

Parallel Computing with MATLAB on the AI and PARAM Shakti HPC Cluster

This document provides the steps to configure MATLAB to submit jobs to a cluster, retrieve results, and debug errors.

Initial Configuration - Running MATLAB on the AI and PARAM Shakti HPC Cluster

After logging into the cluster, configure MATLAB to run parallel jobs on the cluster by calling the shell script `configCluster.sh`.

```
$ module load matlab[/VERSION]
$ configCluster.sh
```

Jobs will run across multiple nodes on the cluster rather than on the host machine.

Initial Configuration - Running MATLAB on the Desktop

This setup is intended for job submission when MATLAB is installed on your machine and jobs are run remotely on the cluster. This setup needs to be done once per cluster, per version of MATLAB installed on your machine.

Start MATLAB and run `userpath`

```
userpath
```

Download the MATLAB plugin scripts from [here](#). The contents of the ZIP file should be extracted into the folder returned by the call to `userpath`.

Create a new cluster profile using option [2]

```
configCluster
  [1] ai-cluster
  [2] param-shakti
Select a cluster [1-2]:
```

Submission to the cluster requires SSH credentials. You will be prompted for username followed by 2FA and password.

Jobs will now run on the cluster rather than on the local machine.

NOTE: To run jobs on the local machine instead of the cluster, use the Process profile.

```
% Get a handle to the local resources
c = parcluster('Processes');
```

Configuring Jobs

Prior to submitting the job, various scheduler flags can be assigned, such as queue, e-mail, walltime, etc. *None of these are required.*

```

% Get a handle to the cluster
c = parcluster;

% REQUIRED

% Specify the partition
c.AdditionalProperties.Partition = 'partition-name';

% OPTIONAL

% Specify an account
c.AdditionalProperties.AccountName = 'account-name';

% Specify a constraint
c.AdditionalProperties.Constraint = 'feature-name';

% Request email notification of job status
c.AdditionalProperties.EmailAddress = 'user-id@iitkgp.ac.in';

% Specify number of GPUs (default: 0)
c.AdditionalProperties.GPUsPerNode = 1;

% Specify a particular GPU card
c.AdditionalProperties.GPUCard = 'gpu-card';

% Specify memory to use, per core (default: 4GB)
c.AdditionalProperties.MemPerCPU = '6GB';

% Specify the partition
c.AdditionalProperties.Partition = 'partition-name';

% Specify cores per node
c.AdditionalProperties.ProcsPerNode = 4;

% Specify the QoS
c.AdditionalProperties.QoS = 'qos-name';

% Set node exclusivity (default: false)
c.AdditionalProperties.RequireExclusiveNode = true;

% Specify a reservation
c.AdditionalProperties.Reservation = 'reservation-name';

% Specify the wall time (e.g., 1 day, 5 hours, 30 minutes)
c.AdditionalProperties.WallTime = '1-05:30';

```

To persist changes made to AdditionalProperties between MATLAB sessions, save the profile

```
c.saveProfile
```

To see the values of the current configuration options, display AdditionalProperties.

```
c.AdditionalProperties
```

Unset a value when no longer needed.

```
% Turn off email notifications
c.AdditionalProperties.EmailAddress = '';

% Don't request an entire node
c.AdditionalProperties.RequireExclusiveNode = false;
```

Interactive Jobs - Running MATLAB on the AI and PARAM Shakti HPC Cluster

To run an interactive pool job on the cluster, continue to use `parpool` as before.

```
% Get a handle to the cluster
c = parcluster;

% Open a pool of 64 workers on the cluster
pool = c.parpool(64);
```

Rather than running a local pool on the host machine, the pool can now run across multiple nodes on the cluster.

```
% Run a parfor over 1000 iterations
parfor idx = 1:1000
    a(idx) = rand;
end
```

Delete the pool when it's no longer needed.

```
% Delete the pool
pool.delete
```

