CUDA 7.0
Performance Report

May 2015

CUDA 7.0 Performance Report

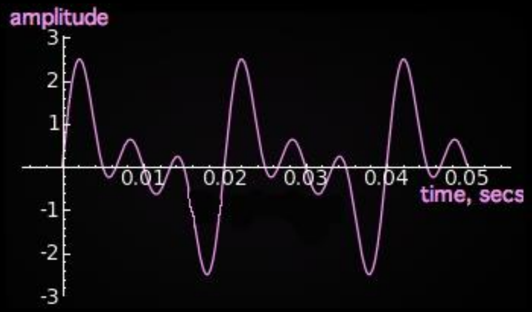
- cuFFT Fast Fourier Transforms Library
- cuBLAS Complete BLAS Library
- cuSPARSE Sparse Matrix Library
- cuSOLVER Linear Solver Library New in CUDA 7.0
- cuRAND Random Number Generation (RNG) Library
- NPP Performance Primitives for Image & Video Processing
- Thrust Templated Parallel Algorithms & Data 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

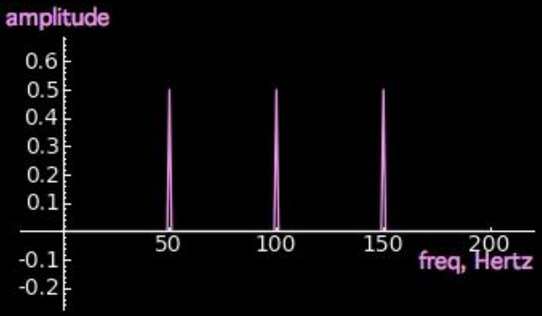
cuFFT: Multi-dimensional FFTs

- Real and complex, Single- and double-precision data types
- 1D, 2D and 3D batched transforms
- Flexible input and output data layouts
- Also supports simple “drop-in” replacement of a CPU FFTW library
- XT interface supports up to 4 GPUs
- Device callbacks optimize use cases such as FFT + datatype conversion

