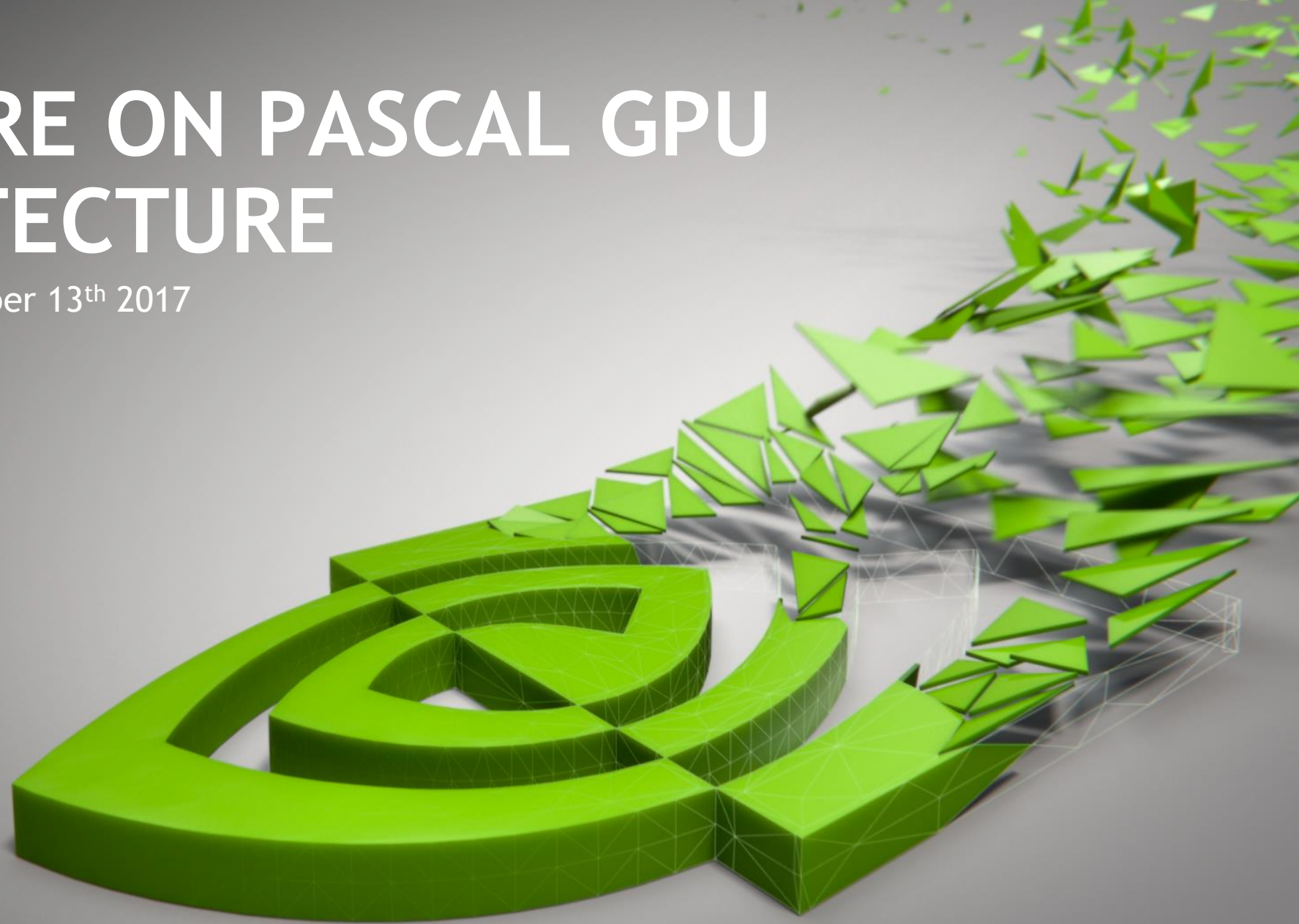


LECTURE ON PASCAL GPU ARCHITECTURE

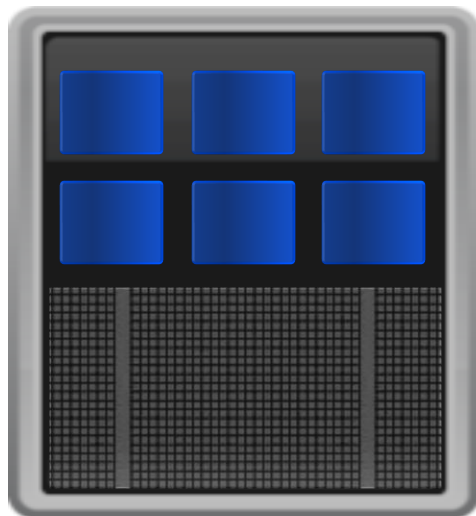
Jiri Kraus, November 13th 2017



ACCELERATED COMPUTING

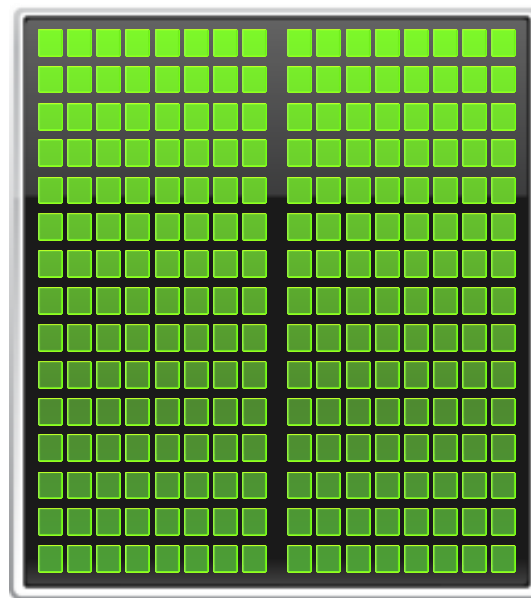
CPU

Optimized for
Serial Tasks



GPU Accelerator

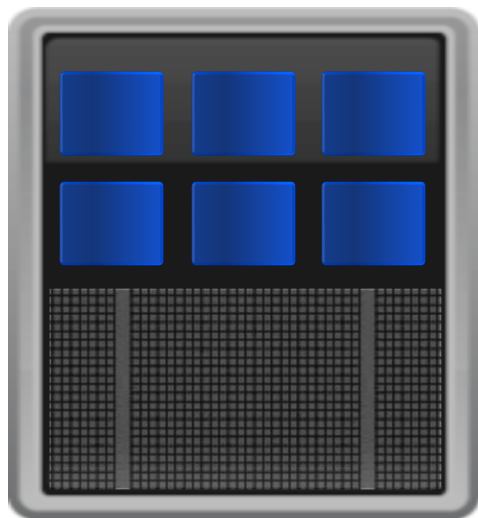
Optimized for
Parallel Tasks



ACCELERATED COMPUTING

CPU

Optimized for
Serial Tasks



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

ACCELERATED COMPUTING

GPU Strengths

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

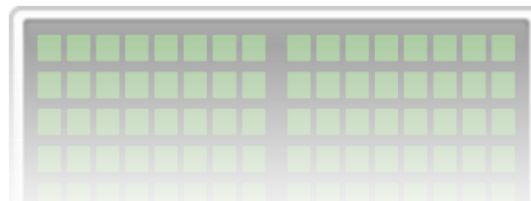
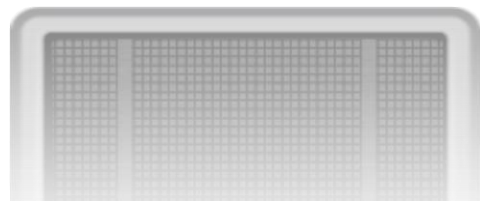
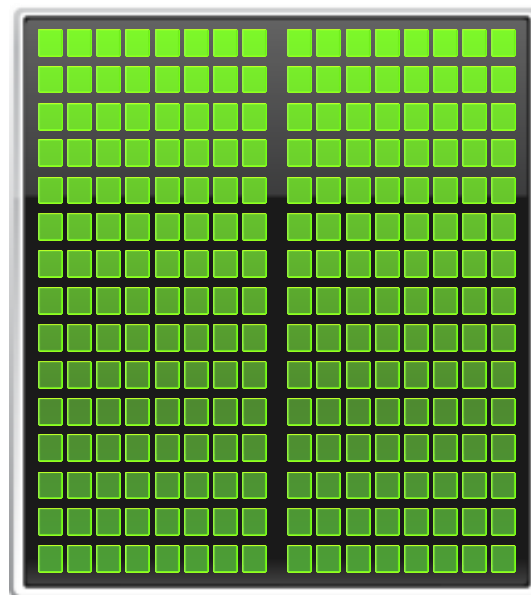
GPU Weaknesses

