Running MATLAB computations on the Trantor cluster

MATLAB Parallel Server is available on the Trantor cluster, allowing to execute long-running CPU and/or GPU intensive computations on high-performance compute nodes. Furthermore, **jobs submission and results retrieval is performed directly from the MATLAB instance running on your personal workstation**.

Service availability

In accordance with the terms of the license subscribed by Scuola Normale Superiore, the use of this service is **restricted to SNS students and research staff only**. In addition, access to the service **must be explicitly requested** by writing to hpcstaff@sns.it.

At this time, only the following queues support the execution of MATLAB jobs:

- q07helicon
- q14helicon
- q07daneel
- q14daneel

Please use the '*daneel*' queues only if your computation requires GPU hardware.
You can find additional information on the characteristics of these queues and their associated compute nodes on our Website:
https://hpccenter.sns.it/page/wiki/pages/Resources_available_on_Trantor.html
https://hpccenter.sns.it/page/wiki/pages/Submitting_Inspecting_and_Cancelling_PBS_Jobs.html#Resources_and_queues

Finally remember that the Trantor cluster can be accessed only **within the SNS internal network**. When working offsite, you can establish a VPN connection to the SNS network (see https://ict.sns.it/en/service/access/vpn for details).

License agreement

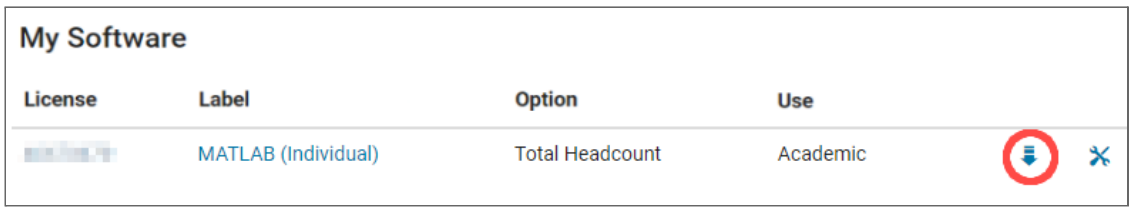
MathWorks products can be used for **academic purposes only** and their use is subject to the **MathWorks, Inc. Software License Agreement and Program Offering Guide**. You can read both documents by selecting the *Help → Terms of Use* menu entry from the MATLAB toolbar, or by opening the file `license_agreement.txt` stored in your MATLAB installation folder. **Carefully read the terms and conditions before using the software**.

Installing MATLAB on your workstation

1. Visit the SNS MathWorks Portal, then click on the "Sign in to get started" button.
2. Under the "Campus-Wide Access" section, click the "Get MATLAB access now" button.

3. Sign in (if you already have a MathWorks account) or create a new account using your SNS email address.

4. In your "*My Account*" page, you should see a "*My Software*" table. Click on the download button on the right side of the row labeled as "*MATLAB (Individual)*". You will be redirected to the "Download" page.



5. Select the **R2022a release**, then download the MATLAB installer for your Operating System. Once downloaded, you can disconnect from the MathWorks portal.

6. Run the installer and follow the wizard to install MATLAB on your workstation, signing in with your MathWorks account when required.

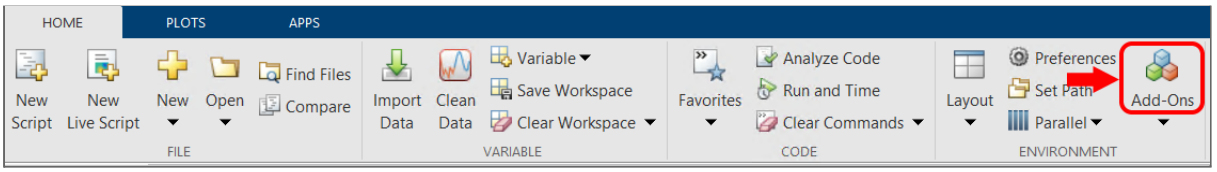
Installing the Parallel Computing Toolbox and configuring the cluster integration

The ***Parallel Computing Toolbox*** plugin provides high-level constructs to parallelize MATLAB programs, allowing to exploit the full processing power of multi-core CPUs and GPUs (refer to the official documentation to learn more). In addition, is it possible to configure the *Parallel Computing Toolbox* to submit computations on a HPC cluster and retrieve the results directly from the MATLAB interface.

