This personal note describes how to run Matlab on a node of Grid’5000. It is a work in progress.

An account on a machine allowed to access Matlab license server at Université de Lorraine is needed to run Matlab on Grid’5000, in addition to a Grid’5000 account.

Contents

0 Summary: running Matlab with an SSH tunnel 2

1 Creating a Grid’5000 account 3

2 Connecting to Grid’5000 3
   2.1 Connecting to front-end .............................. 3
   2.2 Connecting to a resource .......................... 3

3 Connecting to UL license server 5
   3.1 Setting up an SSH tunnel ........................... 5
   3.2 Alternatively: Setting up UL VPN .................. 5

4 Running Matlab 6

5 To do 7

6 Benchmarks 7
   6.1 paralleldemo_gpu_bench ........................... 7
   6.2 paralleldemo_parfor_bench ........................ 7
0 Summary: running Matlab with an SSH tunnel

1. Connect to Grid’5000 front-end:

   `localhost:~$ ssh -i path_to_G5000pkey login@access.nancy.grid5000.fr`

   where `path_to_G5000pkey` is the path to your private (id_rsa) key for Grid’5000, and `login` is your Grid’5000 login.

2. From Grid’5000 frontend, connect to a resource with `oarsub`, for instance:

   `login@fnancy:~$ oarsub -t production -p "cluster='talc'" \
   -l nodes=1,walltime=2 -t allow_classic_ssh -I`

3. Open an SSH tunnel from the resource to `bastionssh` in order to forward the ports used by the license server:

   `login@talc-1:~$ ssh -f -N -L 27000:flexlm1.univ-lorraine.fr:27000 \
   -L 27001:flexlm1.univ-lorraine.fr:27001 \
   -i path_to_bastionpkey lorialogin@bastionssh.loria.fr`

   where `path_to_bastionpkey` is the path to your private (id_rsa) key for `bastionssh`, which must be copied to your Grid’5000 home directory (do not forget to use a strong passphrase), and `lorialogin` is your LORIA login to access `bastionssh`.

4. Run Matlab in console mode, using the local ports (forwarded to `bastionssh`) to contact the license server:

   `login@talc-1:~$ matlab -c 27000@localhost,27001@localhost -nodesktop -nosplash`

   where the Matlab executable is in your `path`; it can be found in the following directory: `/talc/magrit/MathWorks/linux-R2016A/bin/`.

   Of course, it is more convenient to run a script with the two commands (SSH tunnel and Matlab) on the reserved node.

   The following sections give more details on these steps.
1 Creating a Grid’5000 account

Follow the instructions:
https://www.grid5000.fr/mediawiki/index.php/Grid5000:Get_an_account

The following URL may be useful:
in particular Getting started:
https://www.grid5000.fr/mediawiki/index.php/Getting_Started
and URLs related to Nancy:

2 Connecting to Grid’5000

2.1 Connecting to front-end

Connect to Nancy front-end:
localhost:> ssh login@access.nancy.grid5000.fr
where login is your Grid’5000 login.
Add the option -i ~/path/.ssh/id_rsa to ssh if your id_rsa file is not in the standard
~/.ssh/ directory.
Files can be moved from local host to Grid’5000 home directory with scp.

2.2 Connecting to a resource

2.2.1 Resource reservation

Connect to a production node from the front-end:
login@fnancy:~$ oarsub -t production -p "cluster='graphique'" \
-l nodes=1,walltime=2 -t allow_classic_ssh -I
Here nodes=1 means that one full node is reserved (with all CPUs and GPUs), walltime=2
set the walltime to two hours (default walltime is one hour; after the limit expires, the job is
terminated) and cluster asks for a node on a specific production cluster (here graphique,
can be changed to talc, grmani, or graouly), and allow_classic_ssh allows the user
to connect to the node with SSH (required for SSH port forwarding, not needed when using
VPN).
The user is connected as soon as a resource is available. Note the value of OAR_JOB_ID.
To reserve a specific node:
We assume that the graphique-1 node has been reserved in this step.

It is possible to monitor the node in the following ways:

- To get information on the GPUs (if available; some nodes have two GPUs):
  
  ```
  login@graphique-1:~$ nvidia-smi
  ```

- To get information on the CPUs:
  
  ```
  login@graphique-1:~$ cat /proc/cpuinfo
  ```

- To monitor processes, memory, and CPU usage:
  
  ```
  login@graphique-1:~$ htop
  ```

2.2.2 Connecting to a previously reserved node

It is possible to connect to a previously reserved node, for instance to monitor the execution of a running job. It is required to

First, connect to the front-end with SSH from another local terminal.

From the front-end:

```
login@fnancy:~$ OAR_JOB_ID=1189451 oarsh graphique-1
```

to connect to the node graphique-1, assuming that the job ID is 1189451.

Note that

```
login@fnancy:~$ oarstat -u login
```

lists resources reserved by the user login and gives the job ID.

Alternatively, connect to a previously reserved node (job ID 1189451) with:

```
login@fnancy:~$ oarsub -C 1189451
```

Screen may also be used:

3 Connecting to UL license server

3.1 Setting up an SSH tunnel

The license.dat file (e.g., /talc/magrit/MathWorks/linux-R2016A/license.dat) indicates the name of the three flexlm servers at Université de Lorraine, with one of the ports to contact.

