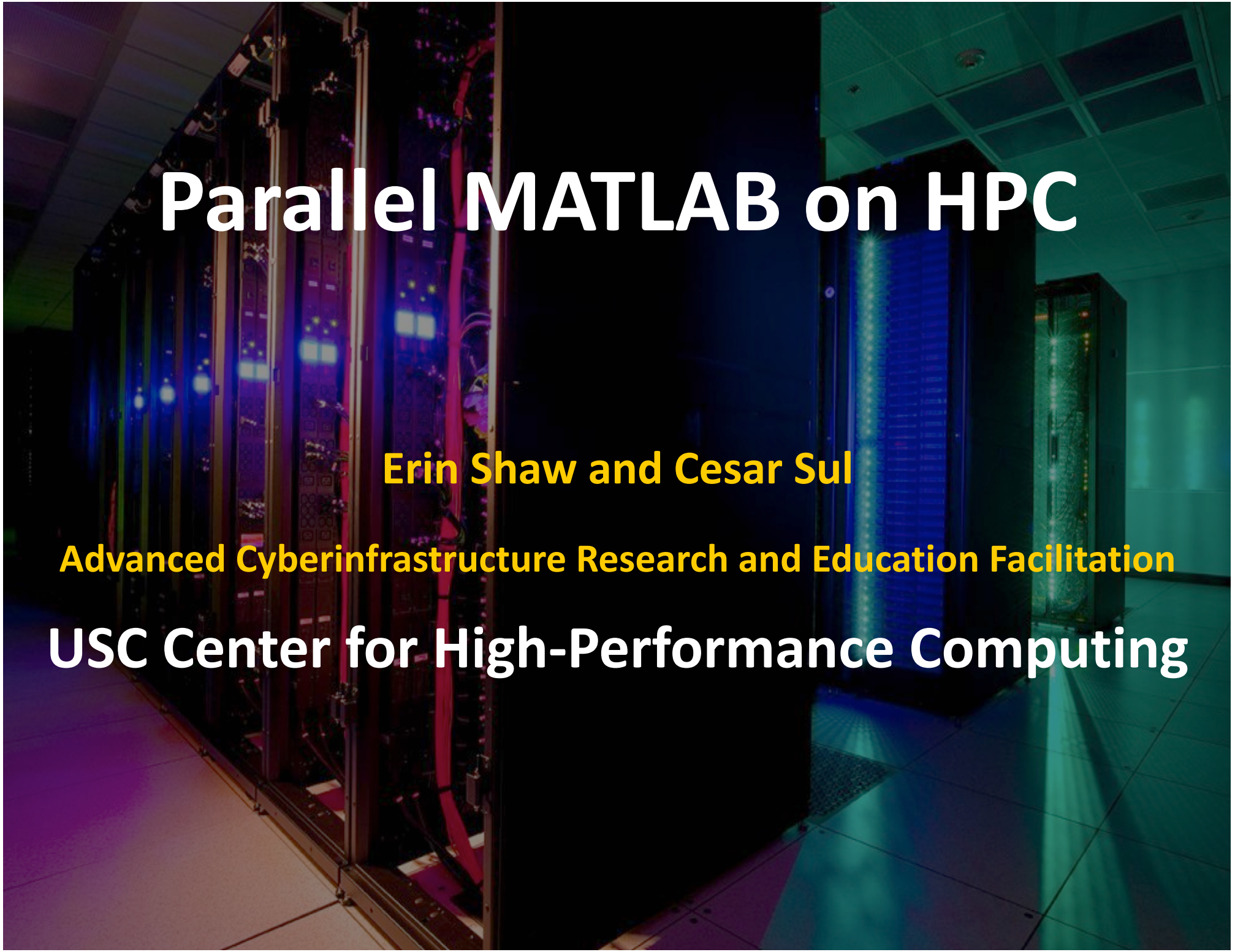


Parallel MATLAB on HPC

Erin Shaw and Cesar Sul

Advanced Cyberinfrastructure Research and Education Facilitation

USC Center for High-Performance Computing

Welcome HPC Workshop - Setup

- Sign in at goo.gl/Qan5Kg
- Best to have X display forwarding enabled
  - On Mac, download and install XQuartz from www.quartz.org
  - On PC, enable X forwarding within your ssh client
- Log in to hpc-login2 or hpc-login3 with -X or –Y option
  - Go to your /staging directory and create workshop space

```
$cd /staging/<project>/<user>
$mkdir matlab
```

