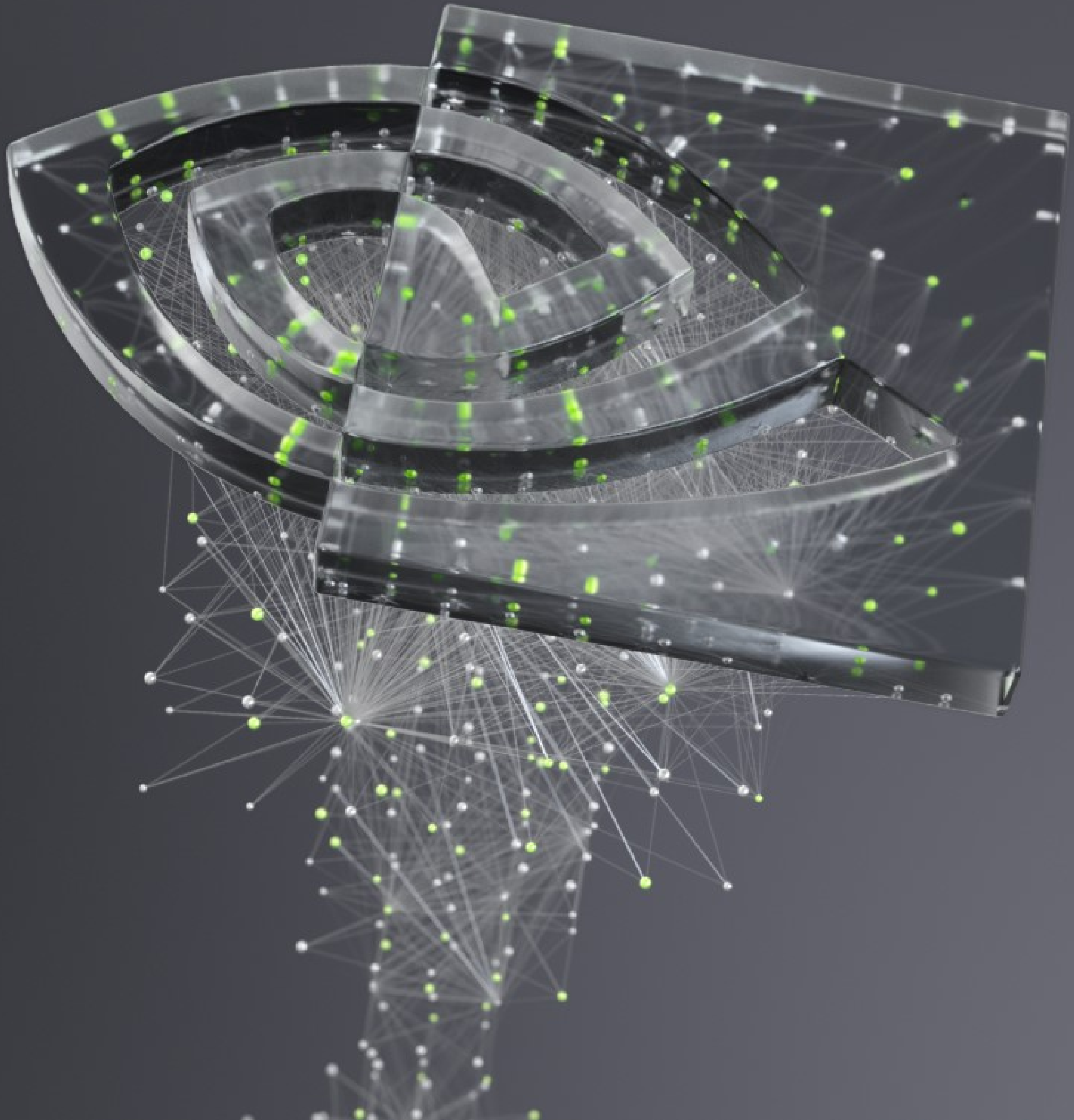


N-WAYS TO GPU COMPUTING

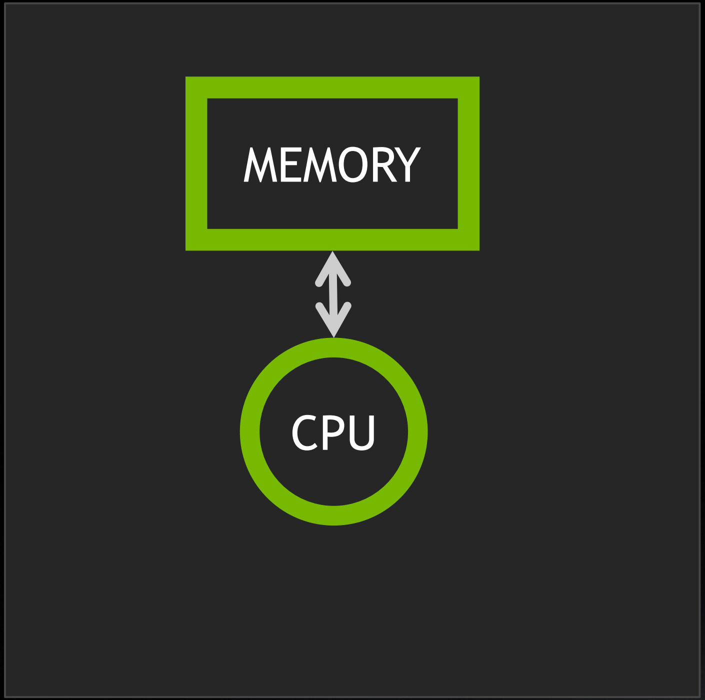
INTRODUCTION TO GPU COMPUTING

What to expect?

- Broad view on GPU Stack
- Fundamentals of GPU Architecture
- Good starting point

