# CUDA 5 YEARS

### **CUDA 4.1**

Application Acceleration Made Easier

New LLVM-Based Compiler Delivers +10% performance for many applications

1,000+ New Image Processing Functions "Drop-in" acceleration with NPP library

**Re-Designed Visual Profiler** *Automated Analysis & Integrated Expert Guidance* 

### **New LLVM-based CUDA Compiler**

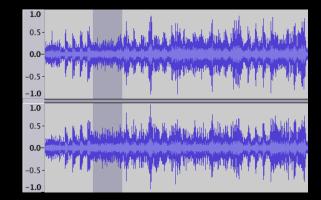


- Delivers up to 10% faster application performance
- Faster compilation for increased developer productivity
- Modern compiler with broad support
  - Will bring more languages to the GPU
  - Easier to support CUDA to more platforms



# 1000+ New Imaging Functions in NPP 4.1

- NVIDIA Performance Primitives (NPP) library includes over 2200 GPU-accelerated functions for image & signal processing Arithmetic, Logic, Conversions, Filters, Statistics, etc.
- Up to 40x faster performance than Intel IPP



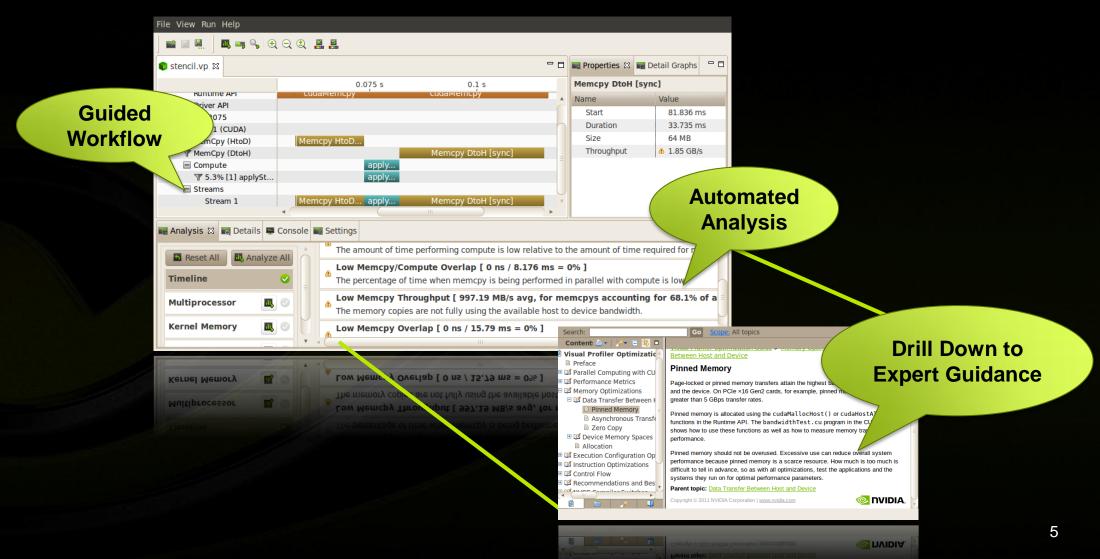


http://developer.nvidia.com/content/graphcuts-using-npp

\* NPP 4.1, NVIDIA C2050 (Fermi) \* IPP 6.1, Dual Socket Core™ i7 920 @ 2.67GHz

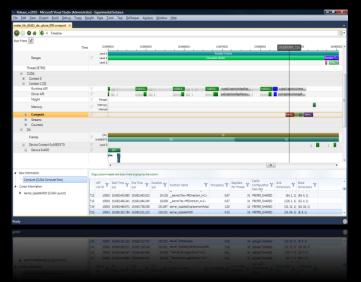
### **Re-designed Visual Profiler**





### **NVIDIA Parallel Nsight™ 2.1 for Visual Studio**

00 Releas	se voil 11 (Debugging) - Mic	rosoft Visual Studio	(Administrator) - E	perimenta	al Enstance							04	X
	t View VAssist& Broject					Help							
	mections: localhost		. 6 0 8										
	emcuda.cu 🔍 🗶 vector 🛙				Disastembly			• # x cup	A Wats Web	vib 1		• 1	
				_		rte grotello di e dfl.billEfvFN4vec.417.E4TvpeE		• *	- wapna	411.4			
	al Scope)		Bodiestypename v		Viewing Opti		scage:		Name	position			
240					Decotion 4 wears	e10000000011cc5							11
241		e they cancel of so4CD : : Type ve			2431		317-130 [HO]' H			iocal_float4			
243		eracisti file of	recred - serfa		244:	velocity.x += accel.x * del	LtaTime;		0	= -1.6635864, y =	4.6042924, z = -3.151	9358, == = 1)	
244		+= accel.x = d			0x0004e488		[439+16];		1 🔅	* -5.1818962, y *	0.15936305, z = -0.83	475435, w = 1}	
245		+- accel.y.* d			Cx0004e468	480 Dc D0D 40 20 1 c D3	IADD RD, RZ, Ox	:10:	2 5	4 803510 v -	-1.0198283, z = -1.4526	002 m = 11	
246 247		+= accel.z * d	beltaTime;		0x00046490	280000000001de4 c00000000001c85	MOV R0, R0; 131 R0, [R0];						-
240		** dempiner			0x01046402	[3093] mul.f32 4f35, 4f34, 4f3				= -0.90191/05, y	= -2.8533263, z = -3.66	195209, == 1}	
249	velocity.y	*= dsaping;			0x00046460	580.00.000-4c.00.dc.00	FMUL R3, R0, R1	97	4 (*	= 0.63947666, y =	= ~4.8788266, z = 0.517	22354, w = 1}	
250		*- dsaping;			0x0004e4e5		[439+48];		5 5	z -3.5690832 v z	-1.2420981 z = -3.877	646 w = 11	
251					0x0004a4a3 0x0004a4b0	4800c000c0201c03 280000000001de4	IADD RD, RZ, 0x MUV RD, RD;		6 5	. 1 73361	7225604 z = -3.451830	0	- 1
253		+= velocity.x		1 N	0x0004o4b8		104 R0, (R01;						- 1
254	position.y	+= velocity.y	deltaTime;		0x000464c0	[3097] add.E32 4E37, 4E36, 4E3	151				: 1.7973436, z = 3.7060		
255		+= velocity.z	* deltaTime;		0x0004e4c0	500000000e00de00	7620 R3, R0, R3	U I	8 6	= -2.8979087, y =	3.3188901, z = 1.7311	301, w = 1}	
256					0x0004e4c8 0x0004e4c8	[3099] st.local.f32 [359+40 4000c000c0201c03	1], 3237; IADO RO, RZ, OX		9 5	* -2.8265455 v *	2.5229209.z + -3.230	(729 == 1)	
257 258		iceOffset + ind			0x00046400	2800000000001de4	MUV RO, RD; 00				-2.2989305 z = 1.0172		- 1
259		Offset + index1			0x0004e4d8	c80000000000000085	STL [RD], R3;						- 1
260					245:	velocity.y += accel.y * del			11 🔅	= 0.87436295, y =	= -2.2763903, z = 4.400	5332, == 1)	-
261					0x0104a4e0	[3102] 1d.local.f32 %f38, [	[43P+20]J			rtch 1 💽 Solutio	C . 1	,	_
_		-			*					8041 🔍 30000	on Exprorer		
CUDAIN								≁ ų ×				* 8	X
00	Warps 🔹 🗔 🕻	🖬 🦪 🍸 Filten					Vi	ewing 448/448	Name		Value	Type	- 1
Current	Warp Index CUcontext	Orid TO Mark?	dx three	(T.d.)	Lanes		Status Fi	le Name *			L02000000012918232 h000000001 (187578559	device	
	7 0x031618d8		0, 0) (224,	0. 0							x = 0, y = 0, z = 0	cant ui	
				., .				dysystemcu	E 🛛 E		x = 21, y = 0, z = 0]	const ui	
÷ .	0 0x031618d8	47 (21,	0, 0) ( 0,	0, 0			i Brezkpo: bo	dysystemcu 🔳			x = 256, y = 1, z = 1}	const di	
	1 0x031618d8	47 (21,	0, 0) (32,	0, 0			🔒 Breakpo: bo	dysystemcu			x = 56, y = 1, z = 1}	const di	
	2 0x03161848	47 (21,	0, 0) (64.				Breakpo bo	durant and			x = 56.72400, y = 20.495		
				0, 0	~						x = -1.6635864, y = 4.68 x = 110.04307, y = -294		
	3 0x031618d8		0, 0) (96,	0, 0	9)		None bo	dysystemcu			n = 110.04001, y = -234. 1376	43233, _1008_	11
· —	* & A11/10-4		A A) (A)A				· · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	1.00		🖥 Registers 📑 Outes		
_									S Loca	b Match 1	🖬 Regoters 🛄 Outpi	at CUUA Into	
100									STOC .	NAME OF B			
									0.1				
		40 C 33					ana an		- 0				
							11 Openants and						



