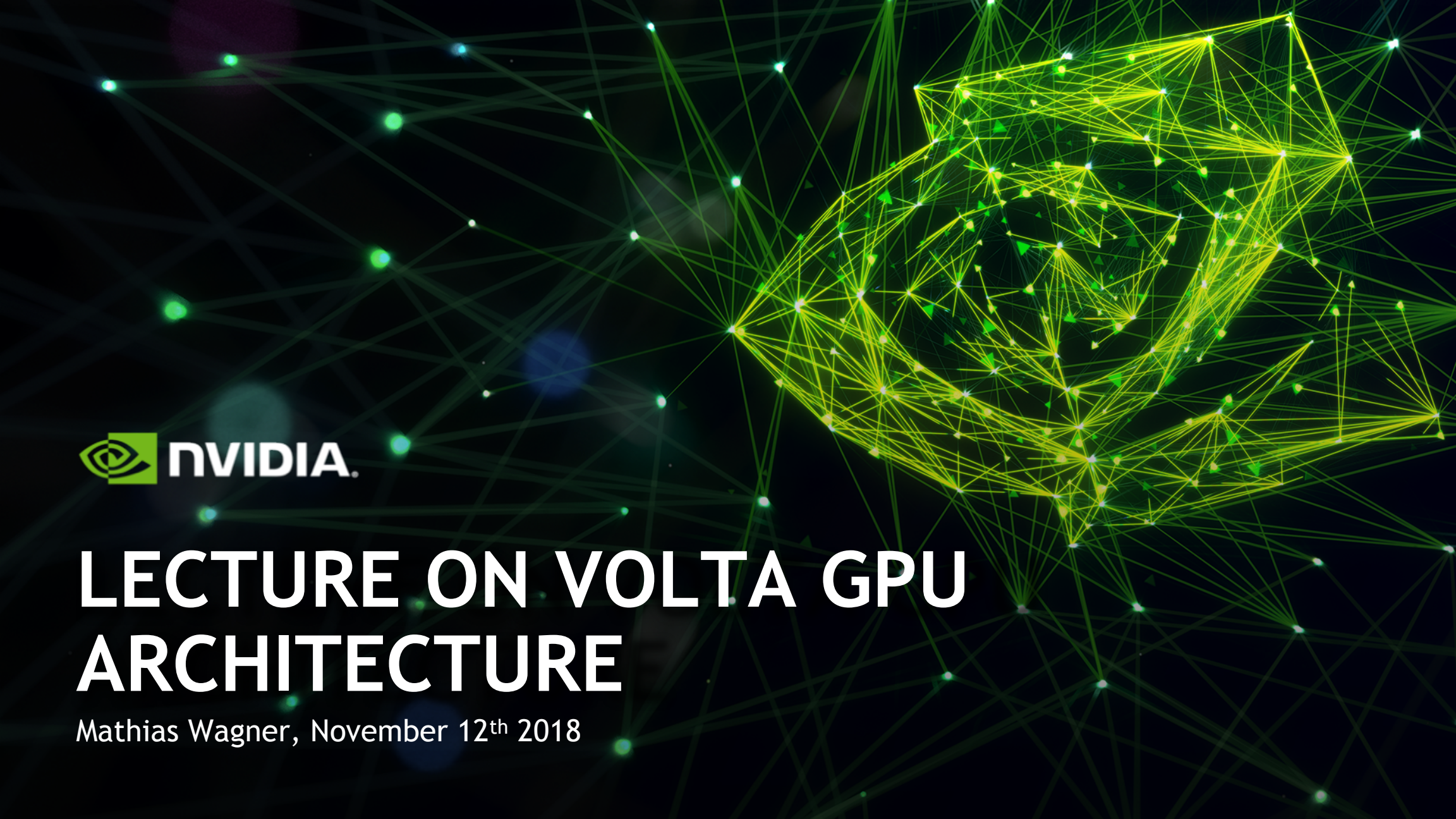
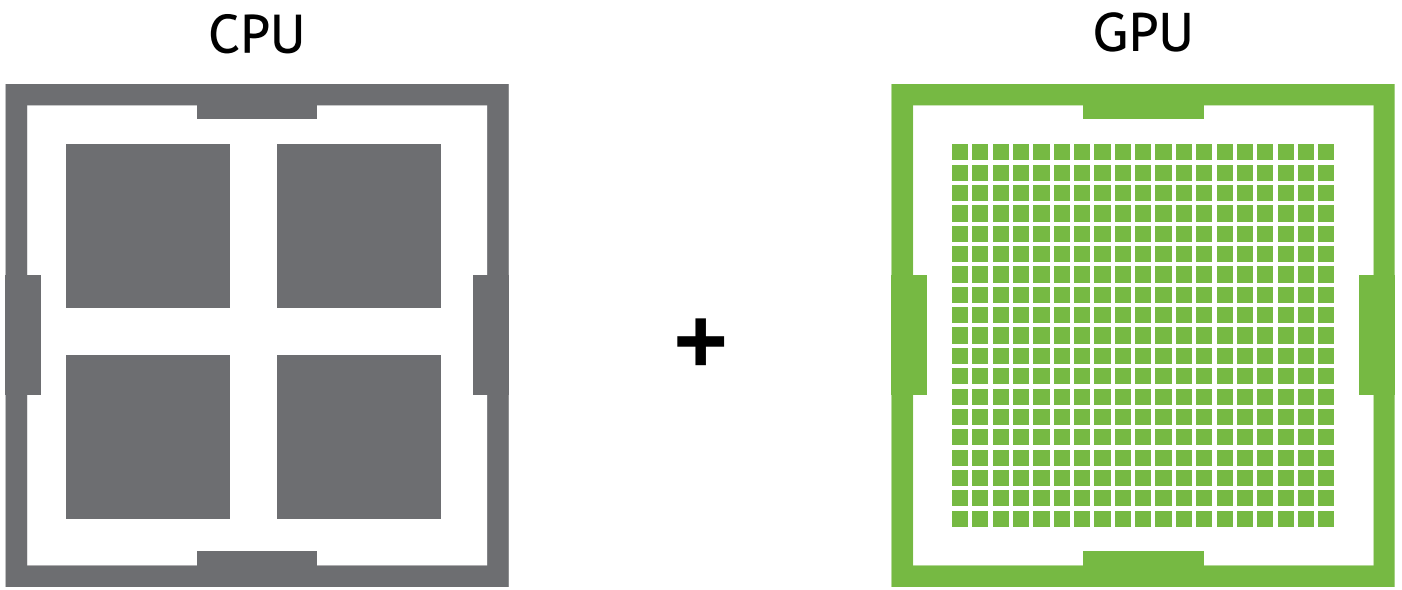


LECTURE ON VOLTA GPU ARCHITECTURE

Mathias Wagner, November 12th 2018

ADD GPUS

Accelerate science applications



ADD GPUS

Accelerate science applications

CPU

CPU Strengths

- Very large main memory
- Very fast clock speeds
- Latency optimized via large caches
- Small number of threads can run very quickly

CPU Weaknesses

- Relatively low memory bandwidth
- Cache misses very costly
- Low performance per watt

ADD GPUS

Accelerate science applications

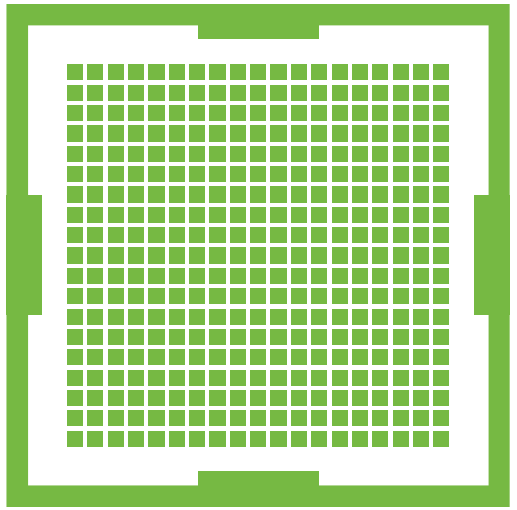
GPU Strengths

- High bandwidth main memory
- Latency tolerant via parallelism
- Significantly more compute resources
- High throughput
- High performance per watt

