GPU TECHNOLOGY CONFERENCE

Languages, APIs and Development Tools for GPU Computing

Will Ramey | Sr. Product Manager for GPU Computing

San Jose Convention Center, CA | September 20–23, 2010

PRESENTED BY NVIDIA.

“GPU Computing”

- Using all processors in the system for the things they are best at doing
  - Evolution of CPUs makes them good at sequential, serial tasks
  - Evolution of GPUs makes them good at parallel processing

GPU Computing Ecosystem

Libraries

$$\oint \mathbf{E} \cdot d\mathbf{A} = \frac{q_{enc}}{\varepsilon_0}$$

$$\oint \mathbf{B} \cdot d\mathbf{A} = 0$$

$$\oint \mathbf{E} \cdot d\mathbf{s} = -\frac{d\Phi_B}{dt}$$

$$\oint \mathbf{B} \cdot d\mathbf{s} = \mu_0 \varepsilon_0 \frac{d\Phi_E}{dt} + \mu_0 i_{enc}$$

Mathematical Packages

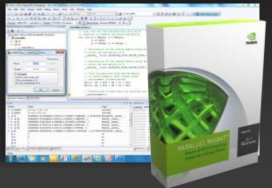
Research & Education

Consultants, Training & Certification

DESIGNED FOR NVIDIA® CUDA™

Integrated Development Environment

Parallel Nsight for MS Visual Studio



Microsoft® Visual Studio PARTNER

Tools & Partners

TOTALVIEW TECHNOLOGIES
A Rogue Wave Software Company

allinea ddt
the distributed debugging tool

Languages & API's

CUDA C/C++

Microsoft® DirectX 11

PGI WORKSTATION
Fortran
Compilers & Tools to Harness the Power of Multiple Processors

OpenCL

All Major Platforms

CUDA - NVIDIA's Architecture for GPU Computing

Broad Adoption

- Over 250M installed CUDA-enabled GPUs
- Over 650k CUDA Toolkit downloads in last 2 Yrs
- Windows, Linux and MacOS Platforms supported
- GPU Computing spans HPC to Consumer
- 350+ Universities teaching GPU Computing on the CUDA Architecture

GPU Computing Applications

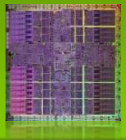
CUDA C/C++
- Over 100k developers
- Running in Production since 2008
- SDK + Libs + Visual Profiler and Debugger

OpenCL
- Commercial OpenCL Conformant Driver
- Public Availability across all CUDA Architecture GPU's
- SDK + Visual Profiler

Direct Compute
- Microsoft API for GPU Computing
- Supports all CUDA-Architecture GPUs (DX10 and DX11)

Fortran
- PGI Accelerator
- PGI CUDA Fortran

Python, Java, .NET, …
- PyCUDA
- GPU.NET
- jCUDA



NVIDIA GPU
with the CUDA Parallel Computing Architecture

GPU Computing Software Stack

Your GPU Computing Application

Application Acceleration Engines (AXEs)
Middleware, Modules & Plug-ins

Foundation Libraries
Low-level Functional Libraries

Development Environment
Languages, Device APIs, Compilers, Debuggers, Profilers, etc.

CUDA Architecture

Languages & APIs

Many Different Approaches

- Application level integration
- High level, *implicit* parallel languages
- Abstraction layers & API wrappers
- High level, *explicit* language integration
- Low level device APIs

GPUs for MathWorks Parallel Computing Toolbox™ and Distributed Computing Server™

Workstation

Compute Cluster

MATLAB Parallel Computing Toolbox (PCT)

- PCT enables high performance through parallel computing on workstations
- **NVIDIA GPU acceleration now available**

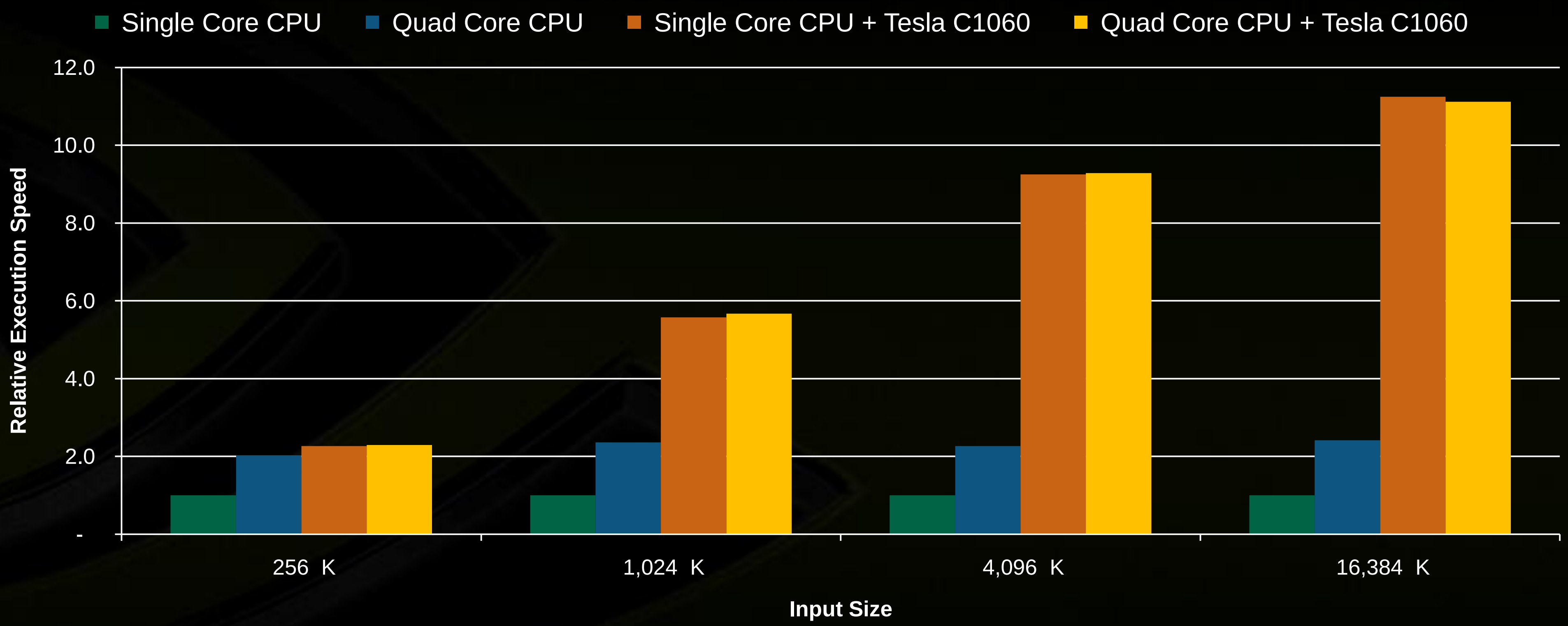
MATLAB Distributed Computing Server (MDCS)

- MDCS allows a MATLAB PCT application to be submitted and run on a compute cluster
- **NVIDIA GPU acceleration now available**

MATLAB Performance with Tesla

Relative Performance, Black-Scholes Demo

Compared to Single Core CPU Baseline



Core 2 Quad Q6600 2.4 GHz, 6 GB RAM, Windows 7 64-bit, Tesla C1060, single precision operations

PGI Accelerator Compilers

```
      SUBROUTINE SAXPY (A,X,Y,N)
      INTEGER N
      REAL A,X(N),Y(N)
!$ACC REGION
      DO I = 1, N
         X(I) = A*X(I) + Y(I)
      ENDDO
!$ACC END REGION
      END
```

compile

Host x64 asm File

```
saxpy_:
        …
        movl    (%rbx), %eax
        movl    %eax, -4(%rbp)
        call    __pgi_cu_init
        . . .
        call    __pgi_cu_function
        …
        call    __pgi_cu_alloc
        …
        call    __pgi_cu_upload
        …
        call    __pgi_cu_call
        …
        call    __pgi_cu_download
        …
```

+

Auto-generated GPU code

```
typedef struct dim3{ unsigned int x,y,z; }dim3;
typedef struct uint3{ unsigned int x,y,z; }uint3;
extern uint3 const threadIdx, blockIdx;
extern dim3 const blockDim, gridDim;
static __attribute__((__global__)) void
pgicuda(
    __attribute__((__shared__)) int tc,
    __attribute__((__shared__)) int i1,
    __attribute__((__shared__)) int i2,
    __attribute__((__shared__)) int _n,
    __attribute__((__shared__)) float* _c,
    __attribute__((__shared__)) float* _b,
    __attribute__((__shared__)) float* _a )
{  int i;  int p1; int _i;
    i = blockIdx.x * 64 + threadIdx.x;
    if( i < tc ){
        _a[i+i2-1] = ((_c[i+i2-1]+_c[i+i2-1])+_b[i+i2-1]);
        _b[i+i2-1] = _c[i+i2];
        _i = (_i+1);
        p1 = (p1-1);
    } }
```

Unified a.out

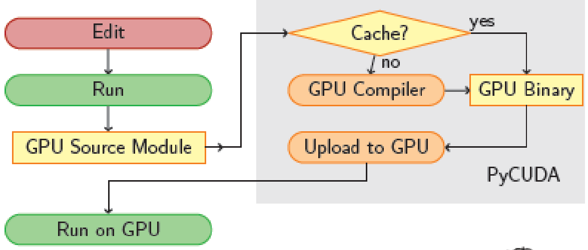
execute

… no change to existing makefiles, scripts, IDEs, programming environment, etc.

Python + CUDA = PyCUDA

- All of CUDA in a modern scripting language
- Full Documentation
- Free, open source (MIT)
- Also: PyOpenCL

- CUDA C Code = Strings
- Generate Code Easily
  - Automated Tuning
- Batteries included: GPU Arrays, RNG, ...
- Integration: `numpy` arrays, Plotting, Optimization, ...



BROWN

http://mathema.tician.de/software/pycuda

CUDA C: C with a few keywords

```
void saxpy_serial(int n, float a, float *x, float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}
// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

Standard C Code

```
__global__ void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n)  y[i] = a*x[i] + y[i];
}
// Invoke parallel SAXPY kernel with 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

CUDA C Code

TidePowerd / GPU.NET

- Write GPU kernels in C#, F#, VB.NET, etc.
- Exposes a minimal API accessible from any .NET-based language
  - Learn a new API instead of a new language
- JIT compilation = *dynamic* language support
- Don’t rewrite your existing code
  - Just give it a “touch-up”

OpenCL

- Cross-vendor open standard
  - Managed by the Khronos Group
- Low-level API for device management and launching kernels
  - Close-to-the-metal programming interface
  - JIT compilation of kernel programs
- C-based language for compute kernels
  - Kernels must be optimized for each processor architecture

http://www.khronos.org/opencl

NVIDIA released the first OpenCL conformant driver for Windows and Linux to thousands of developers in June 2009

DirectCompute

- Microsoft standard for all GPU vendors
  - Released with DirectX® 11 / Windows 7
  - Runs on all 100M+ CUDA-enabled DirectX 10 class GPUs and later
- Low-level API for device management and launching kernels
  - Good integration with DirectX 10 and 11
- Defines HLSL-based language for compute shaders
  - Kernels must be optimized for each processor architecture

Language & APIs for GPU Computing

Approach	Examples
Application Integration	MATLAB, Mathematica, LabVIEW
Implicit Parallel Languages	PGI Accelerator, HMPP
Abstraction Layer/Wrapper	PyCUDA, CUDA.NET, jCUDA
Language Integration	CUDA C/C++, PGI CUDA Fortran
Low-level Device API	CUDA C/C++, DirectCompute, OpenCL