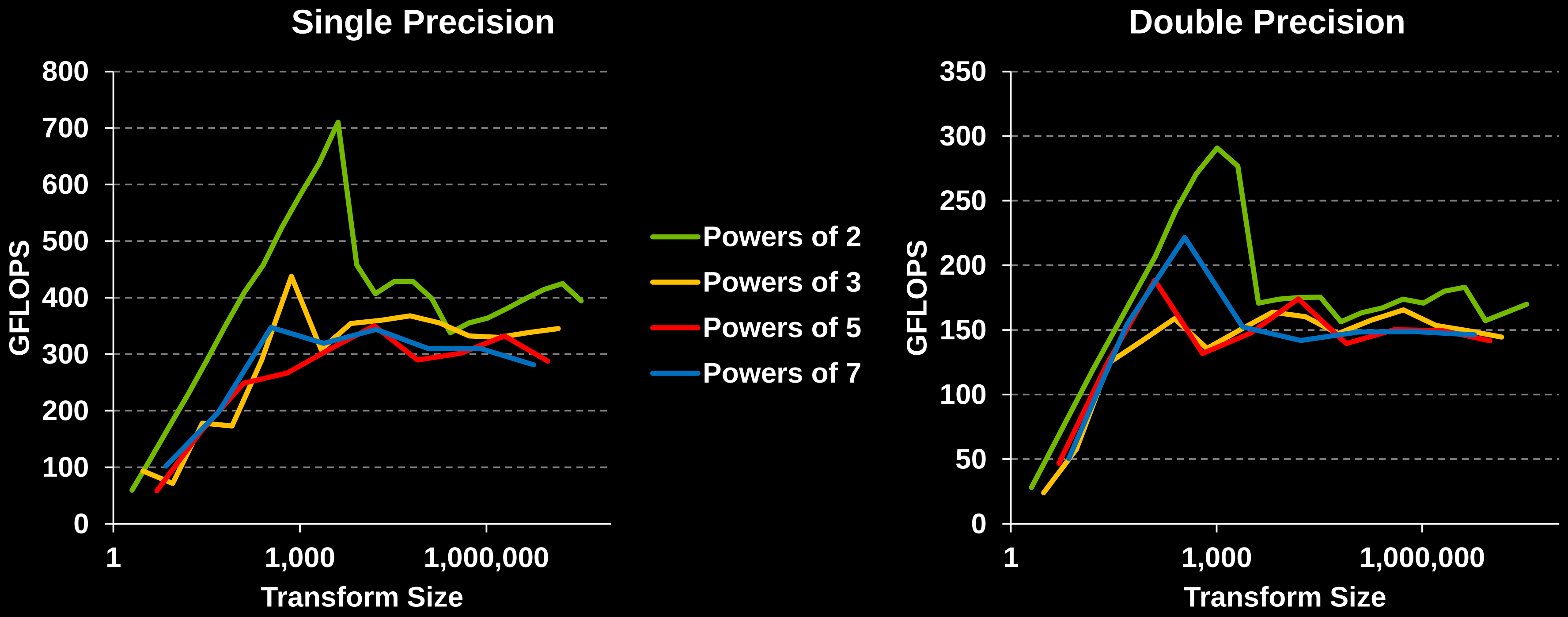


$$F(x) = \sum_{n=0}^{N-1} f(n)e^{-j2\pi(x\frac{n}{N})}$$

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



cuFFT: up to 700 GFLOPS



1D Complex, Batched FFTs

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

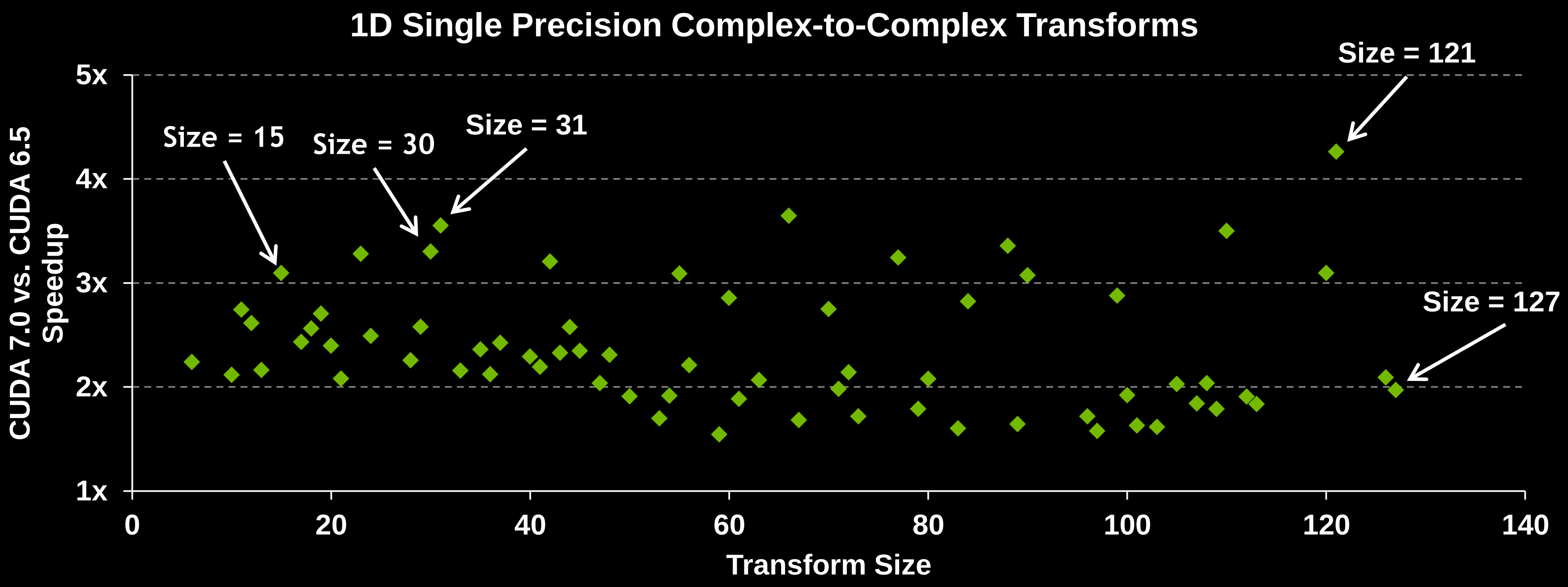
Performance may vary based on OS and software versions, and motherboard configuration

- cuFFT 7.0 on K40m, Base clocks, ECC ON
- Batched transforms on 28M-33M total elements, input and output data on device
- Excludes time to create cuFFT "plans"

New in CUDA 7.0

cuFFT up to 3x Faster

for sizes that are composites of small primes



- cuFFT 6.5 and 7.0 on K20m, ECC ON
- Batched transforms on 32M total elements, input and output data on device
- Excludes time to create cuFFT “plans”

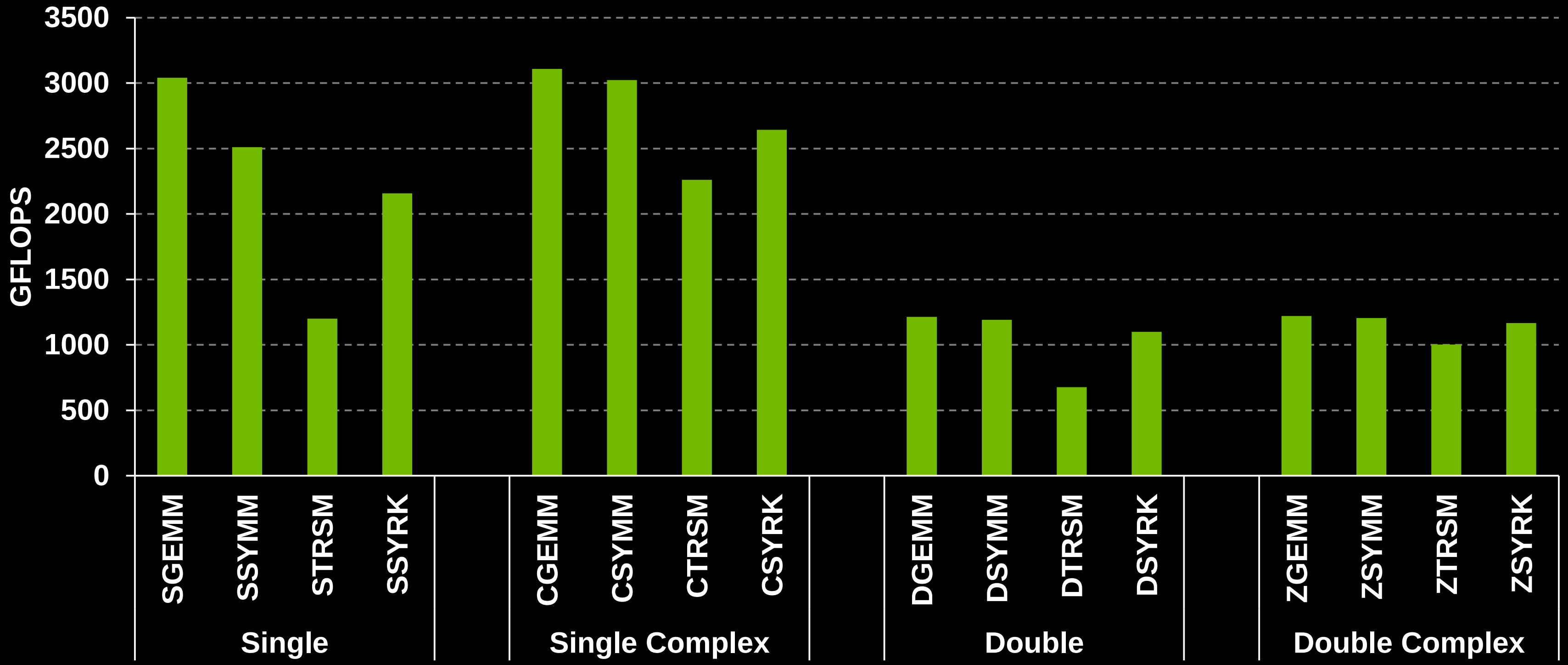
Performance may vary based on OS and software versions, and motherboard configuration