Follow these steps to install and configure *Parallel Computing Toolbox* in MATLAB:

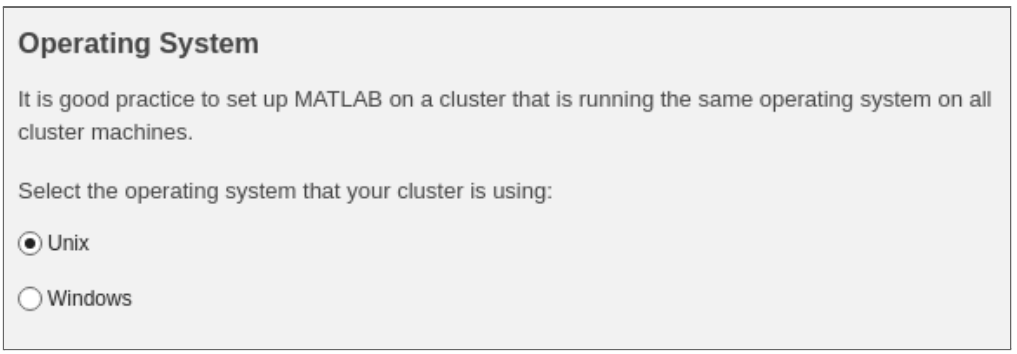
1. In the toolbar, select the *HOME* tab and click on the *Add-Ons* button.
   Then, from the *Add-On Explorer* window, install the following plugins:

   1.1. **Parallel Computing Toolbox**
   1.2. **Parallel Computing Toolbox plugin for MATLAB Parallel Server with PBS**

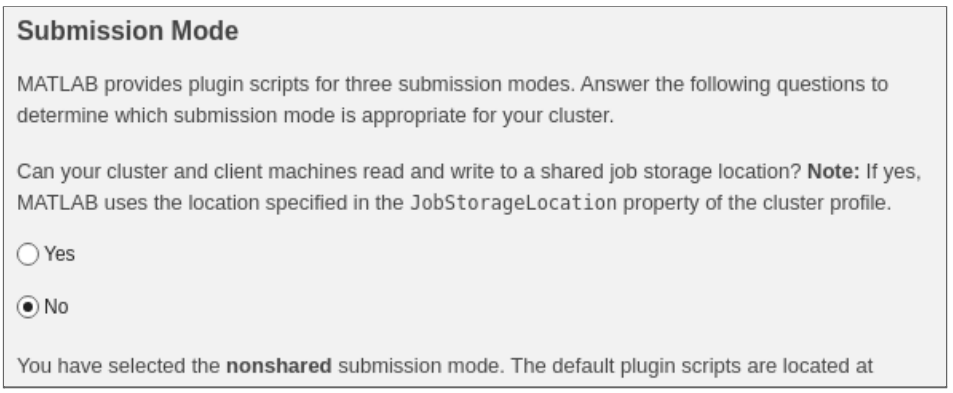


2. After installing "*Parallel Computing Toolbox plugin for MATLAB Parallel Server with PBS*", a configuration wizard will start. Follow the guided procedure, answering as follows:

   2.1. Q: *Select the operating system that your cluster is using*
   A: **Unix**



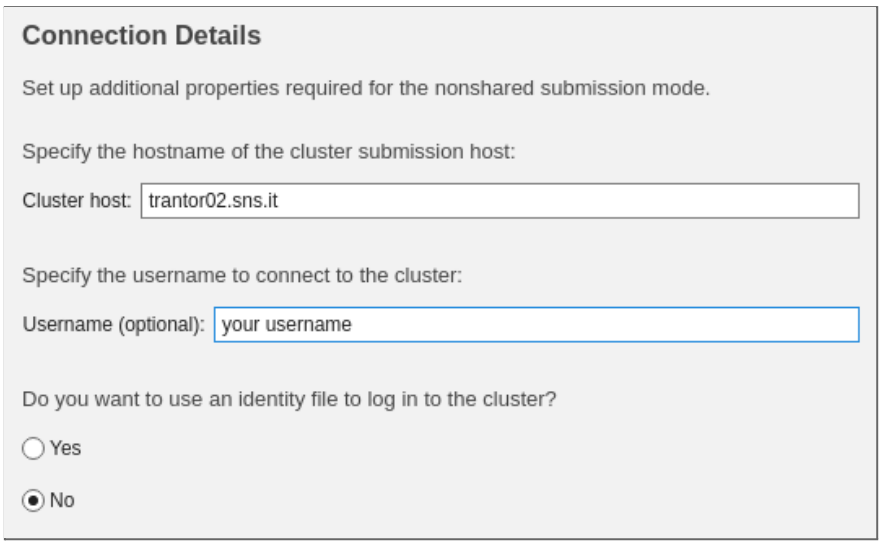
2.2. Q: *Can your cluster and client machines read and write to a shared job storage location?*
A: **No**