o Reli	ase_vs2010 - Mic	crosoft Visual	Studio (Adr	ninistrator) - Exp	erimental Ins	tance												х,
Eile	dit Yiew Proj	ect Build	Debug Te	em Nisight D	gta ∐ools	Test <u>BeSharp</u>	er Agabjze	Window	Help									
Black	choles111025pt	ture_000.nvre	port X Ac															
0	046	<ul> <li>CUDA L</li> </ul>	urches															3.
																	Max	ing: 2 / 2
																	Ten	-9.474
Drag	a column header and	drop it here to g	roup by that co	unn														
R	nction Name 🍸	Module Y	Function V	Start Time 🕎	Duration V	Occupancy 😵	Registers per Thread		per Y		Y per		Local Memory V (bytes)	Cache Configuration V	Grid Dimensions	Block Dimension	s Y API Call	Y .
1.8	exiScholesGPU	6		1 12,977,657,492		0.6		Block (b 25	ytes)	Block (bytes)	D Dy1	(cs) 36		Executed PREFER SHARED	(480.1.1	(12	111	139
	sckScholesGPU	6		1 24,784,756,196				3	0		0	72		PREFER SHARED	(480, 1, 1		1.1.10	240
•																		•
4.8	uckScholesGPU (CU	(DA Launch)	VevBr	Size								Name		Total	Per Warp	Pe	r Second	
1	BlackScholesGPU	ICUDA Kerne										A Global						
1	Experiment Resul											Request		625,0	00.00	325.52	290,765,7	00.00
	- CUDA Occup			625.00 kReg	Gasa	76.29 M8						Loads		375,0	00.00	195.31	150,459,4	00.00
		ction Statistics									ŝ	Stores		250,0	00.00	130.21	100,306,3	00.00
	CUDA Branch			6.00 MReg	Local	732.42 MB				0.008		Transact	ions.	625,0	00.00	325.52	250,765,7	00.00
	CUDA Issue I							748.04 M8			1	Loads		375,0	00.00	195.31	150,459,4	00.00
	CUDA Achiev			0.00 Reg	ATOMS	0.00 8	15.9%				S.	Stores		250,0	00.00	130.21	100,306,3	00.00
	CUDA Mene	sy statistics	1		AIOMS				L2 Cache			Size		76.2	9 MB	40.69 k8	29.89	68/5
			3	0.00 Reg		0.00 8			27.0%			Londs		45.7	5 MB	24.41 kB	17.94	68/5
					REDs						5	Stores		30.5	2 M8	16.28 kB	11.96	68/5
				0.00 Reg		0.00 8				546.03 MB	ğ	Replay C	Verhead	0.	46 %			
					Shared		Shared Mem	ery			1	<ul> <li>Local</li> </ul>						
				0.00 Reg		0.00 8	Cache	0.00 5			8	Request		6,000,0	00.00	3,125.00	2,407,351,0	00.00
					Texture		0.0 %	1.00 0				Loads		3,000,00	00.00	1,562.50	1,203,676,0	00.00
			-								-	Stores		3,000,0	00.00	1,562.50	1,203,676,0	00.00
																		****
			Overvie	Giobai Loca	Atomics	Shared   Texture	Caches	Butters										
Ready																		•
			Onio	2024 7972	Annes   1	Const Contra	Care	THE .										
			-		-		-				-							

#### **CUDA Debugger**

- Debug CUDA kernels directly on GPU hardware
- Examine thousands of threads executing in parallel
- Use on-target conditional breakpoints to locate errors

#### **CUDA Memory Checker**

Enables precise error detection

#### **System Trace**

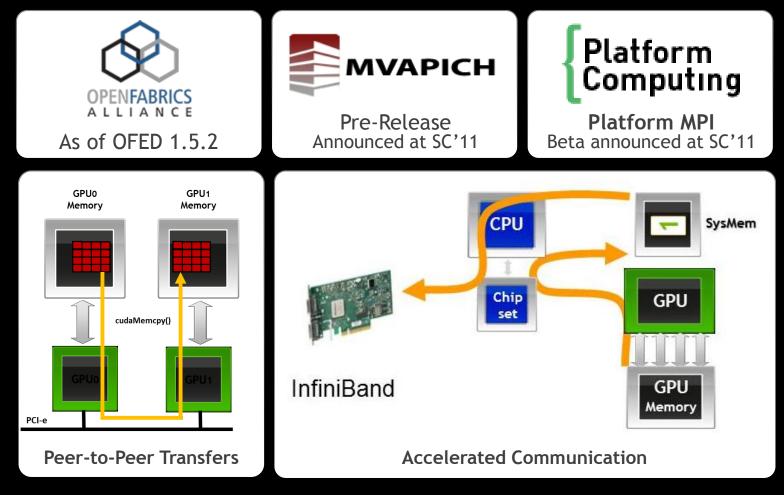
- Review CUDA activities across CPU and GPU
- Perform deep kernel analysis to detect factors limiting maximum performance

