Offline script installation notes

From SangerWiki

These are notes on the install of the scripts for running the backend queuing system that is used by the Pfam website.

Contents

- 1 Background
- 2 Third Party Software
  - 2.1 HMMer
  - 2.2 Intel Compiler
    - 2.2.1 PVM
    - 2.2.2 Multi-threaded Support
    - 2.2.3 WU-BLAST
    - 2.2.4 GeneWise
- 3 Installing the offline code and data files
  - 3.1 Perl prerequisites
  - 3.2 Pfam flatfiles
  - 3.3 CVS checking out of the offline code base
  - 3.4 Setting up the config file
  - 3.5 Final checks
    - 3.5.1 Yet more config
    - 3.5.2 And finally...
- 4 Keeping the list of DAS servers up-to-date
  - 4.1 Configuration
  - 4.2 Running the cron script
- 5 A .bashrc file

Background

The offline compute is designed to be fairly flexible in terms of installation, but it does necessitate some configuration. This means there is slightly more to do here than I would like, but these scripts have been daemonised so that they can be included as part of an init.d config. The offline compute can be run on any machine or sets of machines as long as they can connect to the web user database. At WTSI, the offline compute is completely divorced from the font end. However, in SBC it all runs on the same machine.

The different types of offline compute fall into three classes. For each class, there is a "dequeuer" that connects to the web_user database and pulls out the oldest pending job of that class and runs the jobs.

hmmer
: dedicated hmmer class for running fast single sequence searches, hopefully by running over many cores on a single machine or using a pvm cluster. When hmmer3 comes out, this may become deprecated and run under the fast queue.

fast
: for jobs that typically take less than about 3 secs to run. This currently includes wu-blast searches to

provide Pfam-B matches and hmmalign.

slow
: for jobs that take many seconds to run. Currently, results for these jobs are mailed back to the user. We currently use this queue to run the DNA and batch protein sequence searches. The mechanism for actually running jobs from this queue is very specific to the institute, so you will have to write the snippet of code yourself. At WTSI, we use a LSF to run these jobs, but if you do not have such a queuing system, then the jobs will run on the machine running the dequeuer script.

At WTSI, we run each dequeuer on a separate node, whereas at SBC the dequeuers are all running on the same machine (Erik, you need to buy some more kit). As some of our and the third party software require "flat-files" there will be the need to get these.

Below is hopefully all you need to get the offline compute up and running.

Third Party Software

Typically, I have made a directory offline and installed everything (barring Perl and system wide stuff) in there.

HMMer

Download the hmmer source code from here (http://hmmer.janelia.org/#download) . Now decide how you are going to run hmmer for the single sequence and batch searches. This may require you compiling it more than once.

HMMer is extremely well documented, so go the the HMMer site (http://hmmer.janelia.org) . Below are some quick tips.

```
shell% wget ftp://selab.janelia.org/pub/software/hmmer/CURRENT/hmmer-2.3.2.tar.gz
shell% tar zxvf hmmer-2.3.2.tar.gz
shell% cd hmmer-2.3.2
```

Read the help pages in here. Also, the following command is your friend.

```
shell% ./configure --help
```

Intel Compiler

The intel compiler can produce hmmer binaries that run 2-4x faster than gcc compiled ones. The following compiler options are a good start for PIV Xeons.

```
CFLAGS=-O3 -ipo -axKWP
CC=icc
```

PVM

Setting up the pvm version of hmmer from scratch is not for the faint hearted. The options are: get a geekie sysadmin to do it; download a package (http://packages.qa.debian.org/h/hmmer.html).

Multi-threaded Support

You need to run the HMMer configure script with the flag `--enable-threads`. Even if you have a single

core machine, you need to compile hmmer with this flag unless you are using a pvm version of hmmer. This is because the backend script uses the `--cpu` flag, even if it is set to one. Also, you may want to specify the location of where the executables are stored.

```
shell% ./configure --enable-treads --prefix=/somelocation/offline
```

For more details on installing the HMMer read the manual.

WU-BLAST

Download binaries for WU-blast. This is not straight forward. You need a license for this software. It is free to academic institutes so just go through the procedure of getting the appropriate binaries starting from http://blast.wustl.edu/.

