Agenda

- Part I – Parallel Computing with MATLAB on the Desktop
  - Parallel Computing Toolbox
- Part II – Scaling MATLAB to PACE clusters
  - MATLAB Parallel Server

http://docs.pace.gatech.edu/software/Matlab
http://docs.pace.gatech.edu/software/matlab-distributed
Agenda

- Part I – Parallel Computing with MATLAB on the Desktop
  - Parallel Computing Toolbox
- Part II – Scaling MATLAB to PACE clusters
  - MATLAB Parallel Server

http://docs.pace.gatech.edu/software/Matlab
http://docs.pace.gatech.edu/software/matlab-distributed
Chatting

- Send to at least the Host, Presenter & Panelists
- Ideally, send to All Attendees
Scaling MATLAB to PACE clusters

- Accessing and running MATLAB on local HPC clusters
- Running parallel and multi-node MATLAB jobs
A note about today’s workshop…

- The workflow and examples are about process, not performance
Accessing and running MATLAB on local HPC clusters (1)

- Two options
  - ssh
    - Command line interface
    - Useful for either low-bandwidth or automation
  - VNC
    - Graphical interface
    - [http://docs.pace.gatech.edu/interactiveJobs/setupVNC_Session](http://docs.pace.gatech.edu/interactiveJobs/setupVNC_Session)
VPN into GT

https://docs.pace.gatech.edu/gettingStarted/vpn
Connect to PACE-ICE

- Open a terminal client to connect to PACE-ICE via ssh
  - On Mac or Linux, open a Terminal window.
  - On Windows 10, open PowerShell. PuTTY and other ssh software require additional steps for our port forwarding process, so PowerShell is preferred. Use the “plain” version of PowerShell, not ISE, x86, or another one with a longer name. PowerShell is built-in on Windows 10, so you have it already available to you.

- Type ssh USERNAME@pace-ice.pace.gatech.edu
  - Use your GT username (e.g., gburdell3)
  - Enter your GT password when prompted. No asterisks will appear, but type your full password, then Enter. You may be asked to say “yes” to other security prompts.
Start VNC session

- Virtual Network Computing (VNC) allows you to access a graphical desktop on a PACE compute node
- Use PACE’s wrapper to start a VNC job on the scheduler:
  - `pace-vnc-job -q pace-training -l nodes=1:ppn=8 -l walltime=3:00:00`
  - Hint: “-l” is a lowercase L in both cases
- If this is your first time using VNC on PACE-ICE, you’ll be prompted to set up a password
  - Run the `vncpasswd` command
  - This is not your GT password, but something just for VNC
  - Say no (“n”) to a view-only password
  - Then, try to run the `pace-vnc-job` command above again
- Wait for your job to start, then follow the on-screen instructions
Connect to compute node

- Following the on-screen instructions, type Shift +~ +C
  - (hold down Shift the whole time, then press and release ~, then press and release C; don't type the + signs)

- This should show the ssh> prompt. Copy/paste the purple line (step 2 of the on-screen instructions) beginning with –L and hit Enter twice. You should see a message that says, “Forwarding port.”
Troubleshooting

- If you are using PuTTY or another client, follow these instructions instead: [http://docs.pace.gatech.edu/interactiveJobs/pf_3rd_party_ssh](http://docs.pace.gatech.edu/interactiveJobs/pf_3rd_party_ssh)
- If you are using a non-US keyboard without the ~ key, open a separate terminal window and type
  
  ```
  ssh -N -L port:host:port USERNAME@pace-ice.pace.gatech.edu
  
  where port:host:port can be copied from the on-screen instructions in the purple line (step 2).
  ```
Open VNC viewer client

- Open your VNC viewer client on your local computer
  - RealVNC Viewer (https://www.realvnc.com/en/connect/download/viewer) or
  - TurboVNC (https://sourceforge.net/projects/turbovnc/files)
- In the bar in your client, type in the localhost address provided in the instructions in your terminal, e.g., localhost:1 and connect (Enter)
- Enter the VNC password you chose earlier (not your GT password) and accept any security warning
- Your VNC desktop should open and load
Load and run MATLAB