Development Tools

Parallel Nsight for Visual Studio

Integrated development for CPU and GPU

Windows GPU Development for 2010

NVIDIA Parallel Nsight ™ 1.5

nvcc
+
cuda-gdb
+
cuda-memcheck
+
Visual Profiler
+
cudaprof

+

FX Composer
+
Shader Debugger
+
PerfHUD
+
ShaderPerf
+
Platform Analyzer

4 Flexible GPU Development Configurations

Desktop

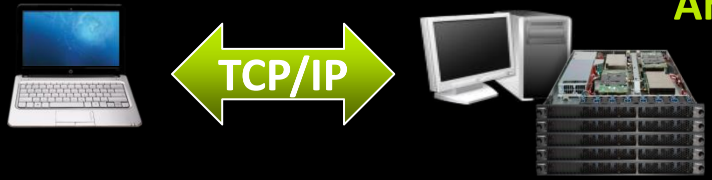
Single machine, Single NVIDIA GPU

Analyzer, Graphics Inspector

NEW **Single machine, Dual NVIDIA GPUs**

Analyzer, Graphics Inspector, Compute Debugger

Networked



Two machines connected over the network

Analyzer, Graphics Inspector, Compute Debugger, Graphics Debugger

Workstation SLI

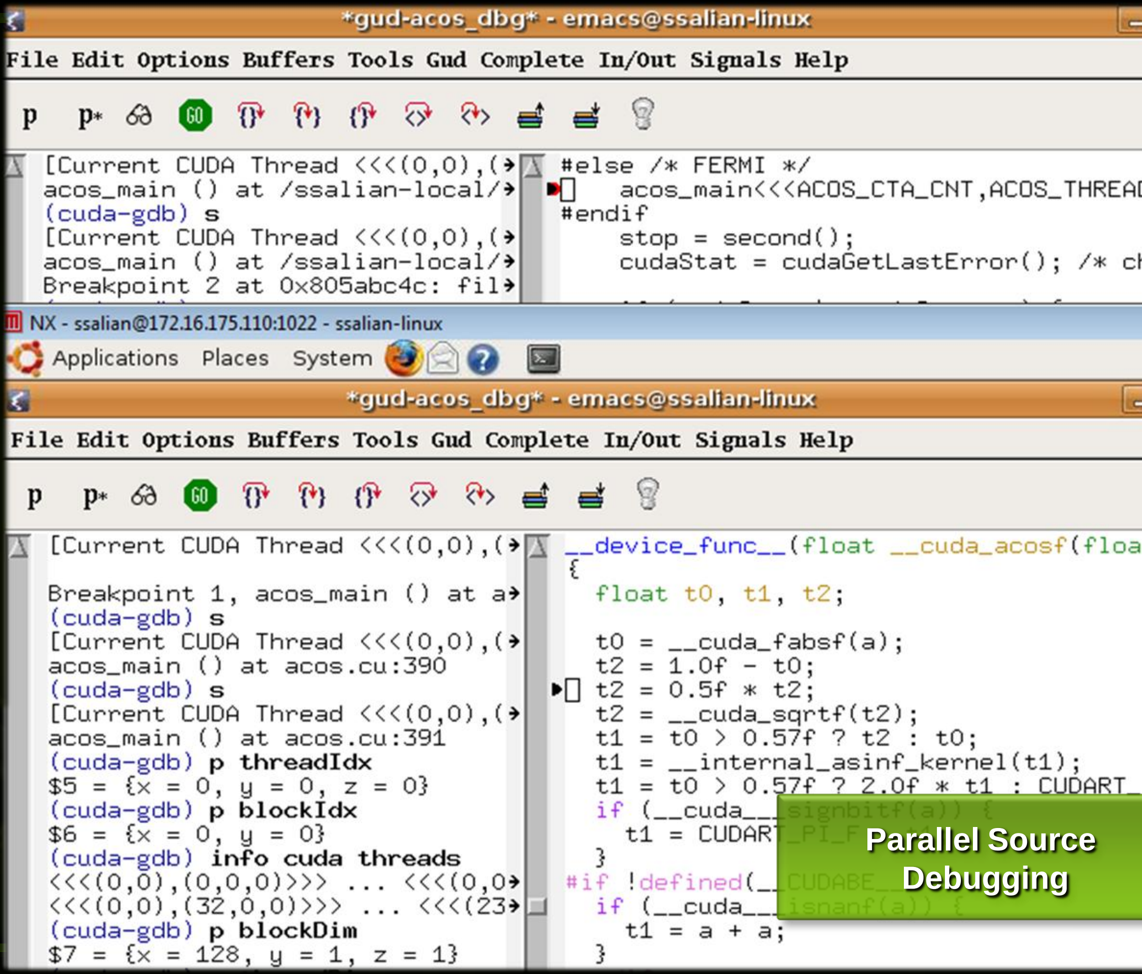
SLI Multi OS workstation with two Quadro GPUs

Analyzer, Graphics Inspector, Compute Debugger, Graphics Debugger

NVIDIA cuda-gdb

CUDA debugging integrated into GDB on Linux

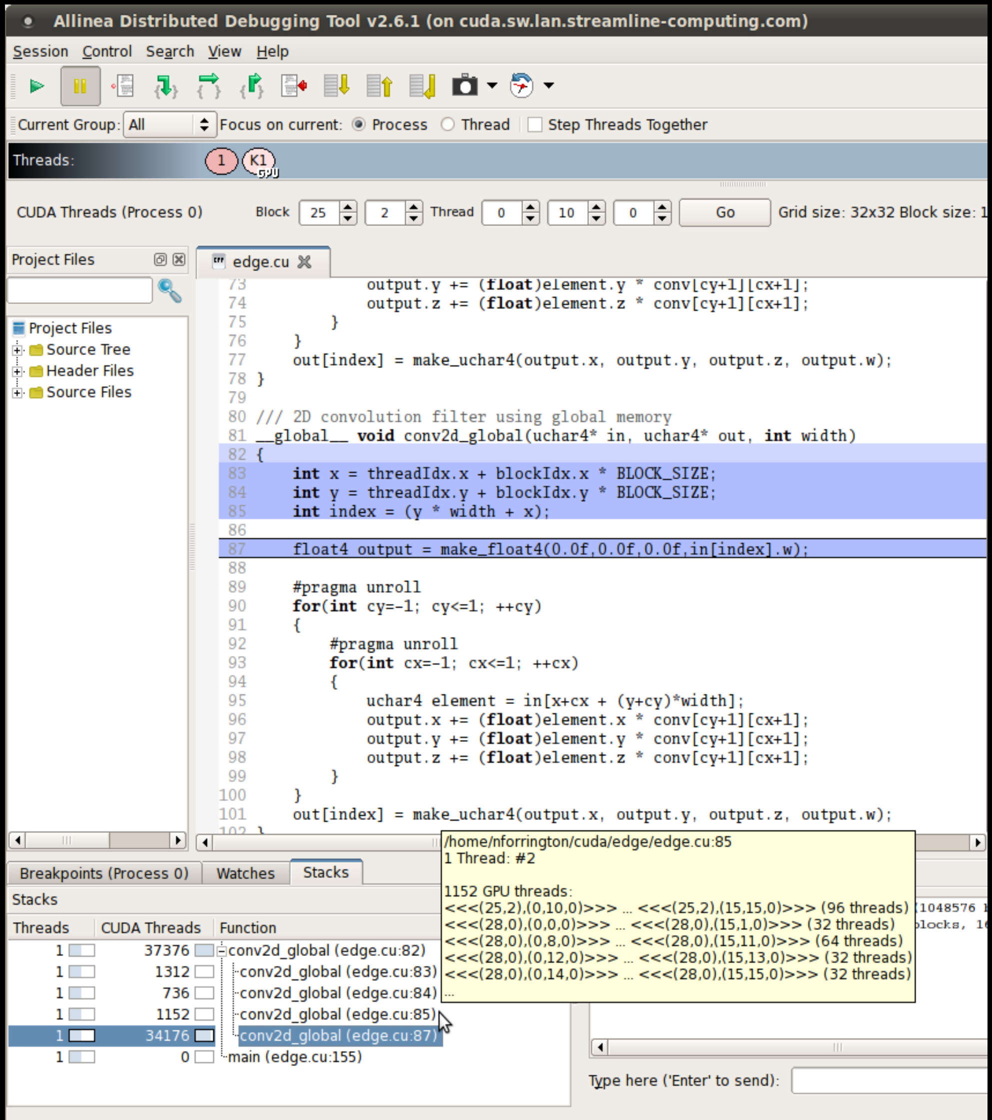
- Supported on 32bit and 64bit systems
- Seamlessly debug both the host/CPU and device/GPU code
- Set breakpoints on any source line or symbol name
- Access and print all CUDA memory allocs, local, global, constant and shared vars



Included in the CUDA Toolkit

GPU TECHNOLOGY CONFERENCE

Allinea DDT debugger



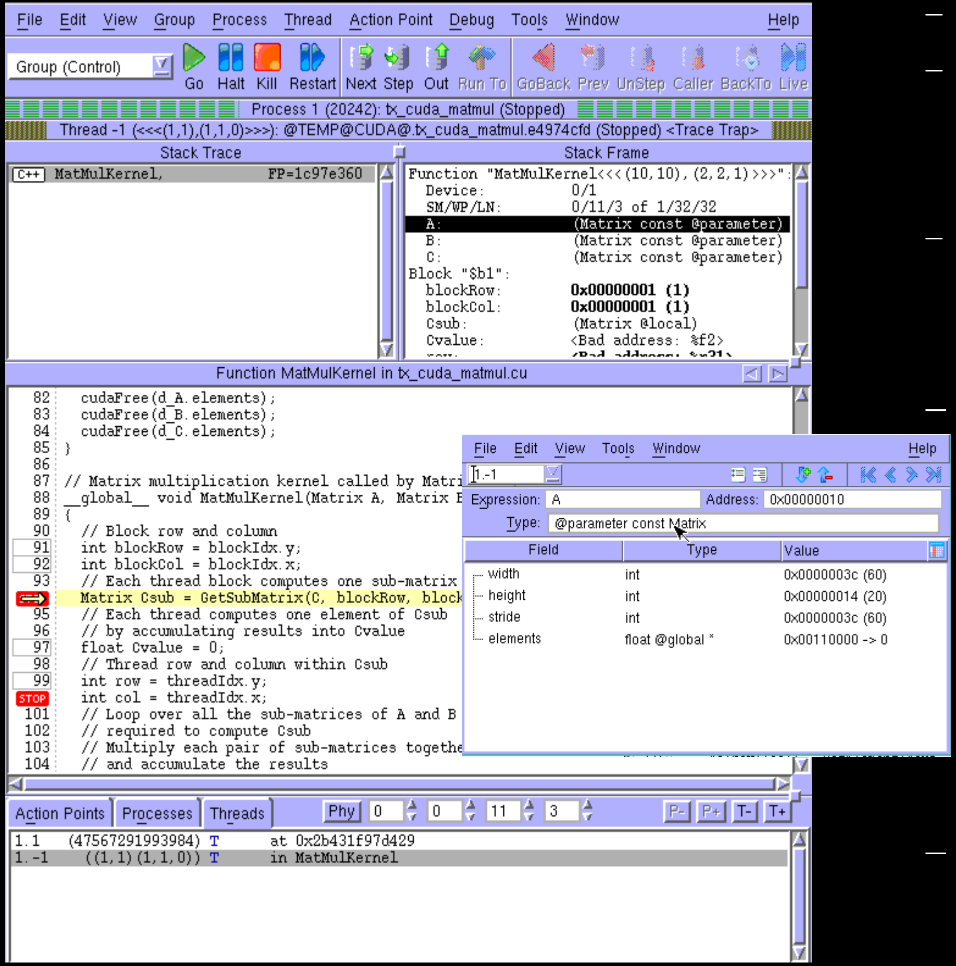
Latest News from Allinea

- CUDA SDK 3.0 with DDT 2.6
  - Released June 2010
  - Fermi and Tesla support
  - cuda-memcheck support for memory errors
  - Combined MPI and CUDA support
  - Stop on kernel launch feature
  - Kernel thread control, evaluation and breakpoints
  - Identify thread counts, ranges and CPU/GPU threads easily
- SDK 3.1 in beta with DDT 2.6.1
- SDK 3.2
  - Coming soon: multiple GPU device support

