Parallel Computing - Matlab – Part 2

Shelley Knuth
shelley.knuth@colorado.edu

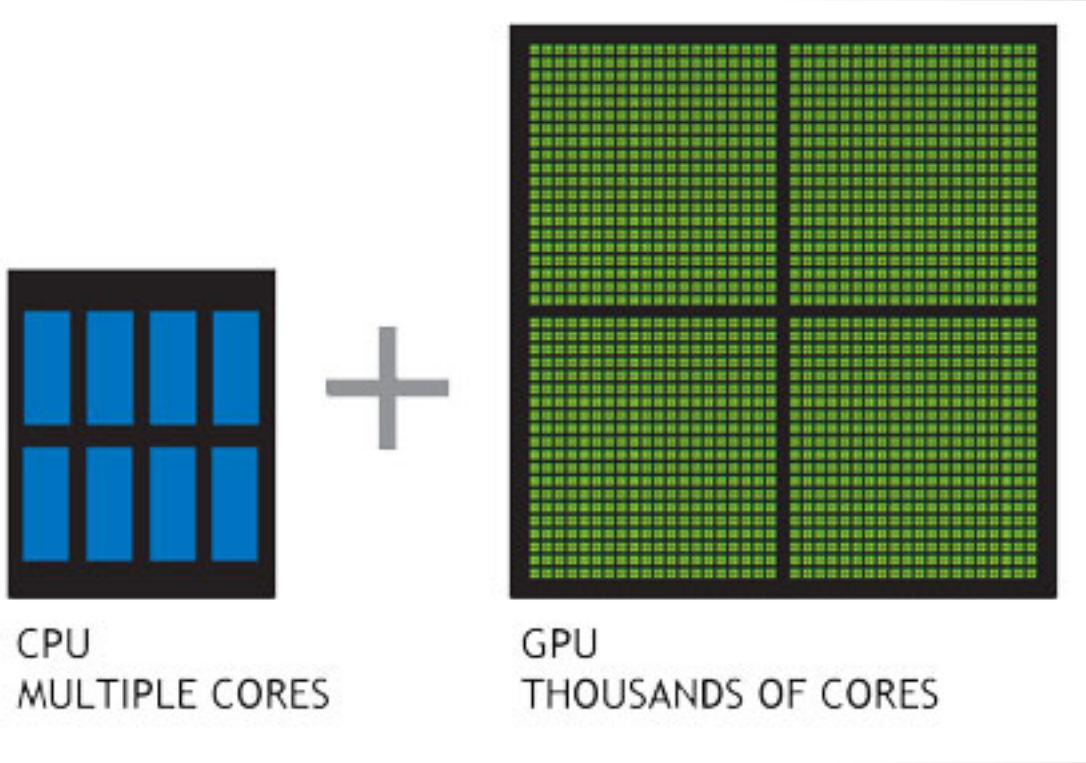
www.rc.colorado.edu

Slides: https://github.com/ResearchComputing/Parallelization_Workshop

Outline

- Continuing the PCT
  - spmd
- Distributed Arrays
- What are GPUs
  - Why do we want to use them
  - How can we leverage Matlab

    to use with GPUs?
- Cloud computing



Spmd Command

- Single program, multiple data
- Creates parallel regions of code
  - Can be useful to load up the data within spmd constructs so that it is available on the workers to later run a parfor loop
- The spmd command ensures more control
  - Can divide work and data between workers
  - Can communicate between workers
- Like a very simplified version of MPI