- Open Terminal client from the Linux desktop
  - Click on Applications > System Tools > Terminal

- Load the matlab module
  - In the terminal, type `module load matlab/r2020b` and hit Enter

- Run MATLAB
  - In the terminal, type `matlab &` and hit Enter
Add MATLAB to the system path

```
bash4.2 # Add MATLAB to the system path
bash4.2 module load matlab/r2020b
bash4.2 # Run MATLAB
bash4.2 matlab &
```
Start MATLAB
Parallel MATLAB – Single Node

```matlab
>> maxNumCompThreads
ans = 4

>> parpool('local');
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
```

-bash4.2 pace-vnc-job -q pace-training -l nodes=1:ppn=4 -l walltime=03:00:00

Submitting job via qsub pace-vnc-job.pbs...
99558.sched-pace-ice.pace.gatech.edu

Job successfully submitted!
Waiting for job to start...

Starting vncserver...
Parallel MATLAB – Multi-node (1)

- In order to run a multi-node MATLAB job, MATLAB will generate and submit a new PBS job
  - Executed during any “job launcher”
    - parpool*, batch, createJob
  - Run asynchronously while MATLAB session is running, except parpool
  - True regardless if we’re running MATLAB desktop or a PBS job script

- Need to generate a new profile for PACE cluster
  - configCluster

MATLAB “outer job”
parallel “inner job”
local profile

“How does MATLAB know about PACE?”
Configure MATLAB to create PACE profile

- Exercises
  - Parallel-Computing
  - StartWorkers...

```matlab
>> configCluster
   Must set MemPerCpu and QueueName before submitting jobs
   >> c = parcluster;
   >> % Request 4 gb per core
   >> c.AdditionalProperties.MemPerCpu = '4gb';
   >> c.AdditionalProperties.QueueName = 'queue-name';
   >> c.saveProfile

>> c = parcluster;
>> c.AdditionalProperties.MemPerCpu = '4gb';
>> c.AdditionalProperties.QueueName = 'pace-training';
>> c.saveProfile
```

Minimum requirements
New PACE profile

Only call `configCluster` once
If there are 3x more workers than a local pool, why did it take the same amount of time?

```matlab
>> % Get handle to PACE cluster
>> c = parcluster;
>>
>> % Start parallel pool
>> p = c.parpool(12);
Starting parallel pool (parpool) using the 'pace R2020b' profile 
additional\SubmitArgs =

    '-l nodes=1:ppn=12 -l pmem=4gb -q pace-training'

Connected to the parallel pool (number of workers: 12)
>> tic, parfor idx = 1:240, pause(3), end, toc
Elapsed time is 60.299550 seconds.
>>
>> parpool('local');
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
>>
>> tic, parfor idx = 1:80, pause(3), end, toc
Elapsed time is 60.296822 seconds.
>>
```
Download workshop files

-bash4.2
-bash4.2 # Make a local copy of the Workshop files (Part II)
bash4.2 cp -frp /storage/home/hpaceicel/shared-classes/materials/mathworks/matlab-workshop $HOME/Documents/MATLAB
-bash4.2
-bash4.2
Change directories to workshop

```matlab
>> cd(fullfile(userpath,'matlab-workshop'))
```
Exercise: Calculate $\pi$

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

\[F(x) = \frac{4}{1 + x^2}\]
function calc_pi

% Query for available cores (assume either Slurm or PBS)
sz = str2num([getenv('SLURM_CPUS_PER_TASK') getenv('PBS_NUM'));]
if isempty(sz), sz = maxNumCompThreads; end

c.parpool(sz);

spmd
    a = (labindex - 1)/numlabs;
b = labindex/numlabs;
    fprintf('Subinterval: [%-4g, %-4g]\n', a, b)
    myIntegral = integral(@quadpi, a, b);
    fprintf('Subinterval: [%-4g, %-4g] Integral: %4g\n', a, b, myIntegral)
    piApprox = gplus(myIntegral);
end

approx1 = piApprox(1); % 1st element holds value on worker 1
fprintf('pi : %.18f\n', pi)
fprintf('Approximation: %.18f\n', approx1)
fprintf('Error : %g\n', abs(pi - approx1))

function y = quadpi(x)
%QUADPI Return data to approximate pi.