#### **CUDA Profiler**

Advanced experiments to measure memory utilization, instruction throughput and stalls

### **GPU-Aware MPI Libraries**

#### **Integrated Support for GPU Computing**



### **CUDA 4.1 Highlights**

Advanced Application Development

GPU-Accelerated Libraries New & Improved Developer Tools

- New LLVM-based compiler
- 3D surfaces & cube maps
- Peer-to-Peer between processes
- GPU reset with nvidia-smi
- New GrabCut sample shows interactive foreground extraction
- New code samples for optical flow, volume filtering and more...

- 1000+ new imaging functions
- Tri-diagonal solver 10x faster vs. MKL
- MRG32k3a & MTGP11213 RNGs
- New Bessell functions in Math lib
- 2x faster matrix-vector w/ HYB-ELL
- Boost-style placeholders in Thrust
- Batched GEMM for small matrices

- Re-Designed Visual Profiler
- Parallel Nsight 2.1
- Multi-context debugging
- assert() in device code
- Enhanced CUDA-MEMCHECK

### **NVIDIA CUDA Platform**

	Platform	Programming Model	Libraries	Tools
CUDA 4.1 Highlights	New LLVM-based Compiler	P2P Between Processes 3D Surfaces & Cubemaps	1000+ New Imaging Functions New Tri-diagonal solver	New Visual Profiler Parallel Nsight 2.1
	Hardware Support ECC Memory Double Precision Native 64-bit Architecture GPUDirect™ Communication Concurrent Kernel Execution Dual Copy Engines Multi-GPU support 6GB per GPU supported Operating System Support MS Windows 32/64 Linux 32/64 support Mac OSX 32/64 support Mac OSX 32/64 support	<section-header><ul> <li><b>C support</b></li> <li>• NVIDIA C Compiler</li> <li>• CUDA C Parallel Extensions</li> <li>• Function Pointers</li> <li>• Recursion</li> <li>• Atomics</li> <li>• malloc/free</li> </ul> <b>Otest State</b> <ul> <li>• Classes/Objects</li> <li>• new/delete</li> <li>• Class Inheritance</li> <li>• Polymorphism</li> <li>• Operator Overloading</li> <li>• Class Templates</li> <li>• Nirtual Base Classes</li> <li>• Namespaces</li> </ul> <b>Fortran also available</b></section-header>	CUDA Toolkit Libraries Complete math.h Complete BLAS Library (1, 2 and 3) Sparse Matrix Math Library RNG Library FFT Library (1D, 2D and 3D) Thrust Template Library Image Processing Library (NPP) Video Processing Library (NPP) Video Codec Libraries Additional Libraries • MAGMA • IMSL • VSIPL	NVIDIA Developer Tools Parallel Nsight 1.0 IDE cuda-gdb Debugger with multi-GPU CUDA/OpenCL Visual Profiler CUDA Memory Checker CUDA C SDK CUDA Disassembler 3rd Party Developer Tools Allinea DDT, Totalview PAPI, TAU, Vampir Languages & APIs CUDA Fortran CUDA-x86 for CPUs OpenACC PGI Accelerator for C / Fortran CAPS HMPP Python, C#, DirectCompute, OpenCL Many more



### **CUDA Libraries** Performance Report

# **CUDA Math Libraries**

### High performance math routines for your applications:

- cuFFT Fast Fourier Transforms Library
- cuBLAS Complete BLAS Library
- cuSPARSE Sparse Matrix Library
- cuRAND Random Number Generation (RNG) Library
- NPP Performance Primitives for Image & Video Processing
- Thrust Templated Parallel Algorithms & Data Structures
- math.h C99 floating-point Library

Included in the CUDA Toolkit Free download @ www.nvidia.com/getcuda

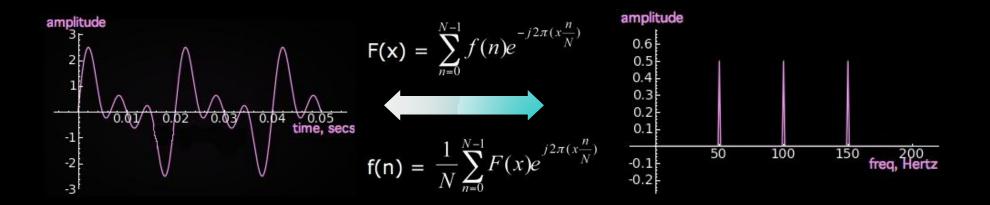
#### More information on CUDA libraries:

http://www.nvidia.com/object/gtc2010-presentation-archive.html#session2216

### cuFFT: Multi-dimensional FFTs

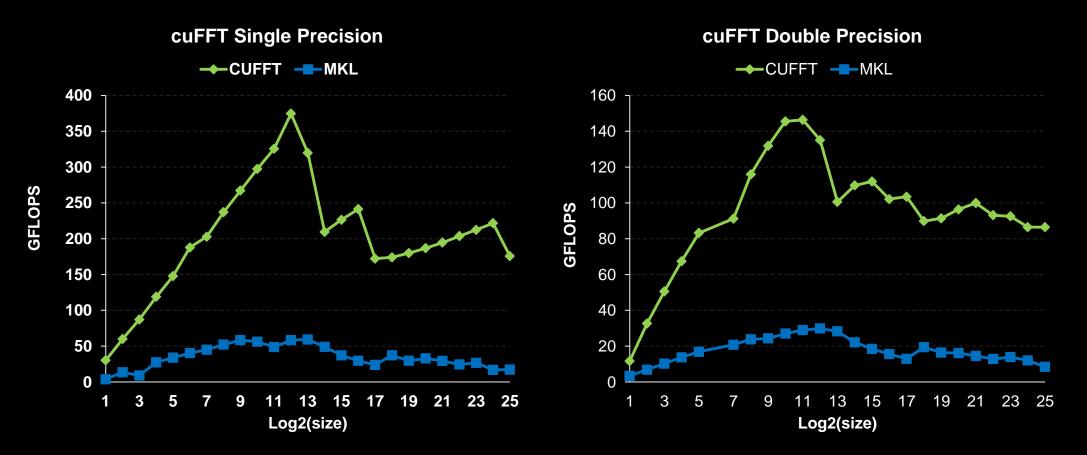
### New in CUDA 4.1

- Flexible input & output data layouts for all transform types
  - Similar to the FFTW "Advanced Interface"
  - Eliminates extra data transposes and copies
- API is now thread-safe & callable from multiple host threads
- Restructured documentation to clarify data layouts



