

Matlab 2013-2014

Introduction

This document relates to the old versions R2013 and R2014. For MATLAB 2015, please use this documentation instead.

Matlab is available in the latest stable version. There are always two variants of the release:

- Non commercial or so called EDU variant, which can be used for common research and educational purposes.
- Commercial or so called COM variant, which can be used also for commercial activities. The licenses for commercial variant are much more expensive, so usually the commercial variant has only subset of features compared to the EDU available.

To load the latest version of Matlab load the module

```
$ module load matlab
```

By default the EDU variant is marked as default. If you need other version or variant, load the particular version. To obtain the list of available versions use

```
$ module avail matlab
```

If you need to use the Matlab GUI to prepare your Matlab programs, you can use Matlab directly on the login nodes. But for all computations use Matlab on the compute nodes via PBS Pro scheduler.

If you require the Matlab GUI, please follow the general informations about running graphical applications.

Matlab GUI is quite slow using the X forwarding built in the PBS (qsub -X), so using X11 display redirection either via SSH or directly by xauth (please see the “GUI Applications on Compute Nodes over VNC” part here) is recommended.

To run Matlab with GUI, use

```
$ matlab
```

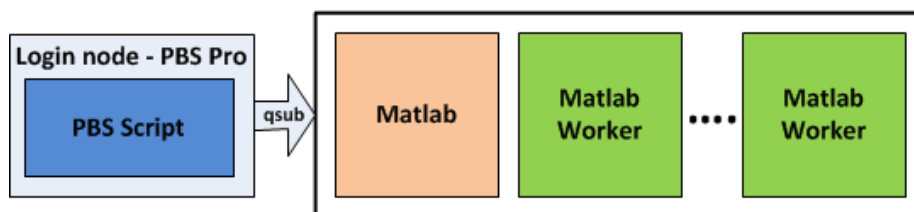
To run Matlab in text mode, without the Matlab Desktop GUI environment, use

```
$ matlab -nodesktop -nosplash
```

plots, images, etc... will be still available.

Running parallel Matlab using Distributed Computing Toolbox / Engine

Recommended parallel mode for running parallel Matlab on Anselm is MPIEXEC mode. In this mode user allocates resources through PBS prior to starting Matlab. Once resources are granted the main Matlab instance is started on the first compute node assigned to job by PBS and workers are started on all remaining nodes. User can use both interactive and non-interactive PBS sessions. This mode guarantees that the data processing is not performed on login nodes, but all processing is on compute nodes.



For the performance reasons Matlab should use system MPI. On Anselm the supported MPI implementation for Matlab is Intel MPI. To switch to system MPI user has to override default Matlab setting by creating new configuration file in its home directory. The path and file name has to be exactly the same as in the following listing:

```
$ vim ~/matlab/mpiLibConf.m

function [lib, extras] = mpiLibConf
%MATLAB MPI Library overloading for Infiniband Networks

mpich = '/opt/intel/impi/4.1.1.036/lib64/';

disp('Using Intel MPI 4.1.1.036 over Infiniband')

lib = strcat(mpich, 'libmpich.so');
mpl = strcat(mpich, 'libmpl.so');
opa = strcat(mpich, 'libopa.so');

extras = {};
```

System MPI library allows Matlab to communicate through 40Gbps Infiniband QDR interconnect instead of slower 1Gb ethernet network.

Please note: The path to MPI library in “mpiLibConf.m” has to match with version of loaded Intel MPI module. In this example the version 4.1.1.036 of Intel MPI is used by Matlab and therefore module impi/4.1.1.036 has to be loaded prior to starting Matlab.

Parallel Matlab interactive session

Once this file is in place, user can request resources from PBS. Following example shows how to start interactive session with support for Matlab GUI. For more information about GUI based applications on Anselm see this page.

```
$ xhost +
$ qsub -I -v DISPLAY=$(uname -n):$(echo $DISPLAY | cut -d ':' -f 2) -A NONE-0-0 -q qexp -l select=2:ncpus=16:mpiprocs=16:ompthreads=1
-l feature__matlab__MATLAB=1
```

This qsub command example shows how to run Matlab with 32 workers in following configuration: 2 nodes (use all 16 cores per node) and 16 workers = mpirocs per node (-l select=2:ncpus=16:mpiprocs=16). If user requires to run smaller number of workers per node then the “mpiprocs” parameter has to be changed.

