Les dernières générations chez NVIDIA : Accélérateurs Tesla K20 et K20X.

Journée cartes graphiques et calcul intensif
Observatoire Midi-Pyrénées
TOULOUSE 17 avril 2013

François Courteille, NVIDIA, Paris, France
(fcourteille@nvidia.com)

NVIDIA

François Courteille

- Solutions Architect @ Nvidia since 2010
- E-mail address : fcourteille@nvidia.com
- 33 years of experience in HPC

- Experience Summary
  - NEC 1995-2010
  - CONVEX 1990-1995
  - EVANS & SUTHERLAND 1989-1990
  - CONTROL DATA - ETA 1979-1989

OUTLINE

- NVIDIA and GPU Computing
  - *GTC 2013 & Roadmaps*
- Inside Kepler Architecture
  - *SXM*
  - *Hyper-Q*
  - *Dynamic Parallelism*
- Programming GPUs – The Software Ecosystem
  - *OpenACC*
  - *Libraries*
  - *Languages and Frameworks*
- CUDA 5
  - *Nsight for Linux & Mac*
  - *GPU Direct RDMA*
  - *Library Object Linking (separate compilation & linking)*

NVIDIA - Core Technologies and Brands

GPU

GeForce®
Quadro® , Tesla®

Mobile

Tegra®

Cloud

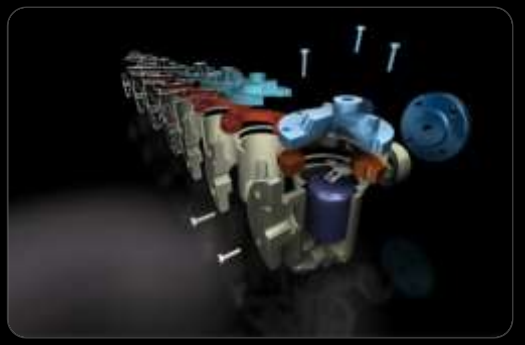
VGX ™
GeForce® GRID

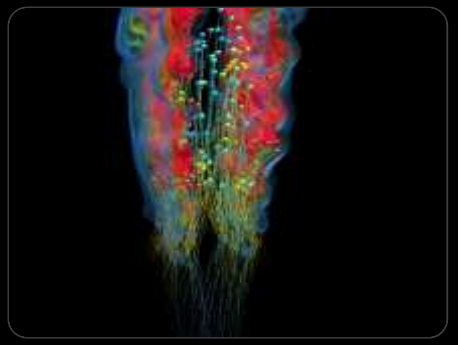
Founded 1993
Invented GPU 1999 – Computer Graphics

In 2013 GTC did expand…

GTC 2013

- Developer / Compute
- HPC / Supercomputing
- Graphics
- Life Science
- Oil & Gas
- Finance
- Manufacturing – CAE / CAD / Styling / Design
- Large Venue Visualization
- M&E – Animation / Editing / Rendering
- Cloud Graphics
- Mobile App & Game Development
- PC Game Development





http://www.gputechconf.com/gtcnew/on-demand-gtc.php

GTC 2013 Announcements

GPU Roadmap
NV Blog · ZDNet

Kayla: ARM + CUDA Platform
NV Blog · AnandTech

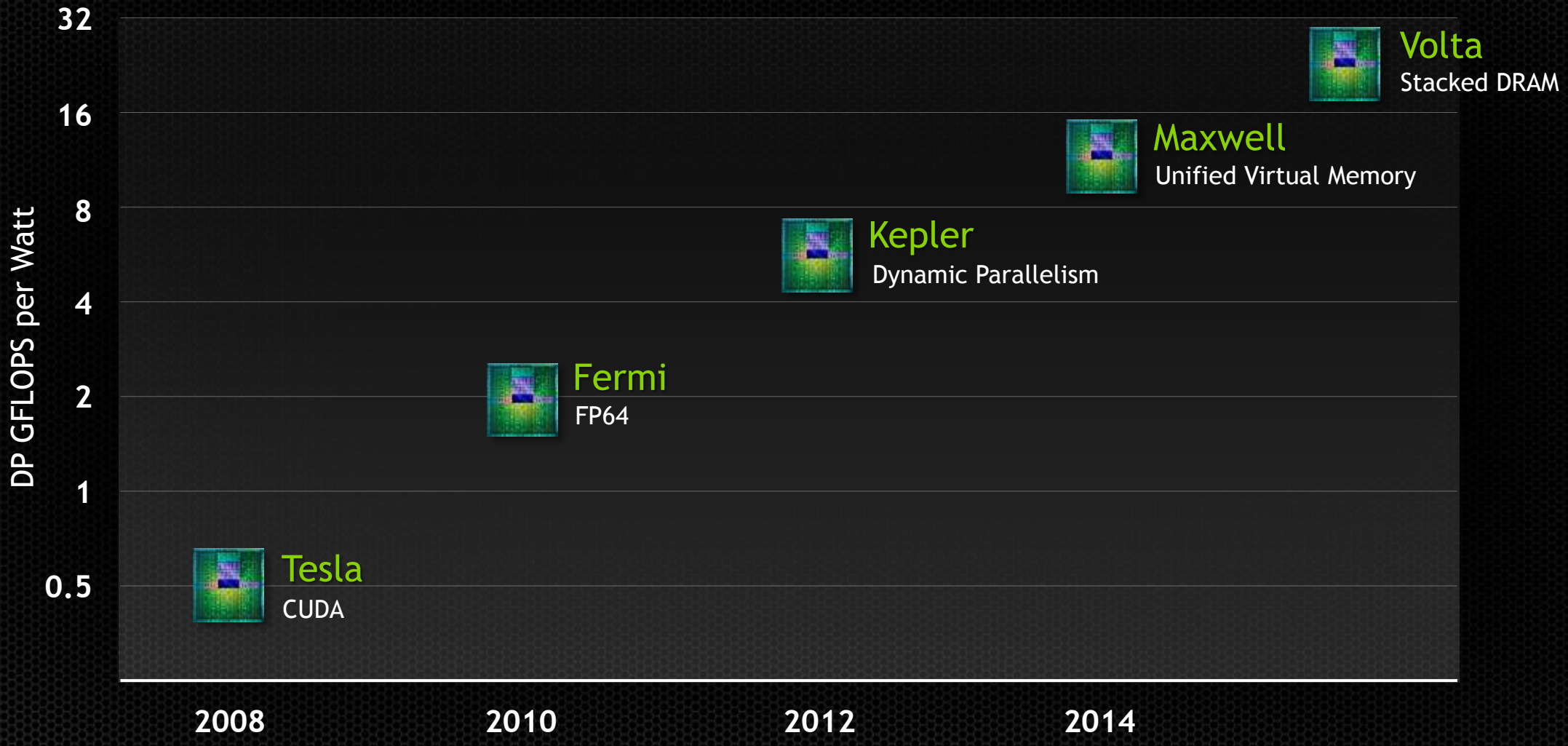
Big Data Analytics
NV Press Release · The Register

CUDA Python
NV Press Release · AnandTech

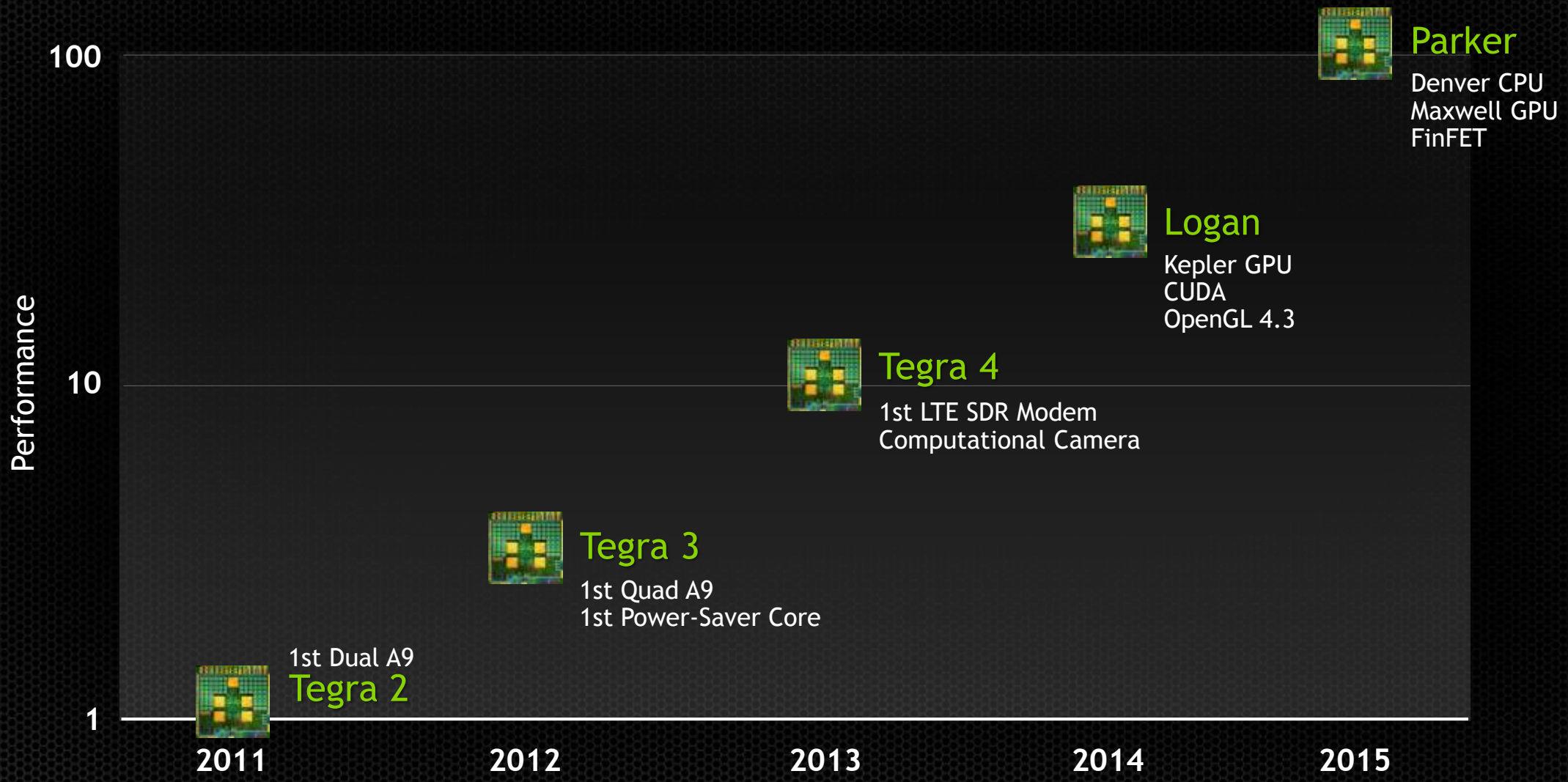
OpenACC 2.0
Draft Spec · HPCWire

CSCS Supercomputer
NV Blog · HPCWire

Tesla CUDA Architecture Roadmap



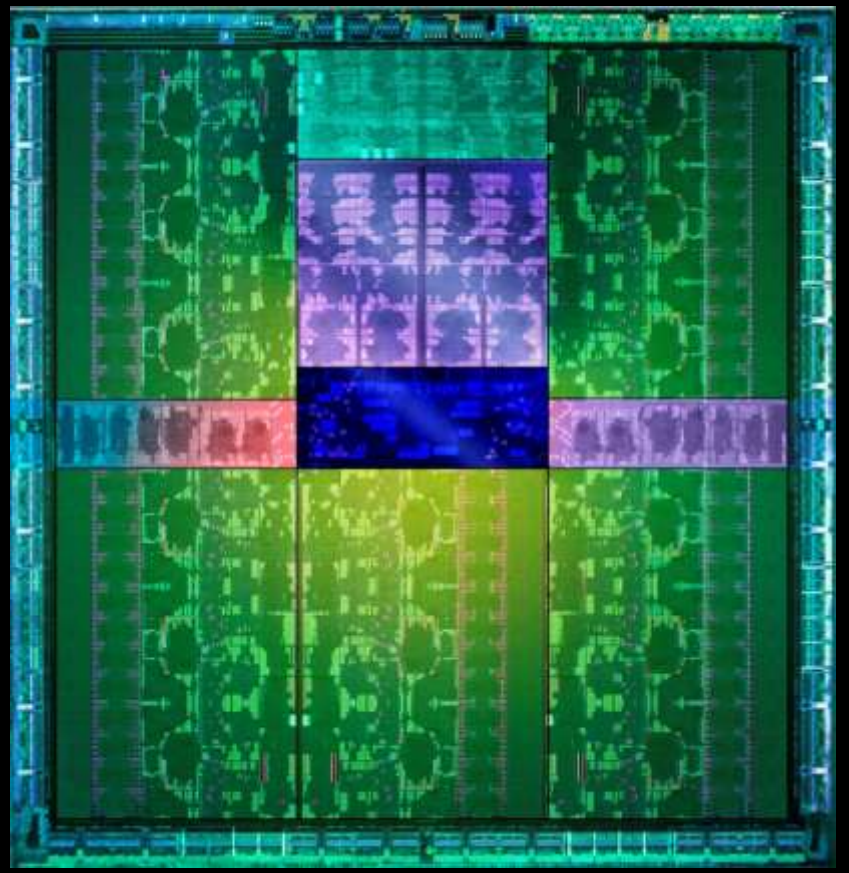
Tegra Roadmap




GTX 690
NVIDIA

Kepler GPU

Fastest, Most Efficient HPC Architecture Ever



SMX	3x Performance per Watt
Hyper-Q	Easy Speed-up for Legacy MPI Apps
Dynamic Parallelism	Parallel Programming Made Easier than Ever

Tesla K10

3x Single Precision

1.8x Memory Bandwidth

Image, Signal, Seismic, Life Sci (MD)

Tesla K20

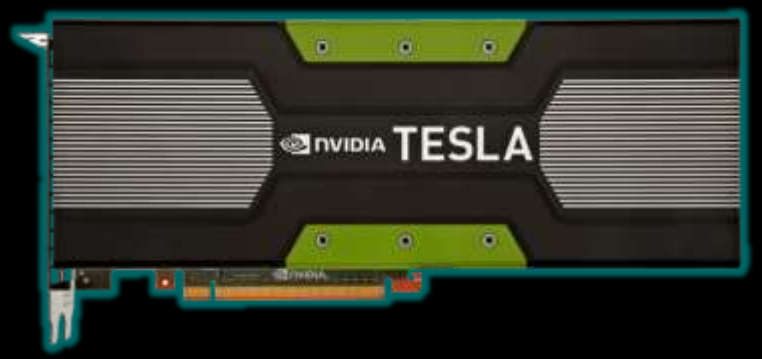
3x Double Precision

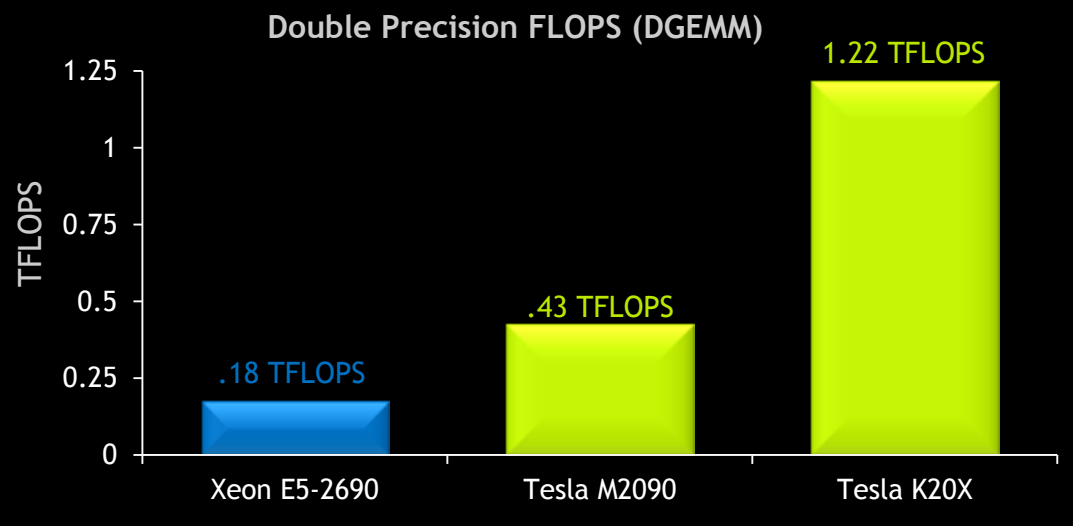
Hyper-Q, Dynamic Parallelism

CFD, FEA, Finance, Physics

Tesla K20 Family: 3x Faster Than Fermi

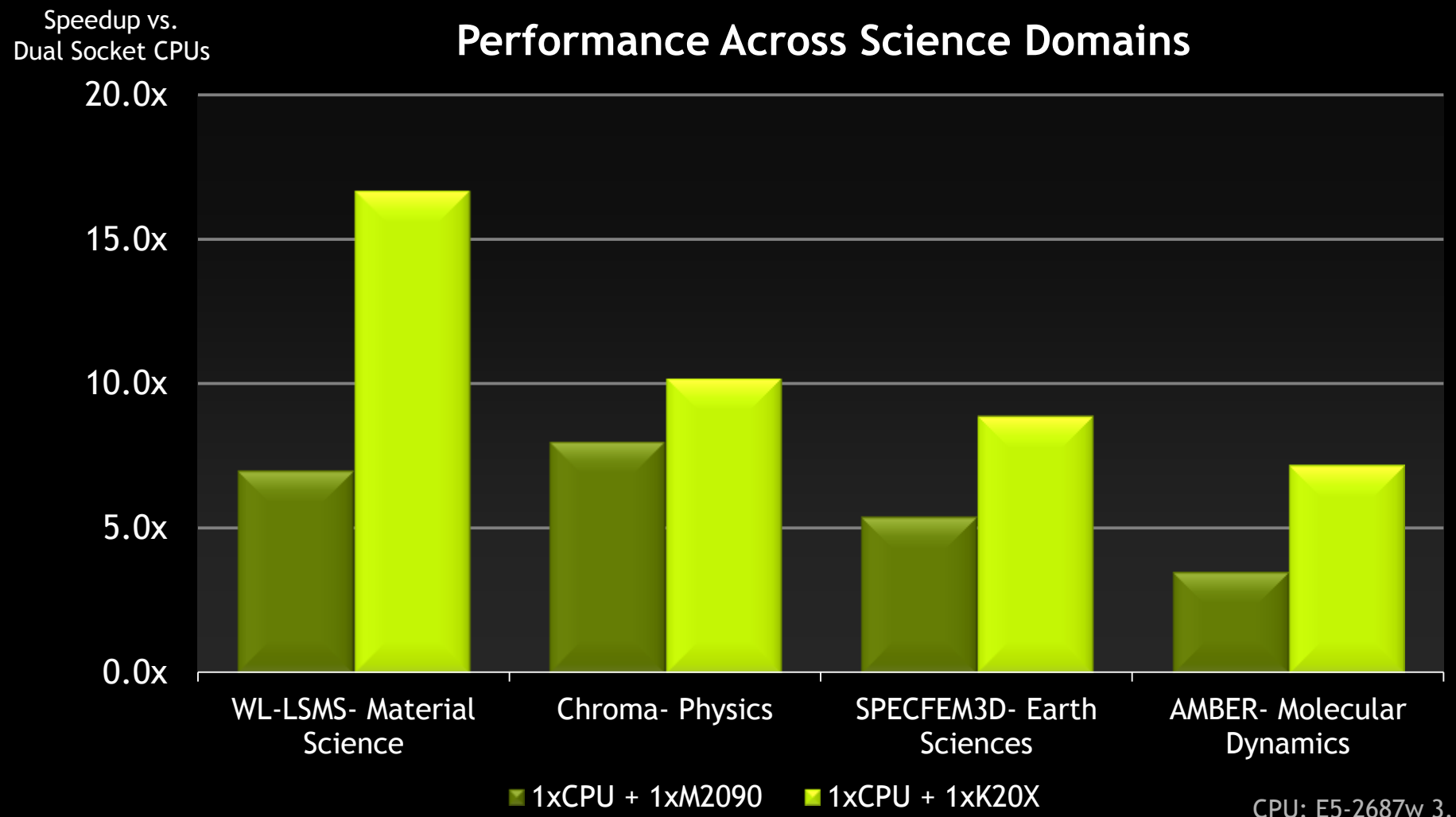
Tesla K20X





	Tesla K20X	Tesla K20
# CUDA Cores	2688	2496
Peak Double Precision	1.32 TF	1.17 TF
Peak DGEMM	1.22 TF	1.10 TF
Peak Single Precision	3.95 TF	3.52 TF
Peak SGEMM	2.90 TF	2.61 TF
Memory Bandwidth	250 GB/s	208 GB/s
Memory size	6 GB	5 GB
Total Board Power	235W	225W