### FFTs up to 10x Faster than MKL

1D used in audio processing and as a foundation for 2D and 3D FFTs



• Measured on sizes that are exactly powers-of-2

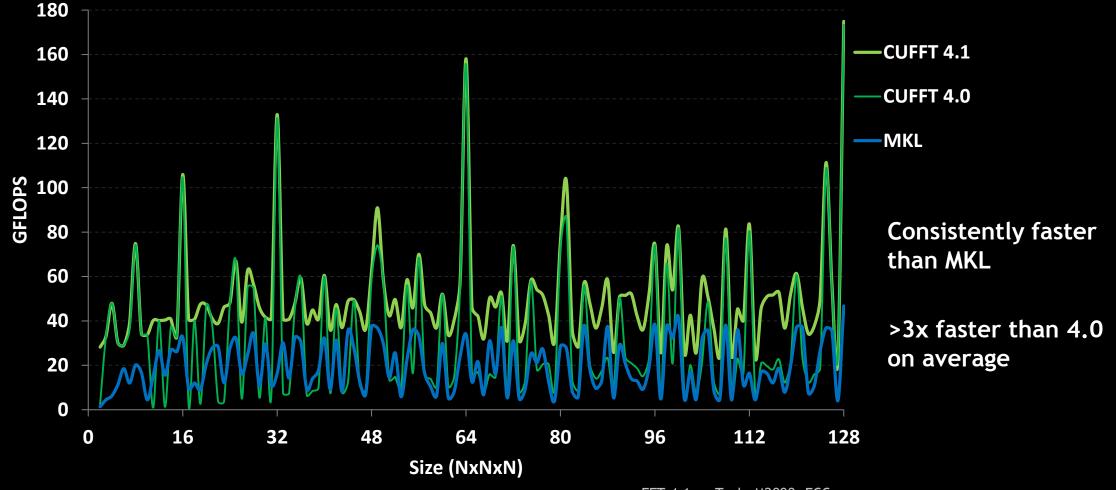
• cuFFT 4.1 on Tesla M2090, ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 13

Performance may vary based on OS version and motherboard configuration

### CUDA 4.1 optimizes 3D transforms

Single Precision All Sizes 2x2x2 to 128x128x128



Performance may vary based on OS version and motherboard configuration

• cuFFT 4.1 on Tesla M2090, ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 14

### cuBLAS: Dense Linear Algebra on GPUs

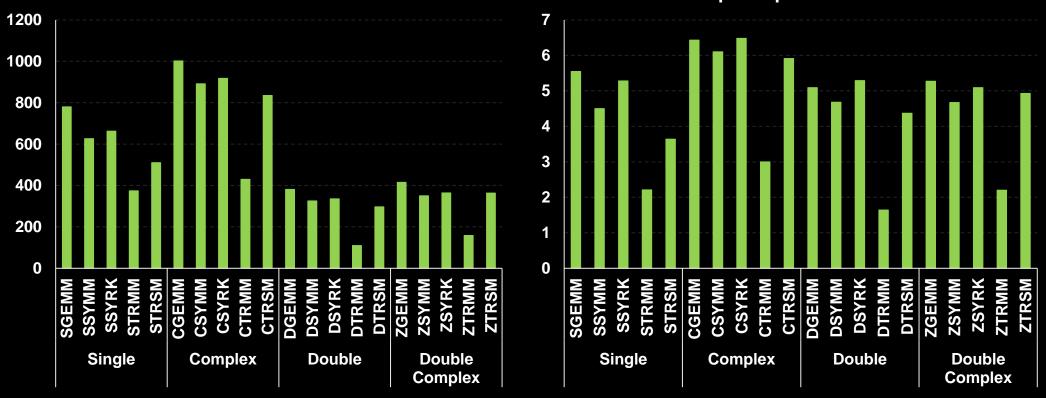
- Complete BLAS implementation plus useful extensions
  - Supports all 152 standard routines for single, double, complex, and double complex

### New in CUDA 4.1

- New batched GEMM API provides >4x speedup over MKL
  - Useful for batches of 100+ small matrices from 4x4 to 128x128
- 5%-10% performance improvement to large GEMMs

# cuBLAS Level 3 Performance

Up to 1 TFLOPS sustained performance and >6X speedup over Intel MKL



GFLOPS

• 4Kx4K matrix size

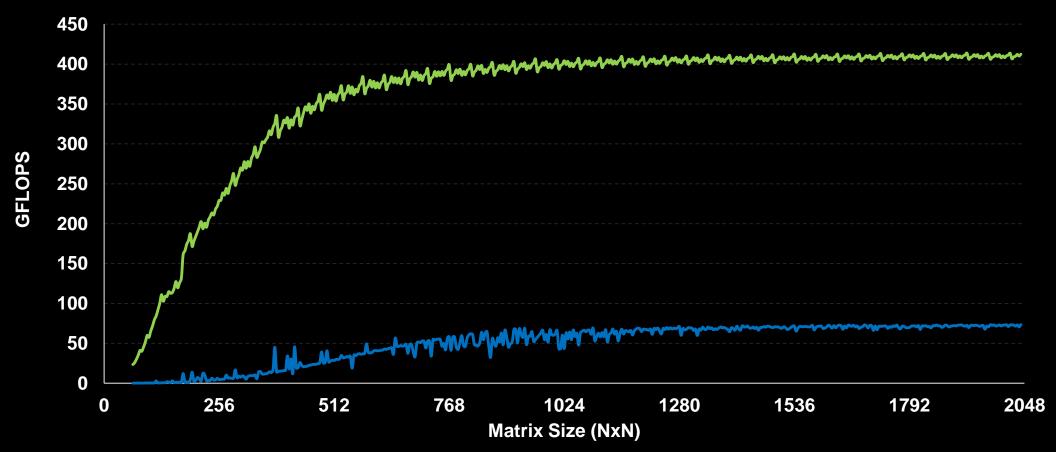
• cuBLAS 4.1, Tesla M2090 (Fermi), ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 16