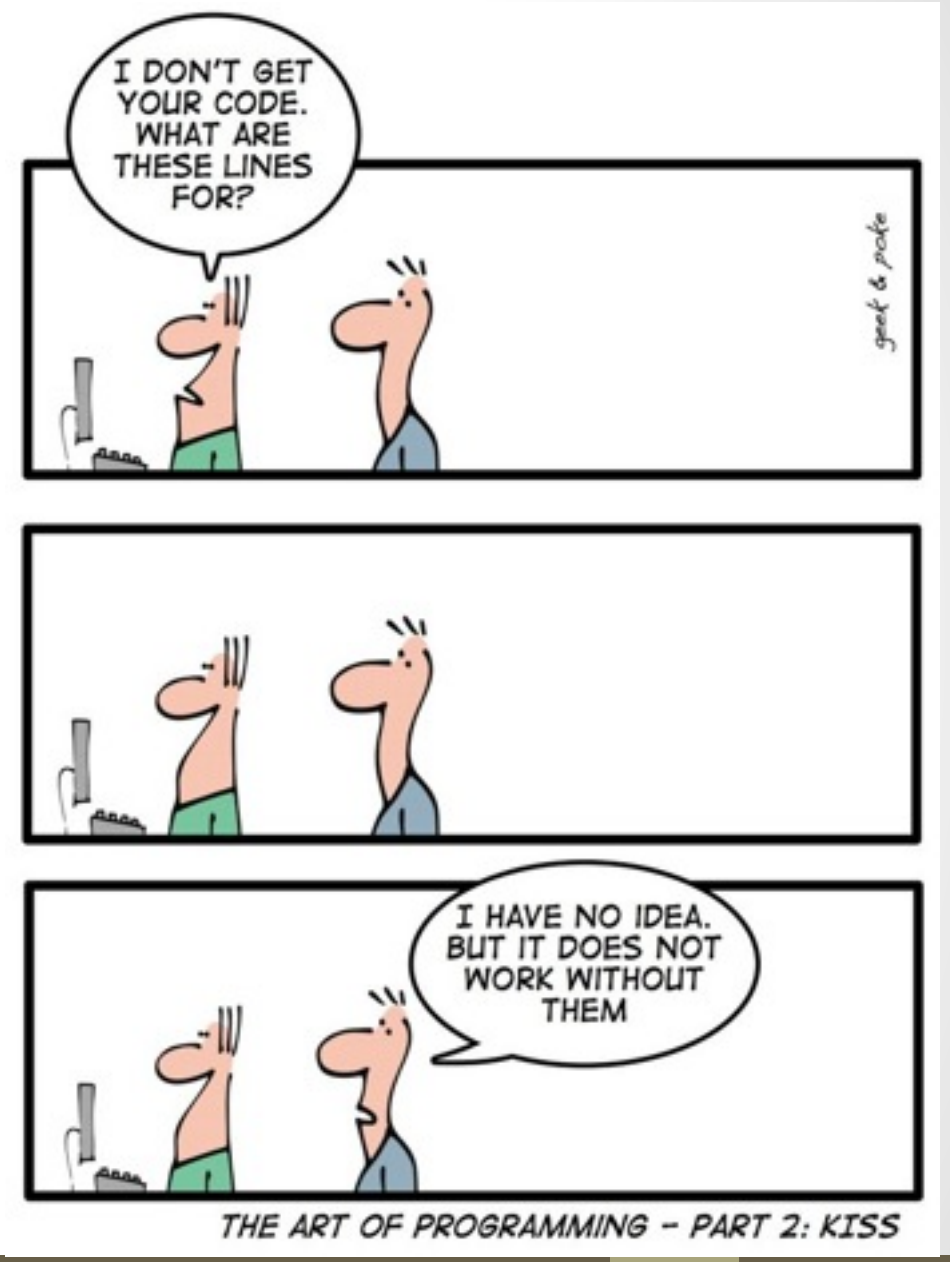
Spmd Command

- In spmd, have one client process
  - Supervises workers who work on a single problem
- The client keeps tracks of the workers via identifiers

  `labindex()`
- Each worker runs on a separate core but uses a common program
  - Meet and talk to each other at certain synchronization points
  - Two workers can communicate

Spmd Code

- Can have several spmd blocks in one program
- Workers workspace remains intact even if pause execution
- Variables will be shared between blocks



Composite Variables

- A composite variable contains references to unique values on each worker
- On a worker, it is accessed like a normal variable
- On the client elements on each worker are accessed using cell-array style notation

https://www.bu.edu/tech/files/2015/09/matlab_pct_slides.pdf

Data exchange between workers

- Labindex – returns unique identifier
- Numlabs – returns total number of workers
- LabReceive – allows a worker to receive data from another
- LabSend – one worker sends data to another
  - These two must be coupled
- LabSendReceive - simultaneous data exchange
- LabBroadcast - send/receive data from all labs
- LabBarrier - pause until all labs reach this call

```
a = 3;
b = 4;
spmd
    c = labindex();
    d = c + a;
end
e = a + d{1};
c{2} = 5;
spmd
    f = c * b;
end
```

Client a b e	Worker1 c d f	Worker2 c d f
3 - -	- - -	- - -
3 **4** -	- - -	- - -
3 4 -	**1** - -	**2** - -
3 4 -	1 **4** -	2 **5** –
3 4 **7**	1 4 -	2 5 –
3 4 7	1 4 -	**5** 5 –
3 4 7	1 4 **4**	5 5 **20**

http://www.icam.vt.edu/Computing/fdi_2012_spmd.pdf

Example Using LabSendReceive

- labsr.sh
- labsr.m

Your Turn

- Do you have a piece of code that you can utilize parallel Matlab?
- Take a few minutes to look at some of the code you're working on
- Can you use parfor or spmd? Is it parallelizable?