Up to 10x on Leading Applications



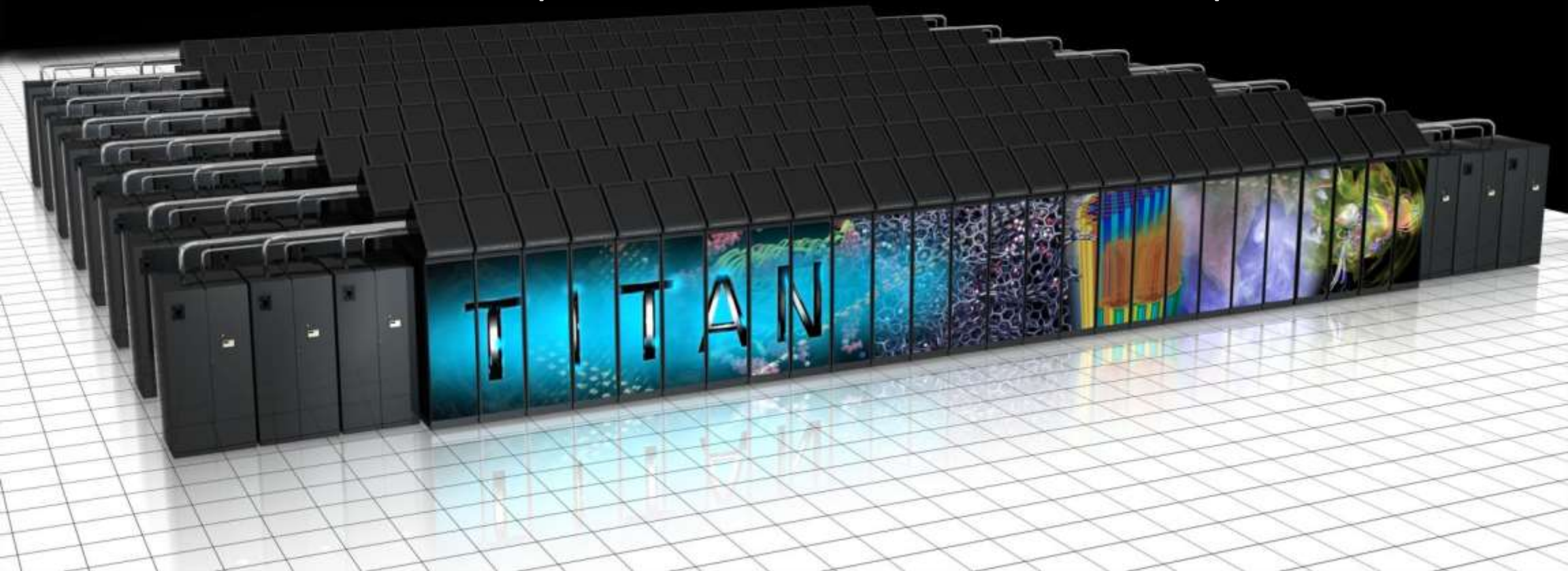
CPU: E5-2687w 3.10 GHz Sandy Bridge

Titan: World's Fastest Supercomputer

18,688 Tesla K20X GPUs

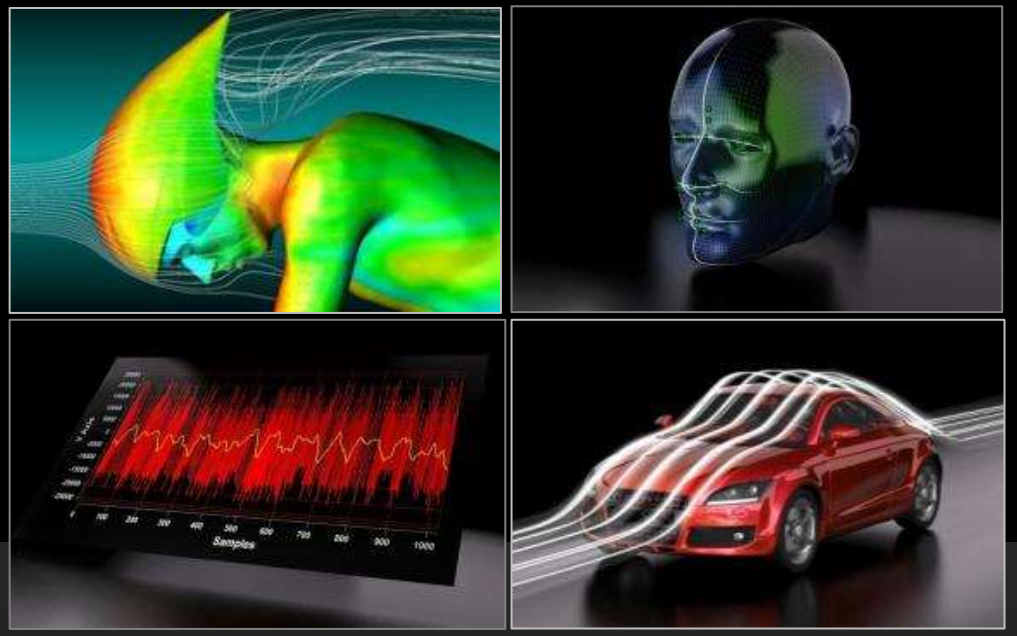
27 Petaflops Peak: 90% of Performance from GPUs

17.59 Petaflops Sustained Performance on Linpack



GPU Test Drive

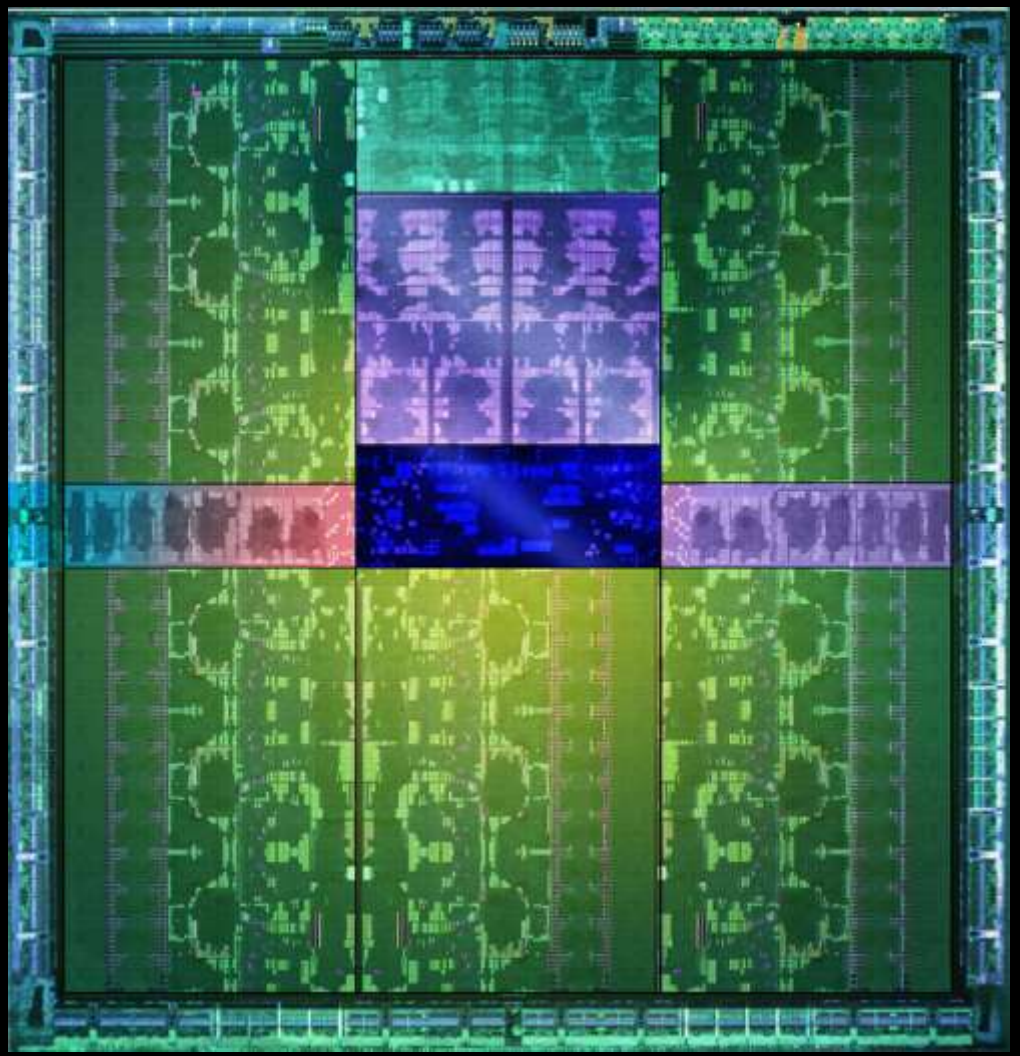
Double your Fermi Performance with Kepler GPUs



www.nvidia.com/GPUTestDrive

NVIDIA

Tesla K20/K20X Details



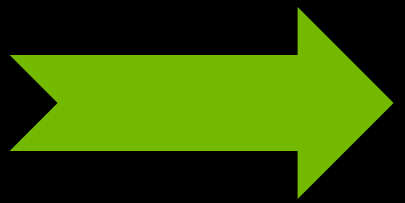
Whitepaper: http://www.nvidia.com/object/nvidia-kepler.html

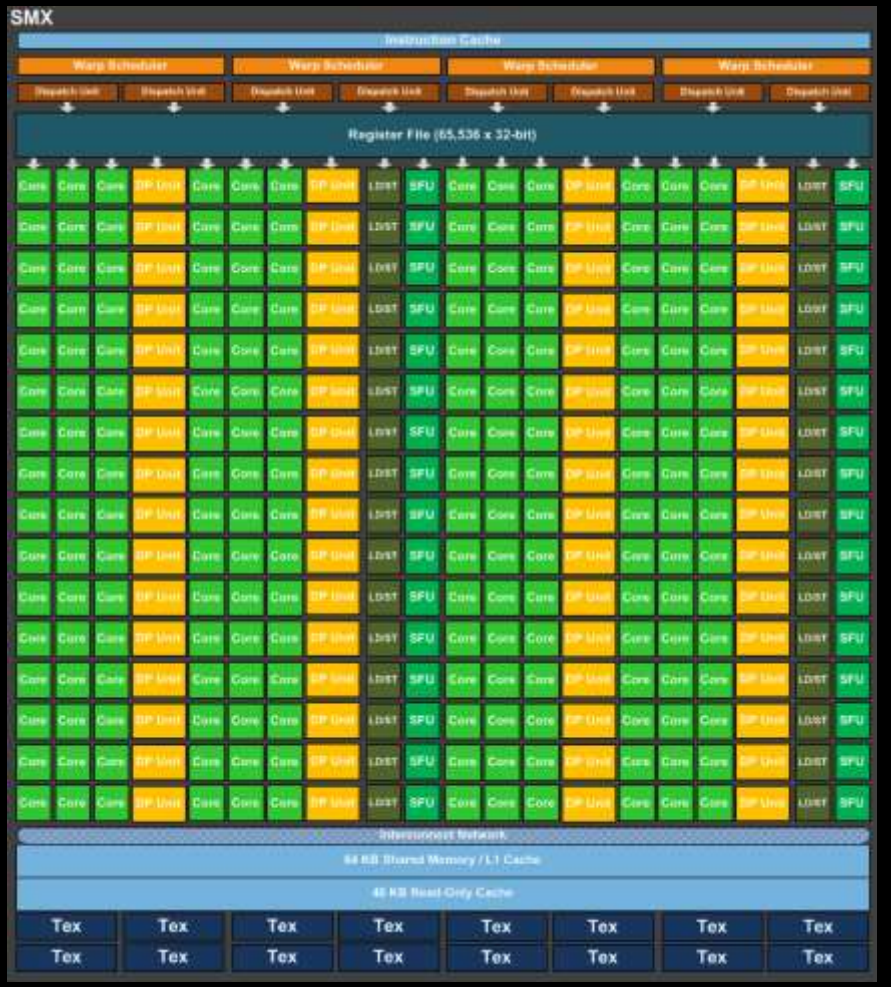
Kepler GK110 Block Diagram

Architecture

- 7.1B Transistors
- 15 SMX units
- > 1 TFLOP FP64
- 1.5 MB L2 Cache
- 384-bit GDDR5
- PCI Express Gen2/Gen3

Kepler GK110 SMX vs Fermi SM





SMX Balance of Resources

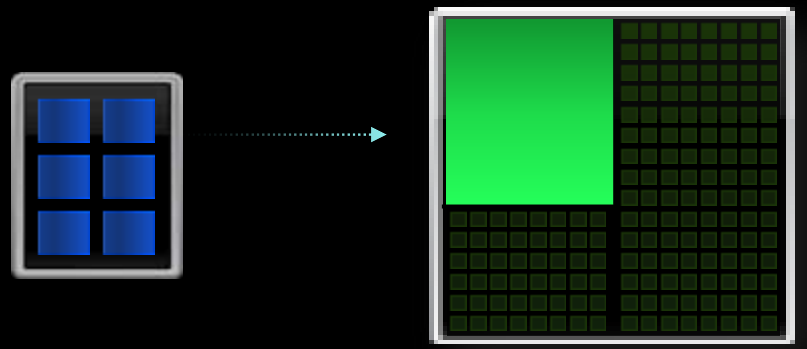
Resource	Kepler GK110 vs Fermi
Floating point throughput	**2-3x**
Max Blocks per SMX	**2x**
Max Threads per SMX	**1.3x**
Register File Bandwidth	**2x**
Register File Capacity	**2x**
Shared Memory Bandwidth	**2x**
Shared Memory Capacity	**1x**