Independent Batch Job - MATLAB on the AI and PARAM Shakti HPC Cluster or Desktop

Use the batch command to submit asynchronous jobs to the cluster. The batch command will return a job object which is used to access the output of the submitted job. See the MATLAB documentation for more help on [batch](#).

```
% Get a handle to the cluster
c = parcluster;

% Submit job to query where MATLAB is running on the cluster
job = c.batch(@pwd, 1, {}, 'CurrentFolder', '.');

% Query job for state
job.State

% If job is finished, fetch the results
job.fetchOutputs{1}

% Delete the job after results are no longer needed
```

```
job.delete
```

To retrieve a list of running or completed jobs, call `parcluster` to return the cluster object. The cluster object stores an array of jobs that are listed as *queued*, *running*, *finished*, or *failed*. Retrieve and view the list of jobs as shown below.

```
c = parcluster;
jobs = c.Jobs

% Get a handle to the second job in the list
job2 = c.Jobs(2);
```

Once the job has been selected, fetch the results as previously done.

`fetchOutputs` is used to retrieve function output arguments; if calling `batch` with a script, use `load` instead. Data that has been written to disk on the cluster needs to be retrieved directly from the file system (e.g., via `sftp`).

```
% Fetch all results from the second job in the list
job2.fetchOutputs{:}

% Alternate: Load results if job was a script instead of a function
job2.load
```

Parallel Batch Job - MATLAB on the AI and PARAM Shakti HPC Cluster or Desktop

The batch command can also support parallel workflows. Let's use the following example for a parallel job, which you should save separately as `parallel_example.m`.

```
function [sim_t, A] = parallel_example(iter)

if nargin==0
    iter = 8;
end

disp('Start sim')

A = nan(iter,1);
t0 = tic;
parfor idx = 1:iter
    A(idx) = idx;
    pause(2)
    idx
end
sim_t = toc(t0);

disp('Sim completed')

save RESULTS A

end
```

This time when using the batch command, specify a Pool argument.

```
% Get a handle to the cluster
c = parcluster;

% Submit a batch pool job using 4 workers for 16 simulations
job = c.batch(@parallel_example, 1, {16}, 'CurrentFolder', '.', 'Pool', 4);

% View current job status
job.State

% Fetch the results after a finished state is retrieved
job.fetchOutputs{1}
ans =
    8.8872
```

The job ran in 8.89 seconds using four workers. Note that these jobs will always request N+1 CPU cores, since one worker is required to manage the batch job and pool of workers. For example, a job that needs eight workers will require nine CPU cores.

Run the same simulation again but increase the Pool size. This time, to retrieve the results later, keep track of the job ID.

NOTE: For some applications, there will be a diminishing return when allocating too many workers, as the overhead may exceed computation time.

```
% Get a handle to the cluster
c = parcluster;

% Submit a batch pool job using 8 workers for 16 simulations
job = c.batch(@parallel_example, 1, {16}, 'CurrentFolder', '.', 'Pool', 8);

% Get the job ID
id = job.ID
id =
    4

% Clear job from workspace (as though MATLAB exited)
clear job
```

With a handle to the cluster, the findJob method searches for the job with the specified job ID.

```
% Get a handle to the cluster
c = parcluster;

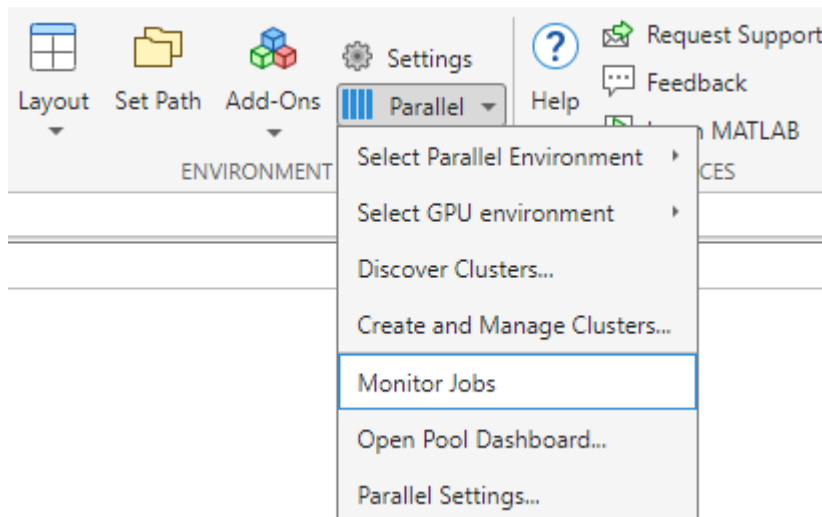
% Find the old job
job = c.findJob('ID', 4);

% Retrieve the state of the job
job.State
ans =
    finished
```

```
% Fetch the results
job.fetchOutputs{1};
ans =
    4.7270
```