ROGUE WAVE SOFTWARE

TotalView Debugger

TOTALVIEW

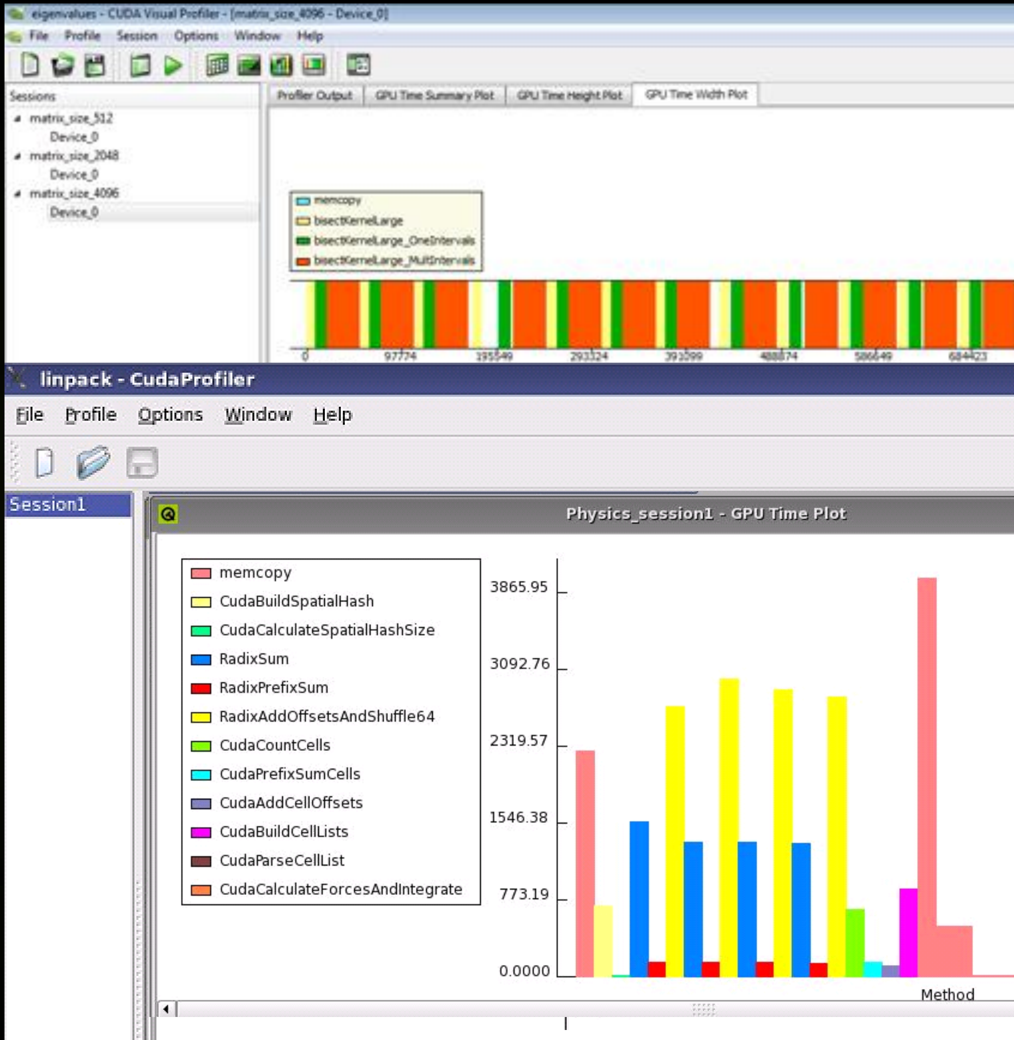


- Latest from TotalView debugger (in Beta)
  - Debugging of application running on the GPU device
  - Full visibility of both Linux threads and GPU device threads
    - Device threads shown as part of the parent Unix process
    - Correctly handle all the differences between the CPU and GPU
  - Fully represent the hierarchical memory
    - Display data at any level (registers, local, block, global or host memory)
    - Making it clear where data resides with type qualification
  - Thread and Block Coordinates
    - Built in runtime variables display threads in a warp, block and thread dimensions and indexes
    - Displayed on the interface in the status bar, thread tab and stack frame
  - Device thread control
    - Warps advance Synchronously
  - Handles CUDA function inlining
    - Step in to or over inlined functions
  - Reports memory access errors
    - CUDA memcheck
  - Can be used with MPI

GPU TECHNOLOGY CONFERENCE

NVIDIA Visual Profiler

- **Analyze GPU HW performance**
  signals, kernel occupancy, instruction throughput, and more
- **Highly configurable**
  tables and graphical views
- **Save/load profiler sessions or export to CSV for later analysis**
- **Compare results visually**
  across multiple sessions to see improvements
- **Windows, Linux and Mac OS X**
  OpenCL support on Windows and Linux



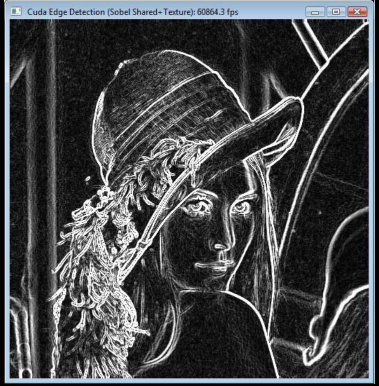
Included in the CUDA Toolkit

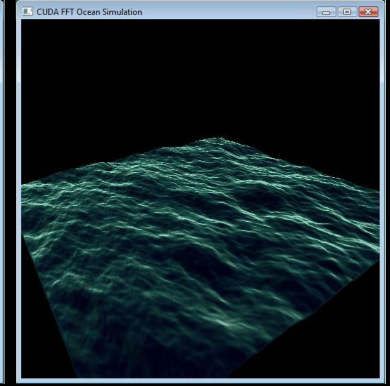
GPU TECHNOLOGY CONFERENCE

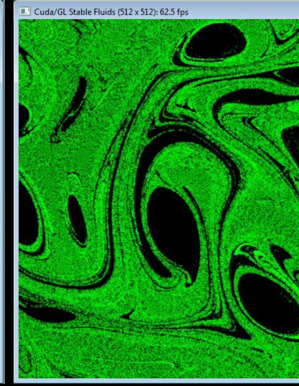
GPU Computing SDK

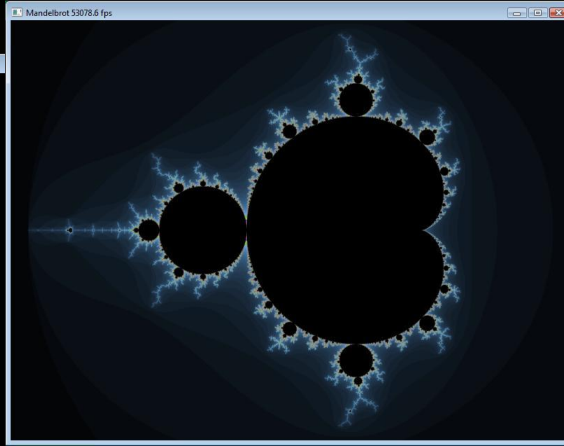
Hundreds of code samples for CUDA C, DirectCompute and OpenCL

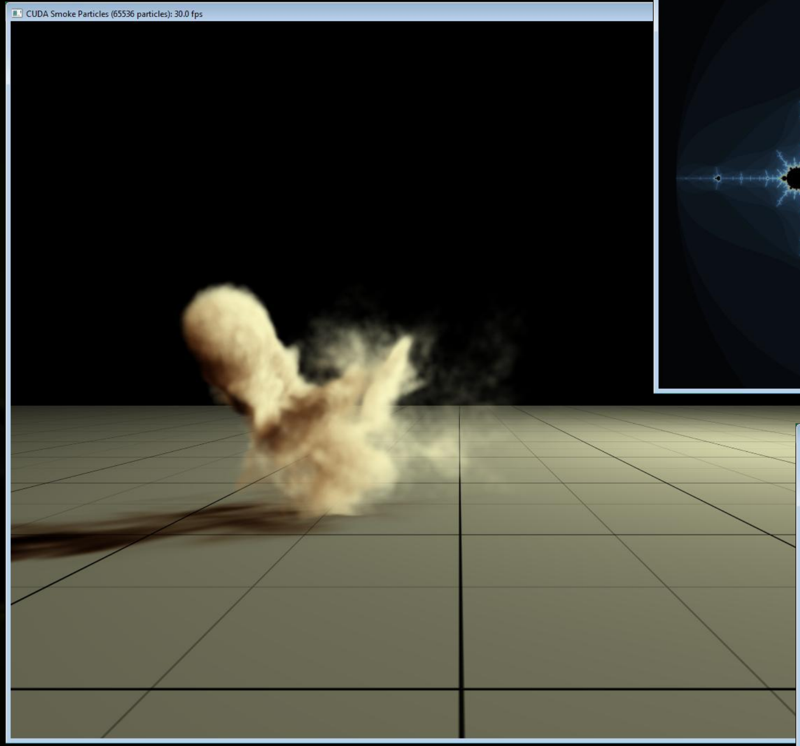
- Finance
- Oil & Gas
- Video/Image Processing
- 3D Volume Rendering
- Particle Simulations
- Fluid Simulations
- Math Functions

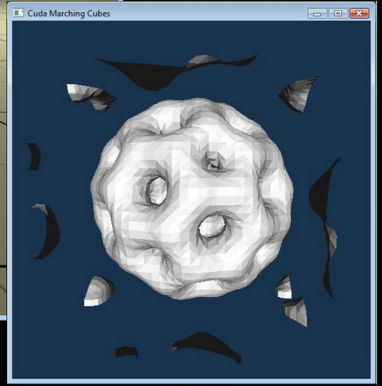


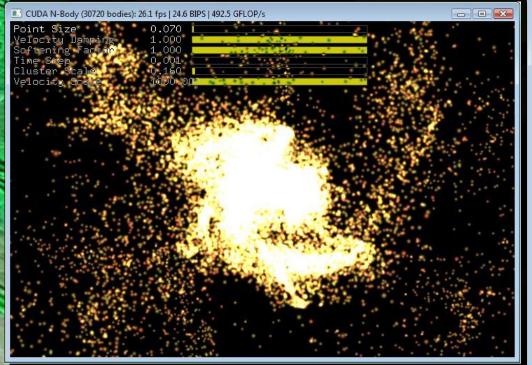


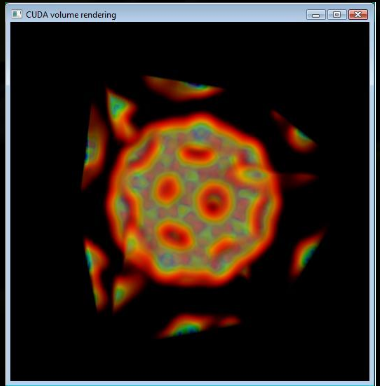












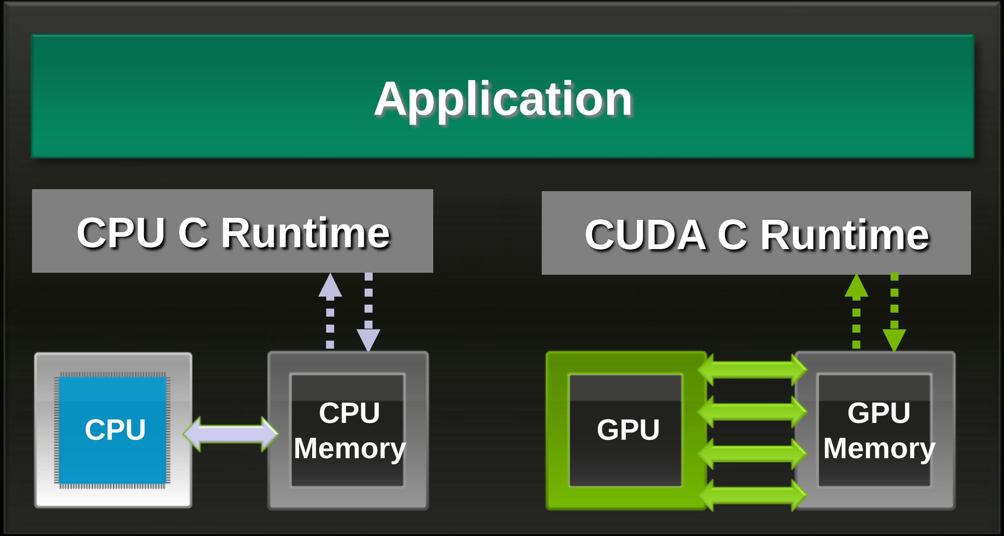
Application Design Patterns

Trivial Application

Design Rules:

- Serial task processing on CPU
- Data Parallel processing on GPU
  - Copy input data to GPU
  - Perform parallel processing
  - Copy results back

- Follow guidance in the CUDA C Best Practices Guide

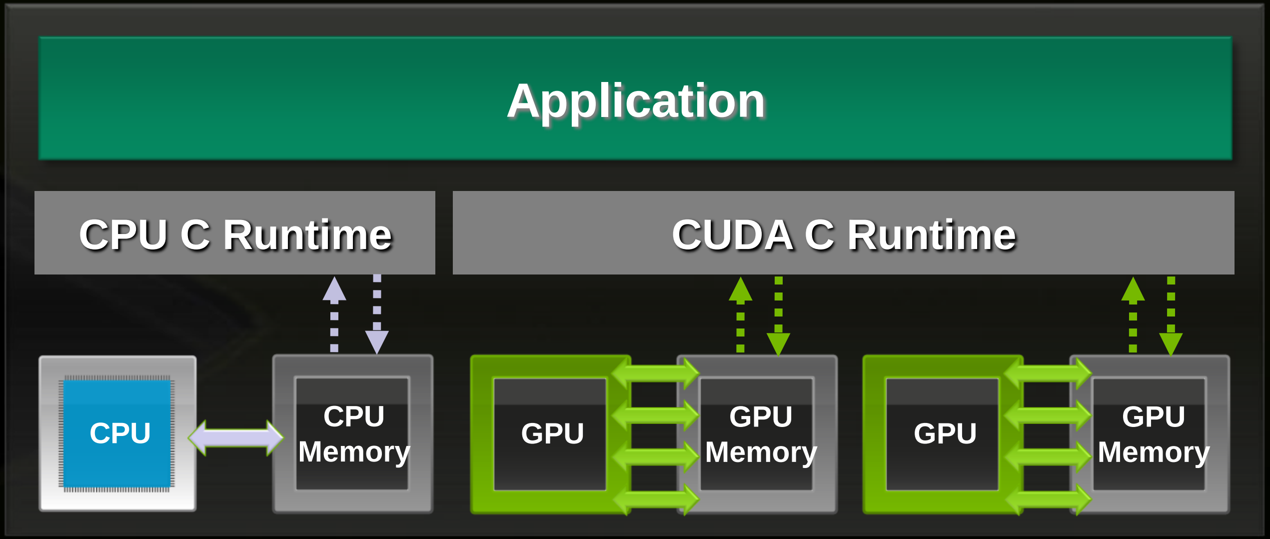


The CUDA C Runtime could be substituted with other methods of accessing the GPU

Basic Application

"Trivial Application" plus:

- Maximize overlap of data transfers and computation
- Minimize communication required between processors
- Use one CPU thread to manage each GPU

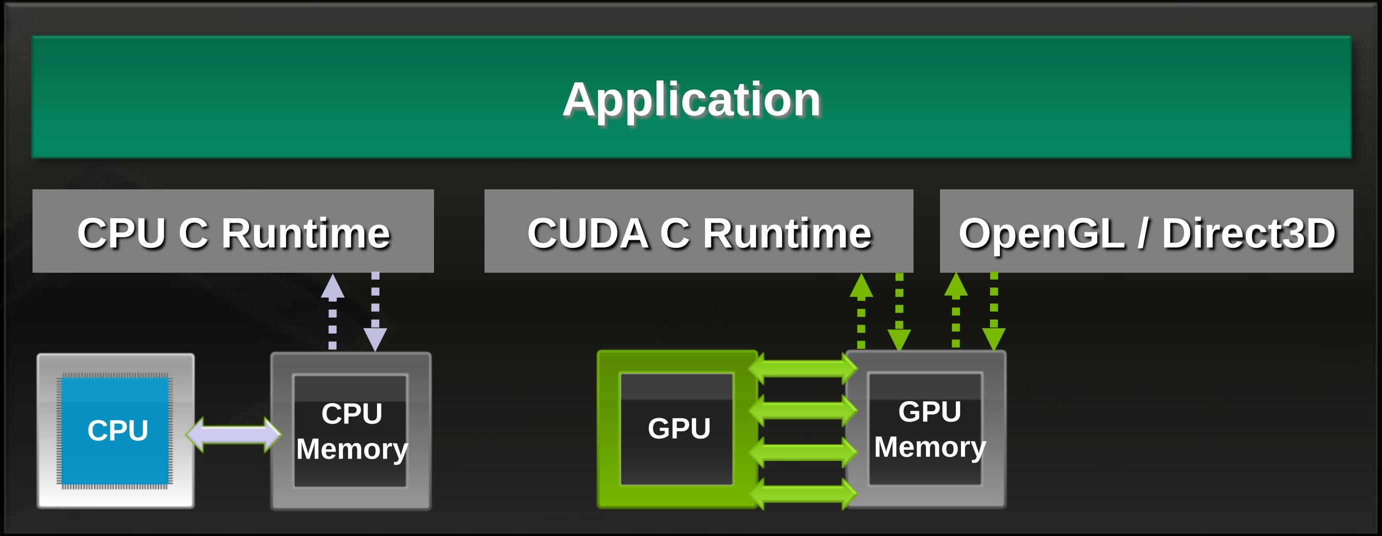


Multi-GPU notebook, desktop, workstation and cluster node configurations are increasingly common

Graphics Application

"Basic Application" plus:

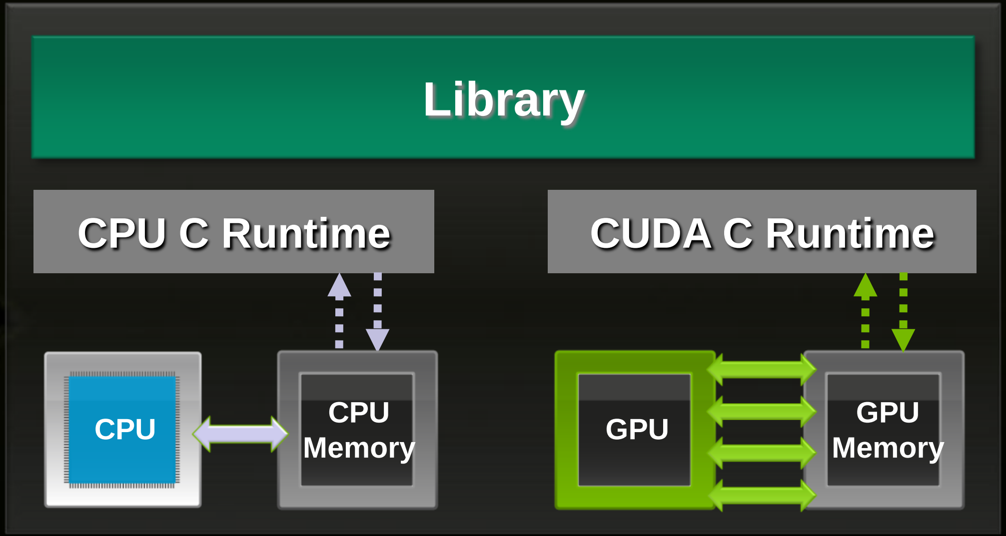
- Use graphics interop to avoid unnecessary copies
- In Multi-GPU systems, put buffers to be displayed in GPU Memory of GPU attached to the display



Basic Library

"Basic Application" plus:

- Avoid unnecessary memory transfers
  - Use data already in GPU memory
  - Create and leave data in GPU memory



These rules apply to plug-ins as well

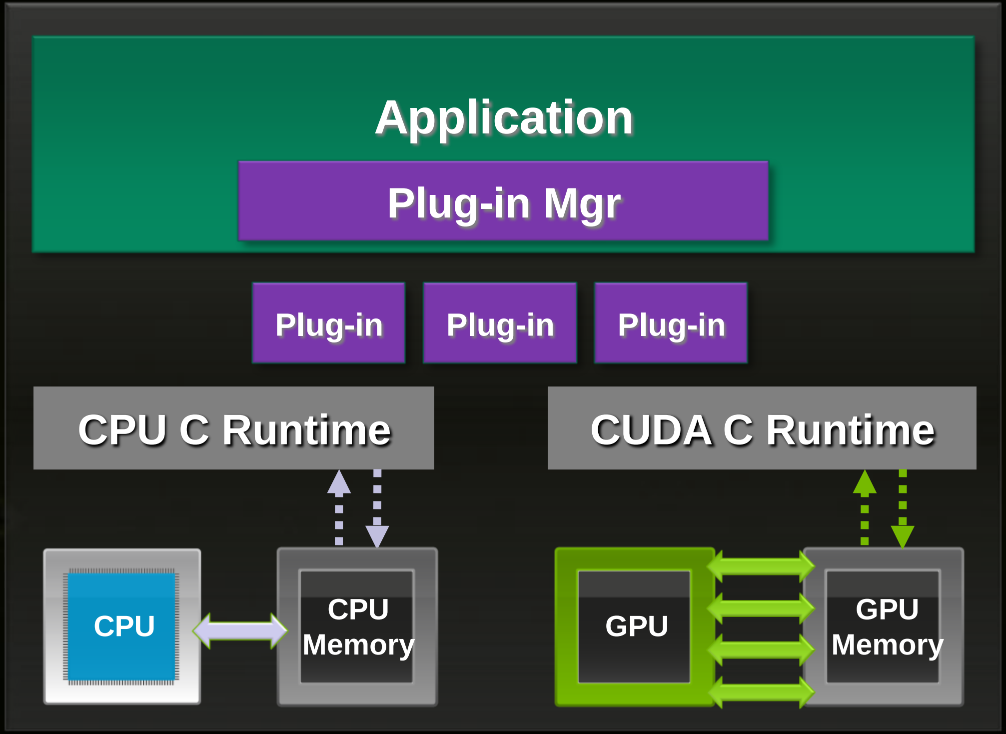
GPU TECHNOLOGY CONFERENCE

Application with Plug-ins

"Basic Application" plus:

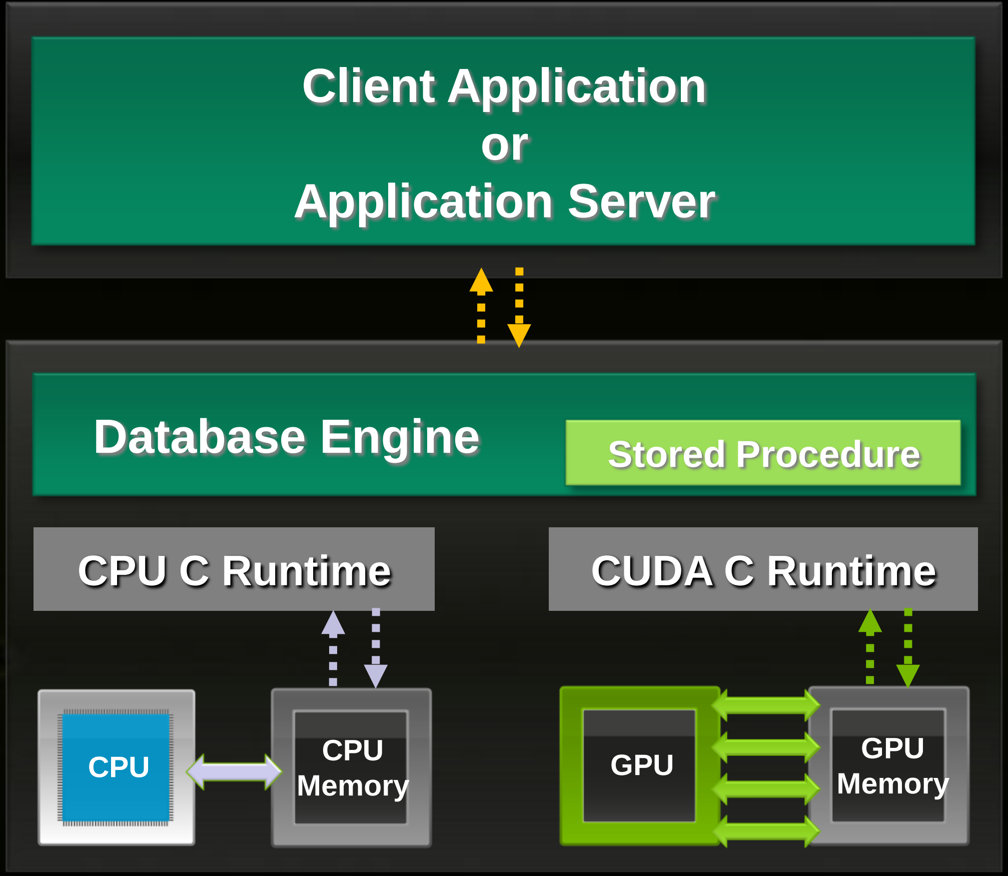
- Plug-in Mgr
  - Allows Application and Plug-ins to (re)use same GPU memory
  - Multi-GPU aware

- Follow "Basic Library" rules for the Plug-ins



Database Application

- Minimize network communication
- Move analysis "upstream" to stored procedures
- Treat each stored procedure like a "Basic Application"
- App Server could also be a "Basic Application"
- Client Application is also a "Basic Application"