Speedup over MKL

Performance may vary based on OS version and motherboard configuration

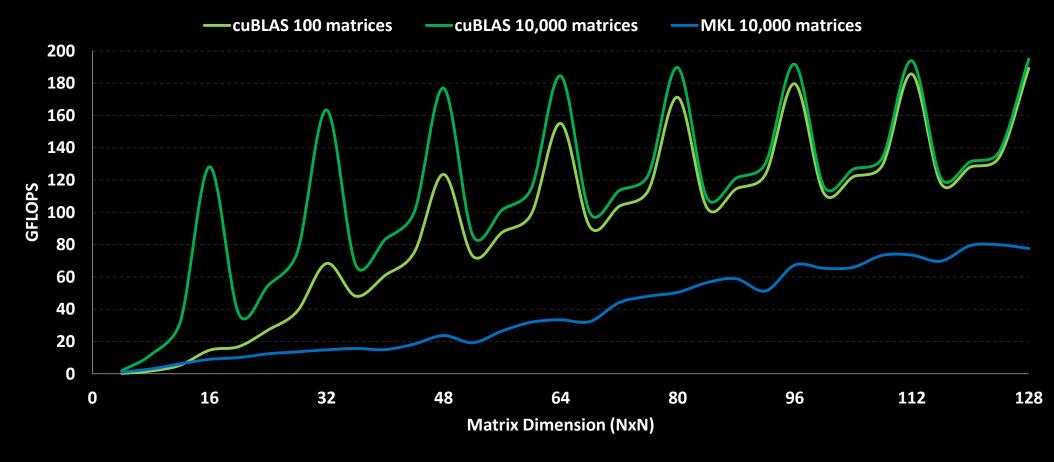
### **ZGEMM Performance vs Intel MKL**



• cuBLAS 4.1 on Tesla M2090, ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 17

# cuBLAS Batched GEMM API improves performance on batches of small matrices



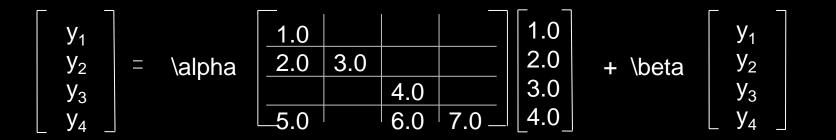
Performance may vary based on OS version and motherboard configuration

• cuBLAS 4.1 on Tesla M2090, ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 18

### cuSPARSE: Sparse linear algebra routines

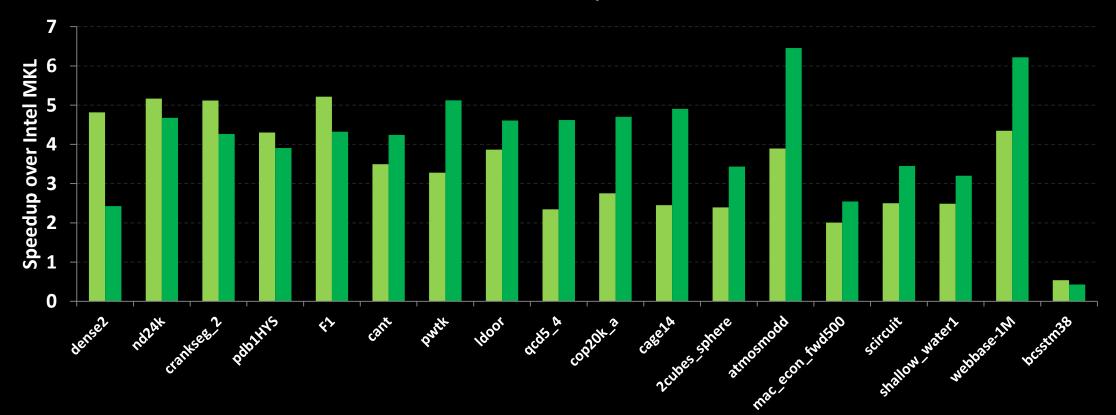
- Sparse matrix-vector multiplication & triangular solve
  - APIs optimized for iterative methods
- New in 4.1
  - Tri-diagonal solver with speedups up to 10x over Intel MKL
  - ELL-HYB format offers 2x faster matrix-vector multiplication



### cuSPARSE is >6x Faster than Intel MKL

#### **Sparse Matrix x Dense Vector Performance**

csrmv\* hybmv\*

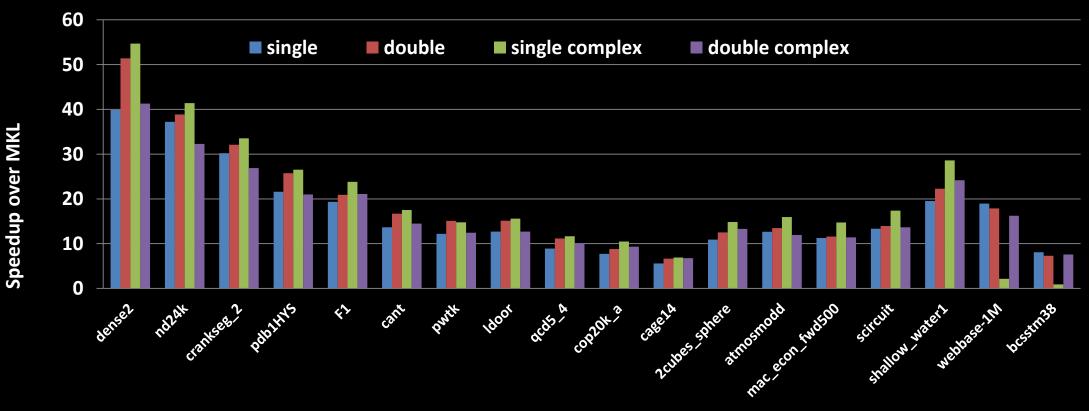


\*Average speedup over single, double, single complex & double-complex Performance may vary based on OS version and motherboard configuration

•cuSPARSE 4.1, Tesla M2090 (Fermi), ECC on • MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 20

### Up to 40x faster with 6 CSR Vectors

cuSPARSE Sparse Matrix x 6 Dense Vectors (csrmm) Useful for block iterative solve schemes



