PHYSICS AT EXTREME SCALE

Peter Messmer

NVIDIA Co-Design Lab @ ETH Zurich

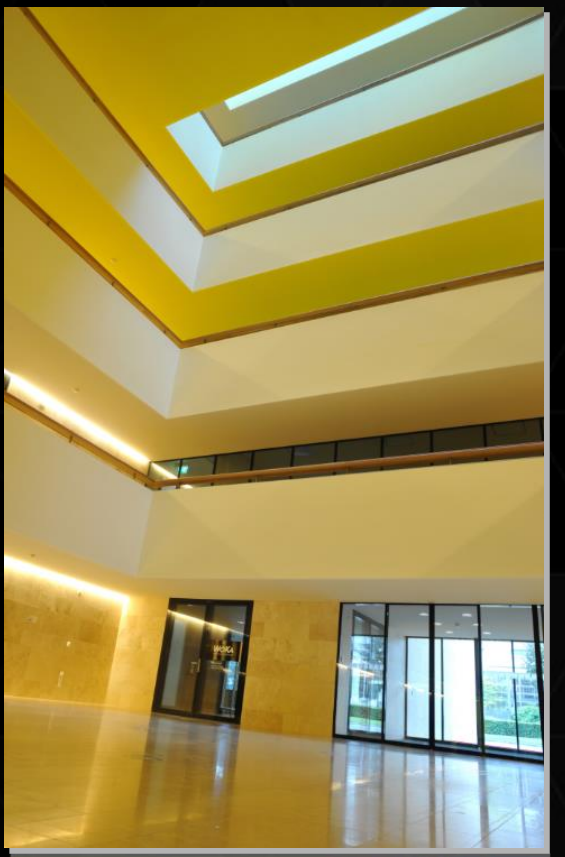
NVIDIA CO-DESIGN LAB AT ETH ZURICH

- Foster collaboration between scientific user community and NVIDIA
- Get input from real-world hybrid applications
- Support community in design decisions

- Opened on November 1, 2013
- Located on ETH Zurich campus (Science City & Center)

www.nvidiacodesignlab.ethz.ch

Similar labs at Julich, Cambridge, ..



2

FOCUS AREAS OF COE AT EXTREME SCALES

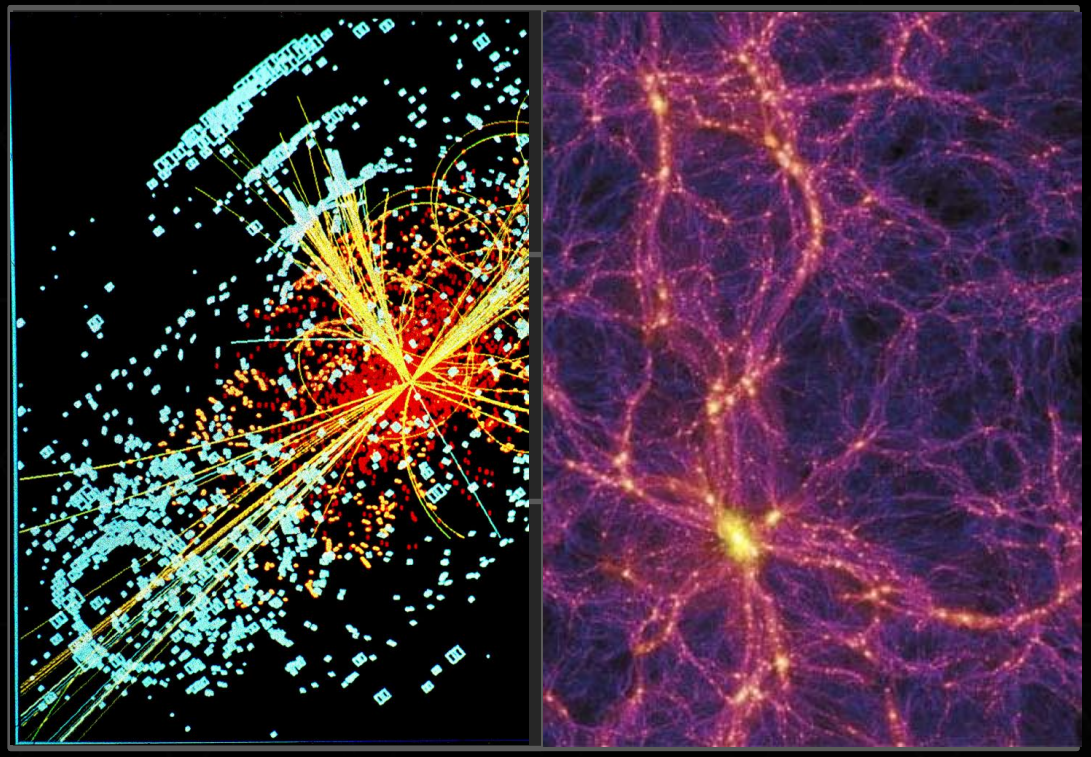
Particle Physics

Cosmology

Theory

Analytics

Data Acquisition



COE APPLICATION TYPES

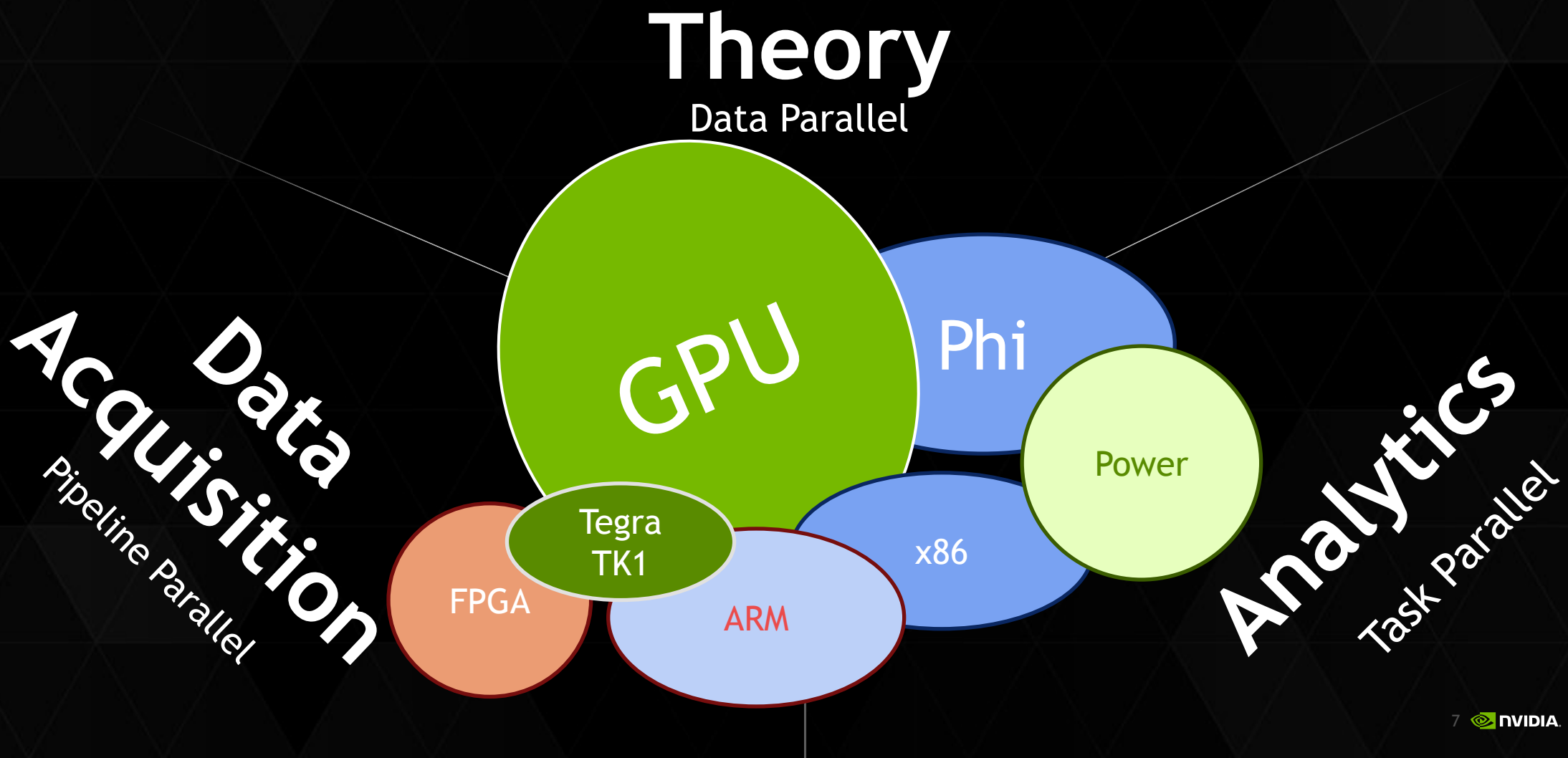
	Particle Physics	Cosmology
Theory	QCD	SPH
Analytics	WLCG	SKA/SDP
Data Acquisition	LHC Triggers	SKA Xrelator

RANGE OF EXTREME SCALE PARALLELISM

	Particle Physics	Cosmology	
Theory	QCD	SPH	Data Parallel
Analytics	WLCG	SKA/SDP	Task Parallel
Data Acquisition	LHC Triggers	SKA Xrelator	Pipeline Parallel