% Derivative of 4*atan(x)
y = 4./((1 + x.^2));

function calc_pi_multi_node

% Set additional properties

c = parcluster;
c.AdditionalProperties.MemPerCpu = '4gb';
c.AdditionalProperties.QueueName = 'pace-training';

c.parpool(12);

spmd
    a = (labindex - 1)/numlabs;
b = labindex/numlabs;
    fprintf('Subinterval: [%-4g, %-4g]\n', a, b)
    myIntegral = integral(@quadpi, a, b);
    fprintf('Subinterval: [%-4g, %-4g] Integral: %4g\n', a, b, myIntegral)
    piApprox = gplus(myIntegral);
end

approx1 = piApprox(1); % 1st element holds value on worker 1
fprintf('pi : %.18f\n', pi)
fprintf('Approximation: %.18f\n', approx1)
fprintf('Error : %g\n', abs(pi - approx1))

function y = quadpi(x)
%QUADPI Return data to approximate pi.

% Derivative of 4*atan(x)
y = 4./((1 + x.^2));
Results

>> calc_pi
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
Lab 1:
   Subinterval: [0 , 0.25]
Lab 2:
   Subinterval: [0.25, 0.5 ]
Lab 3:
   Subinterval: [0.5 , 0.75]
Lab 4:
   Subinterval: [0.75, 1 ]
Lab 1:
   Subinterval: [0 , 0.25]    Integral: 0.979915
Lab 2:
   Subinterval: [0.25, 0.5 ]  Integral: 0.874676
Lab 3:

>> % Close local pool
>> delete(gcp)
Parallel pool using the 'local' profile is shutting down.

>> calc_pi_multi_node
Starting parallel pool (parpool) using the 'pace R2020b' profi

additionalSubmitArgs =
   ' -l nodes=1:ppn=12 -l pmem=4gb -q pace-training'
Connected to the parallel pool (number of workers: 12).
Lab 1:
   Subinterval: [0 , 0.0833333]
Lab 2:
   Subinterval: [0.0833333, 0.166667]
Other settable job properties (1)

```
>> c = parcluster;
>> c.AdditionalProperties
ans =
    AdditionalProperties with properties:
        AccountName: ''
        AdditionalSubmitArgs: ''
        EmailAddress: ' '
        EnableDebug: 0
        Features: ' '
        GpusPerNode: 0
        LocalDiskSize: ' '
        MemPerCpu: '4gb'
        ProcsPerNode: 0
        QueueName: 'pace-training'
        UseSmpd: 0
        WallTime: ' '
```
Other settable job properties (2)

- AccountName
- EmailAddress
- Features
- GpusPerNode
- LocalDiskSize
- MemPerCpu
- ProcsPerNode
- QueueName
- WallTime
GPU Arrays
Start pool with GPU node

```matlab
>> % Start parallel pool with a GPU
>> c = parcluster;
>> c.AdditionalProperties.GpusPerNode = 1;
>> c.AdditionalProperties.QueueName = 'pace-ice-gpu';
>> p = c.parpool(1);
Starting parallel pool (parpool) using the 'pace R2020b' profile
additionalSubmitArgs =
    '-l nodes=1:ppn=1:gpus=1 -l pmem=4gb -q pace-ice-gpu'
Connected to the parallel pool (number of workers: 1).
```
Tesla V100