2.3. Q: *Specify the hostname of the cluster submission host*
A: Insert the FQDN of a head-node, such as **trantor02.sns.it**

2.4. Q: *Specify the username to connect to the cluster*
A: Insert your **Trantor account username**.

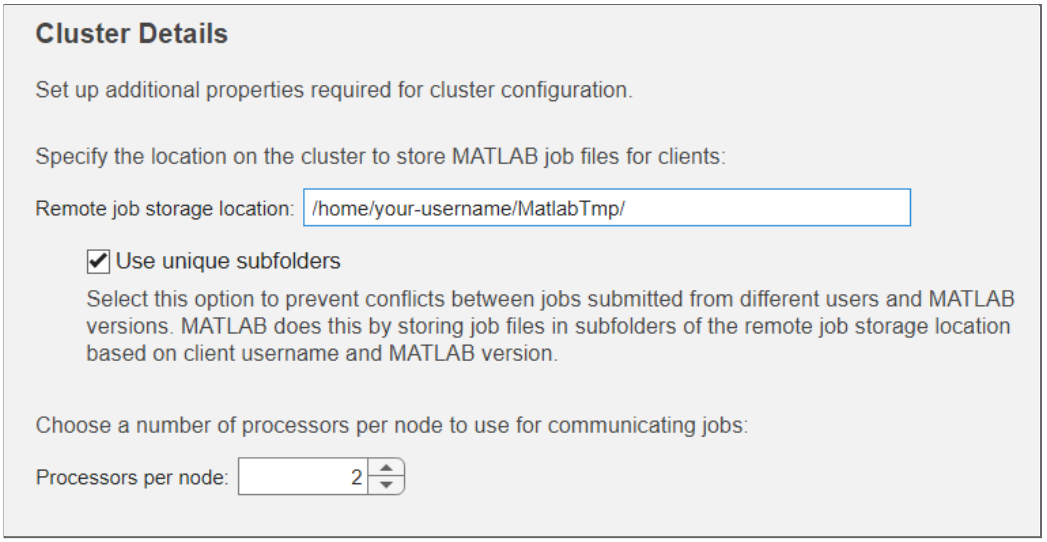
2.5. Q: *Do you want to use an identity file to log in to the cluster?*
A: **No**



2.6. Q: *Specify the location on the cluster to store MATLAB job files for clients*
A: The program is asking for a temporary folder where to store the data exchanged between the Matlab instance running on your workstation and the workers running on the cluster's compute nodes. We will use a temporary folder in your Trantor home directory for that purpose: **`/home/<username>/MatlabTmp/`**
(replace <username> with your actual *Trantor* username).

2.7. Q: *Use unique subfolders*
A: Be sure that the checkbox is **checked**.

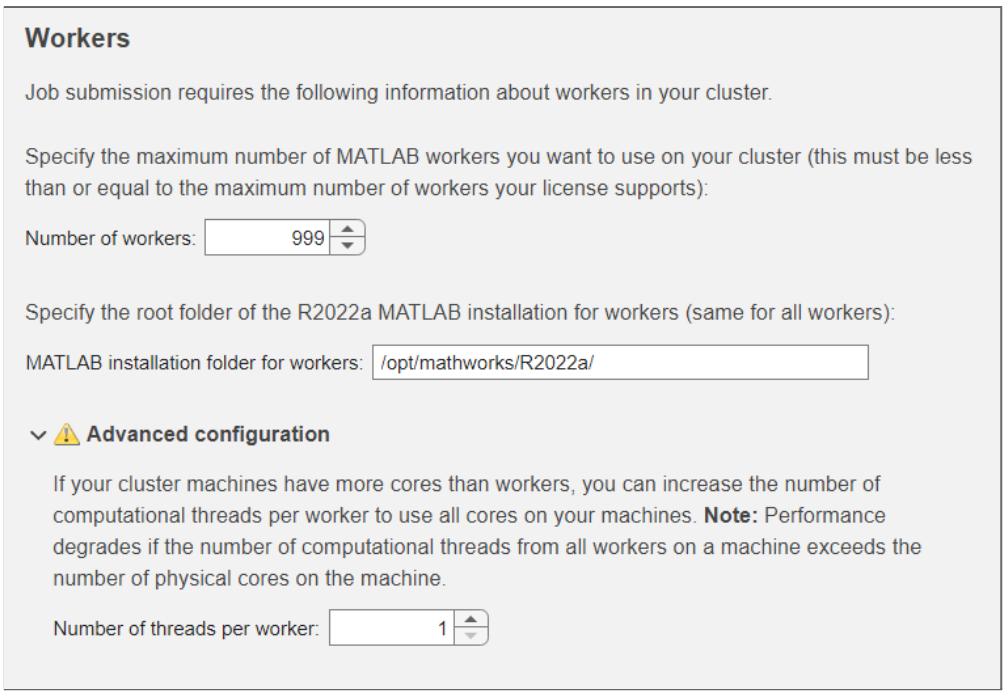
2.8. Q: *Choose a number of processors per node to use for communicating jobs*
A: You can leave this field to its default value of **2**. This parameter represents the number of CPU cores per node that Matlab will ask to the scheduling system when submitting a new job. It is preferable to set the actual value for this parameter by code, as discussed in the following.



