CUDA 6.5
Performance Report

September 2014

NVIDIA

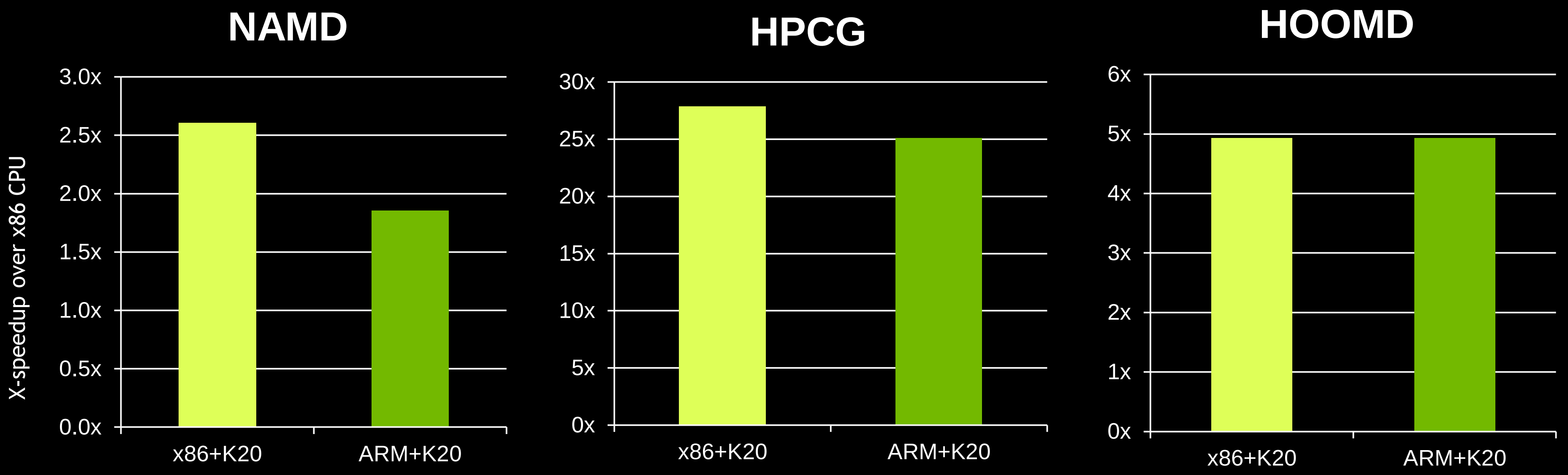
CUDA 6.5 Performance Report

- **CUDART** CUDA Runtime Library
- **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
- **cuDNN** Deep Neural Net building blocks

Included in the CUDA Toolkit (free download):
developer.nvidia.com/cuda-toolkit

For more information on CUDA libraries:
developer.nvidia.com/gpu-accelerated-libraries

ARM64+GPU: Early Benchmarks



Application Workload Profile

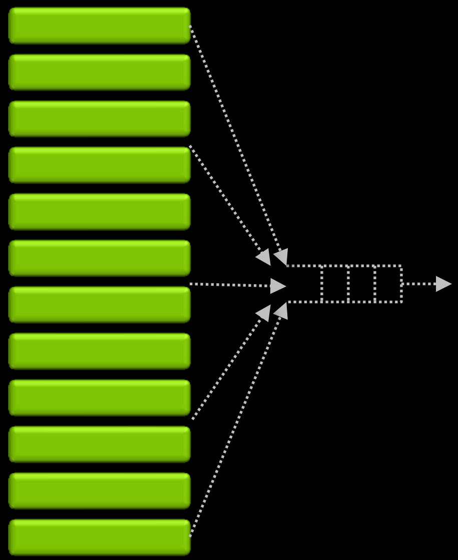
Recompiled to ARM64 (No optimizations), ARM: Applied Micro X-GENE CPU, X86: Xeon E5-2697, X86*: E5-2687W @ 3.10GHz

CUDA Multi-process server (MPS)

Concurrent execution of GPU tasks from >1 MPI Rank

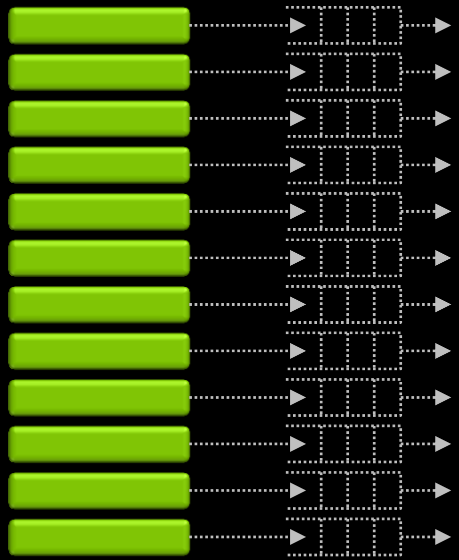
FERMI

1 Queue for Tasks from all MPI Ranks

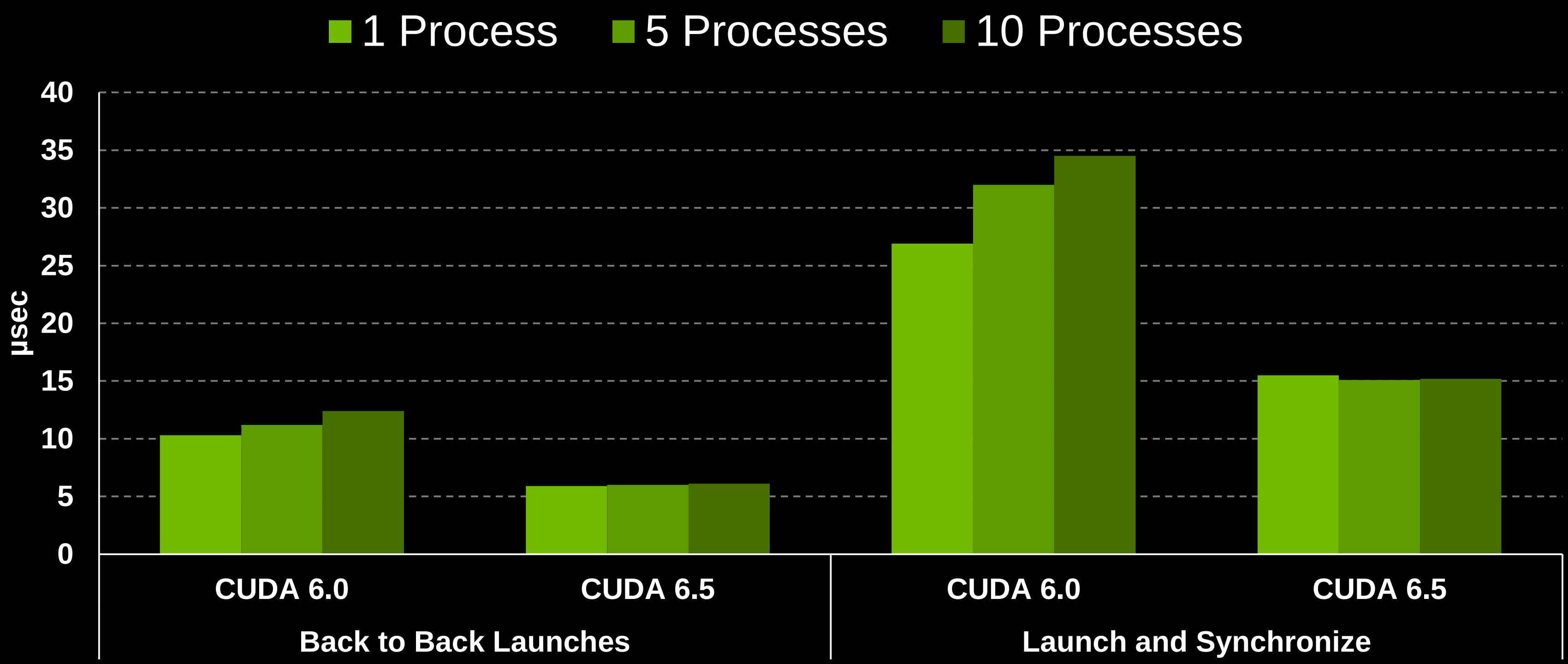


KEPLER

32 Parallel Queues for MPI Tasks



MPS kernel Launches: 1.7x to 2.0x faster




```
a<<<...>>>;
a<<<...>>>;
```

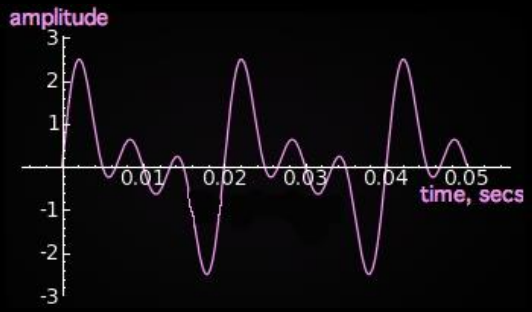
```
a<<<...>>>;
cudaDeviceSynchronize();
```

cuFFT: Multi-dimensional FFTs

- Real and complex
- Single- and double-precision data types
- 1D, 2D and 3D batched transforms
- Flexible input and output data layouts
- XT interface supports dual-GPU cards