>> spmd, gpuDevice, end
Lab 1:
ans =
    CUDADevice with properties:
    Name: 'Tesla V100-PCIE-16GB'
    Index: 1
    ComputeCapability: '7.0'
    SupportsDouble: 1
    DriverVersion: 11.1000
    MaxThreadsPerBlock: 1024
    MaxShmemPerBlock: 49152
    MaxThreadBlockSize: [1024 1024 64]
    MaxGridSize: [2.1475e+09 65535 65535]
    SIMDWidth: 32
    TotalMemory: 1.6946e+10
function [x,y,count,t] = calc_mandelbrot(type)

maxIterations = 1000;
gridSize = 4000;
xlim = [-0.748766713922161, -0.748766707771757];
ylim = [ 0.123640844894862, 0.123640851045266];

t0 = tic;
if strcmp(type,'gpuArray')
    x = gpuArray.linspace(xlim(1),xlim(2),gridSize);
    y = gpuArray.linspace(ylim(1),ylim(2),gridSize);
else
    x = linspace(xlim(1),xlim(2),gridSize);
    y = linspace(ylim(1),ylim(2),gridSize);
end

[xGrid,yGrid] = meshgrid(x,y);
z0 = complex(xGrid,yGrid);
count = ones(size(z0),type);

z = z0;
for n = 0:maxIterations
    z = z.*z + z0;
    inside = abs(z) <= 2;
    count = count + inside;
end

count = log(count);
t = toc(t0);
end
function mandelbrot_example

% Run on CPU
[~; ~; ~; cpu_t] = calc_mandelbrot('double');

% Run on GPU
[~; ~; ~; gpu_t] = calc_mandelbrot('gpuArray');

fprintf('CPU time: %0.2f\n',cpu_t)
fprintf('GPU time: %0.2f\n',gpu_t)

end
function [time_cpu, time_gpu] = calc_fft_cpu_gpu(N)

matrix_cpu = rand(N);

tic
time_cpu = toc;
disp(['Total time on CPU: ' num2str(time_cpu)])

end

t0 = tic;
% Transfer matrix to GPU device
matrix_gpu = gpuArray(matrix_cpu);

t1 = tic;
out_gpu = fft(matrix_gpu);
time_gfft = toc(t1);

% Gather back from GPU to CPU
gather_gpu = gather(out_gpu);

% Wait for transfer to complete
wait(gpuDevice)
time_gpu = toc(t0);

disp(['GPU FFT: ' num2str(time_gfft)])
disp(['Total time on GPU: ' num2str(time_gpu)])

disp(['FFT speed improvement: ' num2str(time_cpu/time_gfft)])
disp(['Total speed improvement: ' num2str(time_cpu/time_gpu)])

whos matrix_cpu matrix_gpu
GPU example – FFT (2)

Why did the GPU code run faster the 2nd time?

% Matrix: 2GB
sz = 2^14;

% Use spmd block
spmd, [cpu_t, gpu_t] = calc_fft_cpu_gpu(sz); end
Lab 1:
- Total time on CPU: 11.9314
- GPU FFT: 1.888
- Total time on GPU: 25.95
- FFT speed improvement: 6.3195
- Total speed improvement: 0.45979

% Why will the GPU run faster the second time?
spmd, [cpu_t, gpu_t] = calc_fft_cpu_gpu(sz); end
Lab 1:
- Total time on CPU: 12.5073
- GPU FFT: 0.081633
- Total time on GPU: 19.848
- FFT speed improvement: 153.2134
- Total speed improvement: 0.63015

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix_cpu</td>
<td>16384x16384</td>
<td>2147483648</td>
<td>double</td>
</tr>
<tr>
<td>matrix_gpu</td>
<td>16384x16384</td>
<td>4</td>
<td>gpuArray</td>
</tr>
</tbody>
</table>
Submit single-node job & multi-node job

If we’re running a multi-node job, why did we only request 1 core?

Why are we asking for more walltime for a multi-node job?