HARDWARE “COVERAGE” OF COE APP SPACE

HARDWARE "COVERAGE" OF COE APP SPACE

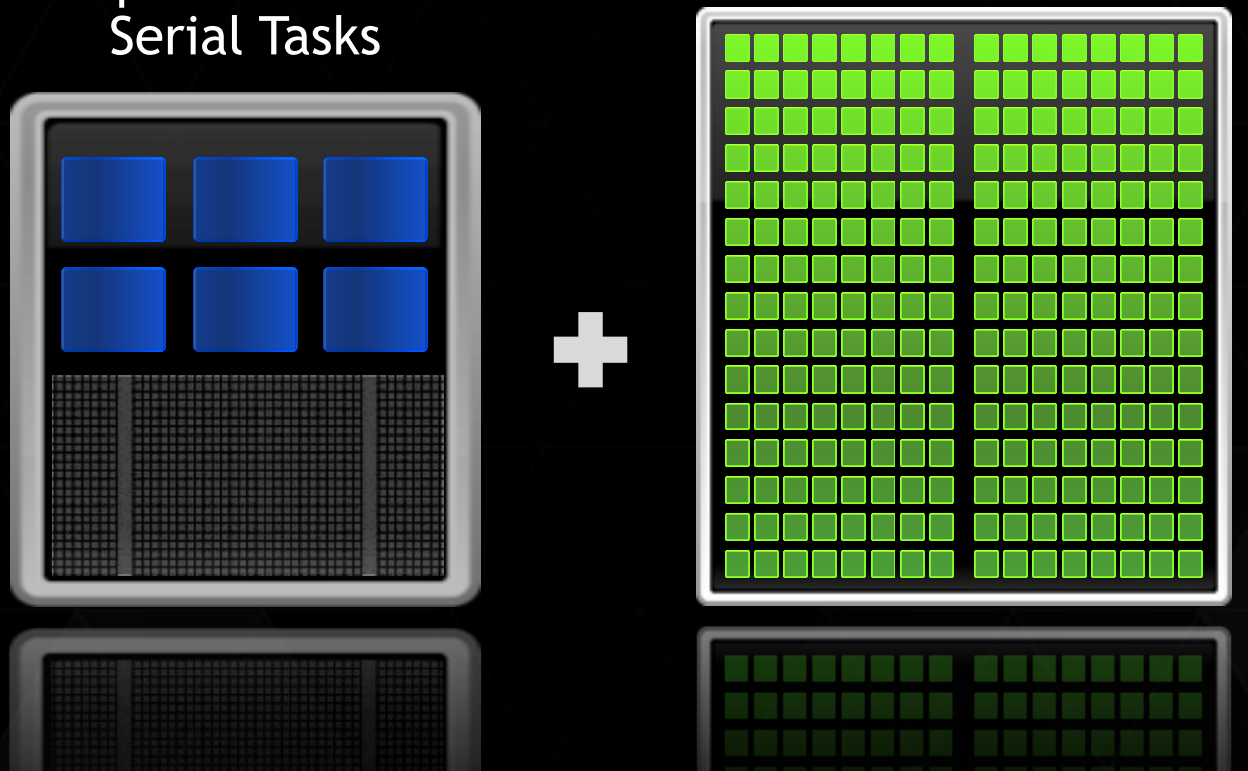


ACCELERATED COMPUTING

10x Performance & 5x Energy Efficiency for HPC

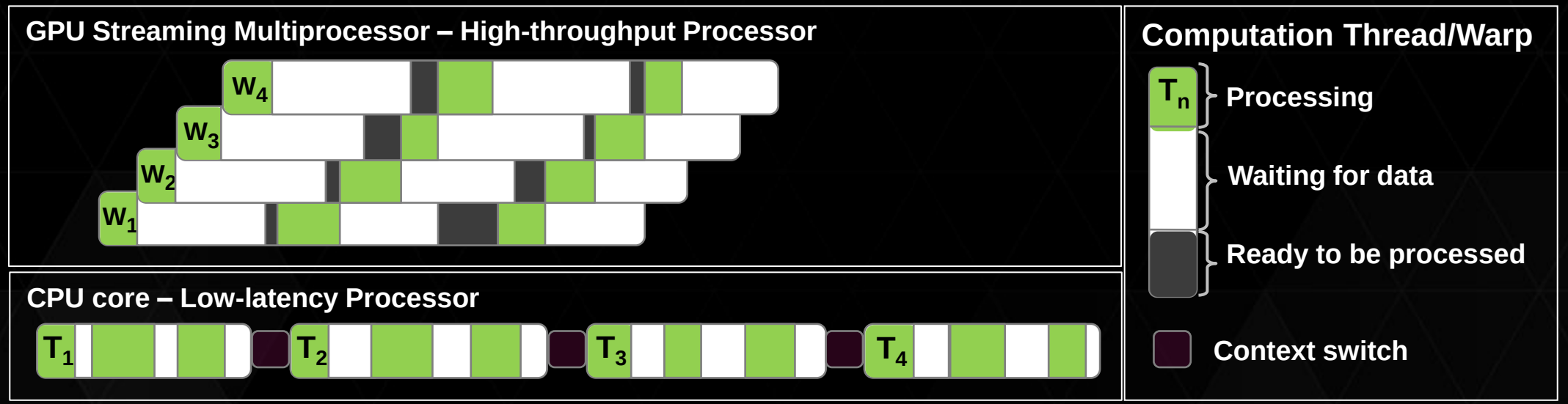
CPU
Optimized for
Serial Tasks

GPU Accelerator
Optimized for
Parallel Tasks



LOW LATENCY OR HIGH THROUGHPUT?

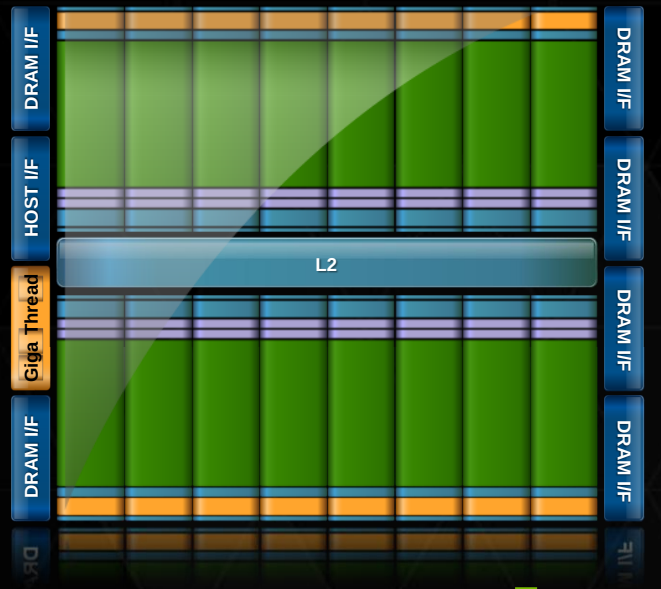
- CPU architecture must minimize latency within each thread
- GPU architecture hides latency with computation from other (warps of) threads



9

GPU ARCHITECTURE: TWO MAIN COMPONENTS

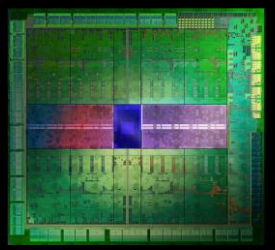
- Global memory
  - Analogous to RAM in a CPU server
  - Accessible by both GPU and CPU
  - Currently up to 12 GB per GPU
  - Bandwidth currently up to ~288 GB/s (Tesla products)
  - ECC on/off (Quadro and Tesla products)
- Streaming Multiprocessors (SMs)
  - Perform the actual computations
  - Each SM has its own:
    - Control units, registers, execution pipelines, caches



10

KEPLER GENERATION OF GPUS

Tesla K10

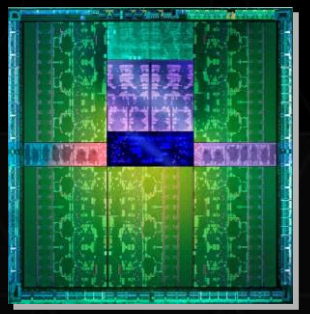


Dual GK104 GPUs

3x Single Precision

Video, Signal, Life Sciences, Seismic

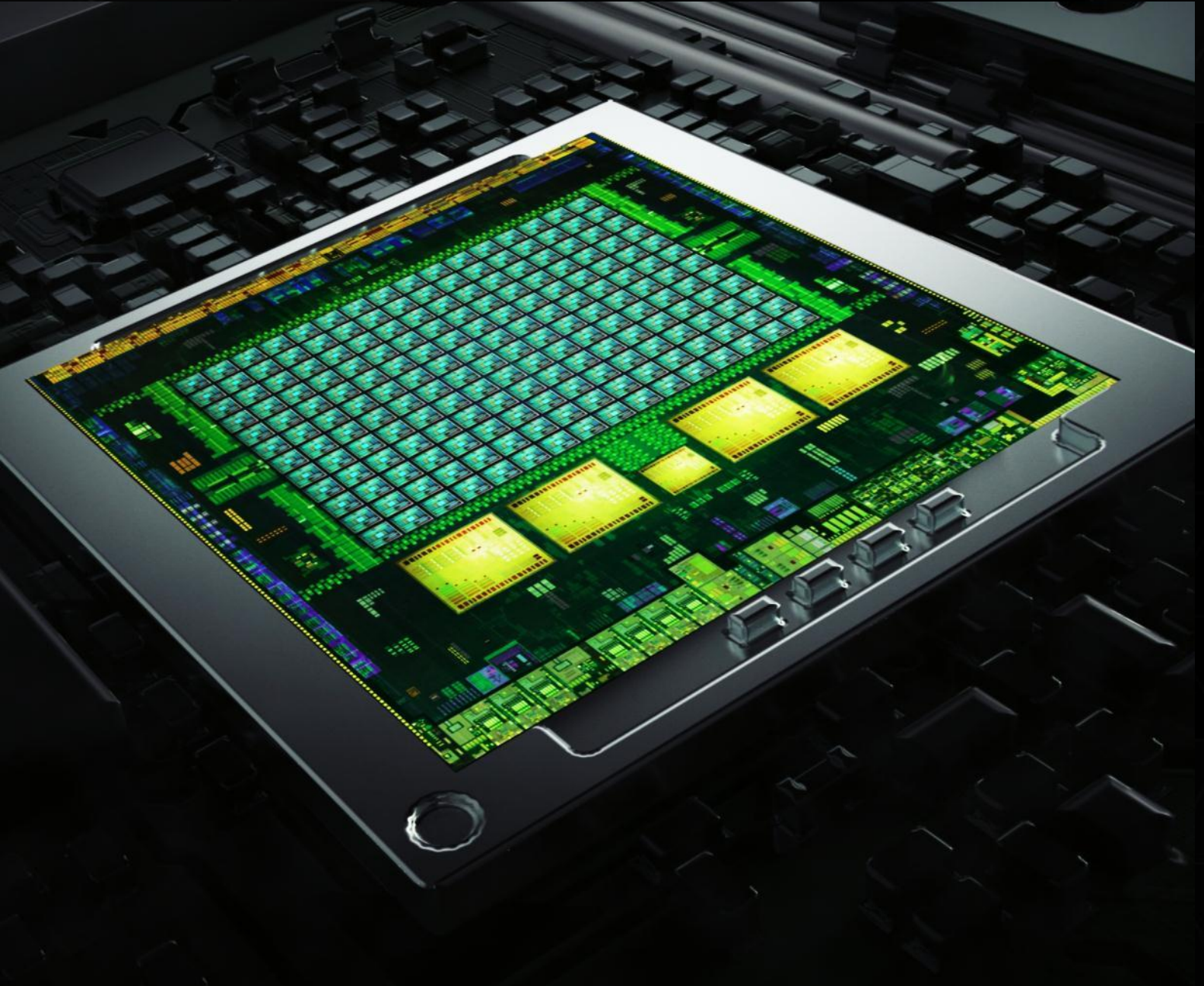
Tesla K20, K20X, K40



GK110 GPU

3x Double Precision

CFD, FEA, Finance, Physics, etc.