Welcome HPC Workshop - Setup

- Slides and files used are here:
  - /home/rcf-proj/workshop/handouts/HPCParallelMatlab*.pdf
  - /home/rcf-proj/workshop/matlab/

- To copy slides to your laptop:
  - a) Use SFTP client like Filezilla
  - b) Open Terminal window, type cd ~', and then:

```
scp <your_NetId>@hpc-login3.usc.edu:/home/rcf-proj/
  workshop/handouts/HPCParallelMATLAB*.pdf <space> .
```

Note: This is all one command and there is a "dot"* at the end!
*"dot" is a symbol that resolves to/is replaced by the path to your current directory.
Note: This will download the file to your home directory (on your mac).

HPC Facilitation

- Request assistance
  - Email hpc@usc.edu (email again!)
  - Drop-in to office hours
    - UPC: every Tuesday, 2:30p (LVL 3M)
    - HSC: 1st & 3rd Thursdays (NML 203, SSB 106)
- Learn more!
  - Webpages online at https://hpcc.usc.edu
  - Request a consultation (Getting Help => Office Hours & Consultations)
  - Attend a workshop

Outline

- Introduction
  - Prerequisites, setup workspace and launch MATLAB
  - Introduction to parallel MATLAB and parfor
- Run MATLAB on single node with Parallel Computing Toolbox (PCT)
- Run MATLAB on multiple nodes with Parallel Server
- Run multiple MATLAB jobs with Slurm
- Beyond parfor
  - labIndex, Composite, and Broadcast&Receive

Outline

- **Introduction**
  - **Prerequisites, setup workspace and launch MATLAB**
  - Introduction to parallel MATLAB and parfor
- Run MATLAB on single node with Parallel Computing Toolbox (PCT)
- Run MATLAB on multiple nodes with Parallel Server
- Run multiple MATLAB jobs with Slurm
- Beyond parfor
  - labIndex, Composite, and Broadcast&Receive

Workspace setup

- Start in your /staging directory

```
$cd /staging/<project>/<user>
$mkdir matlab
```

- Make a workshop directory and copy MATLAB examples

```
$cd matlab
$cp –r /home/rcf-proj/workshop/matlab/* .
$ls
helper_script_examples/  standalone_examples/
```

- Make a storage directory for temporary working files

```
$mkdir matlab_storage
$ls
```

Request an interactive compute node

```
$ salloc --ntasks=1 --cpus-per-task=8 --time=1:00:00 --mem-per-cpu=2gb
```

```
--ntasks=1              # keep all cpus on 1 node
--cpus-per-task=16      # use 16 cpus
--time=1:00:00          #  request up to an hour interactive session
--mem-per-cpu=2g        #  request 2GB of memory per cpu
```

Wait for compute node

```
[tt@hpc-login3 matlab]$ salloc --ntasks=8 --time=1:00:00 --mem-per-cpu=2g
salloc: Pending job allocation 1084759
salloc: job 1084759 queued and waiting for resources
salloc: job 1084759 has been allocated resources
salloc: Granted job allocation 1084759
salloc: Waiting for resource configuration
salloc: Nodes hpc3779 are ready for job
----------------------------------------
Begin SLURM Prolog Thu Jun 21 14:52:49 2018
Job ID:        1084759
Username:      tt
Accountname:   lc_tt1
Name:          sh
Partition:     quick
Nodes:         hpc3779
TasksPerNode:  8
CPUSPerTask:   Default[1]
TMPDIR:        /tmp/1084759.quick
SCRATCHDIR:    /staging/scratch/1084759
Cluster:       uschpc
HSDA Account:  false
End SLURM Prolog
----------------------------------------
[tt@hpc3779 matlab]$ source /usr/usc/matlab/R2018b/setup.sh
[tt@hpc3779 matlab]$ matlab -nodisplay
```

