Accelerate MATLAB Code using a GPU

By: Julian Landers

Contributions from Marek Stastna, Chris Subich

When to use a GPU?

- GPUs take advantage of code that is parallelizable.
- This means conducting the same computation on multiple data points.
- Parallel computing is not well suited to `'for'` loops (however parallel computations inside a for loop are okay).
- Code requiring lots of indexing data on an array stored on a GPU will perform poorly.
- It takes time to copy data from the CPU to the GPU, so small datasets will not benefit from computation on the GPU.
- Longer algorithms will see the benefit of the GPU for a smaller dataset, since the data needs to be copied only once then can be operated on for the entire computation.

How to use a GPU in MATLAB

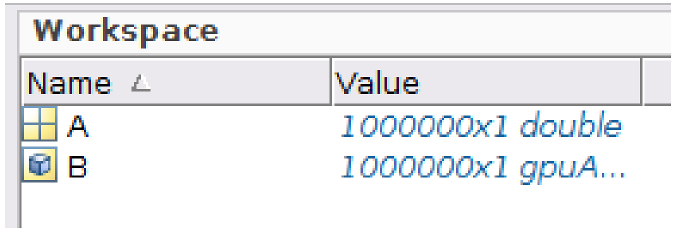
```
>> gpuDeviceTable

ans =

  1×5 table
```

Index	Name	ComputeCapability	DeviceAvailable	DeviceSelected
1	"NVIDIA GeForce RTX 3070 Ti"	"8.6"	true	false

- Before coding:
  - `gpuDeviceTable` lists all available GPU devices.
  - `gpuDevice(#)` selects which device to use. This command also clears the GPU's memory.
- `gpuArray`
  - Copies data to the GPU for computation. Many built-in MATLAB functions are compatible with gpuArray.
- `arrayfun`
  - Applies an input function to each element of an array. Behaves the same as the regular arrayfun but on the GPU. Input array should be a `gpuArray`.
- `parfor`
  - Computes for loop in parallel. This means that each iteration of the loop must not depend on the previous or any other iteration.

gpuArray

- Copies data to the device (GPU) from the host (CPU).
  - `A = rand(100,1); B = gpuArray(A);`
- Some functions also support creating data directly on the GPU.
  - `B = rand(100, 1, 'gpuArray');`
- Most MATLAB functions are 'enabled' for gpuArray so all that needs to be done is to pass a gpuArray to the function. Data created from a gpuArray will also be a gpuArray.
  - `C = max(B); D = isgpuarray(C);` returns `ans = 1`
- Some functions (lots of plotting functions) cannot use gpuArrays so they will need to be copied back to the host.
  - `E = gather(C);`
- The workspace section will show if an array is on the host or the device.



arrayfun

- Applies function to each element of array.
- Useful when a more complicated (not built-in) function is needed.

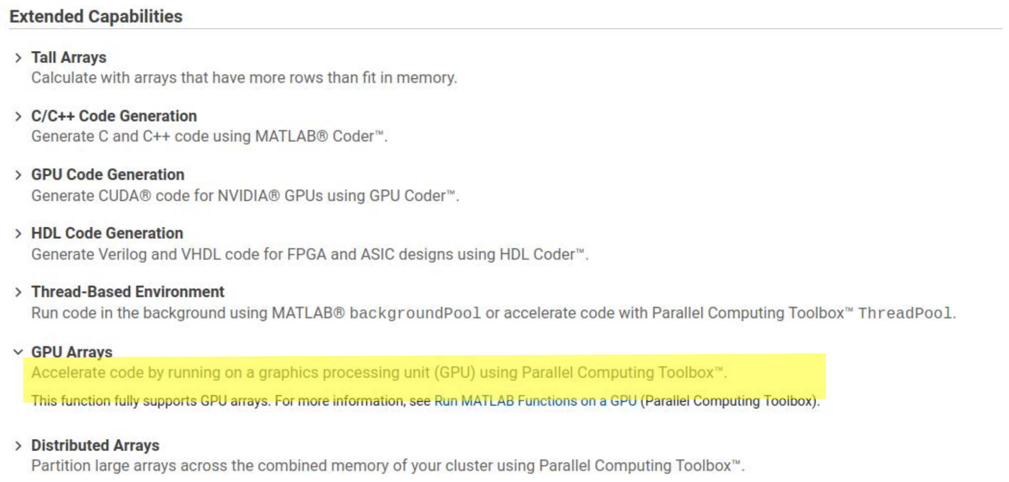
```
function [dist12 dist23] = calcDist(a, b, c)
    dist12 = sqrt(a^2 + b^2)
    dist23 = sqrt(b^2 + c^2)
end
A = rand(100, 1, 'gpuArray');B = rand(100, 1, 'gpuArray');
C = rand(100, 1, 'gpuArray');

[DIST12 DIST23] = arrayfun(@calcDist, A, B, C);
```

parfor

- Used to run a loop in parallel.
- Each iteration of the loop cannot depend on the previous one.
- The loop parameters must be integers.
- `parfor` loops cannot be nested.
- MATLAB will automatically divide up the iterations among the workers.

```
A = rand(100, 100); store_arr = zeros(100, 1);
parfor ii = 1:100
    store_arr(ii) = max(A(ii,:)) - min(A(ii,:));
end
```