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

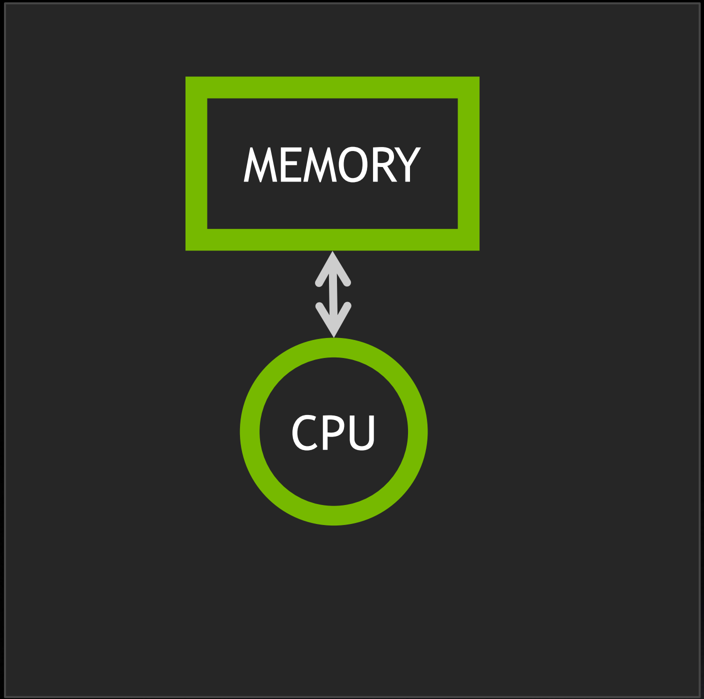
HPC SYSTEM EVOLUTION



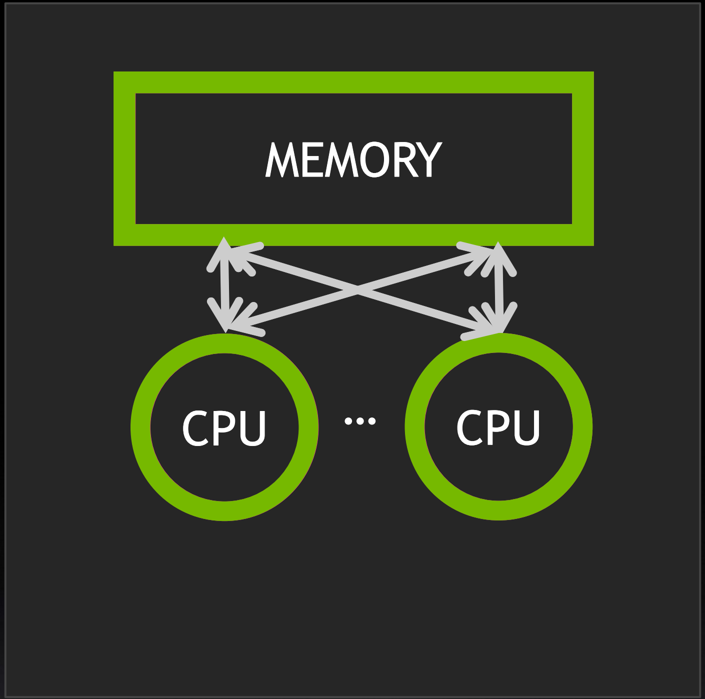
Sequential

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

HPC SYSTEM EVOLUTION



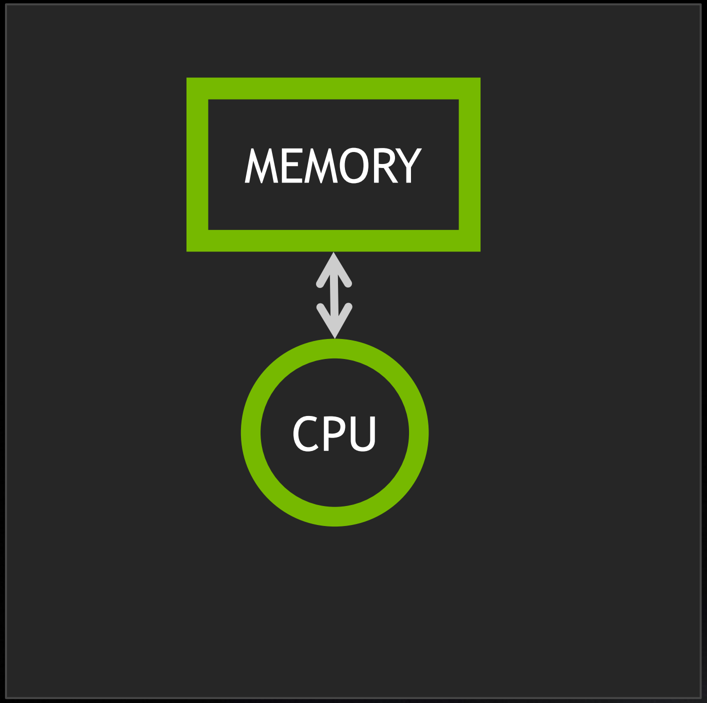
Sequential



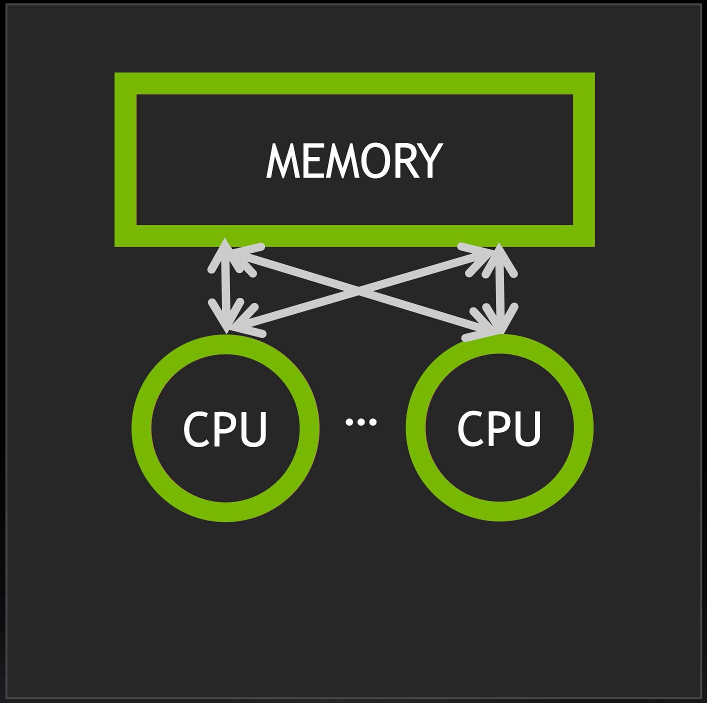
Multithreaded
P-Thread/OpenMP

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