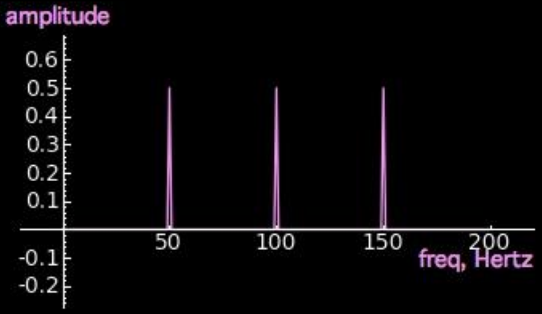
New in CUDA 6.5

- Device callbacks optimize use cases such as FFT + datatype conversion

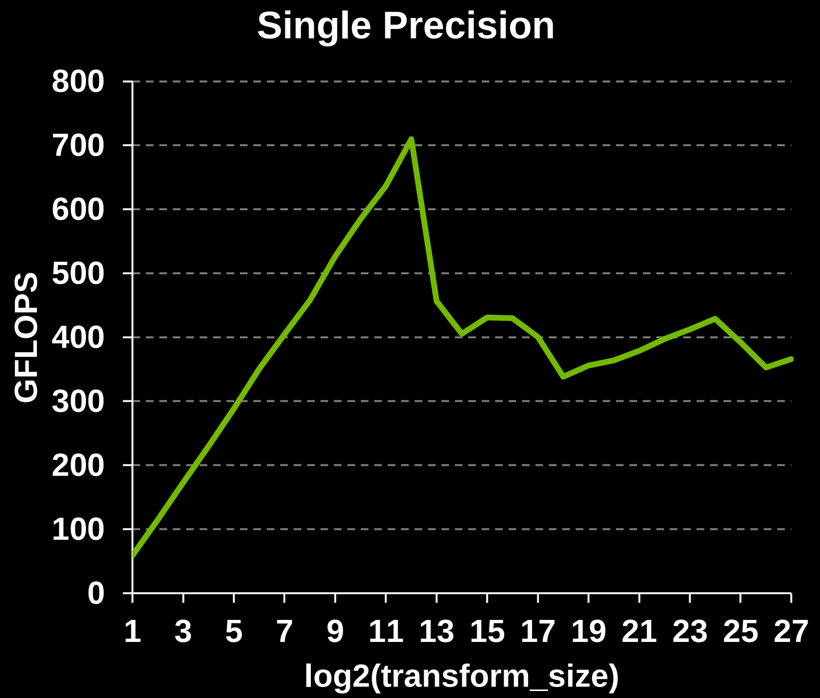


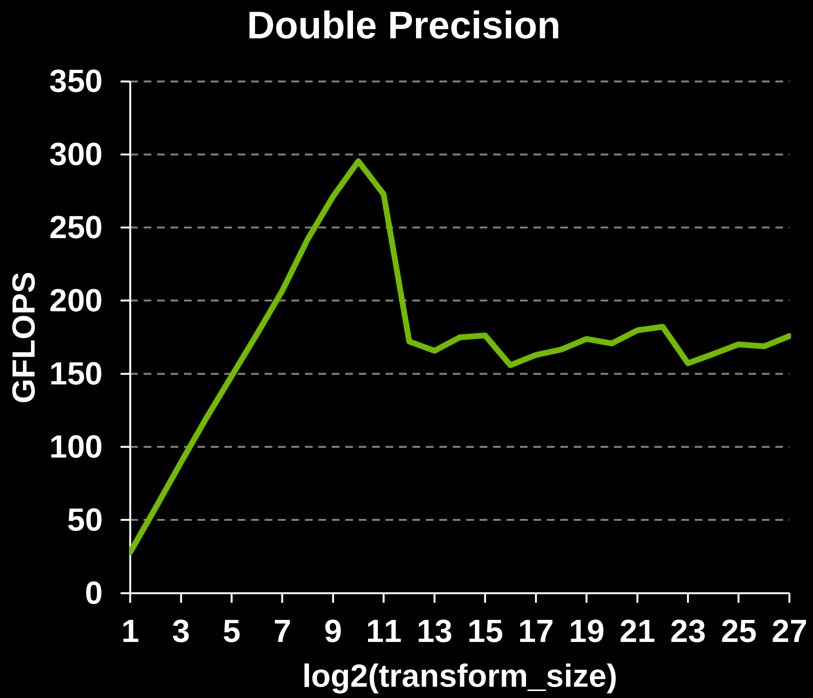
$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(x\frac{n}{N})}$$

$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(x) e^{j2\pi(x\frac{n}{N})}$$



cuFFT: up to 700 GFLOPS





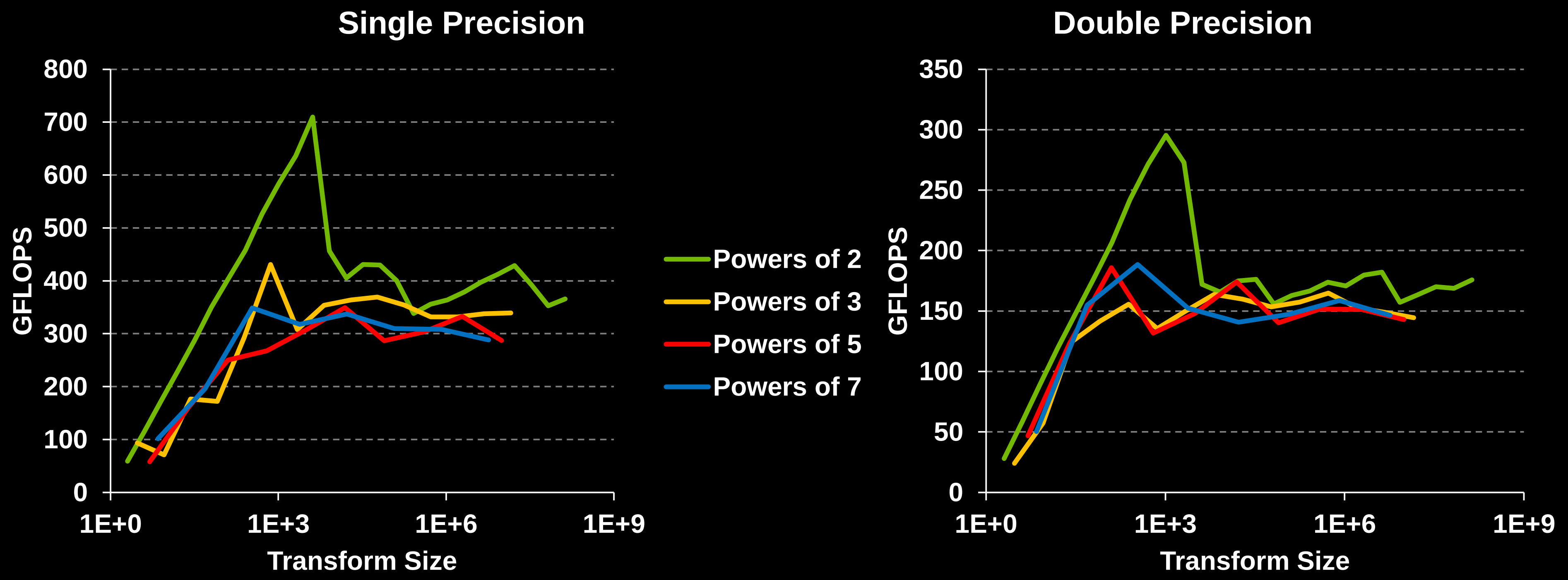
1D Complex, Batched FFTs

Used in Audio Processing and as a Foundation for 2D and 3D FFTs

- cuFFT 6.5 on K40c, ECC ON, 32M elements, input and output data on device
- Excludes time to create cuFFT "plans"

