GPU Computing on FASRC Cluster

https://docs.rc.fas.harvard.edu/kb/gpgpu-computing-on-the-cluster

Cannon Cluster

Compute:

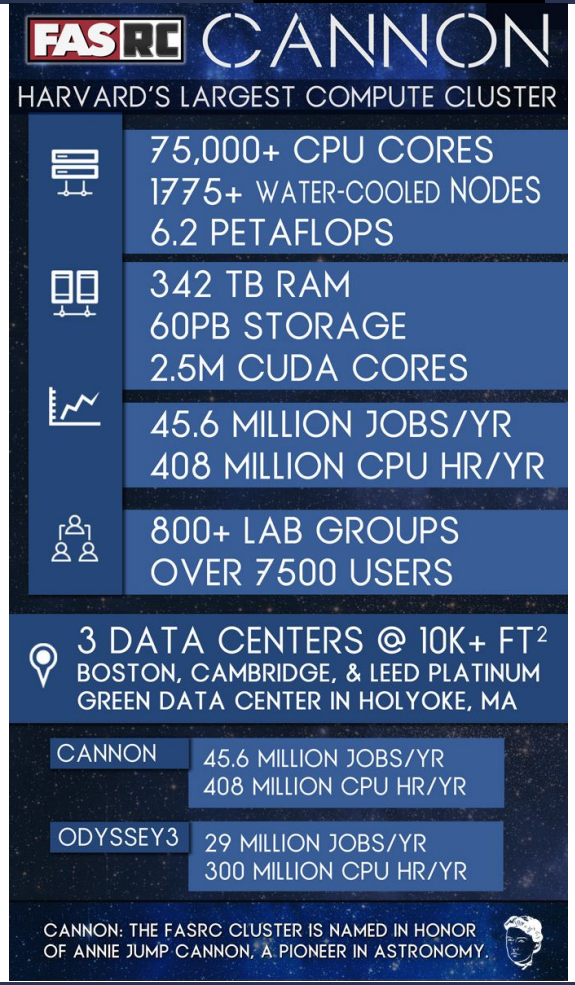
- 100,000 compute cores
- Cores/node: 8 to 64
- Memory/node: 12GB to 512GB (4GB/core)
- 2,500,000 NVIDIA GPU cores

Software:

- Operating System CentOS 7
- Slurm job manager
- 1,000+ scientific tools and programs
  - https://portal.rc.fas.harvard.edu/apps/modules

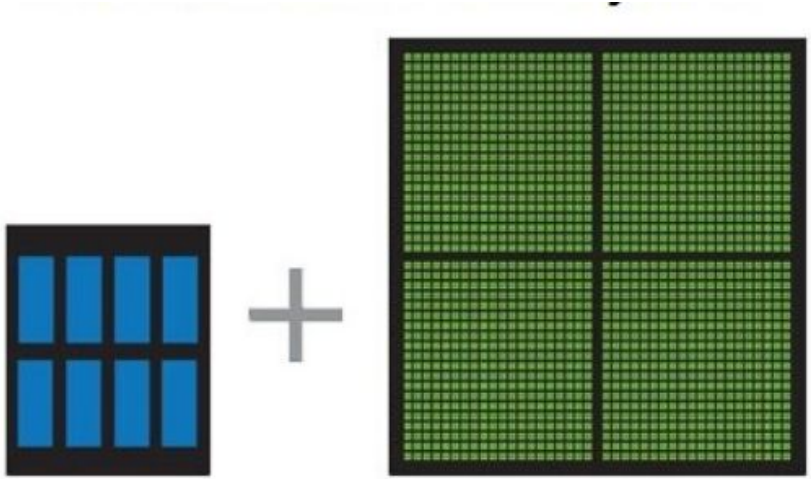
Interconnect:

- 2 underlying networks connecting 3 data centers
- TCP/IP network
- Low-latency 200 GB/s HDR InfiniBand (IB) and 56 GB/s FDR IB network:
  - inter-node parallel computing
  - fast access to Lustre mounted storage



What is GPGPU?

