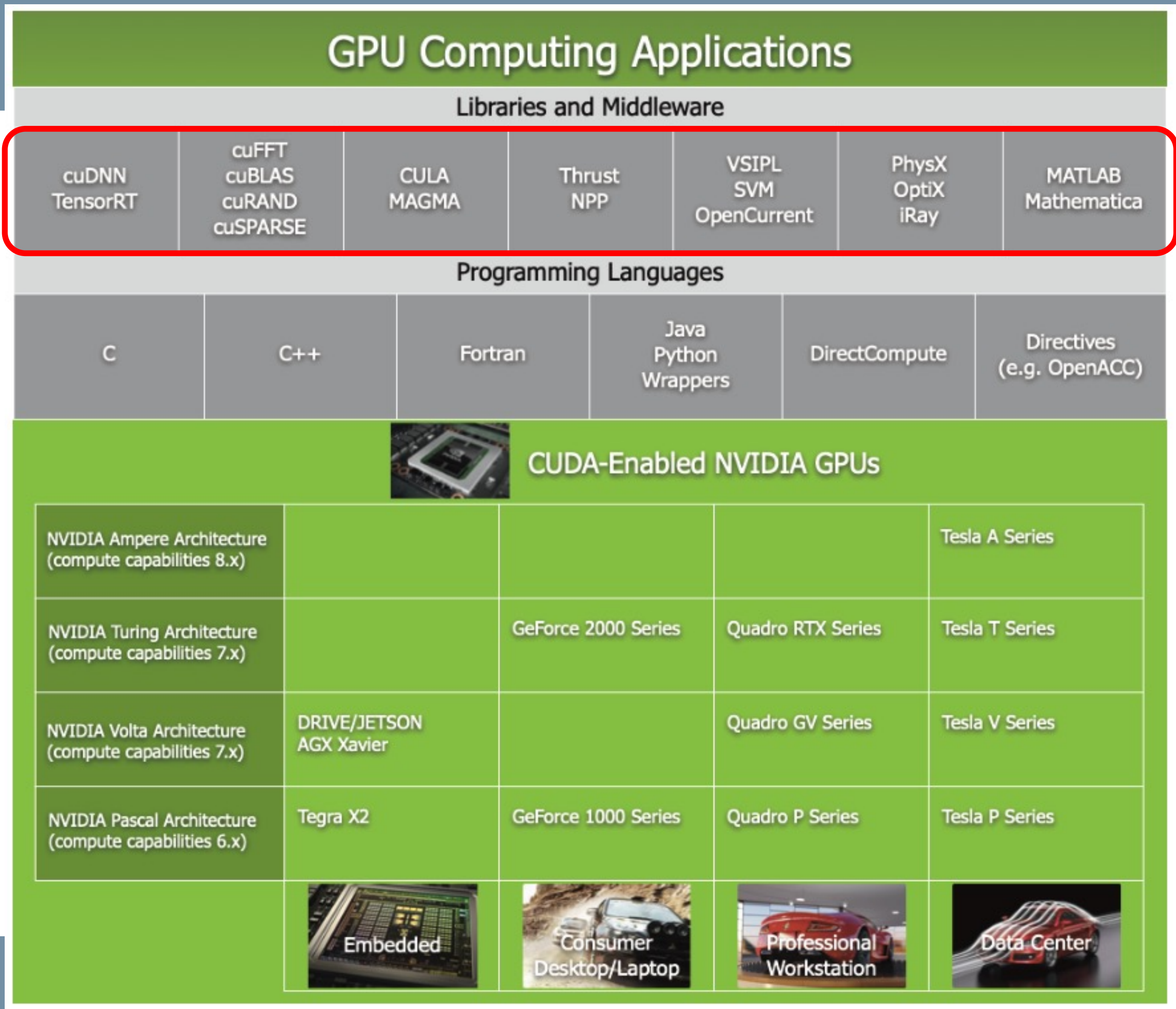
GPUs and Heterogeneous Systems
(programming models and architectures)