The job now runs in 4.73 seconds using eight workers. Run code with different numbers of workers to determine the ideal number to use.

Alternatively, to retrieve job results via a graphical user interface, use the Job Monitor (Parallel > Monitor Jobs).



Job Monitor

Select Profile: HPC-Prod-Cluster (default) ☐ Show jobs from all users

ID	Username	Submit Time	Finish Time	Tasks	State	Storage...	Description
39		Mar 18, 2025, 3:40:56 PM	Mar 18, 2025, 3:42:41 PM	4	⚠ finished	24 KB	Batch job running function
40		Mar 18, 2025, 3:41:34 PM	Mar 18, 2025, 3:43:21 PM	1	❌ finished	12 KB	Batch job running function
41		Mar 18, 2025, 3:42:20 PM	Mar 18, 2025, 3:43:57 PM	1	⚠ finished	14 KB	Batch job running function
42		Mar 18, 2025, 3:43:39 PM	Mar 18, 2025, 3:44:58 PM	1	⚠ finished	14 KB	Batch job running function
49		May 6, 2025, 3:52:12 PM	May 6, 2025, 3:52:31 PM	1	⚠ finished	14 KB	Batch job running function
50		May 6, 2025, 3:54:54 PM	May 6, 2025, 3:55:08 PM	4	✅ finished	21 KB	Batch job running function
51		May 6, 2025, 3:55:18 PM	May 6, 2025, 3:55:51 PM	7	✅ finished	25 KB	Batch job running function
52		May 6, 2025, 3:55:26 PM	May 6, 2025, 3:56:15 PM	3	✅ finished	20 KB	Batch job running function
53		May 6, 2025, 3:55:35 PM		9	queued	16 KB	Batch job running function
54		May 6, 2025, 3:55:45 PM	May 6, 2025, 3:56:13 PM	1	✅ finished	13 KB	Batch job running function

Auto update: Every 5 minutes Update Now

Debugging

If a serial job produces an error, call the `getDebugLog` method to view the error log file.

When submitting an independent job, specify the task.

```
c.getDebugLog(job.Tasks)
```

For Pool jobs, only specify the job object.

```
c.getDebugLog(job)
```

When troubleshooting a job, the cluster admin may request the scheduler ID of the job. This can be derived by calling `getTaskSchedulerIDs`.

```
job.getTaskSchedulerIDs()  
ans =  
    25539
```

Helper Functions

Function	Description	Notes
<code>clusterFeatures</code>	Lists cluster features/constraints	
<code>clusterGpuCards</code>	Lists cluster GPU cards	
<code>clusterPartitionNames</code>	Lists cluster partition names	
<code>disableArchiving</code>	Modifies file archiving to resolve file mirroring issues	Applicable only to Desktop
<code>fixConnection</code>	Reestablishes cluster connection (e.g., after reconnection of VPN)	Applicable only to Desktop
<code>seff</code>	Displays Slurm statistics related to the efficiency of resource usage by the job	
<code>willRun</code>	Explains why job is queued	

To Learn More

To learn more about the MATLAB Parallel Computing Toolbox, check out these resources:

- [Parallel Computing Overview](#)
- [Parallel Computing Documentation](#)
- [Parallel Computing Coding Examples](#)
- [Parallel Computing Tutorials](#)
- [Parallel Computing Videos](#)
- [Parallel Computing Webinars](#)