GPU Weaknesses

- Relatively low memory capacity
- Low per-thread performance

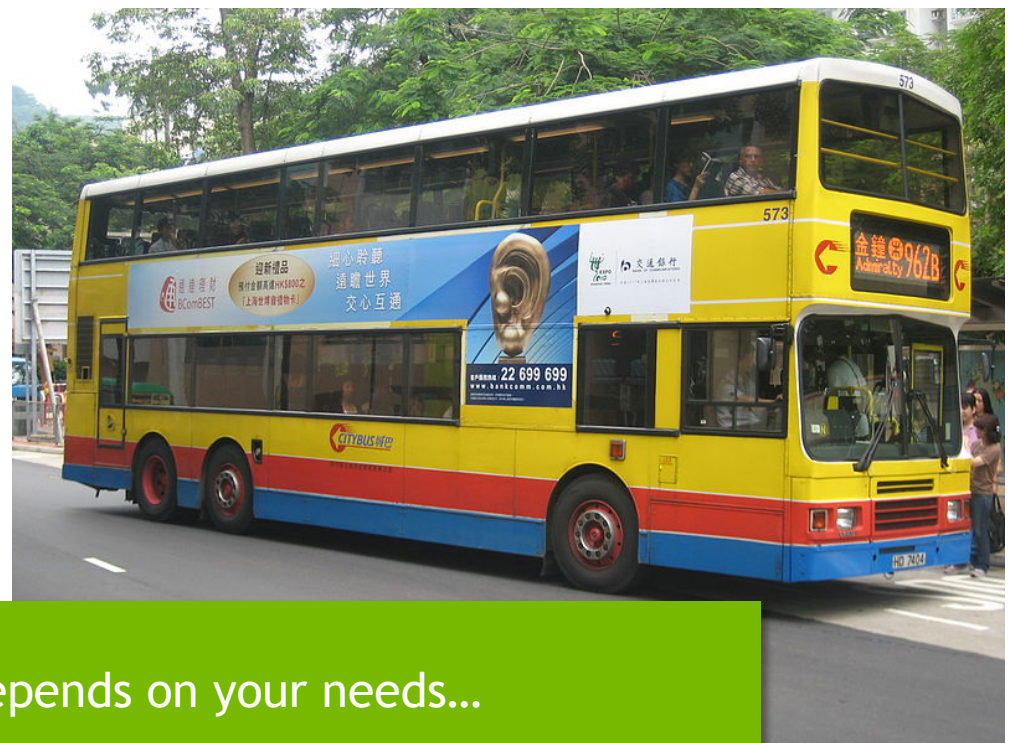
GPU



SPEED V. THROUGHPUT

Speed

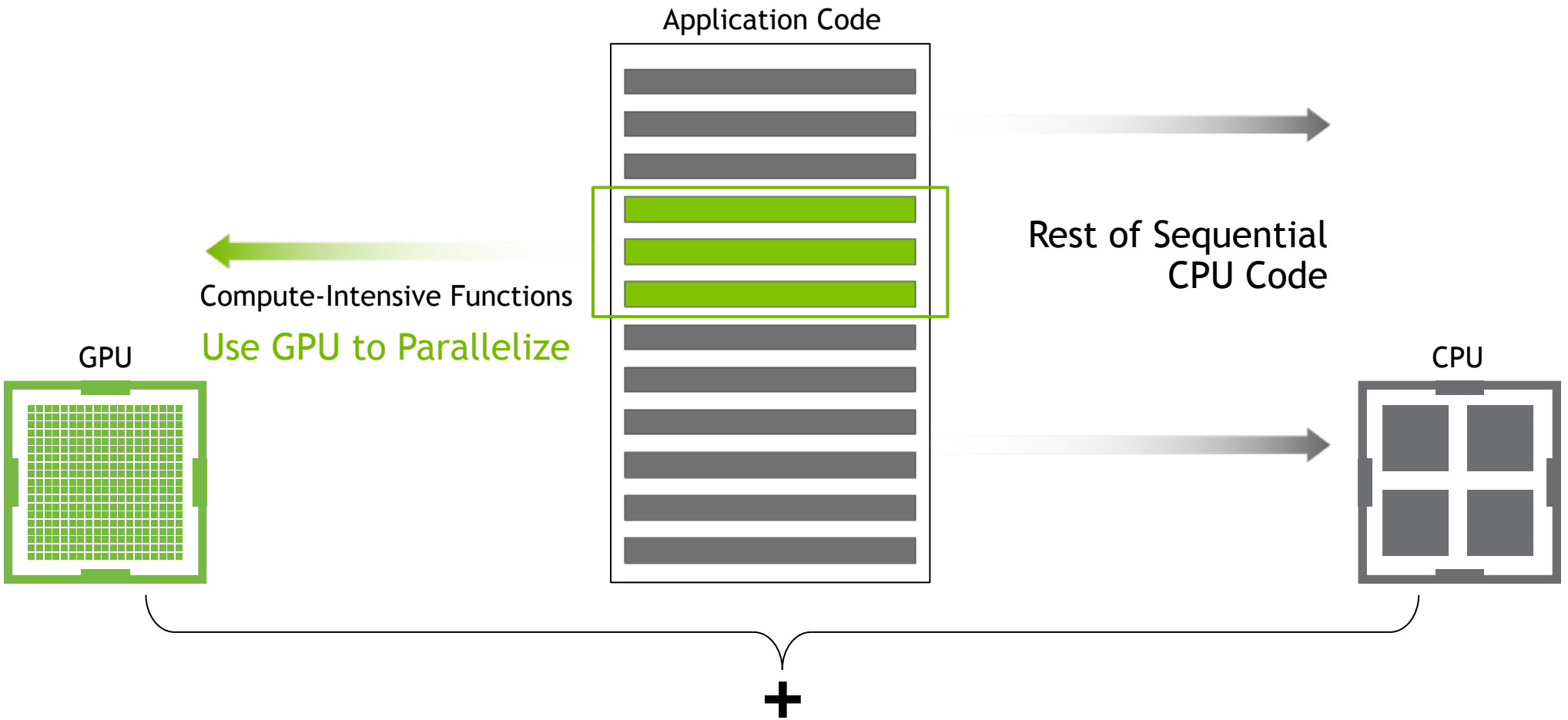
Throughput

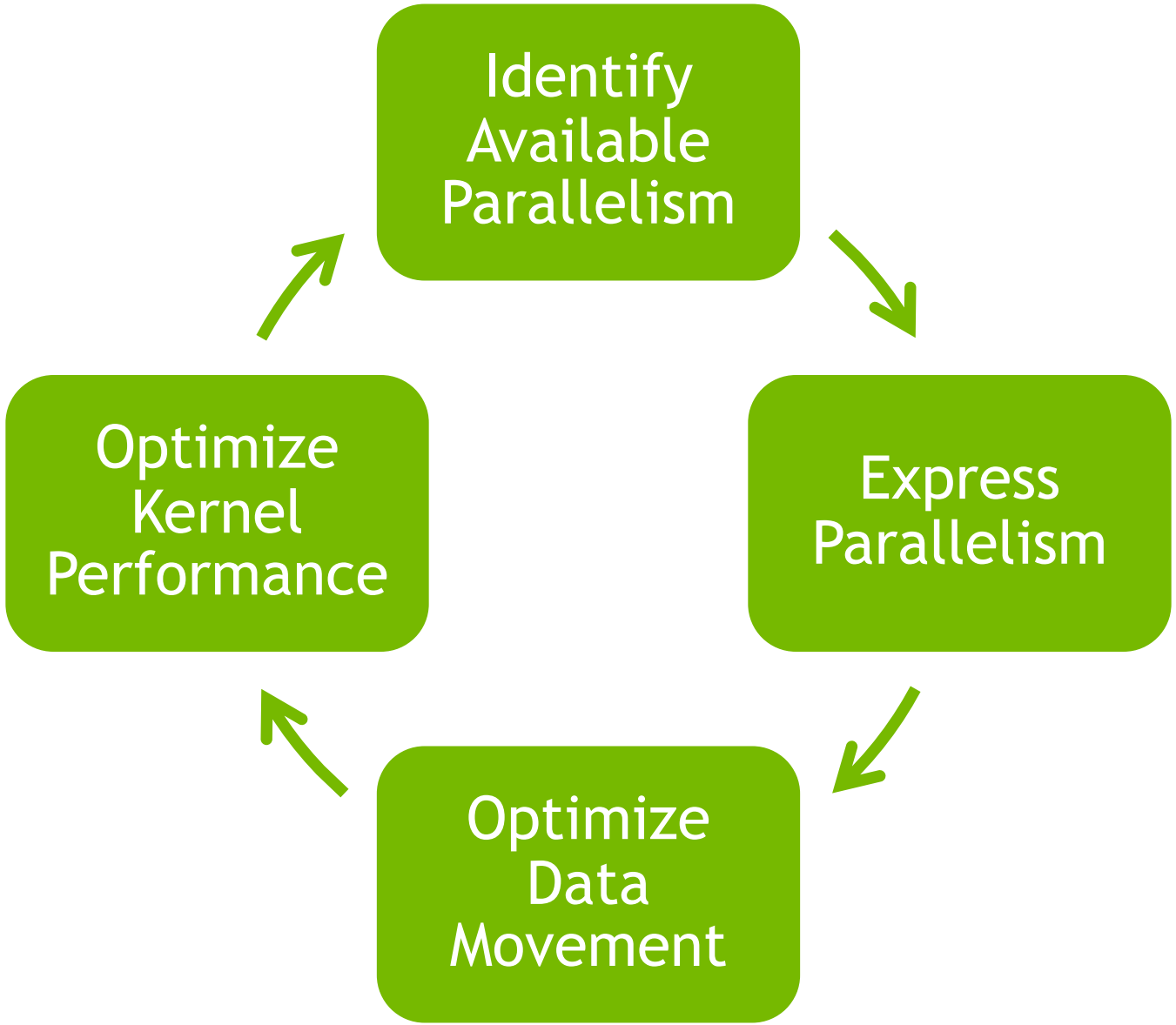


Which is better depends on your needs…

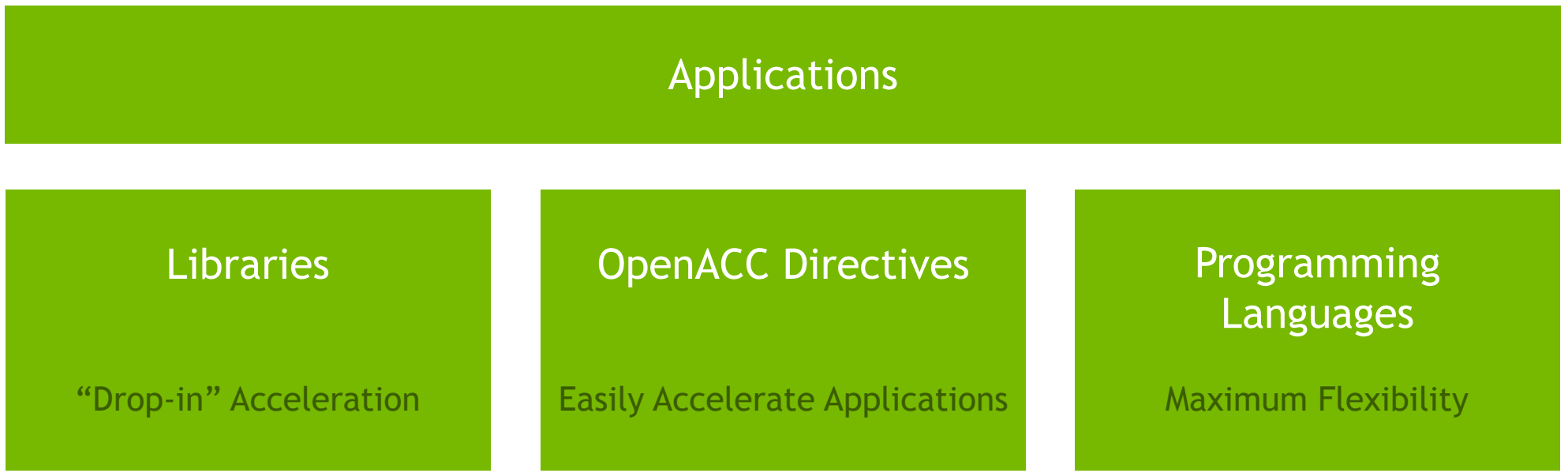
*Images from Wikimedia Commons via Creative Commons

SMALL CHANGES, BIG SPEED-UP




Identify Available Parallelism
Express Parallelism
Optimize Data Movement