- Relatively low memory capacity
- Low per-thread performance



GPU Accelerator

Optimized for
Parallel Tasks



SPEED V. THROUGHPUT

Speed

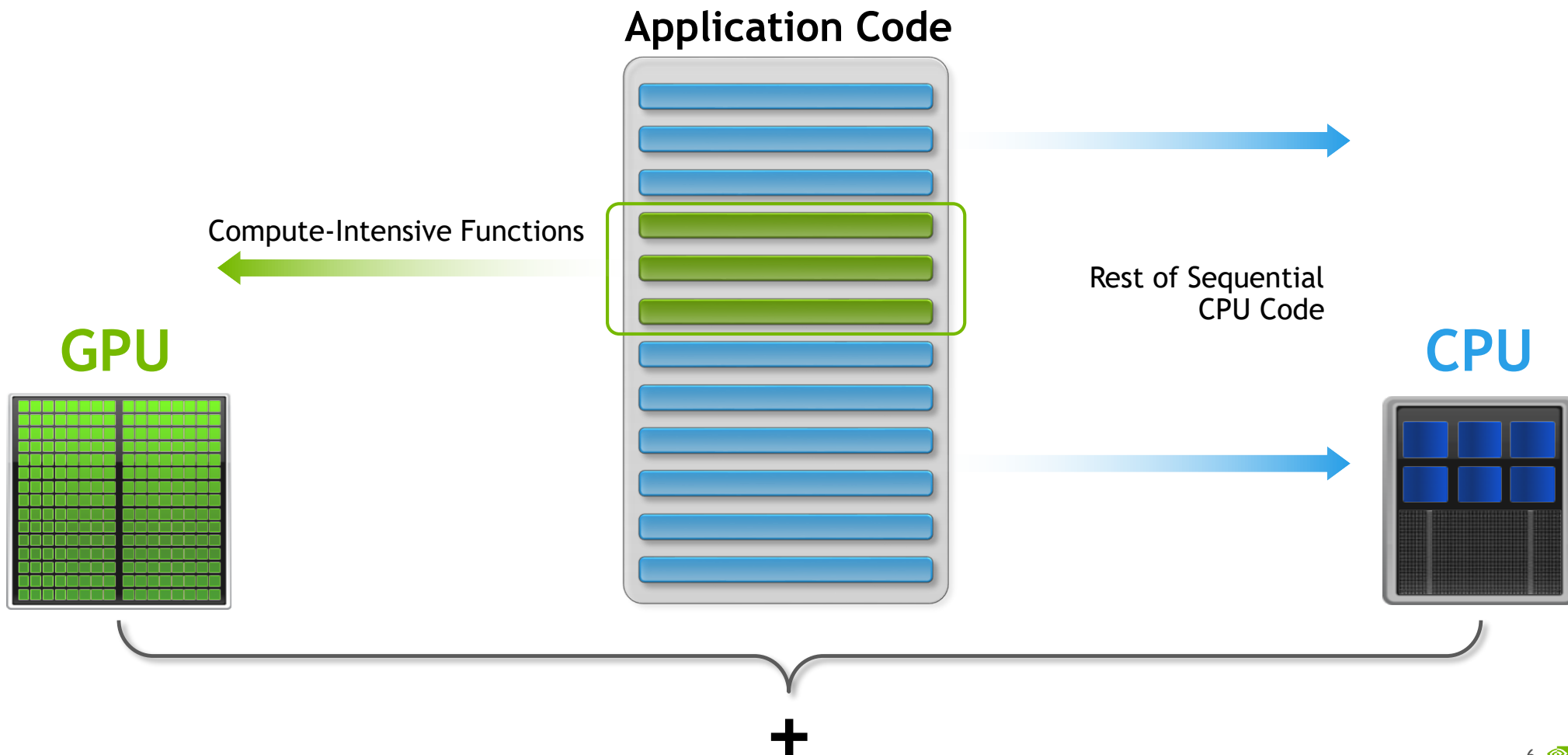


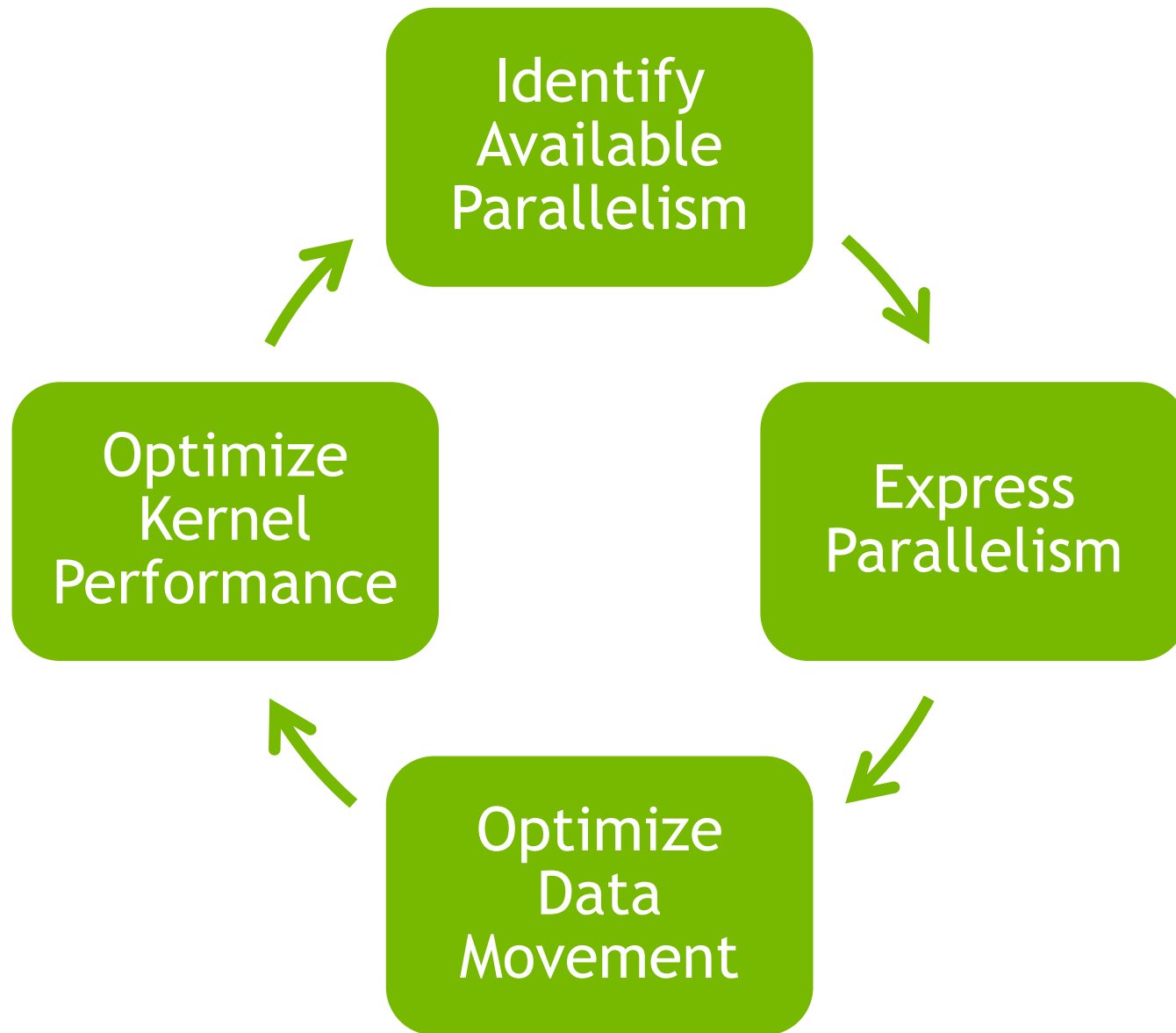
Throughput



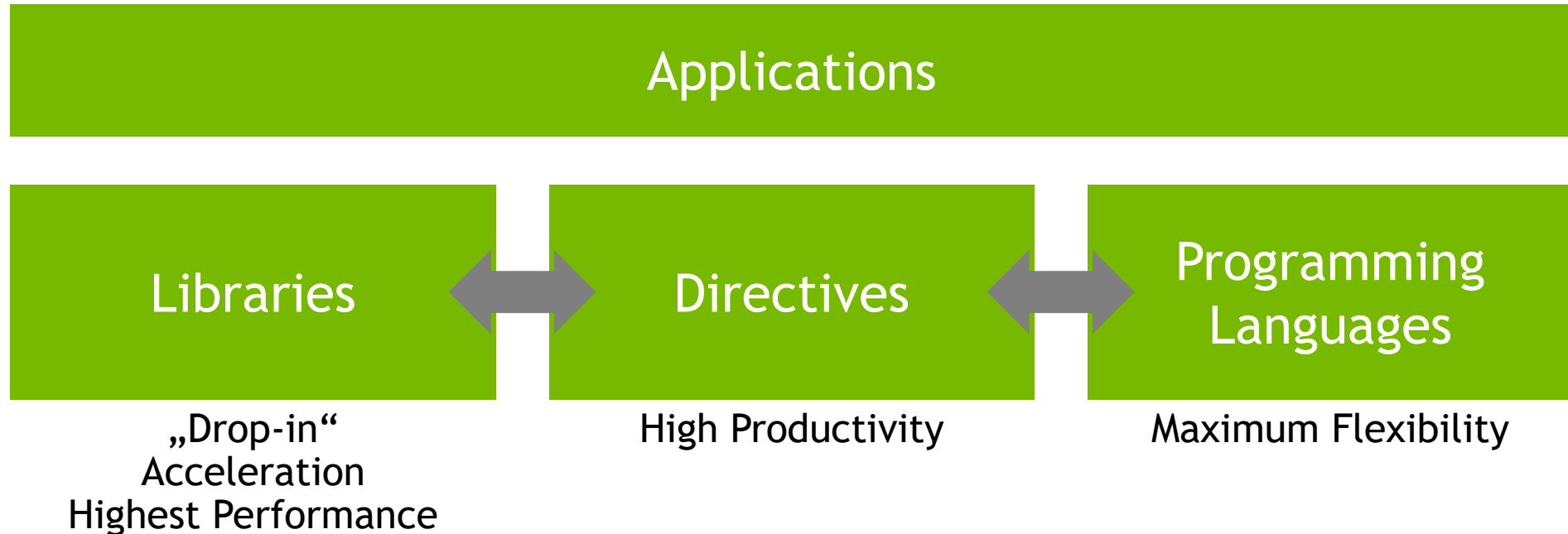
Which is better depends on your needs...

