Introduction to GPGPU Computing with OpenACC

Learning Objectives

1. Understand what OpenACC is and its use cases in scientific applications
2. Learn how to compile and run programs that use OpenACC on Blue Waters.
3. Run an example program (diffusion model), comparing the performance difference between running serial vs. with OpenACC.
4. Learn about following OpenACC implementation of compiler directives:
    a. **`#pragma acc parallel`**
    b. **`#pragma acc parallel loop`**
    c. **`#pragma acc kernel`**
    d. **`#pragma acc data, present, copy, copyin, copyout, create`**
    e. **`#pragma acc update`**

Getting started

Login:

```
    $ ssh <username>@bw.ncsa.illinois.edu<ENTER>
```

Interactive node request:

```
    $ qsub -I -l nodes=1:ppn=16:xk,walltime=03:00:00<ENTER>
```

Download code:

```
$ wget http://shodor.org/~mludin/BW_Capstone/openACC_intro.tar <ENTER>
```

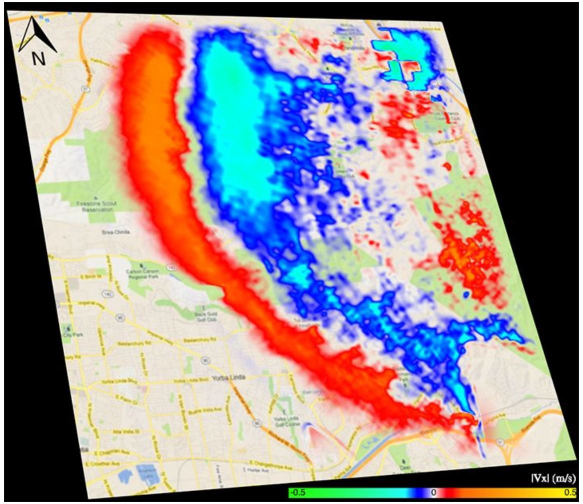
Extract the tar file:

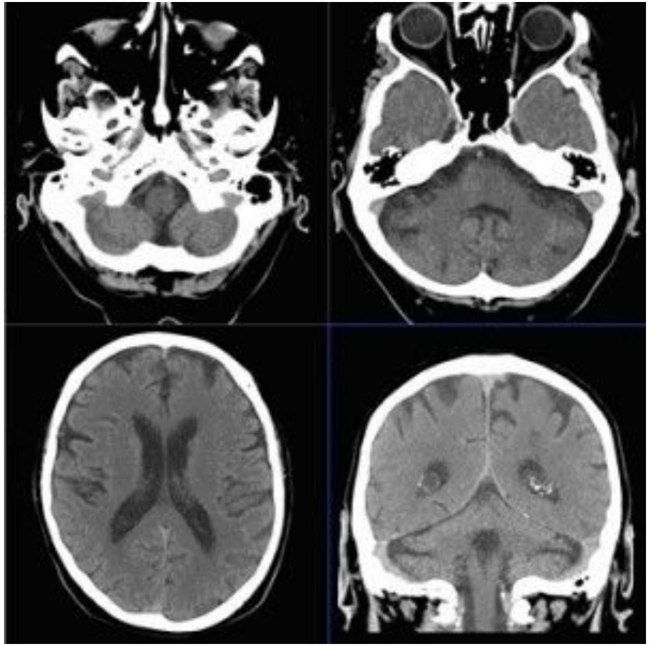
```
    $ tar -xvvf openACC_intro.tar<ENTER>
```

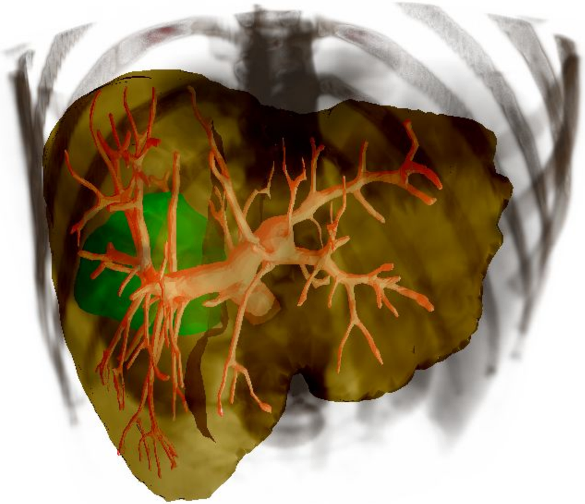
Change folders:

```
    $ cd openACC_intro/<ENTER>
    $ ls <ENTER>
```