Hyper-Q

CPU Cores Simultaneously Run Tasks on Kepler

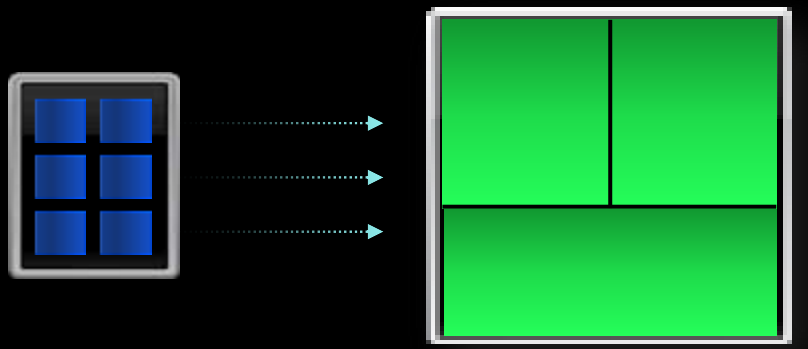
FERMI

1 MPI Task at a Time



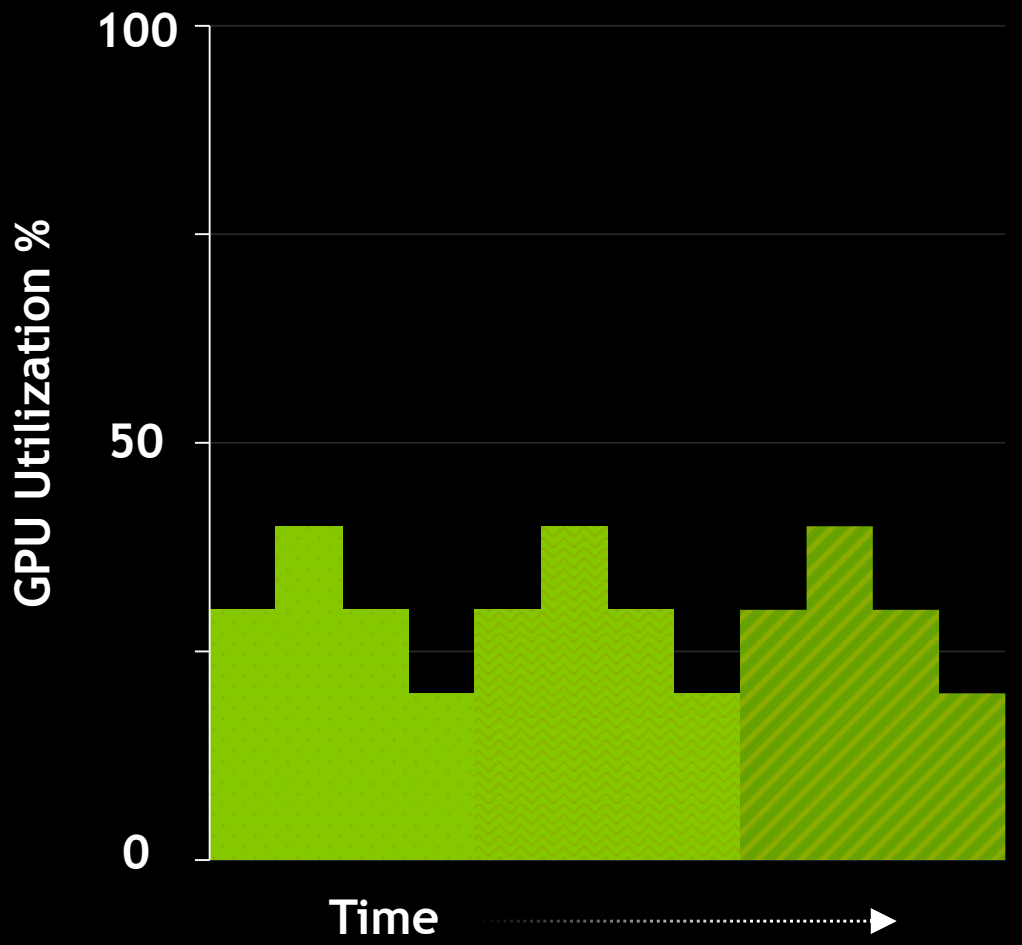
KEPLER

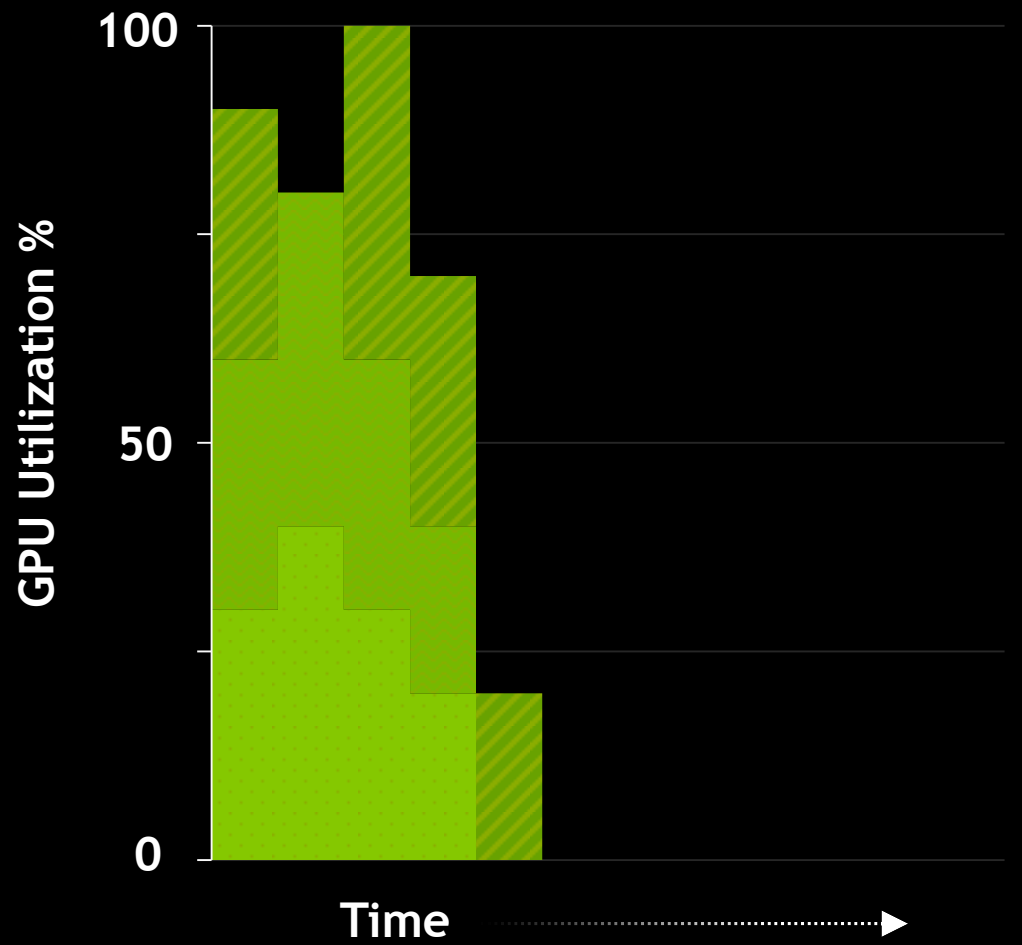
32 Simultaneous MPI Tasks



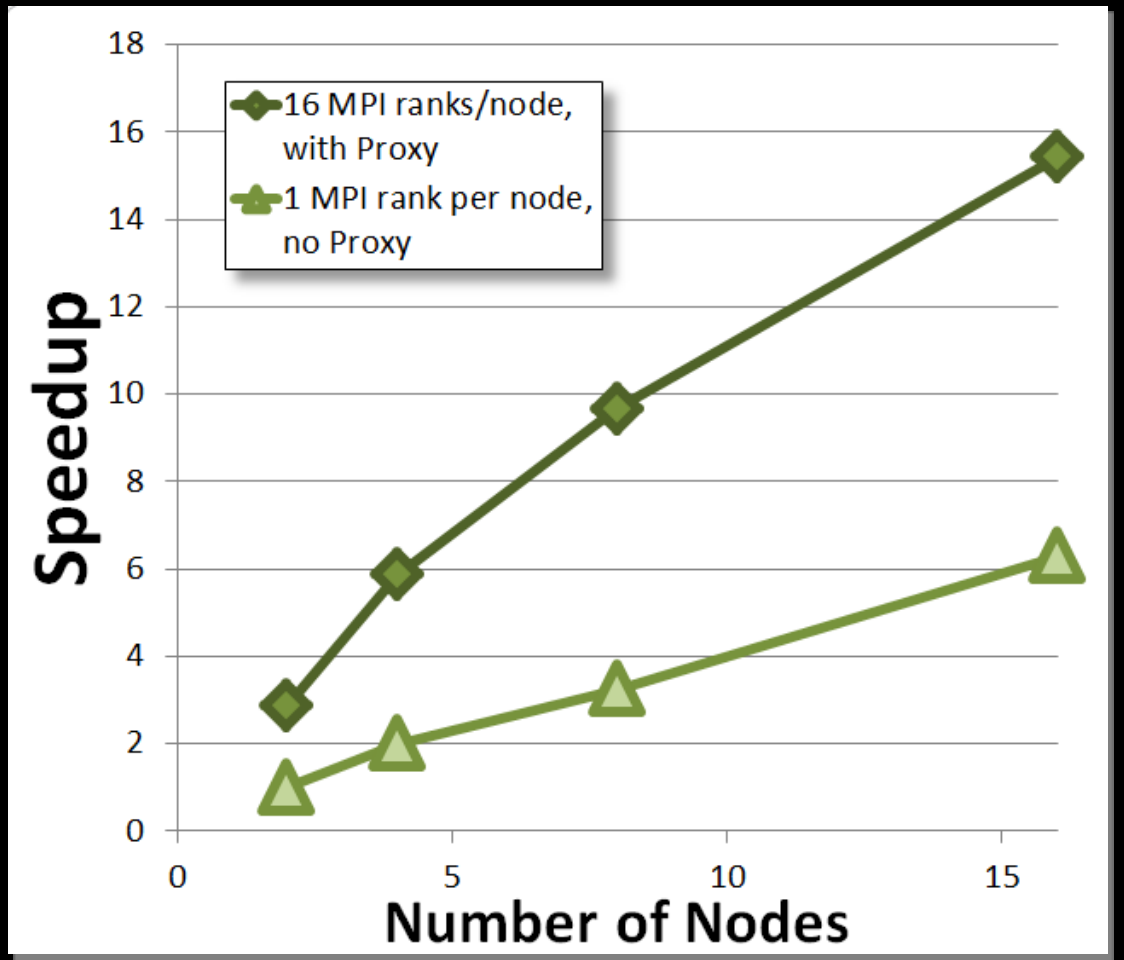
Hyper-Q

Max GPU Utilization, Slashes CPU Idle Time



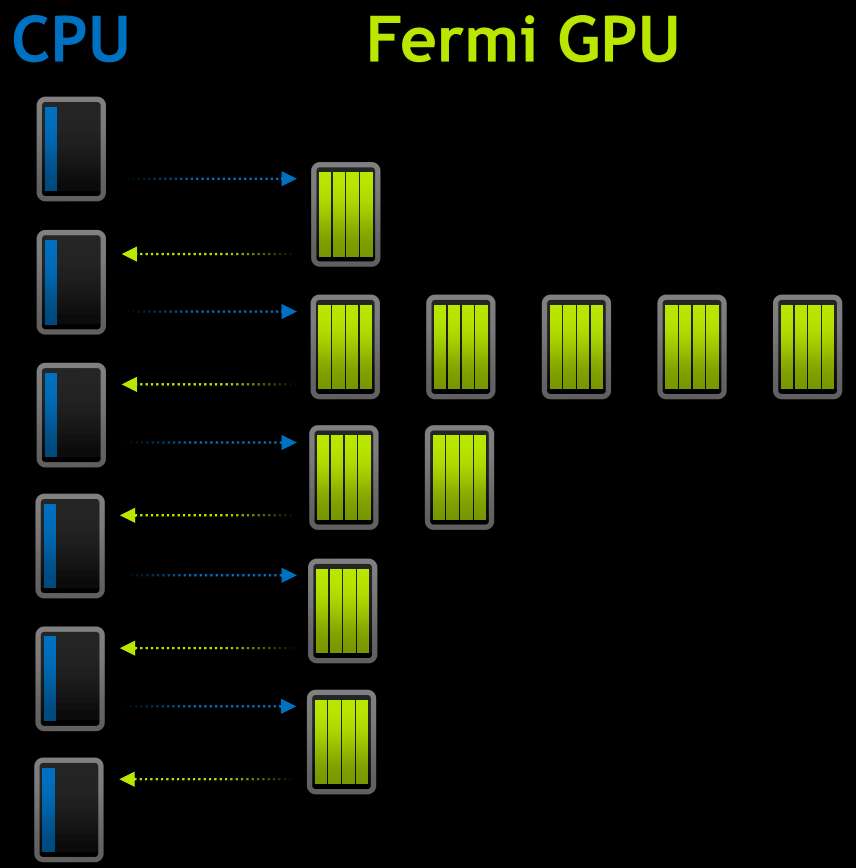


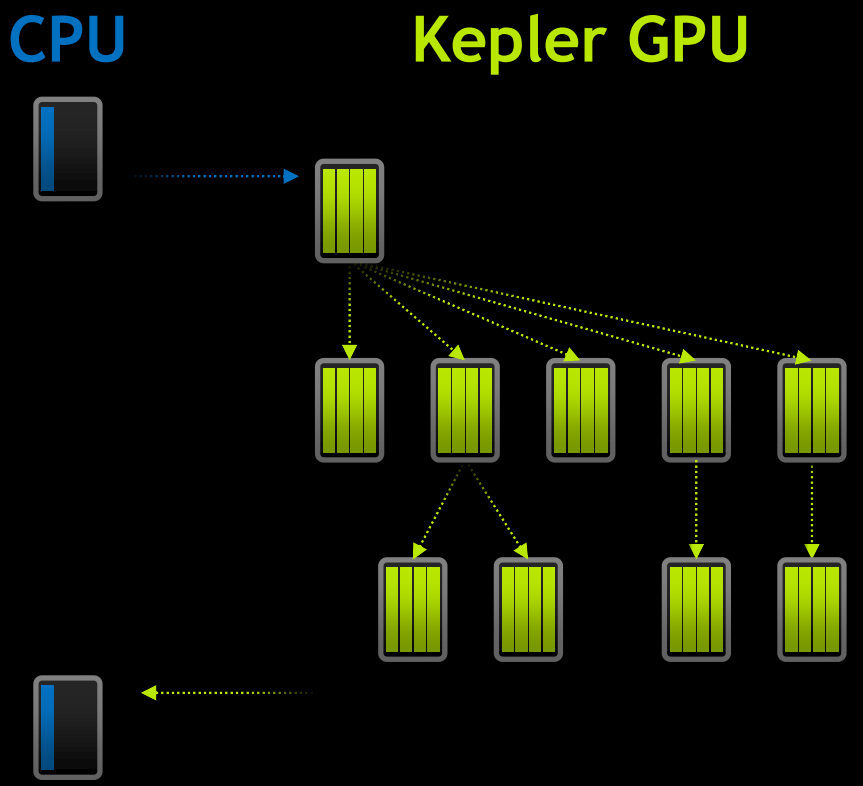
Example: Hyper-Q/Proxy for CP2K



Dynamic Parallelism

GPU Adapts to Data, Dynamically Launches New Threads





CUDA Dynamic Parallelism

Kernel launches grids

Identical syntax as host

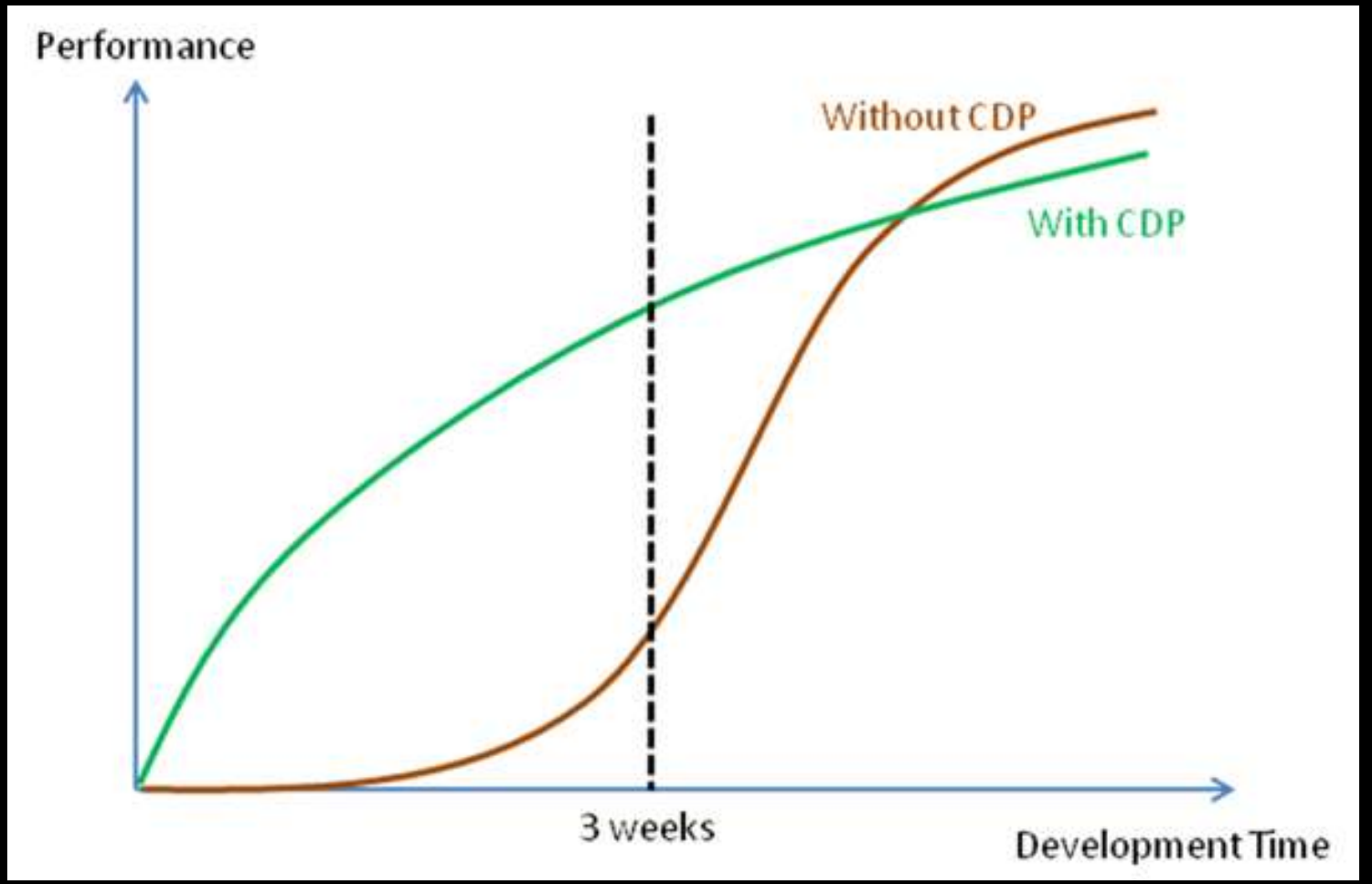
CUDA runtime function in cudadevrt library

```
__global__ void childKernel()
{
 printf("Hello %d", threadIdx.x);
}
```

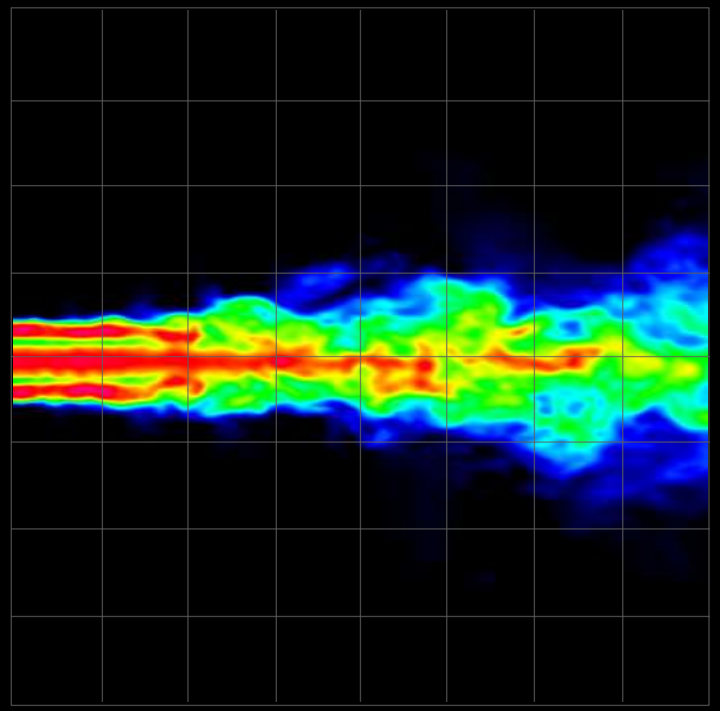
```
__global__ void parentKernel()
{
   childKernel<<<1,10>>>();
   cudaDeviceSynchronize();
   printf("World!\n");
}
```

```
int main(int argc, char *argv[])
{
  parentKernel<<<1,1>>>();
  cudaDeviceSynchronize();
  return 0;
}
```

CUDA Dynamic Parallelism and Programmer Productivity



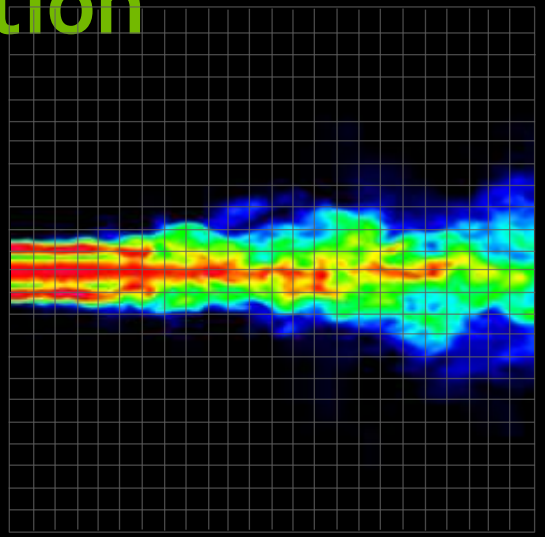
Dynamic Work Generation



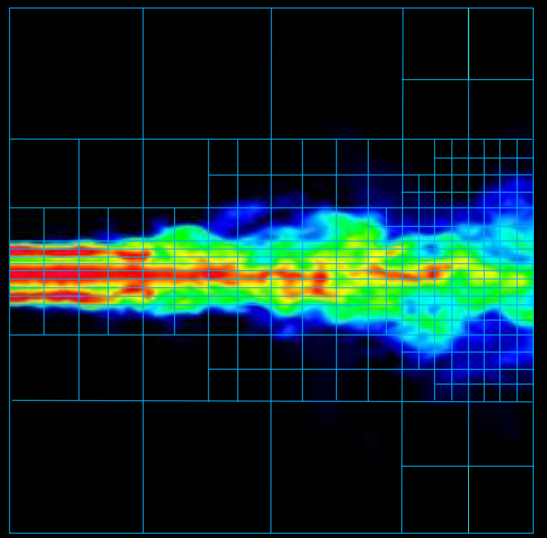
Initial Grid

Statically assign conservative worst-case grid

Fixed Grid



Dynamically assign performance where accuracy is required



Dynamic Grid

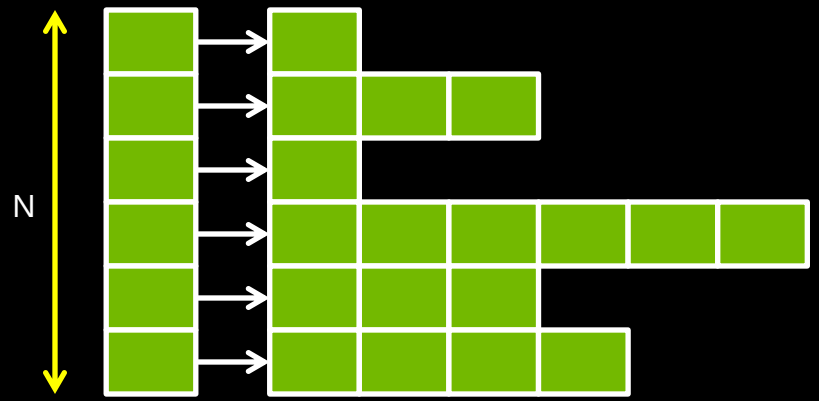
Nested Parallelism Made Possible

Serial Program

```
for i = 1 to N
    for j = 1 to x[i]
        convolution(i, j)
    next j
next i
```