HOW GPU ACCELERATION WORKS





3 WAYS TO ACCELERATED 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 P100 GPU: GP100

56 SMs

3584 CUDA Cores

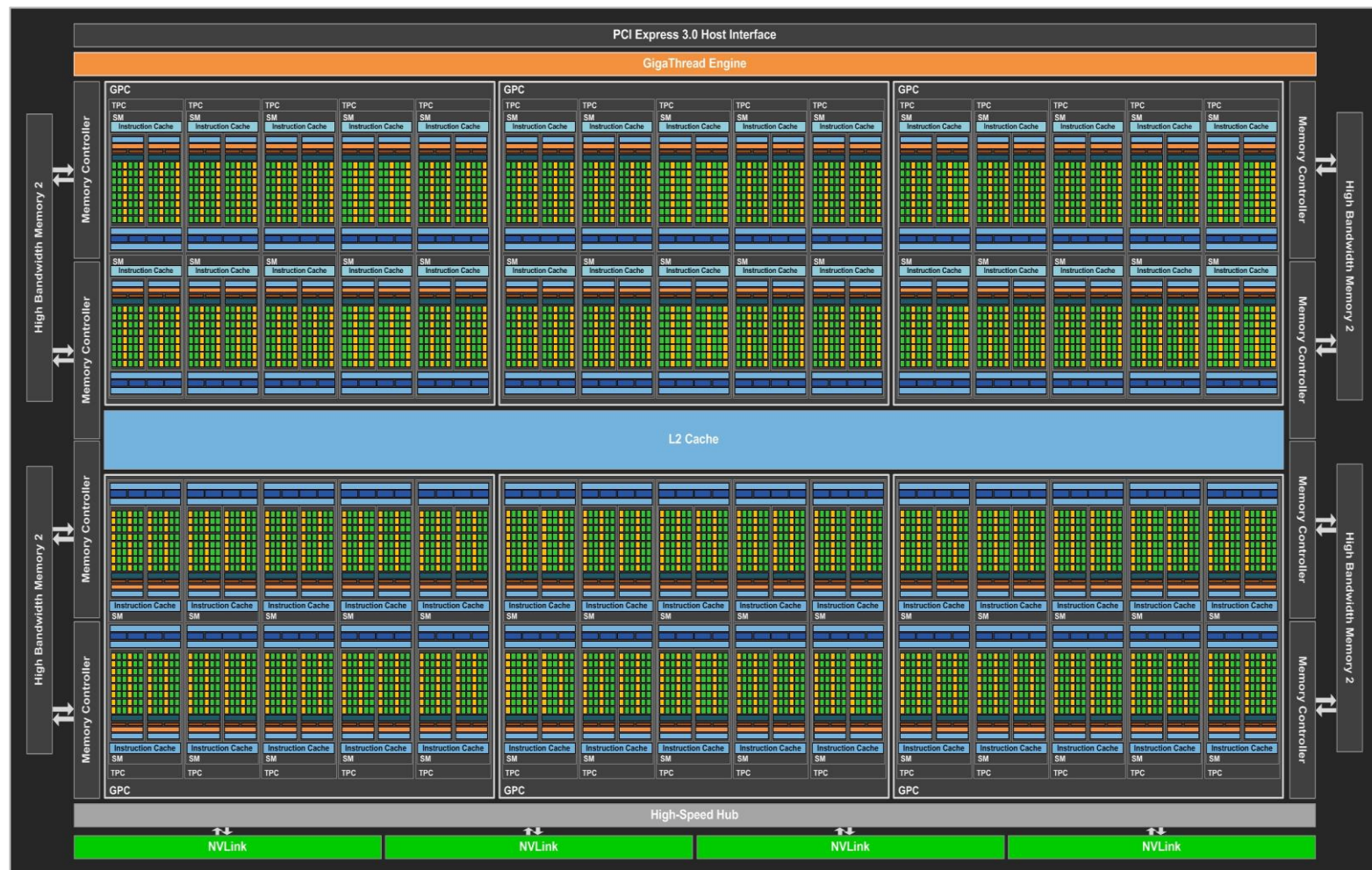
5.3 TF Double Precision

10.6 TF Single Precision

21.2 TF Half Precision

16 GB HBM2

732 GB/s Bandwidth



GP100 SM

GP100

CUDA Cores 64

Register File 256 KB

Shared Memory 64 KB

Active Threads 2048

Active Blocks 32



GPU PERFORMANCE COMPARISON

	P100 (SXM2)	M40	K40
Double/Single/Half TFlop/s	5.3/10.6/21.2	0.2/7.0/NA	1.4/4.3/NA
Memory Bandwidth (GB/s)	732	288	288
Memory Size	16GB	12GB, 24GB	12GB
L2 Cache Size	4096 KB	3072 KB	1536 KB
Base/Boost Clock (Mhz)	1328/1480	948/1114	745/875
TDP (Watts)	300	250	235

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 present(A,Anew)  
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

