An Update on MATLAB at SHARCNET

Jemmy Hu

SHARCNET HPC Consultant
University of Waterloo

April 15, 2015
Types of MATLAB packages

- MATLAB Site license (Western, McMaster, UWaterloo)
- MATLAB Compile Runtime (mcr)
- MATLAB PCT (Parallel Computing Toolbox)
Site license

- UW, Western, McMaster: license is managed on a campus license server, e.g., by IST at UW.

- Username match: your SHARCNET username should be the same as your institution username.

- Versions on the site specific SHARCNET systems
  - UW: orca, hound, mosaic (R2012b, R2014a)
  - Western: goblin (R2012a, R2014a)
  - McMaster: wobbie, iqaluk (R2012b, R2014a)

- License number is limited
  - UW: 300 basic MATLAB license campus wide, limit to 50 on SHARCNET systems, fewer licenses for many toolboxes
Site license

• Run batch job
  - basic command should include the following flags:
    `matlab -nodisplay -nosplash -nojvm -singleCompThread`
  - run job in the serial (default) queue using `sqsub`
    `sqsub --mpp=2g -r 1.0d -o test.log -i test.m ./uwmatlab`
  where ‘uwmatlab’ is a simple runscript script for UW users:
  `/opt/sharcnet/local/matlab/R2014a/bin/matlab -nodisplay -nosplash -nojvm -singleCompThread`

• Use ‘mcc’ to compile to standard binary code
  `mcc -m -R ' -nosplash -nodisplay -nodesktop -nojvm -singleCompThread' myprog.m`

• Run compiled code job
  `sqsub --mpp=4g -r 1.0h -o run_myprog.log ./run_myprog.sh path/R2014a`
Options for non-site license users

• Versions available on several clusters
  - R2012b.mcr
  - R2013b.mcr
  - R2014a.mcr

• MATLAB Compiler Runtime (mcr)
  - your license has MATLAB compiler, `mcc`, same version
  - compile your MATLAB codes on a Linux system
  - run the binary code on SHARCNET without license check

• Run mcr job (pre-load match version, e.g., R2014a)
  module load matlab/R2014a.mcr
  sqsub --mpp=4g -r 1.0h -o run_myprog.log ./run_myprog.sh $mcrroot
MATLAB PCT

- Server side license by SHARCNET, user group ‘matlab’

- Conditions to use PCT
  - you have a client side PCT license
  - modify your code to make use of PCT
MATLAB PCT Architecture (client-server)
Configure MATLAB and PCT on PC

- **Cluster server side (SHARCNET)**
  - setup MATLAB distributed computing server engine
  - setup ‘matlab’ queue
  - command/script for job submission

- **Client side configuration (user side)**
  - clusterInfo.m (set up cpu, memory, PATH etc., copy and modify)
  - runscript.m (copy and modify)
  - your own .m files
  - create local data directory, e.g., ‘C:\temp’ on a Windows PC
  - create data directory on SHARCNET cluster side (e.g., scratch/userid/matlab)

Install and configure instruction in the online document
Key Function List

- **Job Creation**
  
  - *createJob* Create job object in scheduler and client
  - *createTask* Create new task in job
  - *dfeval* Evaluate function using cluster

- **Interlab Communication Within a Parallel Job**
  
  - *labBarrier* Block execution until all labs reach this call
  - *labBroadcast* Send data to all labs or receive data sent to all labs
  - *labindex* Index of this lab
  - *labReceive* Receive data from another lab
  - *labSend* Send data to another lab
  - *numlabs* Total number of labs operating in parallel on current job

- **Job Management**
  
  - *cancel* Cancel job or task
  - *destroy* Remove job or task object from parent and memory
  - *getAllOutputArguments* Output arguments from evaluation of all tasks in job object
  - *submit* Queue job in scheduler
  - *wait* Wait for job to finish or change states
Demo: calculate pi using MATLAB

use the fact that
\[ \int_0^1 \frac{4}{1 + x^2} dx = 4(\tan(1) - \tan(0)) = \pi \]

to approximate pi by approximating the integral on the left.

divide the work between the labs by having each lab calculate the integral of the function over a subinterval of [0, 1] as shown in the picture

Steps

• All labs/workers will compute the same function: \( F = \frac{4}{1 + x^2} \)

• Each worker/lab will calculate over a subinterval \([a,b]\) of \([0, 1]\), for 2 labs, the subinterval will be:
  
  \[
  [0, 0.50] \\
  [0.50, 1.0]
  \]

  \[a = \frac{(\text{labindex}-1)}{\text{numlabs}}\]
  \[b = \frac{\text{labindex}}{\text{numlabs}}\]

• Use a MATLAB quadrature method to compute the integral
  \[
  \text{myIntegral} = \text{quadl}(F, a, b)
  \]

• Add together to form the entire integral over \([0,1]\)
  \[
  \text{piApprox} = \text{gplus}(\text{myIntegral})
  \]
```matlab
function piApprox = quadpi_1
%QUADPI Approximate pi .
% Approximate pi by the numerical integral of \( F = 4/(1 + x^2) \)
% from 0 to 1. \( F = \theta(x)4./(1 + x.^2) \);

% the integral of F over a subinterval \([a, b] \) of \([0, 1]\). 
a = 0;
b = 1;

% Use a built-in MATLAB quadrature method to approximate
% the integral. 
myIntegral = quadl(F,a,b);

piApprox = myIntegral
end
```

```
sqsub --mpp=2g -r 1.0h -o quadpi_1.log -i quadpi_1.m ./uwmatlab

[jemmyhu@hnd19:~/work/MATLAB/compiler] pwd
/home/jemmyhu/work/MATLAB/compiler
```
MCR:  1) mcc compiling