Performance may vary based on OS version and motherboard configuration

cuFFT: Consistently High Performance

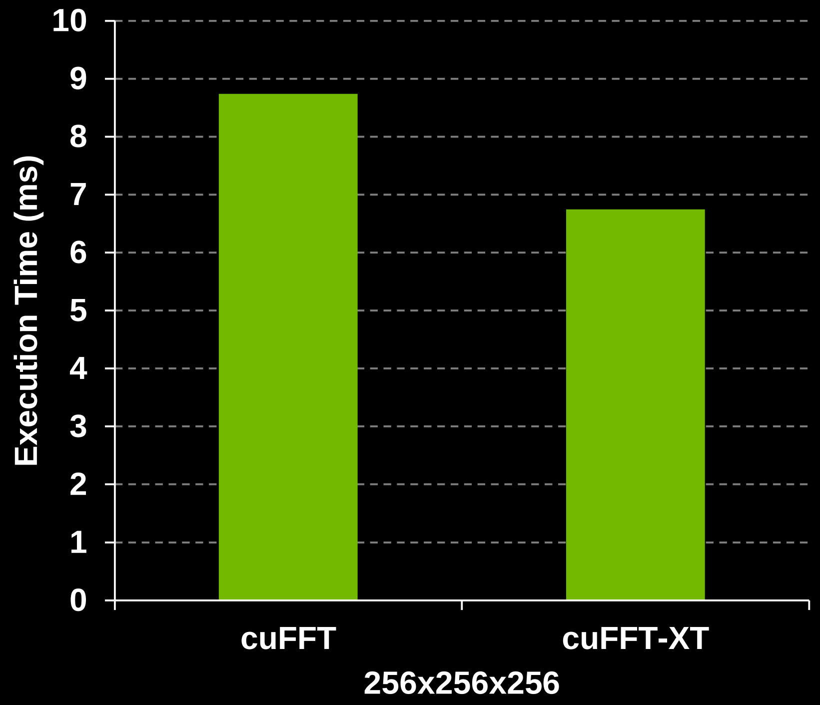


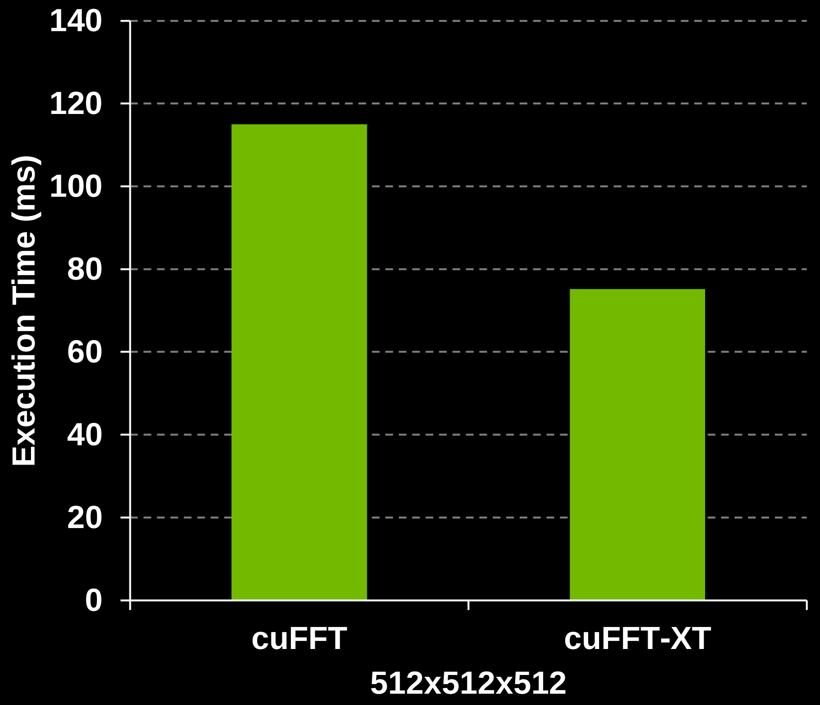
1D Complex, Batched FFTs
Used in Audio Processing and as a Foundation for 2D and 3D FFTs

Performance may vary based on OS version and motherboard configuration

- cuFFT 6.5 on K40c, ECC ON, 28M-33M elements, input and output data on device
- Excludes time to create cuFFT “plans”

cuFFT-XT on K10: 30% Faster on Large 3D FFTs





- cuFFT 6.5 on K10, ECC ON, input and output data on device
- Excludes time to create cuFFT "plans"

Performance may vary based on OS version and motherboard configuration

cuFFT User-Defined Callbacks

Without Callbacks: 3 kernels, 3 memory roundtrips

Read input → Convert to 32-bit → Write 32-bit → Read → Perform FFT → Write → Read FFT output → Convert to 8-bit → Write 8-bit data

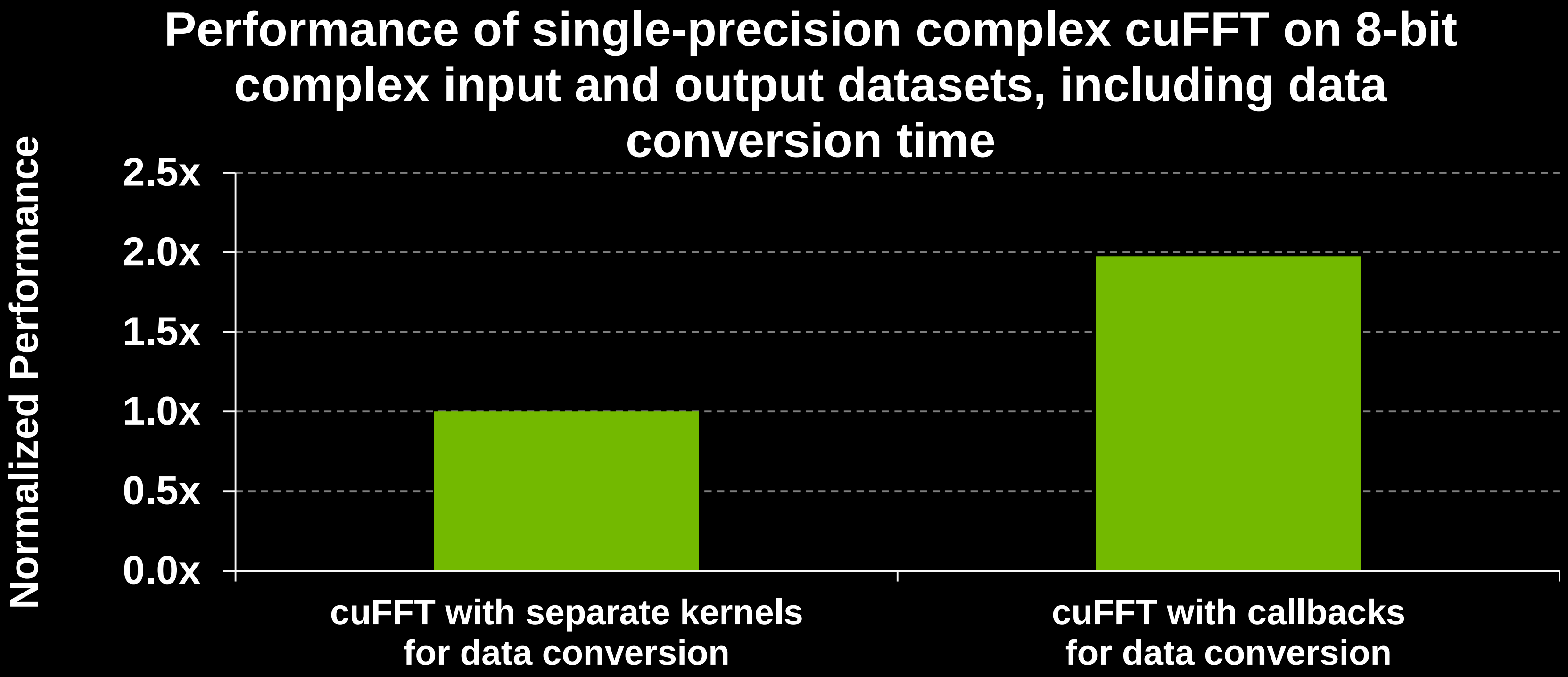
- Read input, Convert to 32-bit, Write 32-bit: User-defined Kernel
- Read, Perform FFT, Write: cuFFT Internal Kernel
- Read FFT output, Convert to 8-bit, Write 8-bit data: User-defined Kernel

With Callbacks: 1 kernel, 1 memory roundtrip

Read 8-bit input → Convert to 32-bit → Perform FFT → Convert to 8-bit → Write output