TEGRA K1

IMPOSSIBLY ADVANCED

NVIDIA Kepler Architecture

4-Plus-1 Quad-Core A15

192 NVIDIA CUDA Cores

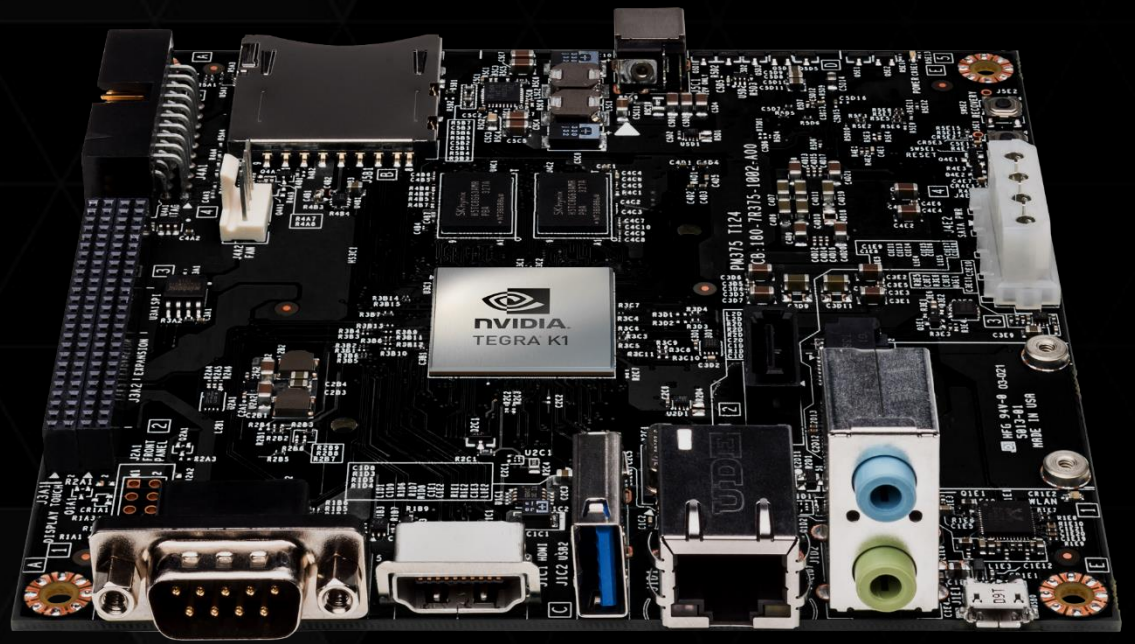
Compute Capability 3.2

326 GFLOPS

5 Watts

JETSON TK1

THE WORLD'S 1st EMBEDDED SUPERCOMPUTER

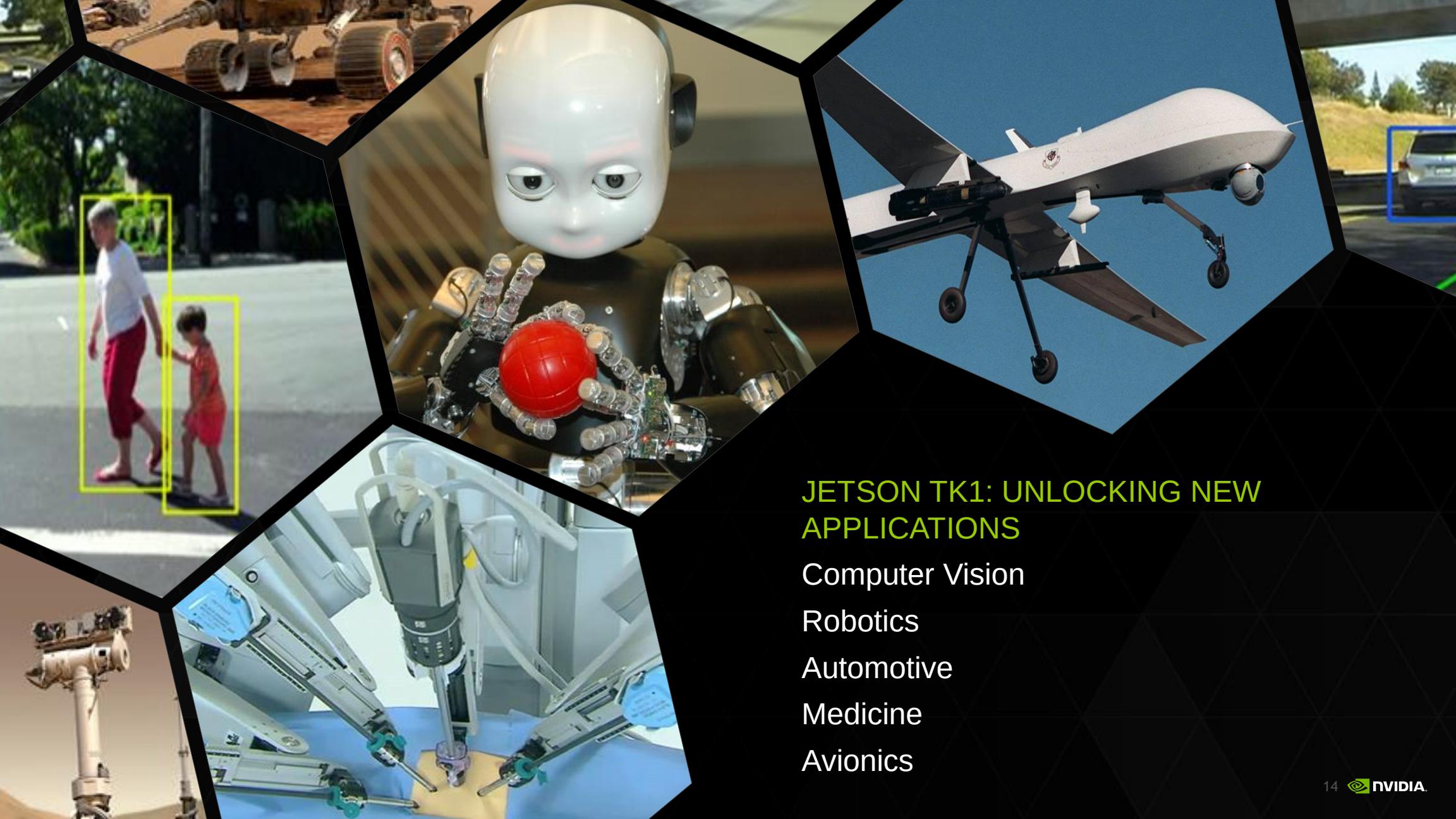


Development Platform for Embedded
Computer Vision, Robotics, Medical

Tegra K1 SoC

CUDA Enabled

$192



JETSON TK1: UNLOCKING NEW APPLICATIONS

Computer Vision

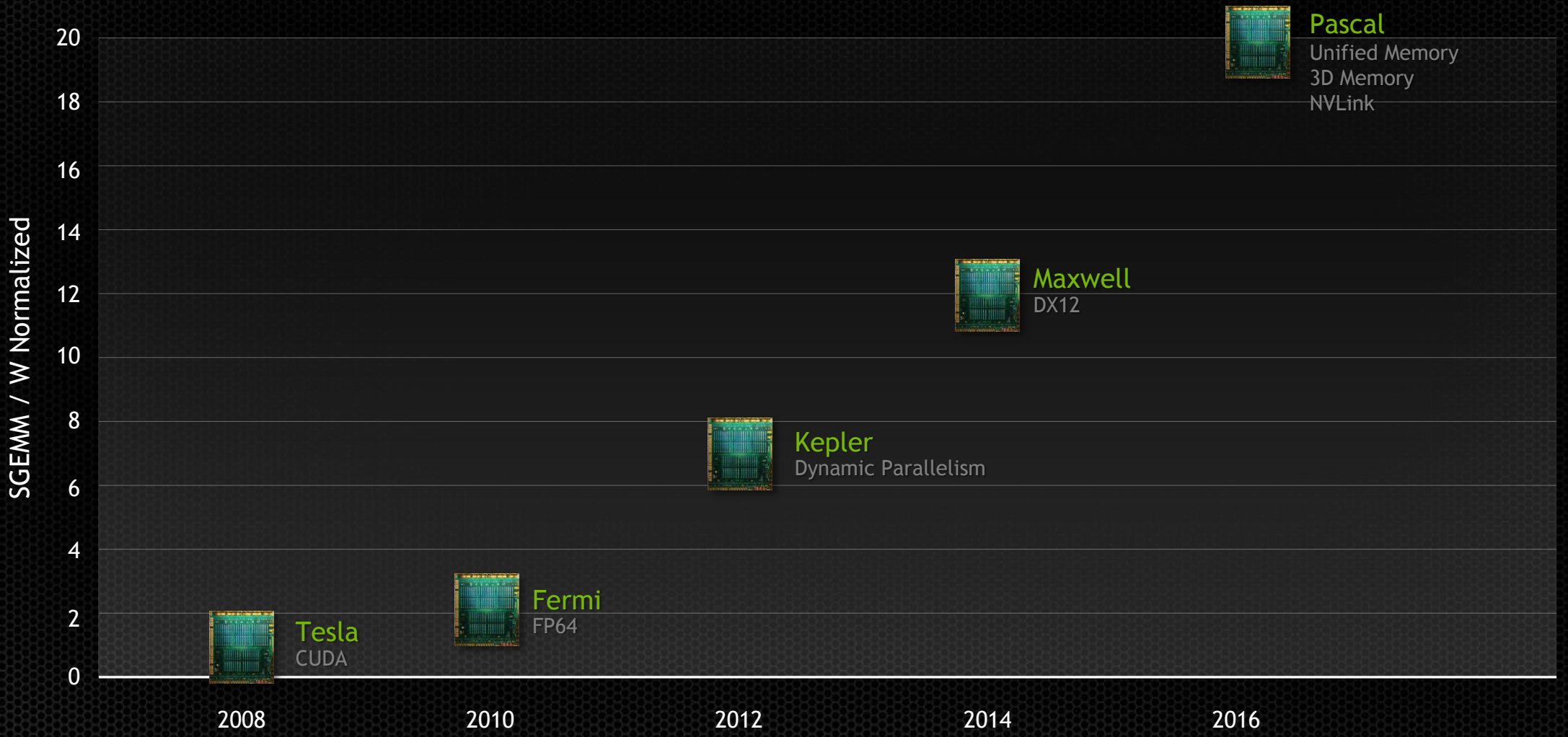
Robotics

Automotive

Medicine

Avionics

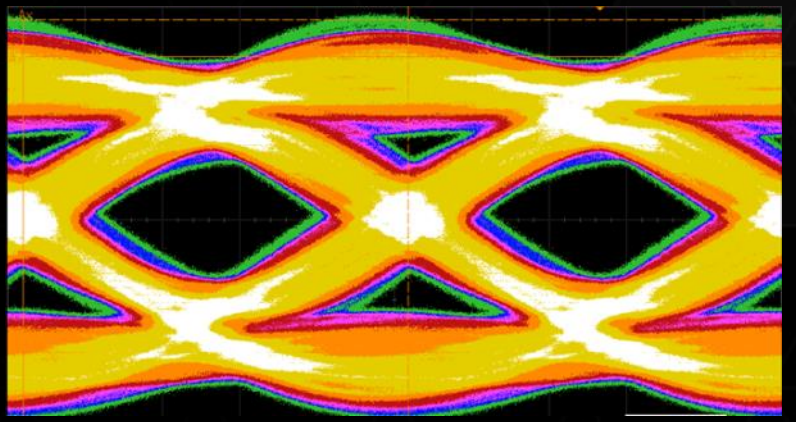
STRONG CUDA GPU ROADMAP



INTRODUCING NVLINK AND STACKED MEMORY

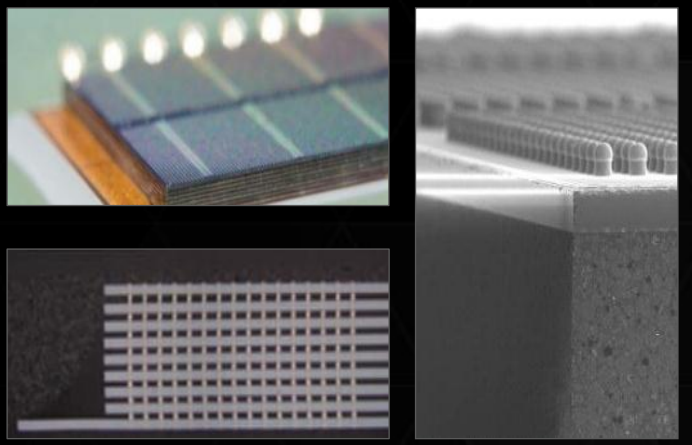
NVLINK

- GPU high speed interconnect
- 80-200 GB/s
- Planned support for POWER CPUs

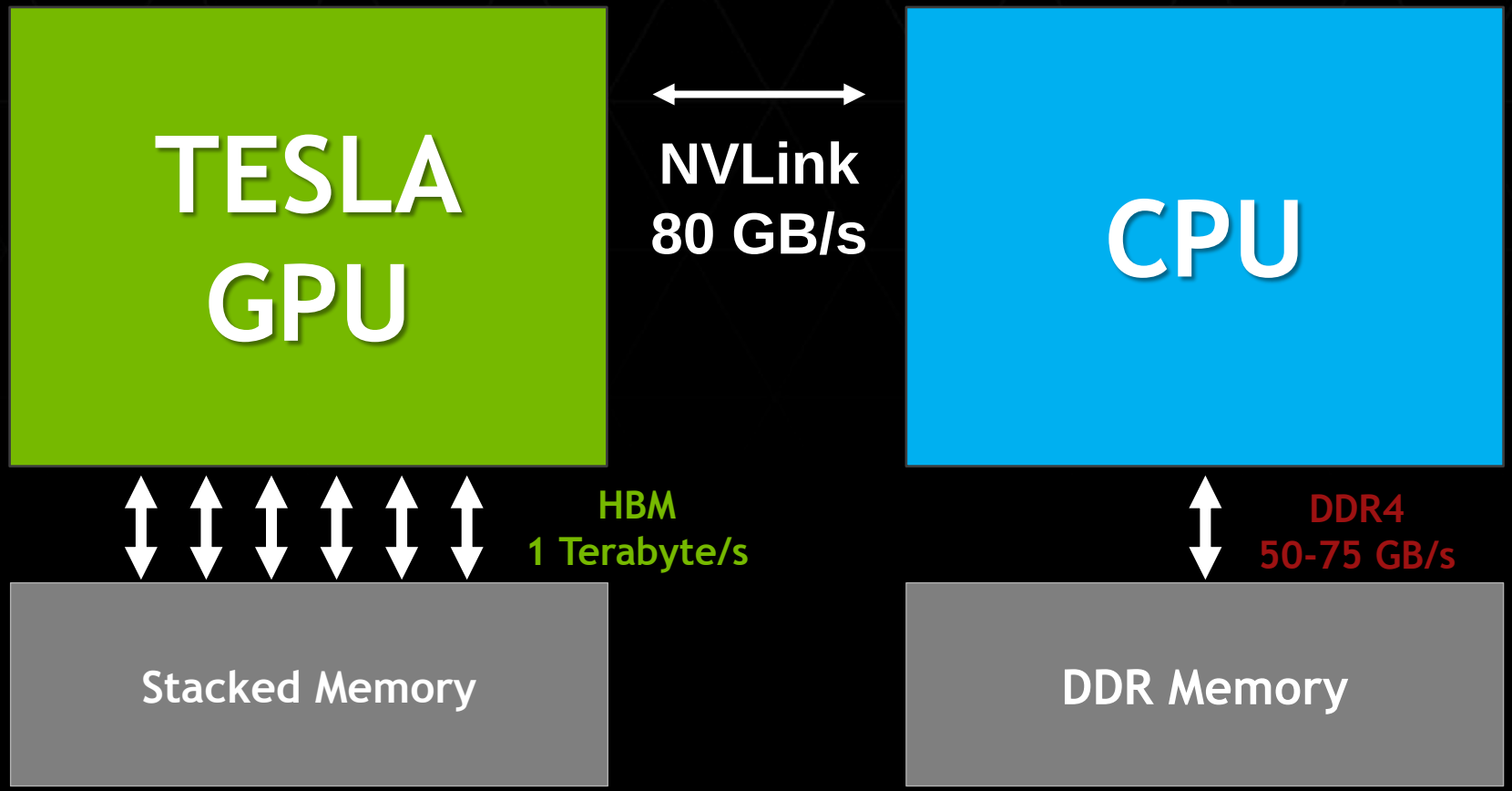


