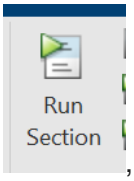
MATLAB Job Submission on MatriCS

This example shows how to submit a MATLAB job on MatriCS.

To execute each section in this script you can either click on 

, or click on the blue bar on the left of the section 

.

Note: Run the sections sequentially to prevent errors.

Table of Contents

Documentation and Examples

- Get Started with Parallel Computing Toolbox
- Run MATLAB Functions on a GPU
- Choose Between spmd, parfor, and parfeval - MATLAB & Simulink (mathworks.com)

Check the available partitions

```
disp(parallel.clusterProfiles)
```

Cluster Configuration

- Choose the partition, the number of workers (or GPUs if using GPU servers partition), and the memory.
- **Number of Workers:** 4
- **Number of GPU:** (if available on the partition)
- Click on Save to save the configuration.

```
c = parcluster("MatriCS-normal-amd");

fprintf("%s has %d available GPUs\n", c.Profile,
c.AdditionalProperties.NumGPUs);

NUM_WORKERS = 1;
NUM_GPUS = 1;
```

```
if c.AdditionalProperties.NumGPUs > 0 % If GPUs are available on the
partition
    c.PreferredPoolNumWorkers = NUM_GPUS;
    c.AdditionalProperties.AdditionalSubmitArgs = strcat('--gres=gpu:',
num2str(NUM_GPUS));
else
    c.PreferredPoolNumWorkers = NUM_WORKERS;
end
```

Job Submission

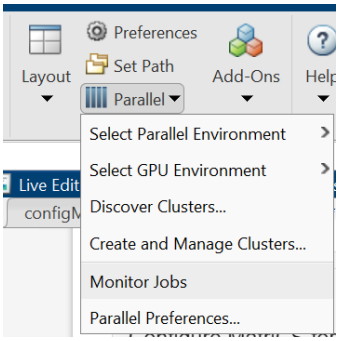
In this example, the code being executed approximates the value of Pi using a Monte Carlo method. See the Job Submission section and the call to the `@computePi` function. The code that approximates pi is written in the Write Your Own Function section. You can write your own MATLAB function here and then test it. If you change the function name, remember to change it in the `batch` function as well.

```
job = batch(c,@computePI,1,{10e3,10e3},Pool=c.PreferredPoolNumWorkers,CurrentFolder='.');
```

```
job = batch(c,@computePI,1,
{10e3,10e3},Pool=c.PreferredPoolNumWorkers,AutoAddClientPath=false,CurrentFo
lder='.');
```

Monitor your job

To monitor your job you can click on Home and then on Parallel (Icon with 4 blue bars) then choose Monitor Jobs.



You can also just display the variable **`job`**

```
job
```

or see all the jobs

```
c.Jobs
```

Get the result

If the job has completed successfully, you should see the estimated value of pi displayed, approximately 3.14.

```
if job.State == "finished"
    results = job.fetchOutputs
else
   disp(['Job state: ', job.State]);
end
```

Compute PI with Monte Carlo Method

```
function PI = computePI(m,n)
if canUseGPU
    c = gpuArray.zeros(1);
    for i = 1:n
        x = gpuArray.rand(m,1);
        y = gpuArray.rand(m,1);
        r = x.^2 + y.^2;
        c = c + sum(r<=1);
    end
    PI = 4/(m*n) * gather(c);
else
    c = 0;
    parfor i = 1:n
        x = rand(m,1);
        y = rand(m,1);
        r = x.^2 + y.^2;
        c = c + sum(r<=1);
    end
    PI = 4/(m*n) * c;
end
end
```

Write Your Own Function

Now to you: write your function and modify the batch command to call your function.

```
function out = myFunction(in)
    out = in;
end
```