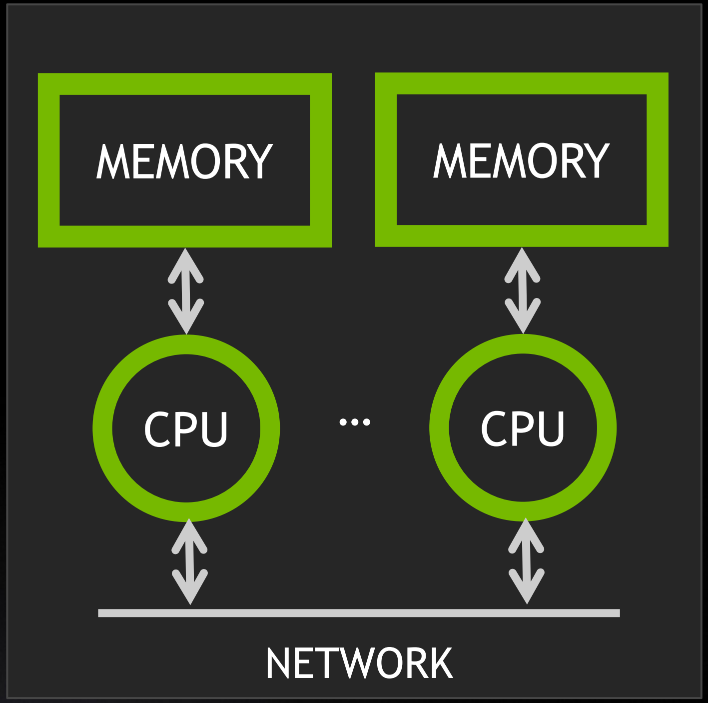
HPC SYSTEM EVOLUTION



Sequential



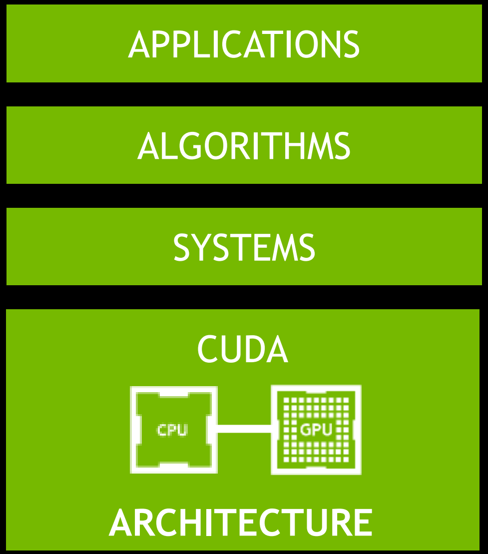
Multithreaded
P-Thread/OpenMP

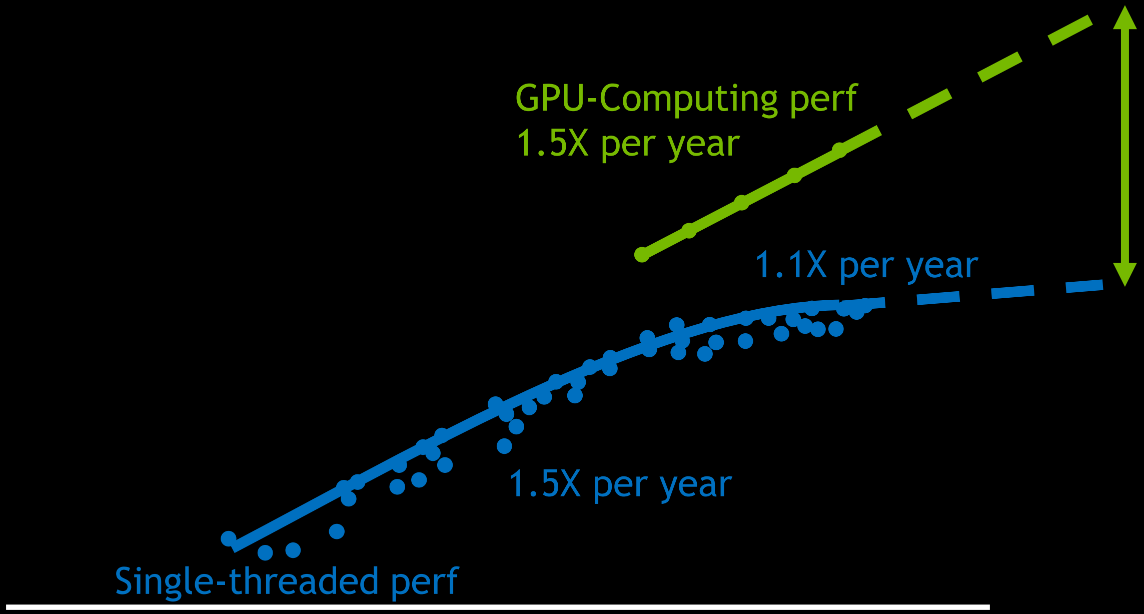


Distributed
MPI

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

GPU ARCHITECTURE CONTINUES TO DELIVER PERFORMANCE

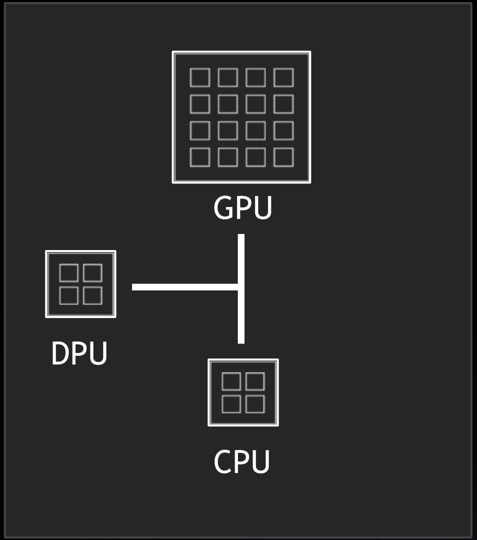




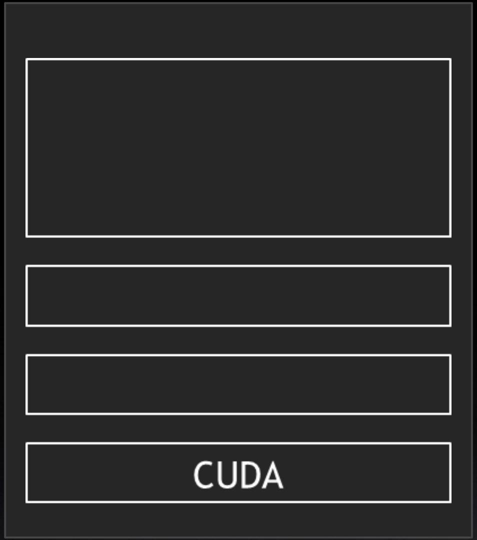
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