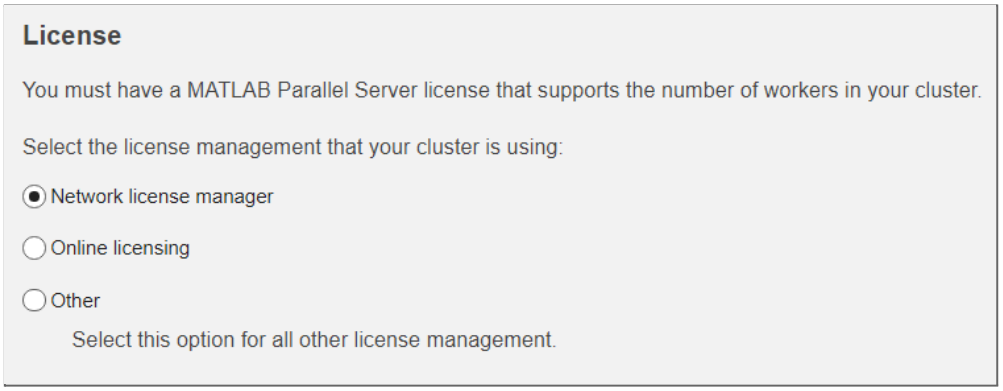
2.9. Q: *Number of workers*
A: In our cluster this parameter is just an upper-bound. Set it to **999**.

2.10. Q: *MATLAB installation folder*
A: Insert **`/opt/mathworks/R2022a/`**

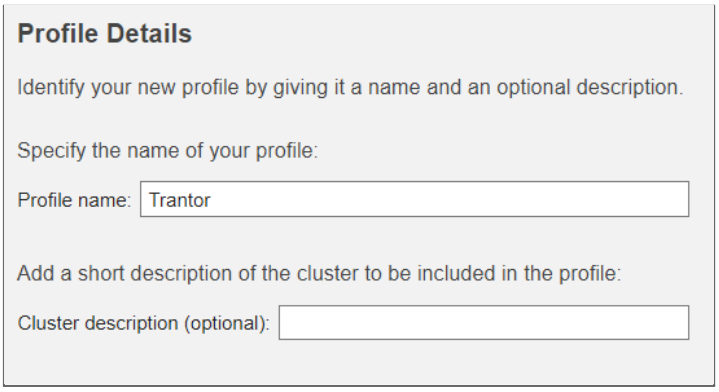
2.11. Q: *Number of threads per worker* (under *Advanced configuration*)
A: Leave this field to the default value of **1**.