Once you have the binary do the following:

```
shell% mkdir wu-blast
shell% cd wu-blast
```

Then move the gzipped file to this directory, otherwise it will litter all over the place.....

```
shell% gunzip blast2.linux26-i686.tar.gz
```

That should be that working. You will need to set the environment variable WUBLASTMAT

```
csh% setenv WUBLASTMAT somePath/wu-blast/matrix/
or
bash% export WUBLASTMAT=somePath/wu-blast/matrix/
```

Note: At WTSI, to distinguish between ncbi-blast and wu-blast we make a symbolic link to the executable, prefixing blast with "wu", e.g.:

```
shell% ln -s blastp wublastp
shell% ln -s blastx wublastx
```

The latter is essential for DNA searches.

GeneWise

Download the latest version of the wise package from the EBI

```
shell% wget ftp://ftp.ebi.ac.uk/pub/software/unix/wise2/wise2.2.0.tar.gz
shell% tar zxvf wise2.2.0.tar.gz
shell% cd wise2.2.0
```

You can read the README etc. here.

```
shell% cd src
shell% make all
```

Then check it works. This did not work at SBC. Ewan is looking in to it at the moment.

```
shell% make test
```

You need to set the environment variable:

```
csh% setenv WISECONFIGDIR somePath/wise2.2.0/wisecfg
or
bash% export WISECONFIGDIR=somePath/wise2.2.0/wisecfg
```

There was a problem compiling genewise on and i686 architecture as Ewan had bundled so obsolete HMMer libraries in with genewise. After a few emails between Ewan and Sean, the problem was identified and genewise has been patched on this site. If you are having problems with installing genewise, let me (Rob) know and I will get hold of a patched version of genewise.

Please make sure that all the third party executables are on your PATH!!!!!

A typical directory containing all of the third party executables should look like this:

```
shell% ls -lrt
total 12632
   8 drwxrwxr-x  2 pfam pfam    4096 Aug 31 18:24 ./
   8 drwxrwxr-x 11 pfam pfam    4096 Sep  7 18:26 ../
 188 -rwxrwxr-x  1 pfam pfam  180995 Sep  7 18:17 dba*
 168 -rwxrwxr-x  1 pfam pfam  162427 Sep  7 18:17 dnal*
1500 -rwxrwxr-x  1 pfam pfam 1524960 Sep  7 18:17 estwise*
1504 -rwxrwxr-x  1 pfam pfam 1530885 Sep  7 18:17 estwisedb*
1508 -rwxrwxr-x  1 pfam pfam 1534194 Sep  7 18:17 genewise*
1512 -rwxrwxr-x  1 pfam pfam 1538759 Sep  7 18:17 genewisedb*
 300 -rwxrwxr-x  1 pfam pfam  296365 Sep  7 18:17 genomewise*
 592 -rwxrwxr-x  1 pfam pfam  597120 Aug 31 17:47 hmmalign*
 648 -rwxrwxr-x  1 pfam pfam  654968 Aug 31 17:47 hmmbuild*
 604 -rwxrwxr-x  1 pfam pfam  608618 Aug 31 17:47 hmmcalibrate*
 588 -rwxrwxr-x  1 pfam pfam  593660 Aug 31 17:47 hmmconvert*
 592 -rwxrwxr-x  1 pfam pfam  595733 Aug 31 17:47 hmmemit*
 584 -rwxrwxr-x  1 pfam pfam  589314 Aug 31 17:47 hmmfetch*
 588 -rwxrwxr-x  1 pfam pfam  590238 Aug 31 17:47 hmmindex*
 608 -rwxrwxr-x  1 pfam pfam  613903 Aug 31 17:47 hmmpfam*
 608 -rwxrwxr-x  1 pfam pfam  614045 Aug 31 17:47 hmmsearch*
 252 -rwxrwxr-x  1 pfam pfam  248882 Sep  7 18:17 psw*
 260 -rwxrwxr-x  1 pfam pfam  255830 Sep  7 18:17 pswdb*
   4 lrwxrwxrwx  1 pfam pfam      25 Aug 31 18:24 wublastn -> ../../src/wu-blast/blastn
   4 lrwxrwxrwx  1 pfam pfam      25 Aug 31 18:24 wublastp -> ../../src/wu-blast/blastp
   4 lrwxrwxrwx  1 pfam pfam      25 Aug 31 18:24 wublastx -> ../../src/wu-blast/blastx
```

Installing the offline code and data files

Perl prerequisites

This is a little tricky to know as I am always using an existing Perl installation. If you are using the same perl as you use for running the website most of the following will be installed. However, modules that I know you will require include:

- BioPerl
- Mail::Mailer
- Data::Dumper
- IPC::Cmd

- Data::UUID
- DBIx::Class

Pfam flatfiles

Generate a data directory. Now get the following files from the ftp site:

- Pfam-A.fasta.gz
- Pfam-A.scan.dat
- Pfam-A.seed.gz
- Pfam-B.fasta
- Pfam_fs.gz
- Pfam_ls.gz
- Pfam-C.gz

```
shell% mkdir -p /somelocation/offline/datadir
shell% cd /somelocation/offline/datadir
shell% wget ftp://ftp.sanger.ac.uk/pub/rdf/Pfam22.1/<file>
```

Uncompress all of the files, then run:

```
shell% hmmconvert Pfam_ls Pfam_ls.bin
shell% hmmconvert Pfam_fs Pfam_fs.bin
shell% hmmindex Pfam_ls.bin
shell% hmmindex Pfam_fs.bin
shell% xdformat -p Pfam-A.fasta
shell% xdformat -p Pfam-B.fasta
```

You should now have a directory like this:

```
shell% ls
total 4033408
-rw-rw-r-- 1 pfam pfam 446159800 Sep  2 13:17 Pfam-A.fasta
-rw-rw-r-- 1 pfam pfam  85451091 Sep  2 13:18 Pfam-A.seed
-rw-rw-r-- 1 pfam pfam 734881830 Sep  2 13:18 Pfam_ls
-rw-rw-r-- 1 pfam pfam 734937255 Sep  2 13:18 Pfam_fs
-rw-r--r-- 1 pfam pfam  89294886 Sep  2 13:20 Pfam-B.fasta
-rw-r--r-- 1 pfam pfam    563574 Sep  2 13:21 Pfam-A.scan.dat
-rw-rw-r-- 1 pfam pfam 734881347 Sep  2 13:27 Pfam_fs.bin
-rw-rw-r-- 1 pfam pfam 734881830 Sep  2 13:30 Pfam_ls.bin
-rw-rw-r-- 1 pfam pfam    531208 Sep  2 13:35 Pfam_ls.bin.ssi
-rw-rw-r-- 1 pfam pfam    531208 Sep  2 13:39 Pfam_fs.bin.ssi
-rw-rw-r-- 1 pfam pfam  18294624 Sep  2 13:43 Pfam-A.fasta.xpt
-rw-rw-r-- 1 pfam pfam 356371376 Sep  2 13:43 Pfam-A.fasta.xps
-rw-rw-r-- 1 pfam pfam  92963929 Sep  2 13:43 Pfam-A.fasta.xpd
-rw-rw-r-- 1 pfam pfam   5111144 Sep  2 13:43 Pfam-B.fasta.xpt
-rw-rw-r-- 1 pfam pfam  63670652 Sep  2 13:43 Pfam-B.fasta.xps
-rw-rw-r-- 1 pfam pfam  27242724 Sep  2 13:43 Pfam-B.fasta.xpd
-rw-rw-r-- 1 pfam pfam    204779 Sep  7 11:09 Pfam-C
```

CVS checking out of the offline code base

```
export CVSROOT=:pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/pfamweb-public
cvs login
```

You will then be prompted for the password - CVSUSER

```
cvs co PfamBackend
cvs co PfamSchemata
cvs co PfamLib
cvs co PfamConfig
```

Setting up the config file

Then edit the config for your site.

```
shell% cd PfamConfig
shell% vi pfam_backend.conf
```

The next sections will take you through the set-up of the config file for each of the dequeue scripts. Below is the basic config for the hmmer dequeuer.

```
<queue hmmer>
  <jobType>
    hmmer "pfam_scan.pl"
  </jobType>
  pvm         1
  cpus        1
  tmpDir      /tmp
  dataFileDir /path/to/Pfam/data
</queue>
```

- The **jobType** encapsulates which job types this dequeue script deals with and the mapping between the job type and the offline script that is to be run. In this case, we use pfam_scan.pl a Perl wrapper around hmmer. This perl script mainly deals with the munging of results from the hmmer jobs.
- The **pvm** parameter indicates where the hmmer binary is compiled with pvm support. 1 is ture, 0 is false.
- The **cpus** parameter denotes how many cpus hmmer should use. If you are running a pvm version, this should be set to 1. If you are not running a pvm version and the script is running on a mutli-core machine, then the # should be set to the number of cores that you wish to use.
- The **tmpDir** parameter the place where the input sequences are written to. The code should clean up after itself, so requirements should be minimal.
- The **dataFileDir** parameter is the location of the downloaded Pfam flatfiles.