Optimize Kernel Performance

3 WAYS TO ACCELERATE APPLICATIONS



OPENACC DIRECTIVES

Manage Data Movement

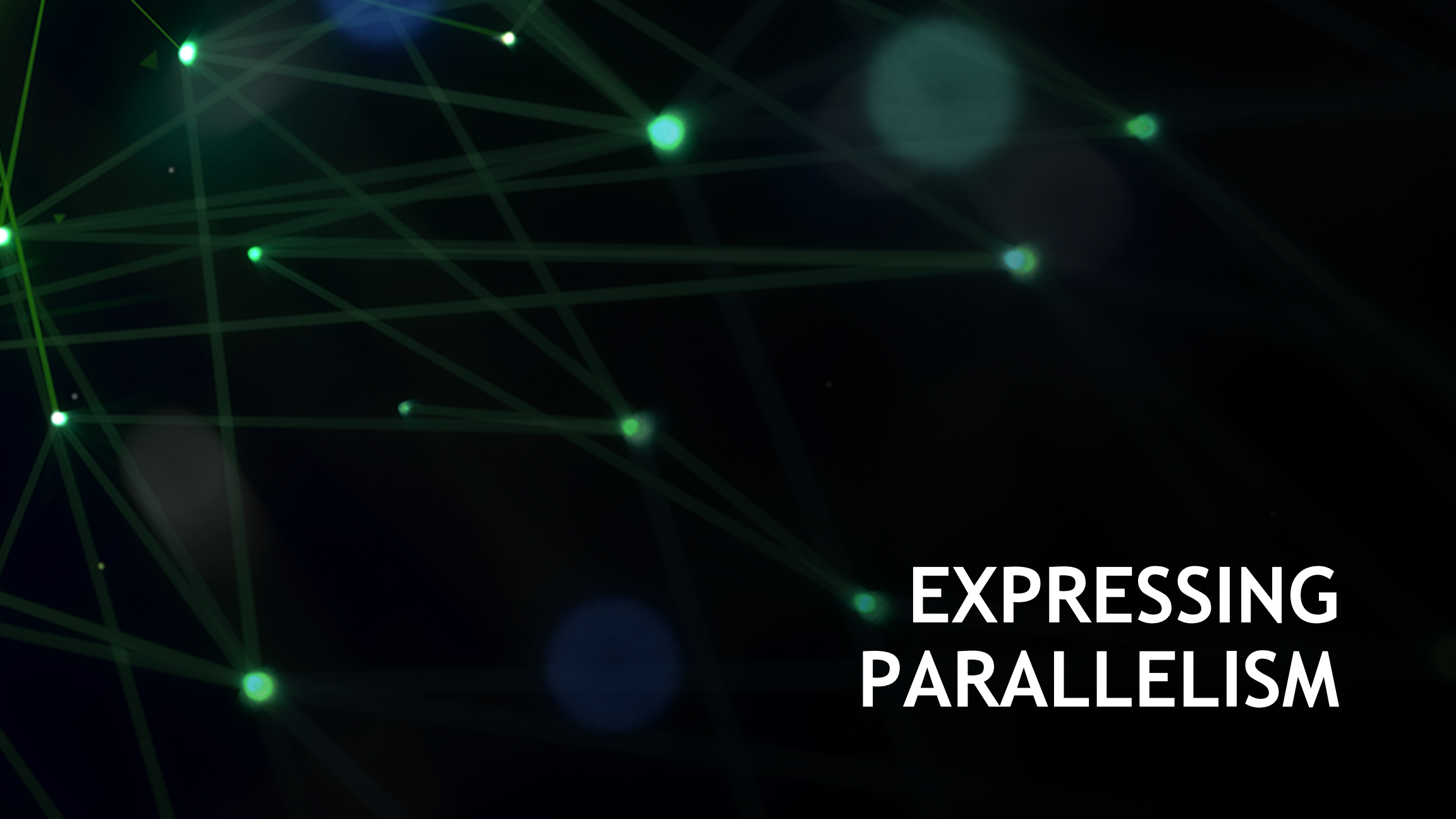
Initiate Parallel Execution

Optimize Loop Mappings

```
#pragma acc data copyin(x,y) copyout(z)
{
  #pragma acc parallel
  {
    #pragma acc loop gang vector
    for (i = 0; i < n; ++i) {
      z[i] = x[i] + y[i];
      ...
    }
  }
  ...
}
```

OpenACC
Directives for Accelerators

- Incremental
- Single source
- Interoperable
- Performance portable
- CPU, GPU, MIC