Stacked Memory

- 4x Higher Bandwidth (~1 TB/s)
- 3x Larger Capacity
- 4x More Energy Efficient per bit

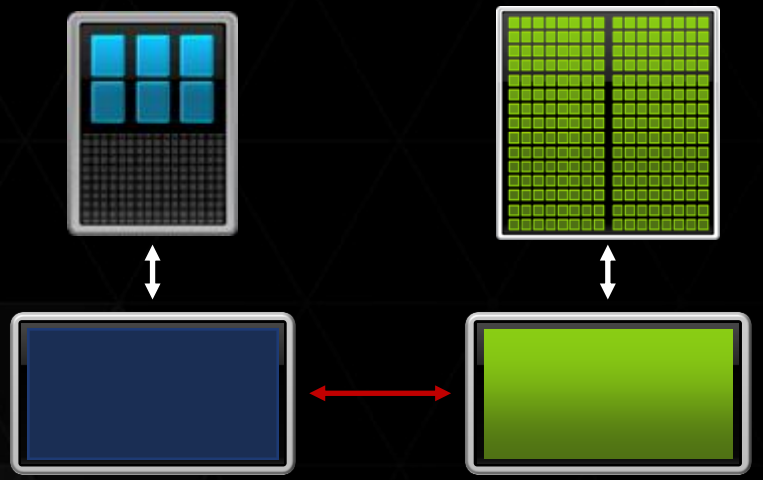


NVLINK UNIFIES MEMORY SPACES

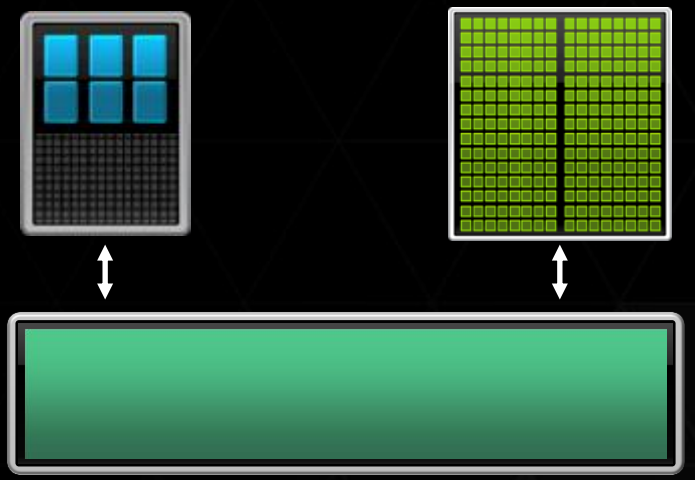


UNIFIED MEMORY
DRAMATICALLY LOWER DEVELOPER EFFORT

Developer View Today



Developer View With Unified Memory



18

UNIFIED MEMORY

Call Sort on CPU

```
void sortfile(FILE *in, FILE *out, int
N)
{
    char *data = (char *)malloc(N);
    fread(data, 1, N, in);

    sort(data, N);

    fwrite(data, 1, N, out);
    free(data);
}
```

Call Sort on Kepler

```
void sortfile(FILE *in, FILE *out, int N)
{
    char *data = (char *)cudaMallocManaged(N);
    fread(data, 1, N, in);

    parallel_sort<<< ... >>>(data, N);

    fwrite(data, 1, N, out);
    cudaFree(gpu_data);
}
```