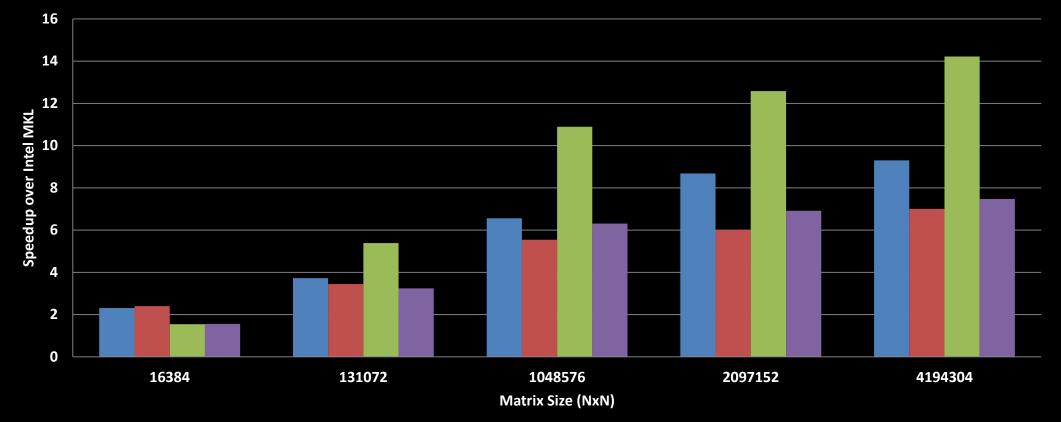
Performance may vary based on OS version and motherboard configuration

cuSPARSE 4.1, Tesla M2090 (Fermi), ECC on
MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 21

### Tri-diagonal solver performance vs. MKL

Speedup for Tri-Diagonal solver (gtsv)\*

■ single ■ double ■ complex ■ double complex



#### \*Parallel GPU implementation does not include pivoting

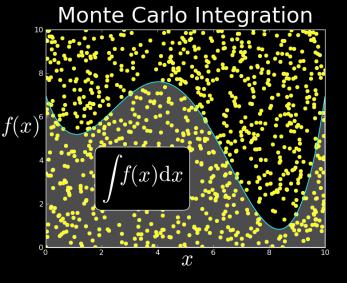
Performance may vary based on OS version and motherboard configuration

• cuSPARSE 4.1, Tesla M2090 (Fermi), ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 22

### cuRAND: Random Number Generation

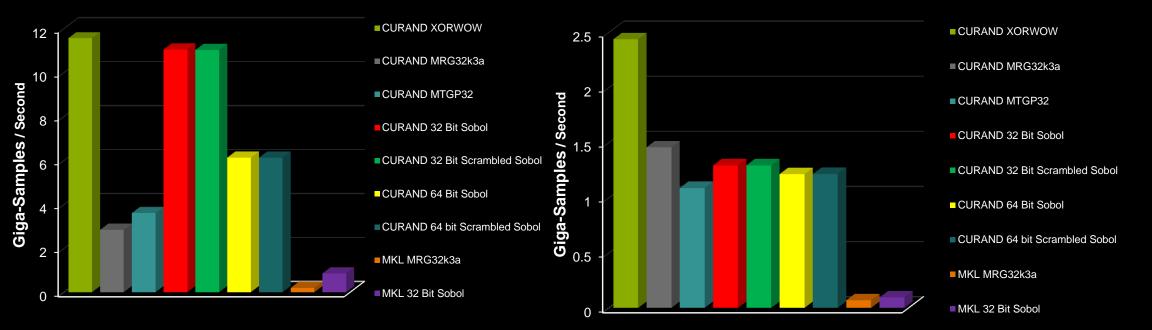
- Pseudo- and Quasi-RNGs
- Supports several output distributions
- Statistical test results reported in documentation
- New commonly used RNGs in CUDA 4.1
  - MRG32k3a RNG
  - MTGP11213 Mersenne Twister RNG



# cuRAND Performance compared to Intel MKL

Double Precision Uniform Distribution

#### Double Precision Normal Distribution



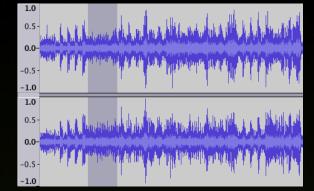
Performance may vary based on OS version and motherboard configuration

•cuRAND 4.1, Tesla M2090 (Fermi), ECC on • MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 @ 3.33 GHz 24

### 1000+ New Imaging Functions in NPP 4.1 Up to 40x speedups



- NVIDIA Performance Primitives (NPP) library includes over 2200 GPU-accelerated functions for image & signal processing Arithmetic, Logic, Conversions, Filters, Statistics, etc.
- Most are 5x-10x faster than analogous routines in Intel IPP





http://developer.nvidia.com/content/graphcuts-using-npp

\* NPP 4.1, NVIDIA C2050 (Fermi) \* IPP 6.1, Dual Socket Core™ i7 920 @ 2.67GHz

### Thrust: CUDA C++ Template Library

- Template library for CUDA mimics the C++ STL
  - Optimized algorithms for sort, reduce, scan, etc.
  - OpenMP backend for portability

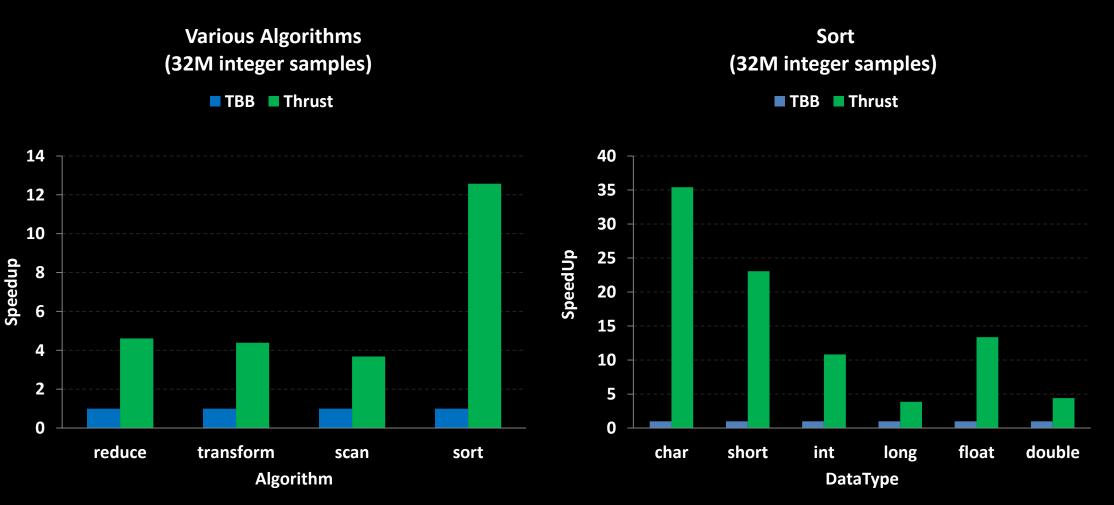
Allows applications and prototypes to be built quickly

### New in 4.1: Boost-style placeholders allow inline functors

Example: saxpy in 1 line:

thrust::transform(x.begin(), x.end(), y.begin(), y.begin(), a \* \_1 + \_2);

### **Thrust Performance compared to Intel TBB**



Performance may vary based on OS version and motherboard configuration

• Thrust 4.1, Tesla M2090 (Fermi), ECC on

• MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz 27

# math.h: C99 floating-point library + extras

Basic: +, \*, /, 1/, sqrt, FMA (all IEEE-754 accurate for float, double, all rounding modes)
Exponentials: exp, exp2, log, log2, log10, ...

