# Running a job on the cluster: BLAST+

Brian.Moore@sdstate.edu University Networking and Research Computing (UNRC)

To run a job on the cluster, you need to configure a PBS script to deploy your job to a node. The script consists of resource requests followed by the program command. You need to know how to run your program before you configure the script. This might require testing/development on other servers or brief interactive tests on the head node. We assume you have done that. Here we present an example of the actual job submission process.

## blastx example run

We have developed a small example run to get started. The query sequence data and the example PBS script are located on the cluster in the folder `/test1/examples/blast`. The query sequences are in the file `example.fa`; the example PBS script is `blastxexample.pbs` This example has been set up to illustrate basic functionality, but not take too long to run.

Links to the example files in the unrc-docs respository:
blastxexample.pbs
example.fa

```
#!/bin/bash
#  File blastxexample.pbs
#  Data and example from Priya Swaminathan
# ~Nov 2017 brian.Moore@sdstate.edu, padmapriya.swaminathan@sdstate.edu

#PBS -q debug
#PBS -l nodes=1:ppn=4
#PBS -l walltime=1:00:00
#PBS -W x=NACCESSPOLICY:SINGLEJOB

cd $PBS_O_WORKDIR
echo "Running in folder:" pwd
echo "Job ran on node(s):"
cat $PBS_NODEFILE | uniq
echo "Total reserved processors:"
echo $PBS_NP

. ${MODULESHOME}/init/sh
module load bio/blast+

date

# 2016-11-09 Note change in latest nr db location!
time blastx -num_threads $PBS_NP -db /test1/mooreb/blastdb/nr/nr  \
-query example.fa -evalue 1e-10 -word_size 3 -gapopen 11 -gapextend 1  \
 -outfmt 10 -out exampleout.csv

date
```

### Running the example

#### Copy input files to local folder

To run this example, you will have to copy the PBS script file and the input data file to a folder in our working area, for example as below.

```
mooreb@blackjack:~> mkdir blasttest
mooreb@blackjack:~> cd blasttest
mooreb@blackjack:~/blasttest> cp /test1/examples/blast/example.fa .
mooreb@blackjack:~/blasttest> cp /test1/examples/blast/blastxexample.pbs .
mooreb@blackjack:~/blasttest> ls
blastxexample.pbs  example.fa
mooreb@blackjack:~/blasttest>
```

#### Submit the PBS script

```
mooreb@blackjack:~/blasttest> qsub blastxexample.pbs
3009832.bigjack
```

Your job number in the PBS scheduler system is shown when you submit your job. To see the status of your job or the PBS queue, some useful commands are `qstat`, `showq`, `checkjob`.

#### Status of the job and the queue

`qstat` shows a list of all your jobs currently running

```
mooreb@blackjack:~/blasttest> qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
3009832.bigjack            ...txexample.pbs mooreb                 0 R debug
mooreb@blackjack:~/blasttest>
```

`showq` shows the jobs in all the queues

```
mooreb@blackjack:~/blasttest> showq

active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME

3009832              mooreb    Running    12    00:56:45  Thu Nov  2 13:31:29
3009803             opdahll    Running    12  2:00:52:40  Tue Oct 31 10:27:24
3009797            smitheei    Running   144  2:05:02:38  Sun Oct 29 12:37:22
3009080            software    Running    16 11:02:37:52  Wed Oct 11 08:12:36

4 active jobs           184 of 564 processors in use by local jobs (32.62%)
                          16 of 47 nodes active      (34.04%)

eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME


0 eligible jobs

blocked jobs-----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME
```

```
0 blocked jobs

Total jobs:  4
```

It's important to remember that the number of available processors shown in `showq` (564 in the example above) is the total of *all* nodes in all queues. If we want to see only a specific queue we can use a qualifier.

```
mooreb@blackjack:~/blasttest> showq -w class=debug

active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME

3009832              mooreb    Running    12    00:54:16  Thu Nov  2 13:31:29

1 active job             12 of 564 processors in use by local jobs (2.13%)
                          16 of 47 nodes active      (34.04%)

[...]
```

To find more detailed information about a single job, use `checkjob` followed by the job number.

```
mooreb@blackjack:~/blasttest> checkjob 3009832
job 3009832

AName: blastxexample.pbs
[...]

Allocated Nodes:
[node002:4]

[...]
```

Some of the output of `checkjob` has been omitted above; one of the useful bits of information you can find is what node your job is running on.