Merged into a single kernel

Performance of cuFFT Callbacks



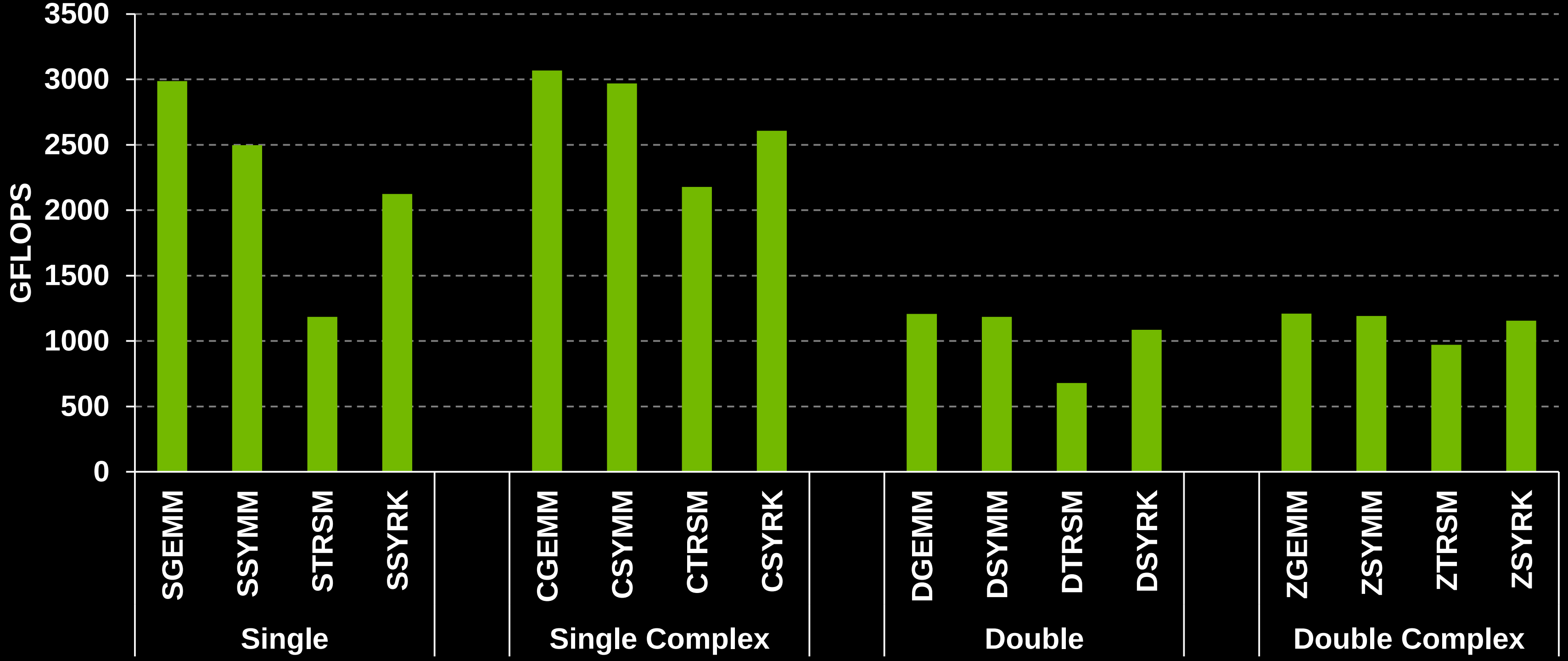
- cuFFT 6.5 on K40, ECC ON, 512 1D C2C forward trasforms, 32M total elements
- Input and output data on device, excludes time to create cuFFT “plans”

cuBLAS: Dense Linear Algebra on GPUs

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

- XT Interface for Level 3 BLAS
  - Distributed computations across multiple GPUs
  - Out-of-core streaming to GPU, no upper limit on matrix size
  - “Drop-in” BLAS intercepts CPU BLAS calls, streams to GPU

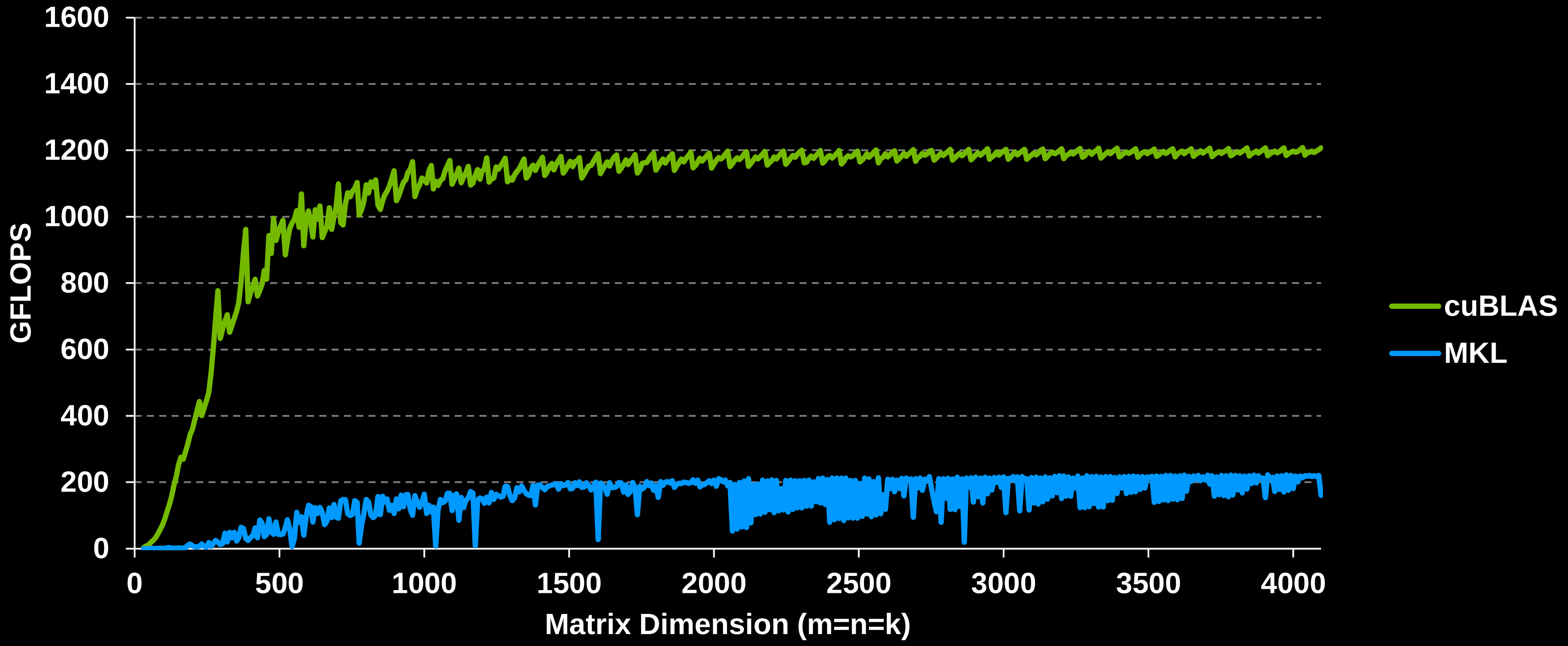
cuBLAS: >3 TFLOPS single-precision >1 TFLOPS double-precision



- cuBLAS 6.5 on K40m, ECC ON, input and output data on device
- m=n=k=4096, transpose=no, side=right, fill=lower