What Are GPUs?

- GPUs – Graphical Processing Units
- Originally for graphics, but realized can be used for any type of computing
- Accelerator
  - Has way more processors per card than a regular CPU
  - Which means… ->
- Great for data parallel operations
  - Same operation performed on different parts of an array
- Best for large data

When Should You Use a GPU?

- If your problem is massively parallel, you might experience significant speed up
  - Vectorized Matlab calculations can fit here too
- Computationally intensive
  - Time spent on computation is greater than time spent transferring data to/from GPU memory
- If these don’t apply, your job could end up running slower on a GPU

Matlab Computations on a GPU

- Workers are not spun up on a GPU like they are on a CPU
- The client sends instructions to the GPU
- There's also device memory where you can transfer data
  - If have started on a CPU, can transfer data to GPU and perform operations
    - `gpuArray`
  - Once finished with computations, then transfer the data back to the CPU
    - `gather`
- You will need the PCT to utilize GPUs

gpuArray example

- On the CPU

```
A1 = rand(3000,3000);
tic;
B1 = fft(A1);
time1 = toc;
```

- On the GPU

```
A2 = gpuArray(A1);
tic;
B2 = fft(A2);
time2 = toc;
```

- Here we are creating the data on the CPU and then using `gpuArray` to transfer it to the GPU
- The transfer time could end up bogging down your code

http://blogs.mathworks.com/loren/2012/02/06/using-gpus-in-matlab/

Example

- solveEquationCPU.m
- WaveEquationCPU.m

- solveEquationGPU.m
- WaveEquationGPU.m

Your Turn

- Take some of your code and convert a portion of it to run on a GPU

Distributed Arrays

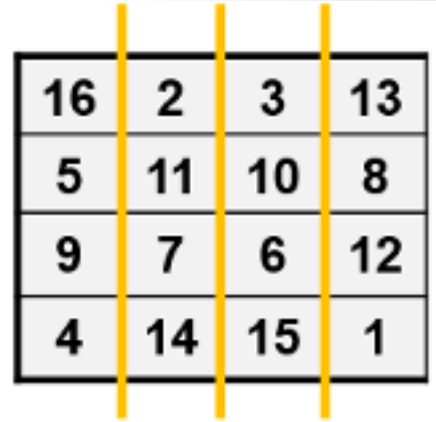
- You can use the `distributed` function to distribute any array that exists in your client workspace to the workers of a parallel pool
- The distributed array is a single variable split up over multiple workers
  - Can work with this variable as a single entity
  - The user is not concerned with how it is distributed
- Distributed arrays use the combined memory of multiple workers in a parallel pool to store the elements of an array
- This is a great way to scale up your big data computation

https://www.mathworks.com/help/distcomp/when-to-use-distributed-arrays.html

Distributed Arrays Example

- In the example below we will create an array in the client workspace then turn it into a distributed array

```
A = magic(4);                % Create magic 4-by-4 matrix
B = distributed(A);          % Distribute to the workers
```



https://www.mathworks.com/help/distcomp/when-to-use-distributed-arrays.html

Distributed and Co-Distributed Arrays

- Distributed arrays do not allow you to control how the data is distributed
- Co-distributed arrays do allow you this control

- Create a co-distributed array. 4 parts of array on worker 1 and 12 parts on worker 2
- Then create a 3x3x16 array of zeros

```
spmd
  codist = codistributor1d(3,[4,12]);
  Z = zeros(3,3,16,codist);
  Z = Z + labindex;
end
```

Matlab and the Cloud

- Very recently more available
- Mathworks supports several different types of cloud-based Matlab access
  - Matlab Mobile
    - Can connect to a Matlab session either on your local computer or in the cloud
    - Use from Apple® (iPhone, iPad) or Android devices
    - Have command line access
    - Can run scripts, create figures, view results
    - Everything up in cloud storage
  - Matlab Online – Matlab in a web browser
    - Can use Matlab from any computer and offers storage, sharing, and command execution
    - For students to use in a class

Matlab and the Cloud

- Distributed Computing Server
  - Scale Matlab computations on virtual clusters running in the cloud
  - Prototype your code in the desktop and then scale up to use virtual resources on a cloud computing service like Amazon EC2
  - There is an extra charge for this
  - However, can allow you to simulate the Distributed Computing Server to run across nodes

Questions?

- Email rc-help@colorado.edu
- Twitter: CUBoulderRC

- Link to survey on this topic: http://tinyurl.com/curc-survey16

- Slides: https://github.com/ResearchComputing/Parallelization_Workshop