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
  - Batched routines for higher performance on small problem sizes

- 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

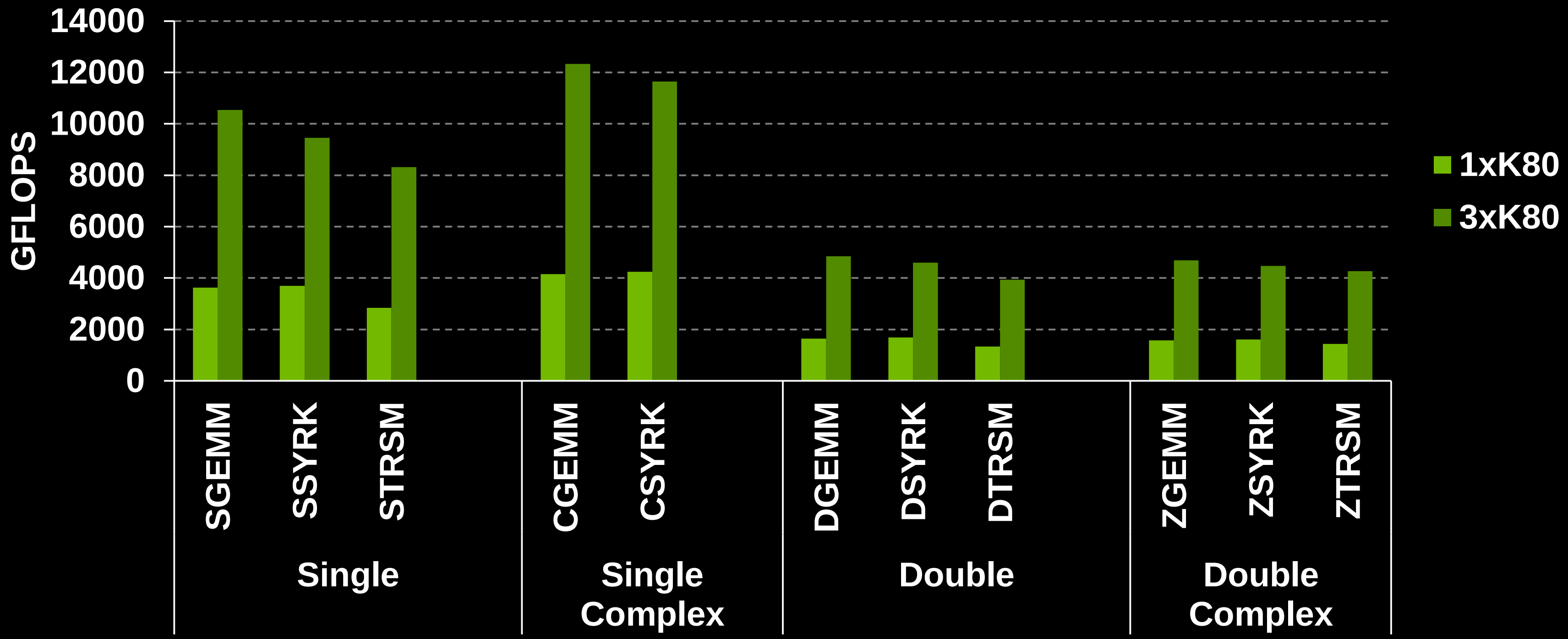
cuBLAS: >3 TFLOPS single-precision >1 TFLOPS double-precision



Performance may vary based on OS and software versions, and motherboard configuration

- cuBLAS 7.0 on K40m, Base clocks, ECC ON, input and output data on device
- m=n=k=4096, transpose=no, side=right, fill=lower

cuBLAS-XT: Multi-GPU Performance Scaling
>12 TF on a single node



Performance may vary based on OS and software versions, and motherboard configuration

- cuBLAS 7.0 on K80, Base clocks, ECC ON
- input and output data on host, m=n=k=32768, transpose=no