The GPU Big Bang







CPU vs. GPU

- A GPU is tailored for highly parallel operation while a CPU executes programs serially

- For this reason, GPUs have many parallel execution units and higher transistor counts, while CPUs have few execution units and higher clock speeds

- GPUs have significantly faster and more advanced memory interfaces as they need to shift around a lot more data than CPUs
  - High bandwidth: NVIDIA Volta (2016) with 1TB/s
  - Nvidia Titan: 288GB/s

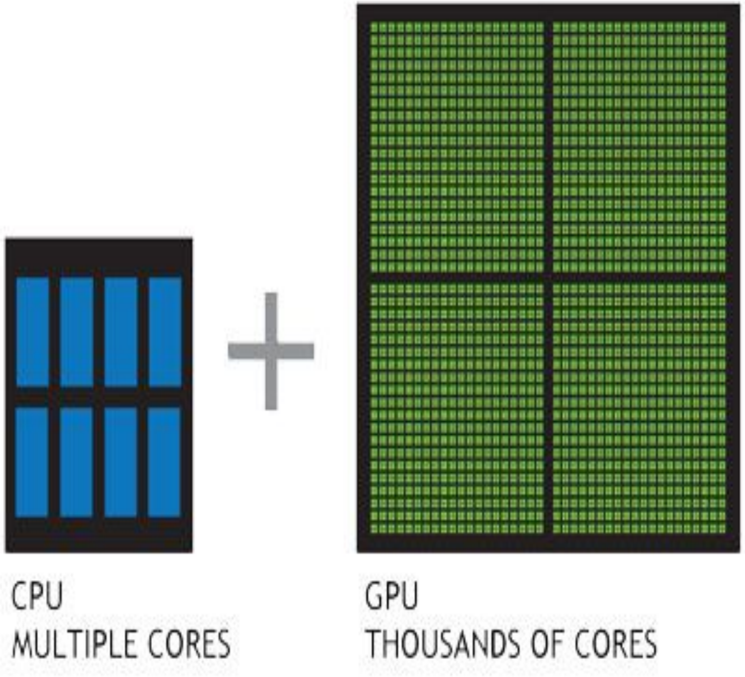
Modern CPUs Vs GPUs

So why not replace CPUs with GPGPUS?

- Single GPU cores = DUMI
- Group of GPU cores = FAST
- Do not have features for modern Operating Systems

Less logic

- No interrupts
- No Virtual Memory that OS Needs



What is OpenACC

```
#pragma acc data copyin(x,y) copyout(z)
while ( dt > MAX_TEMP_ERROR && iteration <= max_iterations ) // create data region before the parallel loops
{ ...
    #pragma acc parallel
    { // initiate parallel execution

        #pragma acc loop gang vector   //optimize loop mapping
        for (i =0; i < num_itterations; i++){
            z[i] = x[i] + y[i];
        } // end for loop

    } //end parallel execution region

} //write back data to host, and end data region and while loop
```

Parallelizing with OpenACC

```
#pragma acc data copyin(x,y) copyout(z)
while ( dt > MAX_TEMP_ERROR && iteration <= max_iterations ) // create data region before the parallel loops
{ ...
    #pragma acc parallel
    { // initiate parallel execution

        #pragma acc loop gang vector  //optimize loop mapping
        for (i =0; i < num_itterations; i++){
            z[i] = x[i] + y[i];
        } // end for loop

    } //end parallel execution region

} //write back data to host, and end data region and while loop
```

Resources:

- http://www.openacc.org/sites/default/files/OpenACC_API_QuickRefGuide.pdf
- https://bluewaters.ncsa.illinois.edu/openacc
- http://www.openacc.org/sites/default/files/OpenACC%202%200.pdf
- http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf
- http://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/
- http://arkanis.de/weblog/2011-04-02-finished-my-practical-term/gpgpu-origins-and-gpu-hardware-architecture.pdf
- http://cinwell.wordpress.com/2013/09/06/overview-of-gpu-architecture-fermi-based/
- http://electronicdesign.com/digital-ics/gpu-architecture-improves-embedded-application-support
- http://www.eecs.berkeley.edu/~sangjin/2013/02/12/CPU-GPU-comparison.html
- http://www.nvidia.com/content/cuda/spotlights/michael-bussmann-hzdr.html
- http://www.technologytell.com/gaming/59208/video-game-gpu-used-to-improve-ct-scans/