Matlab with HPC

Burak Himmetoglu
ETS & CSC
bhimmetoglu@ucsb.edu

11/17/2016

ENTERPRISE TECHNOLOGY SERVICES | University of California Santa Barbara

CNSi UCSB MRL

CSC

UNIVERSITY OF CALIFORNIA SANTA BARBARA
CENTER FOR SCIENTIFIC COMPUTING

Question: My Matlab solution is taking too long on my computer, what can I do?

Possible answers:

- Try parallel computing toolkit ←
- Run your Matlab in a remote cluster ←
  - Large data that don’t fit your computer’s memory
  - Divide and Conquer

- Port your code to C/C++

Matlab on a remote computer cluster

Some (potential) drawbacks

- Almost all computer clusters run Linux
- For long calculations, you cannot use the IDE
- Need to be submitted to a queuing system

Advantages

- Access to a large memory (> 40 GB, up to 1 TB)
- Submit many calculations simultaneously!

Examples in this seminar

If you have an account on Knot:

```
export PATH="/sw/MatLab/R2016b/bin:$PATH"
```

Download the exercises from the command line:

```
svn checkout https://github.com/bhimmetoglu/talks-and-lectures/trunk/CSC-UCSB
```

All exercises are online:

https://github.com/bhimmetoglu/talks-and-lectures

Run Matlab code on Knot cluster

- Remember: No IDE for long calculations!
- Make sure that your code runs from start to end
- Perform tests on your computer first

A simple script (text file) can be used to submit to the queue:

```
#!/bin/bash
#PBS -l nodes=1:ppn=12
#PBS -l walltime=01:00:00
#PBS -N Pi
#PBS -V

cd $PBS_O_WORKDIR

matlab -nodisplay -nodesktop -nosplash < calculate_pi.m > out
```

Run Matlab code on Knot cluster

Let’s say that the name of the script is: `submit.job`

```
qsub submit.job
```

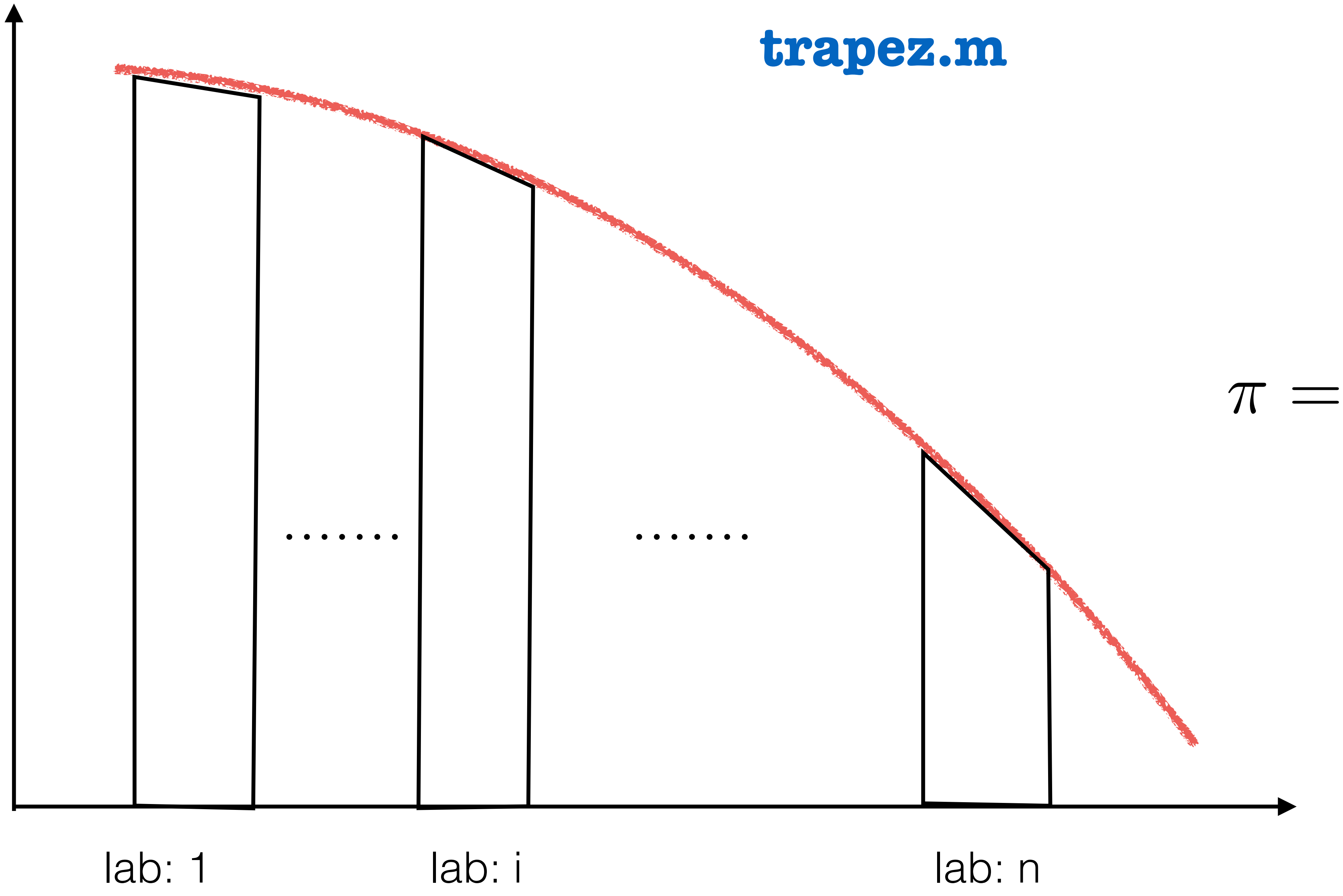
Better use the short queue, since this is a test job < 1 hr

```
qsub -q short submit.job
```