cuSPARSE: Sparse linear algebra routines

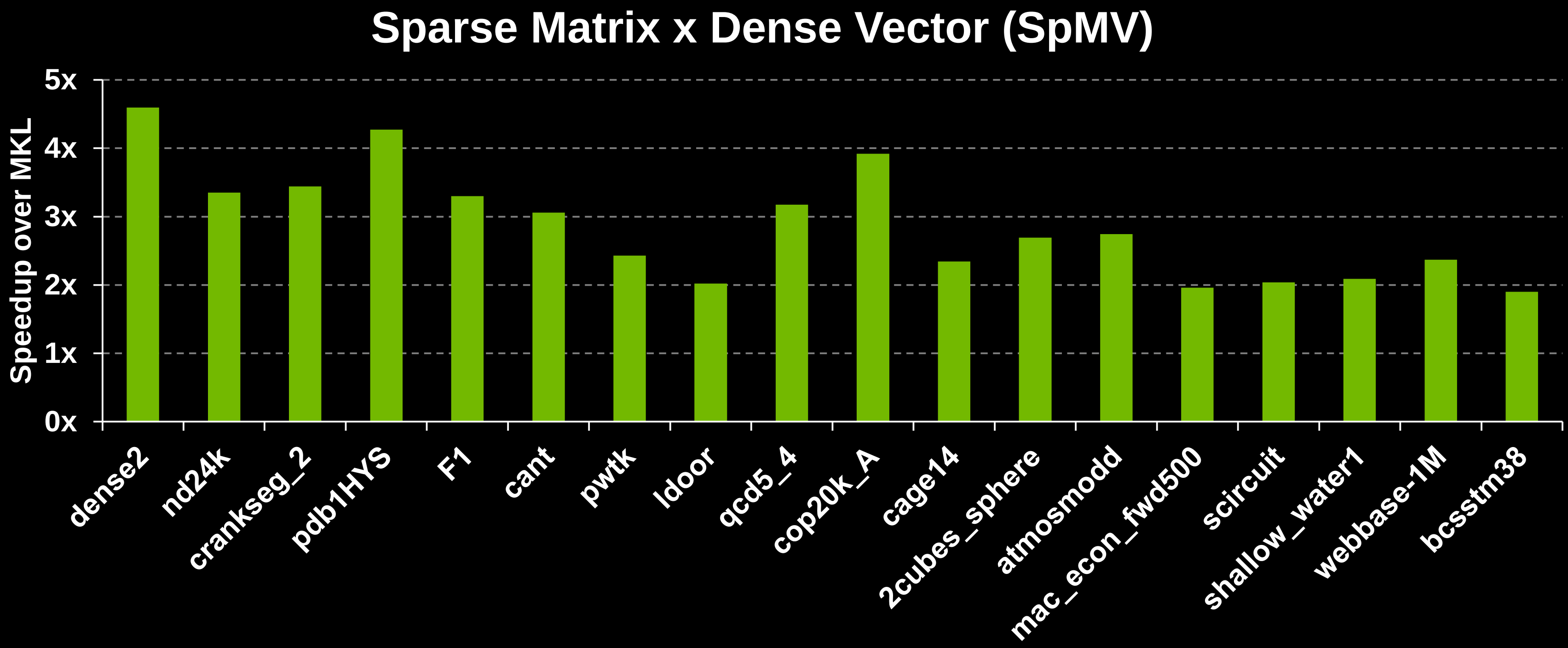
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \alpha \begin{bmatrix} 1.0 & & & \\ 2.0 & 3.0 & & \\ & & 4.0 & \\ 5.0 & & 6.0 & 7.0 \end{bmatrix} \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{bmatrix} + \beta \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

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

New in CUDA 7.0

- Graph Coloring

cuSPARSE: 4x Faster than MKL



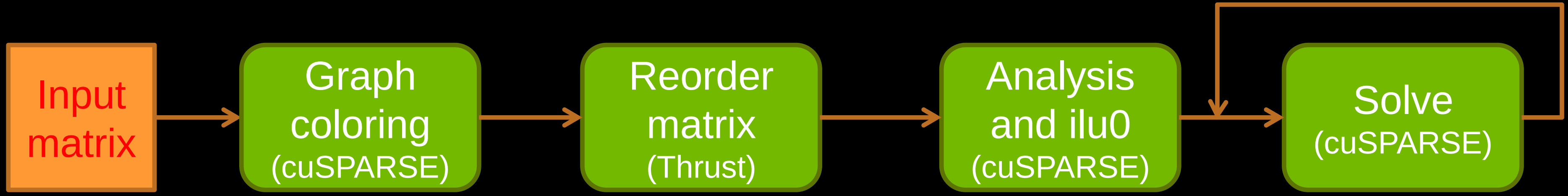
- Average of S/C/D/Z routines
- cuSPARSE 7.0 on K40m, Base clocks, ECC ON, input and output data on device
- MKL 11.0.4 on Intel Xeon Haswell single-socket 16-core E5-2698 v3 @ 2.3GHz, 3.6GHz Turbo
- Matrices obtained from: http://www.cise.ufl.edu/research/sparse/matrices/

Performance may vary based on OS and software versions, and motherboard configuration

New in CUDA 7.0

Graph Coloring & Preconditioned CG Solver

Execution time of ilu0 and Solve phases are limited by available parallelism in input Matrix