2.12. Q: *Select the license management that your cluster is using*
A: **Network license manager**



2.13. Q: *Specify the name of your profile*
A: Name the profile '**Trantor**'.

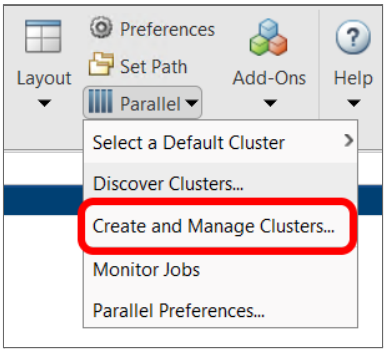


2.14. In the next step, a summary is displayed. Check the inserted values, then press *Create* to save the profile.

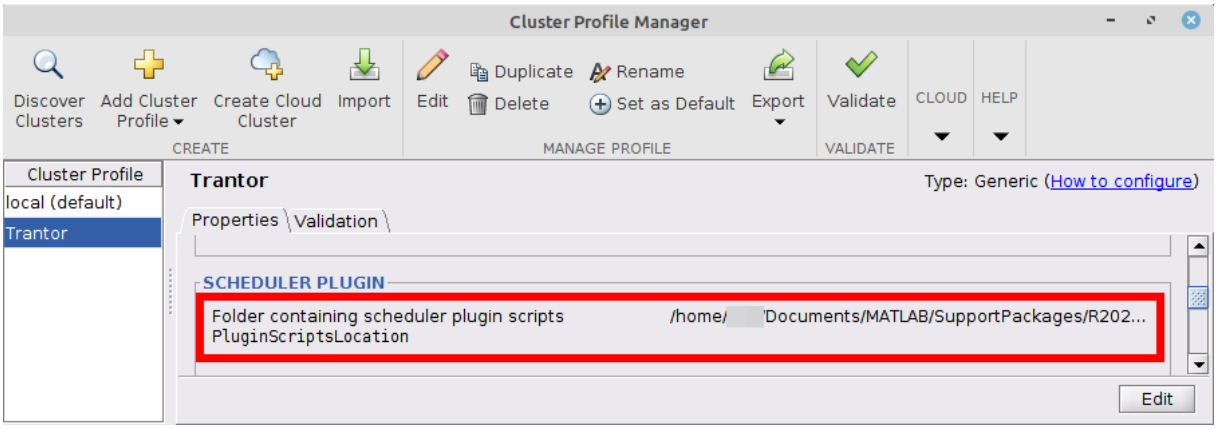
2.15. Finally, a confirmation dialog appears, asking if you want to set the *Trantor* cluster profile as the new default. It is recommended to leave this checkbox **unchecked**, thus leaving the 'local' profile as default. In fact, **you should develop and debug your program on your machine and resort to the cluster only for long-running computations**.

3. Now you need to apply a couple of changes to the scripts of the "*Parallel Computing Toolbox plugin for MATLAB Parallel Server with PBS*" plugin.

3.1. First of all, you need to find where these scripts have been stored. In the toolbar select the *HOME* tab, then click on the *Parallel → Create and Manage Clusters…* menu entry. The *Cluster Profile Manager* window will appear.



3.2. Select ***Trantor*** from the *Cluster Profile* list then and the related *Properties* tab, then scroll the form until you find the *SCHEDULER PLUGIN* section. The path where the scripts are stored is given by the ***PluginScriptsLocation*** property.



3.3. Navigate to this path with your file manager or terminal. You should see the following files (among the others): ***independentSubmitFcn.m*** and ***communicatingSubmitFcn.m***

3.4. Download the sample code archive and unzip it. Within the ***SubmitFunctionsMod*** folder you will find two files: ***independentSubmitFcn_mod*** and ***communicatingSubmitFcn_mod***. They contain the code to be inserted into *independentSubmitFcn.m* and *communicatingSubmitFcn.m* (see below).

3.5. Open ***independentSubmitFcn.m*** and find the following line of code:

```
additionalSubmitArgs = sprintf('-l nodes=1:ppn=%d',
numberOfProcs);
```

Replace this line with the content of ***independentSubmitFcn_mod***.

3.6. Open ***communicatingSubmitFcn.m*** and find the following line of code:

```
additionalSubmitArgs = sprintf('-l nodes=%d:ppn=%d',
numberOfNodes, procsPerNode);
```

Replace this line with the content of the ***communicatingSubmitFcn_mod***.