- Trigonometry: sin, cos, tan, asin, acos, atan2, sinh, cosh, asinh, acosh, ...
- Special functions: lgamma, tgamma, erf, erfc
- Utility: fmod, remquo, modf, trunc, round, ceil, floor, fabs, ...
- Extras: rsqrt, rcbrt, exp10, sinpi, sincos, cospi, erfinv, erfcinv, ...

### New in 4.1

- Bessel functions: j0, j1, jn, y0, y1, yn
- Scaled complementary error function: erfcx
- Average and rounded average: \_\_{u}hadd, \_\_{u}rhadd

# GPU Technology Conference 2012 May 14-17 | San Jose, CA

#### The one event you can't afford to miss

- Learn about leading-edge advances in GPU computing
- Explore the research as well as the commercial applications
- Discover advances in computational visualization
- Take a deep dive into parallel programming

#### Ways to participate

- Speak share your work and gain exposure as a thought leader
- Register learn from the experts and network with your peers
- Exhibit/Sponsor promote your company as a key player in the GPU ecosystem

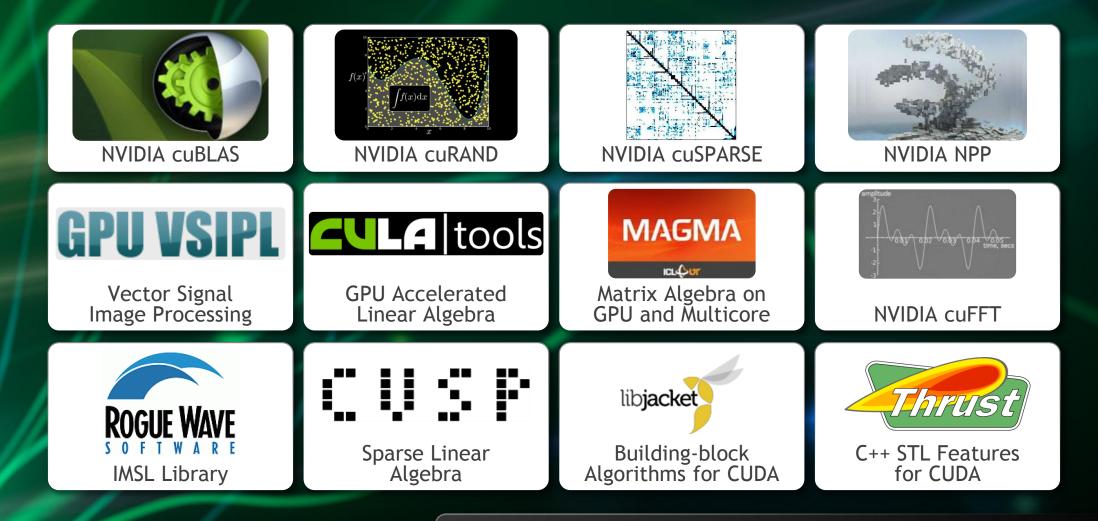




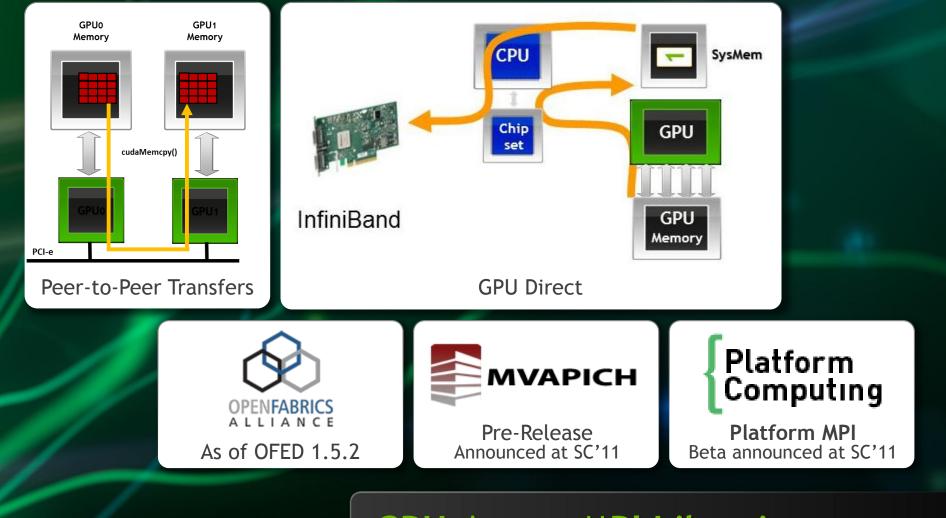
# **ADDITIONAL SLIDES...**

© NVIDIA Corporation 2011

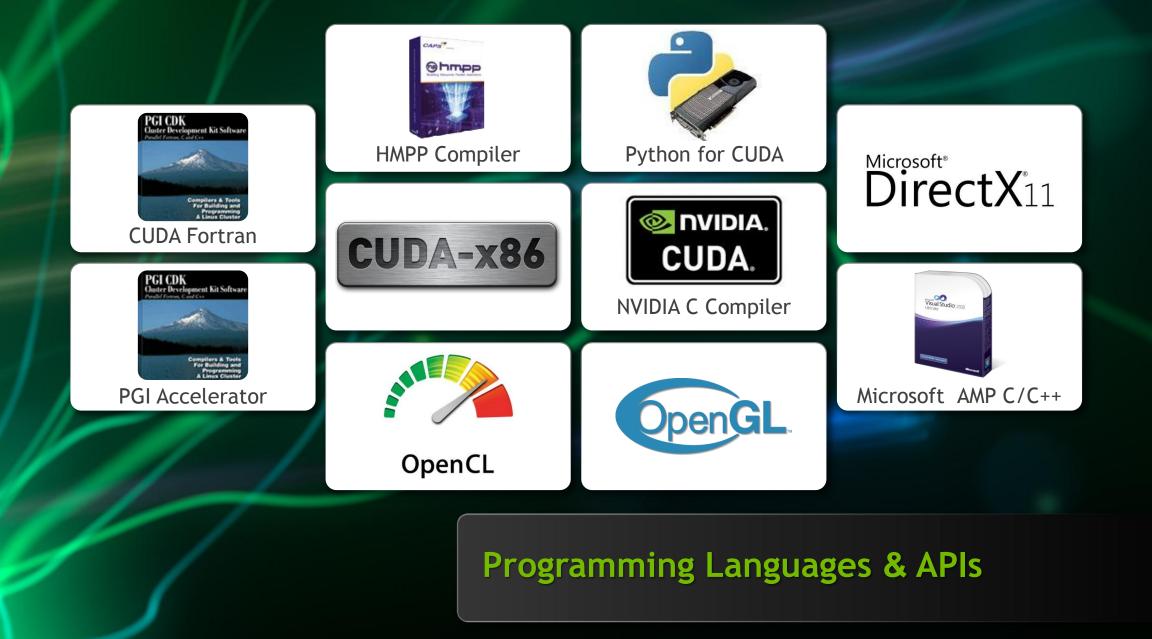




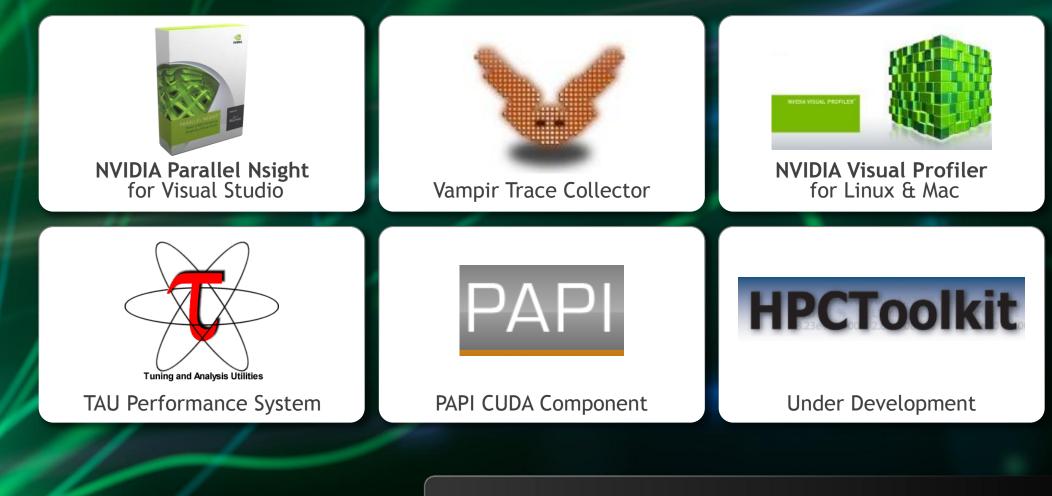
GPU Accelerated Libraries "Drop-in" Acceleration for Your Applications



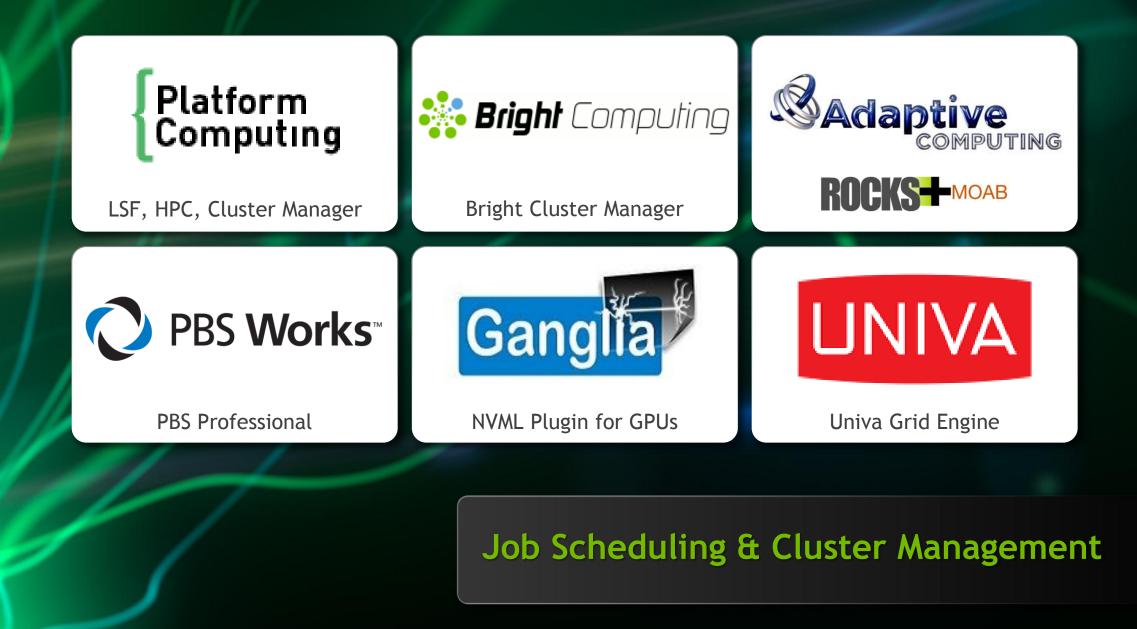
**GPU-Aware MPI Libraries** Integrated Support for GPU Computing







**Performance Analysis Tools** Single GPU to Hybrid Cluster Solutions



### **GPU Technology Conference Worldwide Events**

GTC Asia, Beijing, December 14-15, 2011 Focusing on the very latest scientific research and commercial applications in GPU computing.



GTC 2012, San Jose, CA, May 14-17, 2012 Advancing awareness of High Performance Computing and the transformational impact of GPUs.