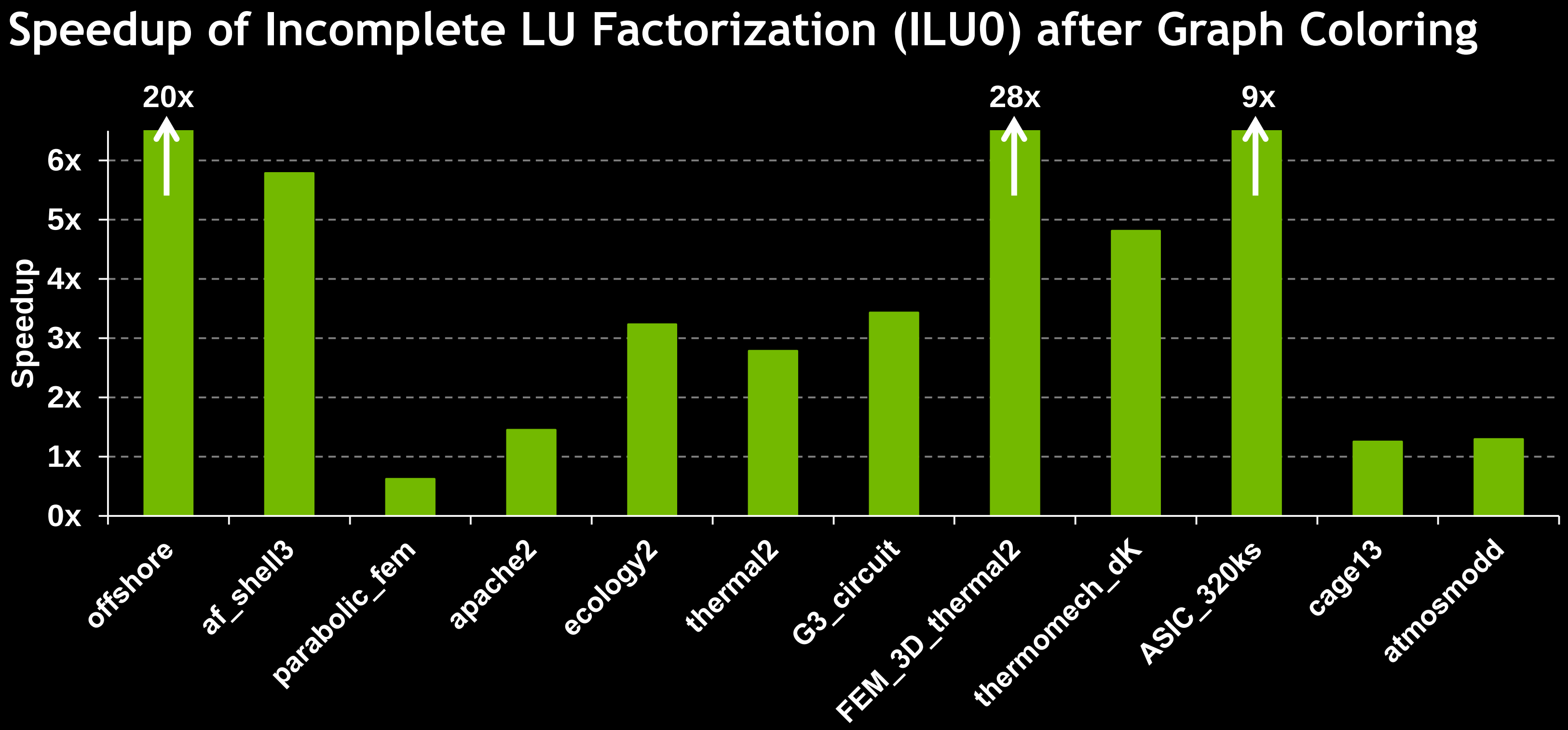


Coloring and reordering the matrix exposes more parallelism

ilu0 and Solve phases run faster due to extra parallelism

New in CUDA 7.0

cuSPARSE Graph Coloring Speeds Up Factorization



Full results at: research.nvidia.com/publication/parallel-graph-coloring-applications-incomplete-lu-factorization-gpu

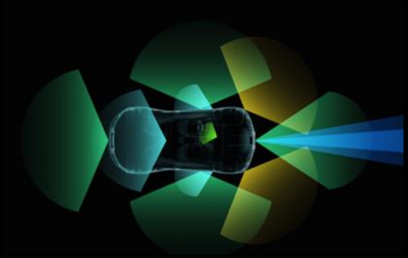
Performance may vary based on OS and software versions, and motherboard configuration

- cuSPARSE 7.0 on K40c
- Matrices obtained from: http://www.cise.ufl.edu/research/sparse/matrices/

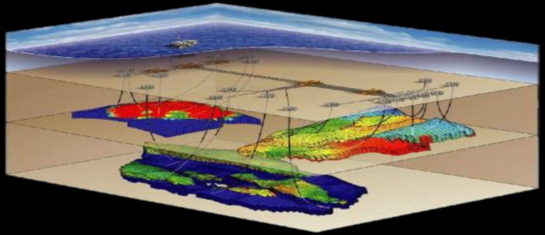
New in CUDA 7.0

cuSOLVER

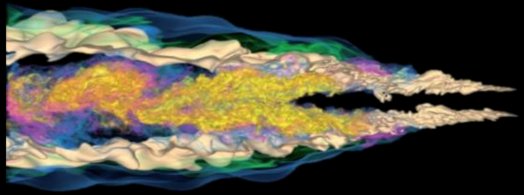
- cusolverDN
  - Dense Cholesky, LU, SVD, (batched) QR
  - Optimization, Computer vision, CFD



- cusolverSP
  - Sparse direct solvers & Eigensolvers
  - Newton's method, Oil & Gas Well Models