4. It is now possible to 'validate' the profile, so to be sure that the configuration is working properly.

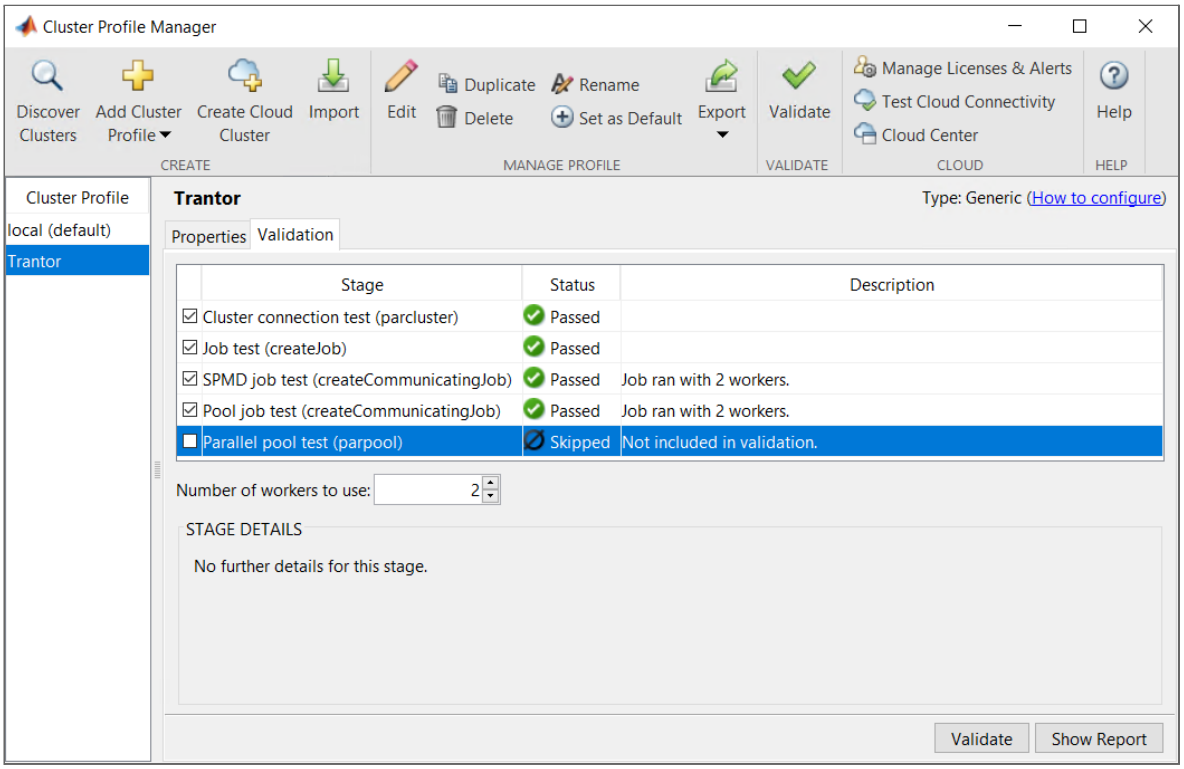
4.1. Open the *Cluster Profile Manager* window again. Be sure to select the *Trantor* cluster profile, then open the *Validation* tab.

4.2. You'll see a list of 'stages' (tests) that can be performed. Select all the stages with the exception of the one named "*Parallel pool test (parpool)*", which must be **unchecked**.

4.3. Set **2** as number of workers.

4.4. Finally, press the *Validate* button to start the tests. After a few seconds, MATLAB will ask you the password for the Trantor cluster.

4.5. All the activated tests should pass.



Tip

You can start a new cluster profile configuration wizard by executing the following command:

```
parallel.cluster.generic.runProfileWizard()
```

Running MATLAB batch jobs on the cluster

In general, Parallel Computing Toolbox supports two modalities to run computations on a cluster:

- By directly making use of a (local or remote) parallel pool of workers. This modality has the advantage of providing an interactive computing experience (see the parpool function documentation for details).
- By submitting the computation as an "off-line" job by means of the batch function. This modality is primarily designed for long-running tasks. Users can close the MATLAB client while the job is running, and retrieve the results from script once terminated.

Please note that, at this time, **only batch jobs are supported on *Trantor***. You can however use an interactive parallel pool on your local machine, by selecting the '**local**' parallel profile. This is particularly useful for development purposes.

The following sections discuss the code required to submit a batch job on *Trantor* and to retrieve the results from your scripts.

Job submission and monitoring

First of all, you need to decide the amount of computational resources to request for the job. Let's say you want to run your parallel job with *N* worker processes. In your script, you can then define the following variable:

```
workers = N;
```

Also note that MATLAB Parallel Server also spawns a 'master' process in addition to the workers:

```
processes = workers + 1;
```

The next step is to choose a group of nodes on which to run the computation. You do so by selecting a submission queue:

```
queueName = '...';
```

Note

At this time, only the following queues support the execution of MATLAB jobs:

- q07helicon

- q14helicon
- q07daneel
- q14daneel

Let *C* be the number of CPU cores available on the nodes you intend to use:

```
maxCoresPerNode = C;
```

Since it is advisable to reserve one CPU core for each process, the number of nodes and the amount of cores per node to be requested are given by:

```
coresPerNode = min(processes, maxCoresPerNode);
numberOfNodes = ceil(processes/coresPerNode);
```

You also have to define the amount of RAM memory to reserve on each node:

```
memPerProcess = M;
memPerNode = memPerProcess * coresPerNode;
```

where *M* is the memory to assign to each process, expressed in **GB**.