Memory Management Becomes Performance Optimization

Call Sort on Pascal

```
void sortfile(FILE *in, FILE *out, int
N)
{
    char *data = (char *)malloc (N);
    fread(data, 1, N, in);

    parallel_sort<<< ... >>>(data, N);

    fwrite(data, 1, N, out);
    free(gpu_data);
}
```

No need for opt-in allocator

THE RIGHT TOOL FOR THE TASK

python

MPI 3.0

OpenACC
Directives for Accelerators

OpenCL

THE C PROGRAMMING LANGUAGE

NVIDIA CUDA

OpenMP

VHDL

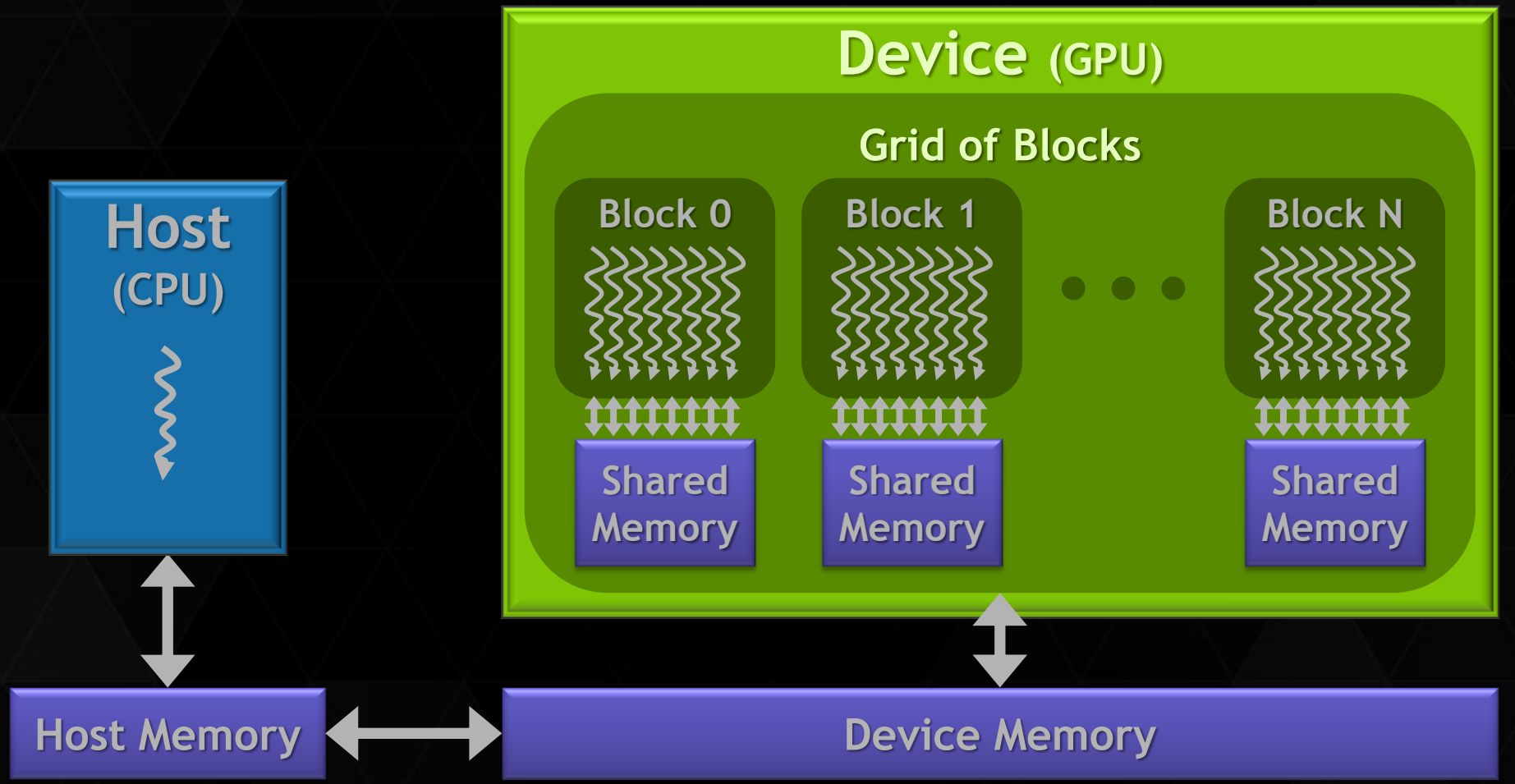
Fortran

Domain-Specific Languages

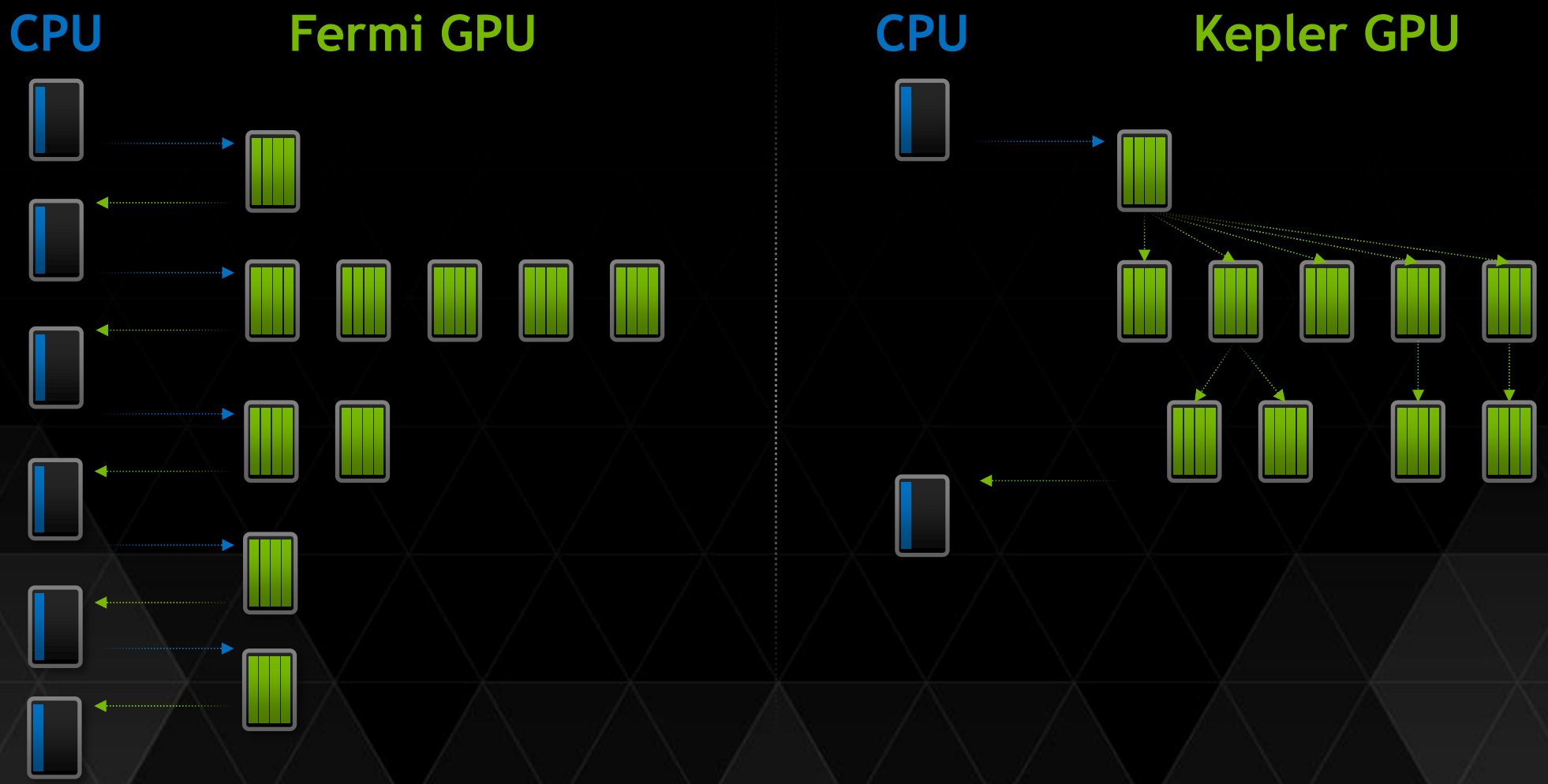
daiw.de

CUDA PROGRAMMING MODEL FOR GPUS

Parallel, hierarchical, heterogeneous

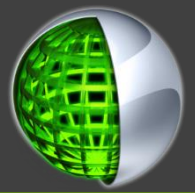


DYNAMIC PARALLELISM



CUDA DEVELOPMENT PLATFORM

Productive tools and higher-level programming approaches

Programming Approaches	Libraries	OpenACC Directives	Programming Languages
	"Drop-in" acceleration	Maximum portability	Maximum control
Development Environment	 Parallel Nsight IDE Linux, Mac and Windows GPU Debugging and Profiling		CUDA-GDB debugger NVIDIA Visual Profiler
Open Compiler Tool Chain	 Enables compiling new languages to CUDA platform, and CUDA languages to other architectures		

C++11 IN CUDA 6.5

- Experimental release in CUDA 6.5

```
nvcc -std=c++11 my_cpp11_code.cu
```

- Support for all C++11 features offered by host compiler in host code

- Currently no support for lambdas passed from host to device

THRUST: STL-LIKE CUDA TEMPLATE LIBRARY

- GPU(device) and CPU(host) vector class

```
thrust::host_vector<float> H(10, 1.f);

thrust::device_vector<float> D = H;
```

- Iterators

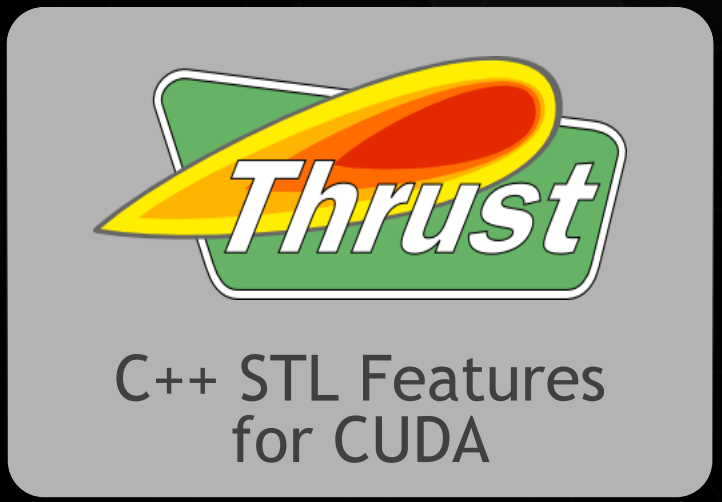
```
thrust::fill(D.begin(), D.begin()+5, 42.f);

float* raw_ptr = thrust::raw_pointer_cast(D);
```