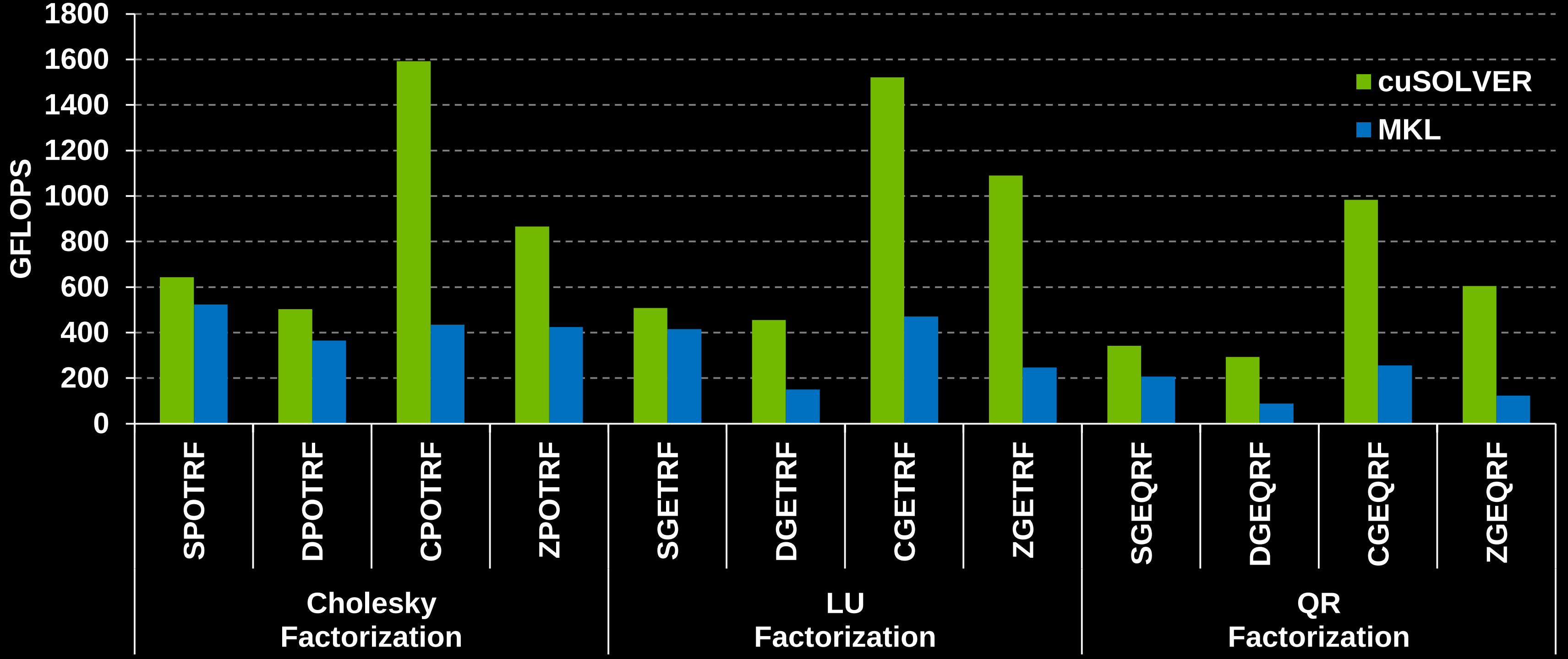


- cusolverRF
  - Sparse refactorization solver
  - Chemistry, ODEs, Combustion, Circuit simulation



cuSOLVER Dense vs. MKL

New in CUDA 7.0

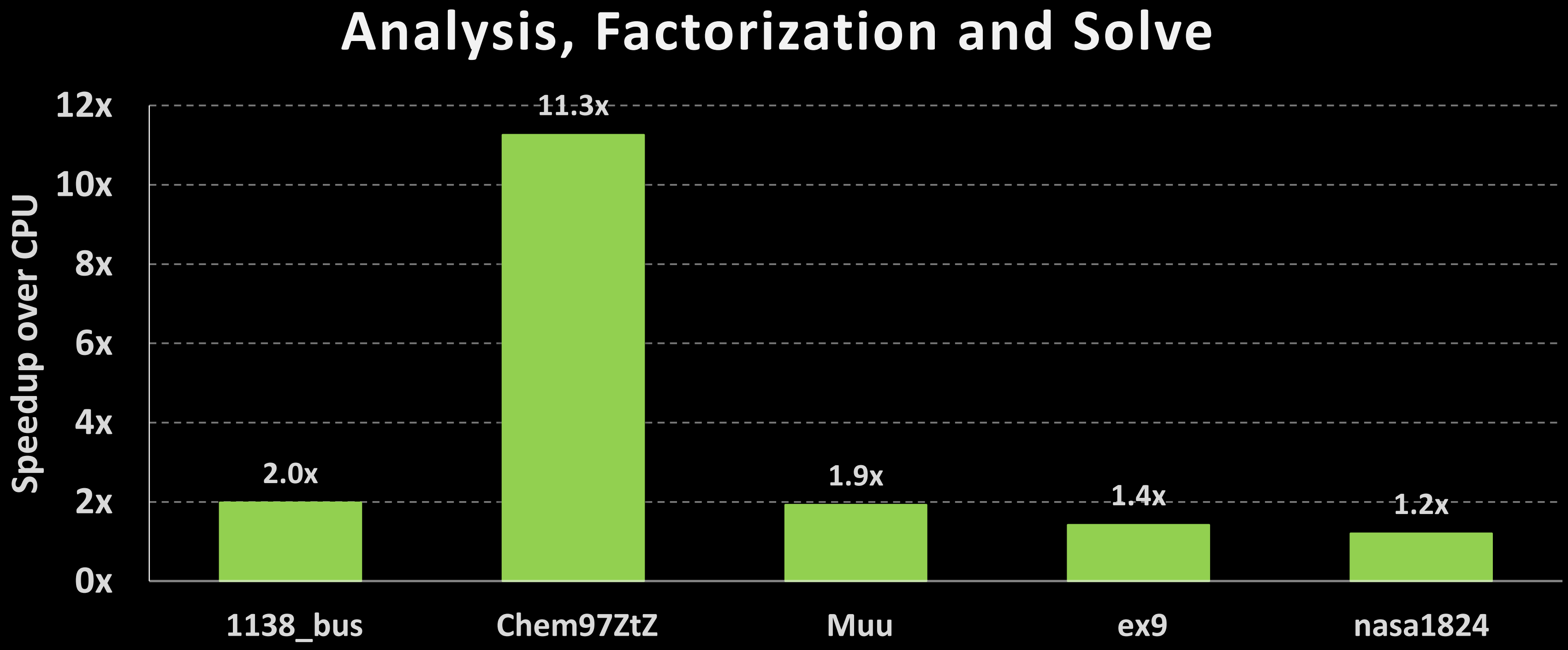


Performance may vary based on OS and software versions, and motherboard configuration

- cuSOLVER 7.0 on K40c, ECC ON, M=N=4096
- MKL 11.0.4 on Intel Xeon Haswell 14-core E5-2697 v3 @ 3.6GHz

New in CUDA 7.0

cuSOLVER Sparse QR



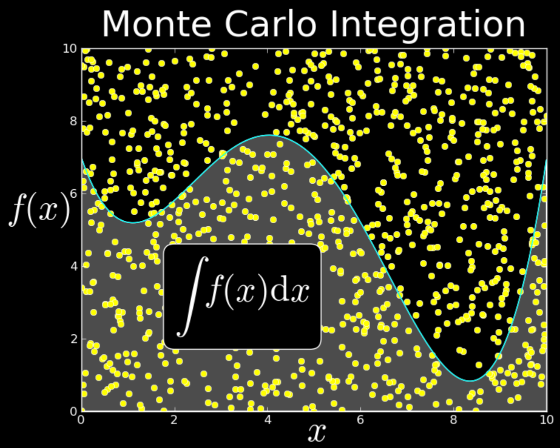
Performance may vary based on OS and software versions, and motherboard configuration

