Running the new transcript variation and variation class pipeline

Prepreamble

NB For this pipeline to work you need to have appropriate entries in your `attrib` table, if you haven't already, you should run the `attrib_entries.sql` file from the `sql/` directory of the ensembl-variation repository on your database, with something like:

```
> mysql --host ens-variation --port=3306 --user=ensadmin --password=XXXXX \
--database will_human_62 < ensembl-variation/sql/attrib_entries.sql
```

Preamble

This pipeline is based on the eHive system developed by the compara team, if you haven't used eHive before you should probably read some of their documentation to make sense of the hive terminology (and tenuous analogies) used here. eHive uses a database to keep track of job dependencies and workers etc. so you'll need to put this database somewhere - if possible it's a good idea to keep this database on a server other than the server hosting the database you're actually working on (henceforth referred to as the *target* database) as the hive uses quite a lot of connections and it also simplifies debugging by keeping the hive and target database queries separate.

You will need to checkout the `ensembl-hive` repository and point to its `modules` directory in your `$PERL5LIB` (as with any ensembl checkout). The compara team maintain a branch of the repository called `stable` which is what I assume you're using below. I have checked that the pipeline works with the `stable` branch as of 6th April 2011. You can checkout the stable branch with a command like:

```
> cvs -d :pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/ensembl checkout -r stable ensembl-hive
```

Configuring the pipeline

All the modules used in the pipeline are stored in the variation CVS repository under `modules/Bio/EnsEMBL/Variation/Pipeline` and the pipeline is configured in a perl module in this directory called `VariationConsequence_conf.pm`, you should edit the various settings in the hash returned by the `default_options` method to suit your configuration. The meanings of these options are hopefully clear from their names and the inline comments (if not ask Graham!) but note that the the `pipeline_db` option defines the eHive pipeline database **not** the target database. You should also create a directory where the hive will store its output and specify this in the `output_dir` option, it can create quite a lot of files so I normally keep this on lustre.

The actual analyses and their inter-dependencies are defined in the `pipeline_analyses` method, which also sets some parameters, such as the `hive_capacity` (number of concurrent workers) for several analyses. Only `transcript_effect` and `set_variation_class` are really parallelisable and can use more than one worker. You can set these values using the `transcript_effect_capacity` and `set_variation_class_capacity` options, either in `default_options` or on the command line when you run `init_pipeline.pl`. For example, I use the default 50 workers for `transcript_effect` for most species, but up it to 300 for human. The bottleneck (when the database is behaving nicely anyway) for these is inserting into the relevant result table, so much larger values may not perform any better (I intend to rejig these so that they will write to temporary files and then `LOAD DATA INFILE` them into the database later on, so this caveat may not always hold).

You will also need to specify a standard ensembl registry file with database connection details for the species you're working on in the `reg_file` parameter in `default_options`. I generally keep this file in the same directory as the `hive_output` directory to keep everything together.

If you aren't working on the Sanger LSF farm you will also probably need to tweak the various resource class options in `default_options`, e.g. `default_lsf_options`, `highmem_lsf_options` etc., or override them on the command line, to specify the correct resource requirements.

Running the pipeline

Once you have set up your config file you can initialise your pipeline with the `init_pipeline.pl` script as follows:

```
> ~/workspace/ensembl-hive/scripts/init_pipeline.pl
Bio::EnsEMBL::Variation::Pipeline::VariationConsequence_conf -hive_db_password XXXXX -species cow
```

The hive lets you specify almost any of the parameters defined in the configuration file on the command line as well, so you can override any

you've set here, but you will need to provide the password for the hive database here (I don't specify it in `default_options` as files in CVS are publicly visible). The pipeline cannot currently run on more than one species at once, so you will also need to specify which species you want to run on here (unless you have defined it in `default_options`) - this can be an alias from your `ensembl.registry` file.

Once you've initialised the pipeline it's probably a good idea to check that the hive database looks sensible (i.e. there should be one `init_transcript_effect` job in the `analysis_job` table with sensible looking parameters in the `input_id` column).

You are now ready to run the pipeline which is done using the `beekeeper.pl` script (geddit), but you should first sync the beekeeper as directed by the output from `init_pipeline.pl` script, but something like:

```
> ~/workspace/ensembl-hive/scripts/beekeeper.pl -url mysql:
//ensadmin:XXXXX@ens-genomics2:3306/grsr_variation_consequence -sync
```

You can then run the pipeline with the `-loop` option as follows:

```
> ~/workspace/ensembl-hive/scripts/beekeeper.pl -url mysql:
//ensadmin:XXXXX@ens-genomics2:3306/grsr_variation_consequence -loop
```

This will run the pipeline in automatic mode (i.e. the hive will notice when dependencies are satisfied and automatically start up new workers as necessary), if you're testing (or just nervous!) you can run it one step at a time with the `-run` option:

```
> ~/workspace/ensembl-hive/scripts/beekeeper.pl -url mysql:
//ensadmin:XXXXX@ens-genomics2:3306/grsr_variation_consequence -run
```

When running the beekeeper in automatic mode for a long running pipeline, I run `beekeeper.pl` in a screen session on a farm login node, it prints out the hive status every minute. You can kill it (but not any running workers) with a `Ctrl-C`.

Error messages from the pipeline will end up in the `job_message` table, and the `analysis_job_file` table links `analysis_jobs` and the files in the `hive_output` directory so you can see anything jobs write on stout and stderr.

NB At the moment, `transcript_effect` is not reentrant in that if individual jobs fail and then get resubmitted by the beekeeper you may end up with duplicate rows in the `transcript_variation` table. If everything runs as expected you should not get any failing jobs, and until I (or someone else!) improves this, if you see any error messages for this analysis you should probably rerun the pipeline (you can try just rerunning the buggy analyses using the `-reset_all_jobs_for_analysis <logic_name>` beekeeper option, but you will have to manually check that you don't get duplicates, or ensure you truncate the `transcript_variation` table before rerunning `transcript_effect`).