- General-Purpose Graphics Processing Unit (**GPGPU**) is a graphics processing unit (GPU) that is programmed for purposes beyond graphics processing, such as performing computations typically conducted by a Central Processing Unit (CPU).

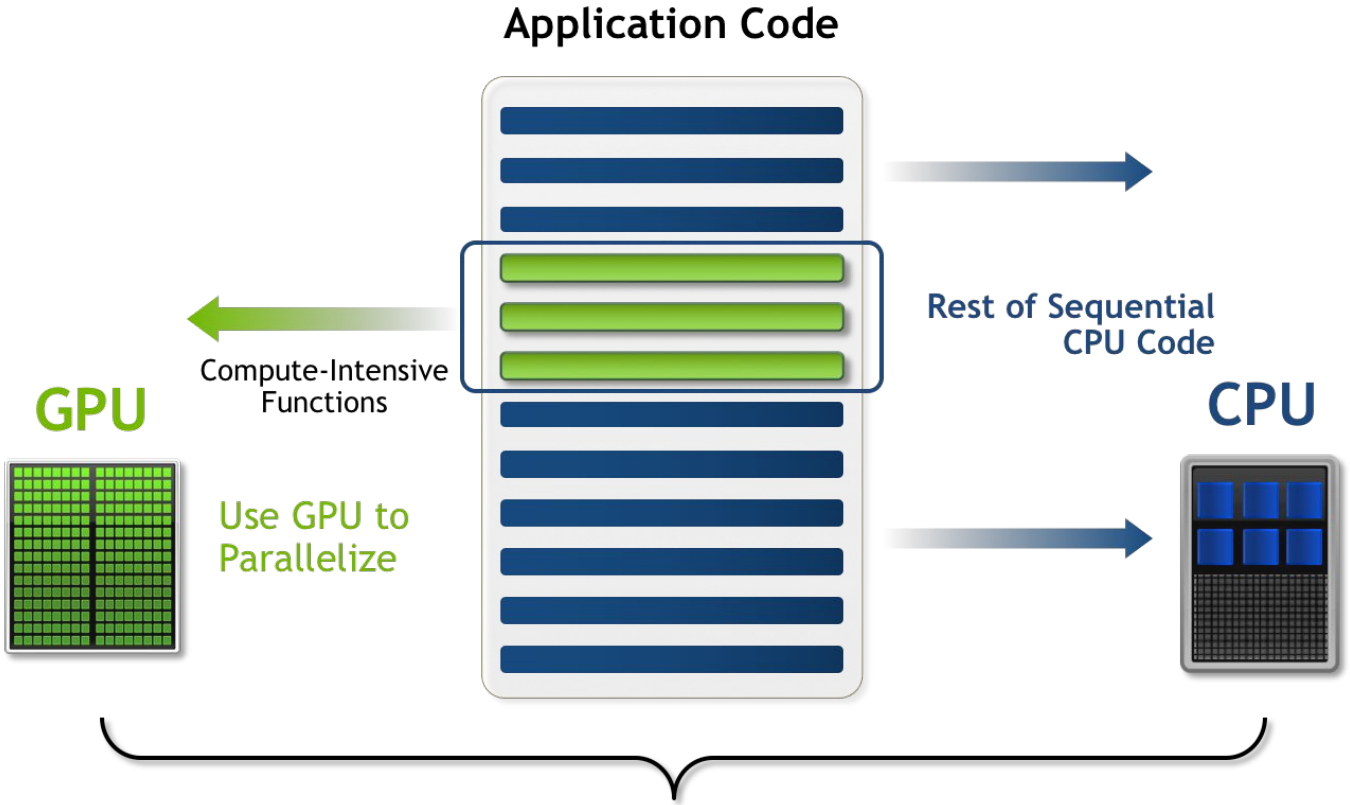


GPU vs CPU

CPU	GPU
Central Processing Unit	Graphics Processing Unit
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once