#!/bin/sh

#PBS -l nodes=1:ppn=4
#PBS -l pmem=4gb
#PBS -l walltime=00:10:00
#PBS -j oe
#PBS -q pace-training

# 4 cores (1 node)
# 4 GB RAM per core
# 10 minutes
# Join output and error file

# Change to working directory
cd $PBS_O_WORKDIR

# Add MATLAB to system path
module load matlab/r2020b

# Run code
matlab -batch calc_pi

matlab-single-node.pbs

#!/bin/sh

#PBS -l nodes=1:ppn=1
#PBS -l pmem=4gb
#PBS -l walltime=00:20:00
#PBS -j oe
#PBS -q pace-training

# Change to working directory
cd $PBS_O_WORKDIR

# Add MATLAB to system path
module load matlab/r2020b

# Run code
matlab -batch calc_pi_multi_node

matlab-multi-node.pbs
Single-node job (1)
Single-node job (2)

How many MATLAB processes are running? 3? 4? 5?

```bash
-bash4.2
-bash4.2 head -30 matlab-single-node.pbs.099577

Start PBS Prologue Fri May 21 21:14:53 EDT 2021
Job ID: 99577.sched-pace-ice-pace.gatech.edu
User ID: rnorris34
Job name: matlab-single-node.pbs
Queue: pace-training
End PBS Prologue Fri May 21 21:14:53 EDT 2021

Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4)

Lab 1:
  Subinterval: [0 , 0.25]  Integral: 0.979915
Lab 2:
  Subinterval: [0.25 , 0.5 ]  Integral: 0.874676
Lab 3:
  Subinterval: [0.5 , 0.75]  Integral: 0.719414
Lab 4:
  Subinterval: [0.75 , 1 ]  Integral: 0.567588
pi : 3.14159265358979323846
Approximation: 3.141592653589793560
Error : 4.44089e-16

-bash4.2 cat calc_pi.m
function calc_pi
    c = parcluster('local');
    sz = str2num([getenv('SLURM_CPUS_PER_TASK') getenv('PBS_NP')]); % #<b>ST2NM</b>
    if isempty(sz), sz = maxNumCompThreads; end
    c.parpool(sz);

#!/bin/sh

#PBS -l nodes=1:ppn=4
#PBS -l pmem=4gb
#PBS -l walltime=00:10:00
#PBS -j oe
#PBS -q pace-training

# Change to working directory
cd $PBS_O_WORKDIR

# Add MATLAB to system path
module load matlab/r2020b

# Run code
matlab -batch calc_pi
```
Multi-node job (1)
Multi-node pool of workers (2)

bash4.2
bash4.2 head -40 matlab-multi-node.pbs 0.99580

Begin PBS Prologue Fri May 21 21:20:34 EDT 2021
Job ID: 99580.sched-pace-ice.pace.gatech.edu
User ID: mnorris34
Job name: matlab-multi-node.pbs
Queue: pace-training
End PBS Prologue Fri May 21 21:20:34 EDT 2021

Starting parallel pool (parpool) using the 'pace R2020b' profile ...

addionalSubmitArgs =

' -l nodes=1:ppn=12 -l pmem=4gb -q pace-training'

Connected to the parallel pool (number of workers: 12).
Lab 1:
  Subinterval: [0 , 0.0833333]
Lab 2:
  Subinterval: [0.0833333, 0.1666667]
Lab 3:
  Subinterval: [0.1666667, 0.25]
Lab 4:
  Subinterval: [0.25, 0.3333333]
Lab 5:
  Subinterval: [0.3333333, 0.4166667]
Lab 6:
  Subinterval: [0.4166667, 0.5 ]
Lab 7:
  Subinterval: [0.5 , 0.5833333]
Lab 8:
  Subinterval: [0.5833333, 0.6666667]
Lab 9:
  Subinterval: [0.6666667, 0.75]
Lab 10:
  Subinterval: [0.75, 0.8333333]
Lab 11:
  Subinterval: [0.8333333, 0.9166667]
Lab 12:
  Subinterval: [0.9166667, 1 ]
Lab 1:
  -bash4.2
Wait, what about Phoenix and Hive? . . .

- Phoenix and Hive operate the same way (i.e., calling configCluster)
  - Adjust the QueueName accordingly
- Phoenix
  - Also requires the AccountName, based on the PI’s user-id. For example

```python
>> c.AdditionalProperties.AccountName='GT-gburdell3';
```