Parallel Computing in Matlab

…

(and other fun things)

Shelley L. Knuth
Research Computing
CU-Boulder

Outline

- What is Matlab? What is parallel computing?
- Setting up Matlab in a serial environment
- Setting up Matlab in parallel
- Running interactive Matlab jobs

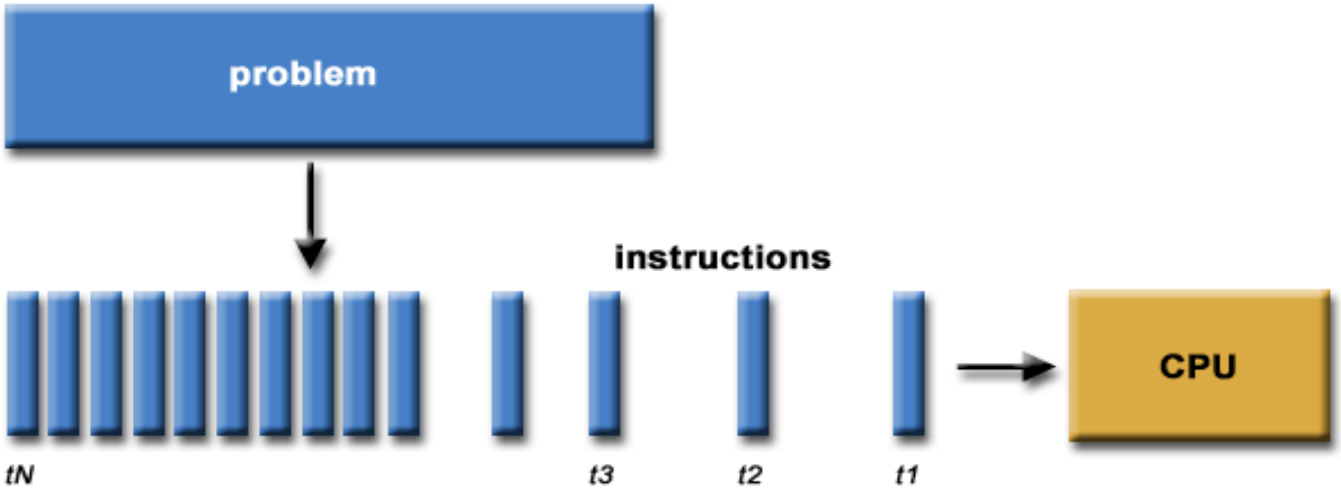
What Is Matlab?

- A high performance tool for technical computing
  - Integrates computation, visualization, and programming
  - Analyze data, develop algorithms, create applications

  www.mathworks.com
- Uses many specialized toolboxes to make our lives easier
- Used by many in the sciences, engineering, math
  - Academia and industry

Serial Processing

- Generally computer code is written such that tasks complete sequentially (serial processing)
  - Task 1 completes, Task 2 waits for Task 1 to finish, etc.
  - Computer has 1 CPU



Source: https://computing.llnl.gov/tutorials/parallel_comp/

Running Serial Matlab Jobs

- No trick to running a serial Matlab job vs. another job
- Serial job
- Run on laptop, HPC

PBS Directives

- matlab_test_serial.sh
- PBS Directives
  - Appear as headers at the beginning of a script
  - Specify the resource requirements of your job
  - Specify specific attributes of your job as well