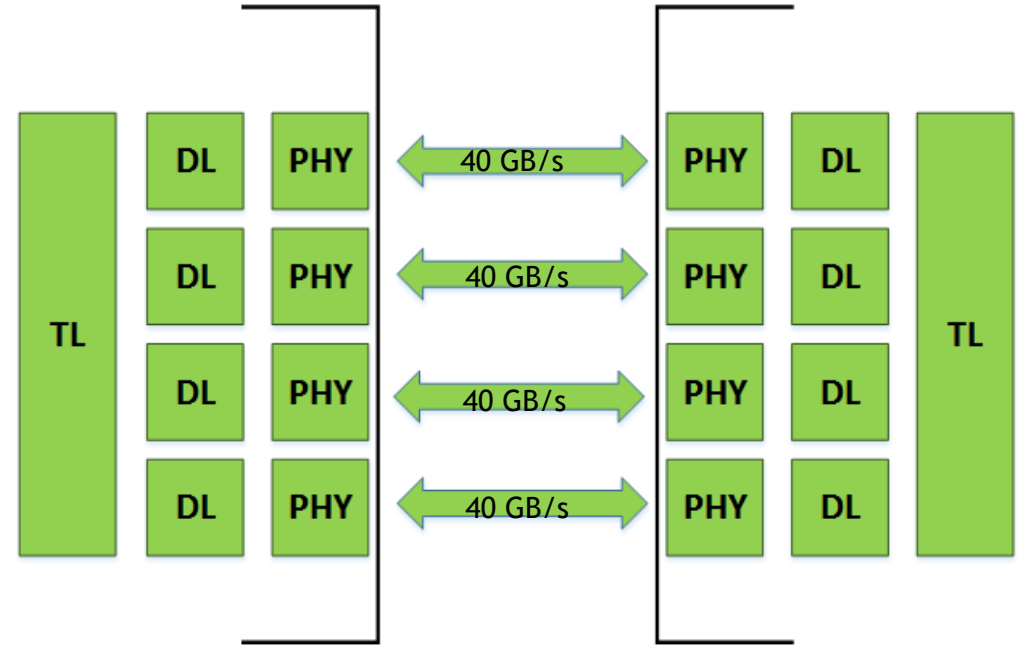
P100 supports 4 NVLinks

Up to 94% bandwidth efficiency

Supports read/writes/atomics to peer GPU

Supports read/write access to NVLink-enabled CPU

Links can be ganged for higher bandwidth



NVLink on Tesla P100

NVLINK - GPU CLUSTER

Two fully connected quads,
connected at corners

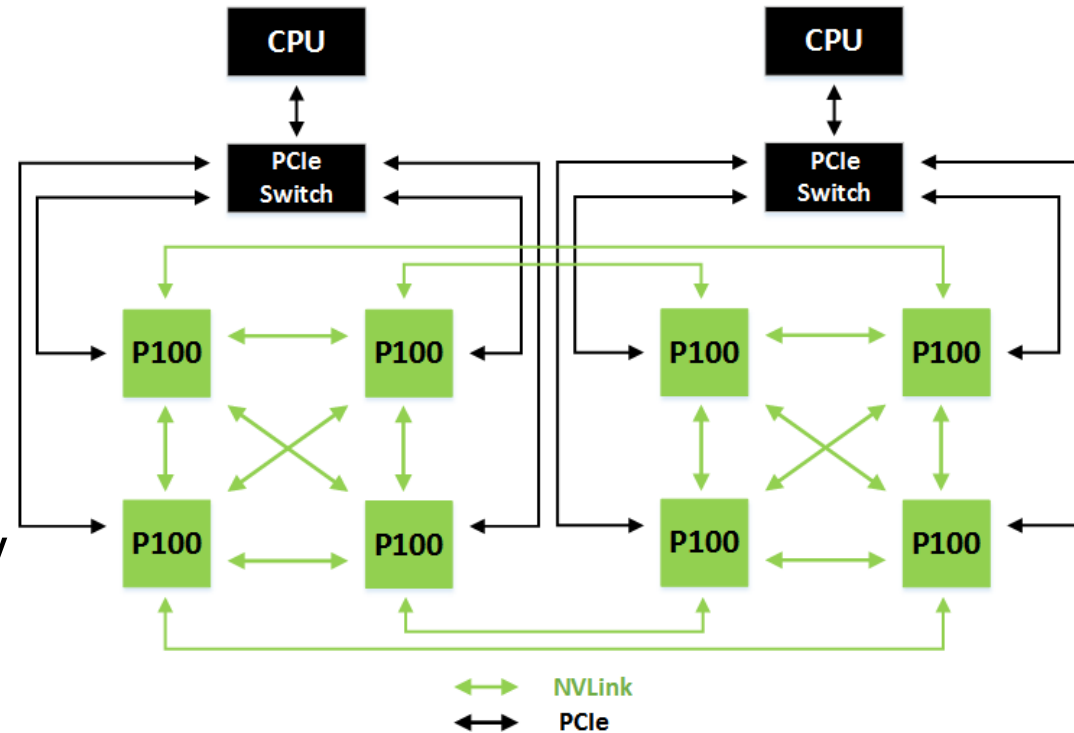
160GB/s per GPU bidirectional to Peers

Load/store access to Peer Memory

Full atomics to Peer GPUs

High speed copy engines for bulk data copy

PCIe to/from CPU



NVLINK TO CPU

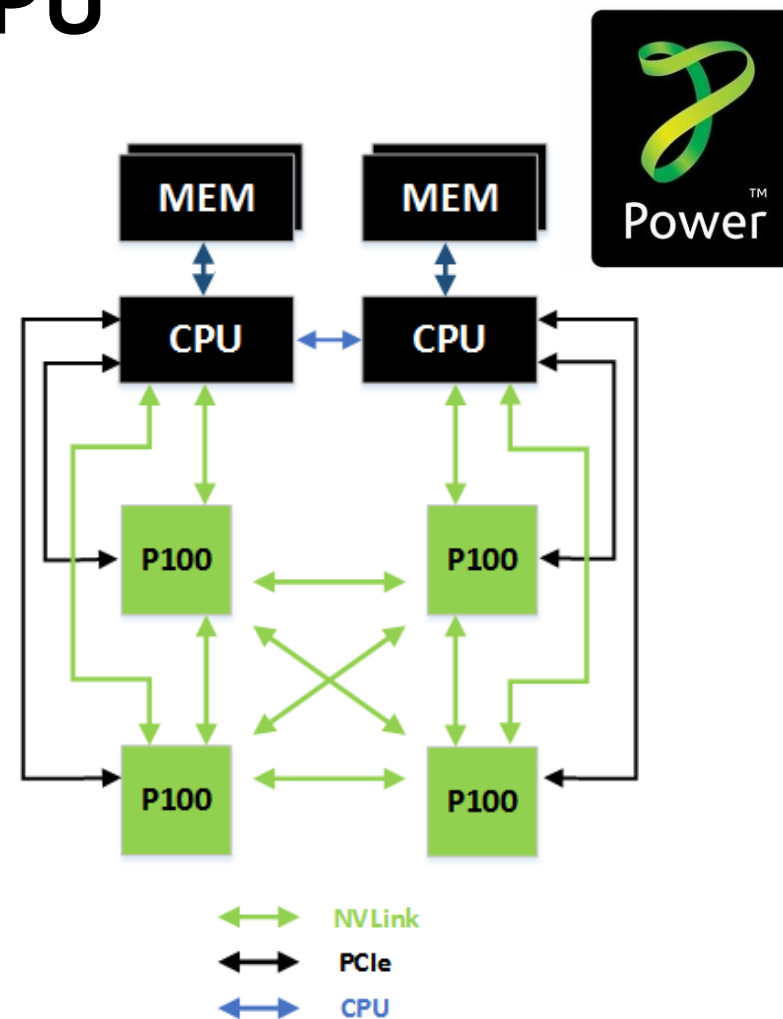
Fully connected quad

120 GB/s per GPU bidirectional for peer traffic

40 GB/s per GPU bidirectional to CPU

Direct Load/store access to CPU Memory

High Speed Copy Engines for bulk data movement



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)
```

