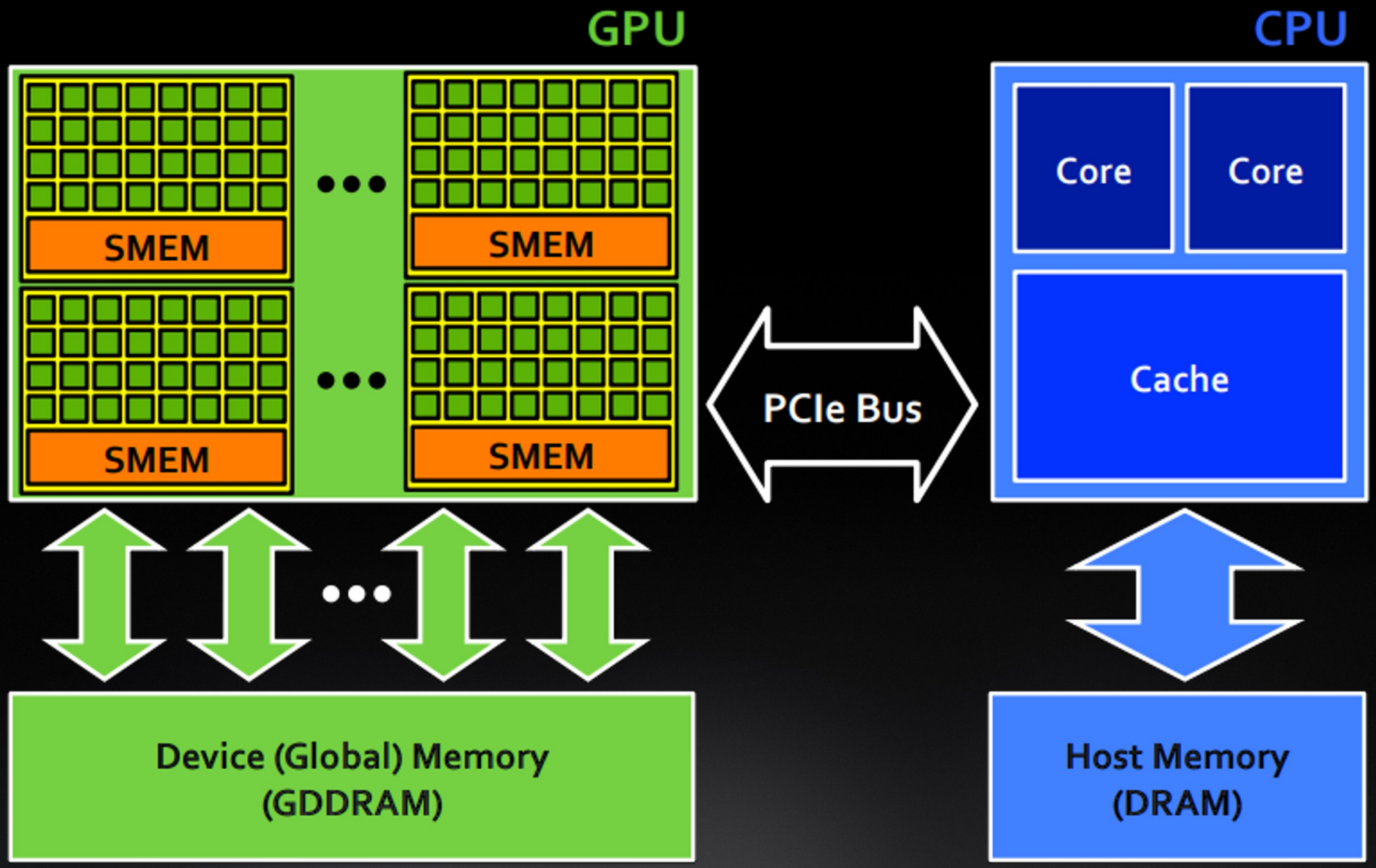
NWAYS BOOTCAMP

- openhackathons-org/nways_accelerated_programming (github.com)
- C programming language
    - std::par / OpenACC / OpenMP / CUDA
- Fortran programming language
    - do-concurrent / OpenACC / OpenMP / CUDA
- Python programming language
    - CuPy / Numba

GPU ARCHITECTURE



USER ANNOTATIONS APIS FOR CPU & GPU
NVTX, OPENGL, VULKAN, AND DIRECT3D PERFORMANCE MARKERS

EXAMPLE: VISUAL MOLECULAR DYNAMICS (VMD) ALGORITHMS VISUALIZED WITH NVTX ON CPU

GPU PROGRAMMING IN 2023 AND BEYOND

Math Libraries | Standard Languages | Directives | CUDA

```
std::transform(par, x, x+n, y, y,
    [=](float x, float y) {
        return y + a*x;
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

```
#pragma acc data copy(x,y)
{

...

std::transform(par, x, x+n, y, y,
    [=](float x, float y) {
        return y + a*x;
});

...