The next step is to create an instance of the *Trantor* cluster profile (see *parallel.Cluster* documentation for details). The resources requested for the job must then be assigned to the fields of the cluster object:

```
cluster = parcluster('Trantor');
cluster.AdditionalProperties.QueueName = queueName;
cluster.AdditionalProperties.ProcsPerNode = coresPerNode;
cluster.AdditionalProperties.MemPerNode = memPerNode;
cluster.AdditionalProperties.GpusPerNode = 0;
```

Note that:

- The number of compute nodes must not be explicitly set. Instead, it will be automatically computed according to the other parameters.
- You can reserve one or more GPUs per node if your computation requires them. If so, be sure to select a queue whose compute nodes are equipped with GPU hardware.

The actual job submission takes place by calling the *batch* function:

```
batch(cluster, @myFunction, 1, {200}, ...
      'Pool', workers,  ...
      'AutoAddClientPath', false);
```

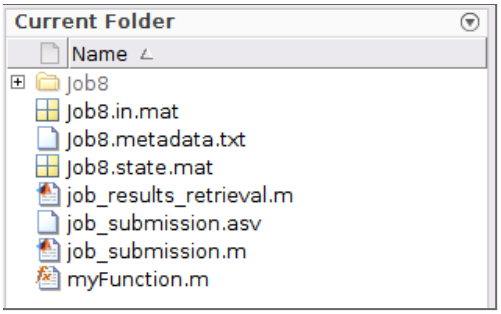
You should provide at least the following arguments to *batch*:

- The cluster object.

- A reference to the function to be executed on the cluster, the number of return values and the number of arguments.
- The number of workers to employ.
- The assignment of the `AutoAddClientPath` property to `false`. This prevents client-related paths from being used by the workers.

On *batch* execution, the following happens:

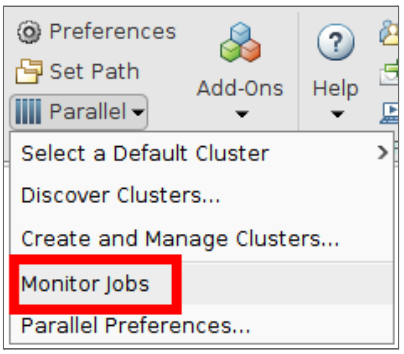
1. The program will ask for your Trantor password.
2. A few job data files will be created on your current working directory, as shown by the following screenshot:

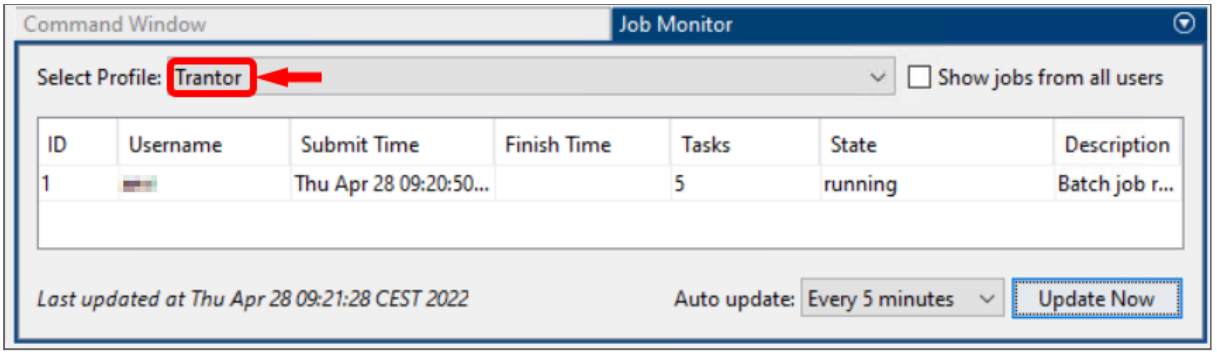


   Do not touch these files!

3. The job data files will be uploaded to your *Trantor* home folder, within the *MatlabTmp* directory.
4. A job is submitted to the PBS scheduling system.