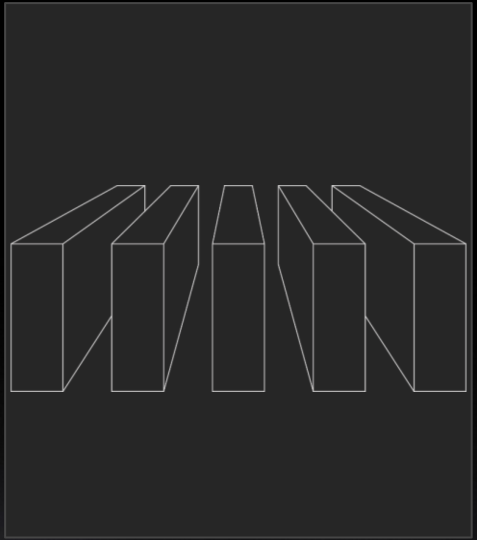
ACCELERATED COMPUTING PILLARS



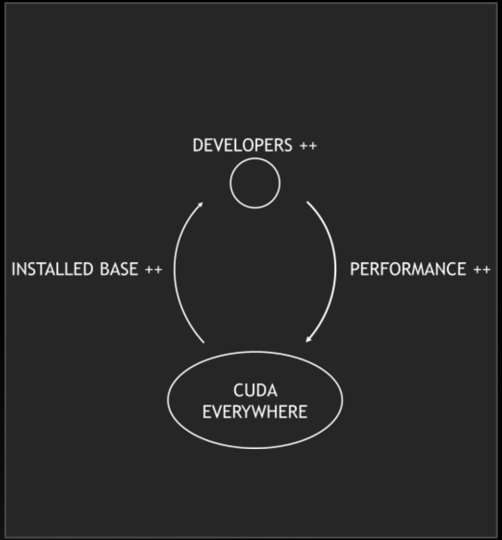
X-FACTOR SPEED UP



FULL STACK



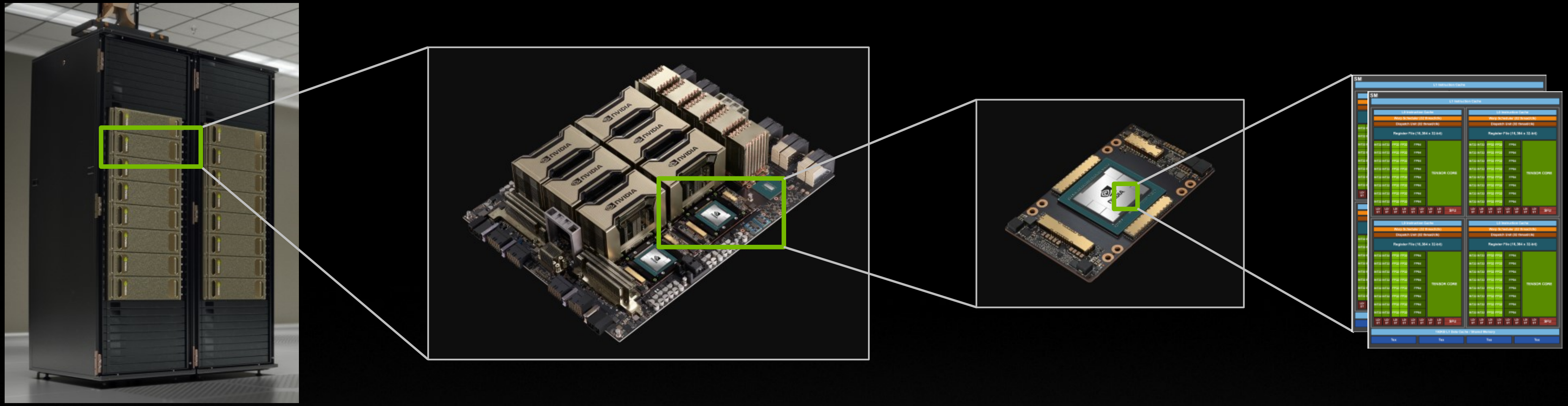
DATA-CENTER SCALE



Developer Productivity

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

HIERARCHY OF SCALES



Multi-System Rack
Unlimited Scale

Multi-GPU System
8 GPUs

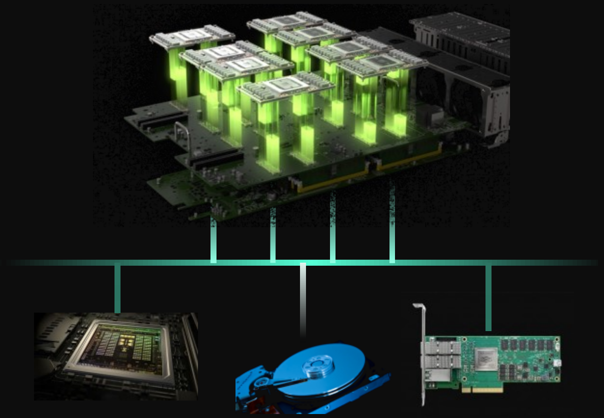
Multi-SM GPU
108 Multiprocessors

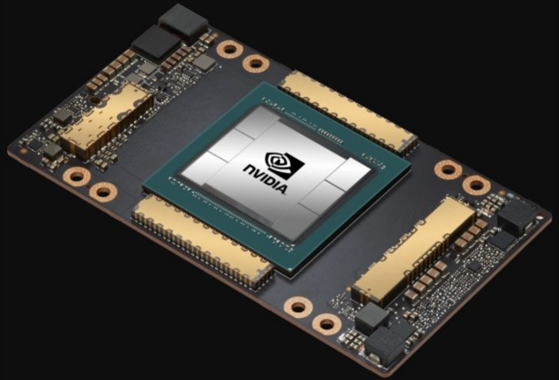
Multi-Core SM
2048 threads

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

CUDA PLATFORM: TARGETS EACH LEVEL OF THE HIERARCHY

The CUDA Platform Advances State Of The Art From Data Center To The GPU





System Scope

FABRIC MANAGEMENT
DATA CENTER OPERATIONS
DEPLOYMENT
MONITORING
COMPATIBILITY
SECURITY

Node Scope

GPU-DIRECT
NVLINK
LIBRARIES
UNIFIED MEMORY
ARM
MIG

Program Scope

CUDA C++
OPENACC
STANDARD LANGUAGES
SYNCHRONIZATION
PRECISION

SCOPE OF THIS SESSION

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

ACCELERATED PLATFORM

BUSINESS APPLICATIONS
- Customer Engagement
- Patient Diagnostics
- Fraud Detection
- Quality Assurance
- Industrial Automation
- Precision Marketing
- Molecular Simulations
- ++

SOFTWARE HUB
- HELM
- *Certified Containers*
- *Pre-trained Models*
- *SDKs*

VALIDATED SOLUTIONS

APPLICATION FRAMEWORKS
- SMART CITY — *Metropolis*
- CONVERSATIONAL AI — *Jarvis*
- AUTONOMOUS VEHICLES — *Drive*
- RECOMMENDATION SYSTEMS — *Merlin*
- HEALTHCARE — *Clara*
- ++ — …

DEVELOPER TOOLKITS
- ML & DATA ANALYTICS — RAPIDS, dmlc XGBoost
- AI TRAINING & INFERENCE — TensorFlow, *TensorRT*, PYTORCH, mxnet
- HIGH PERFORMANCE COMPUTING — *NVIDIA HPC SDK*
- RENDERING & VISUALIZATION — *IndeX*, *OptiX*

ACCELERATION LIBRARIES
- COMPUTE — *CUDA-X*
- NETWORKING, STORAGE & SECURITY — *DOCA*, *MAGNUM IO*

SERVERS & CLOUD
- DGX HGX — Purpose Built
- EGX — Mainstream & Edge
- CSP Instances

HARDWARE TECHNOLOGIES
- GPU
- NVSwitch
- BlueField DPU
- SMART NIC
- Mellanox Switch

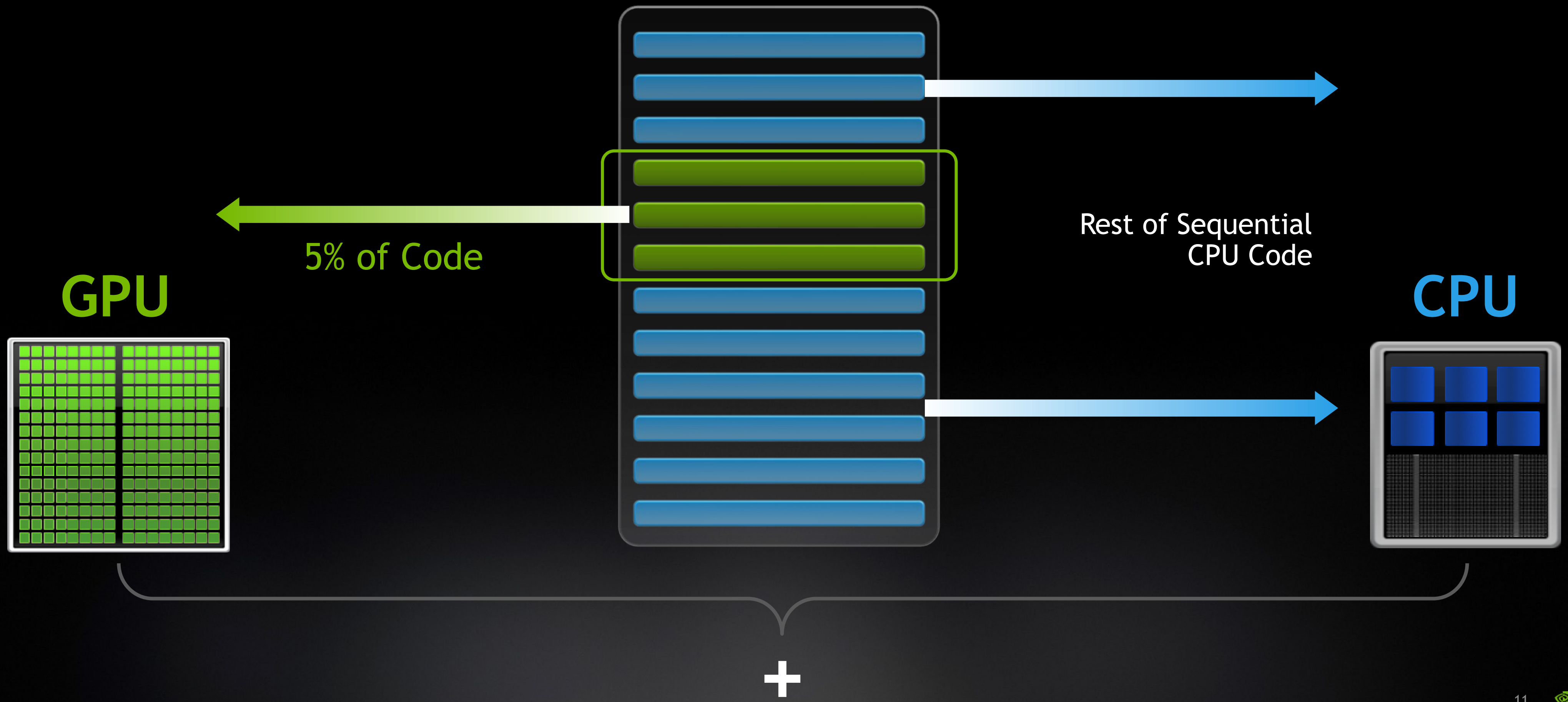
OPERATIONS
- TRITON INFERENCE SERVER
- FLEET COMMAND
- NVIDIA GPU Operator
- Red Hat
- vmware