Performance may vary based on OS version and motherboard configuration

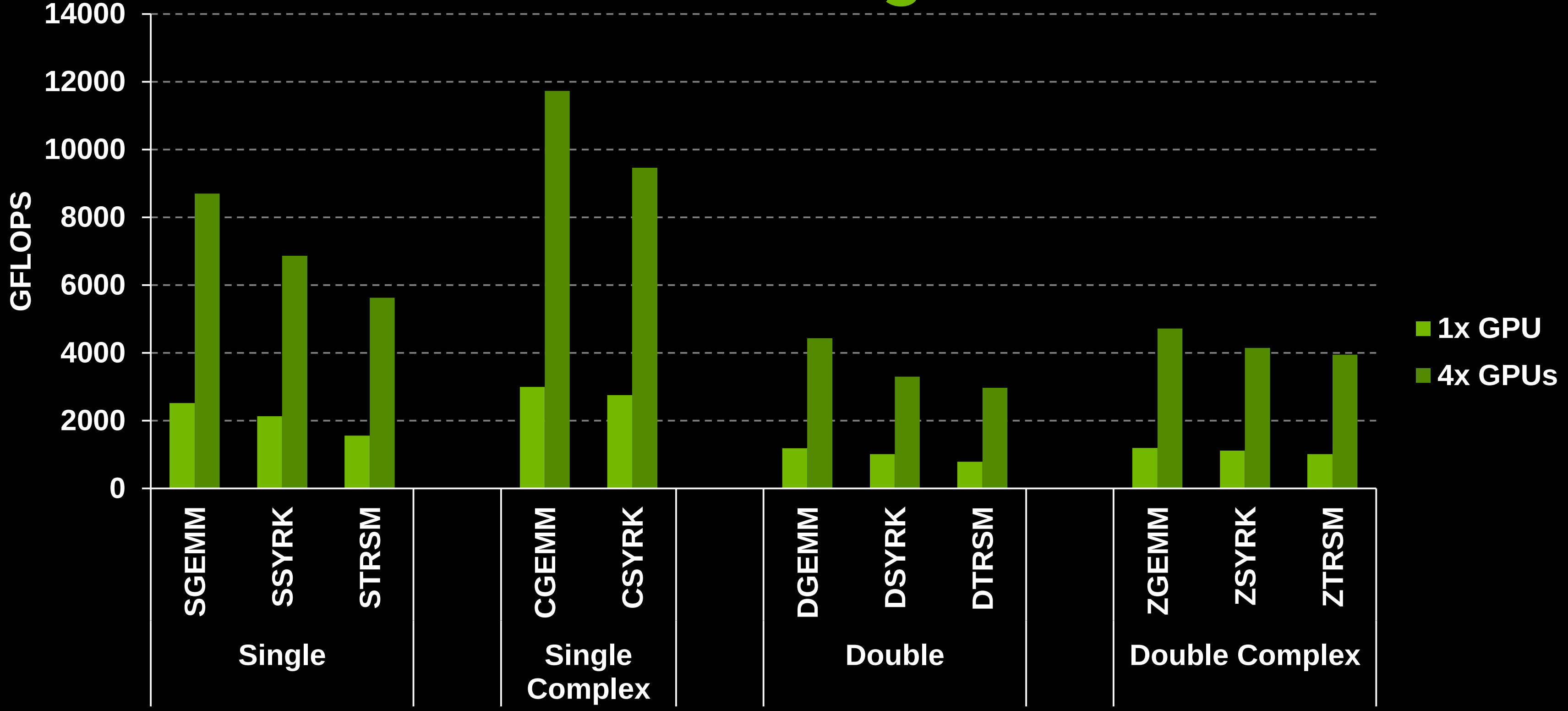
cuBLAS: ZGEMM 6x Faster than MKL



- cuBLAS 6.5 on K40m, ECC ON, input and output data on device
- MKL 11.0.4 on Intel IvyBridge single socket 12-core E5-2697 v2 @ 2.70GHz

Performance may vary based on OS version and motherboard configuration

cuBLAS-XT: Automatic Performance Boost
>10 TF on a single node



- cuBLAS 6.5 on K40, double precision mode
- Input and output data on device, m=n=k=16384, transpose=no

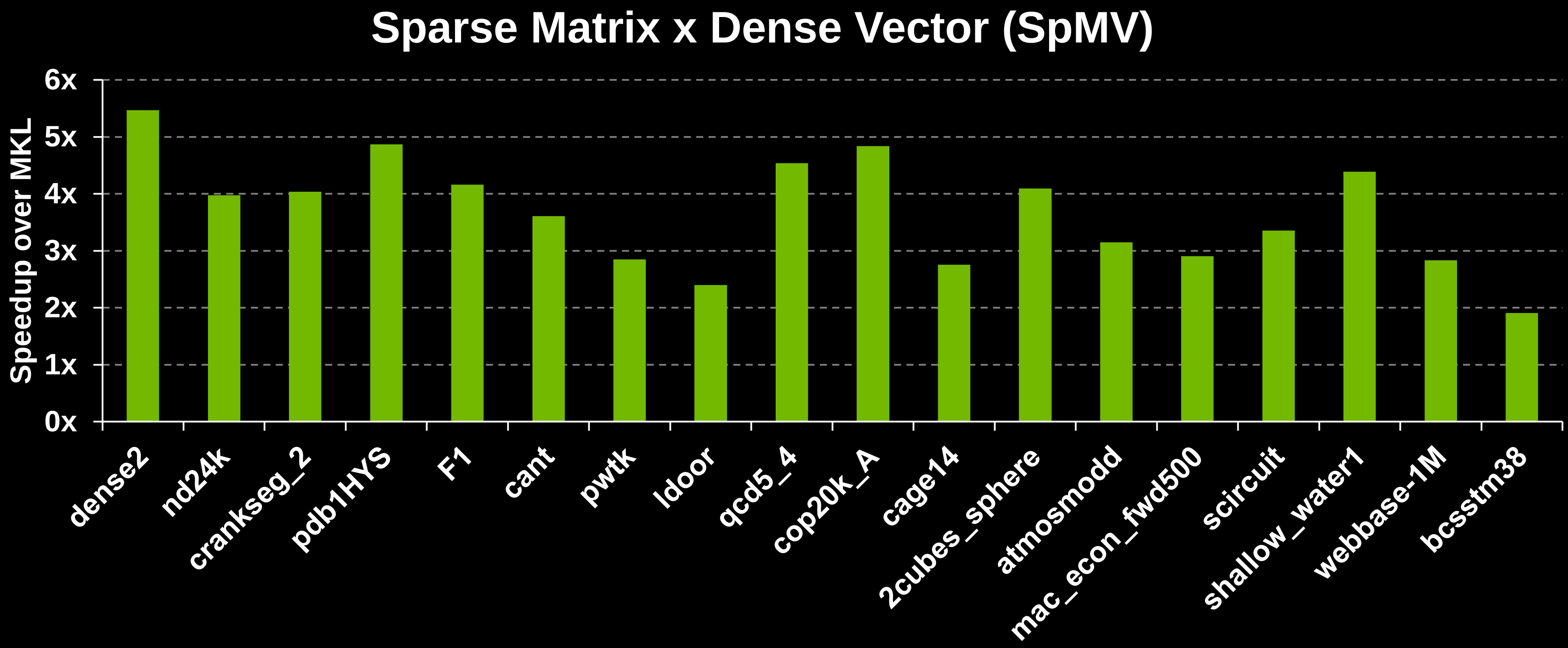
Performance may vary based on OS version and motherboard configuration

cuSPARSE: Sparse linear algebra routines

- **Optimized sparse linear algebra BLAS routines - matrix-vector, matrix-matrix, triangular solve**
- **Support for variety of formats (CSR, COO, block variants)**
- **Incomplete-LU and Cholesky preconditioners**

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \alpha \begin{bmatrix} 1.0 & & & \\ 2.0 & 3.0 & & \\ & & 4.0 & \\ 5.0 & & 6.0 & 7.0 \end{bmatrix} \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{bmatrix} + \beta \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

cuSPARSE: 5x Faster than MKL



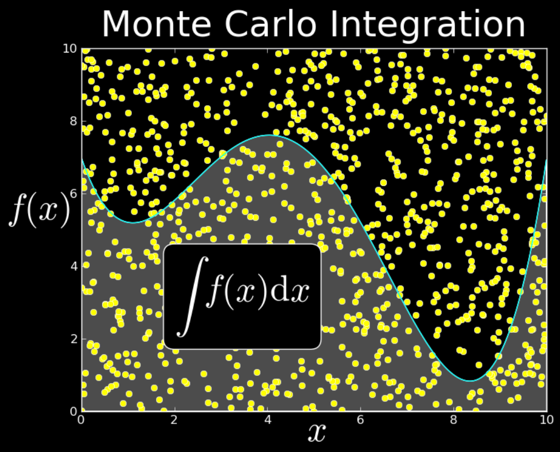
- Average of s/c/d/z routines
- cuSPARSE 6.5 on K40m, ECC ON, input and output data on device
- MKL 11.0.4 on Intel IvyBridge single socket 12-core E5-2697 v2 @ 2.70GHz
- Matrices obtained from: http://www.cise.ufl.edu/research/sparse/matrices/