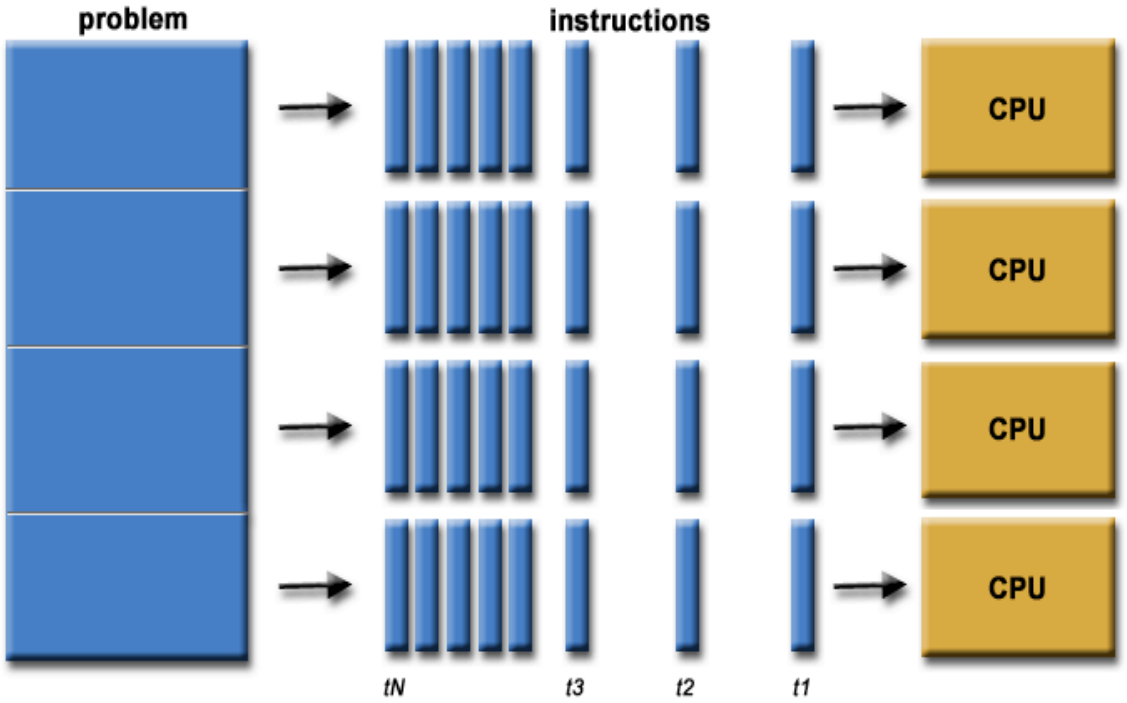
```
#PBS -l nodes=1:ppn=1,walltime=00:10:00

#PBS -N matlab_test_serial

#PBS -q janus-debug

#PBS -m be

#PBS -M shelley.knuth@colorado.edu

#PBS -j oe
```

What Is Parallel Processing?

- Parallel processing is the ability to carry out multiple operations or tasks simultaneously
  - Your brain does this
- For computers, simultaneous use of more than one CPUs or processor cores
- Programs run faster because multiple units running it

Parallel Processing

- In parallel processing we use several CPUs to solve one problem
- One node with several cores
- Several nodes with many cores
- Embarrassingly parallel



Source: https://computing.llnl.gov/tutorials/parallel_comp/

Parallel Processing Musts

- Need to be able to break the problem up into parts that can work independently of each other
- Need to also be able to be worked on simultaneously
  - Can't have the results from one CPU depend on another at each time step

Parallel Processing – Why and When?

- Why?
  - Allows you to solve problems faster
    - Save time and money
  - Solve large problems
    - Can't solve everything on your laptop! Even if it's a Mac!
- When?
  - Examples
    - Monte Carlo simulations (repeated random sampling)
    - Numerical Weather Prediction sensitivity studies

Cannot Use Parallel Processing

- Below is an example of when we **cannot** use parallel computing

```
>> for i=1:10
      x(i)=x(i-1)*x(i-2)
   end
```

Parallel Computing Toolbox (PCT)

- Additional toolbox as part of Matlab
- Perform parallel computations on multicore computers, GPUs, and computer clusters
- Allows you to parallel Matlab without MPI programming
- Many Matlab functions work in concert with the PCT
- Simple to utilize with just the use of certain commands

Parallel and Not Parallel

Not Parallel:

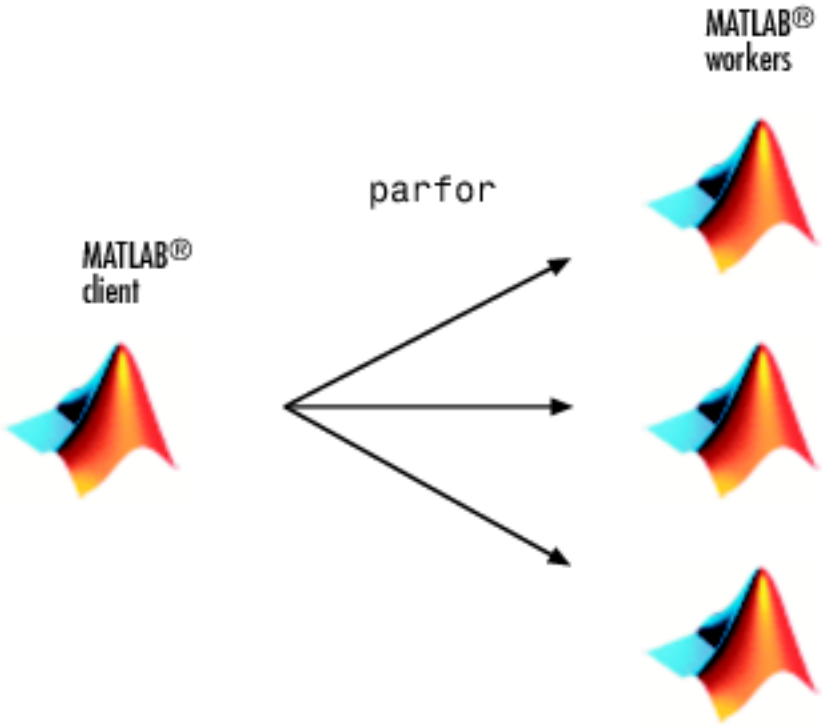
```
for i=1:10
   x=x(i)+1;
end
```

Parallel:

```
matlabpool open 4
   parfor i=1:10
      x=x(i)+1;
   end
matlabpool close
```

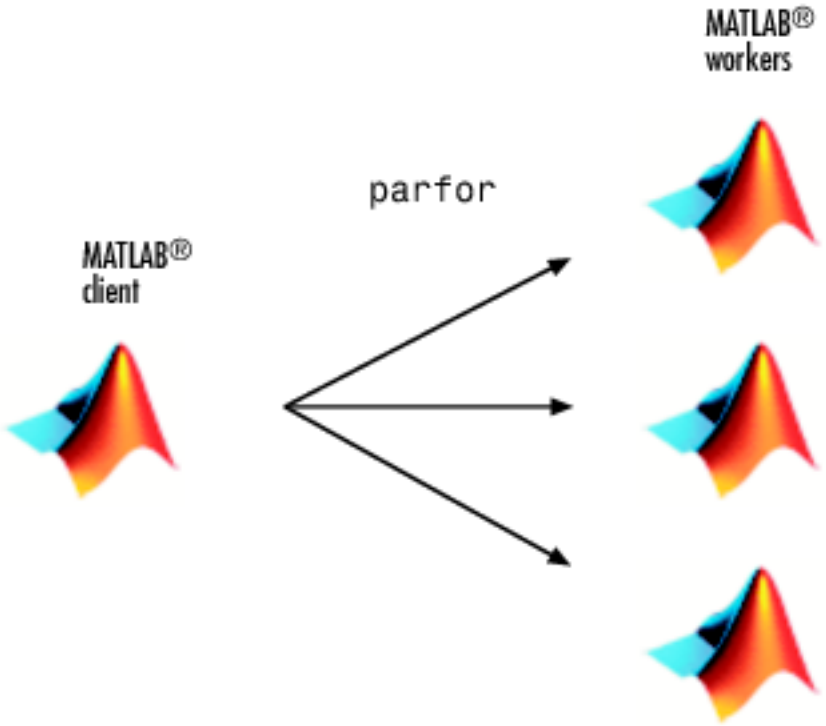
Running Matlab in Parallel

- Easy to convert code from serial to parallel
- Just run commands in PCT
  - parfor
  - spmd
- **Workers:** copies of the original client created to assist in computation