}
```

```
__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];
}

int main(void) {
  cudaMallocManaged(&x, ...);
  cudaMallocManaged(&y, ...);
  ...
  saxpy<<<(N+255)/256,256>>>(...,x, y)
  cudaDeviceSynchronize();
  ...
}
```

GPU Accelerated
C++ and Fortran

Incremental Performance
Optimization with Directives

Maximize GPU Performance with
CUDA C++/Fortran

GPU Accelerated Math Libraries

OVERVIEW OF CUPY

- **CuPy** supports a subset of numpy.ndarray interface which include:

✓ Basic & advance indexing, and Broadcasting

✓ Data types (int32, float32, uint64, complex64,...)

✓ Array manipulation routine (reshape)

✓ Linear Algebra functions (dot, matmul, etc)

✓ Reduction along axis (max, sum, argmax, etc)

For more details on broadcasting visit

(https://numpy.org/doc/stable/user/basics.broadcasting.html)

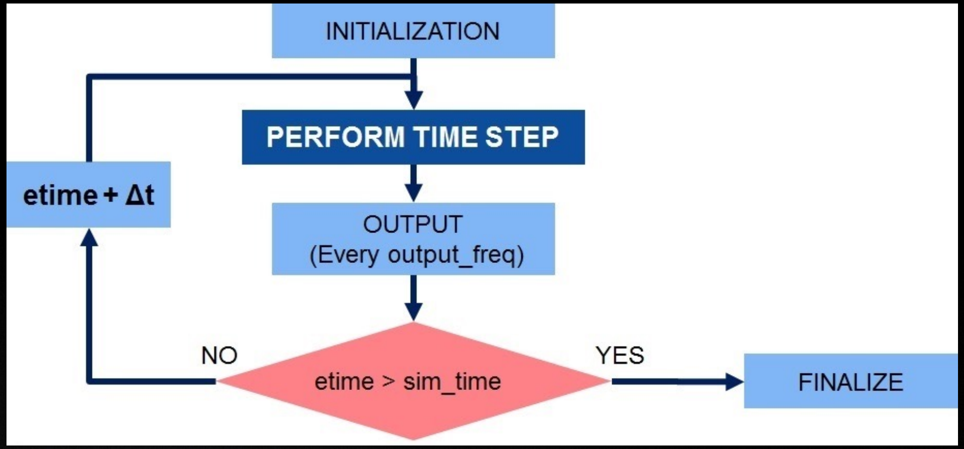
```
>>> import numpy as np
>>> X = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
#Basic indexing and slicing
>>> X[5:]
array([5, 6, 7, 8, 9])
>>> X[1:7:2]
array([1, 3, 5])

#Advance indexing
>>> X = np.array([[1, 2],[3, 4],[5, 6]])
>>> X[[0, 1, 2], [0, 1, 0]]
array([1, 4, 5])

#reduction and Linear Algebra function
>>> max(X)
9.0
>>> B = np.array([1,2,3,4], dtype=np.float32)
>>> C = np.array([5,6,7,8], dtype=np.float32)
>>> np.matmul(B, C)
70.0
#data type and array manipulation routine
>>> A =1j*np.arange(9, dtype=np.complex64).reshape(3,3)
[[0.+0.j 0.+1.j 0.+2.j]
 [0.+3.j 0.+4.j 0.+5.j]
 [0.+6.j 0.+7.j 0.+8.j]]
```

CODE CHALLENGE

Fluid simulation

We will accelerate a Fluid Simulation in the context of atmosphere and weather simulation. The mini weather code mimics the basic dynamics seen in the atmospheric weather and climate.

```
while (etime < sim_time) {
    //If the time step leads to exceeding the simulation time, shorten it for the last step
    if (etime + dt > sim_time) { dt = sim_time - etime; }
    //Perform a single time step
    perform_timestep(state,state_tmp,flux,tend,dt);
    //Inform the user
    if (masterproc) { printf( "Elapsed Time: %lf / %lf\n", etime , sim_time ); }
    //Update the elapsed time and output counter
    etime = etime + dt;
    output_counter = output_counter + dt;
    //If it's time for output, reset the counter, and do output
    if (output_counter >= output_freq) {
      output_counter = output_counter - output_freq;
      output(state,etime);
    }
  }
```



Make use of OpenACC and CUDA C/fortran GPU programming to parallelize and improve the performance.

NVIDIA