MONITORING
- *DCGM*
- Grafana
- Prometheus
- *UFM*

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

NVIDIA

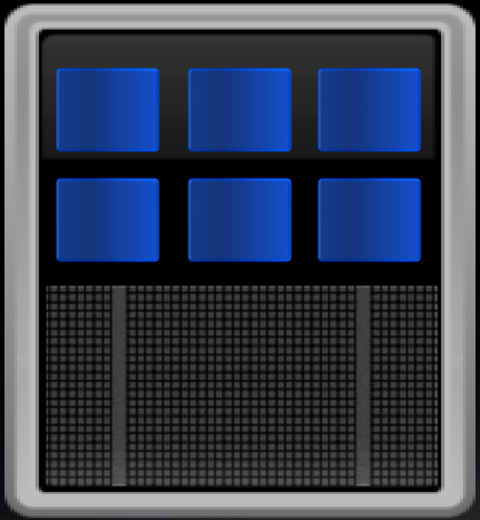
HOW GPU ACCELERATION WORKS



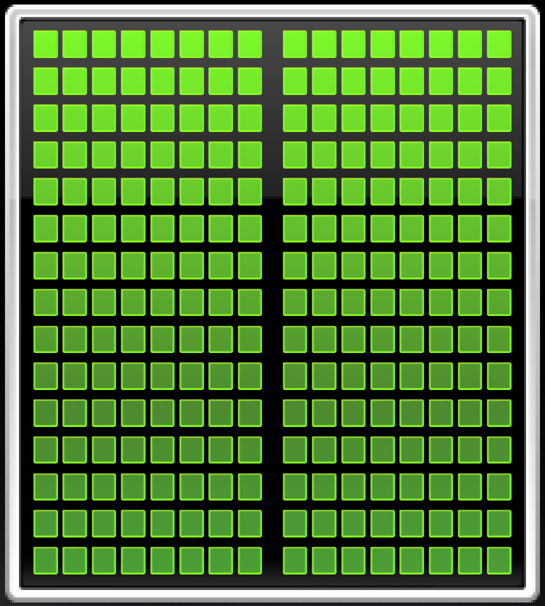
This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

ACCELERATED COMPUTING

CPU
Optimized for
Serial Tasks



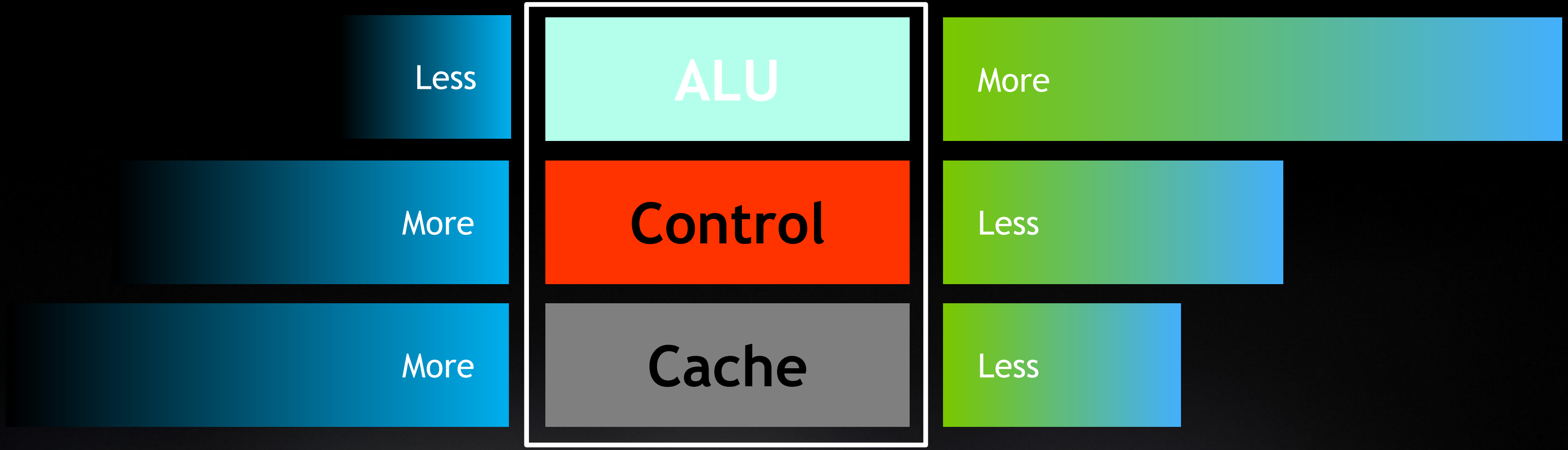
GPU Accelerator
Optimized for
Parallel Tasks