### When the job is done

When the job is done, you will see PBS output and error files for the job number, and also output files produced by our command.

```
mooreb@blackjack:~/blasttest> ll
total 48
-rw-r--r-- 1 mooreb mooreb   683 Nov  2 13:29 blastxexample.pbs
-rw------- 1 mooreb mooreb   133 Nov  2 13:25 blastxexample.pbs.e3009832
-rw------- 1 mooreb mooreb   138 Nov  2 13:25 blastxexample.pbs.o3009832
-rw-r--r-- 1 mooreb mooreb  1238 Nov  2 13:29 example.fa
-rw------- 1 mooreb mooreb 31953 Nov  2 13:39 exampleout.csv
mooreb@blackjack:~/blasttest>
```

```
mooreb@blackjack:~/blasttest> more blastxexample.pbs.o3009832
Running in folder: pwd
Job ran on node(s):
node002
Total reserved processors:
4
```

```
Thu Nov  2 13:17:50 CDT 2017
Thu Nov  2 13:25:21 CDT 2017
mooreb@blackjack:~/blasttest>

mooreb@blackjack:~/blasttest> head exampleout.csv
contig_02,XP_010680960,65.909,44,15,0,116,247,119,162,3.00e-12,68.6
contig_03,ORY90882,76.744,86,20,0,283,26,284,369,5.93e-39,142
contig_03,AFV09324,75.581,86,21,0,283,26,283,368,2.72e-38,140
contig_03,AFV09326,75.581,86,21,0,283,26,283,368,3.03e-38,140
contig_03,XP_016274581,74.419,86,22,0,283,26,283,368,1.26e-37,138
contig_03,KDE05820,72.222,90,25,0,283,14,284,373,5.67e-37,137
contig_03,XP_018272825,72.093,86,24,0,283,26,284,369,9.64e-37,136
contig_03,KWU42238,73.256,86,23,0,283,26,287,372,3.02e-36,135
contig_03,CEQ39119,73.256,86,23,0,283,26,284,369,5.63e-36,134
contig_03,XP_013243034,63.736,91,33,0,283,11,288,378,2.62e-31,122
mooreb@blackjack:~/blasttest>
```

## BLAST+ program information

BLAST (Basic Local Alignment Search Tool) is a program to compare nucleotide or protein sequences to databases and find the significance of the matches. To run the program, one needs the program itself, query sequences and access to a blast database (either local, or accessible through the network) to run against.

For general information about the program itself and how it works, see, for example http://blast.ncbi.nlm.nih.gov. On that page, one can scroll down to choose from various options, for example nucleotide blast, and it will open a web portal where you can run query sequences on NCBI's server. For documentation on the program, the command line user manual (http://www.ncbi.nlm.nih.gov/books/NBK279690/) is a good place to start, for reference on the program itself and also on command line use, which is how the program is used on the cluster. To obtain a copy of the program itself (again, command line version) see ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/. This is what we have installed on the cluster in a shared location (the binary version), but you can also download it for use on your own systems or in your home folder. The program itself is fairly easy to download and use. A much larger concern is getting the database you want to run against.

The program discussed here is more properly called BLAST+, to distinguish it from an earlier version (still in use in some applications) with the Blast name; this older program is now referred to on the NCBI site as "legacy" Blast. This documentation treats only Blast+.

## Blast databases

Pre-formatted Blast databases can be downloaded from the NCBI site. Currently we have two databases for general use, located on the cluster at `/test1/mooreb/blastdb`. The nr database has non-redundant protein sequences; nt is a partially non-redundant nucleotide sequence database. These databases are updated on demand. So far, each time we have updated, we have saved the old version in offline storage so they can be brought back if necessary to repeat a query against an older version. To see when they were updated, look at the oldest date stamp in the folder; a command such as `ls -altr` will list all files with date stamps, showing oldest last.