Outline

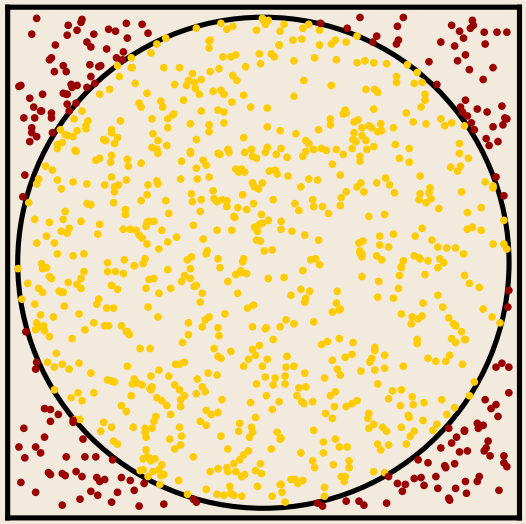
- **Introduction**
  - Prerequisites, setup workspace and launch MATLAB
  - **Introduction to parallel MATLAB and parfor**
- Run MATLAB on single node with Parallel Computing Toolbox (PCT)
- Run MATLAB on multiple nodes with Parallel Server
- Run multiple MATLAB jobs with Slurm
- Beyond parfor
  - labIndex, Composite, and Broadcast&Receive

Parallel MATLAB

- USC has a site-wide license for MATLAB
  - Students can use laptop version free of charge
  - HPC has 1000 licenses for researchers
- HPC license includes use of:
  - MATLAB Parallel Computing Toolbox (PCT) for single-node multi-core jobs
  - MATLAB Parallel Server for multi-node multi-core jobs
- HPC use requires special scripts for Slurm integration
  - MATLAB and Slurm now integrated (>=2018a)
  - /usr/usc/matlab/R2018b/SlurmIntegrationScripts

Parallel for loop (parfor)

```
% EstimatePi.m
% Calculate the value of Pi using a Monte Carlo simulation

max=1e9;
tic;
n=0;
parfor i = 1:max
  x=rand;
  y=rand;
  if (x^2 + y^2 < 1.0)
     n=n+1;
  end
end
elapsedTime = toc;
pi = (4.0 * n / max);
```



Outline

- Introduction
  - Prerequisites, setup workspace and launch MATLAB
  - Introduction to parallel MATLAB and parfor
- **Run MATLAB on single node with Parallel Computing Toolbox (PCT)**
- Run MATLAB on multiple nodes with Parallel Server
- Run multiple MATLAB jobs with Slurm
- Beyond parfor
  - labIndex, Composite, and Broadcast&Receive

Run on single node with PCT

- Simplest way to get started is to add these lines to top of your script

```
cluster=parallel.cluster.Local();
cluster.JobStorageLocation='storage_dir';
[a,b]=evalc('system(''nproc --all'')');
cluster.NumWorkers=str2num(a);
```

  - **storage_dir** is absolute path to directory for temporary files

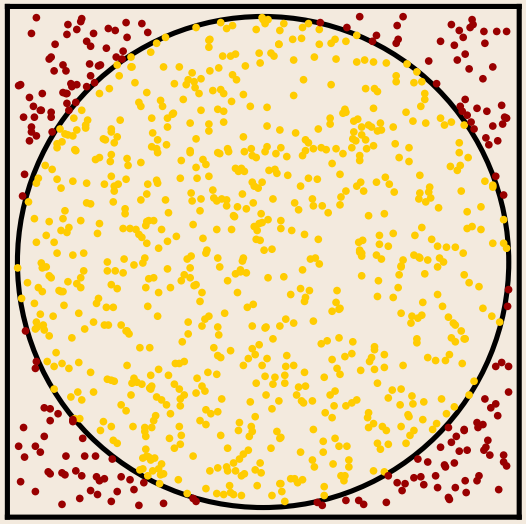
- Start up worker pool with this line in script

```
pool=parpool(cluster,N)
```