The slow dequeuer is much the same:

```
<queue fast>
  <jobType>
    pfamb wublastp
    align align2seed.pl
  </jobType>
  cpus        2
  tmpDir      /tmp
  dataFileDir /path/to/Pfam/data
</queue>
```

The only difference here is that the fast queue can handle two different job types, pfamb and align. The slow queue config is a little more complicated. This is partly due to the emails of results to users, third party queuing systems. An example is below:

```
<queue slow>
  <jobType>
    batch pfam_scan.pl
    dna   pfamDNASearch.pl
```

```
  </jobType>
  <email>
    From a-pfam-mirroer@site.location
    Bcc  "j.bloggs@site.location, smith@site.location"
  </email>
  <thirdPartyQueue>
    location WTSI
  </thirdPartyQueue>
  pvm         0
  cpus        2
  tmpDir      /tmp
  dataFileDir /data/blastdb/Pfam/data
</queue>
```

The two parameter sections that are different are the **email** and **thirdPartQueue**.

As the results are emailed to the user, you will want to say who the results are **From**. We have used the `pfam-help` email address at Sanger. We have also allowed a **Bcc** to be added. This is probably only need for initial monitoring of the system. You can expand this config further as this parameter hash is passed to the perl module `Mail::Mailer`, so anything allowed in the `Mail::Mailer` header hash is allowed in here.

The **thirdPartyQueue** parameter allows you to switch to an institute specific queuing system. The slow queue has more resource overhead, so you may want to a more sophisticated queuing system. If you want to do this, you are also going to have to alter the `slowDequeue.pl` script. If you set this to your location and this site is not in the list, this queue will operate a first-in-first-out queue.

All of the dequeuing scripts need to access the *web_user* database. There is also the need for the higher privilege account (more on this later).

```
<Model WebUser>
  #WebUser schema settings
  host     webUserDBHost
  port     3306
  user     user_name
  password a_password

  # extra config for the script to update the web_user DB. As we're connecting
  # through raw DBI, we need to know the DB name, and, as we're updating the
  # table and using DELETE, we need the admin account details.
  name       database_name
  admin_user admin_user
  admin_pass another_password
</Model>
```

Final checks

Ensure that the all of the perl scripts are executable. Ensure that you can connect to the *webuser* database.

Yet more config

Yes, there is still a little more configuration to do... There is a config section at the top of each control scripts:

```
#!/bin/sh
#
#You will need to configure this for your system.
#General
export PATH="/sbin:/data/bin:/path/to/PfamBackend/scripts:$PATH"
export PERL5LIB="/path/to/PfamLib:$PERL5LIB"

#Dequeuer config file location
```

```
export PFAMOFFLINE_CONFIG=/pfam/to/pfam_backend.conf

#Stuff required for binaries.
export WUBLASTMAT=/path/to/wu-blast/matrix/
export WISECONFIGDIR=/path/to/wise2/wisecfg/

# Set up of the lsf config - WTSI specific
#. /usr/local/lsf/conf/profile.lsf

#Config for this shell script
DAEMON=/path/to/fastDequeue.pl
NAME=fastDequeue
DESC="Fast queue daemon"
```

The **PATH** and **PERL5LIB** should be set such that all required third-party binaries and all required perl modules are found in the respective paths. The **PFAMOFFLINE_CONFIG** variable should contain the location of the config file described above. Then comes the setting up of the *environment variables* required by wublast and genewise. After this, can come local configuration. Although in the example above, this is commented out, you can see how it could be included.

Make sure that you change the **DAEMON** to the location of your shell script. That is about it!

Note that the daemon perl scripts and this shell script try to write their process ID into a file in `/var/lock`. The user that runs the scripts will need to have write permission for this location. If for whatever reason you can not write to this file, you will need to change the file location for the pid file in both the shell script and in the `daemonise` subroutine found in the `Bio::Pfam::WebServices::PfamQueue` module.

And finally...