The nr database currently occupies about 153 GB on disk; nt about 72 GB. For this reason we discourage users from frequently downloading Blast databases to their own local storage. Contact us first. If we can put the database in a central location, it will save on overall disk space.

Blast databases can also be created from unformatted genomic data, nucleotide or protein sequences. This topic is not covered here; contact one of us if you are interested in getting more information.

## Accessing the BLAST+ program on the cluster; options

On the cluster head node (`blackjack`) and worker nodes, we have implmented an environment module system. To see the menu of available modules, type `module avail` at the command line; `module list` is used to see currently loaded modules. Once the module is loaded, the executable commands for BLAST+ become available. There are several commands (see the manual for more detail), for example `blastn` is for searching a nucleotide database using a nucleotide query; `blastx` is for searching a protein database using a translated nucleotide query.

Useful information can be obtained from the help options within the program itself. For brief help, use the `-h` option; for detailed help, use `-help`. The output from `-help` is quite long. To view the help options with a pager, use (for example) `blastx -help | less`. Below is shown an excerpt of the output, from the section about output format options.

```
-outfmt <String>
   alignment view options:
     0 = Pairwise,
     1 = Query-anchored showing identities,
     2 = Query-anchored no identities,
     3 = Flat query-anchored showing identities,
     4 = Flat query-anchored no identities,
     5 = BLAST XML,
     6 = Tabular,
     7 = Tabular with comment lines,
     8 = Seqalign (Text ASN.1),
     9 = Seqalign (Binary ASN.1),
    10 = Comma-separated values,
    11 = BLAST archive (ASN.1),
    12 = Seqalign (JSON),
    13 = Multiple-file BLAST JSON,
    14 = Multiple-file BLAST XML2,
    15 = Single-file BLAST JSON,
    16 = Single-file BLAST XML2
```

### interactive run

Before running the PBS script, if you want, you can run interactively. This is not required, but it illustrates a point that when you are developing a PBS script, it can be very useful to know how the program behaves when from from the command line for a small data set. If you are doing this on your own, check the load on `blackjack` (using the `top` command) and it is OK to run the example interactively on blackjack itself; use no more than `-num_threads 4`.

For the tutorial session, if a few people need to run at the same time, it is better to initiate an interactive PBS session. Example console output to illustrate this follow.

```
mooreb@blackjack:~/blasttest> qsub -I -l nodes=1:ppn=4 -l walltime=1:00:00
qsub: waiting for job 3009833.bigjack to start
qsub: job 3009833.bigjack ready

mooreb@node064:~> module load bio/blast+
mooreb@node064:~> cd blasttest
mooreb@node064:~/blasttest> time blastx -num_threads $PBS_NP -db /test1/mooreb/blastdb/nr/nr -query exampl
blastx: /usr/lib64/libidn.so.11: no version information available (required by blastx)
```

```
real    7m40.465s
user    28m20.670s
sys     0m12.301s
mooreb@node064:~/blasttest>
```

(The warning about libidn.so.11 can be ignored.)

Note that in typing the command, you do not need to use the bash command continuation (\) approach; you can also write it out just as one long command; the output will wrap to the next line. Then hit return to execute.

Once you type the command, the terminal will be occupied for how long it takes to finish. If you want to recover the terminal prompt while you wait, you can use `ctrl-Z` to suspend the command, then `bg` to put it in background. The output of the time command will still be written to the terminal when it finishes. During the run you can use `top` or other commands to monitor the job. Since we only asked for 4 processors, there might be other jobs on your node.

Another thing to note in this case is we chose the csv format for the output report. You can use WinSCP or another file transfer program to copy the file to your local system and view the results quickly in a spreadsheet. In the PBS example below, we also show the XML report option, which can be useful when the results are to be read by another program. Another option is to use the BLAST archive format (`-outfmt 11`) which can be used with the BLAST+ `blast_formatter` command to create csv or XML or other reports from the archive after the run is finished.