This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

SILICON BUDGET

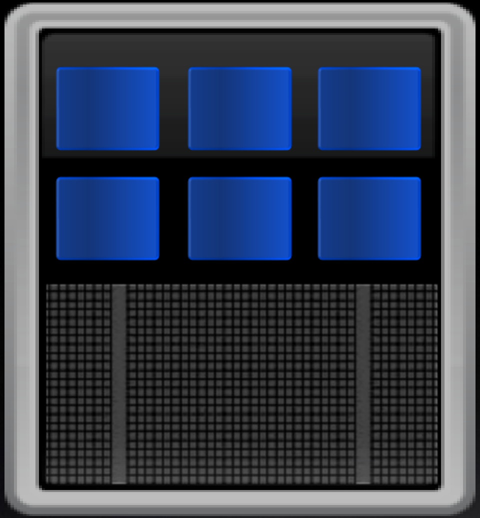
- The three components of any processor



This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

CPU IS A LATENCY REDUCING ARCHITECTURE

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/watt

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

GPU IS ALL ABOUT HIDING LATENCY

GPU Strengths

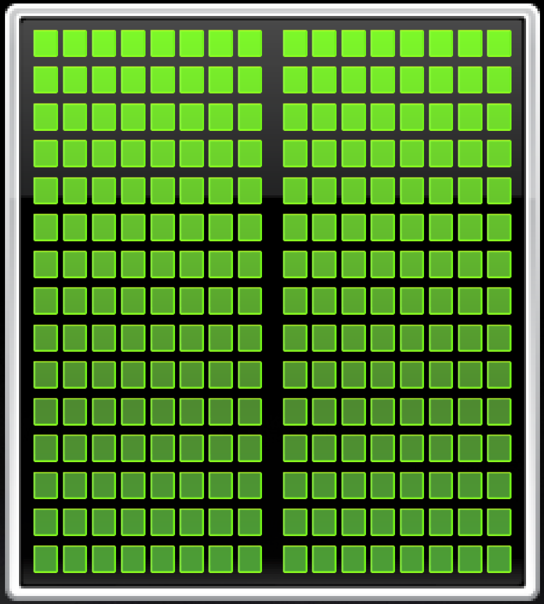
- High bandwidth main memory
- Significantly more compute resources
- Latency tolerant via parallelism
- High throughput
- High performance/watt

GPU Weaknesses

- Relatively low memory capacity
- Low per-thread performance

GPU Accelerator

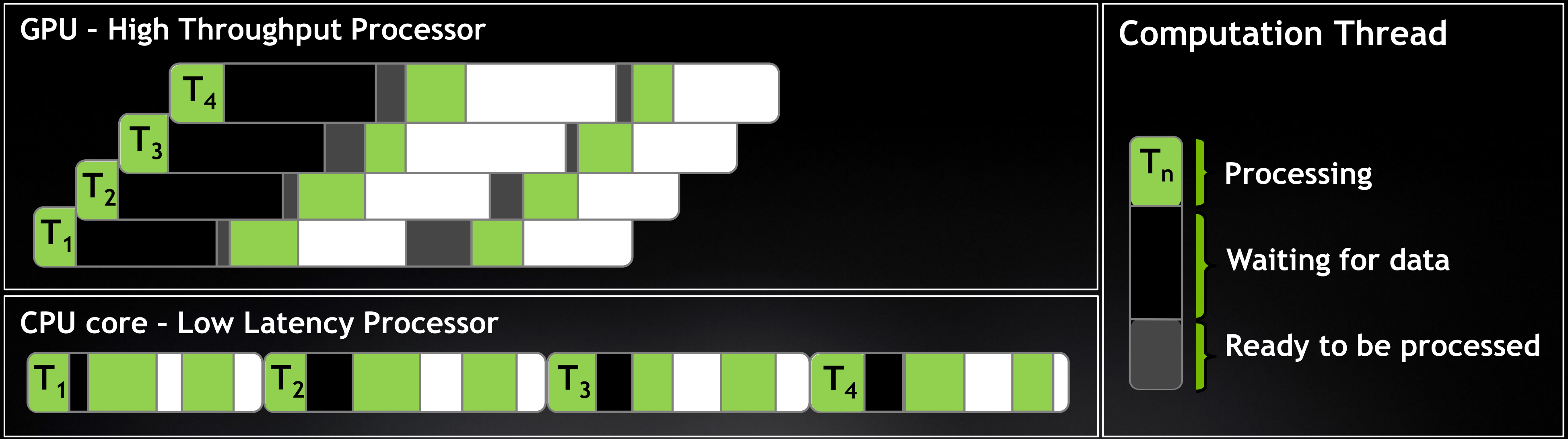
Optimized for Parallel Tasks



This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

LOW LATENCY VS HIGH THROUGHPUT

- CPU architecture must minimize latency within each thread
- GPU architecture hides latency with computation (data-parallelism, to 30k threads!)