Using gpuArray



- For built-in functions, under the "Extended Capabilities" section on the MATLAB Help Centre, the ability to handle a gpuArray is discussed.
- A list of all supported functions can be found here:
  - https://www.mathworks.com/help/referencelist.html?type=function&capability=gpuarrays
- Elements of a Cell Array can be stored on a GPU as well.

  `cell_arr{#} = gpuArray(A);`
- Remember to avoid lots of indexing such as `A(i) = A(i-1)*func`
- For longer methods, if the GPU runs out of memory, it can be cleared using `reset(gpuDevice(#));` where the # is the device number listed from `gpuDeviceTable`.

Simple example

```
A = rand(1e6, 1);
B = gpuArray(A);
tic
C = mod(A, 8);
CpuTime = toc
tic
D = mod(B, 8);
GpuTime = toc
Tst = isequal(C, D);
```

CpuTime is around 0.001940s

GpuTime is around 0.000138s

Tst logical returns 1

Timing test

- Four programs related to Lagrangian particle simulations are tested.
  - See https://github.com/darksc0ur/GPU-Particles
- The number of particles varies between programs since each program uses a slightly different method.
- Three different hardware setups are tested: two server grade GPUs and one desktop GPU.
- The RTX 3070Ti is connected to a Dell laptop with a Razer Core X external GPU enclosure.
  - https://www.razer.com/ca-en/gaming-egpus/razer-core-x

Comparison of GPUs used

Nvidia GeForce RTX 3070ti in GPU enclosure

- Ampere Architecture
- 8GB GDDR6
- Compute Capability 8.6
- Around $390 for the external enclosure and $850 for the GPU card

Nvidia Tesla P100 on Server

- Pascal Architecture
- 12GB HBM2
- Compute Capability 6.0
- Around $8000

Nvidia A100 on Server

- Ampere Architecture
- 40GB HBM2e
- Compute Capability 8.0
- Around $20000

Explanation of code

just_run_parts_red_and_psi.m

- Solves particle positions with a flow streamfunction using a Symplectic Euler method.
- Has a random noise included in the simulation which is solved using Bartosch's method for red noise.

Juliandist2.m

- Searches for particles that have "interacted" by considering all particles below a cutoff distance to have interacted.
- Calculates this by binning all the particles into bins the width of the cutoff distance. All particles in the same bin are said to have interacted.

Explanation of code

low_mem_dist.m

- Searches for particles that have "interacted" by considering all particles below a cutoff distance to have interacted.
- This is implemented by binning the particles, then explicitly computing the distance between all the particles in the bin, and between all the particles in adjacent bins.

low_mem_test_sort.m

- Searches for particles that have "interacted" by considering all particles below a cutoff distance to have interacted.
- This is implemented by binning the particles and using a binary search to find which particles are below the cutoff distance.

Timing Results

Program Name	just_run_parts_red_and_psi.m numparts = 10000	JulianDist2.m Numparts = 1e7	low_mem_dist.m Numparts = 5e5	low_mem_test_sort.m numparts = 5e6
Time on personal computer CPU	419s	6.9798s	37.5337s	538s
Time on personal computer GPU	202s	0.91795s	11.6380s	141s
Time on P100 server CPU	1872s	58.6054s	Effectively forever	Effectively forever
Time on P100 server GPU	1666s	3.6085s	11.6552s	210s
Time on A100 server CPU	1040s	30.586s	177s	1600s
Time on A100 server GPU	326s	1.2503s	5.3788s	157s

Results

- As seen on the previous slide, for medium to large number of particles, the GPU device outperforms the CPU for all calculations.
- While the computations on the server generally perform a bit slower, the GPUs have a much larger memory, so they will be able to handle a higher number of particles compared to the desktop GPU.
- The comparison between the CPU and GPU on the personal machine is a good comparison, since it is a direct comparison between a consumer grade CPU and GPU.
- Converting existing MATLAB code to be computed in parallel on a GPU is very easy. In all the programs tested, the only change was to copy the initial particle position data to the GPU.
- For additional information, the MATLAB Help Centre website should be consulted.
  - https://www.mathworks.com/help/matlab/help-and-support.html