Performance may vary based on OS version and motherboard configuration

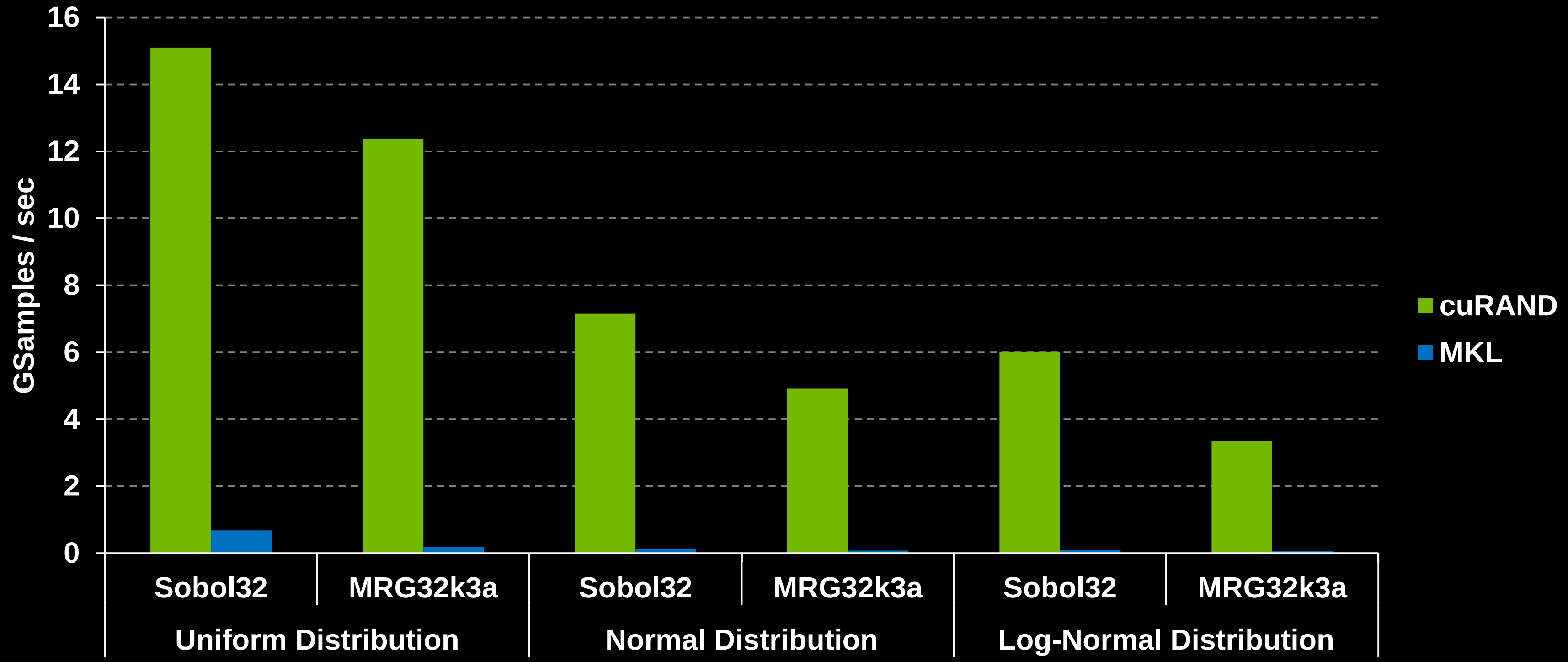
cuRAND: Random Number Generation

- Generating high quality random numbers in parallel is hard
  - Don't do it yourself, use a library!

- Pseudo- and Quasi-RNGs
- Mersenne Twister 19937
- Supports several output distributions
- Statistical test results in documentation



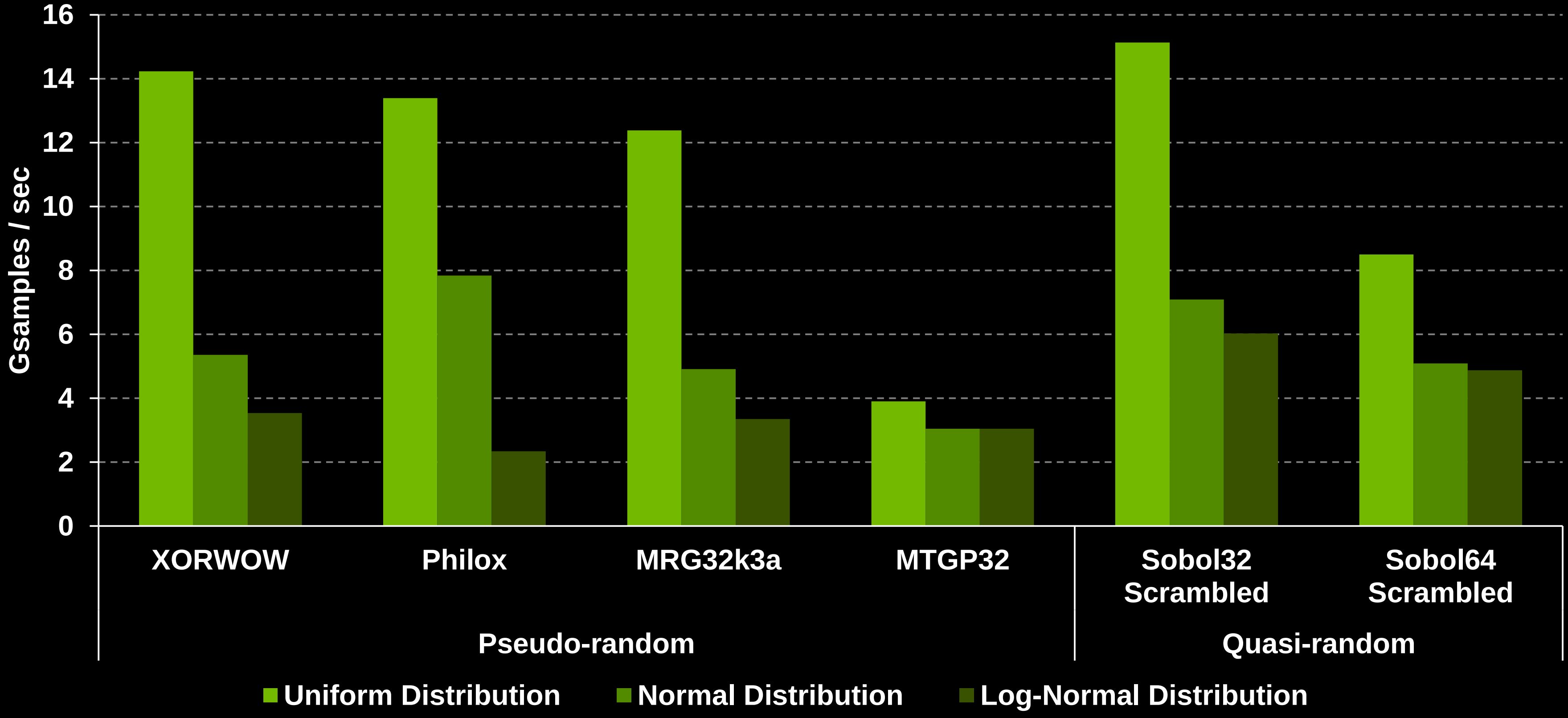
cuRAND: Up to 70x Faster vs. Intel MKL



- cuRAND 6.5 on K40c, ECC ON, double-precision input and output data on device
- MKL 11.0.4 on Intel IvyBridge single socket 12-core E5-2697 v2 @ 2.70GHz

Performance may vary based on OS version and motherboard configuration

cuRAND: High Performance RNGs



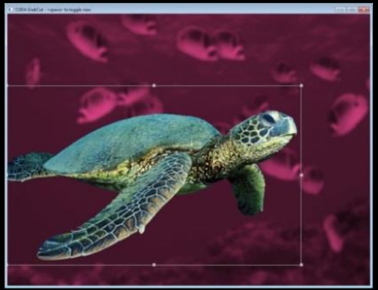
Performance may vary based on OS version and motherboard configuration

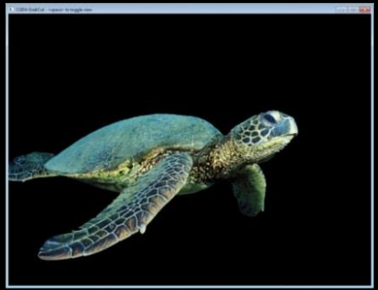
• cuRAND 6.5 on K40m, ECC ON, double precision input and output data on device