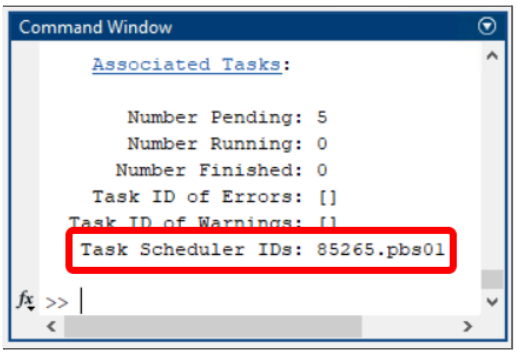
You can monitor the state of your jobs by opening the *Monitor Job* panel from the *Parallel* menu in the toolbar:





In the *Job Monitor* panel **be sure to select the '*Trantor*' profile**.

Of course, you can also use the ***qstat*** utility to monitor your job (see here for details). To this end, you can find the PBS Job ID by right-clicking over the job row and selecting *Show Details* from the contextual menu. The PBS Job ID will be printed on the *Command Window*:



Once the job appears in the *Job Monitor* panel with a **queued or running state, you can safely close MATLAB on your local machine**.

Retrieving the job results

Since batch processing on a cluster is designed for long-running jobs, it is common to open MATLAB from time to time to check if the job is still running and, once finished, to retrieve the results.

Once opened MATLAB, you may need to **change the current working folder to the directory where the job data files are located** (otherwise MATLAB will not be able to recognize the job). Then open the *Job Monitor* panel and **change the selected profile to '*Trantor*'** (if the *Trantor* profile was already selected, you may need to switch to the *local* profile and then select *Trantor* again). The program will ask you for the password of your Trantor account, then the job's details should appear in the *Job Monitor* panel.

Tip

As explained before, a faster way to check if your job is still running is by using the ***qstat*** utility. For example, you can execute the following command from your terminal to connect to the cluster, executing *qstat* to print your **running** jobs and then automatically disconnect:

```
ssh username@trantor02.sns.it qstat -n1 -u username
```

When the job is finished, you can retrieve the results from script, so to visualize them and/or perform further analysis.

First of all, you need to create a new instance of the '*Trantor*' cluster profile:

```
cluster = parcluster('Trantor');
```

Then, use the *findJob* function to (re)create an object representing your job:

```
job = findJob(cluster, 'ID', JOB_ID);
```

where the Job ID is the one shown on the first column of the *Job Monitor* panel.

To retrieve the results it is sufficient to call the *fetchOutputs* function, passing the job object as argument:

```
results = fetchOutputs(job);
```

Finally, it is important to delete the job, so to clean the residual data files from the current working folder on your local machine and from the temporary directory in your Trantor home:

```
delete(job);
```

Sample code

You can download a few examples and the code required by the configuration procedure here. The archive contains the following folders:

- *SubmitFunctionsMod* : replacement code for the "*Parallel Computing Toolbox plugin for MATLAB Parallel Server with PBS*" plugin.
- *JobSubmissionAndRetrieval* : this example shows how to submit a long-running job on the cluster and retrieve the results at a later time.
- *GPUsInfo* : this example shows how to retrieve the characteristics of the GPUs that have been reserved on the compute node by means of the *gpuDeviceTable()* function.

Best practices

- **Prefer the '*local*' cluster profile during development**, since it allows you to develop and test your parallel program in an interactive fashion (on your local machine). Once the codebase is stable, you can then make use of the cluster to speed-up long-running computations.
- If your computation performs I/O intensive operations on files, **make use of the local scratch areas** available on many compute nodes, as discussed here. Then, at the end of the computation, please **remember to copy any relevant file from the**

scratch folder back into your home or project folder. In fact, local scratch areas are **not backed up and can be erased by the technical staff** for maintenance purposes.

- While most of the data files are erased when deleting the job from MATLAB, the program leaves some residues in the *`MatlabTmp`* folder of your Trantor home. Please **clean up the content of this directory from time to time**. However, be sure that no MATLAB jobs are running when deleting these files.

Further documentation

To learn more about Parallel Computing Toolbox and Batch Processing in MATLAB, please refer to the official documentation:
https://www.mathworks.com/help/parallel-computing/index.html
https://www.mathworks.com/help/parallel-computing/batch-processing.html

You can find an introduction to the PBS job scheduler and its use on Trantor at the following Web page:
https://hpccenter.sns.it/page/wiki/pages/Submitting_Inspecting_and_Cancelling_PBS_Jobs.html