PAGE MIGRATION ENGINE

Support Virtual Memory Demand Paging

49-bit Virtual Addresses

Sufficient to cover 48-bit CPU address + all GPU memory

GPU page faulting capability

Can handle thousands of simultaneous page faults

Up to 2 MB page size

Better TLB coverage of GPU memory

UNIFIED MEMORY

Single pointer for CPU and GPU

CPU code

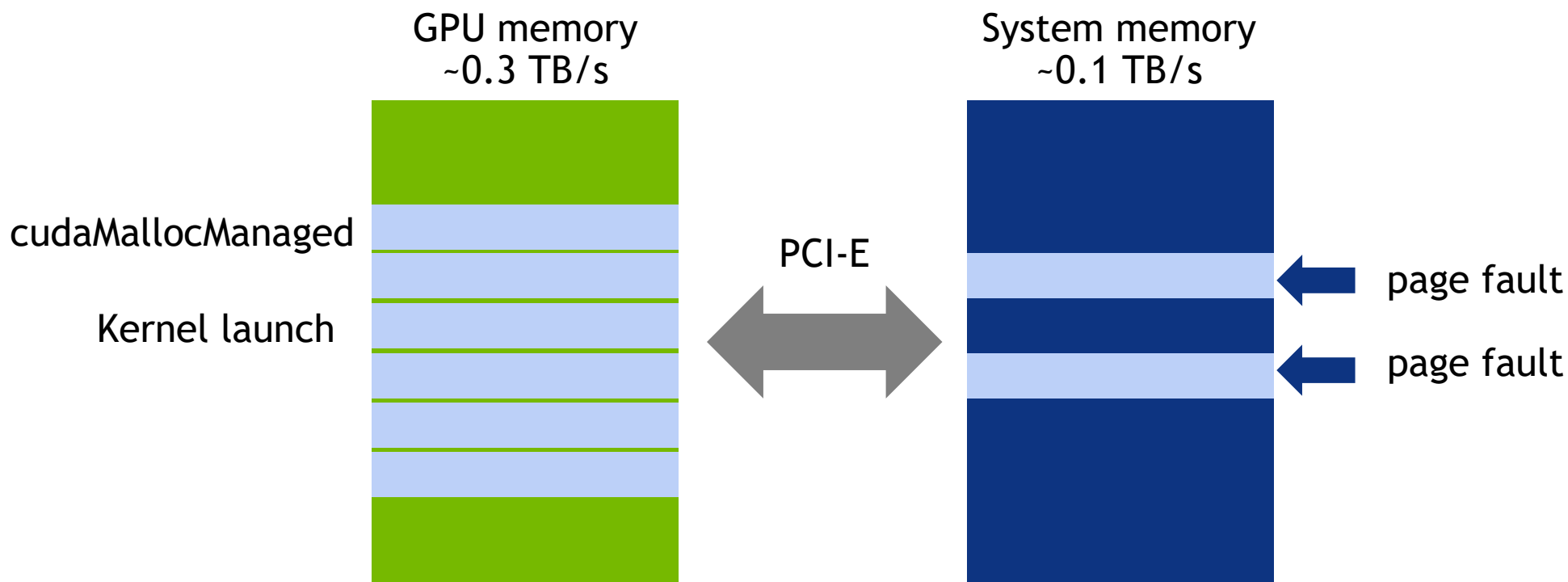
```
void sortfile(FILE *fp, int N) {  
    char *data;  
    data = (char *)malloc(N);  
  
    fread(data, 1, N, fp);  
  
    qsort(data, N, 1, compare);  
  
    use_data(data);  
  
    free(data);  
}
```

GPU code with Unified Memory

```
void sortfile(FILE *fp, int N) {  
    char *data;  
    cudaMallocManaged(&data, N);  
  
    fread(data, 1, N, fp);  
  
    qsort<<<...>>>(data, N, 1, compare);  
    cudaDeviceSynchronize();  
  
    use_data(data);  
  
