Batch

Run MATLAB script or function on worker

Syntax

j = batch('aScript')
j = batch(myCluster,'aScript')
j = batch(fcn,N,{x1, ..., xn})
j = batch(myCluster,fcn,N,{x1,...,xn})
j = batch(...,'*p1*',v1,'*p2*',v2,...)

Arguments

Argument	Description
j	The batch job object.
'aScript'	The script of MATLAB code to be evaluated by the worker.
myCluster	Cluster object representing cluster compute resources.
fcn	Function handle or string of function name to be evaluated by the worker.
N	The number of output arguments from the evaluated function.
{x1, ..., xn}	Cell array of input arguments to the function.
p1, p2	Object properties or other arguments to control job behavior.
v1, v2	Initial values for corresponding object properties or arguments.

Description

j = batch('aScript') runs the script code of the file aScript.m on a worker in the cluster specified by the default cluster profile. (Note: Do not include the .m file extension with the script name argument.) The function returns j, a handle to the job object that runs the script. The script file aScript.m is copied to the worker.

j = batch(myCluster,'aScript') is identical to batch('aScript') except that the script runs on a worker according to the cluster identified by the cluster object myCluster.

j = batch(fcn,N,{x1, ..., xn}) runs the function specified by a function handle or function name, fcn, on a worker in the cluster identified by the default cluster profile. The function returns j, a handle to the job object that runs the function. The function is evaluated with the given

arguments, x1,...,xn, returning N output arguments. The function file for fcn is copied to the worker. (Do not include the .m file extension with the function name argument.)

j = batch(myCluster,fcn,N,{x1,...,xn}) is identical to batch(fcn,N,{x1,...,xn}) except that the function runs on a worker in the cluster identified by the cluster object myCluster.

j = batch(...,'*p1*',v1,'*p2*',v2,...) allows additional parameter-value pairs that modify the behavior of the job. These parameters support batch for functions and scripts, unless otherwise indicated. The supported parameters are:

- 'Workspace' — A 1-by-1 struct to define the workspace on the worker just before the script is called. The field names of the struct define the names of the variables, and the field values are assigned to the workspace variables. By default this parameter has a field for every variable in the current workspace where batch is executed. This parameter supports only the running of scripts.
- 'Profile' — A single string that is the name of a cluster profile to use to identify the cluster. If this option is omitted, the default profile is used to identify the cluster and is applied to the job and task properties.
- 'AdditionalPaths' — A string or cell array of strings that defines paths to be added to the MATLAB® search path of the workers before the script or function executes. The default search path might not be the same on the workers as it is on the client; the path difference could be the result of different current working folders (pwd), platforms, or network file system access. The 'AdditionalPaths' property can assure that workers are looking in the correct locations for necessary code files, data files, model files, etc.
- 'AttachedFiles' — A string or cell array of strings. Each string in the list identifies either a file or a folder, which gets transferred to the worker.
- 'AutoAttachFiles' — A logical value to specify whether code files should be automatically attached to the job. If true, the batch script or function is analyzed and the code files that it depends on are automatically transferred to the worker. The default is true.
- 'CurrentFolder' — A string indicating in what folder the script executes. There is no guarantee that this folder exists on the worker. The default value for this property is the cwd of MATLAB when the batch command is executed. If the string for this argument is '.', there is no change in folder before batch execution.
- 'CaptureDiary' — A logical flag to indicate that the toolbox should collect the diary from the function call. See the diary function for information about the collected data. The default is true.
- 'Pool' — An integer specifying the number of workers to make into a parallel pool for the job *in addition* to the worker running the batch job itself. The script or function uses this pool for execution of statements such as parfor and spmd that are inside the batch code. Because the pool requires N workers in addition to the worker running the batch, there must be at least N+1 workers available on the cluster. You do not need a parallel pool already running to execute batch; and the new pool that batch creates is not related to a pool you might already have open. (See Run a Batch Parallel Loop.) The

default value is 0, which causes the script or function to run on only a single worker without a parallel pool.

Examples

Run a batch script on a worker, without using a parallel pool:

```
j = batch('script1');
```

Run a batch script that requires two additional files for execution:

```
j = batch('myScript','AttachedFiles',{'mscr1.m','mscr2.m'});
wait(j);
load(j);
```

Run a batch pool job on a remote cluster, using eight workers for the parallel pool in addition to the worker running the batch script. Capture the diary, and load the results of the job into the workspace. This job requires a total of nine workers:

```
j = batch('script1','Pool',8,'CaptureDiary',true);
wait(j);   % Wait for the job to finish
diary(j)   % Display the diary
load(j)    % Load job workspace data into client workspace
```

Run a batch pool job on a local worker, which employs two other local workers for the pool. Note, this requires a total of three workers in addition to the client, all on the local machine:

```
j = batch('script1','Profile','local','Pool',2);
```

Clean up a batch job's data after you are finished with it:

```
delete(j)
```

Run a batch function on a cluster that generates a 10-by-10 random matrix:

```
c = parcluster();
j = batch(c,@rand,1,{10,10});

wait(j)   % Wait for the job to finish
diary(j)  % Display the diary

r = fetchOutputs(j); % Get results into a cell array
r{1}              % Display result
```

Tips

To see your batch job's status or to track its progress, use the Job Monitor, as described in Job Monitor. You can also use the Job Monitor to retrieve a job object for a batch job that was created in a different session, or for a batch job that was created without returning a job object from the batch call.

As a matter of good programming practice, when you no longer need it, you should delete the job created by the batch function so that it does not continue to consume cluster storage resources.