CUDA Program

```
__global__ void convolution(int x[])
{
    for j = 1 to x[blockIdx]
        kernel<<< ... >>>(blockIdx, j)
}


convolution<<< N, 1 >>>(x);
```



Now Possible: Dynamic Parallelism

GPU Management: nvidia-smi

Multi-GPU systems are widely available

Different systems are set up differently

Want to get quick information on

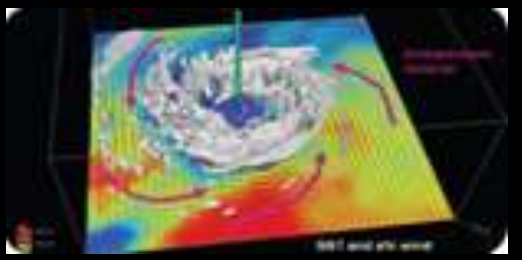
- Approximate GPU utilization
- Approximate memory footprint
- Number of GPUs
- ECC state
- Driver version

Inspect and modify GPU state

```
Thu Nov  1 09:10:29 2012
+------------------------------------------------------+
| NVIDIA-SMI 4.304.51   Driver Version: 304.51         |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     | Bus-Id        Disp.  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage         | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K20X               | 0000:03:00.0     Off |                  Off |
| N/A   30C    P8    28W / 235W |   0%   12MB / 6143MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20X               | 0000:85:00.0     Off |                  Off |
| N/A   28C    P8    26W / 235W |   0%   12MB / 6143MB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Compute processes:                                               GPU Memory |
|  GPU       PID  Process name                                     Usage      |
|=============================================================================|
|  No running compute processes found                                         |
+-----------------------------------------------------------------------------+
```

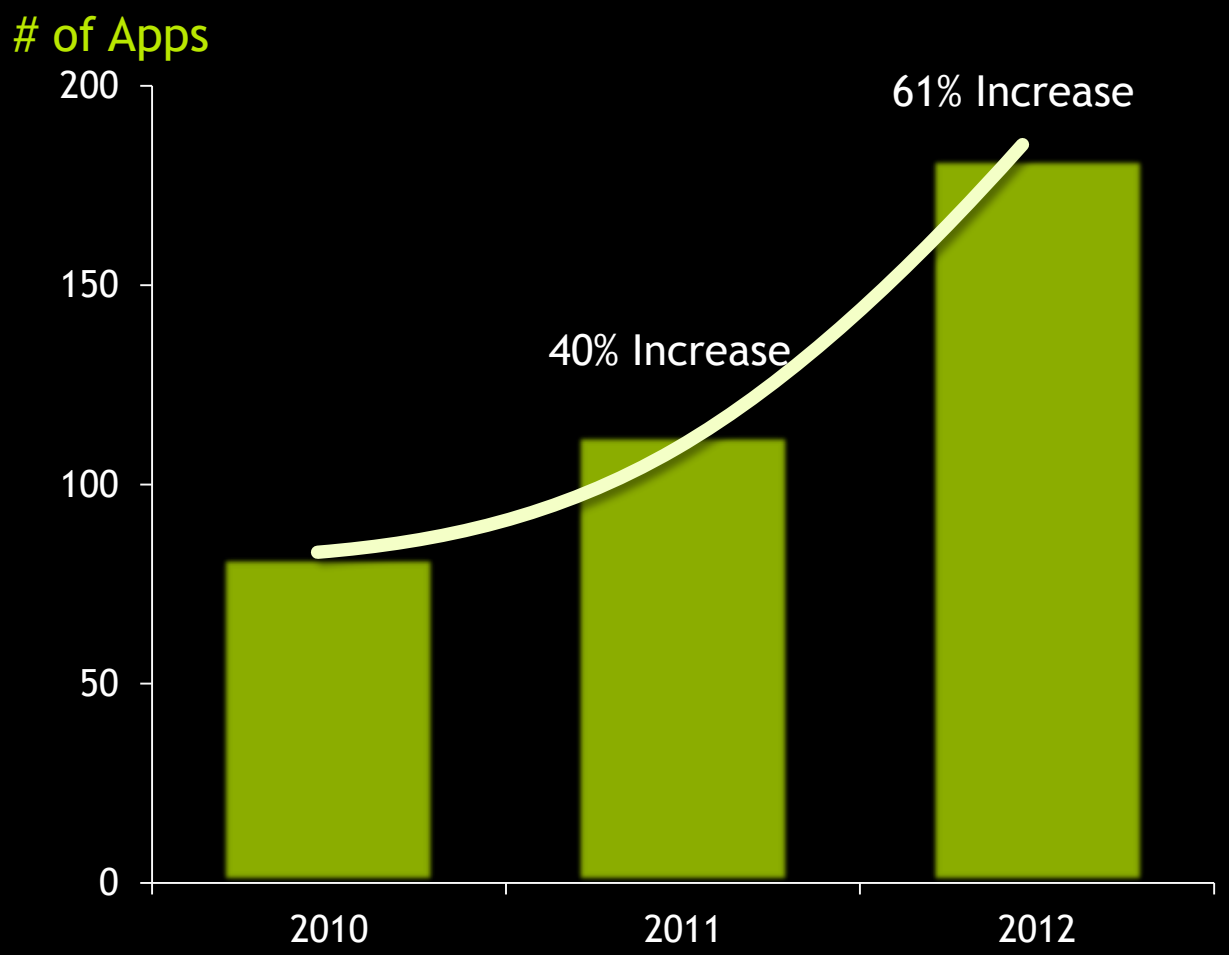
OpenGL and Tesla

- Tesla K20/K20X for high performance Compute
- Tesla K20/K20X for Graphics and Compute
- Use interop to mix OpenGL and Compute



Tesla K20 / K20X

CUDA Accelerating Key Apps



Top Supercomputing Apps		
Computational Chemistry	AMBER CHARMM GROMACS	LAMMPS NAMD DL_POLY
Material Science	QMCPACK Quantum Espresso GAMESS	Gaussian NWChem VASP
Climate & Weather	COSMO GEOS-5	CAM-SE NIM WRF
Physics	Chroma Denovo GTC	GTS ENZO MILC
CAE	ANSYS Mechanical MSC Nastran SIMULIA Abaqus	ANSYS Fluent OpenFOAM LS-DYNA

Accelerated, In Development

200+ GPU-Accelerated Applications

www.nvidia.com/appscatalog

CUDA Parallel Computing Platform

www.nvidia.com/getcuda

Programming Approaches

- **Libraries** — "Drop-in" Acceleration
- **OpenACC Directives** — Easily Accelerate Apps
- **Programming Languages** — Maximum Flexibility

Development Environment

Nsight IDE
Linux, Mac and Windows
GPU Debugging and Profiling

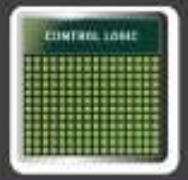
CUDA-GDB debugger
NVIDIA Visual Profiler

Open Compiler Tool Chain

Enables compiling new languages to CUDA platform, and CUDA languages to other architectures

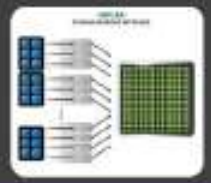
Hardware Capabilities

SMX



Dynamic Parallelism

HyperQ



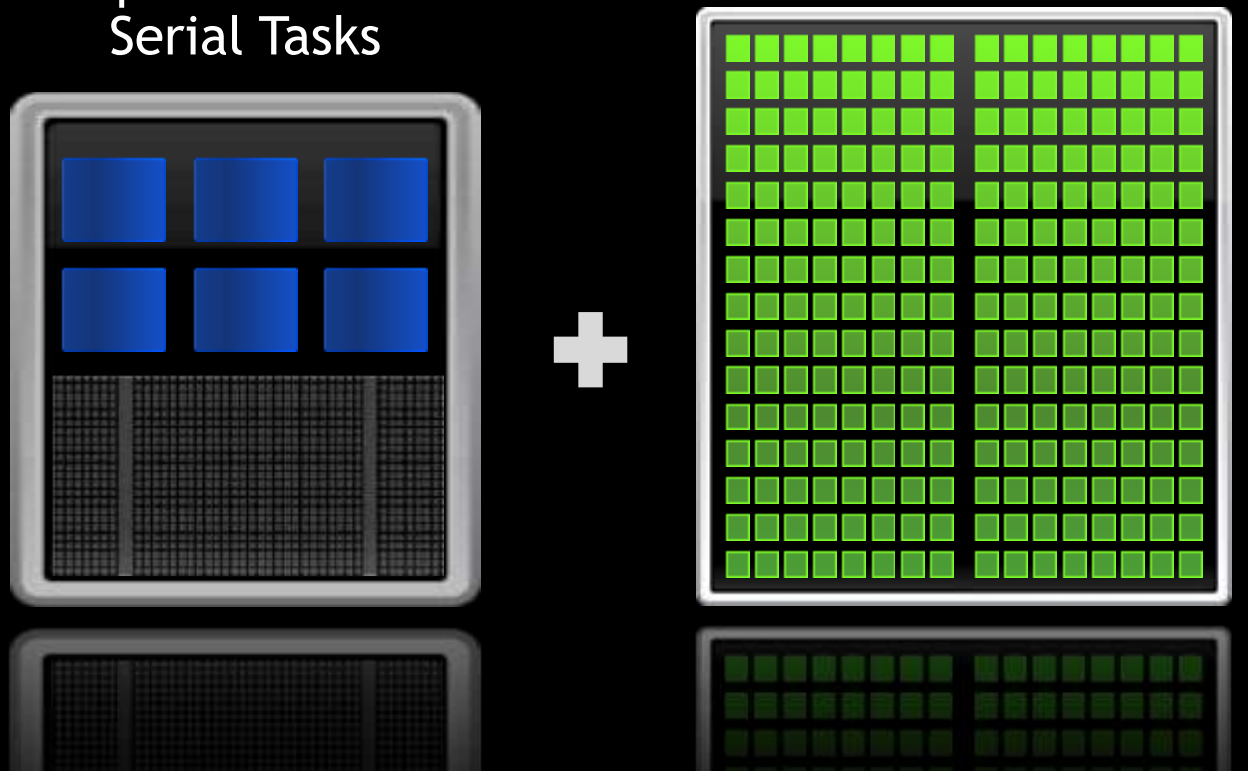
GPUDirect

Accelerated Computing

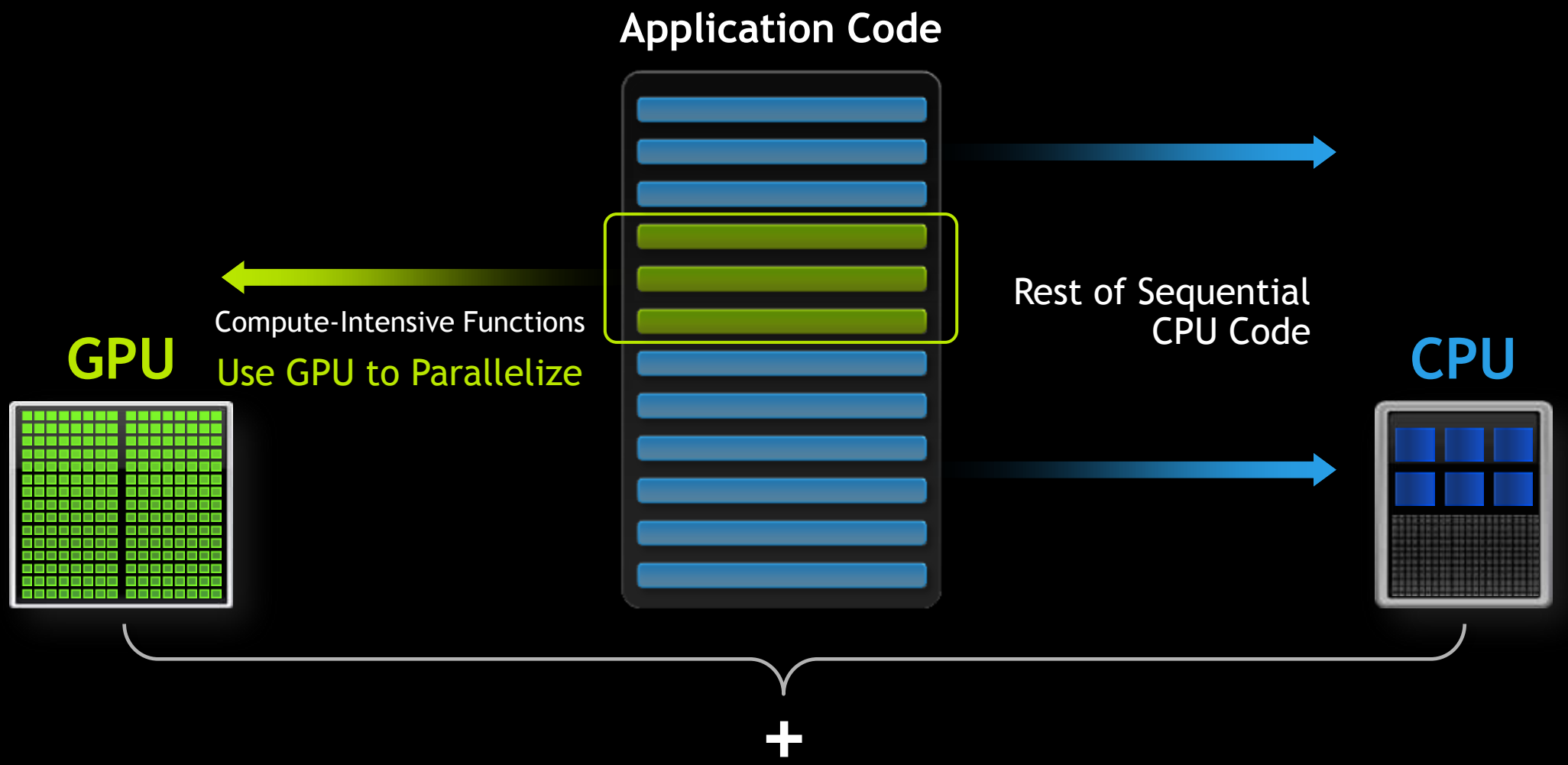
10x Performance, 5x Energy Efficiency

CPU
Optimized for
Serial Tasks

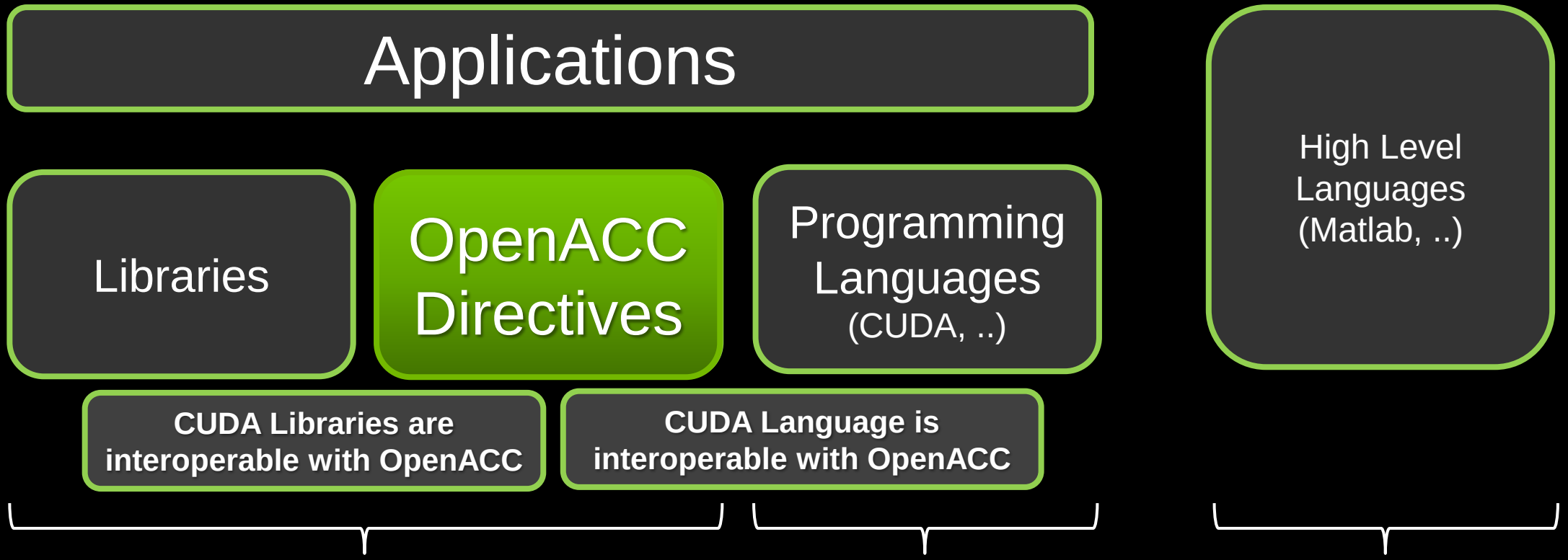
GPU Accelerator
Optimized for Many
Parallel Tasks