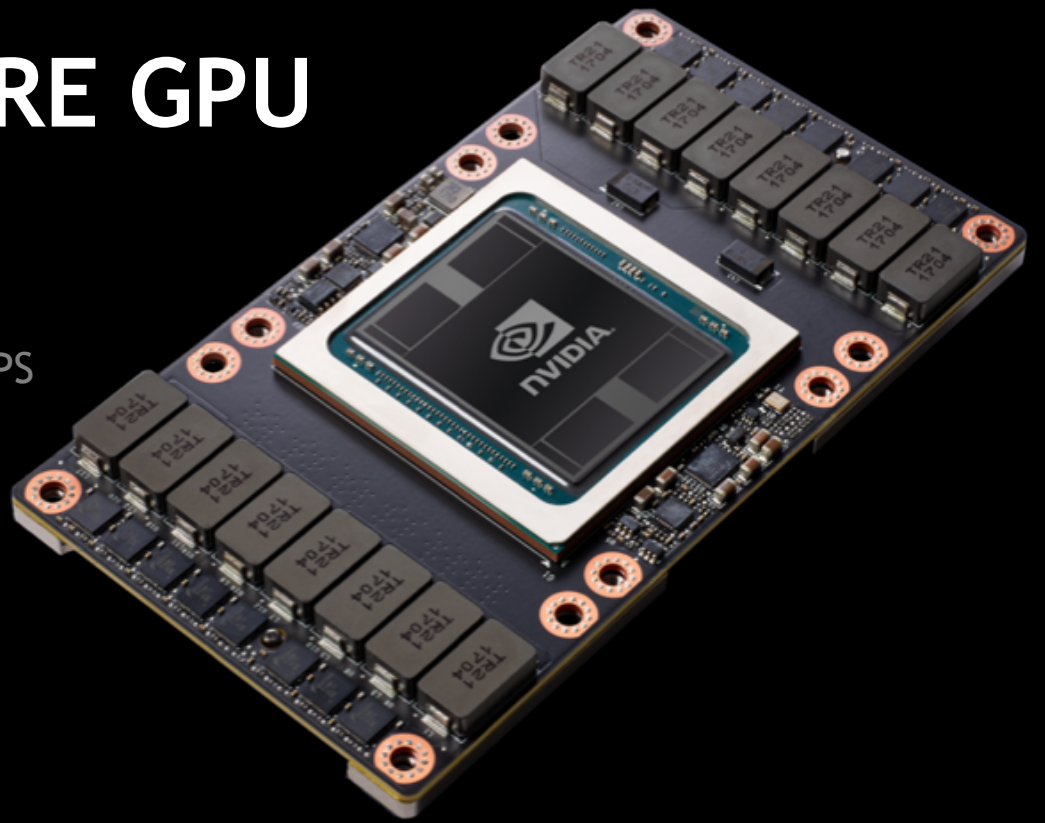


EXPRESSING PARALLELISM

TESLA V100 TENSOR CORE GPU

FUSING AI AND HIGH PERFORMANCE COMPUTING

5,120 CUDA cores
640 NEW Tensor cores
7.8 FP64 TFLOPS | 15.7 FP32 TFLOPS | 125 Tensor TFLOPS
20MB SM RF | 16MB Cache
16GB/ 32GB HBM2 @ 900GB/s | 300GB/s NVLink



TESLA V100

21B transistors
815 mm²

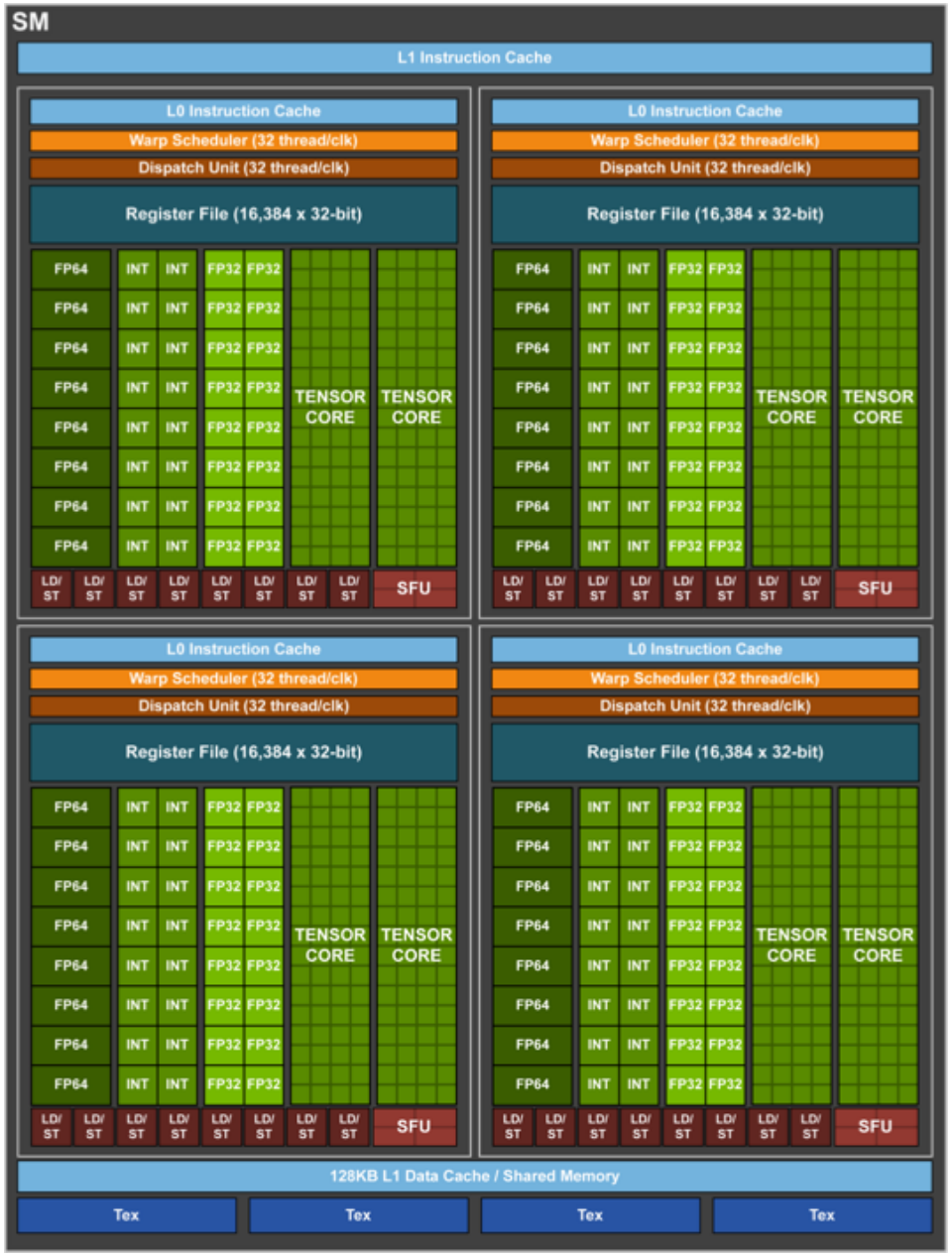
80 SM
5120 CUDA Cores
640 Tensor Cores

16/32 GB HBM2
900 GB/s HBM2
300 GB/s NVLink

*full GV100 chip contains 84 SMs

VOLTA GV100 SM

	GV100
FP32 units	64
FP64 units	32
INT32 units	64
Tensor Cores	8
Register File	256 KB
Unified L1/Shared memory	128 KB
Active Threads	2048



GPU PERFORMANCE COMPARISON

	P100	V100	Ratio
Training acceleration	10 TOPS	120 TOPS	12x
Inference acceleration	21 TFLOPS	120 TOPS	6x
FP64/FP32	5/10 TFLOPS	7.5/15 TFLOPS	1.5x
HBM2 Bandwidth	720 GB/s	900 GB/s	1.2x
NVLink Bandwidth	160 GB/s	300 GB/s	1.9x
L2 Cache	4 MB	6 MB	1.5x
L1 Caches	1.3 MB	10 MB	7.7x

OPENACC DIRECTIVES

Express Parallelism

```
real* restrict const A = (real*) malloc(nx*ny*sizeof(real));

real* restrict const Aref = (real*) malloc(nx*ny*sizeof(real));

...

#pragma acc parallel loop

for (int iy = iy_start; iy < iy_end; iy++) {

  for( int ix = ix_start; ix < ix_end; ix++ ) {

    A[iy*nx+ix] = Anew[iy*nx+ix];

  }

}
```