- cuSOLVER 7.0 on K40c, ECC ON
- SuiteSparse v4.4 on Intel Xeon Haswell 14-core E5-2697 v3 @ 3.6GHz
- Matrices obtained from: http://www.cise.ufl.edu/research/sparse/matrices/

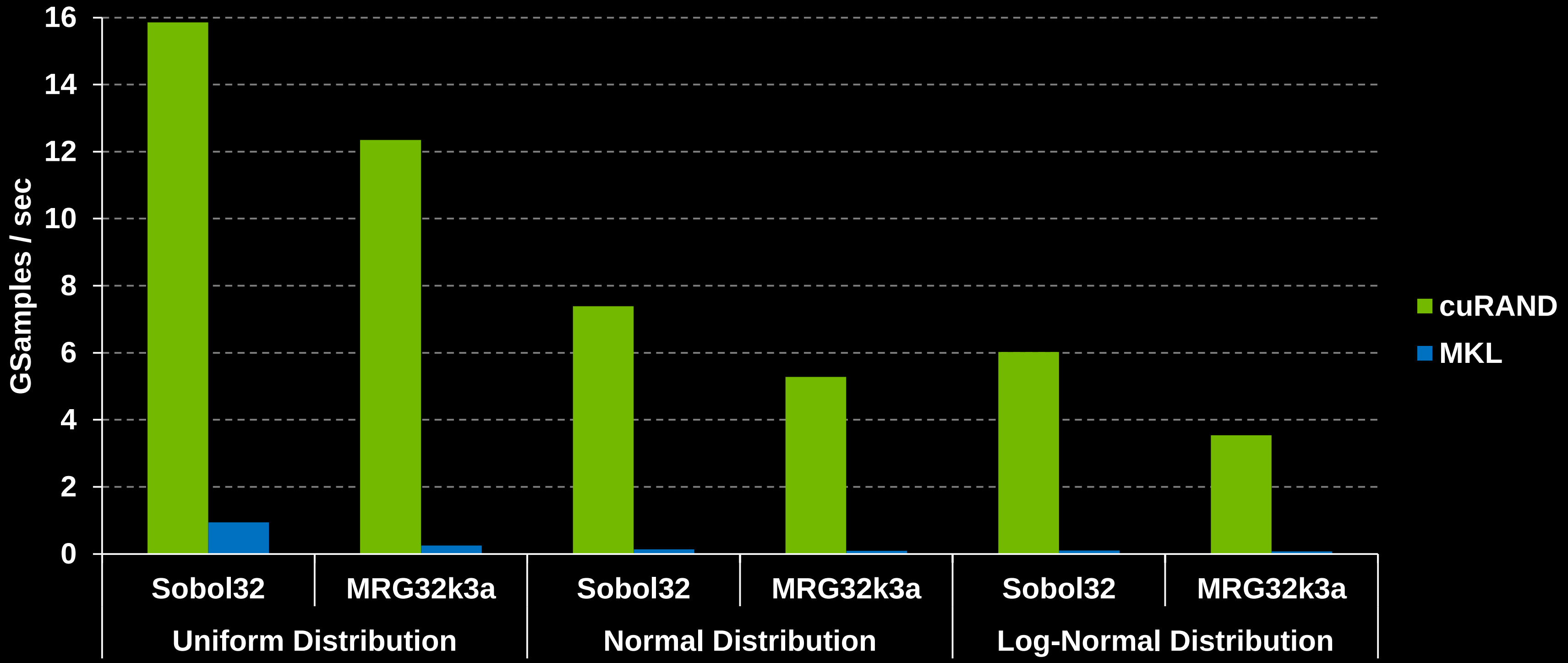
cuRAND: Random Number Generation

- 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



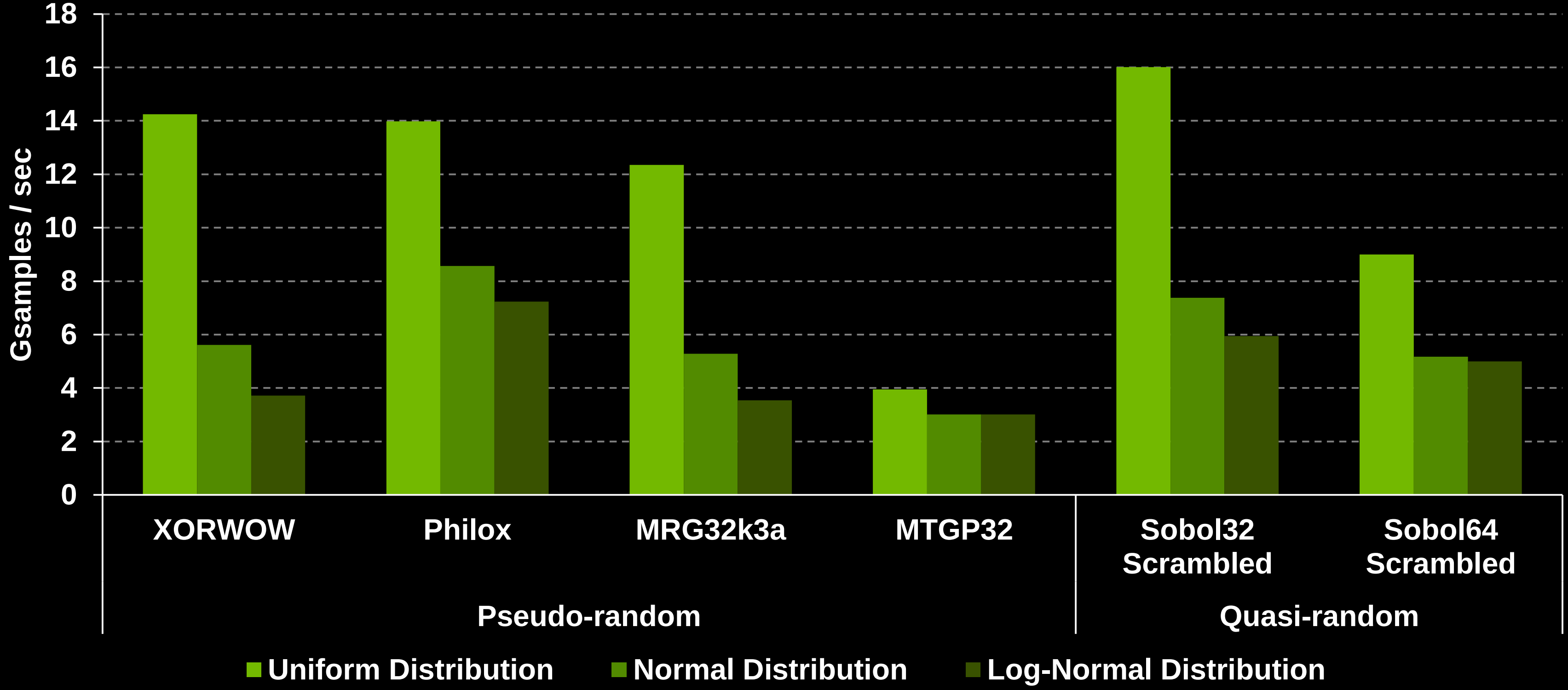
cuRAND: > 50x Faster vs. Intel MKL



Performance may vary based on OS and software versions, and motherboard configuration

- cuRAND 7.0 on K40m, Base clocks, ECC ON, double-precision input and output data on device
- MKL 11.0.1 on Intel Xeon Haswell single-socket 16-core E5-2698 v3 @ 2.3GHz, 3.6GHz Turbo

cuRAND: High Performance RNGs



Performance may vary based on OS and software versions, and motherboard configuration

- cuRAND 7.0 on K40m, Base clocks, ECC ON, double-precision input and output data on device

CUDA C++ Parallel Template Library