[jemmyhu@hnd19:~/work/MATLAB/compiler] ls
quadpi_1.m
[jemmyhu@hnd19:~/work/MATLAB/compiler] which mcc
/opt/sharcnet/local/matlab/R2014a/bin/mcc
[jemmyhu@hnd19:~/work/MATLAB/compiler] mcc -m -R ' -nosplash -nodisplay
-nodesktop --nojvm -singleCompThread' quadpi_1.m
Warning: ignore some warning lines
[jemmyhu@hnd19:~/work/MATLAB/compiler] ls
quadpi_1 readme.txt mccExcludedFiles.log quadpi_1.m run_quadpi_1.sh

2) Run job using R2014a.mcr
[jemmyhu@hnd19:~/work/MATLAB/compiler] sqsub --mpp=4g -r 1.0h -o
  run_quadpi_1.log ./run_quadpi_1.sh /opt/sharcnet/local/matlab/R2014a
function piApprox = quadpi

% QUADPI Approximate pi via parallel numerical quadrature.
% Copyright 2005 The MathWorks, Inc.
% Approximate pi by the numerical integral of F = 4/(1 + x^2) from 0 to 1.
% F = @(x)4./(1 + x.^2);

% Each lab calculates the integral of F over a subinterval [a, b] of [0, 1].
a = (labindex - 1)/numlabs;
b = labindex/numlabs;

% Use a built-in MATLAB quadrature method to approximate the integral.
myIntegral = quadl(F,a,b);

% The labs have now all calculated their portions of the integral of F,
% and will all send their results to lab 1, which will add them together
% to form the entire integral over [0, 1].
if (labindex == 1)
    % Receive the integral contribution from all the other labs.
    piApprox = myIntegral;
else
    % Send the integral contribution to lab 1.
    piApprox = []
    labSend(myIntegral, 1)
end
function run_quadpi_14a()

    cluster = clusterInfo_14a_hound

    % Create a Job Scheduler object
    Nprocs = 4;
    pjob = createCommunicatingJob(cluster, 'Type', 'SPMD')
    set(pjob, 'NumWorkersRange', Nprocs)

    % We need to include this file on the cluster
    set(pjob, 'AttachedFiles', {'quadpi.m'})

    % Create and submit the task, wait for results
    t = createTask(pjob, @(quadpi, 1, {})

    % Submit and Wait for results
    submit(pjob)
    wait(pjob)
    results = fetchOutputs(pjob)
    y = results(1)
% Function [ cluster ] = clusterInfo_14a_hound()
% all cpus in one compute node setting.
% Change the following 3 lines for cpus/workers, runtime (in second) and memory usage (in gb) to fit your job
Nprocs = 4;
Wtime=172800;
Memory= 2;
additionalSubmitArgs = sprintf('-1 npus=%d -1 walltime=%d,cput=%d -1 pmem=%dgb', Nprocs,Wtime,Wtime*Nprocs,Memory);
% Specify a cluster environment and use a local folder as the JobStorageLocation
cluster = parallel.cluster.Generic('JobStorageLocation', 'C:\Temp');
% Define the additional inputs to the submit functions
clusterHost = 'hound.sharcnet.ca';
remoteJobStorageLocation = '/scratch/jennyhu/matlab';
%set(cluster, 'JobStorageLocation', 'C:\Temp');

% Specify file system and MATLAB Root
set(cluster, 'HasSharedFilesystem', false);
set(cluster, 'ClusterMatlabRoot', '/opt/sharcnet/matlab/R2014a');
set(cluster, 'OperatingSystem', 'unix');

% The IndependentSubmitFcn must be a MATLAB cell array that includes the
% three additional inputs for serial tasks
%set(cluster, 'IndependentSubmitFcn', @(IndependentSubmitFcn, clusterHost, remoteJobStorageLocation, additionalSubmitArgs)
% If you want to run communicating jobs (including matlabpool), you must specify a CommunicatingSubmitFcn
set(cluster, 'CommunicatingSubmitFcn', @(CommunicatingSubmitFcn, clusterHost, remoteJobStorageLocation, additionalSubmitArgs)
set(cluster, 'GetJobStateFcn', @getjobstateFcn);
set(cluster, 'DeleteJobFcn', @deletejobFcn);
>> run('C:\Documents and Settings\JennyHu\My Documents\MATLAB\run_quadpi_14a.m')

cluster =

Generic Cluster

Properties:

    Profile:          Modified: true
    Host: 129.197.111.253
    NumWorkers: Inf

    JobStorageLocation: C:\Temp
    ClusterMatlabRoot: /opt/sharcnet/matlab/R2014a
    OperatingSystem: UNIX

    IndependentSubmitFcn: []
    CommunicatingSubmitFcn: [1x4 cell]
    GetJobStateFcn: @getJobStateFcn
    CancelJobFcn: []

results =

[3.1415]
[ ]
[ ]

y =

3.1416