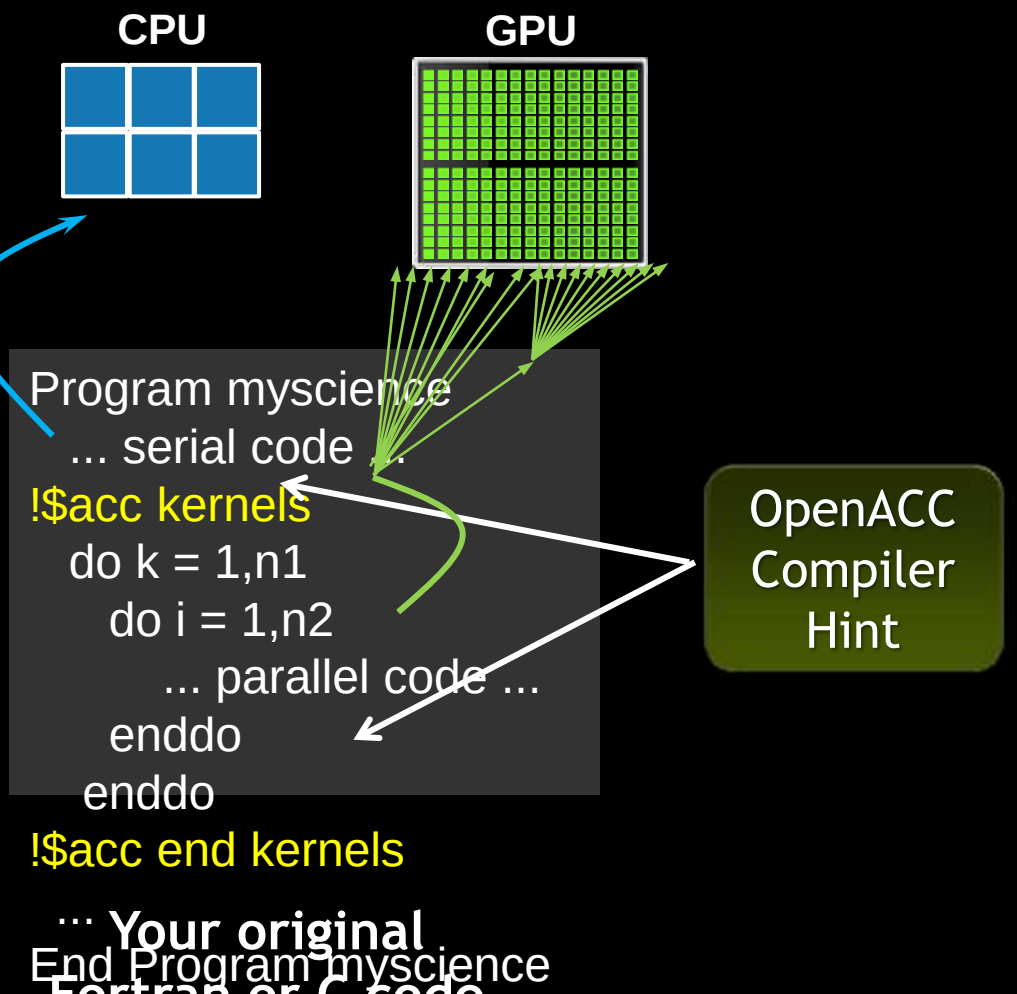
Small Changes, Big Speed-up



3 Ways to Accelerate Applications



OpenACC Directives



Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs & multicore CPUs

OpenACC
The Standard for GPU Directives

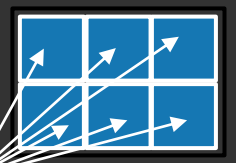
- **Easy:** Directives are the easy path to accelerate compute intensive applications
- **Open:** OpenACC is an open GPU directives standard, making GPU programming straightforward and portable across parallel and multi-core processors
- **Powerful:** GPU Directives allow complete access to the massive parallel power of a GPU

OpenACC®
DIRECTIVES FOR ACCELERATORS

Familiar to OpenMP Programmers

OpenMP

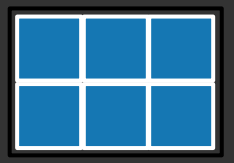
CPU



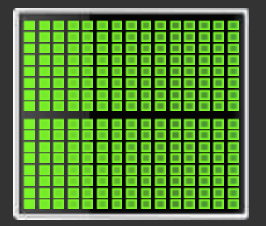
```
main() {
  double pi = 0.0; long i;


  #pragma omp parallel for reduction(+:pi)
  for (i=0; i<N; i++)
  {
    double t = (double)((i+0.05)/N);
    pi += 4.0/(1.0+t*t);
  }

  printf("pi = %f\n", pi/N);
}
```

OpenACC

CPU



GPU



```
main() {
  double pi = 0.0; long i;

  #pragma acc kernels
  for (i=0; i<N; i++)
  {
    double t = (double)((i+0.05)/N);
    pi += 4.0/(1.0+t*t);
  }

printf("pi = %f\n", pi/N);
}
```

Small Effort. Real Impact.

Large Oil Company

3x in 7 days

Solving billions of equations iteratively for oil production at world's largest petroleum reservoirs

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Univ. of Houston

Prof. M.A. Kayali

20x in 2 days

Studying magnetic systems for innovations in magnetic storage media and memory, field sensors, and

THE UNIVERSITY OF MELBOURNE

Uni. Of Melbourne

Prof. Kerry Black

65x in 2 days

Better understand complex reasons by lifecycles of snapper fish in Port Phillip Bay

УГАТУ

Ufa State Aviation

Prof. Arthur Yuldashev

7x in 4 Weeks

Generating stochastic geological models of oilfield reservoirs with

GAMESS-UK

Dr. Wilkinson, Prof. Naidoo

10x

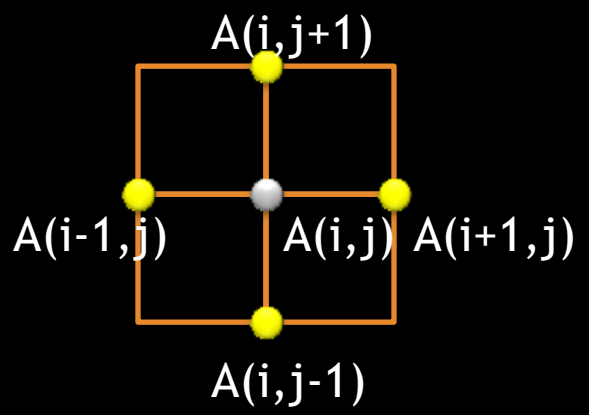
Used for various fields such as investigating biofuel production and molecular sensors

* borehole data

Achieved using the PGI Accelerator Com

Example: Jacobi Iteration

- Iteratively converges to correct value (e.g. Temperature), by computing new values at each point from the average of neighboring points.
  - Common, useful algorithm
  - Example: Solve Laplace equation in 2D: $\nabla^2 f(x, y) = 0$



$$A_{k+1}(i,j) = \frac{A_k(i-1,j) + A_k(i+1,j) + A_k(i,j-1) + A_k(i,j+1)}{4}$$

Jacobi Iteration Fortran Code

```
do while ( err > tol .and. iter < iter_max )
  err=0._fp_kind

  do j=1,m
    do i=1,n


      Anew(i,j) = .25_fp_kind * (A(i+1, j  ) + A(i-1, j  ) + &
                                 A(i  , j-1) + A(i  , j+1))


      err = max(err, Anew(i,j) - A(i,j))
    end do
  end do

  do j=1,m-2
    do i=1,n-2
      A(i,j) = Anew(i,j)
    end do
  end do

  iter = iter +1
end do
```

Iterate until converged

Iterate across matrix elements

Calculate new value from neighbors

Compute max error for convergence

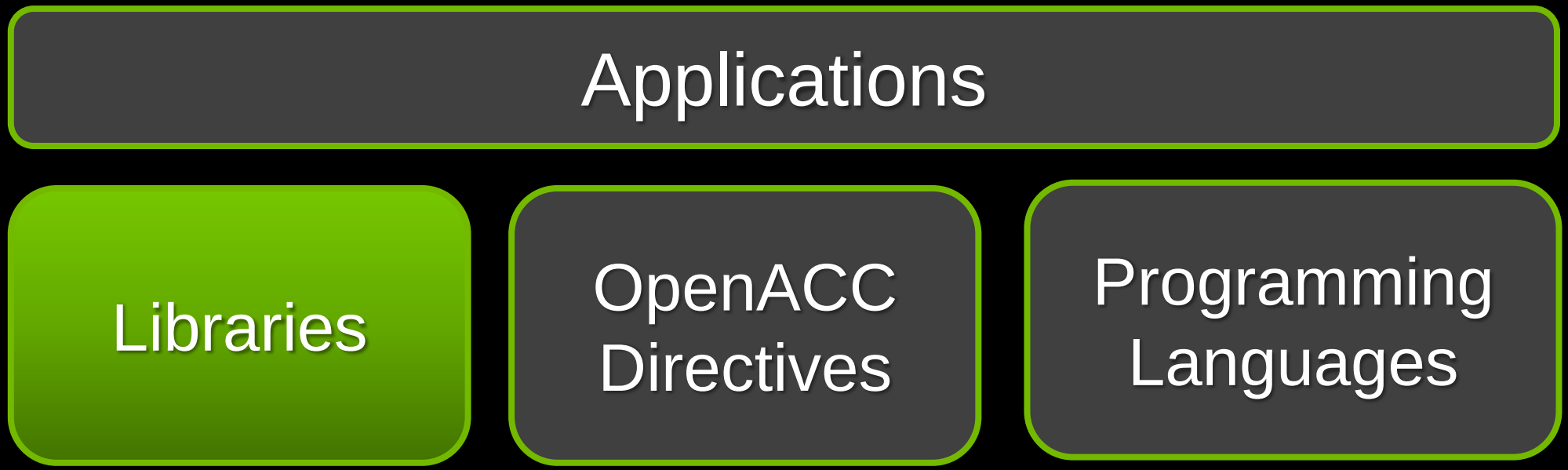
Swap input/output arrays

Jacobi Iteration: OpenACC Fortran Code

```
!$acc data copy(A), create(Anew)
do while ( err > tol .and. iter < iter_max )
  err=0._fp_kind

!$acc kernels
  do j=1,m
    do i=1,n

      Anew(i,j) = .25_fp_kind * (A(i+1, j  ) + A(i-1, j  ) + &
                                 A(i  , j-1) + A(i  , j+1))

      err = max(err, Anew(i,j) - A(i,j))
    end do
  end do
!$acc end kernels

  ...

iter = iter +1
end do
!$acc end data
```

Copy A in at beginning of loop, out at end. Allocate Anew on accelerator

3 Ways to Accelerate Applications



"Drop-in" Acceleration

Easily Accelerate Applications

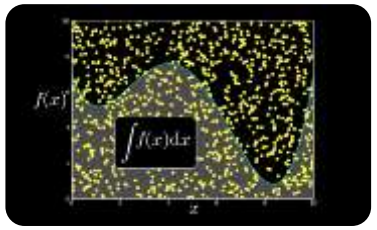
Maximum Flexibility

Libraries: Easy, High-Quality Acceleration

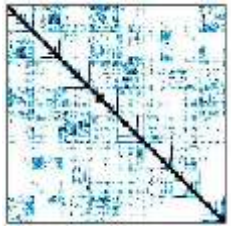
- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **"Drop-in":** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
- **Quality:** Libraries offer high-quality implementations of functions encountered in a broad range of applications
- **Performance:** NVIDIA libraries are tuned by experts

Some GPU-accelerated Libraries

NVIDIA cuBLAS



NVIDIA cuRAND



NVIDIA cuSPARSE

NVIDIA NPP

GPU VSIPL

Vector Signal Image Processing

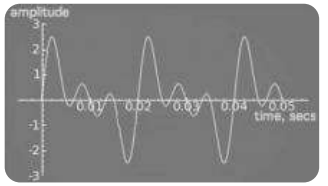
CULA tools

GPU Accelerated Linear Algebra

MAGMA

ICL UT

Matrix Algebra on GPU and Multicore



NVIDIA cuFFT

ROGUE WAVE SOFTWARE

IMSL Library

ArrayFire Matrix Computations

CUSP

Sparse Linear Algebra

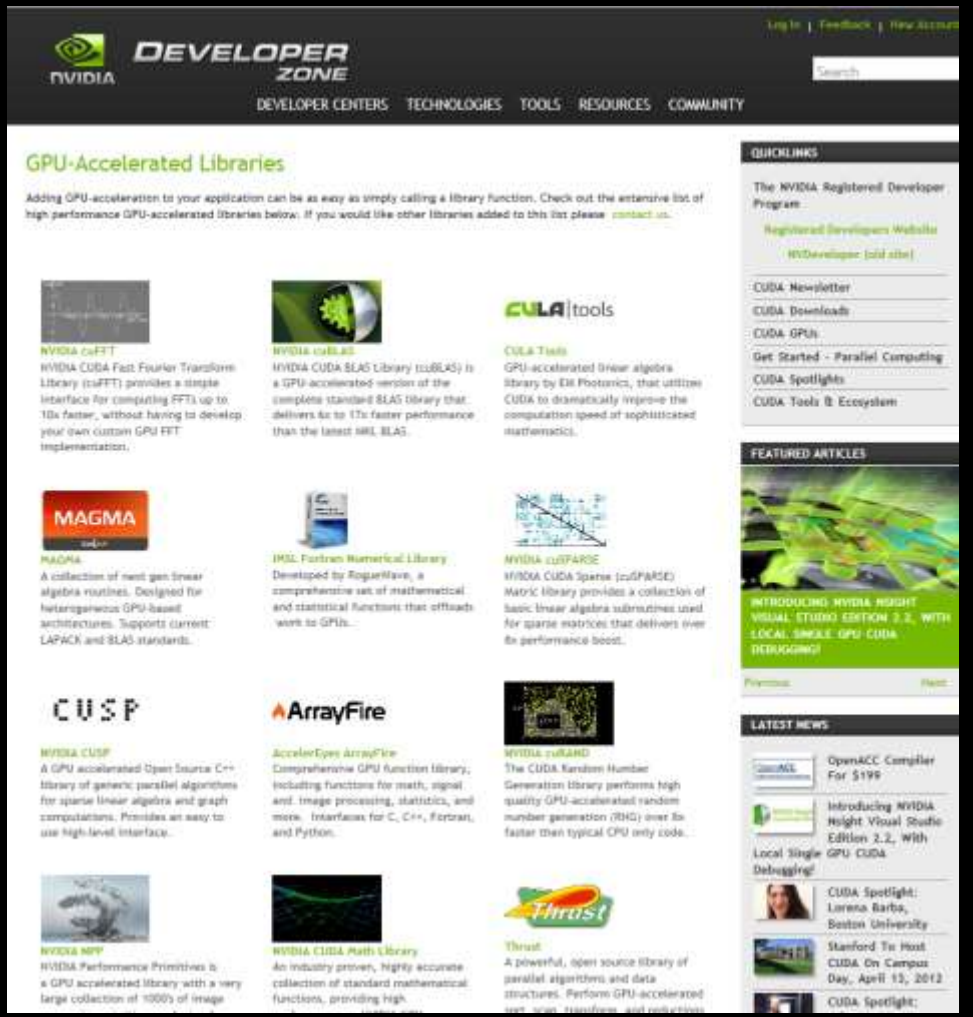
Thrust

C++ STL Features for CUDA

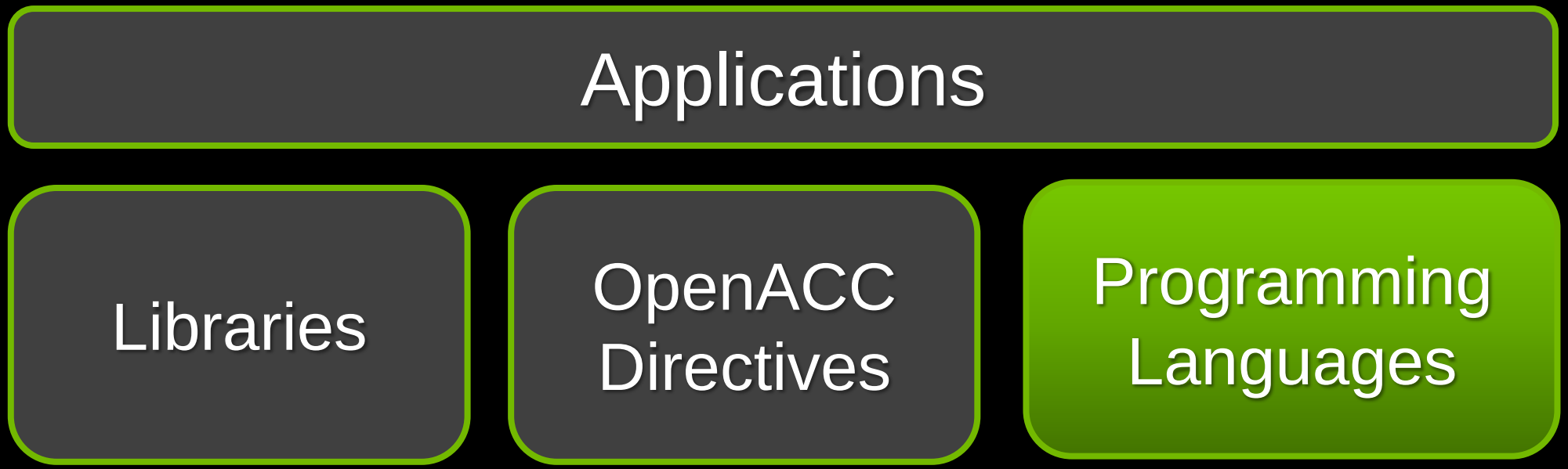
Explore the CUDA (Libraries) Ecosystem

- **CUDA Tools and Ecosystem described in detail on NVIDIA Developer Zone:**

  developer.nvidia.com/cuda-tools-ecosystem



3 Ways to Accelerate Applications



"Drop-in" Acceleration

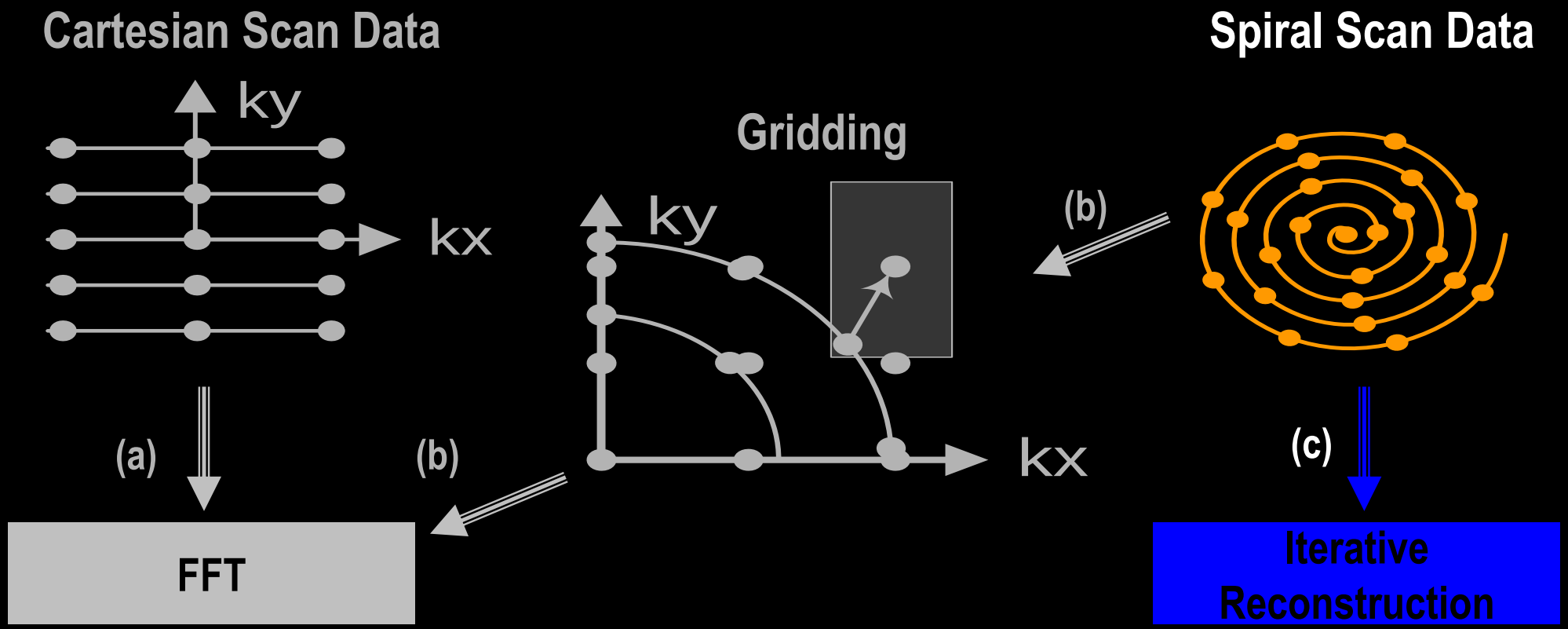
Easily Accelerate Applications

Maximum Flexibility

GPU Programming Languages

Numerical analytics	MATLAB, Mathematica, LabVIEW
Fortran	OpenACC, CUDA Fortran
C	OpenACC, CUDA C
C++	Thrust, CUDA C++
Python	PyCUDA, Copperhead, NumbaPro (Continuum Analytics)
C#	GPU.NET, Hybridizer(AltiMesh)