CUDA libraries

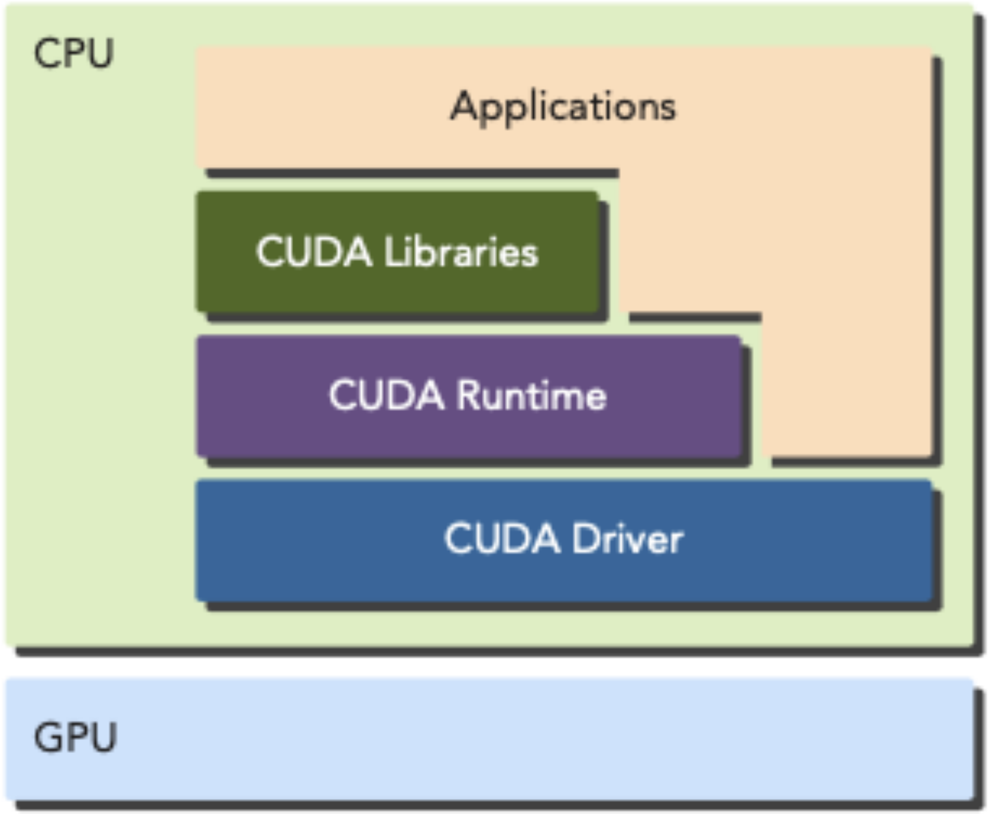
CUDA software platform



CUDA GPU-accelerated libraries

- CUDA offers **GPU-accelerated libraries** for executing "popular" functions/algorithms
  - Optimized to achieve **high performance**
  - Increase software **productivity**
    - High usability
    - Low maintainability effort

- https://developer.nvidia.com/gpu-accelerated-libraries



CUDA GPU-accelerated libraries

- **Math libraries**
  - cuBLAS, cuFFT, CUDA Math Library, cuRAND, cuSOLVER, cuSPARSE, cuTENSOR, AmgX
- **Parallel algorithm libraries**
  - Thrust
- **Image and video libraries**
  - nvJPEG, Performance Primitives, Video Codec SDK, NVIDIA Optical Flow SDK
- **Communication libraries**
  - NVSHMEM, NCCL
- **Deep learning libraries**
  - cuDNN, TensorRT, Riva, DeepStream SDK, DALI
- **Partner libraries**
  - OpenCV, Ffmpeg, ArrayFire, MAGMA, IMSL Fortran Numerical Library, Gunrock, CHOLMOD, Triton Ocean SDK, CUVIlib

Common library workflow

1. Create a library-specific handle maintaining contextual information
2. Allocate device memory for input/output of the library function
3. If necessary, convert inputs in the library format
4. Copy inputs in the pre-allocated device memory
5. Configure the library to execute
6. Run library function
7. Copy back outputs to the host
8. If necessary, convert outputs to the application specific format
9. Release CUDA resources
10. Continue with the rest of the application

An example of random number generation

```
#include <curand.h>
#include <curand_kernel.h>
#define N ...

__global__ void init(unsigned int seed, curandState_t* states);
__global__ void randoms(curandState_t* states, unsigned int* numbers);

int main() {
  curandState_t* states;

  cudaMalloc(&states, N*sizeof(curandState_t));

  /*...*/
```

