Massive parallel BLAST jobs at HPC cluster

Requirements

1. You need small basics of Linux to use HPC cluster based BLAST jobs.
2. You need a valid GWDG account.
3. "batch_blast.script" (and all other here used scripts) are usable **on HPC cluster only**.
4. You must be registered for use of HPC systems.
   If not, send an email to support@gwdg.de (subject: registration for HPC cluster)
5. You need enough disk space of your home directory.
   If you are not sure how many disk space is free, log in to gwdu05.gwdg.de and type "Quota". Look for the difference of used kilobytes and your limit. If you do not have enough disk space send an email to support@gwdg.de (subject: disk quota UNIX cluster) and ask for more.
6. The identically configured HPC frontend computers "gwdu101.gwdg.de" and "gwdu102.gwdg.de" are reachable via Goenet **only**. If necessary use a VPN tunnel to connect to the frontends (for further details see: http://www.gwdg.de/index.php?id=1732).
7. BLAST runs are done in **batch mode only.** Apart from one small test run during workflow (see below) do not run any BLAST searches directly on the frontend in dialog mode!

Workflow

1. Now log in to one of the frontends (s. above no.6 of Requirements) and type:
   ```
   module load NCBI-BLAST
   ```
   This launches environment of NCBI-BLAST *and* several scripts used here.
2. **Create a new empty directory** for your project (use a directory name corresponding to your query sequence project) and copy into this directory your **query** sequence database in **multiple FastA format.** This format must look like:
   ```
   >ABC1234567
   ATGCGTT
   ```
   or
   ```
   >ABC1234567
   RSTKFB
   ```
   In words: each header line must start with **one** '>' character and end with CR (= carriage return) character. All following lines must contain valid sequence symbols only! Do not use **lines** of more than 64,000 characters (columns).
3. Create a separate directory for your BLAST database (if you haven't done this in any projects before). If you have a target database of FastA format, change to your database directory and type in a line like the following one (replace string "*fasta_db*" with the name of your FastA database and replace string "*blast_db*" with the name of your new BLAST database):
   ```
   makeblastdb -in fasta_db -out blast_db
   ```
   Now move away your database of FastA format into a subdirectory (e.g. named "fasta_dbs") because database directory must contain BLAST formatted databases **only**.

4. **Change to your project directory:**
   `cd` *`my_project_dir`*
   and call
   `batch_blast.script`
   from command-line without any further parameter. This will copy a template of **bbs.config** file to your project directory. Open this file with an editor of your choice (emacs, nano, joe or - if your home directory is mounted by SAMBA service by your MSWindows system - with an editor like "editpad lite", "notepad++" etc. which are able to handle Linux ASCII file correctly) and reconfigure variable values if necessary. Please read attentively the comments! You can test your modifications by typing
   `batch_blast.script test`
5. Now you had to "prepare" (for details see below) your query sequences for your BLAST run. This is necessary because "batch_blast.script" does neither use your *complete* query sequence database nor the *single* sequences to start BLAST runs.
   Why?
   In the first case a run will possibly take ***much more*** than 120 CPU hours (= limit of walltime in our batch system) even if we use 64 cores. In the second case hundred thousands of jobs must be handled which should be avoided (if possible). The best strategy will be to split query sequence database into small pieces containing several sequences each and to request for the "necessary cores" only.
   What does "necessary cores" mean?
   A BLAST run needs a little bit more memory as the size of your database because program *and* complete database must be read successively into memory during each alignment process. In batch queue "fat" with option "/scratch file system" (we need this file system!) a maximum of 64 cores with 256 GB RAM is available. The number of requested cores determines the amount of RAM, e.g. you will get 4 GB RAM per requested core (256GB / 64 cores!). It does not make any sense to request for more memory as your jobs need and it is better to use further cores for other jobs. Batch jobs requesting for a small number of cores (and memory) and of relatively short walltime are favoured by the batch system, because these reduces idle time of the nodes (often big parallelised jobs are waiting for enough free nodes). In other words: such small jobs fills-up time frames of idle time of the batch system and this strategy results in a relatively fast job time of your BLAST job in total.
   For estimation of avarage running time of your query sequences make a directory called "000" by the following command
   `mkdir 000`
   create a very small query sequence test file containing some of your query sequences (e.g. the first 30 lines), copy this file into directory "000", and rename it to "0000000.mfa" like follows:
   `head -30` *`my_real_query_sequence_file`***`.mfa > 000/0000000.mfa`**
7. Call "batch_blast.script" from command-line with parameter "test":
   `batch_blast.script test`
   If everything is well configured (if not correct: your variable values!) you will get a test command. Run this test command and set a value for variable "$wt_sec_per_kb" as