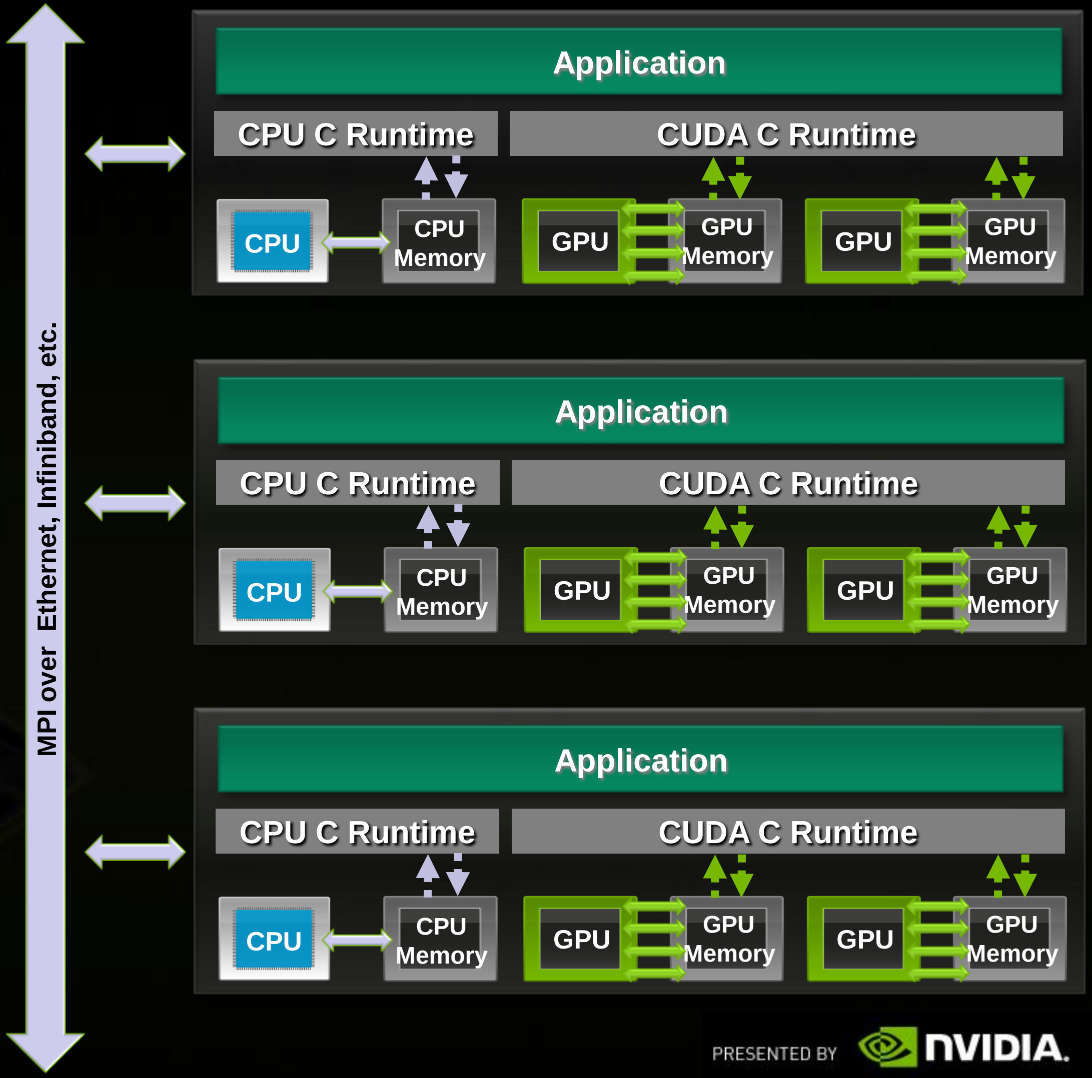


Data Mining, Business Intelligence, etc.

Multi-GPU Cluster Application

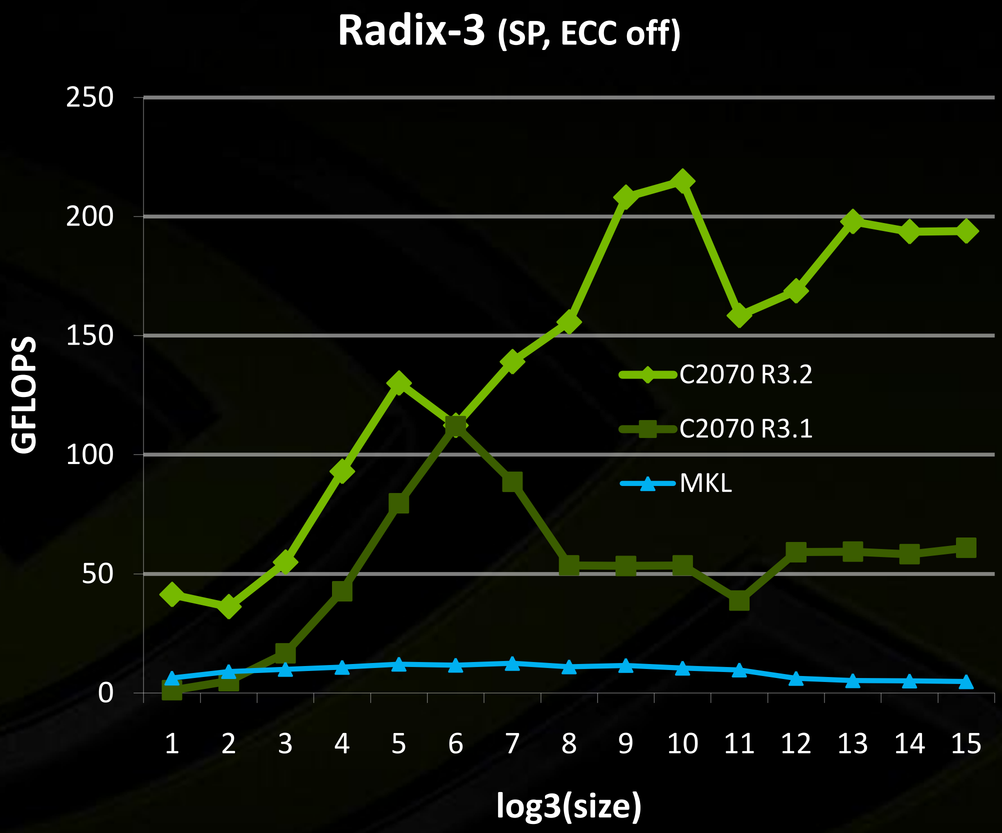
"Basic Application" plus:

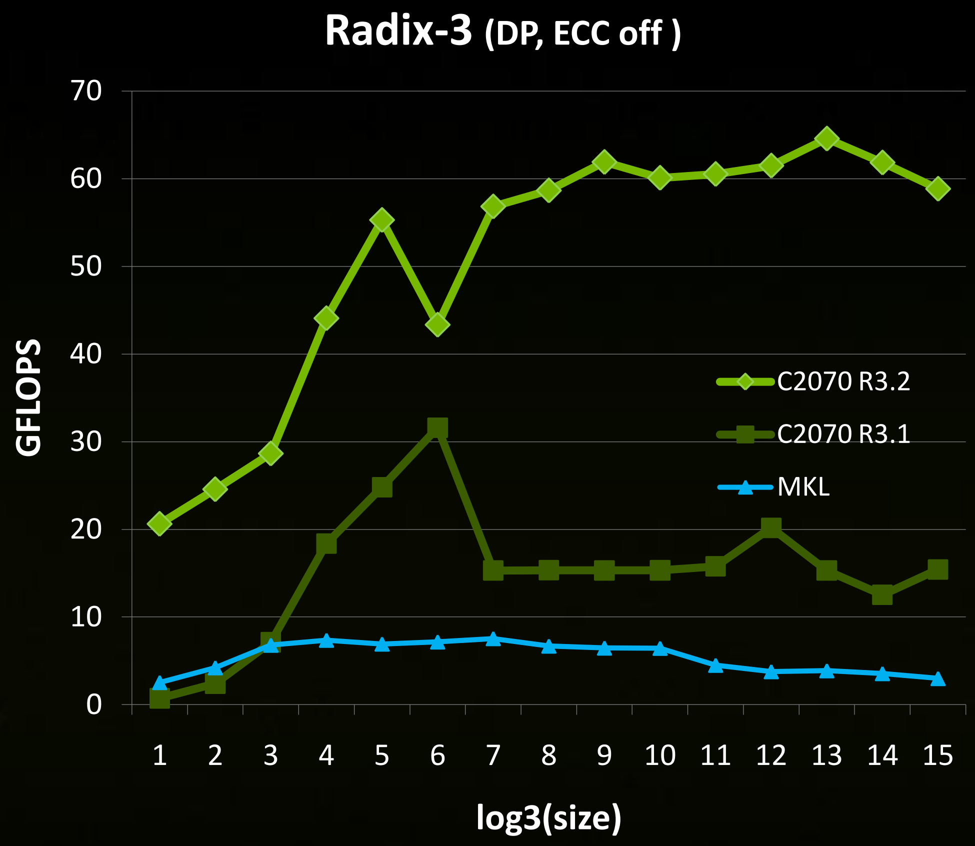
- Use Shared Memory for intra-node communication
  - or pthreads, OpenMP, etc.
- Use MPI to communicate between nodes



Libraries

CUFFT 3.2: Improved Radix-3, -5, -7



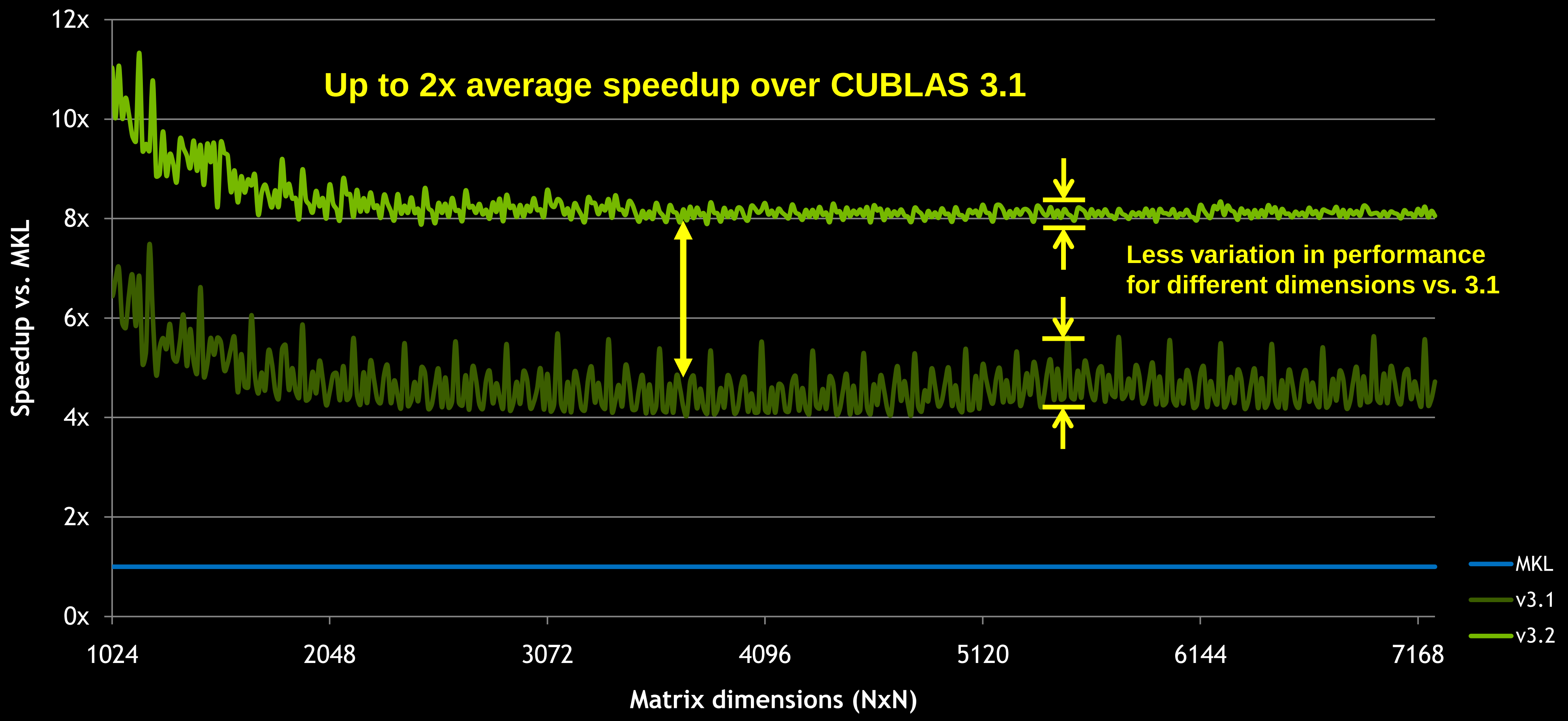


Radix-5, -7 and mixed radix improvements not shown

CUFFT 3.2 & 3.1 on NVIDIA Tesla C2070 GPU

MKL 10.2.3.029 on Quad-Core Intel Core i7 (Nehalem)

CUBLAS Performance



Average speedup of {S/D/C/Z}GEMM x {NN,NT,TN,TT}

CUFFT 3.2 & 3.1 on NVIDIA Tesla C2050 GPU

MKL 10.2.3.029 on Quad-Core Intel Core i7 (Nehalem)

CULA (LAPACK for heterogeneous systems)