https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

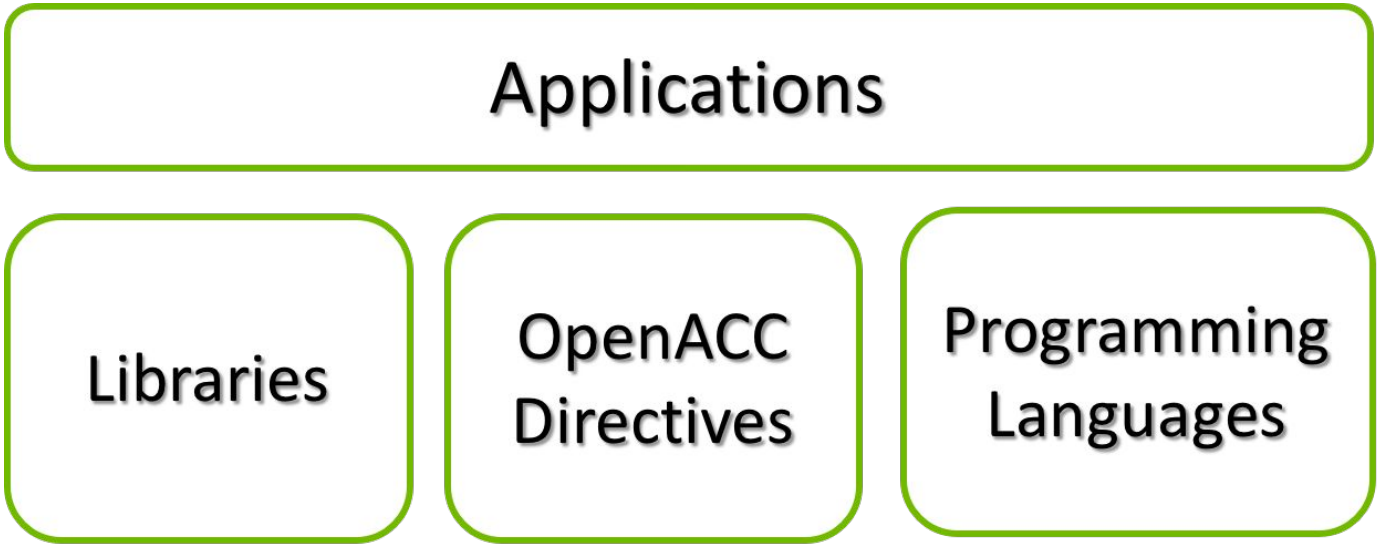
Heterogeneous Computing



Using GPGPUs

- GPU enabled applications requires a parallel computing platform and application programming interface (API) that allows software developers and software engineers to build algorithms to modify their application and map compute-intensive kernels to the GPU.
- GPGPU supports several types of memory in a memory hierarchy for designers to optimize their programs.
- GPGPU memory is used for transferring data between device and host -- shared memory is an efficient way for threads in the same block to share their runtime and data.

Ways to Accelerate your Applications



Drop-in Library: Cublas

CPU version

```
// define size

int N = 1 << 20;

// allocate cpu data

x = (float *)malloc(N * sizeof(float));

y = (float *)malloc(N * sizeof(float));

initData(x, y);

// Perform SAXPY on 1M elements: y[]=a*x[]+y[]

saxpy(N, 2.0, x, 1, y, 1);
```

GPU Acceleration

```
/ define size

int N = 1 << 20;

// allocate GPU memory

cudaMalloc(&d_x, N * sizeof(float));

cudaMalloc(&d_y, N * sizeof(float));

initData(x, y);

// Copy working data from CPU->GPU

cublasSetVector(N, sizeof(x[0]), x, 1, d_x, 1);
cublasSetVector(N, sizeof(y[0]), y, 1, d_y, 1);

// Perform SAXPY on 1M elements: y[]=a*x[]+y[]

cublasSaxpy(N, 2.0, d_x, 1, d_y, 1);

// Bring the result back to the CPU

cublasGetVector(N, sizeof(y[0]), d_y, 1, y, 1);
```

OpenACC

- OpenACC (for Open Accelerators) is a programming standard for parallel computing on accelerators (mostly on NIVDIA GPU).
- It is designed to simplify GPU programming.
- The basic approach is to insert special comments (directives) into the code so as to offload computation onto GPUs and parallelize the code at the level of GPU (CUDA) cores.
- It is possible for programmers to create an efficient parallel OpenACC code with only minor modifications to a serial CPU code.

