think this is a TestDAQ-specific term. This is "PedestalPattern-ATWD0-TurnOffPowerManagement" in the example. DaqDispatch does not appear to interpret this, but for parsing reasons, it must exist and obey the rule that no '_' or ' ' is allowed.

This directory contains one and only one data file that must have the name:

<ProducingServer>_runNNNNNNN_<SteeringFile>.<extension>

That is, the same string as the leaf directory node followed by '.' followed by the string given as the second attribute of the dd-control Mbean in its deployment descriptor. This is "events" in the example.

The protocol is for Monolith, or whatever, to write the data file into the directory. After the data file is written and closed, then another, empty file - <semaphore> - is written to the same directory to indicate that the data file is complete. <semaphore> is the name of the signaling file and must be exactly the string given as the third attribute of the dd-control Mbean in its deployment descriptor. This is "events.ready" in the example. Note that "events.ready" is the _complete_ file name for the semaphore; it is not an extension. All data files use the same name for the semaphore file, hence, one data file per directory.

Each data file directory must be writable by jboss from the machine running dd-control, as dd-control will write its own file into it while it is operating.

Dd-control should skip missing runs, days, months, so holidays are possible.

I believe that Dan Wharton is working on making Monolith obey these rules for the files it produces.

Contact information

If you have problems, questions, comments, gripes, or whatever, please contact me:

email: blaufuss@icecube.umd.edu
office: 1+301- 405- 6077
cell: 1+301- 318- 4485

Additional information

Add links to docushare things

Additional information regarding the Offline Software Suite:
http://glacier.lbl.gov/OFFLINE-SOFTWARE