This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

SPEED V. THROUGHPUT

Speed

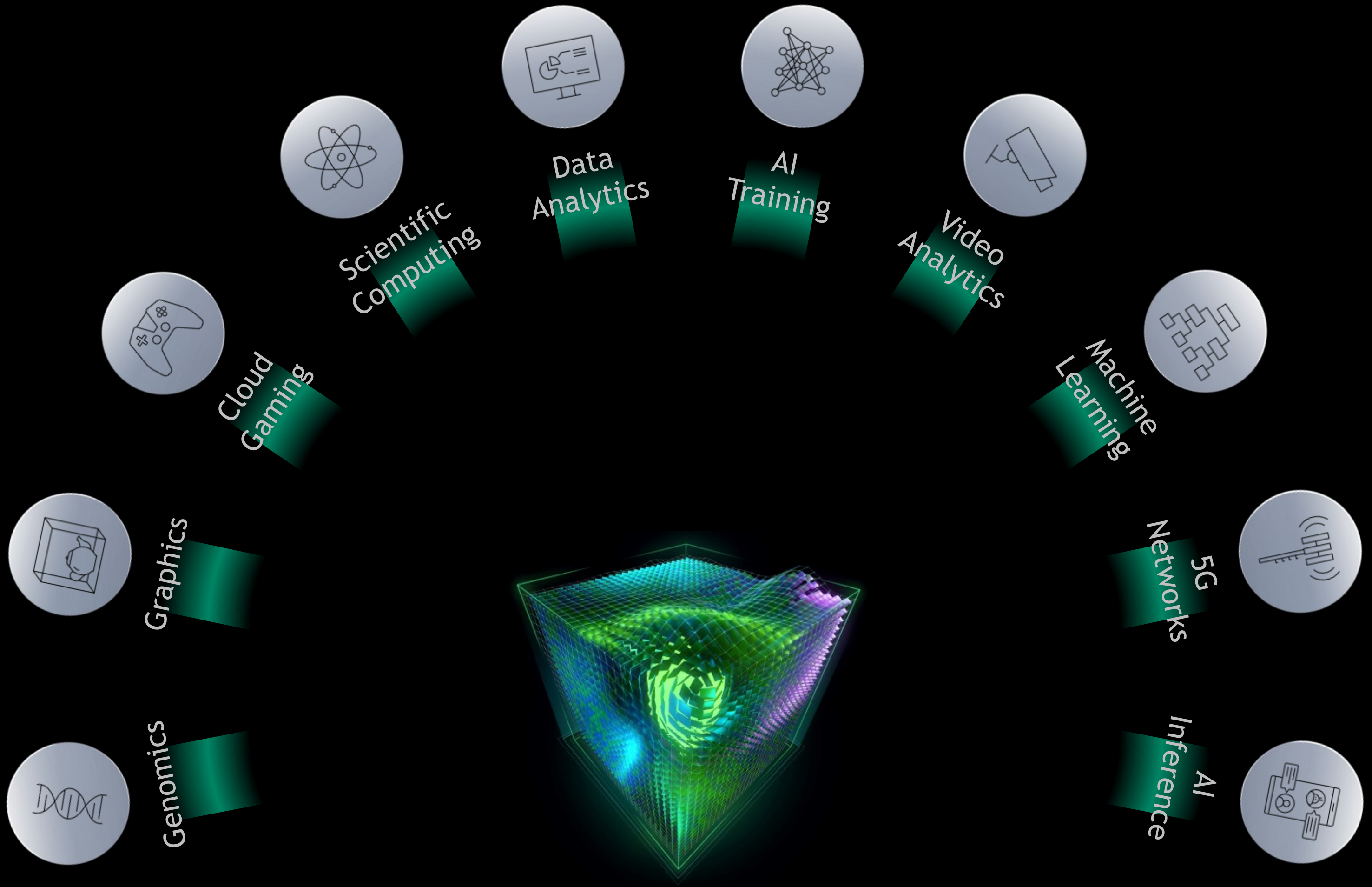
Throughput

Which is better depends on your needs…

*Images from Wikimedia Commons via Creative Commons

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

HUGE BREADTH OF PLATFORMS, SYSTEMS, LANGUAGES



This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

NVIDIA HPC SDK

Download at developer.nvidia.com/hpc-sdk

NVIDIA HPC SDK

DEVELOPMENT

Compilers
- nvcc
- nvc
- nvc++
- nvfortran

Math Libraries
- cuBLAS
- cuTENSOR
- cuSPARSE
- cuSOLVER
- cuFFT
- cuRAND

Communication Libraries
- Open MPI
- NVSHMEM
- NCCL

Programming Models
- Standard C++ & Fortran
- OpenACC & OpenMP
- CUDA

ANALYSIS

Profilers
- Nsight
  - Systems
  - Compute

Debugger
- cuda-gdb
  - Host
  - Device

DEPLOYMENT

Container
- HPC Container Maker / NVIDIA Container Runtime

Develop for the NVIDIA HPC Platform: GPU, CPU and Interconnect

HPC Libraries | GPU Accelerated C++ and Fortran | Directives | CUDA

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

N-WAYS TO GPU PROGRAMMING

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

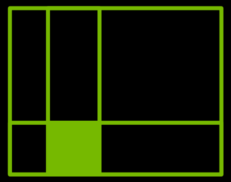
Incremental Performance Optimization with Directives

Maximize GPU Performance with CUDA C++/Fortran

GPU Accelerated Math Libraries

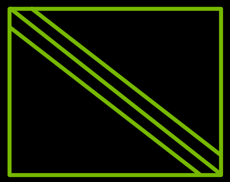
This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

GPU ACCELERATED MATH LIBRARIES



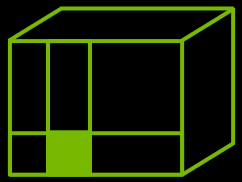
cuBLAS

BF16, TF32 and FP64 Tensor Cores



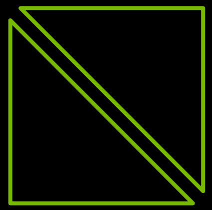
cuSPARSE

Increased memory BW, Shared Memory & L2



cuTENSOR

BF16, TF32 and FP64 Tensor Cores

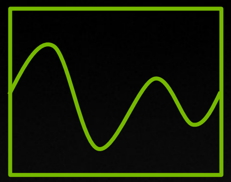


cuSOLVER

BF16, TF32 and FP64 Tensor Cores

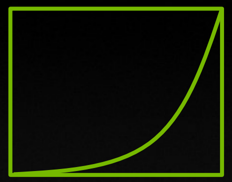
nvJPEG

Hardware Decoder



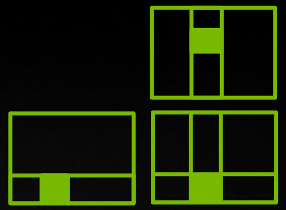
cuFFT

BF16, TF32 and FP64 Tensor Cores



CUDA Math API

Increased memory BW, Shared Memory & L2



CUTLASS

BF16 & TF32 Support

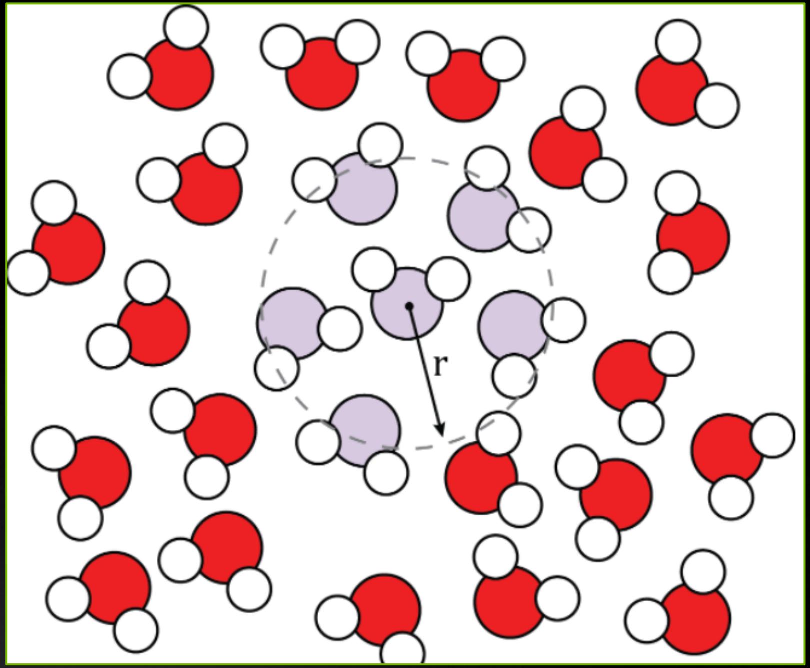
This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

APPLICATION

Molecular Simulation

RDF

The radial distribution function (RDF) denoted in equations by g(r) defines the probability of finding a particle at a distance r from another tagged particle.



This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

RDF

Pseudo Code - C

```
for (int frame=0;frame<nconf;frame++){

    for(int id1=0;id1<numatm;id1++)
    {
        for(int id2=0;id2<numatm;id2++)
        {
            dx=d_x[]-d_x[];
            dy=d_y[]-d_y[];
            dz=d_z[]-d_z[];
            r=sqrtf(dx*dx+dy*dy+dz*dz);

            if (r<cut) {
                ig2=(int)(r/del);
                d_g2[ig2] = d_g2[ig2] +1 ;
            }
        }
    }
}
```

- Across Frames
- Find Distance
- Reduction

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)