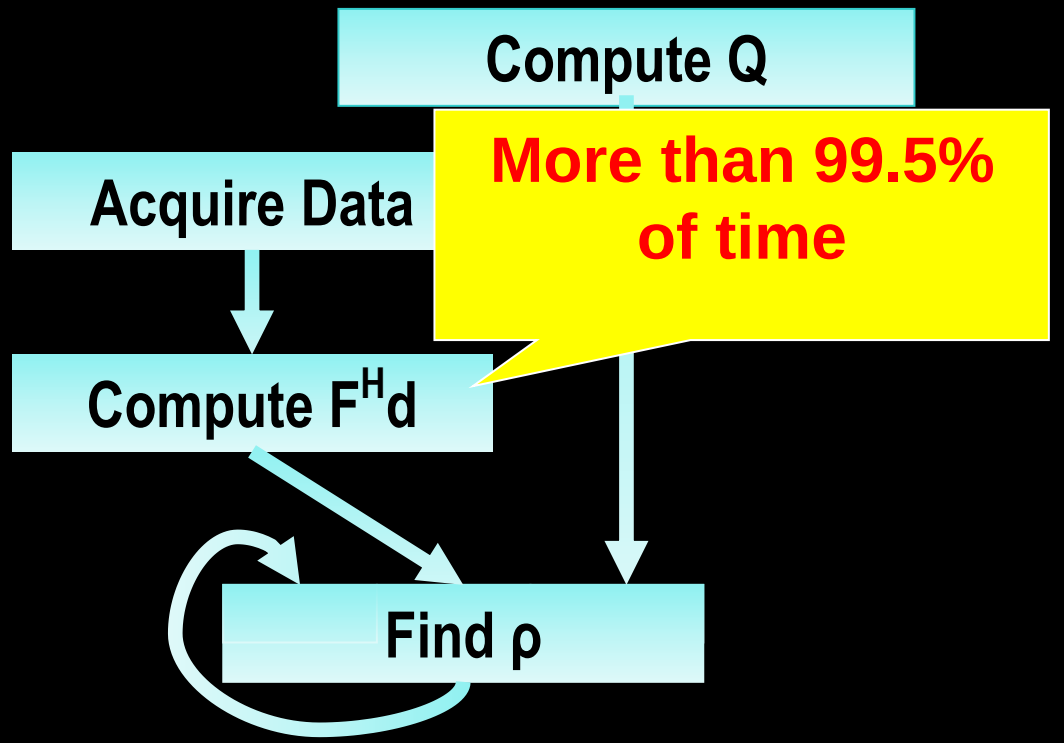
Advanced MRI Reconstruction



Spiral scan data + Iterative reconstruction
Reconstruction requires a lot of computation

Advanced MRI Reconstruction

$$(F^H F + \lambda W^H W)\rho = F^H d$$



Haldar, et al, “Anatomically-constrained reconstruction from noisy data,” MR in Medicine.

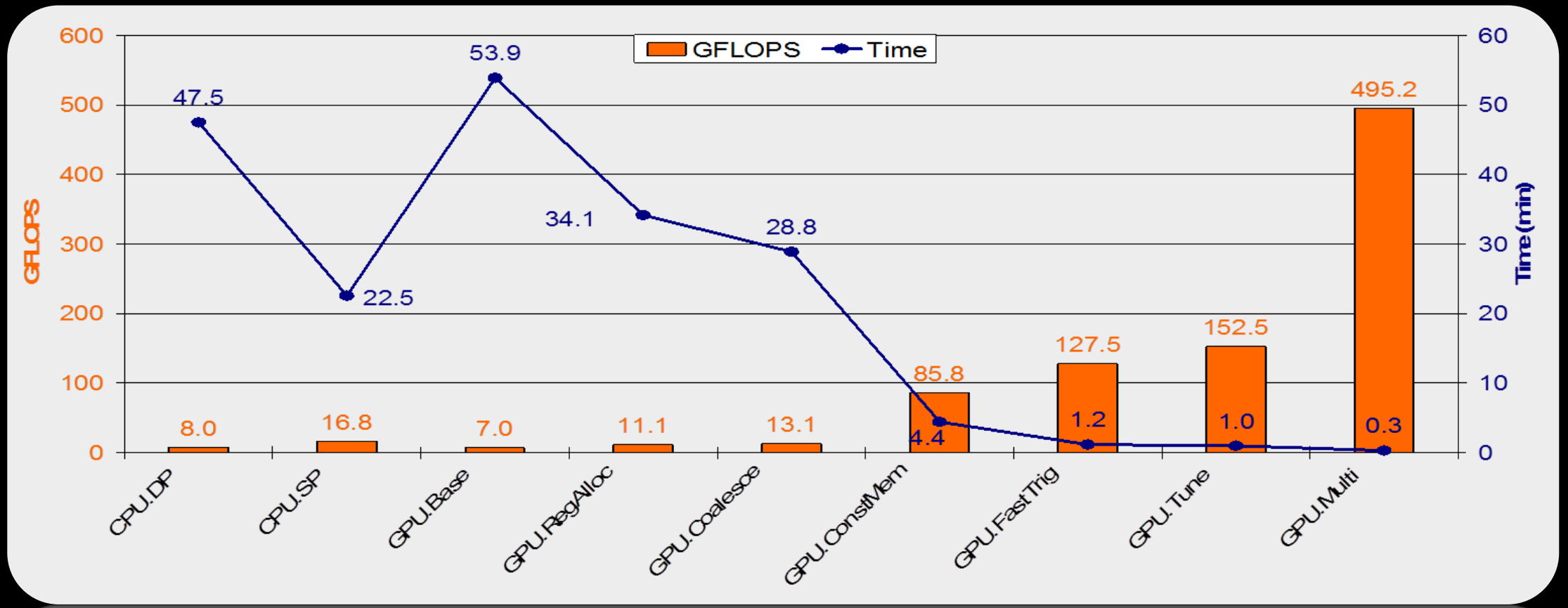
Code

CPU

```
for (p = 0; p < numP; p++) {
  for (d = 0; d < numD; d++) {
    exp = 2*PI*(kx[d] * x[p] +
                ky[d] * y[p] +
                kz[d] * z[p]);
    cArg = cos(exp);
    sArg = sin(exp);
    rFhD[p] += rRho[d]*cArg -
               iRho[d]*sArg;
    iFhD[p] += iRho[d]*cArg +
               rRho[d]*sArg;
  }
}
```

GPU

```
__global__ void
cmpFhD(float* gx, gy, gz, grFhD, giFhD) {
  int p = blockIdx.x * THREADS_PB + threadIdx.x;

  // register allocate image-space inputs & outputs
  x = gx[p];  y = gy[p];  z = gz[p];
  rFhD = grFhD[p];  iFhD = giFhD[p];

  for (int d = 0; d < SCAN_PTS_PER_TILE; d++) {
    // s (scan data) is held in constant memory
    float exp = 2 * PI * (s[d].kx * x +
                          s[d].ky * y +
                          s[d].kz * z);
    cArg = cos(exp);  sArg = sin(exp);
    rFhD += s[d].rRho*cArg - s[d].iRho*sArg;
    iFhD += s[d].iRho*cArg + s[d].rRho*sArg;
  }
  grFhD[p] = rFhD;  giFhD[p] = iFhD;
}
```



S.S. Stone, et al, "Accelerating Advanced MRI Reconstruction using GPUs," ACM Computing Frontier Conference 2008, Italy, May 2008.

Get Started Today

These languages are supported on all CUDA-capable GPUs.

You might already have a CUDA-capable GPU in your laptop or desktop PC!

CUDA C/C++
http://developer.nvidia.com/cuda-toolkit

Thrust C++ Template Library
http://developer.nvidia.com/thrust

CUDA Fortran
http://developer.nvidia.com/cuda-toolkit

PyCUDA (Python)
http://mathema.tician.de/software/pycuda

GPU.NET
http://tidepowerd.com

MATLAB
http://www.mathworks.com/discovery/matlab-gpu.html

Mathematica
http://www.wolfram.com/mathematica/new-in-8/cuda-and-opencl-support/

Easiest Way to Learn CUDA

50k Enrolled

127 Countries

coursera

Heterogeneous Parallel Programming
www.coursera.org

UDACITY

Introduction to Parallel Programming
www.udacity.com

Learn from the Best
- Prof. John Owens – UC Davis
- Dr. David Luebke – NVIDIA Research
- Prof. Wen-mei W. Hwu – U of Illinois

Anywhere, Any Time
- Online
- Worldwide
- Self Paced

It's Free!
- No Tuition
- No Hardware
- No Books

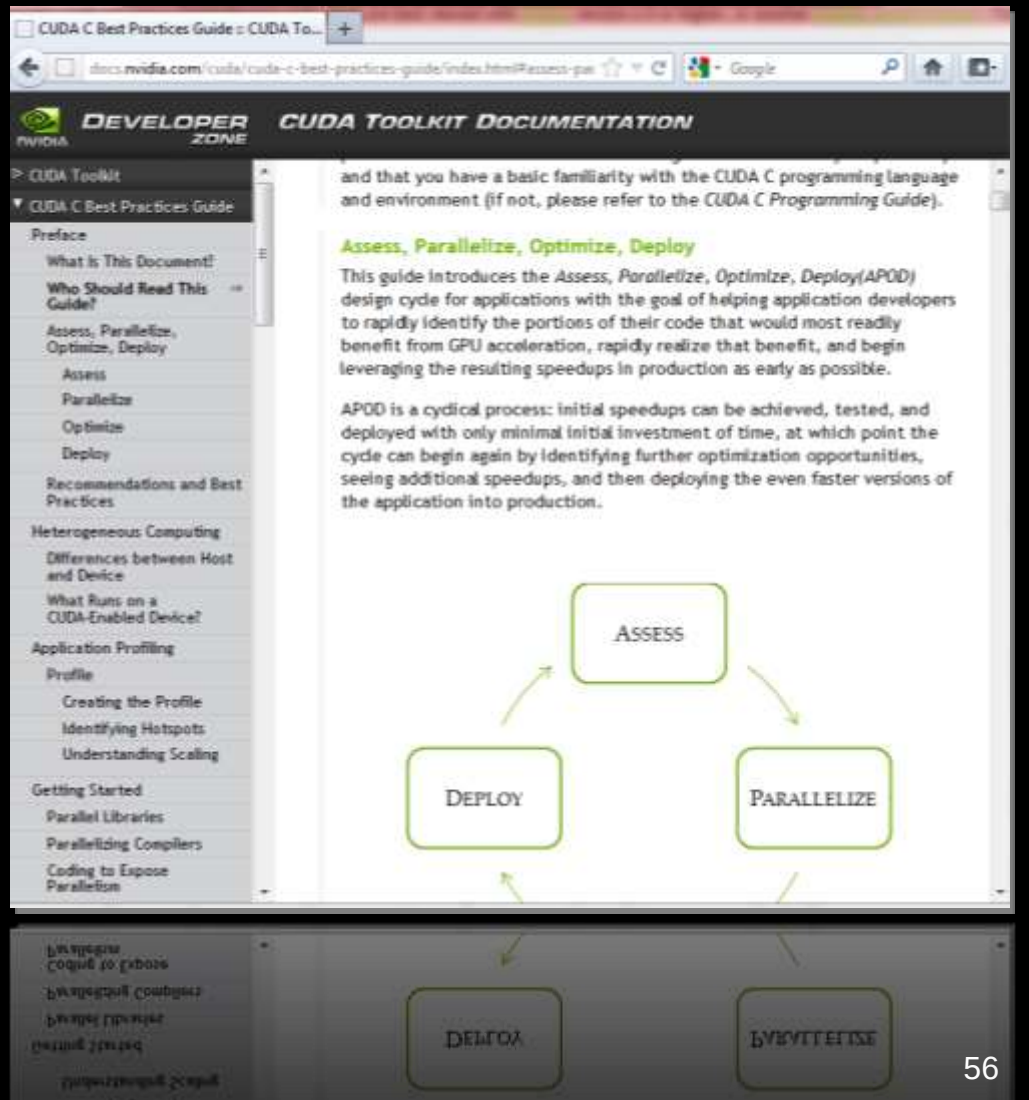
Engage with an Active Community
- Forums and Meetups
- Hands-on Projects

Where to find additional information

CUDA documentation [1]

- **Best Practice Guide** [2]
- **Kepler Tuning Guide** [3]

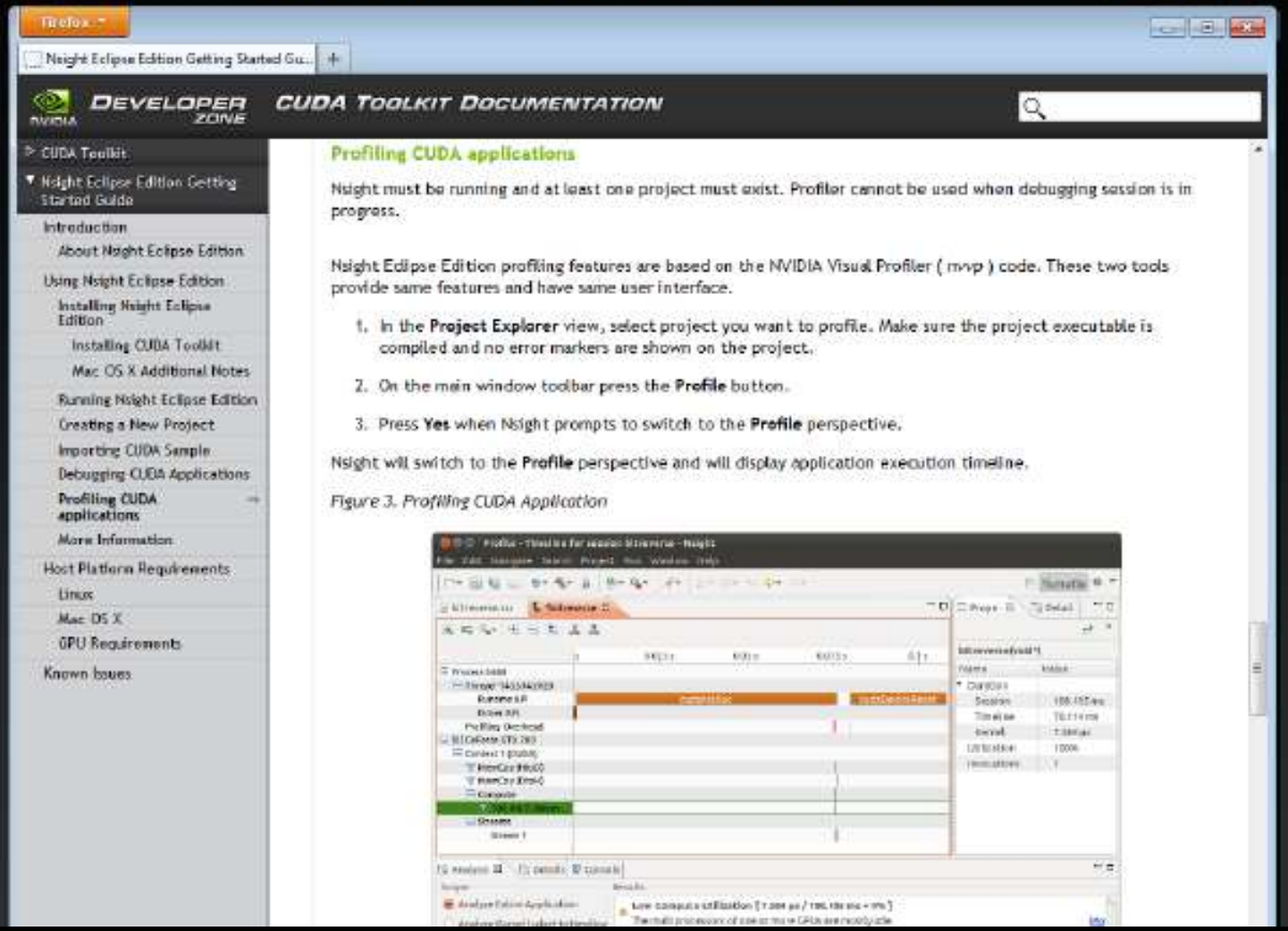
Kepler whitepaper [4]



[1] http://docs.nvidia.com
[2] http://docs.nvidia.com/cuda/cuda-c-best-practices-guide
[3] http://docs.nvidia.com/cuda/kepler-tuning-guide
[4] http://www.nvidia.com/object/nvidia-kepler.html