  - The value of **N** must be 1 less than total number of cpus allocated

Create parallel cluster and start parallel pool

```
% EstimatePi.m
% Calculate the value of Pi using a Monte Carlo simulation
cluster=parallel.cluster.Local();
cluster.JobStorageLocation=storage_dir;
[a,b]=evalc('system(''nproc --all'')');
cluster.NumWorkers=str2num(a);

pool=parpool(cluster,N)

max=1e9;
tic;
n=0;
parfor i = 1:max
   x=rand;
   y=rand;
   if (x^2 + y^2 < 1.0)
      n=n+1;
   end
end
elapsedTime = toc;
pi = (4.0 * n / max);
```



Run EstimatePi.m

- Replace **`storage_dir`** in standalone_examples/estimatePi.m
- Launch MATLAB and run from command line

```
$ matlab -nodisplay -r "estimatePi"

                            < M A T L A B (R) >
Starting parallel pool (parpool) using the 'local' profile ...
connected to 16 workers.
ans =
Pool with properties:
            Connected: true
           NumWorkers: 16
              Cluster: local
        AttachedFiles: {}
          IdleTimeout: 30 minute(s) (30 minutes remaining)
          SpmdEnabled: true
ans =
Elapsed time = 91.211017, pi = 3.1415924292
```

PCT Helper Script

- You can simplify the setup by using helper scripts

```
get_LOCAL_cluster(storage_dir)

pool=parpool(cluster,N)
```

  - **`storage_dir`** is absolute path to directory for temporary files
  - The value of **`N`** must be 1 less than total number of cpus allocated
- You must define `get_LOCAL_cluster`
  - There is a copy of `get_LOCAL_cluster.m` in helper_script_examples
  - Copy it to your ~/matlab or working directory to use

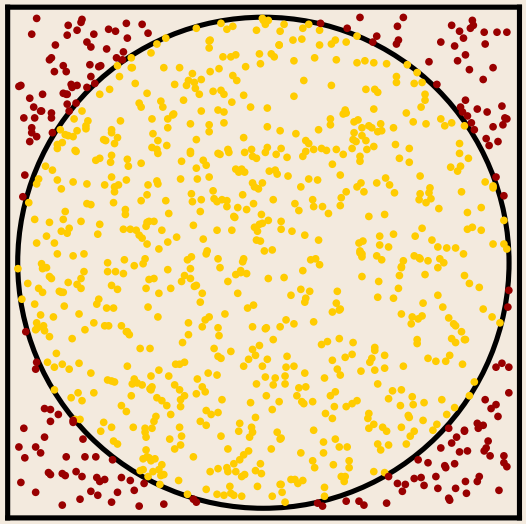
```
cp /home/rcf-proj/workshop/matlab/get_LOCAL_cluster.m
~/matlab/
```

get_LOCAL_cluster.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get_LOCAL_cluster(storage_dir)
% HPC helper function to create a single node MATLAB cluster under Slurm.
% Place this file in MATLAB's search path and call it from your program.
% Or place the lines of the function directly within your program.
%
% Arguments:
%  storage_dir: The path to an *existing* directory where MATLAB can store
%               files, e.g., "/home/rcf-proj/tt/ttrojan/matlab_storage".
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function cluster = get_LOCAL_cluster(storage_dir)
   cluster=parallel.cluster.Local();
   cluster.JobStorageLocation=storage_dir;
   [a,b]=evalc('system(''nproc --all'')');
   cluster.NumWorkers=str2num(a);
end
```

PCT Helper Script: EstimatePi.m

```
% Calculate the value of Pi using a Monte Carlo simulation

get_LOCAL_cluster(storage_dir)
pool=parpool(cluster,N)

max=1e9;
tic;
n=0;
parfor i = 1:max
   x=rand;
   y=rand;
   if (x^2 + y^2 < 1.0)
      n=n+1;
   end
end
elapsedTime = toc;
pi = (4.0 * n / max);
```



Submit EstimatePi.m to queue

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=1g
#SBATCH --time=00:20:00
#SBATCH --export=none

source /usr/usc/matlab/R2018a/setup.sh

#Suppress time warning
export TZ=America/Los_Angeles

matlab  -nodisplay -r "estimatePi"
```