- Algorithms
  - Sort, reduce, transformation, scan, ..

```
thrust::transform(D1.begin(), D1.end(), D2.begin(), D2.end(),

 thrust::plus<float>());    // D2 = D1 + D2
```



GPU-ACCELERATED LIBRARIES

"Drop-in" acceleration

NVIDIA cuBLAS

NVIDIA cuSPARSE

NVIDIA NPP

NVIDIA cuFFT

MAGMA

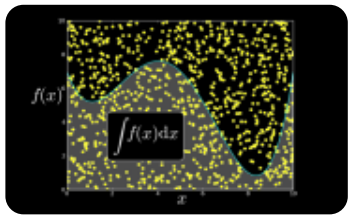
Matrix Algebra on GPU and Multicore

CULA | tools

GPU Accelerated Linear Algebra

GPU VSIPL

Vector Signal Image Processing



NVIDIA cuRAND

ROGUE WAVE SOFTWARE

IMSL Library

CenterSpace NMath

ArrayFire

Building-block Algorithms

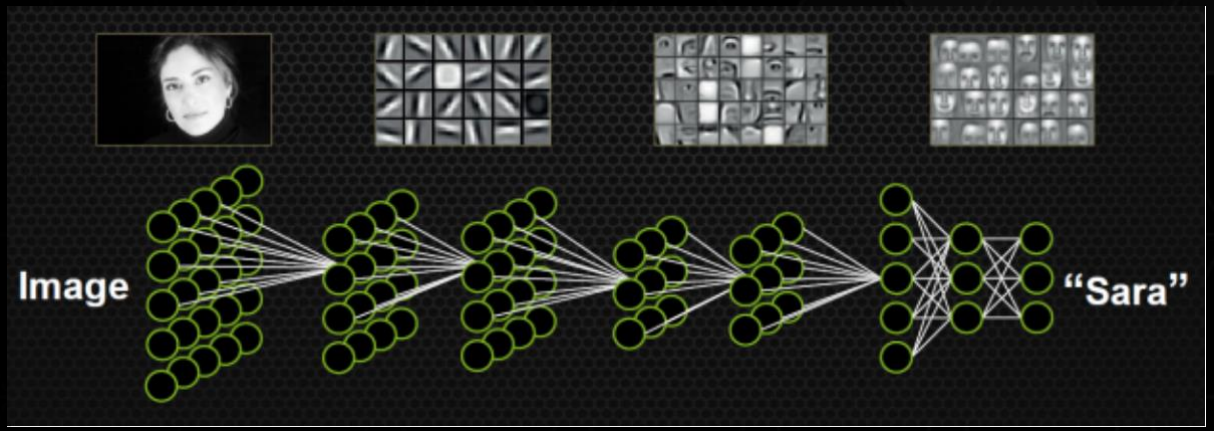
Thrust

C++ Templated Parallel Algorithms

CUDNN

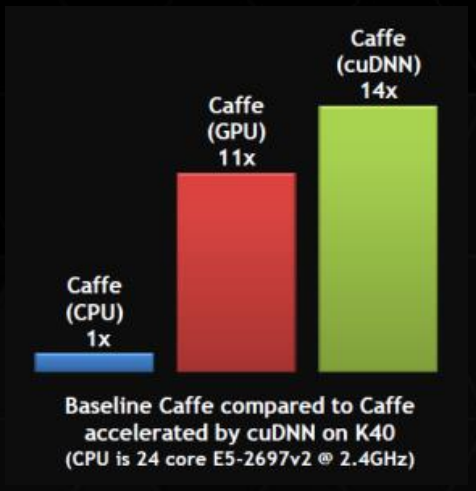
- Deep Neural Network Library
- Pre-packaged kernels for
  - convolution, pooling, softmax
  - activations
  - ..

developer.nvidia.com/cuDNN

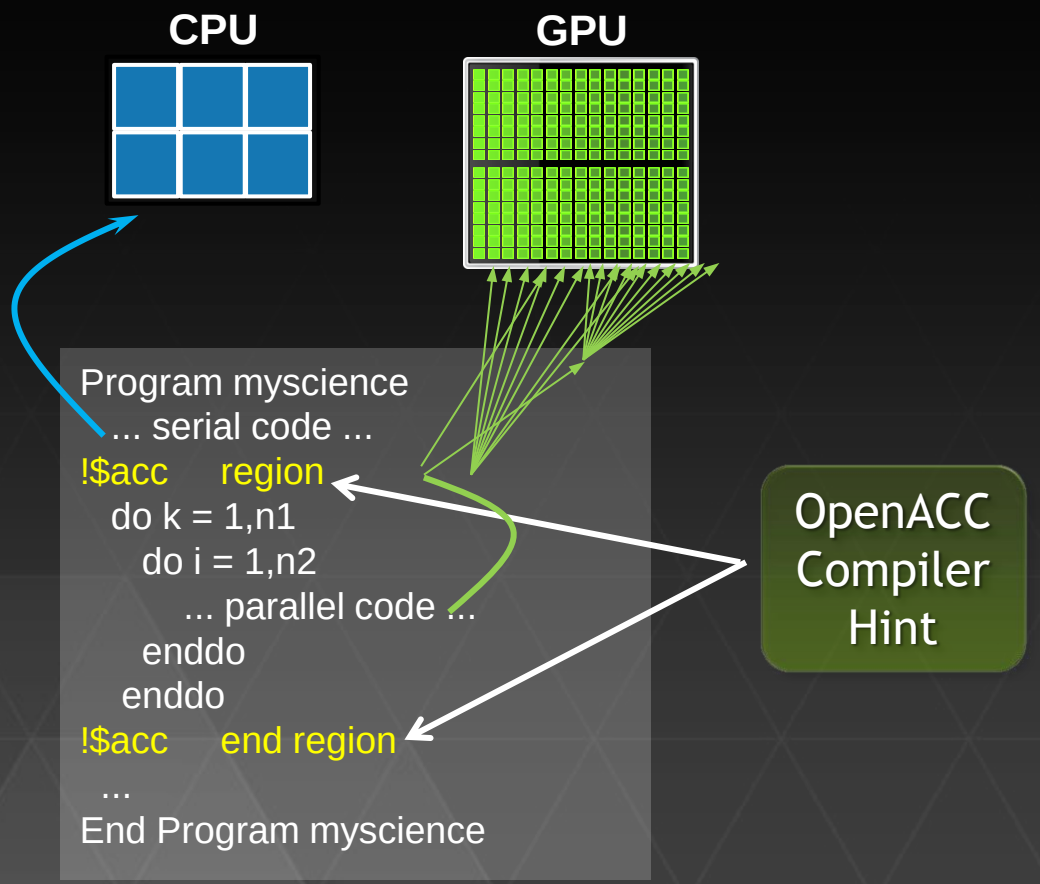



```
/* Set decriptors */
cudnnSetTensor4dDescriptor(InputDesc, CUDNN_TENSOR_NCHW, 128, 96, 221
cudnnSetFilterDescriptor(FilterDesc, 256, 96, 7, 7 );
cudnnSetConvolutionDescriptor(convDesc, InputDesc, FilterDesc,
                              pad_x, pad_y, 2, 2, 1, 1, CUDNN_CONVOLU

/* query output layout */
cudnnGetOutputTensor4dDim(convDesc, CUDNN_CONVOLUTION_FWD, &n_out, &c
                          &h_out, &w_out);
```



OPENACC DIRECTIVES



Easy, Open, Powerful

- Simple Compiler hints
- Works on multicore CPUs & many core GPUs
- Compiler Parallelizes code