OPTIMIZING DATA
LOCALITY

NVLINK - GPU CLUSTER

Two fully connected quads,
connected at corners

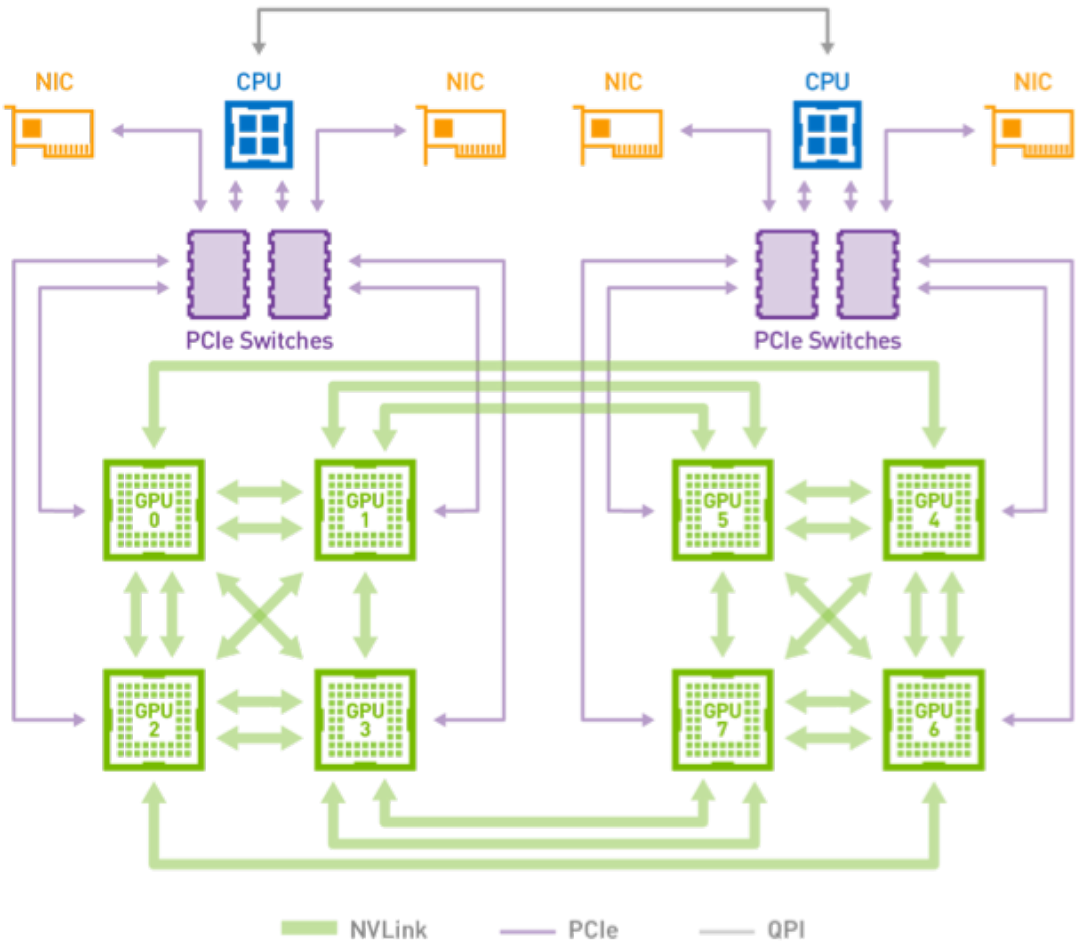
300 GB/s per GPU bidirectional to Peers

Load/store access to Peer Memory

Full atomics to Peer GPUs

High speed copy engines for bulk data copy

PCIe to/from CPU



NVLINK - NVSWITCH GPU CLUSTER

16 GPUs

Fully-connected

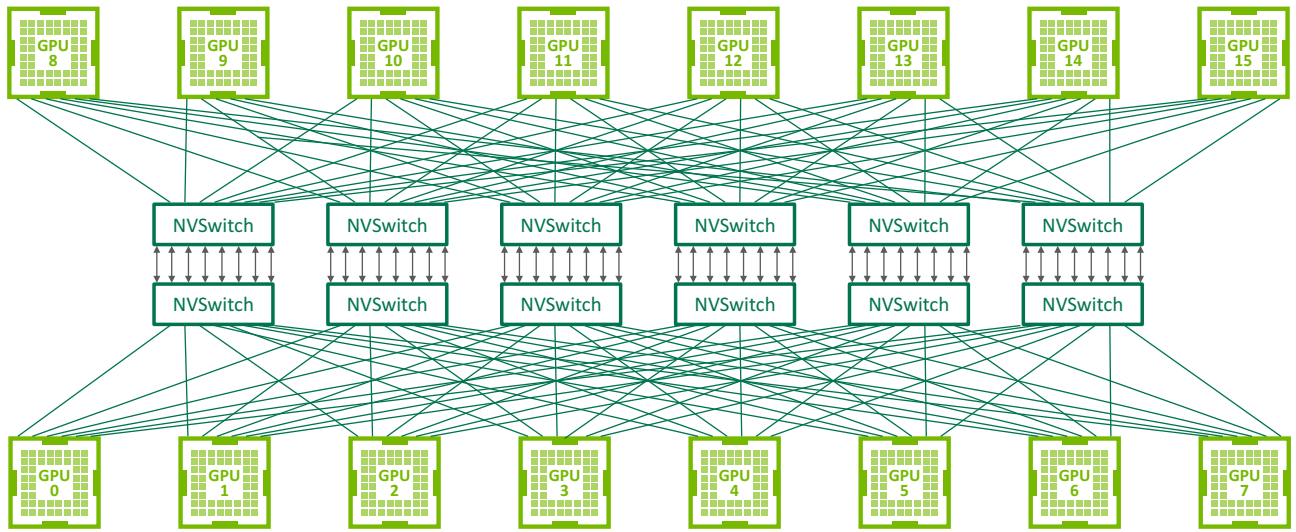
300 GB/s bidirectional bandwidth between any pair of GPUs

2.4 TB/s bisection bandwidth

Full atomics to Peer GPUs

High speed copy engines for bulk data copy

PCIe to/from CPU



NVLINK TO CPU

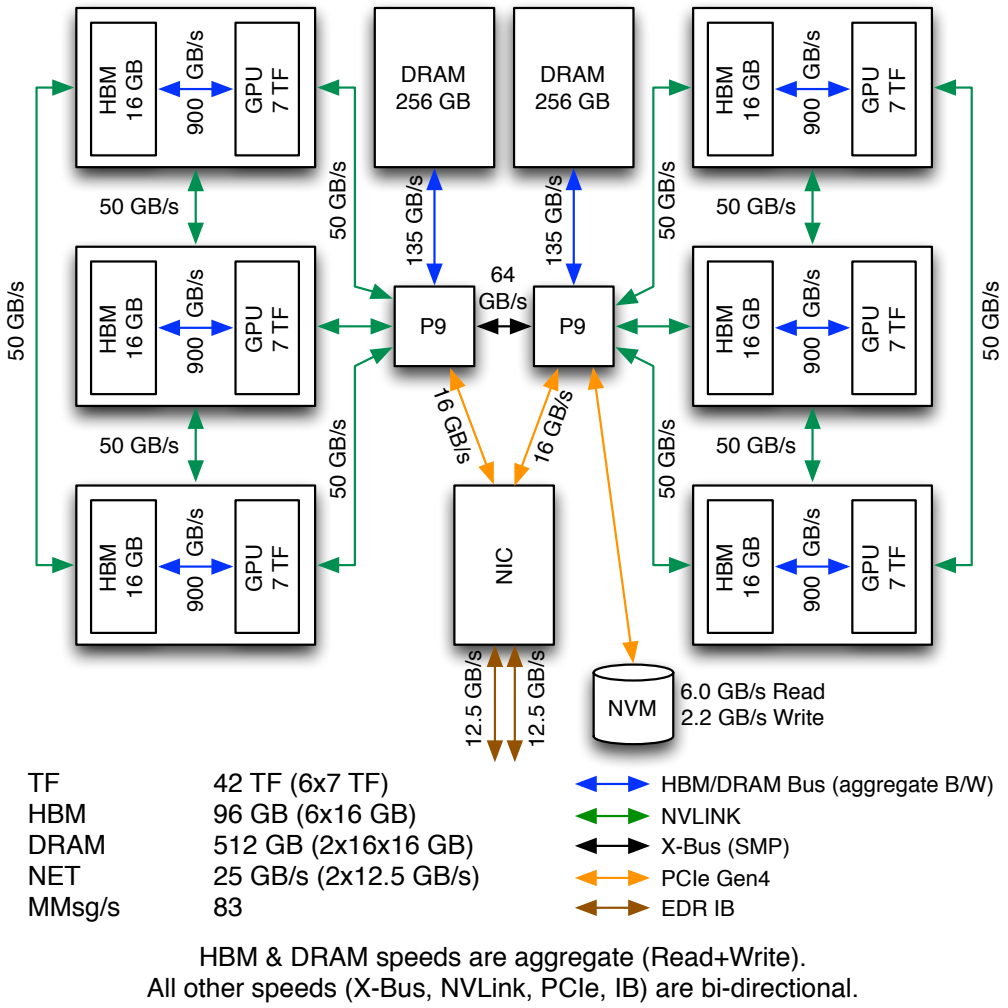
Fully connected quad

150 GB/s per GPU bidirectional for peer traffic

50 GB/s per GPU bidirectional to CPU

Direct Load/store access to CPU Memory

High Speed Copy Engines for bulk data movement



HBM & DRAM speeds are aggregate (Read+Write).
All other speeds (X-Bus, NVLink, PCIe, IB) are bi-directional.

OPENACC DIRECTIVES

Data Management

```
#pragma acc enter data create(A[0:nx*ny],Aref[0:nx*ny],Anew[0:nx*ny],rhs[0:nx*ny])
...
#pragma acc update device(A[(iy_start-1)*nx:((iy_end-iy_start)+2)*nx])
#pragma acc update device(rhs[iy_start*nx:(iy_end-iy_start)*nx])
while ( error > tol && iter < iter_max ) {
  ...
  iter++;
}
#pragma acc update self(A[(iy_start-1)*nx:((iy_end-iy_start)+2)*nx])
...
#pragma acc exit data delete(A,Aref,Anew,rhs)
```