CULA | tools

GPU Accelerated Linear Algebra

"CULAPACK" Library

» Dense linear algebra
» C/C++ & FORTRAN
» 150+ Routines

Partnership

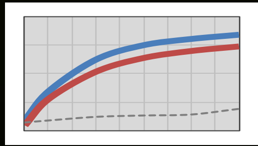
Developed in partnership with NVIDIA

MATLAB Interface

» 15+ functions
» Up to 10x speedup

Supercomputer Speeds

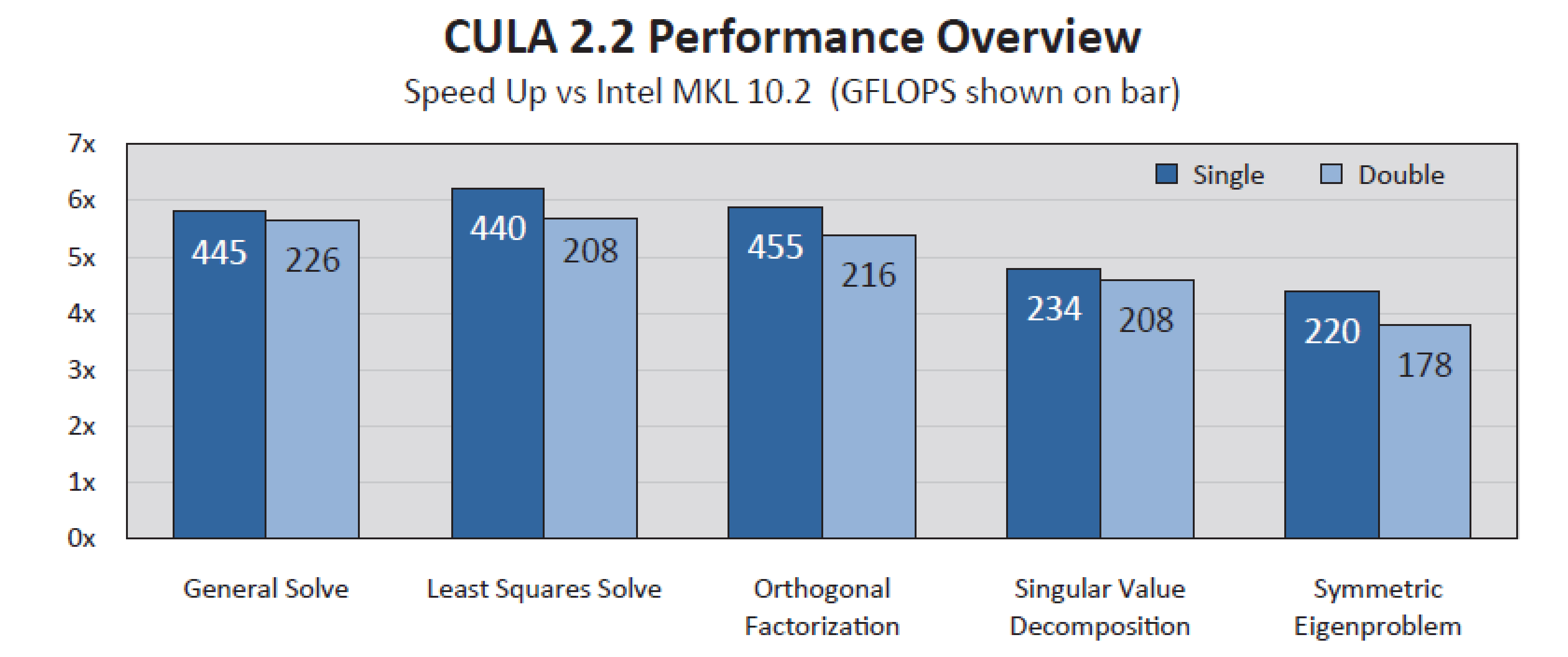
Performance 7x of Intel's MKL LAPACK



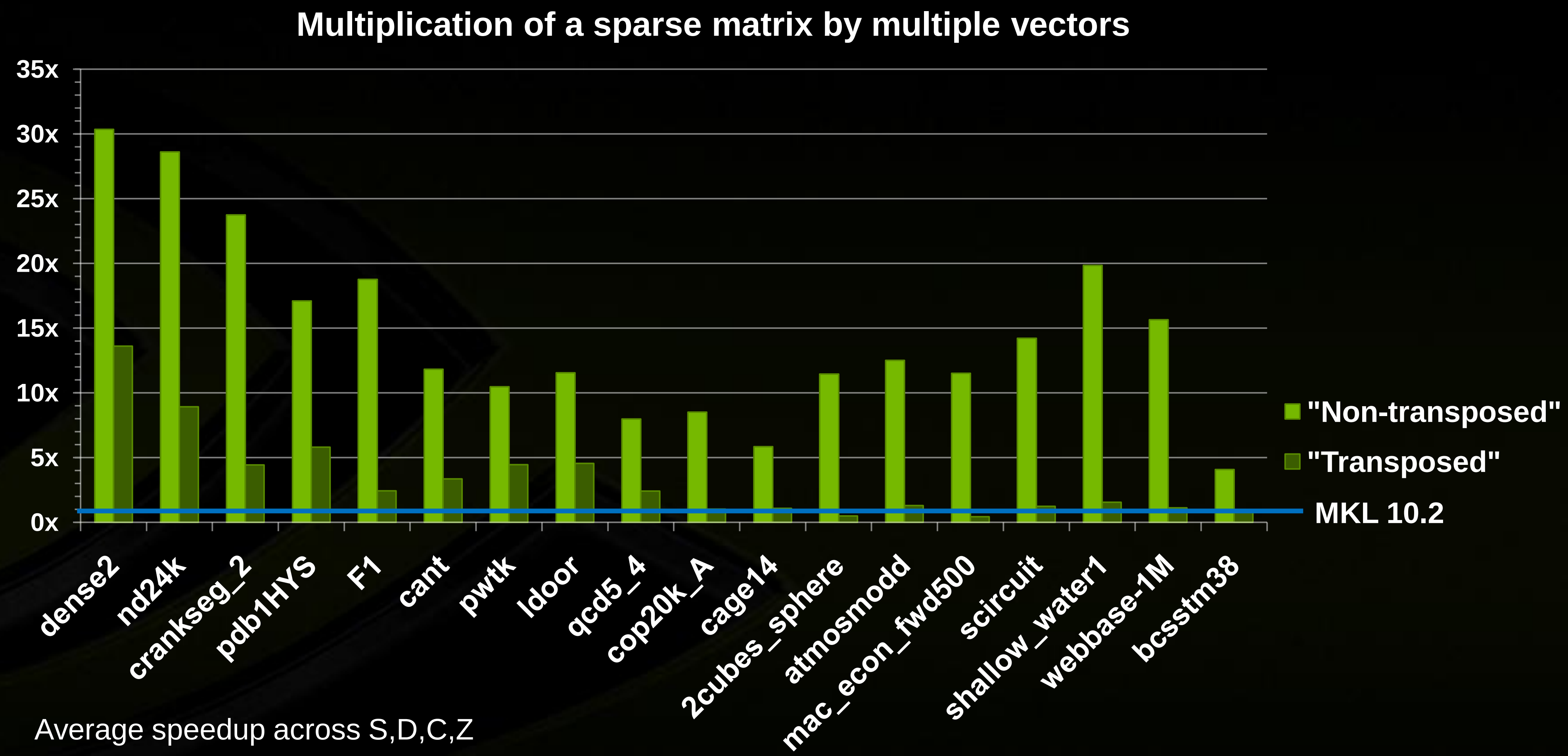
CULA - Performance

Supercomputing Speeds

This graph shows the relative speed of many CULA functions when compared to Intel's MKL 10.2. Benchmarks were obtained comparing an NVIDIA Tesla C2050 (Fermi) and an Intel Core i7 860. More at www.culatools.com



Sparse Matrix Performance: CPU vs. GPU

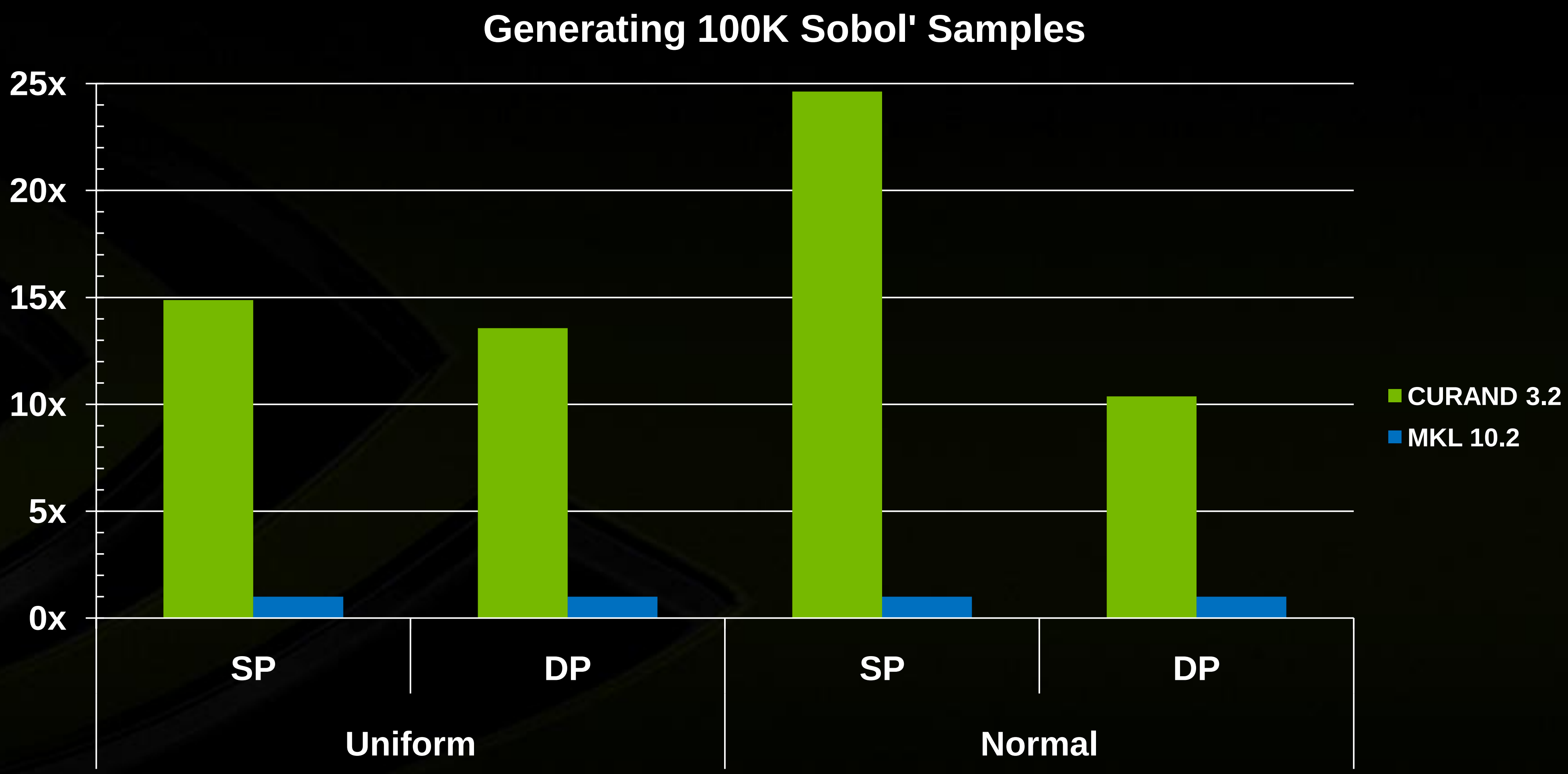


Average speedup across S,D,C,Z

CUSPARSE 3.2 on NVIDIA Tesla C2050 GPU

MKL 10.2.3.029 on Quad-Core Intel Core i7 (Nehalem)

RNG Performance: CPU vs. GPU



CURAND 3.2 on NVIDIA Tesla C2050 GPU

MKL 10.2.3.029 on Quad-Core Intel Core i7 (Nehalem)