Outline

- Introduction
  - Prerequisites, setup workspace and launch MATLAB
  - Introduction to parallel MATLAB and parfor
- Run MATLAB on single node with Parallel Computing Toolbox (PCT)
- **Run MATLAB on multiple nodes with Parallel Server**
- Run multiple MATLAB jobs with Slurm
- Beyond parfor
  - labIndex, Composite, and Broadcast&Receive

Run on multiple nodes with Parallel Server

- R2018/home/usc/matlab/R2018b/SlurmIntegrationScripts
- Simplest way to get started, add this to top of Matlab script

```
cluster = parallel.cluster.Generic;
set(cluster,'JobStorageLocation', 'storage_dir');
set(cluster,'HasSharedFilesystem', true);
set(cluster,'IntegrationScriptsLocation','scripts_dir');
cluster.AdditionalProperties.SlurmArgs='sbatch_args';
```

storage_dir	Path to where temp Matlab files are saved
scripts_dir	Path to Slurm integration scripts: /usr/usc/matlab/**version**/SlurmIntegrationScripts
sbatch_args	Arguments for job scheduler

Run on multiple nodes

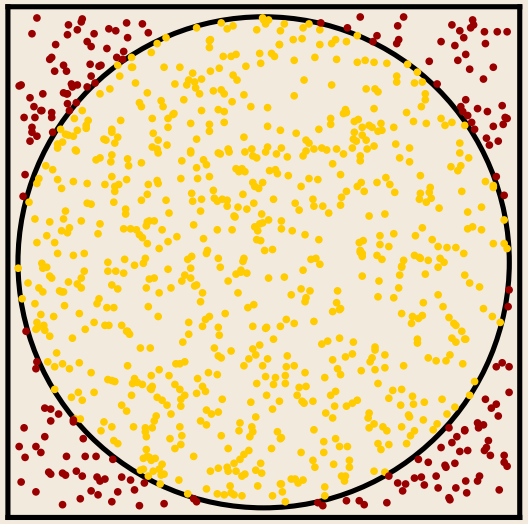
- Matlab will launch a **second** job after the parpool command

```
pool=parpool(cluster,N)
```

- Memory requests, wallclock time specified in **`sbatch_args`**
- Matlab will use **`N`** to determine number of cpus to request.

Setup cluster and start parallel pool

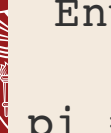
```
% Calculate the value of Pi using a Monte Carlo simulation
cluster = parallel.cluster.Generic;
set(cluster,'JobStorageLocation', storage_dir);
set(cluster,'HasSharedFilesystem', true);
set(cluster,'IntegrationScriptsLocation',scripts_dir);
cluster.AdditionalProperties.SlurmArgs=sbatch_args;

pool=parpool(cluster,N)

max=1e9;
tic;
n=0;
parfor i = 1:max
   x=rand;
   y=rand;
   if (x^2 + y^2 < 1.0)
      n=n+1;
   end
end
elapsedTime = toc;
pi = (4.0 * n / max);
```



Run EstimatePi.m

- Launch Matlab and run from command line

```
$ matlab -nodisplay -r "estimatePi"

                            < M A T L A B (R) >
Warning: The IntegrationsScriptsLocation
/usr/usc/matlab/R2018a/SlurmIntegrationScripts has been put at
the top of the path.
connected to 7 workers.
ans =
Pool with properties:
            Connected: true
           NumWorkers: 7
              Cluster: Generic
        AttachedFiles: {}
    AutoAddClientPath: true
          IdleTimeout: 30 minutes (30 minutes remaining)
          SpmdEnabled: true
 EnvironmentVariables: {}

pi = 3.1416, elapsedTime = 2.5654
```



Parallel Server Helper Script

- If you want your scripts to be simple

```
get_SLURM_cluster(storage_dir,scripts_dir,sbatch_args)
pool=parpool(cluster,N)
```

storage_dir	Path to where temp Matlab files are saved
scripts_dir	Path to Slurm integration scripts: /usr/usc/matlab/**version**/SlurmIntegrationScripts
sbatch_args	Arguments for job scheduler
N	Number of workers to use