New Online CUDA Resource Center

All the Latest Information, All in One Place

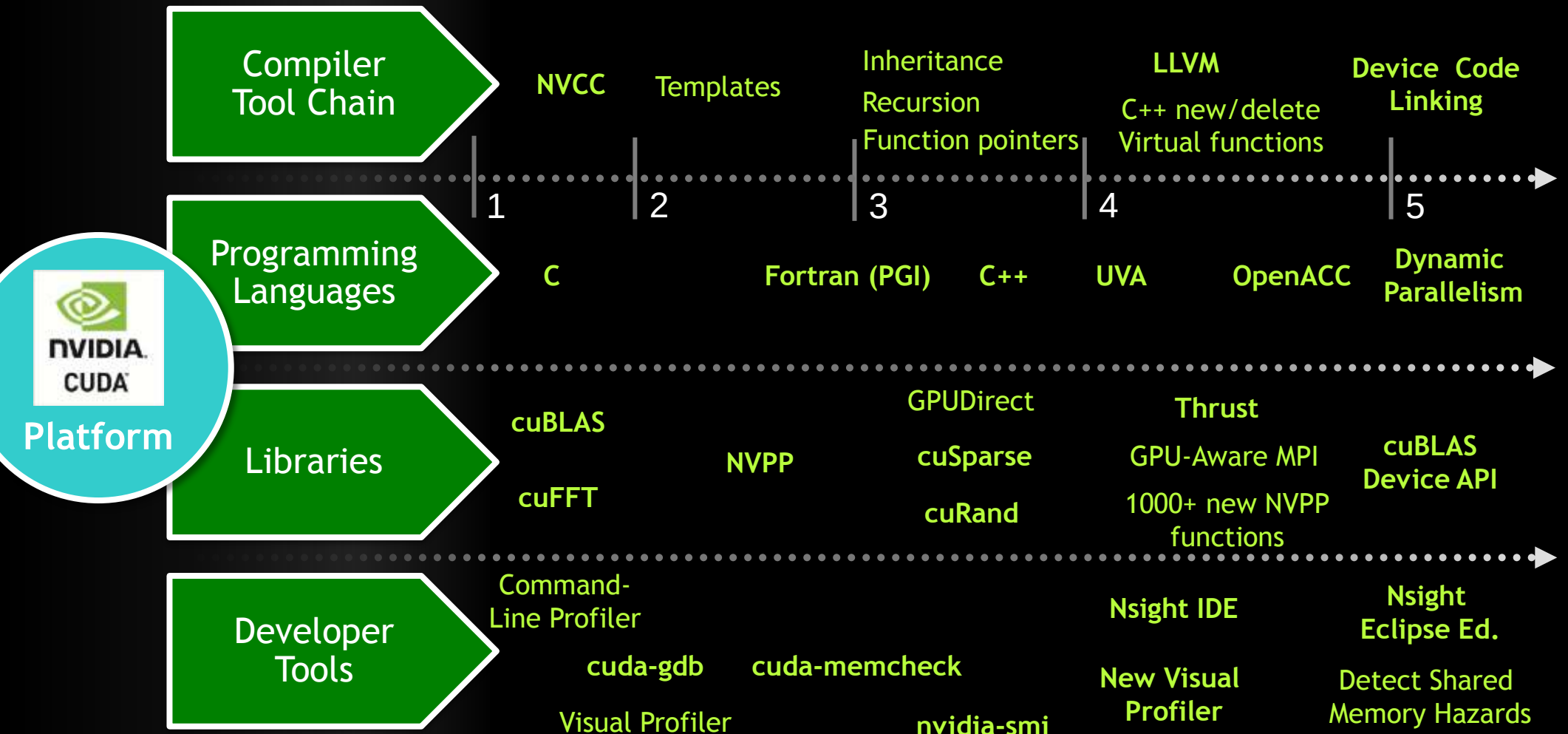


- Programming Guides
- API Reference
- Library Manuals
- Code Samples
- Tools Manuals
- Platform Specs

Over 1600 Files!

http://docs.nvidia.com

CUDA Progress



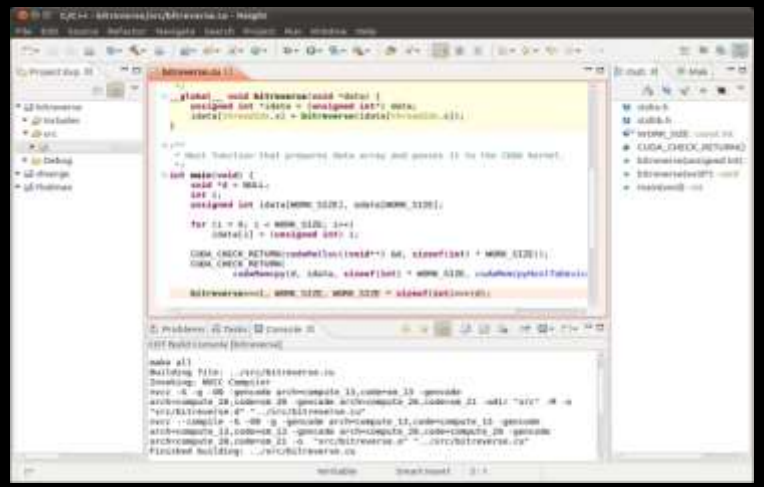
CUDA 5

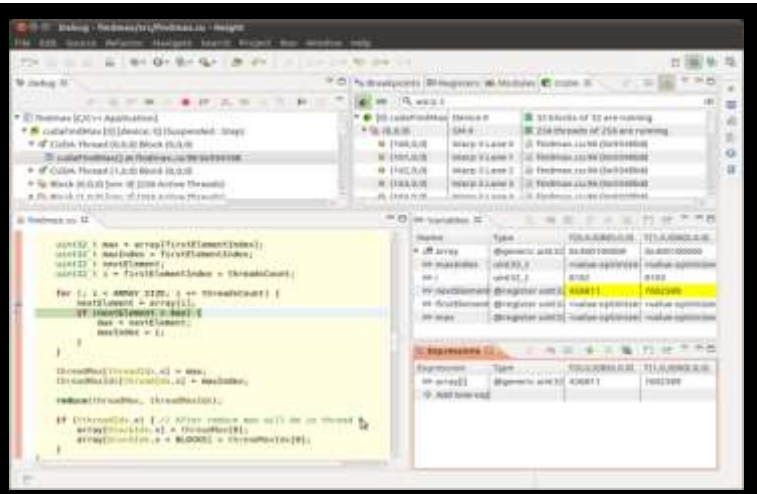
Nsight™ for Linux & Mac

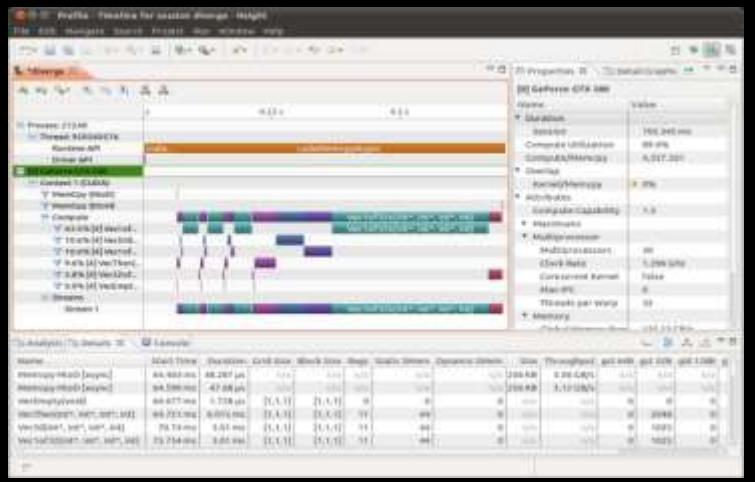
NVIDIA GPUDirect™

Library Object Linking

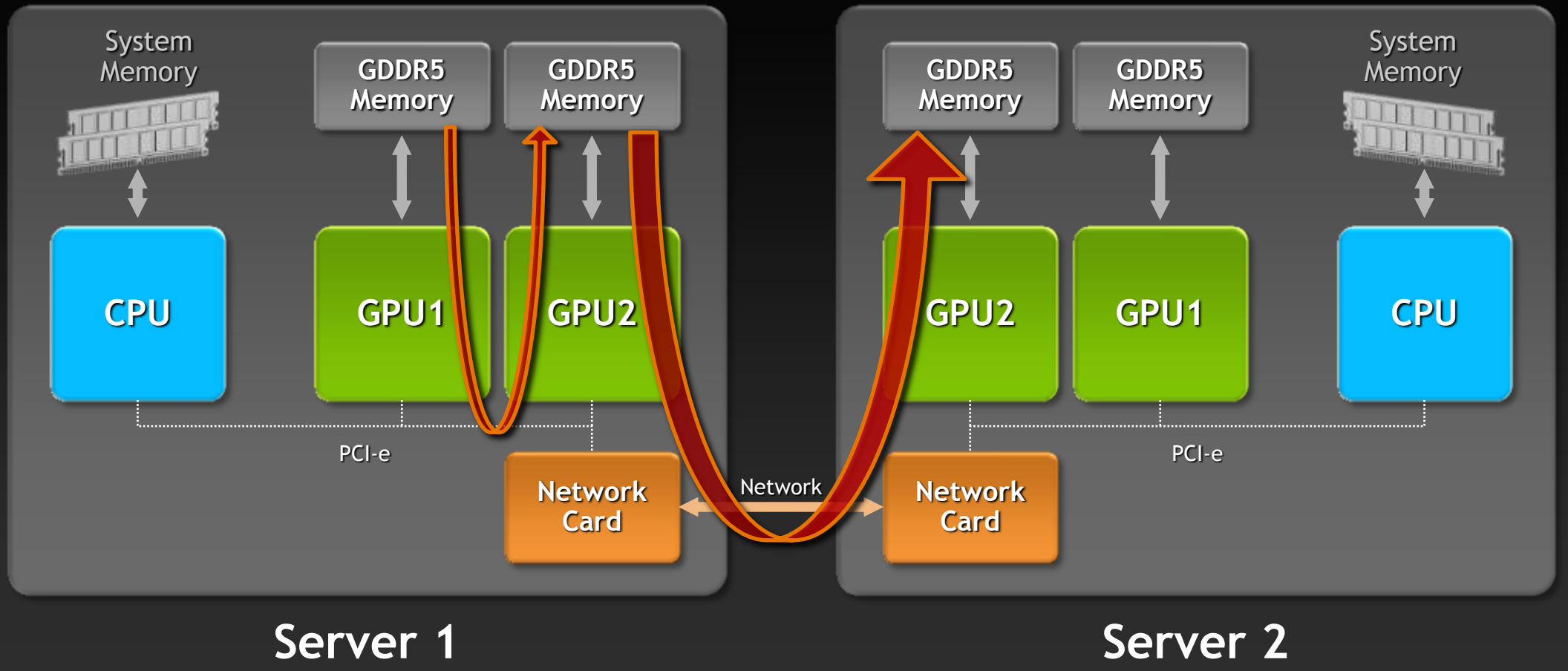
NVIDIA Nsight™ for Linux & Mac (and Windows of course)



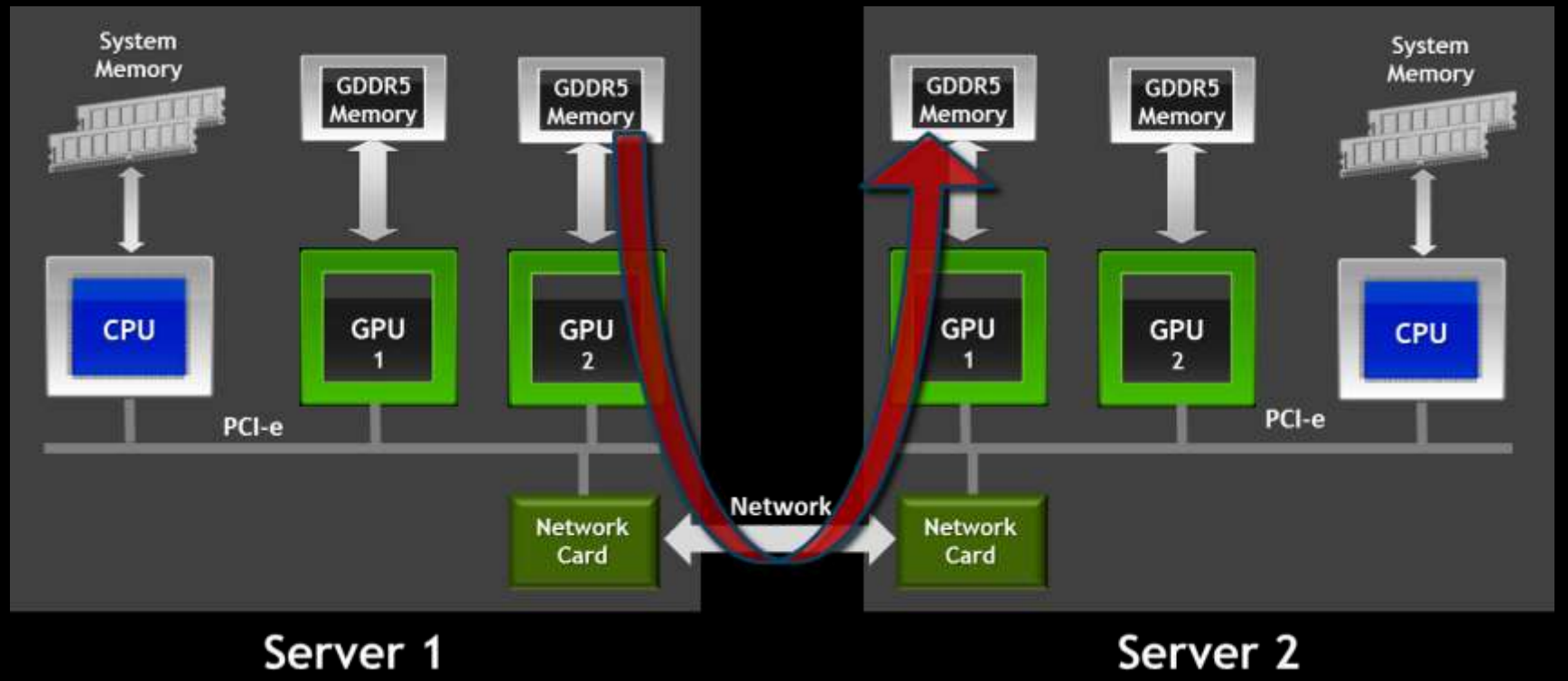




Kepler Enables Full NVIDIA GPUDirect™



GPUDirect enables GPU-aware MPI



GPU-GPU transfer across NIC
Without CPU participation

Unified Virtual addresses allows
to detect location of buffer pointed to

GPUDirect enables GPU-aware MPI

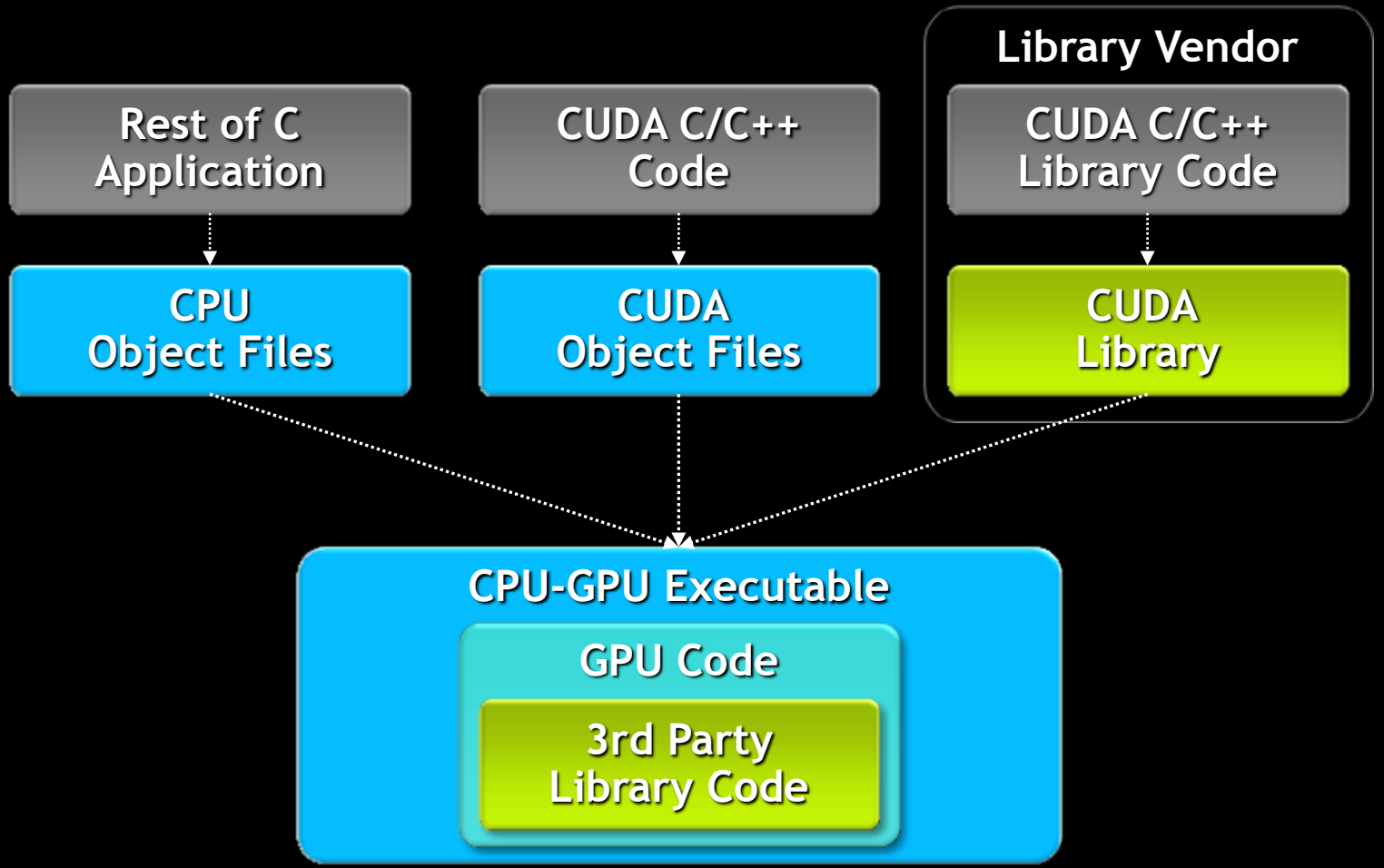
```
cudaMemcpy(s_buf_h,s_buf_d,size,cudaMemcpyDeviceToHost);
MPI_Send(s_buf_h,size,MPI_CHAR,1,100,MPI_COMM_WORLD);

MPI_Recv(r_buf_h,size,MPI_CHAR,1,100,MPI_COMM_WORLD);
cudaMemcpy(r_buf_h,r_buf_d,size,cudaMemcpyHostToDevice);
```