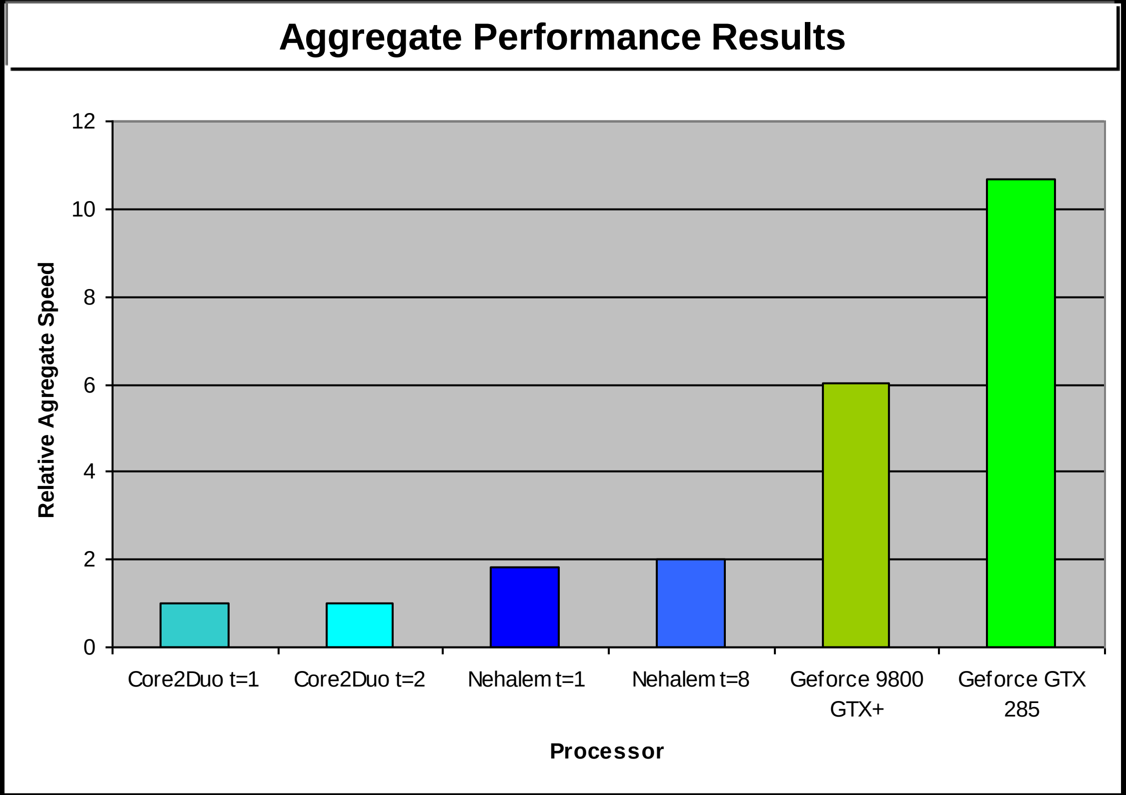
NAG GPU Library

- Monte Carlo related
  - L'Ecuyer, Sobol RNGs
  - Distributions, Brownian Bridge
- Coming soon
  - Mersenne Twister RNG
  - Optimization, PDEs
- Seeking input from the community
- For up-to-date information:
  www.nag.com/numeric/gpus

NAG 1970–2010
40TH ANNIVERSARY
Celebrating 40 years of
numerical excellence

NVIDIA Performance Primitives

- Similar to Intel IPP focused on image and video processing
- 6x - 10x average speedup vs. IPP
  - 2800 performance tests
- Core i7 (new) vs. GTX 285 (old)
- Now available with CUDA Toolkit



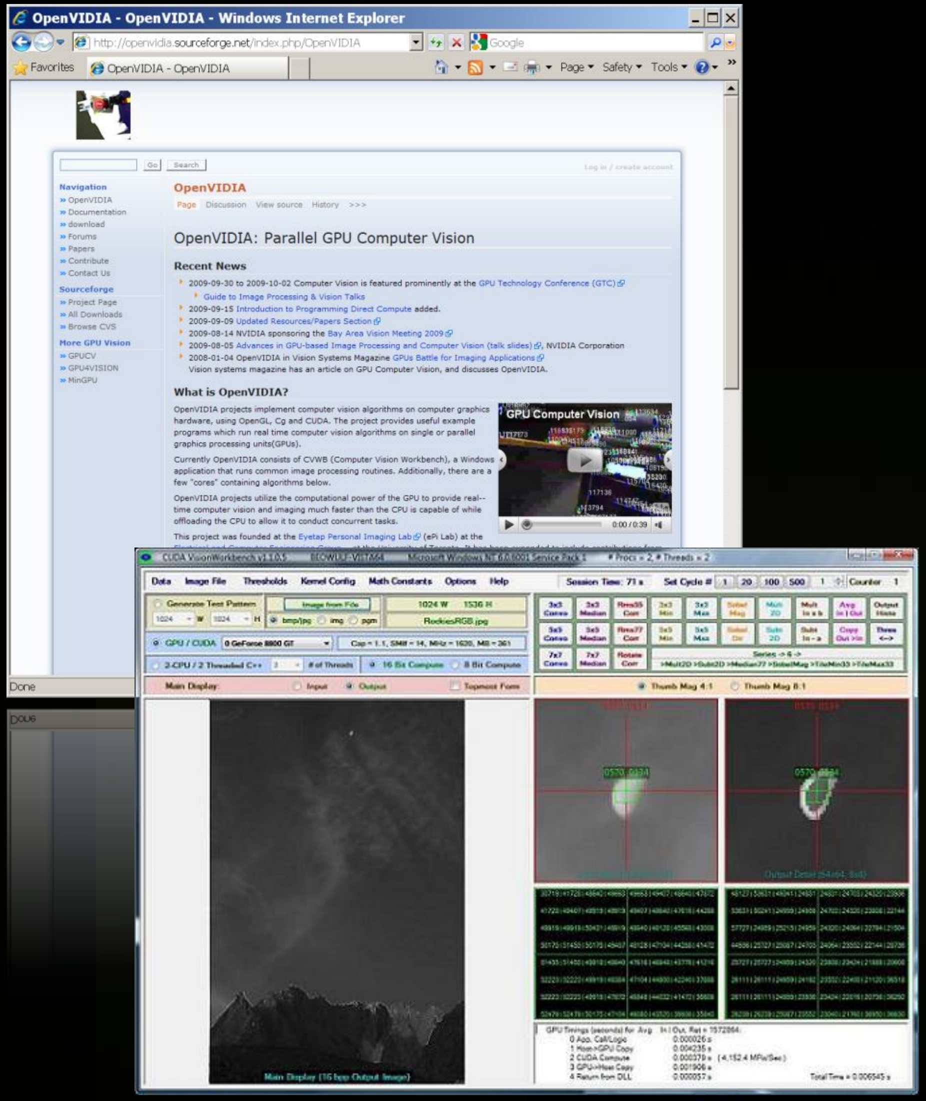
www.nvidia.com/npp

GPU TECHNOLOGY CONFERENCE

OpenVIDIA

- ✓ **Open source, supported by NVIDIA**
- ✓ **Computer Vision Workbench (CVWB)**
  - GPU imaging & computer vision
  - Demonstrates most commonly used image processing primitives on CUDA
  - Demos, code & tutorials/information

http://openvidia.sourceforge.net



PRESENTED BY NVIDIA

More Open Source Projects

- Thrust: Library of parallel algorithms with high-level STL-like interface

Thrust
http://code.google.com/p/thrust

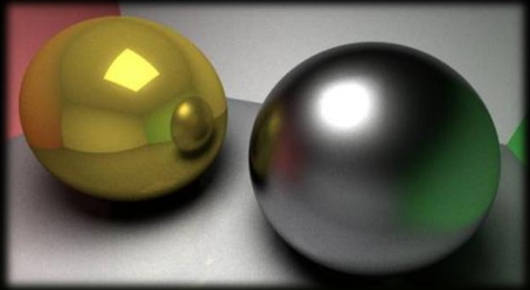
- OpenCurrent: C++ library for solving PDE's over regular grids http://code.google.com/p/opencurrent

- 200+ projects on Google Code & SourceForge
  - Search for CUDA, OpenCL, GPGPU

NVIDIA Application Acceleration Engines - AXEs

OptiX – ray tracing engine

- Programmable GPU ray tracing pipeline that greatly accelerates general ray tracing tasks
- Supports programmable surfaces and custom ray data



OptiX shader example

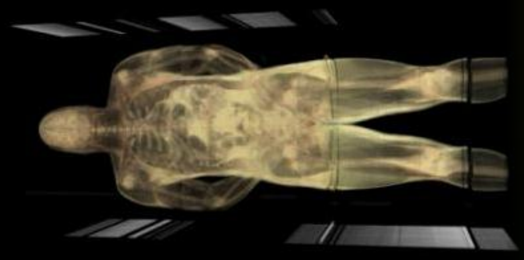
SceniX– scene management engine

- High performance OpenGL scene graph built around CgFX for maximum interactive quality
- Provides ready access to new GPU capabilities & engines

Autodesk Showcase customer example

CompleX – scene scaling engine

- Distributed GPU rendering for keeping complex scenes interactive as they exceed frame buffer limits
- Direct support for SceniX, OpenSceneGraph, and more



15GB Visible Human model from N.I.H.

NVIDIA PhysX™

The World's Most Deployed Physics API

Major PhysX Site Licensees

SEGA™ THQ™ 2K EA ELECTRONIC ARTS™

Integrated in Major Game Engines

- UE3
- Gamebryo
- Vision
- Instinct
- Trinigy
- Diesel
- Unity 3d
- Hero
- BigWorld

Cross Platform Support

Middleware & Tool Integration

- SpeedTree
- Natural Motion
- Fork Particles
- Emotion FX
- Max
- Maya
- XSI

Cluster & Grid Management

GPU Management & Monitoring

NVIDIA Systems Management Interface (nvidia-smi)

Products	Features
All GPUs	• List of GPUs • Product ID • GPU Utilization • PCI Address to Device Enumeration
Server products	• Exclusive use mode • ECC error count & location (Fermi only) • GPU temperature • Unit fan speeds • PSU voltage/current • LED state • Serial number • Firmware version

Use CUDA_VISIBLE_DEVICES to assign GPUs to process

```
[user@cuda-linux ~]$ nvidia-smi -q

Timestamp                        : Wed JUN 9 10:01:01 2010
Unit 0:
        Product Name             : NVIDIA Tesla SXYZ
        Product ID               : 123-45678-012
        Serial Number            : 0123456789012
        Firmware Ver             : X.Y
        GPU 0:
                Product Name             : Tesla C2050
                PCI ID                   : 6d110de
                Temperature              : 63 C
                ECC errors               :
                Single bit               : 0
                Double bit               : 0
                Total                    : 0
                Aggregate single bit     : 0
                Aggregate double bit     : 10
                Aggregate total          : 10
        Fan Tachs:
                #00: 263 Status: NORMAL
                #01: 263 Status: NORMAL
                #02: 263 Status: NORMAL

                ...
        PSU:
                Voltage          :  12.37 V
                Current          :  12.07 A
        LED:
                State            : AMBER
```