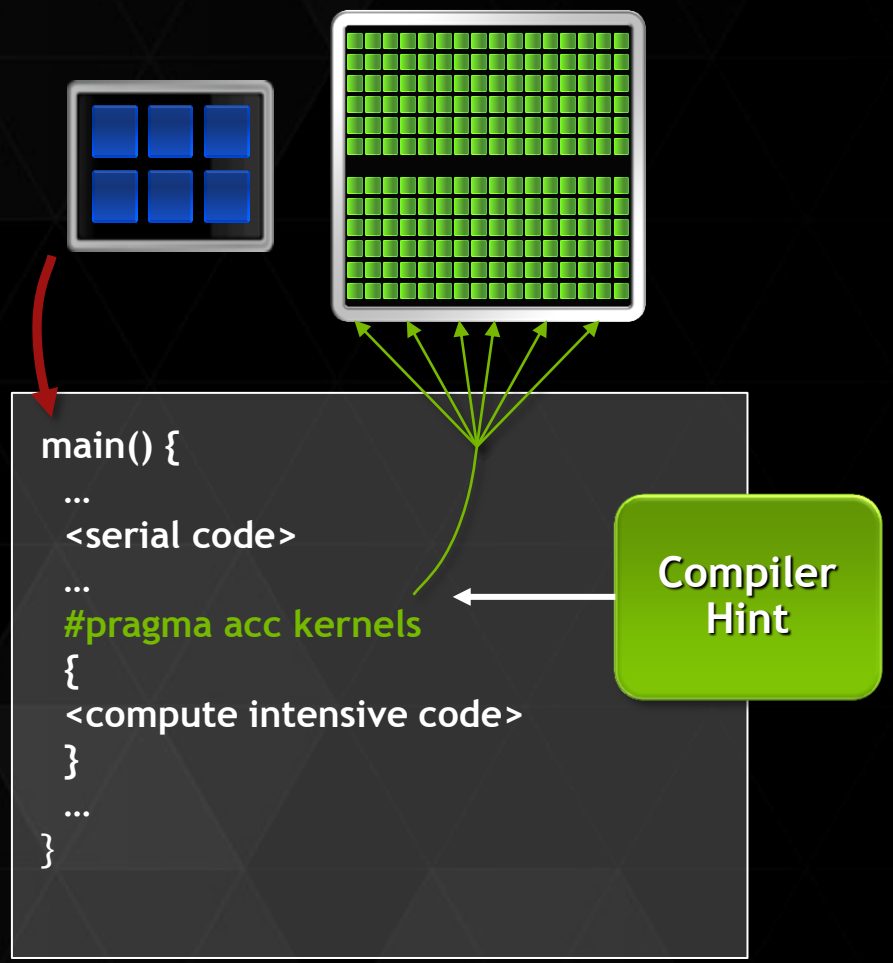
http://www.openacc.org

NVIDIA

PGI

CRAY THE SUPERCOMPUTER COMPANY

OPENACC: OPEN, SIMPLE, PORTABLE



- Open Standard
- Easy, Compiler-Driven Approach
- Portable on GPUs and Xeon Phi

CAM-SE Climate
6x Faster on GPU
Top Kernel: 50% of Runtime

29

STANDARDIZATION EFFORTS

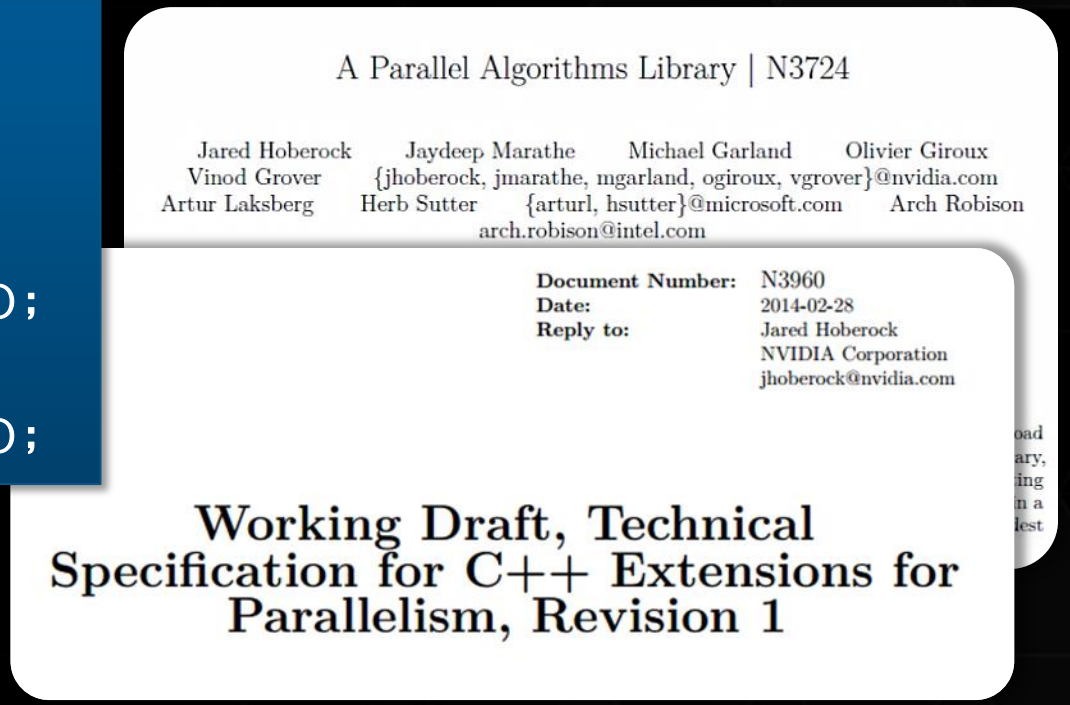
A standard C++ parallel library

```
std::vector<int> vec = ...

// previous standard sequential loop
std::for_each(vec.begin(), vec.end(), f);

// explicitly sequential loop
std::for_each(std::seq, vec.begin(), vec.end(), f);

// permitting parallel execution
std::for_each(std::par, vec.begin(), vec.end(), f);
```


A Parallel Algorithms Library | N3724

Jared Hoberock Jaydeep Marathe Michael Garland Olivier Giroux
Vinod Grover {jhoberock, jmarathe, mgarland, ogiroux, vgrover}@nvidia.com
Artur Laksberg Herb Sutter {arturl, hsutter}@microsoft.com Arch Robison
arch.robison@intel.com

Document Number: N3960
Date: 2014-02-28
Reply to: Jared Hoberock
NVIDIA Corporation
jhoberock@nvidia.com

Working Draft, Technical Specification for C++ Extensions for Parallelism, Revision 1

- Complete set of parallel primitives: for_each, sort, reduce, scan, etc.
- ISO C++ committee voted unanimously to accept as official technical specification working draft

N3960 Technical Specification Working Draft:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n3960.pdf
Prototype:
https://github.com/n3554/n3554

Linux GCC Compiler to Support GPU Accelerators

Open Source

GCC Efforts by Samsung & Mentor Graphics

Pervasive Impact

Free to all Linux users

Mainstream

Most Widely Used HPC Compiler

> *"Incorporating OpenACC into GCC is an excellent example of open source and open standards working together to make accelerated computing broadly accessible to all Linux developers."*
>
> Oscar Hernandez
> Oak Ridge National Laboratories

OAK RIDGE National Laboratory

OpenACC Directives for Accelerators

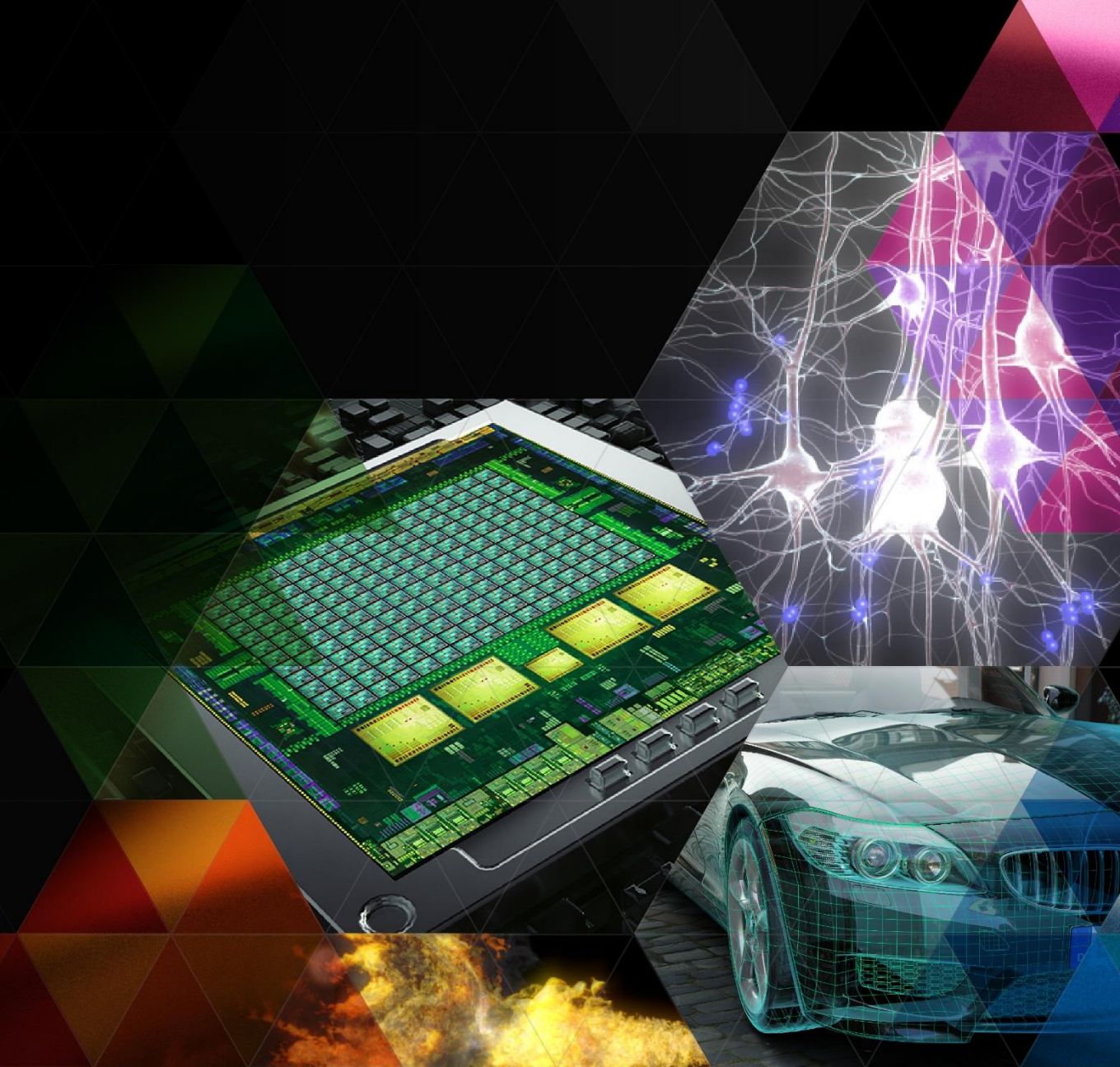
NUMBA PYTHON COMPILER

- Free and open source compiler for array-oriented Python
- NEW numba.cuda module integrates CUDA directly into Python