    cudaFree(data);  
}
```

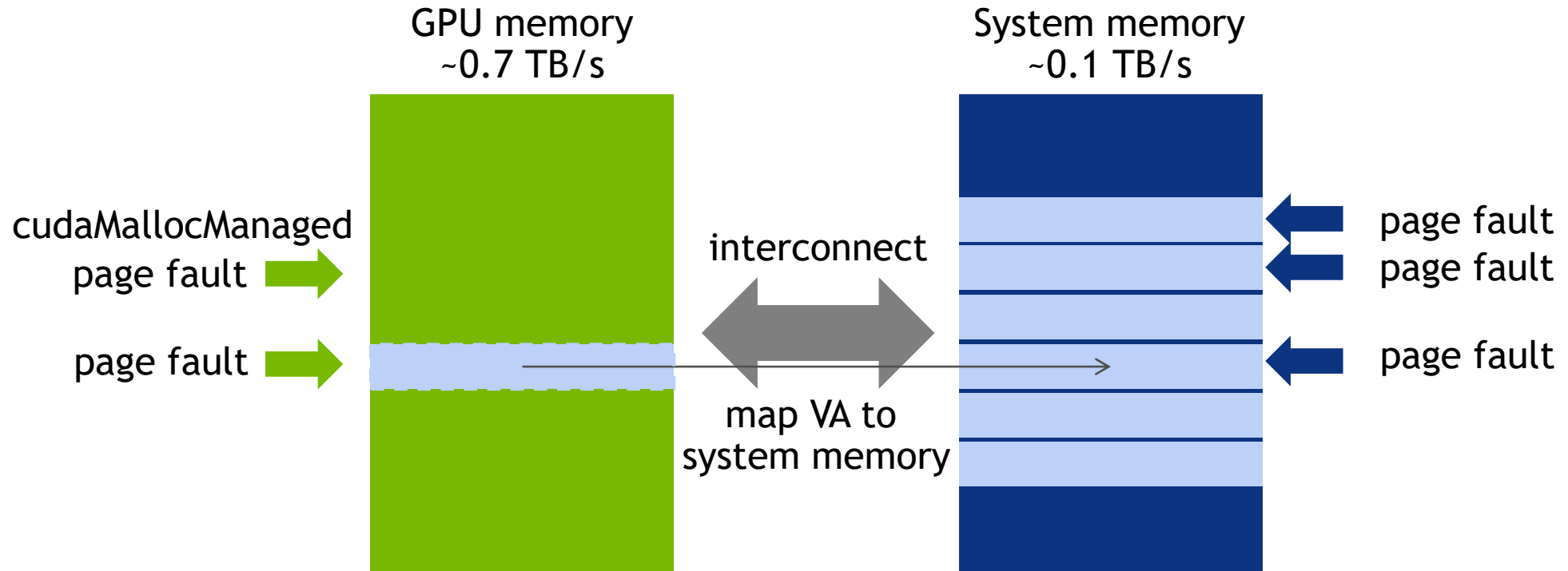
UNIFIED MEMORY ON PRE-PASCAL

Kernel launch triggers bulk page migrations



UNIFIED MEMORY ON PASCAL

On-demand page migrations



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

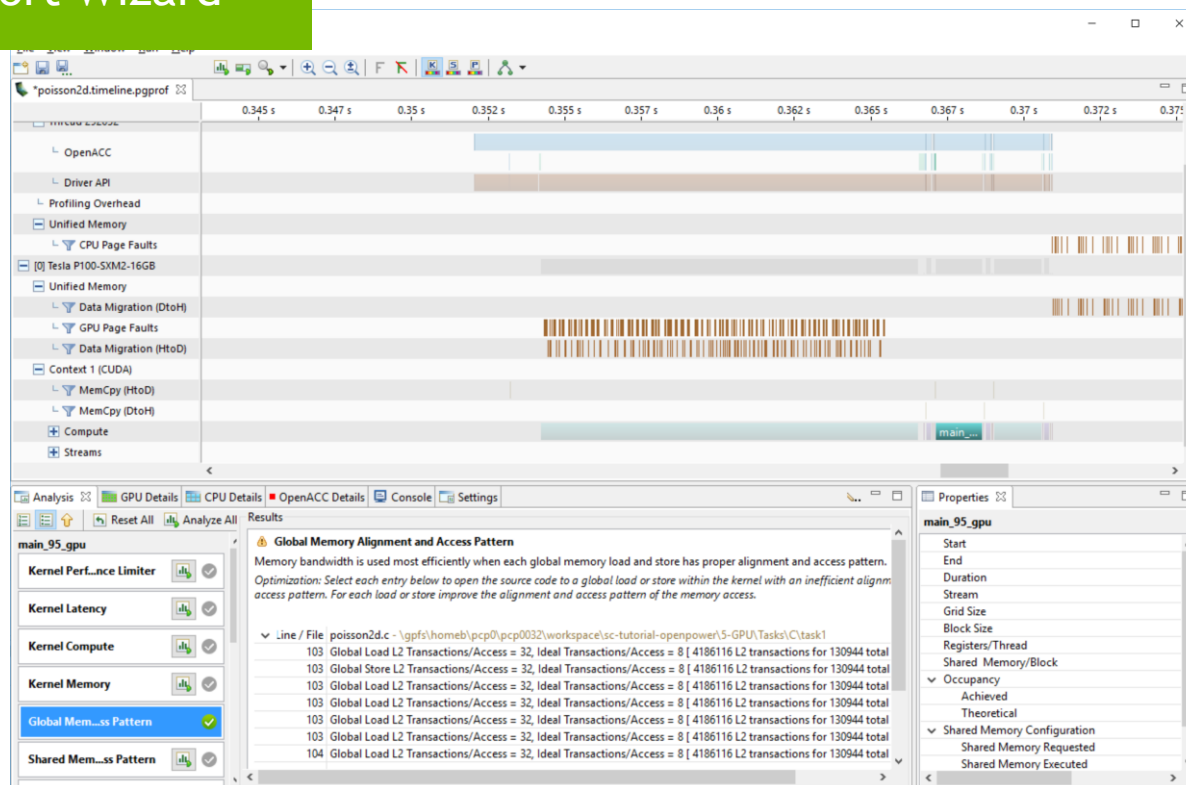
Using pgprof

```
juron1.fz-juelich.de - PuTTY
-bash-4.2$ make profile
bsub -env "all, TMPDIR=/tmp" -n 1 -I -R "rusage[ngpus_sh
u-profiling off -o poisson2d.timeline.pgprof ./poisson2d
Job <3428> is submitted to default queue <normal.i>.
<<Waiting for dispatch ...>>
<<Starting on juroncl5>>
==23439== PGPROF is profiling process 23439, command: ./p
==23439== Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU executio
0, 0.249999
GPU execution.
0, 0.249999
2048x2048: 1 CPU: 0.1301 s, 1 GPU: 0.0234 s, speedup:
Generated result file: /gpfs/homeb/pcp0/pcp0032/workspace
5-GPU/Tasks/C/task1/poisson2d.timeline.pgprof
bsub -env "all, TMPDIR=/tmp" -n 1 -I -R "rusage[ngpus_sh
u-profiling off --analysis-metrics -o poisson2d.metrics.p
Job <3429> is submitted to default queue <normal.i>.
<<Waiting for dispatch ...>>
<<Starting on juroncl6>>
===== Warning: Metric "l1_shared_utilization" cannot b
===== Warning: Metric "l1_shared_utilization" cannot b
===== Warning: Metric "l1_shared_utilization" cannot b
===== Warning: Metric "l1_shared_utilization" cannot b
===== Warning: Metric "l2_l1_read_throughput" cannot be found on device 3.
===== Warning: Metric "l2_l1_write_throughput" cannot be found on device 0.
===== Warning: Metric "l2_l1_write_throughput" cannot be found on device 1.
===== Warning: Metric "l2_l1_write_throughput" cannot be found on device 2.
===== Warning: Metric "l2_l1_write_throughput" cannot be found on device 3.
===== Warning: Metric "nc_l2_read_throughput" cannot be found on device 0.
===== Warning: Metric "nc_l2_read_throughput" cannot be found on device 1.
===== Warning: Metric "nc_l2_read_throughput" cannot be found on device 2.
===== Warning: Metric "nc_l2_read_throughput" cannot be found on device 3.
===== Warning: Metric "atomic_throughput" cannot be found on device 0.
===== Warning: Metric "atomic_throughput" cannot be found on device 1.
===== Warning: Metric "atomic_throughput" cannot be found on device 2.
===== Warning: Metric "atomic_throughput" cannot be found on device 3.
==21760== PGPROF is profiling process 21760, command: ./poisson2d 3
==21760== Some kernel(s) will be replayed on device 0 in order to collect all ev
ents/metrics.
==21760== Generated result file: /gpfs/homeb/pcp0/pcp0032/workspace/sc-tutorial-
openpower/5-GPU/Tasks/C/task1/poisson2d.metrics.pgprof
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
0, 0.249999
GPU execution.
0, 0.249999
2048x2048: 1 CPU: 0.1313 s, 1 GPU: 13.9098 s, speedup: 0.01
```

PROFILING OPENACC APPLICATIONS

Using pgprof

Use the import Wizard

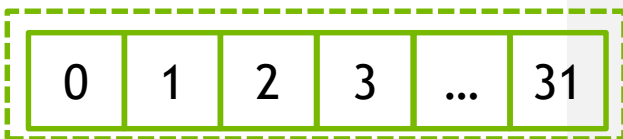


ACCESSING GLOBAL MEMORY

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

threadIdx.x