described in the output of the script. Furthermore compute the number of lines to get jobs running about 5-15 minutes each. If this is all done delete directory "000" typing:
```
rm -Rf 000
```
8. Call "split_mfa.script" with corresponding file size limit (e.g. 30000 bytes) and define maximum of files per directory (e.g. 4000 files - do not specify more than 8000 files):
```
split_mfa.script test my_query_sequence_file.mfa 30000 4000
```
Script output will give you an estimation of run time and if necessary the strongly advise to run it in batch mode.
If everything seems to be okay run the same command without parameter "test" as a dialog process from command-line
```
split_mfa.script my_query_sequence_file.mfa 30000 4000
```
or **if indicated by script in batch!**
Fragmentation of your query sequence file could take a while (several minutes to several hours!) and results in one or more directories (beginning with "000", ending with "999") containing max. 4000 files each. Have a look on it with:
```
ls -l 000/* | more
```
If "split_mfa.script" has end successfully check again your project with:
```
batch_blast.script test
```
9. **If evrything is okay** type
```
nohup batch_blast.script > my_project.out &
```
Now BLAST-Jobs of first directory (= 000) will be submitted automatically to batch system. If these jobs are all done jobs of next directory (=001) will be submitted and so on. The "nohup" command avoids an automatic kill of your job if you are logged out. Error output and standard output will be redirected into your output file.

Control of batch_blast.script and BLAST job

1. Progress of a Run

There are several possibilities to get information about status of your job:

- Command "`ps x`" shows if your batch_blast.script is running
- Command "`bjobs`" shows status of your running or pending batch jobs ("`bjobs -w`" is more verbose)
- Logfile "batch_blast.script.log" shows progress in job *submission*.
  Command "`tail -f batch_blast.script.log`" sends every new line to screen and can be stopped by CTRL C.
  [**Remark:** For jobs which are not submitted you can change variable wt_sec_per_kb e.g. to avoid crashes of jobs due tu run out of walltime.]
- Logfile "batch_blast.script.todo" shows number of sequence files (= batch jobs) per directory and in total (anyway if they are done or not)
- Logfile "batch_blast.script.runtime" shows runtime of *done* jobs. Command
  ```
  cat batch_blast.script.runtime| wc -l
  ```
  shows number of done jobs. The difference to "batch_blast.script.todo" will inform you about the remaining jobs.

Please wait until "batch_blast.script" has come to an end *by itself* (even if you think all jobs are done) because in some cases it tooks some time to write down the messages into logfiles or to move the result files. Also batch system needs some time to handle jobs too.

2. Break of a Run

If you want to break a run type the following command within project directory:

```
touch ./batch_blast.script.stop
```

Please wait until "batch_blast.script" has come to an end by itself because it tooks some time to write down the last messages into logfiles. In this case all your running batch jobs are **not** stopped.

If you want to stop all your running or pending batch jobs type:

```
bkill $(echo $(bjobs 2> /dev/null | grep -E 'RUN|PEND' | awk '{print $1}'
```

Be aware that batch system could need some time to handle jobs.

3. Performing a new Run with *the same* query sequence database

If you want t o start a new run with the same query sequence (but changing other parameters) type:

```
batch_blast.script reset
```

Parameter "reset" moves all results to a directory named "RUN_*YYYY.MM.DD-hh:mm:ss"* (with time stamp format as given), copies your used "bbs.config" to this directory, and restores your sequence directories (./000 - ./999). So you are able to run your set of query sequences with different parameters (databases, blast flavours, BLAST options ....).This option may need a lot of time due to copying of many files - but don't forget to modify your "bbs.config" file!

4. Performing a new Run with *another* query sequence database

If you want to perform a new run with the same (or other) parameters but with *another* query sequence database you **must** create a complete new project directory. Than copy in query sequence database and "bbs.config" file. After editing "bbs.config" file start at no. 5 of "workflow" (see above).

If you want to use another BLAST database create this if necessary (see above in no. 3 of "workflow") and continue with no. 5 of "workflow".

Future Prospects

- In the near future a resume option to "batch_blast.script" will be added.
- Currently an automatic run triggered by Galaxy or another program (script) is tested.