- Copy `get_SLURM_cluster.m` to your home directory

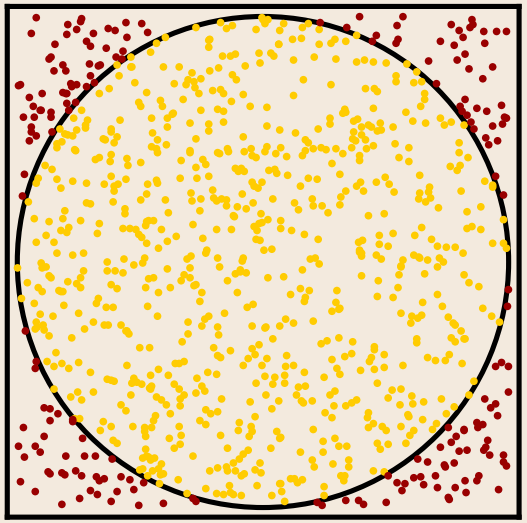
```
cp /home/rcf-proj/workshop/matlab/get_SLURM_cluster.m ~/matlab/
```

get_SLURM_cluster.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get_SLURM_cluster(storage_dir,scripts_dir,sbatch_args)
% HPC helper function to create a multi-node MATLAB cluster under Slurm.
% Place this file in MATLAB's search path and call it from your program.
% Or place the lines of the function directly within your program.
%
% Arguments:
%  storage_dir: The path to an *existing* directory where MATLAB can store
%               files, e.g., "/home/rcf-proj/tt/ttrojan/matlab_storage".
%  scripts_dir: The path to the /usr/usc/matlab//SlurmIntegrationScripts
%               directory, e.g.,
"/usr/usc/matlab/R2018a/SlurmIntegrationScripts".
%  sbatch_arg:  A string to pass to slurm with all slurm options *except*
%               ntasks, e.g., '--time=12:00:00 --partition=scec --mem-per-
cpu=2G'.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function cluster = get_SLURM_cluster(storage_dir,scripts_dir,sbatch_args)
   cluster = parallel.cluster.Generic;
   set(cluster,'JobStorageLocation', storage_dir);
   set(cluster,'HasSharedFilesystem', true);
   set(cluster,'IntegrationScriptsLocation',scripts_dir);
   cluster.AdditionalProperties.SlurmArgs=sbatch_args;
end
```

Helper Script: estimatePi.m

```
% Calculate the value of Pi using a Monte Carlo simulation
cluster = get_SLURM_cluster('storage_dir', ...
'/usr/usc/matlab/R2018a/SlurmIntegrationScripts', ...
'--mem-per-cpu=1g --time=00:30:00');

pool=parpool(cluster,N)

max=1e9;
tic;
n=0;
parfor i = 1:max
   x=rand;
   y=rand;
   if (x^2 + y^2 < 1.0)
      n=n+1;
   end
end
elapsedTime = toc;
pi = (4.0 * n / max);
```



Submit EstimatePi.m to queue

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=1g
#SBATCH --time=00:20:00
#SBATCH --export=none

source /usr/usc/matlab/R2018a/setup.sh

#Suppress time warning
export TZ=America/Los_Angeles

matlab  -nodisplay -r "estimatePi"
```

Outline

- Introduction
  - Prerequisites, setup workspace and launch MATLAB
  - Introduction to parallel MATLAB and parfor
- Run MATLAB on single node with Parallel Computing Toolbox (PCT)
- Run MATLAB on multiple nodes with Parallel Server
- **Run multiple MATLAB jobs with Slurm**
- Beyond parfor
  - labIndex, Composite, and Broadcast&Receive

Slurm Job Array

- If one node of your parallel cluster fails, the whole cluster fails.
  - This can be a problem.
  - If your workers are running independent calculations, you can use Slurm's job array construct instead
  - To run the following 100 times: sbatch --array=1-100 myjob.sl