The second part of the command shows how to request all necessary licenses. In this case 1 Matlab-EDU license and 32 Distributed Computing Engines licenses.

Once the access to compute nodes is granted by PBS, user can load following modules and start Matlab:

```
cn79$ module load matlab/R2013a-EDU
cn79$ module load impi/4.1.1.036
cn79$ matlab &
```

Parallel Matlab batch job

To run matlab in batch mode, write an matlab script, then write a bash jobscript and execute via the qsub command. By default, matlab will execute one matlab worker instance per allocated core.

```
#!/bin/bash
#PBS -A PROJECT ID
#PBS -q qprod
#PBS -l select=2:ncpus=16:mpiprocs=16:ompthreads=1

# change to shared scratch directory
SCR=/scratch/$USER/$PBS_JOBID
mkdir -p $SCR ; cd $SCR || exit

# copy input file to scratch
cp $PBS_O_WORKDIR/matlabcode.m .

# load modules
module load matlab/R2013a-EDU
module load impi/4.1.1.036
```

```
# execute the calculation
matlab -nodisplay -r matlabcode > output.out
```

```
# copy output file to home
cp output.out $PBS_O_WORKDIR/.
```

This script may be submitted directly to the PBS workload manager via the qsub command. The inputs and matlab script are in matlabcode.m file, outputs in output.out file. Note the missing .m extension in the matlab -r matlabcodefile call, **the .m must not be included**. Note that the **shared /scratch must be used**. Further, it is **important to include quit** statement at the end of the matlabcode.m script.

Submit the jobscript using qsub

```
$ qsub ./jobscript
```

Parallel Matlab program example

The last part of the configuration is done directly in the user Matlab script before Distributed Computing Toolbox is started.

```
sched = findResource('scheduler', 'type', 'mpiexec');
set(sched, 'MpiexecFileName', '/apps/intel/impi/4.1.1/bin/mpirun');
set(sched, 'EnvironmentSetMethod', 'setenv');
```

This script creates scheduler object “sched” of type “mpiexec” that starts workers using mpirun tool. To use correct version of mpirun, the second line specifies the path to correct version of system Intel MPI library.

Please note: Every Matlab script that needs to initialize/use matlabpool has to contain these three lines prior to calling matlabpool(sched, ...) function.

The last step is to start matlabpool with “sched” object and correct number of workers. In this case qsub asked for total number of 32 cores, therefore the number of workers is also set to 32.

```
matlabpool(sched,32);
```

```
... parallel code ...
```

```
matlabpool close
```

The complete example showing how to use Distributed Computing Toolbox is show here.

```
sched = findResource('scheduler', 'type', 'mpiexec');
set(sched, 'MpiexecFileName', '/apps/intel/impi/4.1.1/bin/mpirun')
```

```

set(sched, 'EnvironmentSetMethod', 'setenv')
set(sched, 'SubmitArguments', '')
sched

matlabpool(sched,32);

n=2000;

W = rand(n,n);
W = distributed(W);
x = (1:n)';
x = distributed(x);
spmd
[~, name] = system('hostname')

    T = W*x; % Calculation performed on labs, in parallel.
           % T and W are both codistributed arrays here.
end
T;
whos           % T and W are both distributed arrays here.

matlabpool close
quit

```

You can copy and paste the example in a .m file and execute. Note that the matlabpool size should correspond to **total number of cores** available on allocated nodes.

Non-interactive Session and Licenses

If you want to run batch jobs with Matlab, be sure to request appropriate license features with the PBS Pro scheduler, at least the " -l ___feature___matlab___MATLAB=1" for EDU variant of Matlab. More information about how to check the license features states and how to request them with PBS Pro, please look [here](#).

In case of non-interactive session please read the following information on how to modify the qsub command to test for available licenses prior getting the resource allocation.

Matlab Distributed Computing Engines start up time

Starting Matlab workers is an expensive process that requires certain amount of time. For your information please see the following table:

compute nodes	number of workers	start-up time[s]				
534	4	64	333	2	32	210
16	256	1008	8	128		