An example of random number generation

```
#include <curand.h>
#include <curand_kernel.h>
#define N ...

__global__ void init(unsigned int seed, curandSta
__global__ void randoms(curandState_t* states, u

int main() {
  curandState_t* states;

  cudaMalloc(&states, N*sizeof(curandState_t));

  /*...*/
```

Library headers

Number of random values to be generated at the same time. N is divisible by 32 for the sake of simplicity

Declare and allocate a random number generator handle for each thread

An example of random number generation

```
/*...*/
unsigned int h_nums[N];
unsigned int* d_nums;
cudaMalloc(&d_nums, N * sizeof(unsigned int));
init<<<N/32, 32>>>(time(NULL), states);
/*...*/
```

```
__global__ void init(unsigned int seed, curandState_t* states) {
  curand_init(seed,
              threadIdx.x,
              0,
              &states[threadIdx.x]);
}
```

An example of random number generation

```
/*...*/
unsigned int h_nums[N];
unsigned int* d_nums;
cudaMalloc(&d_nums, N * sizeof(unsigned int));
init<<<N/32, 32>>>(time(NULL), states);
/*...*/
```

Allocate host and device memory to store generated random values

```
__global__ void init(unsigned int seed, curandState_t* states) {
  curand_init(seed,
              threadIdx.x,
              0,
              &states[threadIdx.x]);
}
```

An example of random number generation

```
/*...*/
unsigned int h_nums[N];
unsigned int* d_nums;
cudaMalloc(&d_nums, N * sizeof(unsigned int));
init<<<N/32, 32>>>(time(NULL), states);
/*...*/
```

Initialization of random numbers generators is performed in a device kernel: each thread initialize a handle

```
__global__ void init(unsigned int seed, curandState_t* states) {
  curand_init(seed,
              threadIdx.x,
              0,
              &states[threadIdx.x]);
}
```

An example of random number generation

```
/*...*/
unsigned int h_nums[N];
unsigned int* d_nums;
cudaMalloc(&d_nums, N * sizeof(unsigned int
init<<<N/32, 32>>>(time(NULL), states);
/*...*/
```

- The seed can be the same for all threads
- The sequence number, initialized with the thread id, is a sort of additional seed to have a different random number generator in each thread)
- The initialized handle

```
__global__ void init(unsigned int seed, curandState_t* states) {
  curand_init(seed,
              threadIdx.x,
              0,
              &states[threadIdx.x]);
}
```

An example of random number generation

```
/*...*/
randoms<<<N/32, 32>>>(states, d_nums);
/*...*/
```

```
__global__ void randoms(curandState_t* states, unsigned int* numbers) {
  numbers[threadIdx.x] = curand(&states[threadIdx.x]) % 100;
}
```

An example of random number generation

```
/*...*/
randoms<<<N/32, 32>>>(states, d_nums);
/*...*/
```

- Random number generation is here performed in a user kernel
- Each random number generator can generate a random value in parallel with the other ones
- Various possible distributions can be used: `curand`, `curand_uniform`, …
- The API offers random generator functions to be called directly in the host code

```
__global__ void randoms(curandState_t* states, unsigned int* numbers) {
  numbers[threadIdx.x] = curand(&states[threadIdx.x]) % 100;
}
```

An example of random number generation

```
  /*...*/
  cudaMemcpy(h_nums, d_nums, N*sizeof(unsigned int), cudaMemcpyDeviceToHost);

  /* use numbers */

  cudaFree(states);
  cudaFree(d_nums);

  return 0;
}
```

An example of random number generation

```
  /*...*/
  cudaMemcpy(h_nums, d_nums, N*sizeof(unsigned int), cudaMemcpyDeviceToHost);

  /* use numbers */

  cudaFree(states);
  cudaFree(d_nums);

  return 0;
}
```

Generated values can be used both on the device and on the host

As usual, resources have to be released

References

- Slides mainly based on:
  - J. Chen, M. Grossman, T. McKercher, **Professional Cuda C Programming**, **Chapter 8**
  - https://ianfinlayson.net/class/cpsc425/samples/cuda/random3.cu