```
95, Accelerator kernel generated
```

```
Generating Tesla code
```

```
96, #pragma acc loop gang /* blockIdx.x */
```

```
99, #pragma acc loop vector(128) /* threadIdx.x */
```

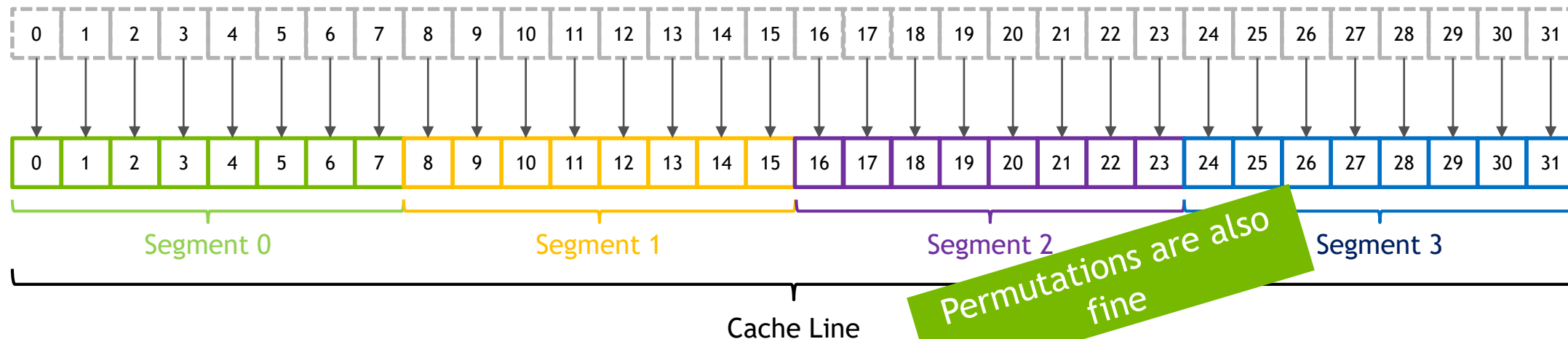
```
104, Generating implicit reduction(max:error)
```

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;
```

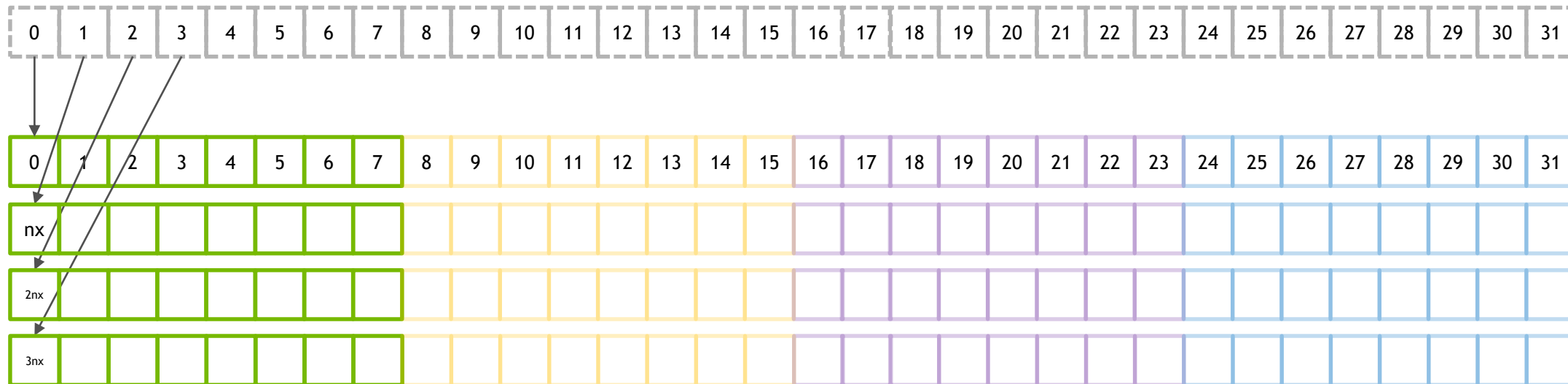


ACCESSING GLOBAL MEMORY

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 P100

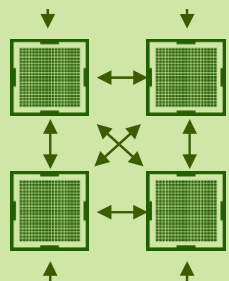
New GPU Architecture to Enable the World's Fastest Compute Node

Pascal Architecture



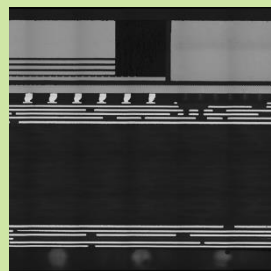
Highest Compute Performance

NVLink



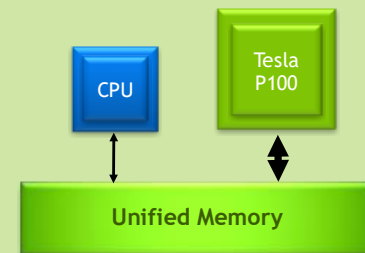
GPU Interconnect for Maximum Scalability

HBM2 Stacked Memory



Unifying Compute & Memory in Single Package

Page Migration Engine



Simple Parallel Programming with 512 TB of Virtual Memory

Find out more at <http://devblogs.nvidia.com/parallelforall/inside-pascal>