RDF

Pseudo Code - Fortran

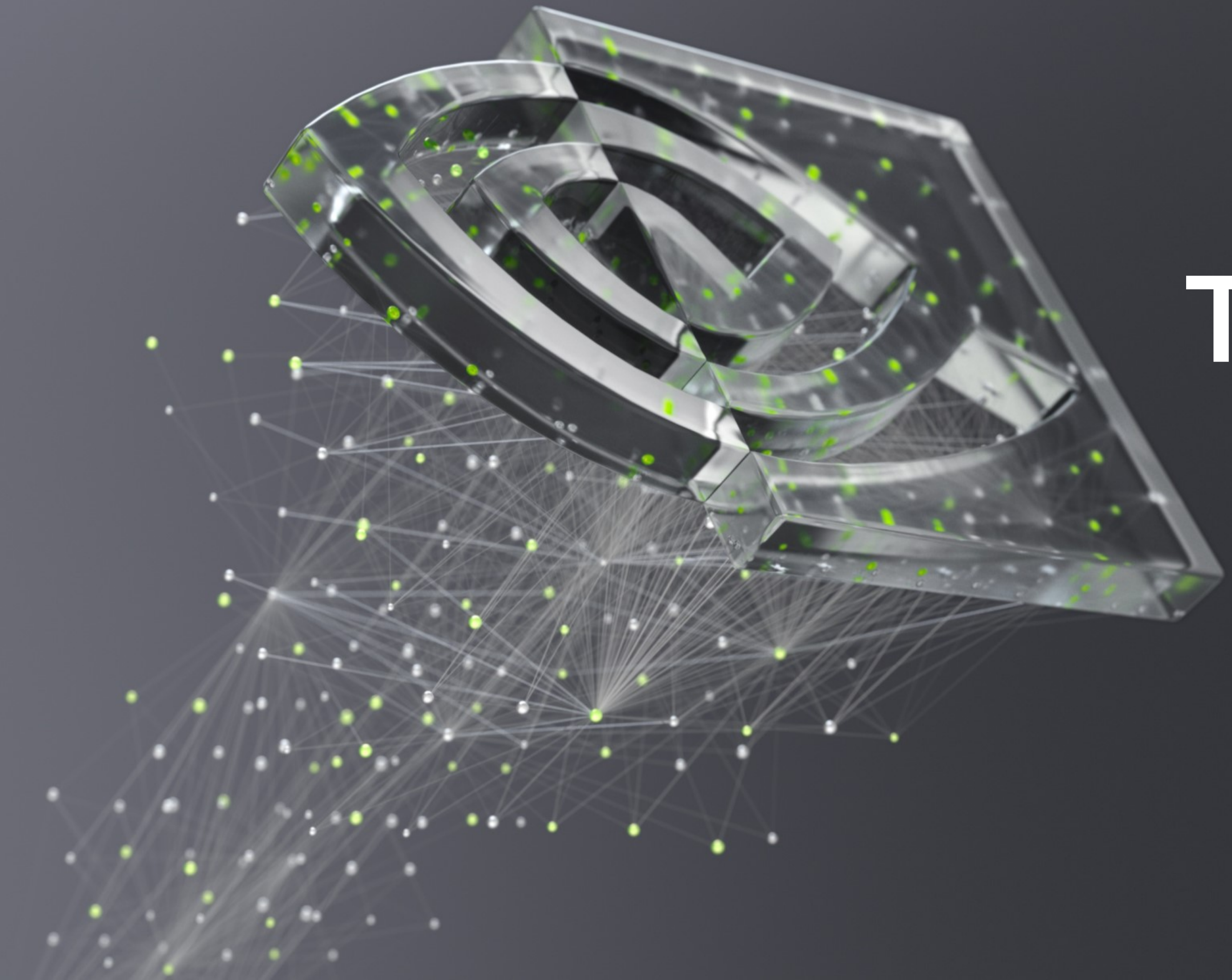
```
do iconf=1,nframes
   if (mod(iconf,1).eq.0) print*,iconf

   do i=1,natoms
       do j=1,natoms
                dx=x(iconf,i)-x(iconf,j)
                dy=y(iconf,i)-y(iconf,j)
                dz=z(iconf,i)-z(iconf,j)

                r=dsqrt(dx**2+dy**2+dz**2)
                if(r<cut)then
                        g(ind)=g(ind)+1.0d0
                endif
        enddo
   enddo
enddo
```

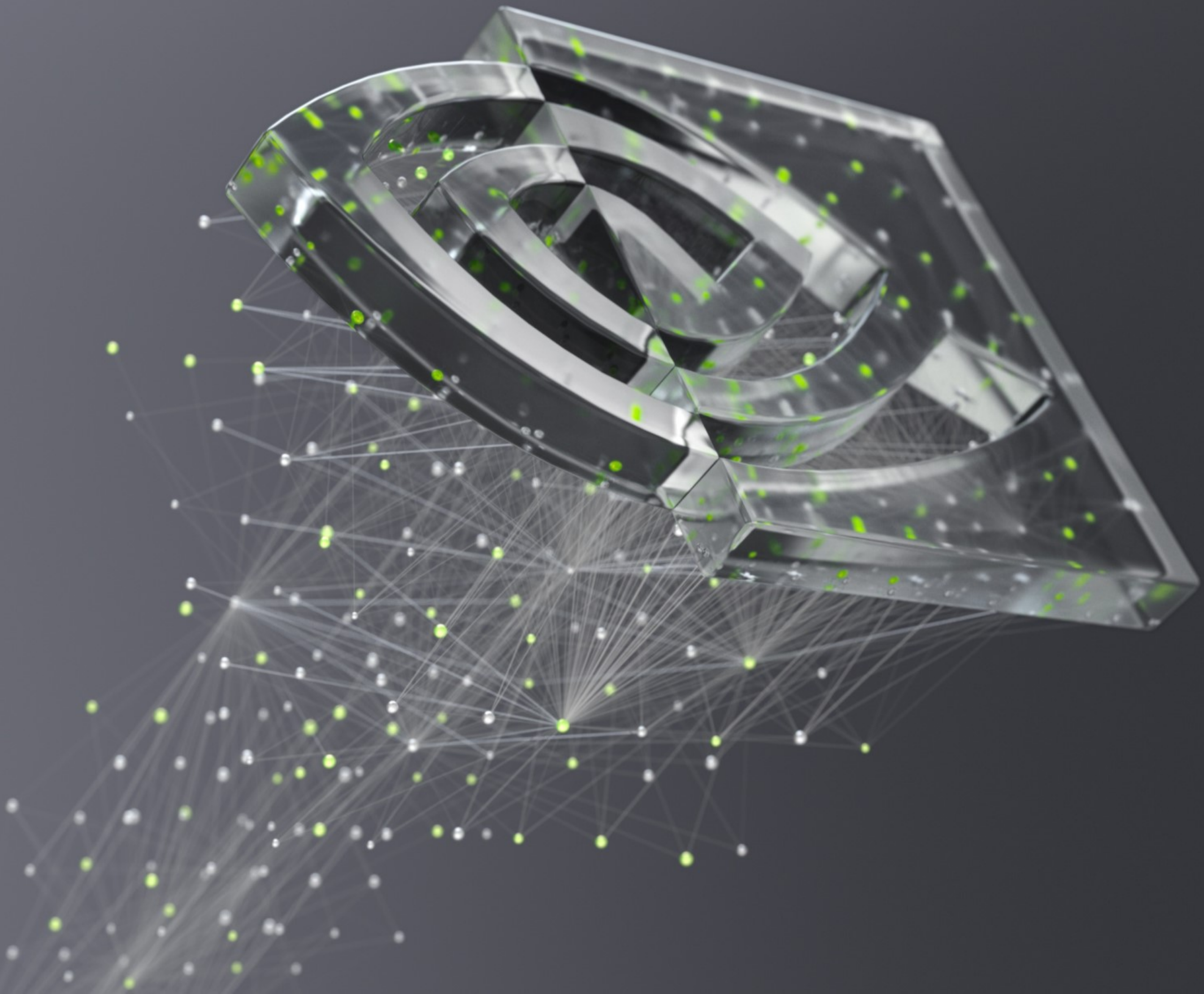
- Across Frames
- Find Distance
- Reduction

This material is released by NVIDIA Corporation under the Creative Commons Attribution 4.0 International (CC BY 4.0)



THANK YOU

NVIDIA



BACKUP

NVIDIA