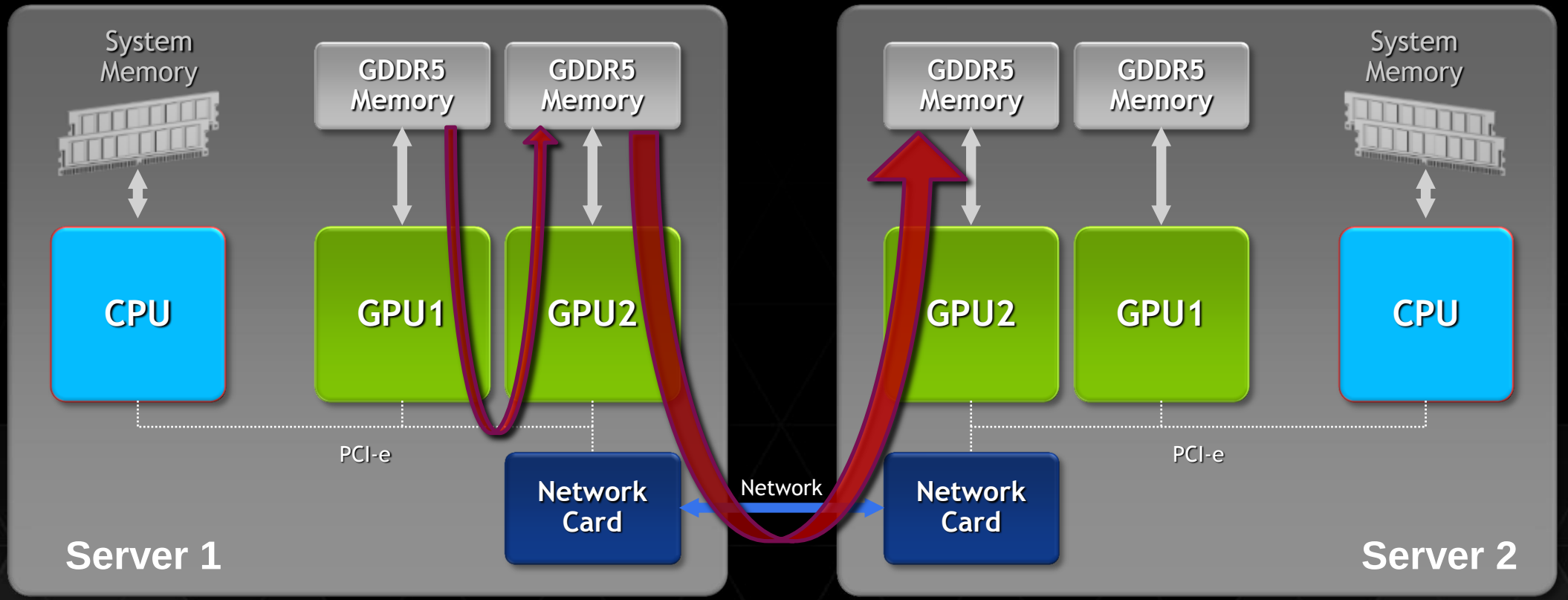
```
@cuda.jit("void(float32[:], float32, float32[:], float32[:])")
def saxpy(out, a, x, y):
    i = cuda.grid(1)
    out[i] = a * x[i] + y[i]

# Launch saxpy kernel
saxpy[griddim, blockdim](out, a, x, y)
```

- http://numba.pydata.org/

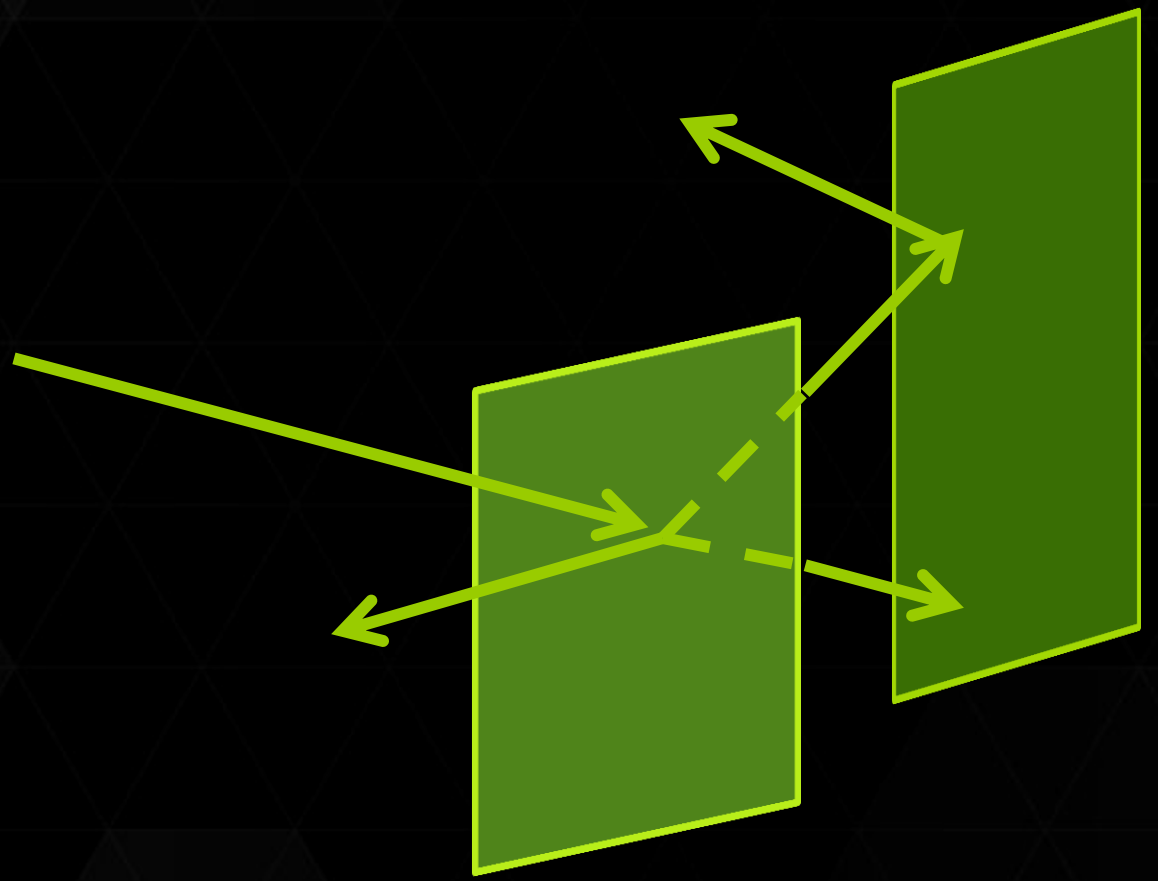


KEPLER ENABLES NVIDIA GPUDIRECT™ RDMA



http://docs.nvidia.com/cuda/gpudirect-rdma

IF YOUR APPLICATION LOOKS LIKE THIS..

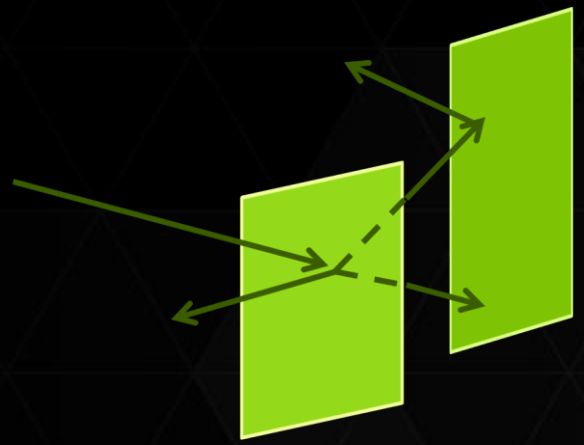


35

.. YOU MIGHT BE INTERESTED IN OPTIX

- Ray-tracing framework
  - Build your own RT application

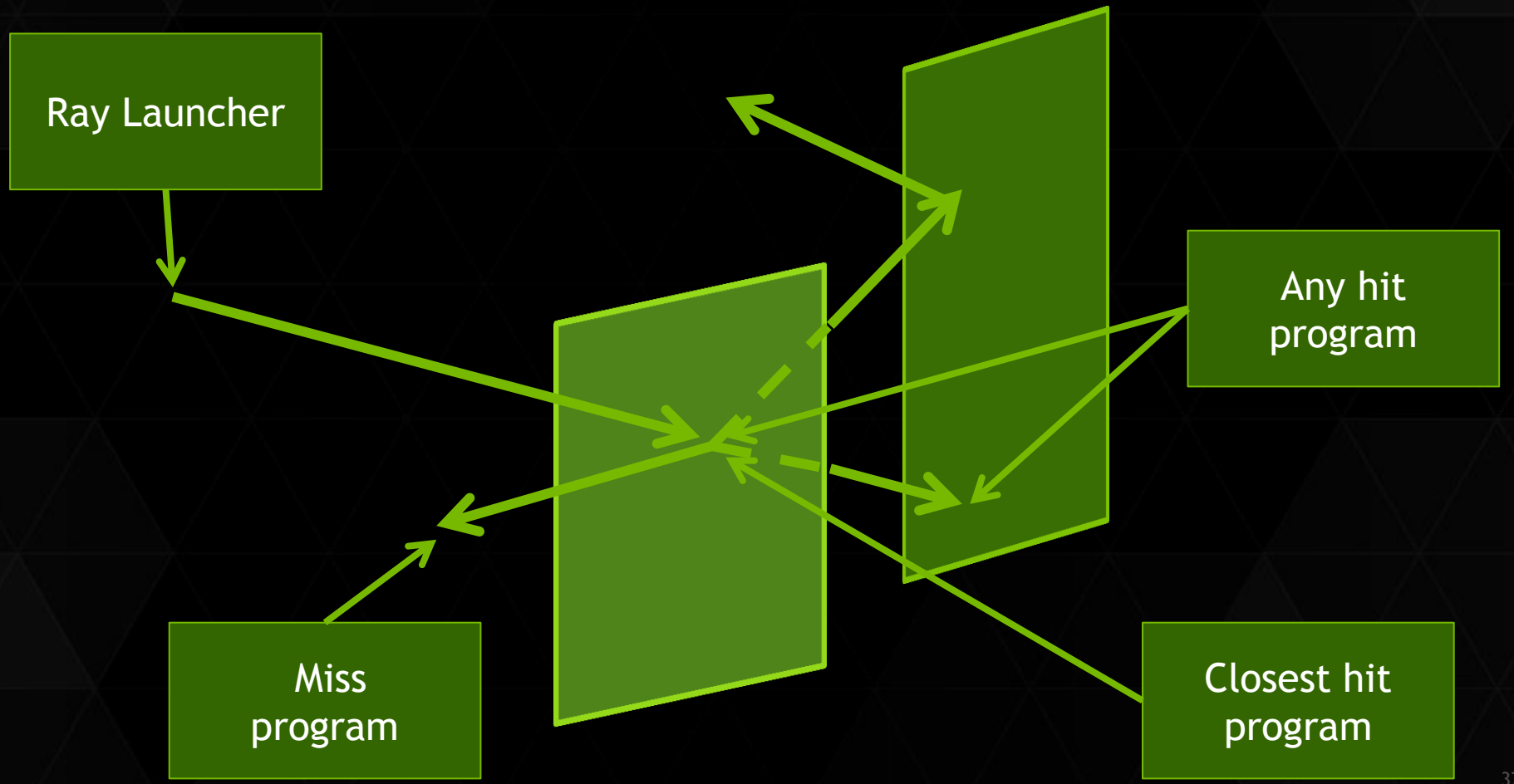
- Generic Ray-Geometry interaction
  - Rays with arbitrary payloads



- Multi-GPU support

36

DIFFERENT PROGRAMS GET INVOKED FOR DIFFERENT RAYS



37