Check status:

```
showq -u $USER
```

Example 1: Calculate pi in parallel



$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

- Each “lab” (parallel thread) will compute the area of a trapezoid.

Example 1: Calculate pi in parallel

- Create parallel regions in your code
  - E.g.: **spmd**

```
% Start parallel region
spmd
  loc_a = (labindex -1)/numlabs; % labindex & numlabs are variables generated once spmd is called
  loc_b = labindex / numlabs;
  fprintf('Lab %d integrates oves [%f, %f] \n', labindex, loc_a, loc_b);
end
% End parallel region
```

- Work in parallel regions will be distributed across cores
- There is an overhead of launching parallel “labs”
- Performance gain is usually observed for jobs that run long enough

Example 1: Calculate pi in parallel

- Compute the area of the local trapezoid for each “lab”

```
% Start parallel region
spmd
  x = linspace(loc_a,loc_b,n); % Divide the local region into n intervals
  fx = f( x );                 % Get the values of the function on this sequence
  % Trapezoidal rule
  loc_result = (loc_b - loc_a) / 2.0 / (n-1) * ( fx(1) + fx(n) + 2 * sum(fx(2:n-1)) );
  fprintf (' Lab %d obtained: %f\n', labindex, loc_result );
end
% End parallel region
```

- Reduction: Collect results from all labs and add them up

```
% Start parallel region
spmd
  tot_result = gplus( loc_results );
end
% End parallel region
```

Example 2: Monte Carlo integral in parallel
spmd vs parfor

run_mcarlo_*.m

Monte Carlo integration:

$$Z = \int_0^1 \int_0^1 \dots \int_0^1 dx_1\, dx_2\, \dots dx_n\, e^{-x_1^2 - x_2^2 - \dots - x_n^2}$$

```
For (i = 1, NumSimulations){
        Pick $\{x_1, x_2, \dots, x_n\}$ randomly
        Z ⟵ (Volume of region) x (Integrand at $\{x_1, x_2, \dots, x_n\}$)
}
Average results (Z's)
```

Example 2: Monte Carlo integral in parallel

a) spmd

- Each “lab” runs a number of simulations for its own integral
- At the end, results from each lab averaged.

a) parfor

- The for loop over simulations are distributed across “labs”
- The distribution is automatic

```
% Start parallel region by parfor: Work will be divided automatically
parfor i = 1:nSim
        [v1, v2] = monteCarlo(nDim);
        z = z + v1;
        s2 = s2 + v2;
end
% End parallel region
```

Spmd vs parfor

- Parfor is much easier.
- Parfor determines potential issues (like race conditions) and will run serial if necessary.

- Spmd is more flexible, and allows more user control
- Careful code modification is usually necessary
- Race conditions?

Resources for Learning Matlab

- Coursera : https://www.coursera.org/learn/matlab
- LeanPub: https://leanpub.com/rprogramming
- Lynda : Up and Running with Matlab