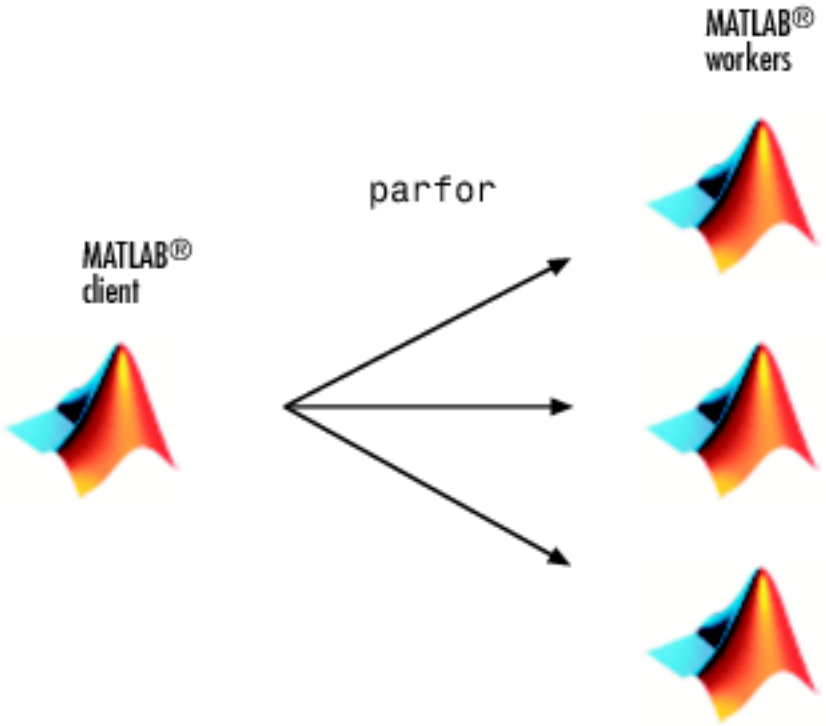
Running Matlab in Parallel

- On Janus at CU can run up to 12 workers on one node
  - Used to be able to only run one Matlab job at a time
  - Now can run as many as you want
  - Can also run as many workers on the high memory nodes



parfor

- Easy to use
- Allows parallelism in terms of loops
- When client reaches a parfor loop iterations of loop are automatically divided up among workers
- Parfor requires results be completely independent
- Cannot determine how loops are divided
- Don't know which worker runs which iteration
- Memory is shared (copied and returned) and workers are anonymous



Running Matlab in Parallel

- Let's take ordinary code that is already running and convert it to run in parallel
- matlab_parallel_tutorial.m

Running Matlab in Parallel On Lots of Cores

- We found that it takes 7 seconds to open the matlabpool on my laptop
- 11 seconds to run the code
- 18 seconds total
- 18 seconds to run the parforloop without opening the matlab pool

Running Matlab in Parallel On Lots of Cores

- Let's see what we can do on Janus where we'll have more cores
- First, let's discuss running a job interactively

Interactive Matlab

- Can also run Matlab interactively on supercomputing systems
- Use this if interested in testing commands and scripts from the command line
- Not a good idea to run Matlab Desktop
- Use the Janus supercomputer to provide an example

Interactive Matlab

- Once you're logged into the systems, submit the following command:

  qsub –I –q janus-debug –l nodes=1:ppn=12

  -I runs the job interactively

  -q janus-debug: specifies the queue

  -l nodes=1:ppn=12: specifying to use all the cores on one node

- Want to use all cores on one node because Matlab is intensive

Interactive Matlab

- Once the qsub command is submitted, must wait for enough space to become available
- Once this occurs, logged into a compute node
- Start Matlab without the desktop

  matlab –nosplash –nodesktop

- Don’t forget to exit!!

Running Matlab in Parallel On Lots of Cores

- On Janus we see that if we run serially it takes longer to run
  - ~35 seconds
- If we use matlabpool to open all workers (32) on a node, it takes a long time (~17 seconds)
  - ~2 seconds to run the code
  - 19 seconds total
- If we open matlabpool on less workers, i.e. 12, it takes ~12 seconds to open matlabpool
  - ~4 seconds to run the code
  - 16 seconds total

Scalability

- http://nbviewer.ipython.org/github/ResearchComputing/meetup_spring_2014/blob/master/notebooks/lecture_16_parallel.ipynb

Important Notes

- Cannot nest parfor loops
- A parfor loop cannot contain break or return statements
- Body of parloop cannot contain global variable declarations

Spmd Command

- The spmd command ensures more control over parallel computing jobs
- Like a very simplified version of MPI
  - One client process, supervising workers that collaborate on a single program
  - Define which tasks are assigned to each worker
  - Executes spmd body on several Matlab workers simultaneously
  - Not as easy to implement as parfor

Spmd Command

- Each worker (lab) has an identifier
  - labindex
- Knows the total number of workers and knows its tasks based on that number
- Each worker runs on a separate core and separate workspace
- Common program is used
- Workers meet at synchronization points
- Client can examine or modify data on any worker
- Communication between workers

Spmd Command

- Workers execute the spmd commands
  - Client idly watches
- Any variable defined by client is visible to the workers and can be used when doing calculations
- Variables defined by workers are composite variables
  - Each worker has its own value
  - Accessible by the client using the worker's index
  - Not visible to other workers except through the client
- Variables in one spmd block are available in another block

Spmd Example

- Matlab_parallel_tutorial.m

Parfor vs. Spmd

- spmd command:
  - Worker communication if necessary
  - Pass data between different labs (workers)
    - LabSend and LabReceive
- Parfor command:
  - Runs for loops in parallel
  - No worker communication
  - Simpler

Parfor vs. Spmd

- spmd command:
  - Data is broadcast to workers at beginning and remains there until you pull it back
- Parfor command:
  - Data is broadcast to workers at start of every parloop
  - Starts from scratch