OpenACC

OpenACC COMPILER DIRECTIVES

Parallel C Code

```
void saxpy(int n,
           float a,
           float *x,
           float *y)
{
#pragma acc kernels
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

...
// Perform SAXPY on 1M elements
saxpy(1<<20, 2.0, x, y);
...
```

Parallel Fortran Code

```
subroutine saxpy(n, a, x, y)
  real :: x(:), y(:), a
  integer :: n, i
!$acc kernels
  do i=1,n
    y(i) = a*x(i)+y(i)
  enddo
!$acc end kernels
end subroutine saxpy


...
! Perform SAXPY on 1M elements
call saxpy(2**20, 2.0, x_d, y_d)
...
```

http://developer.nvidia.com/openacc or http://openacc.org

Slide from Jeff Larkin - Nvidia

For more information: https://www.bu.edu/tech/files/2017/04/OpenACC-2017Spring.pdf

Compute Unified Device Architecture (CUDA)

- CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements for the execution of compute kernels.
- Designed to work with programming languages such as C, C++, and Fortran
- CUDA is an accessible platform, requiring no advanced skills in graphics programming, and available to software developers through CUDA-accelerated libraries and compiler directives.
- CUDA-capable devices are typically connected with a host CPU and the host CPUs are used for data transmission and kernel invocation for CUDA devices.
- The CUDA Toolkit includes GPU-accelerated libraries, a compiler, programming guides, API references, and the CUDA runtime.

https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html

Using Programming Languages

```
void saxpy(int n, float a,
      float *x, float *y)
{
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

int N = 1<<20;



// Perform SAXPY on 1M elements
saxpy(N, 2.0, x, y);
```

CUDA C

```
__global__
void saxpy(int n, float a,
     float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

int N = 1<<20;
cudaMemcpy(d_x, x, N, cudaMemcpyHostToDevice);
cudaMemcpy(d_y, y, N, cudaMemcpyHostToDevice);

// Perform SAXPY on 1M elements
saxpy<<<4096,256>>>(N, 2.0, d_x, d_y);

cudaMemcpy(y, d_y, N, cudaMemcpyDeviceToHost);
```

Slide from Jeff Larkin - Nvidia
https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html

Using GPU's on FASRC

- NVIDIA Tesla general purpose graphics processing units (GPGPU).
- 15 nodes with 4 V100 per node is available for general use from the `gpu` partition
- Your lab may have access to other partitions with GPU's or bought few nodes
  - "grep -ri "idreos_parkes" /etc/slurm/slurm.conf"

```
sinfo -p idreos_parkes
PARTITION     AVAIL  TIMELIMIT  NODES  STATE
NODELIST
idreos_parkes    up 7-00:00:00      1    mix holygpu2c1125
```

- Several other nodes with 4 V100 are available in `gpu_requeue`. These nodes are owned by various research groups available and may be available when idle.
- FAS members have access to the `fas_gpu` partition which has 34 nodes with 2xK80s.
- SEAS members has access to few other partitions so visit https://docs.rc.fas.harvard.edu/kb/seas-compute-resources/

Examples and Questions?

https://github.com/fasrc/User_Codes

VDI - Open OnDemand

For applications that need a GUI: https://vdi.rc.fas.harvard.edu

Supports:

- Remote Desktop
- Jupyter Notebooks
- Rstudio
- Matlab

Notes:

- Need to be on the RC VPN to use
- Sessions are submitted as jobs on the cluster and thus use fairshare but also can run on any partition

Request Help - Resources

- https://docs.rc.fas.harvard.edu/kb/support/
  - Documentation
    - https://docs.rc.fas.harvard.edu/
  - Portal
    - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
  - Email
    - rchelp@rc.fas.harvard.edu
  - Office Hours
    - Wednesday noon-3pm https://harvard.zoom.us/j/255102481
  - Consulting Calendar
    - https://www.rc.fas.harvard.edu/consulting-calendar/
  - Training
    - https://www.rc.fas.harvard.edu/upcoming-training/