www.gputechconf.com

# GPU Technology Conference 2012 May 14-17 | San Jose, CA

#### The one event you can't afford to miss

- Learn about leading-edge advances in GPU computing
- Explore the research as well as the commercial applications
- Discover advances in computational visualization
- Take a deep dive into parallel programming

#### Ways to participate

- Speak share your work and gain exposure as a thought leader
- Register learn from the experts and network with your peers
- Exhibit/Sponsor promote your company as a key player in the GPU ecosystem





#### Co-located with GTC 2012...

### Accelerated High Performance Computing Symposium (AHPC) Hosted by Los Alamos National Laboratory & NVIDIA

- Learn how accelerator technologies can be leveraged in innovative ways to advance the state-of-theart for simulations on large-scale systems
- Identify hardware and software requirements that can meet the requirements of power, scalability and fault tolerance needed for the next generation of HPC
- Understand how legacy codes can be adapted to make use of modern computing architectures
- Provide feedback to the vendor community to aid in the adoption of accelerator technologies

"The growing success of GTC makes it a natural venue for co-hosting the Accelerated HPC Symposium. This event draws senior scientists from national research labs across the globe, and their interests in hardware and software development make for a perfect match with GTC."

~Ben Bergen, Research Scientist, Los Alamos National Laboratory

Sign up for announcements at www.gputechconf.com



### **CUDA 4.1**



### The 'Super' Computing Company From Super Phones to Super Computers

6.00