NPP: NVIDIA Performance Primitives

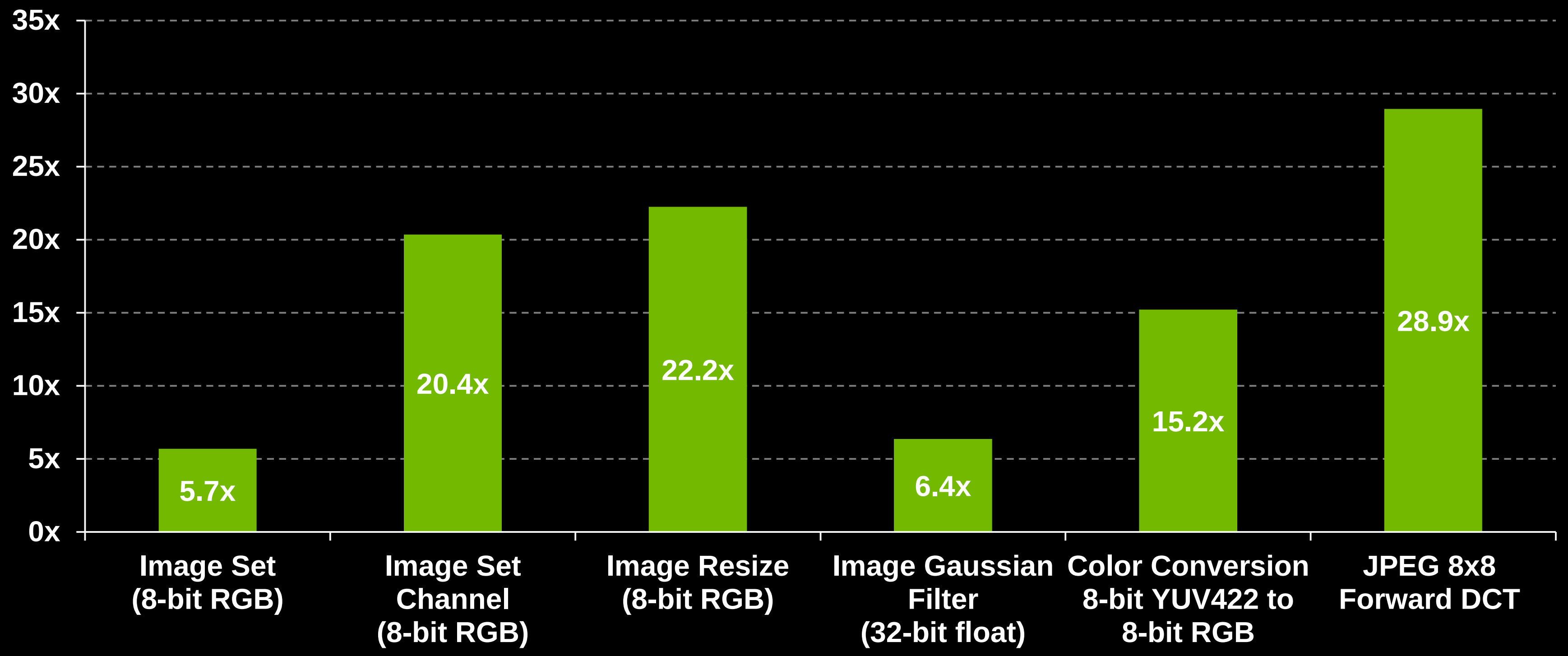
- Over 5000 image and signal processing routines:

  color transforms, geometric transforms, move operations, linear filters, image & signal statistics, image & signal arithmetic, JPEG building blocks, image segmentation, median filter, BGR/YUV conversion, 3D LUT color conversion





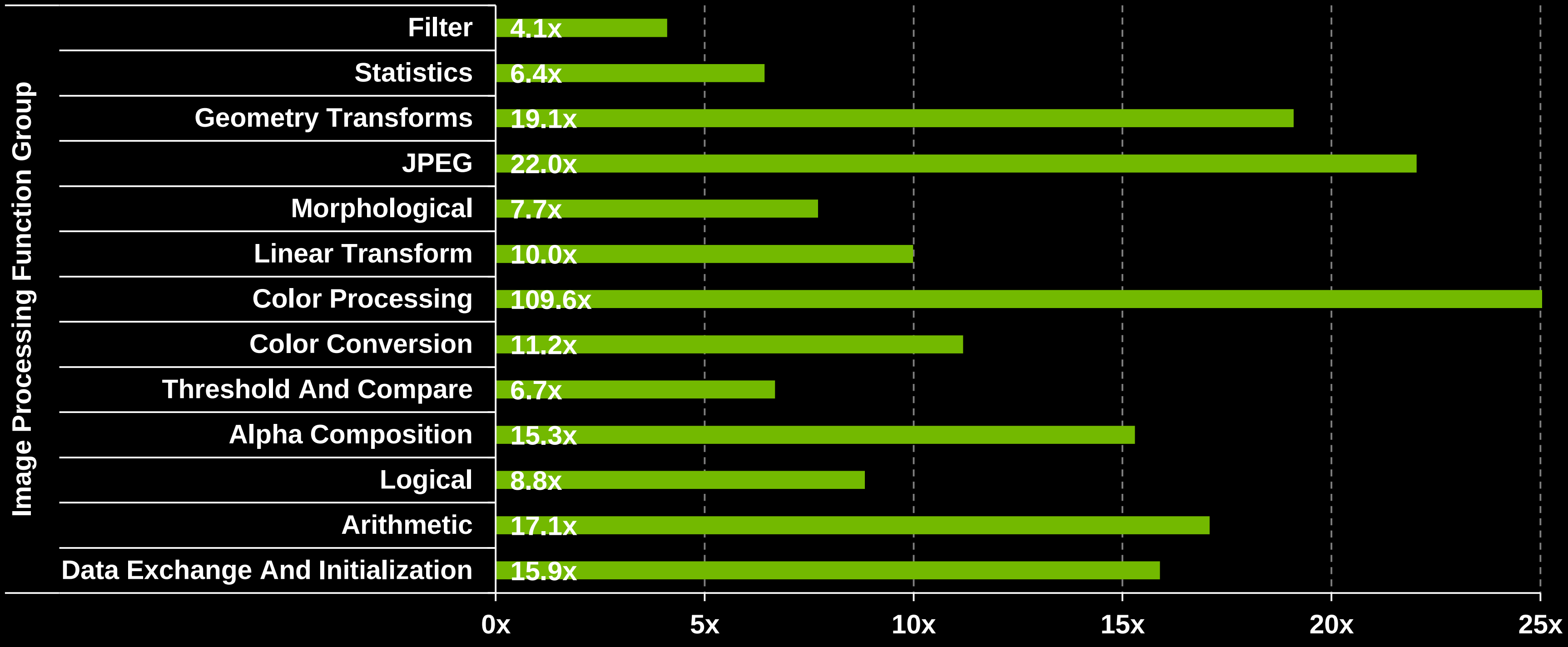
NPP Speedup vs. Intel IPP



- NPP 6.5 on K40m, input and output data on device
- IPP 7.0 on Intel IvyBridge single socket 12-core E5-2697 v2 @ 2.70GHz

Performance may vary based on OS version and motherboard configuration

NPP Speedup vs. Intel IPP



- NPP 6.5 on K40m, input and output data on device
- Each bar represents the average speedup over all routines in the function group
- IPP 7.0 on Intel IvyBridge single socket 12-core E5-2697 v2 @ 2.70GHz