Simplifies to

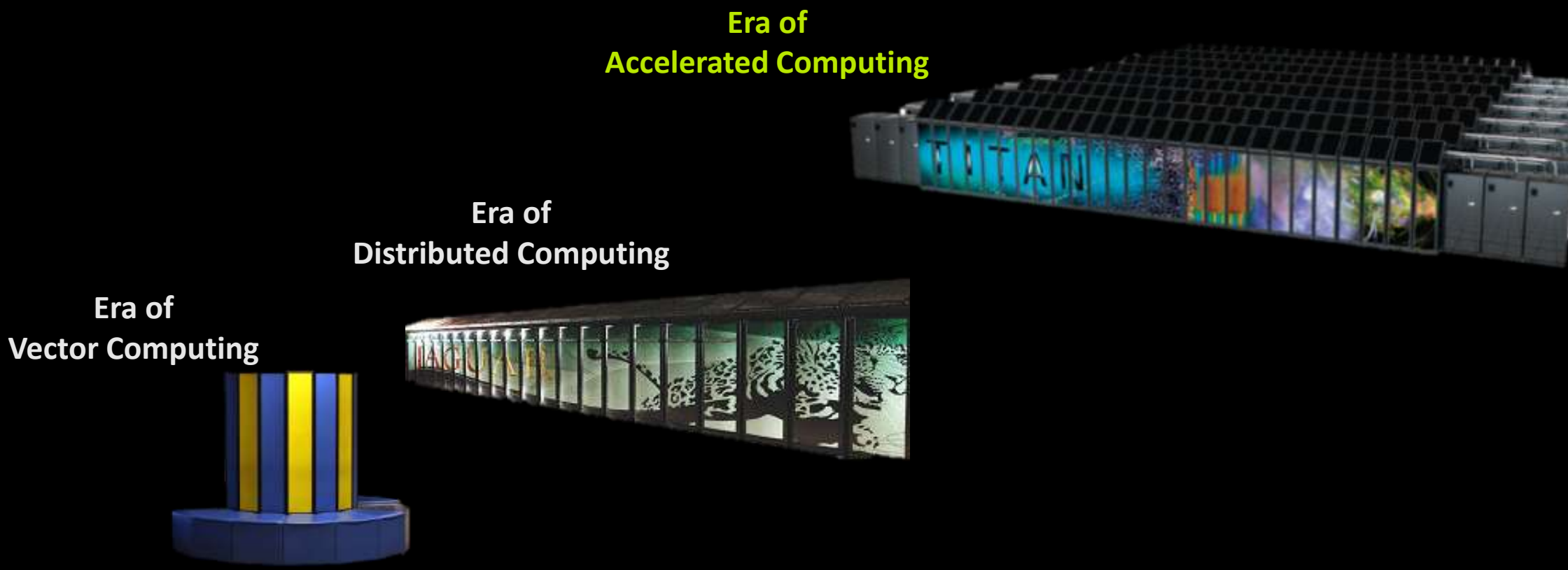
```
MPI_Send(s_buf,size,MPI_CHAR,1,100,MPI_COMM_WORLD);
MPI_Recv(r_buf,size,MPI_CHAR,1,100,MPI_COMM_WORLD);
```

(for CPU and GPU buffers)

3rd Party GPU Library Object Linking

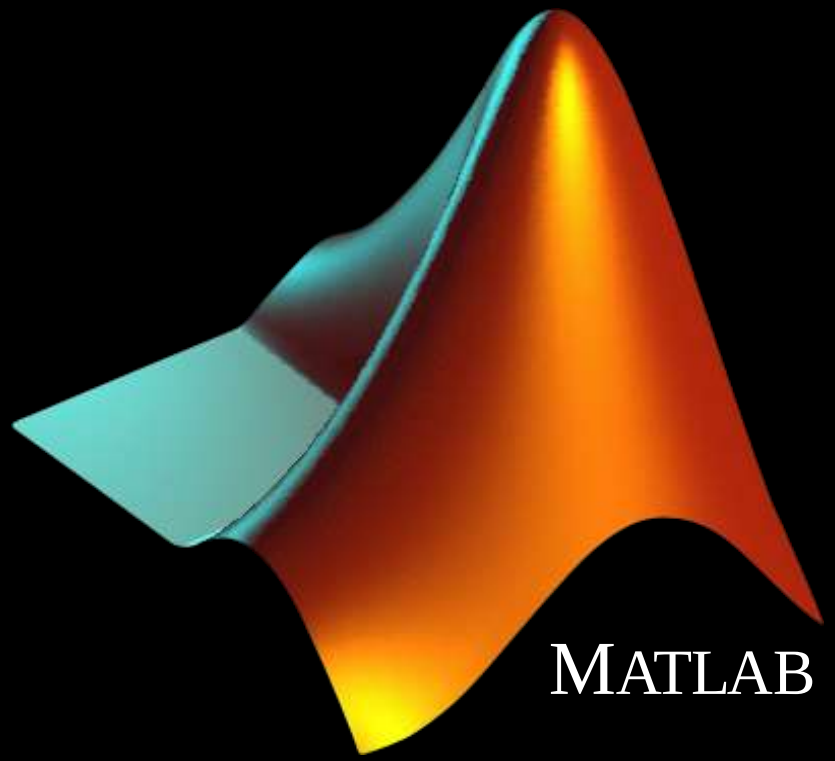


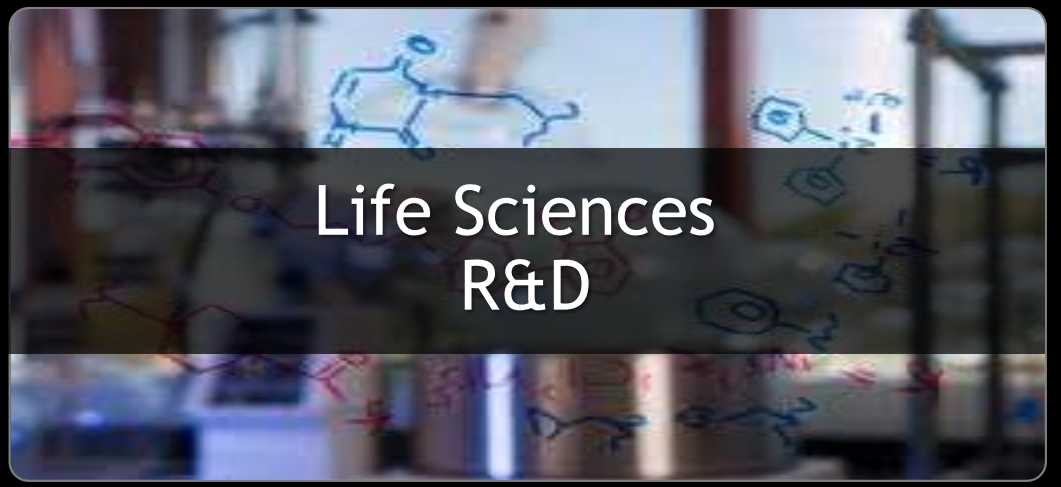
The Era of Accelerated Computing is Here

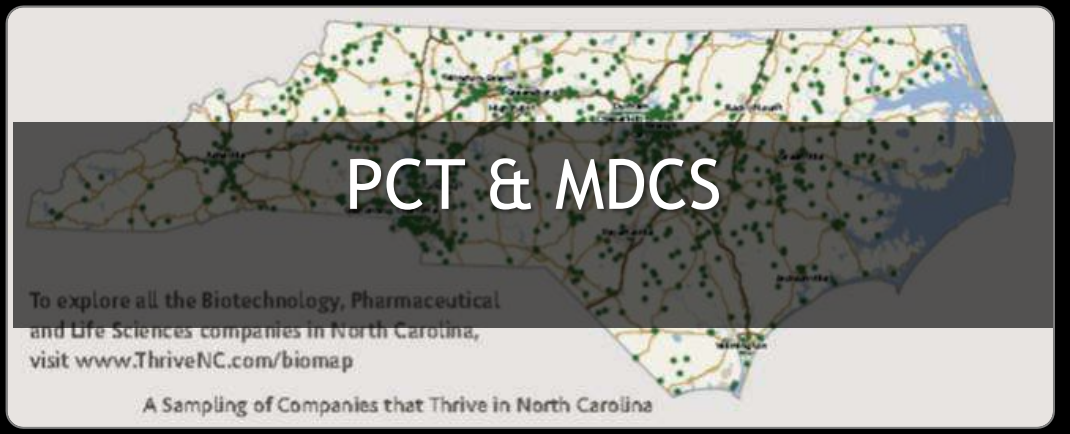


Making a Difference

MATLAB

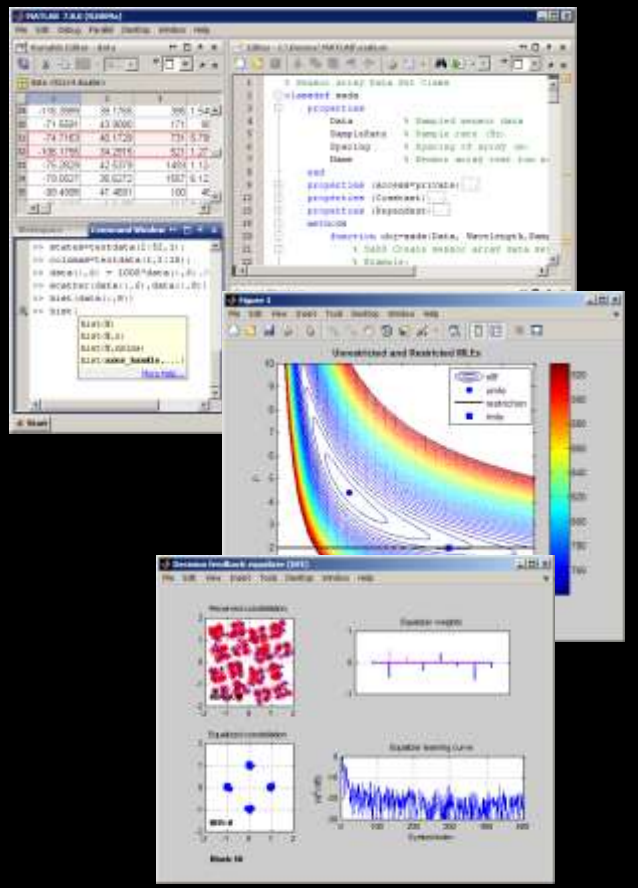

MATLAB


Life Sciences
R&D


PCT & MDCS
To explore all the Biotechnology, Pharmaceutical
and Life Sciences companies in North Carolina,
visit www.ThriveNC.com/biomap
A Sampling of Companies that Thrive in North Carolina

MATLAB Parallel Computing Toolbox

Industry standard high-level language for algorithm development & data analysis



GPU Value

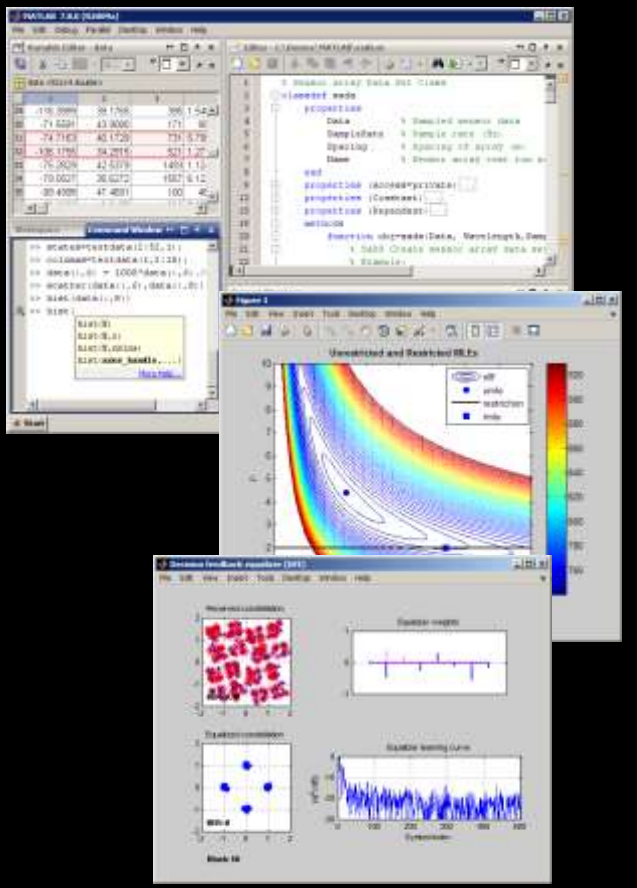
- Allows practical analysis of large datasets for the first time
- Scales from GPU workstations (*Parallel Computing Toolbox*) up to GPU clusters (*MATLAB Distributed Computing Server*)
- Significant acceleration for spectral analysis, linear algebra, and stochastic simulations, etc.

Highlights

- GPU accelerated native MATLAB operations
- Integration with user CUDA kernels in MATLAB
- MATLAB Compiler support (GPU acceleration without MATLAB)

MATLAB Parallel Computing

On-ramp to GPU Computing

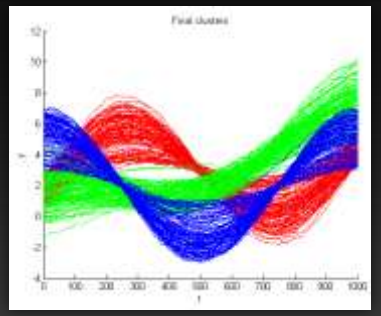


MATLAB R2012b

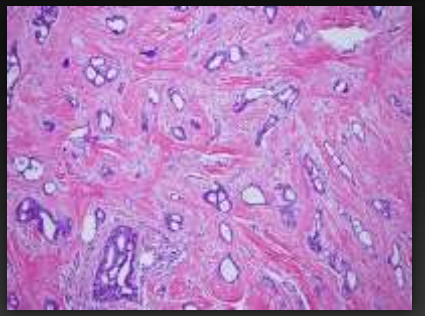
- Over 200 of the most popular MATLAB functions on GPUs
  Including:
  - Random number generation
  - FFT
  - Matrix multiplications
  - Solvers
  - Convolutions
  - Min/max
  - SVD
  - Cholesky and LU factorization
- MATLAB Compiler support (GPU acceleration without MATLAB)
- GPU features in Communications Systems Toolbox
- Performance enhancements

GPU-Accelerated MATLAB Results

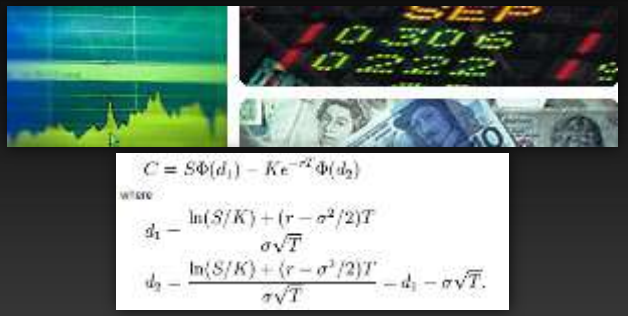
MathWorks



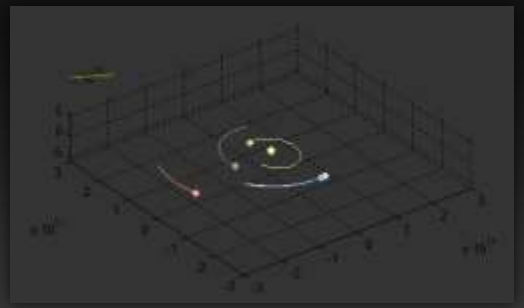
10x speedup in data clustering via K-means clustering algorithm



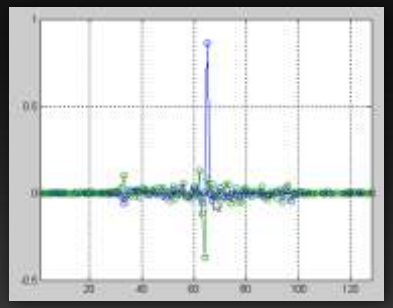
14x speedup in template matching routine (part of cancer cell image analysis)



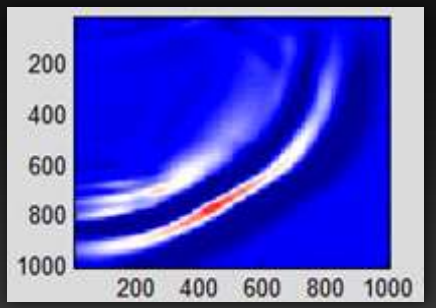
3x speedup in estimating 7.6 million contract prices using Black-Scholes model



17x speedup in simulating the movement of 3072 celestial objects

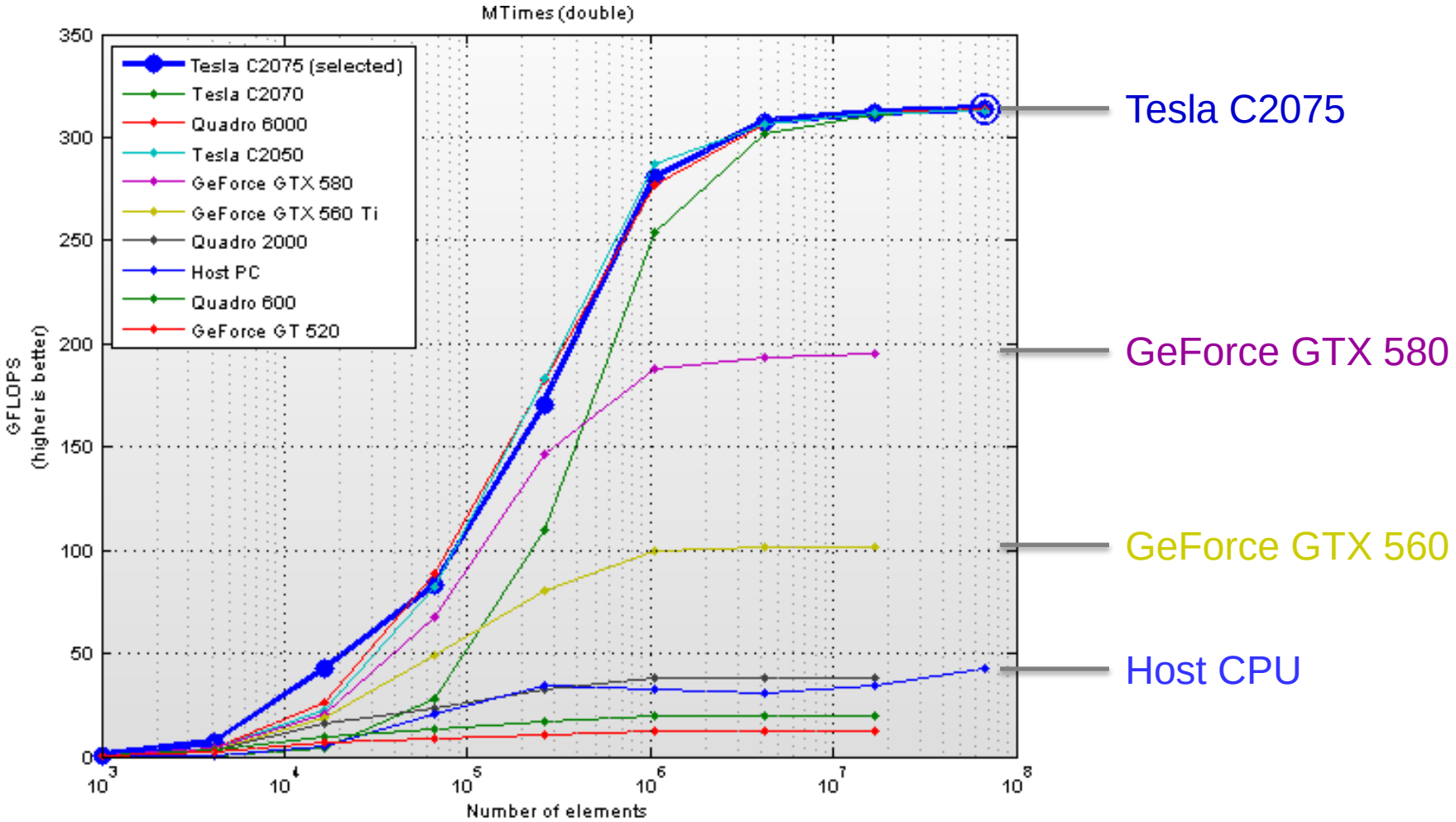


4x speedup in adaptive filtering routine (part of acoustic tracking algorithm)



4x speedup in wave equation solving (part of seismic data processing algorithm)

GPU Value in MATLAB - Bigger is Better



Double precision MTimes performance as measured by GPUBench – available from MATLAB Central File Exchange