    $ matlab –nodisplay –r "mymatlab"
- Partitions have per user limits on cpu, job and mem allocations
  - Main: Per user: up to 200 cpus, 10 jobs
  - Scavenge: Per user: up to 500 cpus, 2500G RAM (no limit on jobs)
- When limits are reached your job(s) go to the back of the queue
  - They are still in the queue!
  - You do not have to manage them, they will eventually run

Slurm Job Array

$ cat matlab.sl

```
#!/bin/bash
#SBATCH --ntasks=100
cd $SLURM_SUBMIT_DIR
source /usr/usc/matlab/R2018b/setup.sh
matlab –nodisplay –r “disp(‘hello’);exit”
```

$ sbatch --array=1-10 --partition=scavenge matlab.sl

- What happens?
  - Hits max CPUs after 5 jobs
- Now set --ntasks=**10** and –array=1-**100,** what happens?
  - Would hit max CPUs after 50 jobs. Instead hits max jobs at 10 jobs

Slurm Job Array

$ sbatch --array=1-100 --partition=scavenge matlab.sl #ntasks=100

$ squeue -u $USER

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  3724946_[1-100]  scavenge   array  ttrojan PD       0:00     13 (Priority)
```

$ squeue -u $USER

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  3724946_[6-100]  scavenge   array  erinshaw PD       0:00     13 (Priority)
```

Outline

- Introduction
  - Prerequisites, setup workspace and launch MATLAB
  - Introduction to parallel MATLAB and parfor
- Run MATLAB on single node with Parallel Computing Toolbox (PCT)
- Run MATLAB on multiple nodes with Parallel Server
- Run multiple MATLAB jobs with Slurm
- **Beyond parfor**
  - **labIndex, Composite, and Broadcast&Receive**

Beyond Parfor

- Parfor is the most basic way that Matlab can parallelize your code
- Sometimes you'll need to have more control over what gets run on each worker
- Here are some examples to give you an idea of the capabilities of the Parallel MATLAB
- Files saved in /home/rcf-proj/workshop/matlab/

submitTasks.m

```
% Demo create job and task and submit

% Create cluster object (single or multi node)
cluster=get_LOCAL_cluster(storage_dir)


% create cluster with 2 workers
cluster.parpool(2)


% create job
j = c.createJob;

a = rand(4);
% create task
j.createTask(@sin,1,{a});
j.createTask(@cos,1,{a});

% have two independent jobs to submit
j.submit;

%wait till job finish
j.wait('finished');
pause(20);

% fetch output
result = j.fetchOutputs;
f = [result{:}];

% save data
save('submitTasks.out','f', '-ascii' );
delete(gcp('nocreate'))
```

labIndex.m

```
% Demo of parpool, Composite and labindex

% Create cluster object (single or multi node)
cluster=get_LOCAL_cluster(storage_dir)

% request 3 workers
cluster.parpool(3)
```

```
% compute process
a =Composite();
spmd
  temp=labindex*ones(10);
  a = temp*temp;
end

% save worker results
d=[a{:}];
Save('labIndex.out', 'd', '-ascii' );


%cleanup
delete(gcp('nocreate') )
```

broadcastReceive.m

```
% Demo labindex and mpi communication

% Create cluster object (single or multi node)
cluster=get_LOCAL_cluster(storage_dir)

% request workers
cluster.parpool(3)

disp('Test one')
spmd
switch labindex
  case 1
    A = 1:5;
    labSend(A,2,199); %send to worker 2
  case 2
     A = labReceive(1,199); %receive from worker 1
  otherwise
     A = 11:15; %create his own
end
end

A{:}

%what does this do?
disp('Test two')

spmd
if labindex == 1
  A=labBroadcast (1,1:5); %broadcast to all
else
  A=labBroadcast (1); %receive broadcast
end
end

A{:}
delete(gcp('nocreate')) )
```

The End
Questions?

- To view licenses available
  - Type on command line

$ /auto/usc/matlab/<version>/etc/glnxa64/lmutil lmstat -a -c 27000@hpc-licenses.usc.edu | grep MATLAB

Users of MATLAB: (Total of 14146 licenses issued; Total of 4 licenses in use)

Users of MATLAB_Distrib_Comp_Engine: (Total of 1000 licenses issued; Total of 0 licenses in use)