Now start the dequeuers on the appropriate machine:

```
shell% nameofDequeuer.sh start
```

There is one shell script for each queue, `fastDequeue.sh`, `hmmerDequeue.sh` and `slowDequeue.sh`. Although these scripts are found in the scripts directory of the PfamBackend CVS module, they can be run from anywhere. These shell scripts make use of `start-stop-daemon (http://www.annodex.net/cgi-bin/man/man2html?start-stop-daemon+8)` (a program that start and stop system daemon programs). The shell script takes one of the following command line parameters for controlling the daemon:

- start
- stop
- restart
- force-reload

For example:

```
shell% fastDequeue.sh start
```

Keeping the list of DAS servers up-to-date

We maintain a list of the currently available DAS servers in a table in the `web_user` database. The table should be updated nightly, in order to keep the list of servers reasonably current. We provide a `cron` script that can do this, which reads its configuration from the same config file as the rest of the back-end scripts.

Configuration

The first section of the cron script configuration sets up the DAS connection parameters:

dasDsn
: the URL of the DAS registry; leave at default setting

dasTo
: the time (in seconds) after which the DAS client gives up on a DAS source

dasProxy
: if the machine running the script needs to connect to the internet via a proxy, give the proxy URL here

For example:

```
dasDsn   "http://das.sanger.ac.uk/das/pfam"
dasTo    100
dasProxy "http://your.local.proxy:3128"
```

If your server has a direct connection to the internet, comment out the `dasProxy` line entirely.

Next in the configuration is a list of default DAS sources. These will be turned on by default in the features tab of the protein page. Give the DAS ID for each source that you want to add to the list of defaults:

```
# the set of default servers
defaultServer DS_109 # uniprot
defaultServer DS_120 # superfamily
defaultServer DS_210 # SMART
defaultServer DS_311 # Pfam Other Features
defaultServer DS_327 # interpro
defaultServer DS_359 # phobius
```

Finally, you can list DAS sources that you want the features viewer to ignore entirely. For example, because the features viewer always uses the Pfam DAS source for the domain graphics row of the view, we don't want it repeated later on. Hence, we add the ID for the Pfam source to the "ignoreServer" list:

```
# the set of servers to ignore entirely
ignoreServer DS_241
ignoreServer DS_241
```

Note that if you have only one server in this list, you will need to add it twice. This is because of the way that the `Config::General` module handles this type of configuration file: if there is a single value "ignoreServer" is given as a scalar, where our code requires an array.

```


# configuration for the script that populates the DAS feature table in the
# web_user DB
<das>

  # DasLite connection parameters
  dasDsn   "http://das.sanger.ac.uk/das/pfam"
  dasTo    100
  dasProxy "http://your.local.proxy:3128"

  # the set of default servers
  defaultServer DS_109 # uniprot
  defaultServer DS_120 # superfamily
  defaultServer DS_210 # SMART
  defaultServer DS_311 # Pfam Other Features
```

```
  defaultServer DS_327 # interpro
  defaultServer DS_359 # phobius

  # the set of servers to ignore entirely
  # (a single server must be included twice, because of the way that
  # Config::General treats single values as a scalar, whereas we want this to
  # be an array at all times)
  ignoreServer DS_241
  ignoreServer DS_241

</das>
```

Running the cron script

The `update_das_sources.pl` script should be run as a nightly `cron` job. This can be done in various ways, depending on the system, but something like the following crontab entry, which runs the script at 4am each day, should work:

```
# Update the list of available DAS sources in the web_user database
0 4 * * *      /path/to/update_das_sources.pl -f /path/to/pfam_backend.conf
```

A `.bashrc` file

This may be of use... A typical bash configuration file looks like this:

```
shell% cat ~/.bashrc
export PERL5LIB=/someLoc/perl/lib/perl5/site_perl:/someLoc/offline/BPModules:/someLoc/offline/PfamLi
export WISECONFIGDIR=/someLoc/offline/src/wise2.2.0/wisecfg
export WUBLASTMAT=/someLoc/offline/src/wu-blast/matrix
export PATH=$PATH:/someLoc/offline/bin:/someLoc/PfamBackend/scripts
export PFAMOFFLINE_CONFIG=/someLoc/offline/PfamConfig/pfam_backend.conf
shell%
```

Retrieved from "http://scratchy.internal.sanger.ac.uk/wiki/index.php/Offline_script_installation_notes"

Categories: PfamGroup | PfamWeb | PfamWebNotes

- This page was last modified 14:20, 12 September 2007.