We establish an SSH connection from the current Grid’5000 node to bastionssh.loria.fr forwarding port localhost:27000 to flexlm1.univ-lorraine.fr:27000, and also port localhost:27001 to flexlm1.univ-lorraine.fr:27001, with:

login@graphique-1:~$ ssh -f -N -L 27000:flexlm1.univ-lorraine.fr:27000 \
-L 27001:flexlm1.univ-lorraine.fr:27001 \
-i .ssh/id_rsa_b lorialogin@bastionssh.loria.fr

where lorialogin is the user’s login at LORIA. It is assumed here that the private key for bastionssh is .ssh/id_rsa_b. Do not forget to define a strong passphrase!

SSH options: -f requests SSH to run in background and -N option not to execute any remote command.

To close the SSH process in background: retrieve its PID with ps -fu login, then kill -9 PID.

Do not forget -t allow_classic_ssh when connecting with oarsub.

If flexlm1 is down, change it to flexlm2 or flexlm3.

More information on how to connect to bastionssh with SSH: http://infodoc.loria.fr/display/SUPPORT/SSH

3.2 Alternatively: Setting up UL VPN

Openconnect allows the user to use UL VPN in order to reach Matlab license server.

Documentation of Openconnect:

Documentation of UL VPN:
then “Documentation pour les informaticiens et le personnel” (UL login required).

First, install openconnect package:

login@graphique-1:~$ sudo-g5k apt-get install openconnect

Run UL VPN:
login@graphique-1:~$ sudo openconnect -b vpn.lothaire.net

with:

GROUP: Universite-de-Lorraine
Username: loginUL@ul
Password: passwordUL

where loginUL is your UL login (do not forget to add @ul), and passwordUL is your UL password.

The following error message appears in the terminal every 90 seconds:

DTLS handshake failed: Resource temporarily unavailable, try again.

This does not seem to prevent Matlab from accessing the license server. A workaround is to redirect stderr when launching openconnect:

login@graphique-1:~$ sudo openconnect -b --authgroup=Universite-de-Lorraine --user=loginUL@ul vpn.lothaire.net 2> /dev/null

then type passwordUL when the script stops (the login message is also redirected).

Remark: I was not able to use vpnc:
https://doc.ubuntu-fr.org/vpnc
(“no response from target” error message)

4 Running Matlab

A copy of /local/logiciels/MathWorks/ (LORIA NFS) is available on the talc-data NFS server at /talc/magrit/MathWorks, for example.

Run Matlab in console mode:

login@graphique-1:~$ alias matlab='/talc/magrit/MathWorks/linux-R2016A/bin/matlab'

When using ssh port forwarding:

login@graphique-1:~$ matlab -c 27000@localhost,27001@localhost -nodesktop -nosplash

Alternatively, when using UL VPN:

login@graphique-1:~$ matlab -nodesktop -nosplash

Instead of using an alias, add:
PATH="/talc/magrit/MathWorks/linux-R2016A/bin:$PATH"

in the .profile file

Useful Matlab commands: `parpool` to create a parallel pool of workers (`parpool('local',n)` to set the number of workers to n), `gpuDevice` to get information on the GPU (if available) and `gpuDeviceCount` to get the number of available GPUs.

Matlab provides useful benchmarks, for instance `paralleldemo_parfor_bench` and `paralleldemo_gpu_backslash`.

5 To do

- Customize the software environment with `kadeploy`? Not really needed with SSH port forwarding.
- Test scripts to run Matlab on several nodes since Matlab Distributed Computing Server is not available. Problem: the user has to enter his/her passphrase to connect to `bastionssh`.
- For an unknown reason, `parpool` seems occasionally to crash Matlab.

6 Benchmarks

6.1 `paralleldemo_gpu_bench`

Figure 6.1 shows the output of `paralleldemo_gpu_bench`, for the system configurations given in the following table.

```
<table>
<thead>
<tr>
<th>Config. Nr.</th>
<th>CPU</th>
<th>GPU</th>
<th>Grid’5000 node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 Intel Xeon E5-1650 v4 @ 3.60GHz, 6 cores</td>
<td>Titan X</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2 Intel Xeon E5-2603 v3 @ 1.60GHz, 6 cores/CPU,</td>
<td>Tesla K40</td>
<td>grimani</td>
</tr>
<tr>
<td>3</td>
<td>2 Intel Xeon E5-2620 v3 @ 2.40GHz, 6 cores/CPU,</td>
<td>Titan Black</td>
<td>graphique-1</td>
</tr>
<tr>
<td>4</td>
<td>2 Intel Xeon E5-2620 v3 @ 2.40GHz, 6 cores/CPU,</td>
<td>GTX-980</td>
<td>graphique (node ≠ 1)</td>
</tr>
</tbody>
</table>
```

6.2 `paralleldemo_parfor_bench`

The results of `paralleldemo_parfor_bench` are given in the following table.

```
<table>
<thead>
<tr>
<th>Grid’5000 node</th>
<th>run time using a sequential for-loop</th>
<th>median speedup</th>
<th>efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>talc</td>
<td>1.04</td>
<td>6.94</td>
<td>0.87</td>
</tr>
<tr>
<td>graouly</td>
<td>0.48</td>
<td>9.53</td>
<td>0.79</td>
</tr>
<tr>
<td>grimani</td>
<td>0.77</td>
<td>10.07</td>
<td>0.84</td>
</tr>
<tr>
<td>graphique</td>
<td>0.47</td>
<td>8.78</td>
<td>0.73</td>
</tr>
</tbody>
</table>
```
Figure 1: Results of paralleldemo_gpu_backslash.