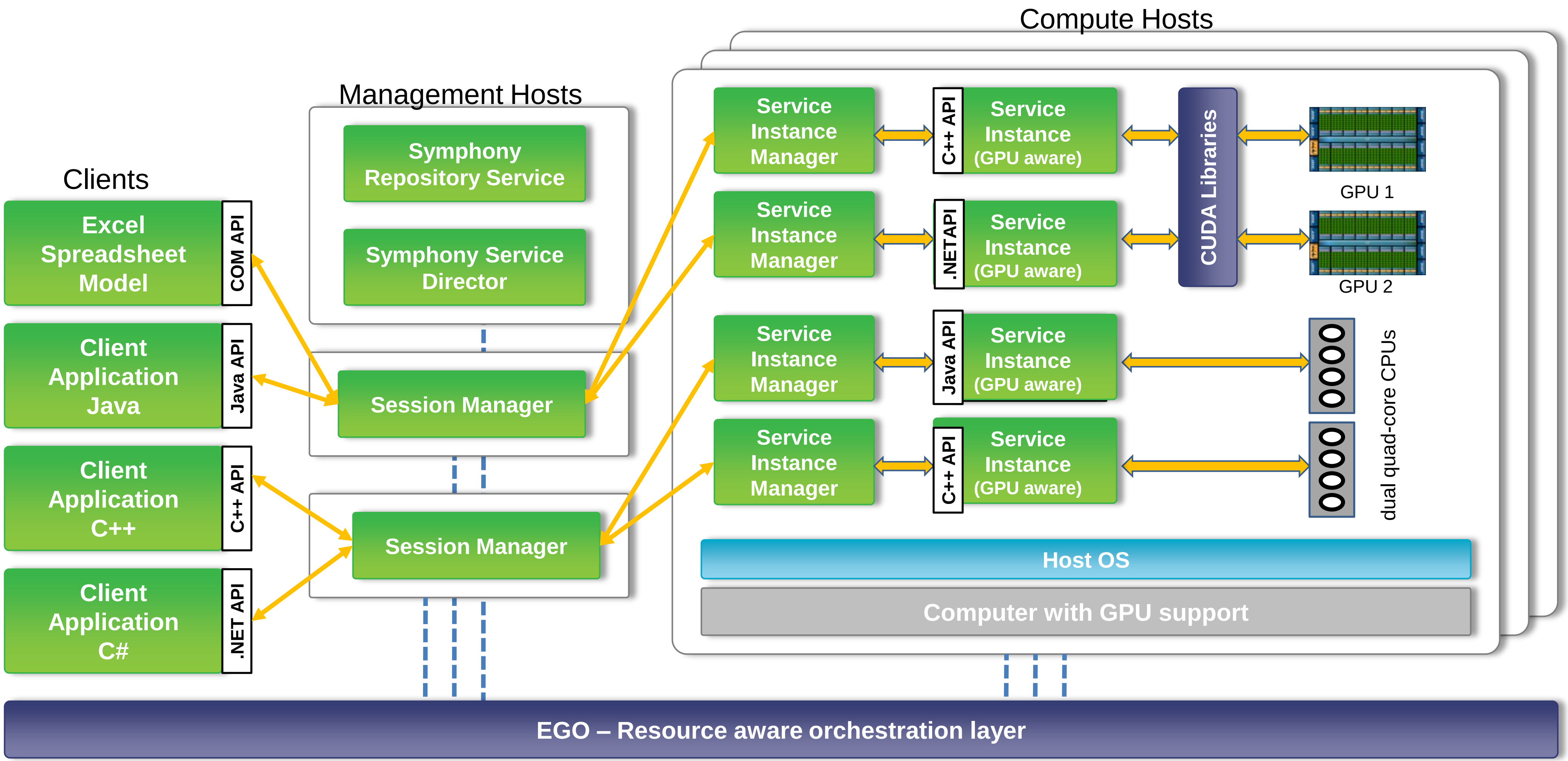
Bright Cluster Manager

Most Advanced Cluster Management Solution for GPU clusters

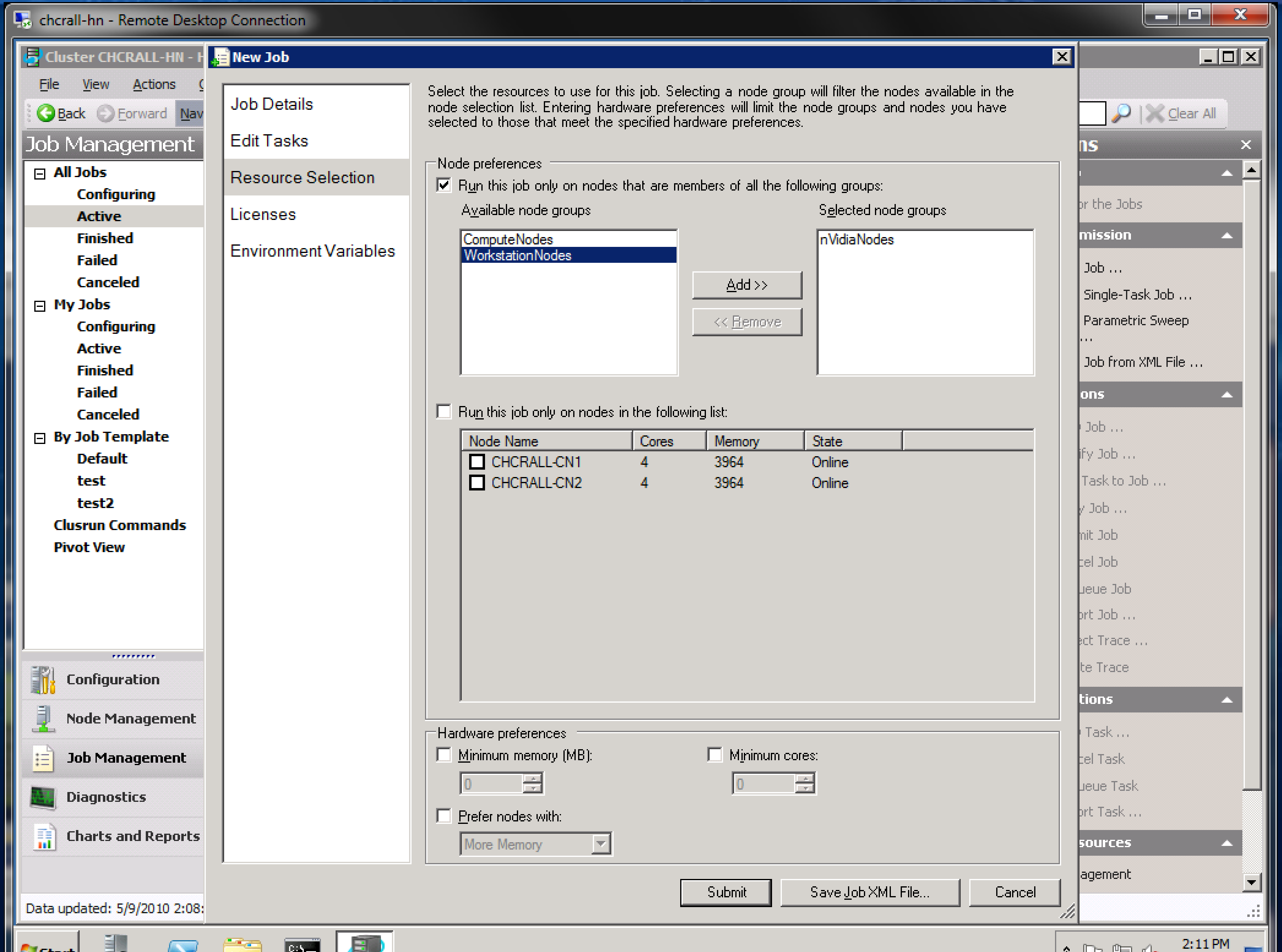
Includes:

- NVIDIA CUDA, OpenCL libraries and GPU drivers
- Automatic sampling of all available NVIDIA GPU metrics
- Flexible graphing of GPU metrics against time
- Visualization of GPU metrics in Rackview
- Powerful cluster automation, setting alerts, alarms and actions when GPU metrics exceed set thresholds
- Health checking framework based on GPU metrics
- Support for all Tesla GPU cards and GPU Computing Systems, including the most recent “Fermi” models

Symphony Architecture and GPU



Selecting GPGPU Nodes



Developer Resources

GPU TECHNOLOGY CONFERENCE

NVIDIA Developer Resources

DEVELOPMENT TOOLS

CUDA Toolkit
Complete GPU computing development kit

cuda-gdb
GPU hardware debugging

Visual Profiler
GPU hardware profiler for CUDA C and OpenCL

Parallel Nsight
Integrated development environment for Visual Studio

NVPerfKit
OpenGL|D3D performance tools

FX Composer
Shader Authoring IDE

SDKs AND CODE SAMPLES

GPU Computing SDK
CUDA C, OpenCL, DirectCompute code samples and documentation

Graphics SDK
DirectX & OpenGL code samples

PhysX SDK
Complete game physics solution

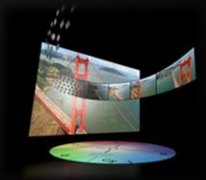
OpenAutomate
SDK for test automation

VIDEO LIBRARIES

Video Decode Acceleration
NVCUVID / NVCUVENC
DXVA
Win7 MFT

Video Encode Acceleration
NVCUVENC
Win7 MFT

Post Processing
Noise reduction / De-interlace/ Polyphase scaling / Color process



ENGINES & LIBRARIES

Math Libraries
CUFFT, CUBLAS, CUSPARSE, CURAND, …

NPP Image Libraries
Performance primitives for imaging

App Acceleration Engines
Optimized software modules for GPU acceleration

Shader Library
Shader and post processing

Optimization Guides
Best Practices for GPU computing and Graphics development

http://developer.nvidia.com

PRESENTED BY NVIDIA

GPU TECHNOLOGY CONFERENCE

10 Published books with 4 in Japanese, 3 in English, 2 in Chinese, 1 in Russian

Google scholar | CUDA GPU | Search | Advanced Scholar Search

Scholar | Articles excluding patents | since 2010 | include citations | Create email alert | Results **1** - **10** of about **1,300**. (**0.17** sec)

An empirically tuned 2D and 3D FFT library on **CUDA GPU**
L Gu, X Li, J Siegel - Proceedings of the 24th ACM International ..., 2010 - portal.acm.org
Page 1. An Empirically Tuned 2D and 3D FFT Library on **CUDA GPU** Liang Gu Department of ECE University of Delaware Newark, DE, USA lianggu@udel.edu **...** A **CUDA GPU** is most eas- ily described as a collection of Multiprocessors(MPs). **...**
Related articles

Hybrid **CUDA**, OpenMP, and MPI parallel programming on multicore **GPU** clusters
CT Yang, CL Huang, CF Lin - Computer Physics Communications, 2010 - Elsevier
Nowadays, NVIDIA's **CUDA** is a general purpose scalable parallel programming model for writing highly parallel applications. It provides several key abstractions – a hierarchy of thread blocks, shared memory, and barrier synchronization. This model has proven quite **...**

Accelerating SSL with **GPUs** — psu.edu [PDF]
K Jang, S Han, S Han, S Moon, KS ... - ACM SIGCOMM Computer ..., 2010 - portal.acm.org
... General Terms Design, experimentation, performance Keywords SSL, **CUDA**, **GPU** 1. INTRODUCTION Secure Sockets Layer (SSL) and Transport Layer Security (TLS) have served as a secure communication channel in the Internet for the past 15 years. **...**

[PDF] High-Precision Numerical Simulations of Rotating Black Holes Accelerated by **CUDA** — arxiv.org [PDF]
R Ginjupalli, G Khanna, G Carbone, M Scaraggi ... - Arxiv preprint arXiv: ..., 2010 - arxiv.org
... It is this code that we accelerate in our work using the Tesla **CUDA GPU** and also the Cell BE. **...** II. NVIDIA **CUDA GPU** AND STI CELL BE All processor manufacturers have moved towards multi-core designs today in the quest for higher performance. **...**
Related articles - View as HTML - All 4 versions

[PDF] An MPI-**CUDA** Implementation for Massively Parallel Incompressible Flow Computations on Multi-**GPU** Clusters — boisestate.edu [PDF]
DA Jacobsen, JC Thibault, I ... - Mechanical and ..., 2010 - scholarworks.boisestate.edu
∗Boise State University †Boise State University, jcv.thibault@gmail.com ‡Boise State University, senocak@boisestate.edu This paper is posted at ScholarWorks. http://scholarworks.boisestate.edu/mecheng facpubs/5 **...** An MPI-**CUDA** Implementation **...**
Cited by 1 - All 5 versions

An effective **GPU** implementation of breadth-first search
L Luo, M Wong, W Hwu - Proceedings of the 47th Design ..., 2010 - portal.acm.org
... General Terms Algorithms, Performance Keywords **CUDA**, **GPU** computing, BFS **...** 273-282. [2] P. Harish and PJ Narayanan, "Accelerating large graph algorithms on the **GPU** using

NVIDIA

GPU TECHNOLOGY CONFERENCE

GPU Computing Research & Education

World Class Research Leadership and Teaching

- University of Cambridge
- Harvard University
- University of Utah
- University of Tennessee
- University of Maryland
- University of Illinois at Urbana-Champaign
- Tsinghua University
- Tokyo Institute of Technology
- Chinese Academy of Sciences
- National Taiwan University

NVIDIA CUDA™ CENTER OF EXCELLENCE

Premier Academic Partners

Proven Research Vision

Launched June 1st with 5 premiere Centers and more in review

- John Hopkins University , USA
- Nanyan University, Singapore
- Technical University of Ostrava, Czech
- CSIRO, Australia
- SINTEF, Norway

NVIDIA CUDA™ RESEARCH CENTER

Exclusive Events, Latest HW, Discounts

Quality GPGPU Teaching

Launched June 1st with 7 premiere Centers and more in review

- McMaster University, Canada
- Potsdam, USA
- UNC-Charlotte,USA
- Cal Poly San Luis Obispo, USA
- ITESM, Mexico
- Czech Technical University, Prague, Czech
- Qingdao University, China

NVIDIA CUDA™ TEACHING CENTER

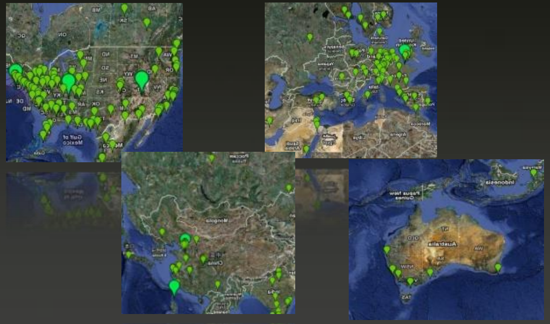
Teaching Kits, Discounts, Training

Academic Partnerships / Fellowships

Supporting 100's of Researchers around the globe ever year

NV Research
http://research.nvidia.com

Education
350+ Universities



Thank you!