UNIFIED MEMORY FUNDAMENTALS

Single pointer

On-demand migration

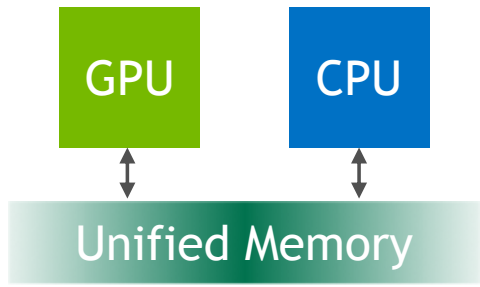
GPU memory oversubscription

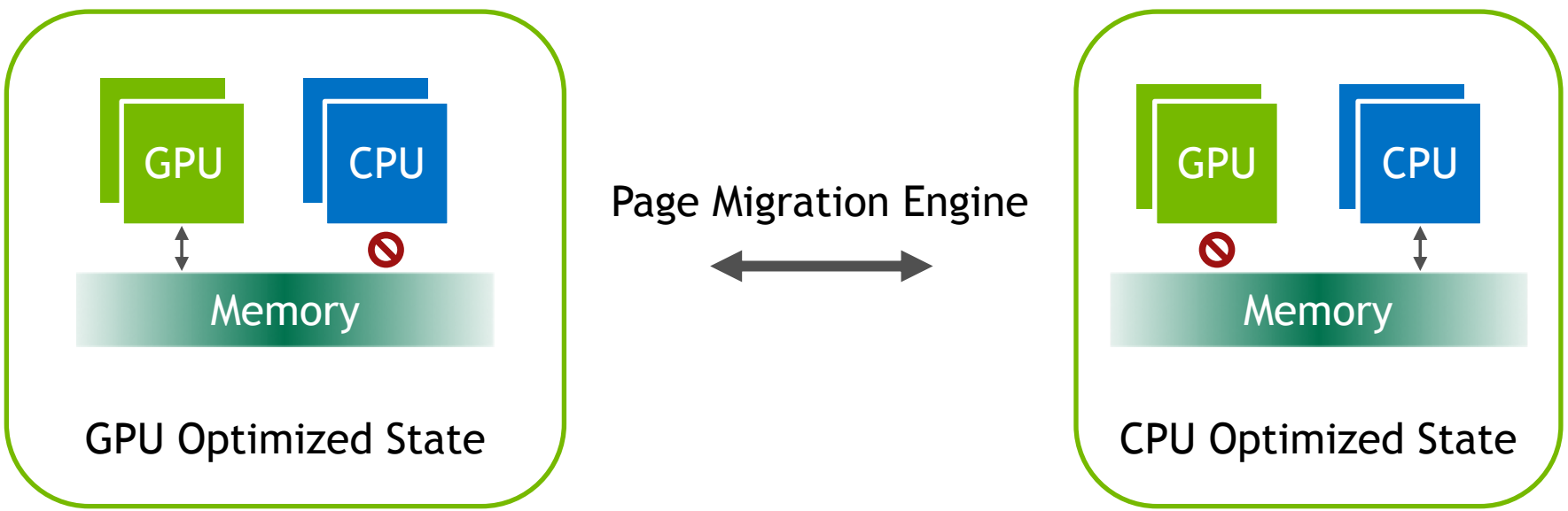
Concurrent CPU/GPU access

System-wide atomics

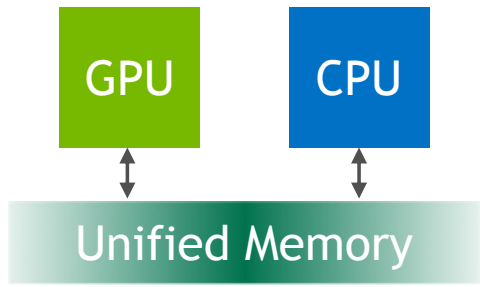
```
void *data;
data = malloc(N);

cpu_func1(data, N);

gpu_func2<<<...>>>(data, N);
cudaDeviceSynchronize();

cpu_func3(data, N);

free(data);
```

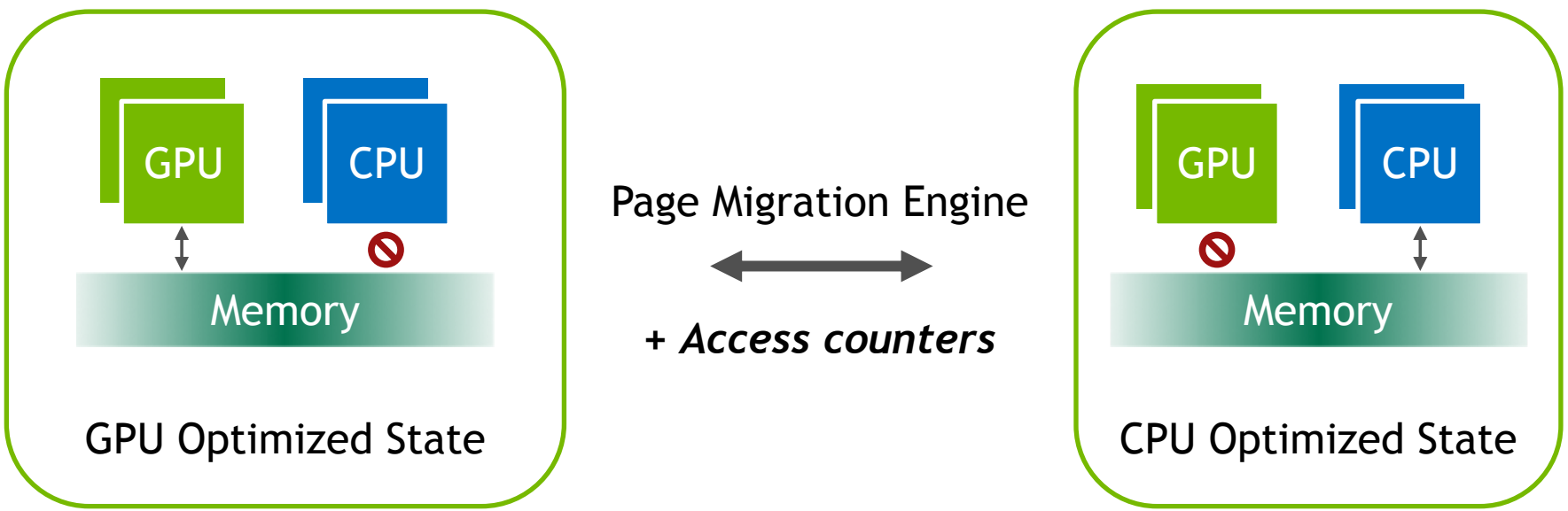
PASCAL UNIFIED MEMORY



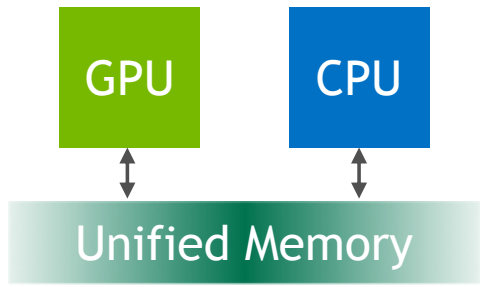


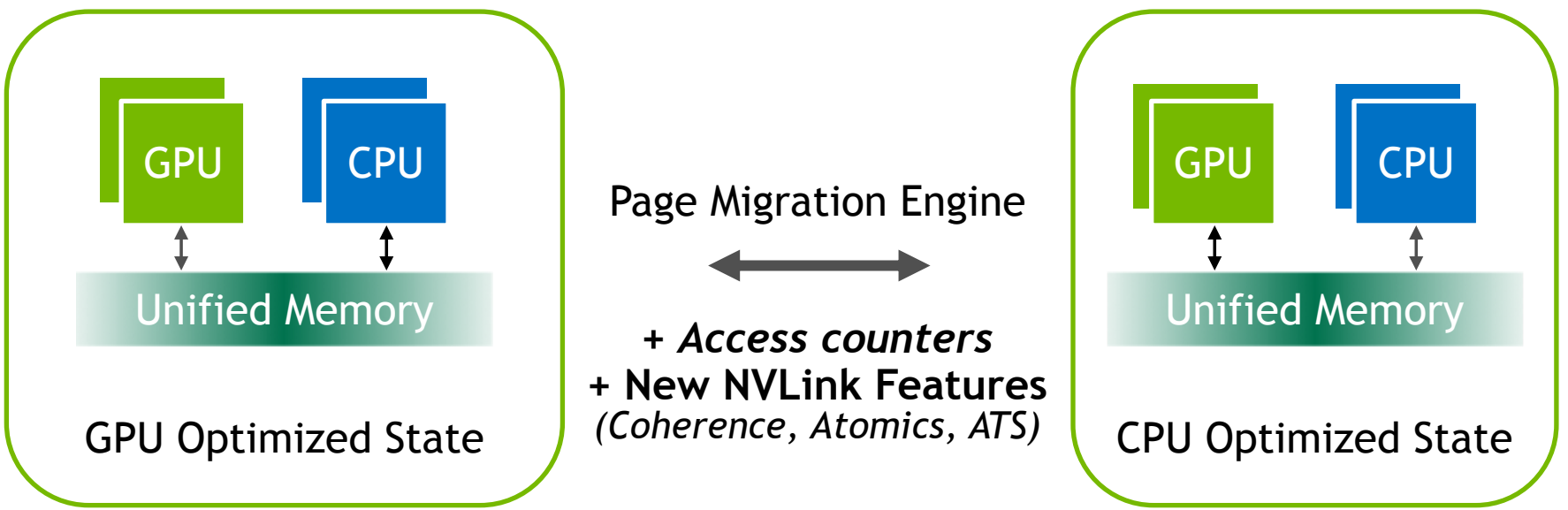
VOLTA + PCIE CPU UNIFIED MEMORY





VOLTA + NVLINK CPU UNIFIED MEMORY





P9+V100 NEW FEATURES

NVLINK2: increased migration throughput

NVLINK2: enabling HW coherency (CPU access GPU memory)