Performance may vary based on OS version and motherboard configuration

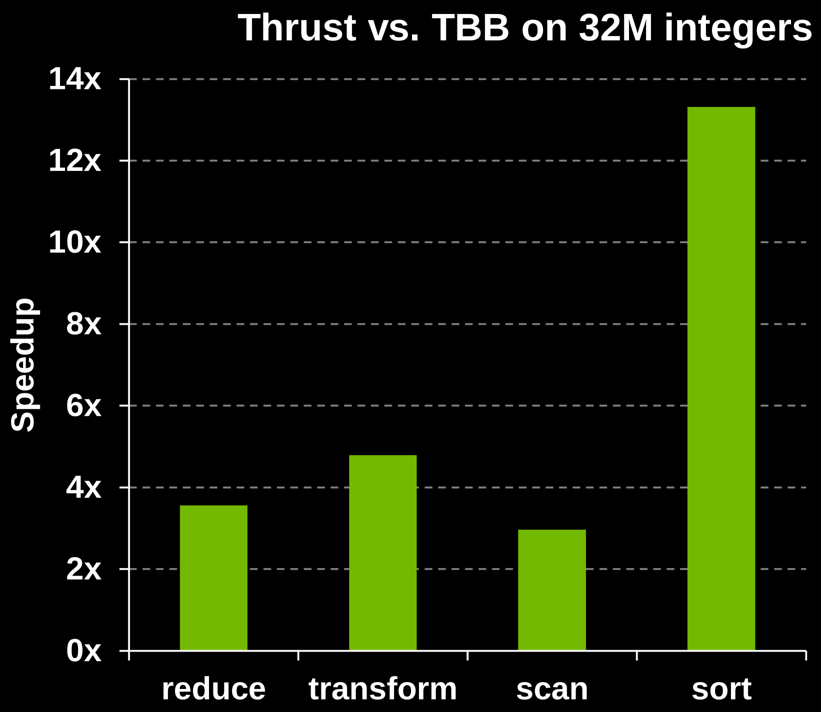
Thrust CUDA C++ Template Library

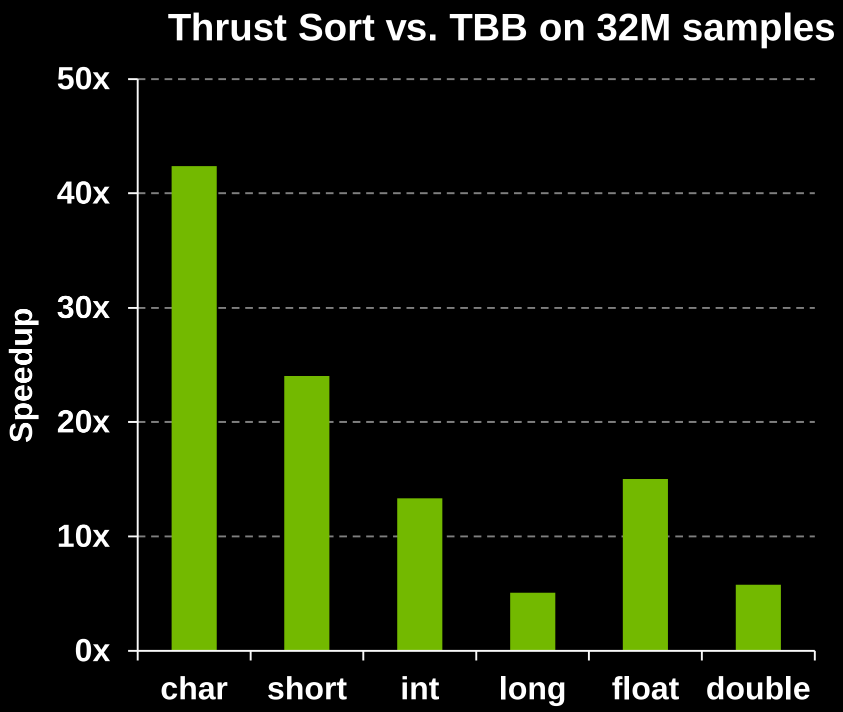
- Template library for CUDA C++
  - Host and Device Containers that mimic the C++ STL
  - Optimized Algorithms for sort, reduce, scan, etc.
  - OpenMP Backend for portability

- Also available on github: thrust.github.com

- Allows applications and prototypes to be built *quickly*

Thrust Performance vs. Intel TBB





- Thrust v1.7.2 on K40m, ECC ON, input and output data on device
- TBB 4.2 on Intel IvyBridge single socket 12-core E5-2697 v2 @ 2.70GHz

Performance may vary based on OS version and motherboard configuration

math.h: C99 floating-point library + extras

CUDA math.h is industry proven, high performance, accurate

- 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, ...
- Bessel: j0, j1, jn, y0, y1, yn, cyl_bessel_i0, cyl_bessel_i1
- Vector SIMD: vadd, vsub, vavrg, vabsdiff, vmin, vmax, vset
- Extras: rsqrt, rhypot, rcbrt, exp10, sinpi, sincos[pi], cospi, erf[c]inv, normcdf[inv]

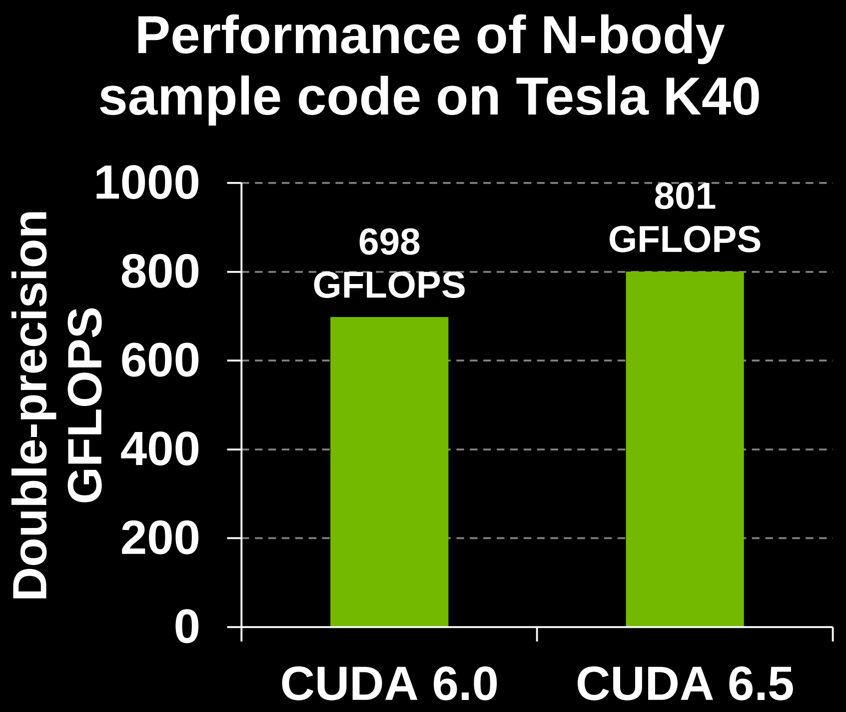
New in CUDA 6.5

- Performance improvements
  - [r]hypot[f], cbrt, sqrt, rsqrt, atan[f], exp[10][m1]f

Math Performance Improvements

RSQRT Improvement Increases N-Body Performance by 15%

- Significantly improved double-precision functions:
  - [r]sqrt(), [r]hypot(), cbrt(), atan(), acosh()
- Significantly improved single-precision functions:
  - [r]hypot(), atanf(), expf(), exp10f(), expm1f()



- CUDA 6.0 and CUDA 6.5 on K40m
- Sample code: http://docs.nvidia.com/cuda/cuda-samples/index.html#cuda-n-body-simulation

Introducing NVIDIA® cuDNN

- Forward and backward convolution routines tuned for NVIDIA GPUs
- Will be optimized for all future NVIDIA GPU generations
- Arbitrary dimension ordering, striding, and subregions for 4d tensors means easy integration into any neural net implementation
- Forward and backward paths for common layer types - ReLu, Sigmoid, Tanh, Pooling, Softmax

Download: https://developer.nvidia.com/cudnn
Contact: cudnn@nvidia.com

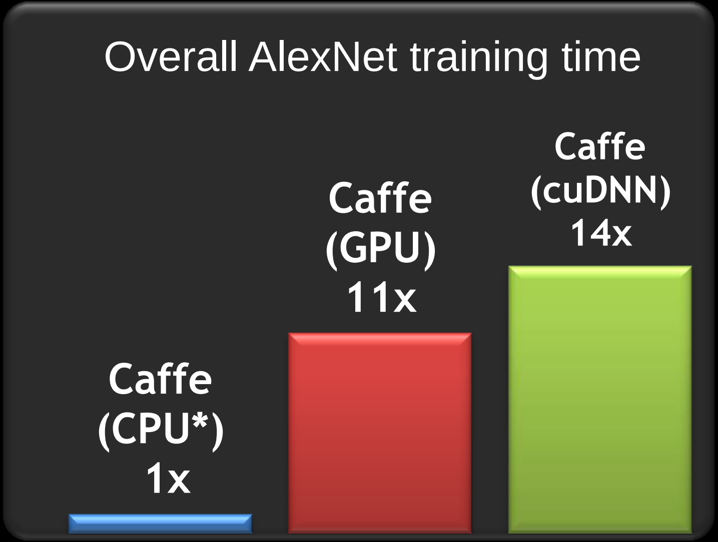
Using Caffe with cuDNN

- Accelerate Caffe layer types by 1.2 - 3x

    Example: AlexNet Layer 2 forward:

    1.9x faster convolution, 2.7x faster pooling

- Integrated into Caffe dev branch today!
(targeting official release with Caffe 1.0)



Baseline Caffe compared to Caffe accelerated by cuDNN on K40

*CPU is 24 core E5-2697v2 @ 2.4GHz
Intel MKL 11.1.3

CUDA Registered Developer Program

Sign up for free at: www.nvidia.com/paralleldeveloper

- Exclusive access to pre-release CUDA Installers
- Submit bugs and features requests to NVIDIA
- Keep informed about latest releases and training opportunities
- Access to exclusive downloads
- Exclusive activities and special offers