Indirect peers: GPU access memory of remote GPUs on a different socket

Native atomics support for all accessible memory

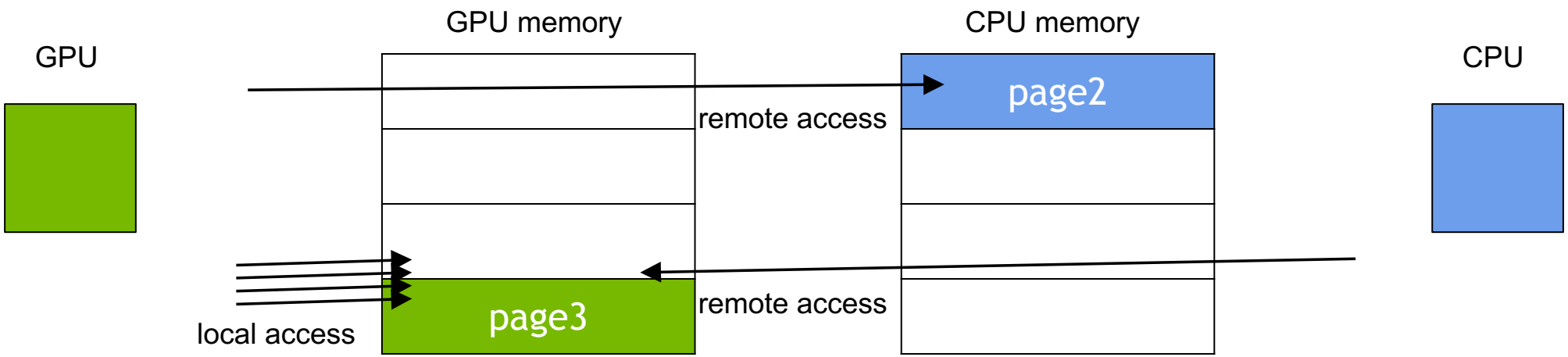
cudaMallocManaged may use access counters to guide migrations (opt-in)

ATS: GPU can access all system memory (malloc, stack, mmap files)

NVLINK2: COHERENCY

Works for cudaMallocManaged and malloc

CPU can directly access and cache GPU memory; *native* CPU-GPU atomics



page2 maps to CPU physical memory, page3 maps to GPU physical memory

LINUX AND UNIFIED MEMORY

ANY memory will be available for GPU*

CPU code

```
void sortfile(FILE *fp, int N) {
  char *data;
  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

GPU code with Unified Memory

```
void sortfile(FILE *fp, int N) {
  char *data;
  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort<<<...>>>(data,N,1,compare);
  cudaDeviceSynchronize();

  use_data(data);

  free(data);
}
```

*on supported operating systems


OPTIMIZE KERNEL
PERFORMANCE

PROFILING OPENACC APPLICATIONS

Using pgprof

Use on the command line to get a flat profile

```
pgprof ./app
```

Collect performance metrics

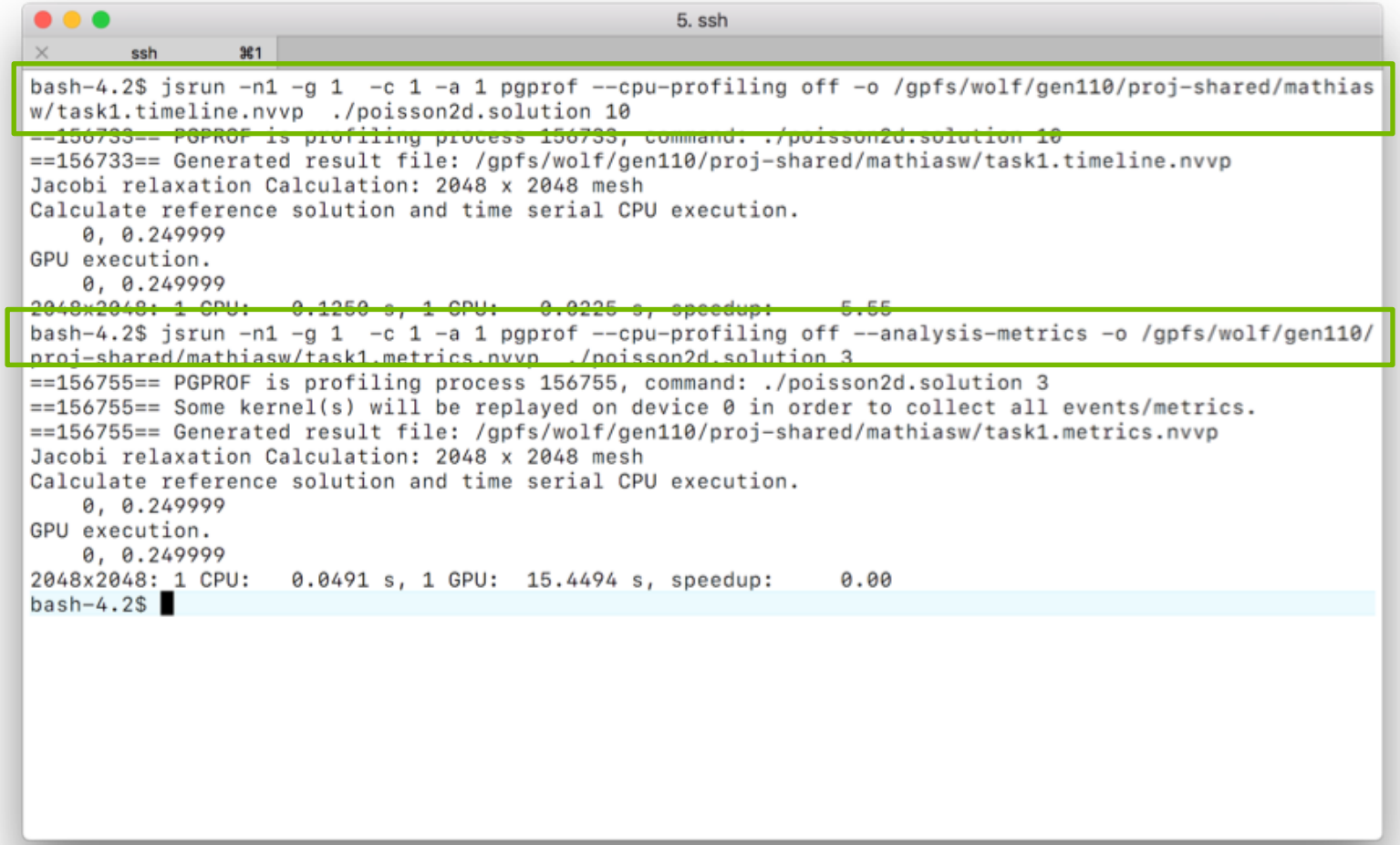
```
pgprof --metrics gld_efficiency,gst_efficiency ./app
```

or write output to file and import into pgprof GUI:

```
pgprof -o timeline.pgprof ./app

pgprof --analysis-metrics -o metrics.pgprof ./app
```

PROFILING OPENACC APPLICATIONS

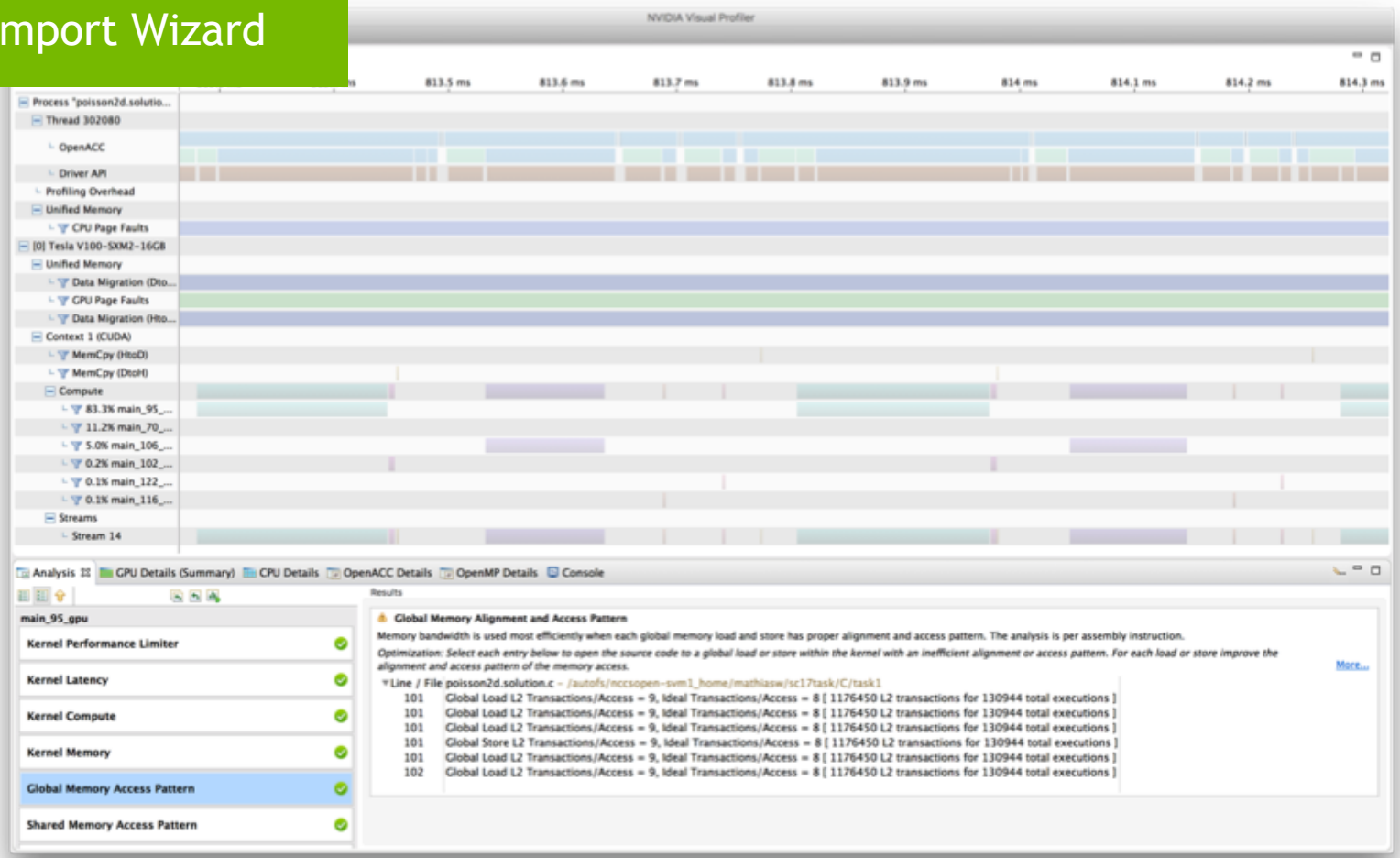
Using pgprof



PROFILING OPENACC APPLICATIONS

Using `pgprof`

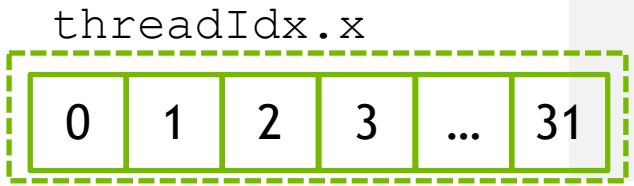
Use the import Wizard



ACCESSING GLOBAL MEMORY

Best Practices

- Threads issue instructions in groups of 32 threads (warp) operating in SIMT mode
- Each issued memory instructions generates multiple 32byte transactions
- 4 consecutive 32byte segments form a cache line
- The memory subsystem works most efficient if all moved data is consumed and as few memory instructions as possible are used

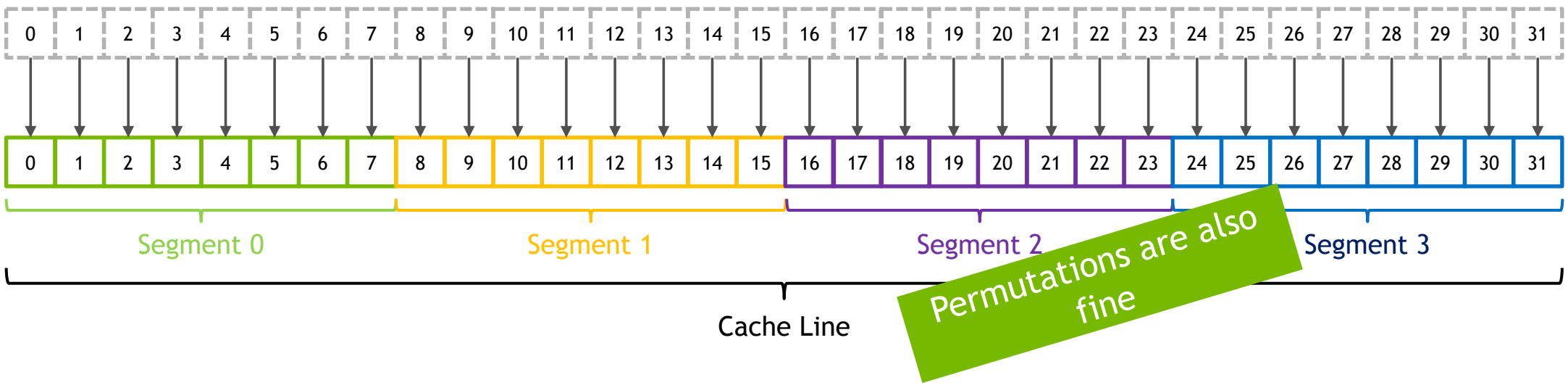



```
95, Accelerator kernel generated
    Generating Tesla code
    96, #pragma acc loop gang /* blockIdx.x */
    99, #pragma acc loop vector(128) /* threadIdx.x */
   104, Generating implicit reduction(max:error)
```

ACCESSING GLOBAL MEMORY

optimal access pattern (4byte words) – fully coalesced

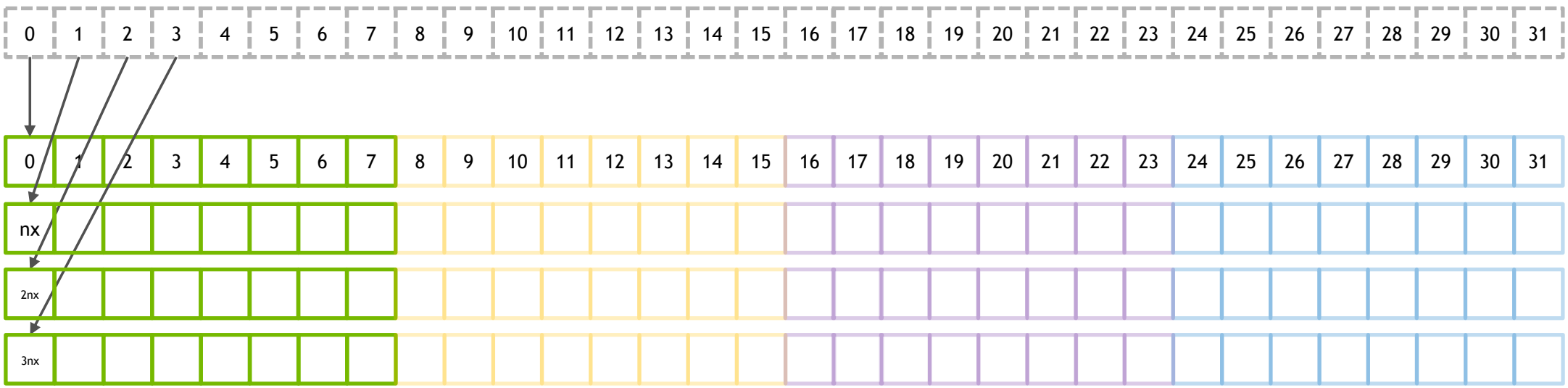
```
#pragma acc loop vector

for( int ix = 0; ix < nx; ix++ ) A[iy*nx+ix] = 0.0;
```



ACCESSING GLOBAL MEMORY

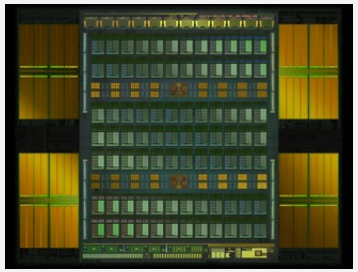
worst case access pattern (4byte words) – fully uncoalesced

```
#pragma acc loop vector

for( int iy = 0; iy < ny; iy++ ) A[iy*nx+ix] = 0.0;
```



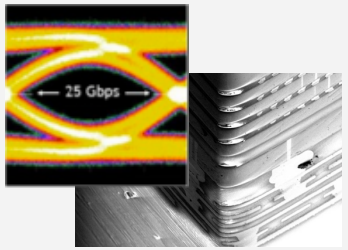
INTRODUCING TESLA V100

Volta Architecture



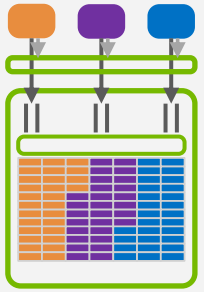
Most Productive GPU

Improved NVLink & HBM2



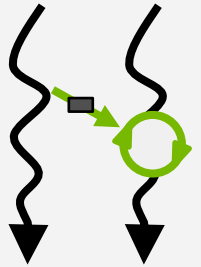
Efficient Bandwidth

Volta MPS



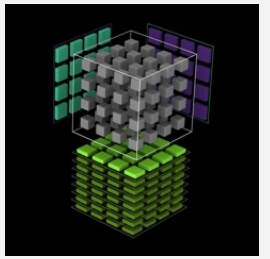
Inference Utilization

Improved SIMT Model



New Algorithms

Tensor Core



120 Programmable TFLOPS Deep Learning

More V100 Features: 2x L2 atomics, int8, new memory model, copy engine page migration, and more …

The Fastest and Most Productive GPU for Deep Learning and HPC


NVIDIA