- Parallel Template library for CUDA C++
  - Host and Device Containers that mimic the C++ STL
  - Optimized Parallel Algorithms for sort, reduce, scan, etc.
  - TBB and OpenMP CPU Backends
  - Performance portable

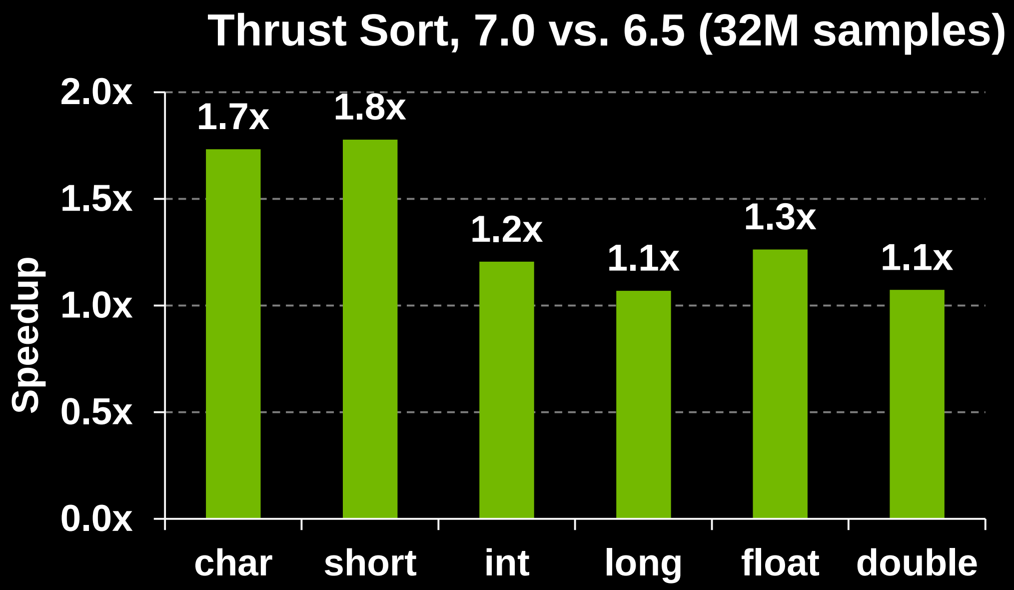
- Also available on GitHub: thrust.github.com

- Allows applications and prototypes to be built quickly

New in CUDA 7.0

Thrust Performance Improvements

sort: 1.1x - 1.8x faster
(3x for user-defined types)
merge: 2x faster
scan: 1.15x faster
reduce_by_key: 1.25x faster



- CUDA 7.0 and 6.5 on K40m, ECC ON, input and output data on device
- Performance may vary based on OS and software versions, and motherboard configuration

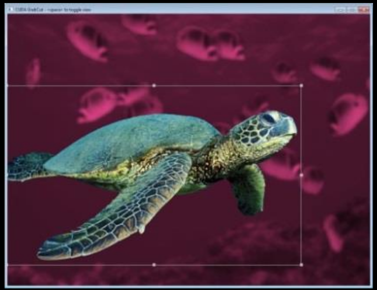
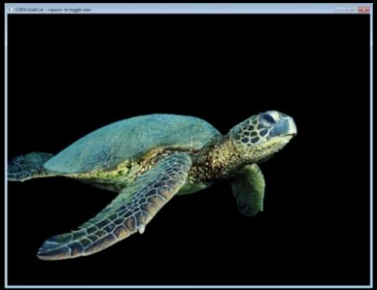
- CUDA streams argument (concurrency between threads)

```
thrust::count_if(thrust::cuda::par.on(stream1), text, text+n, myFunc());
```

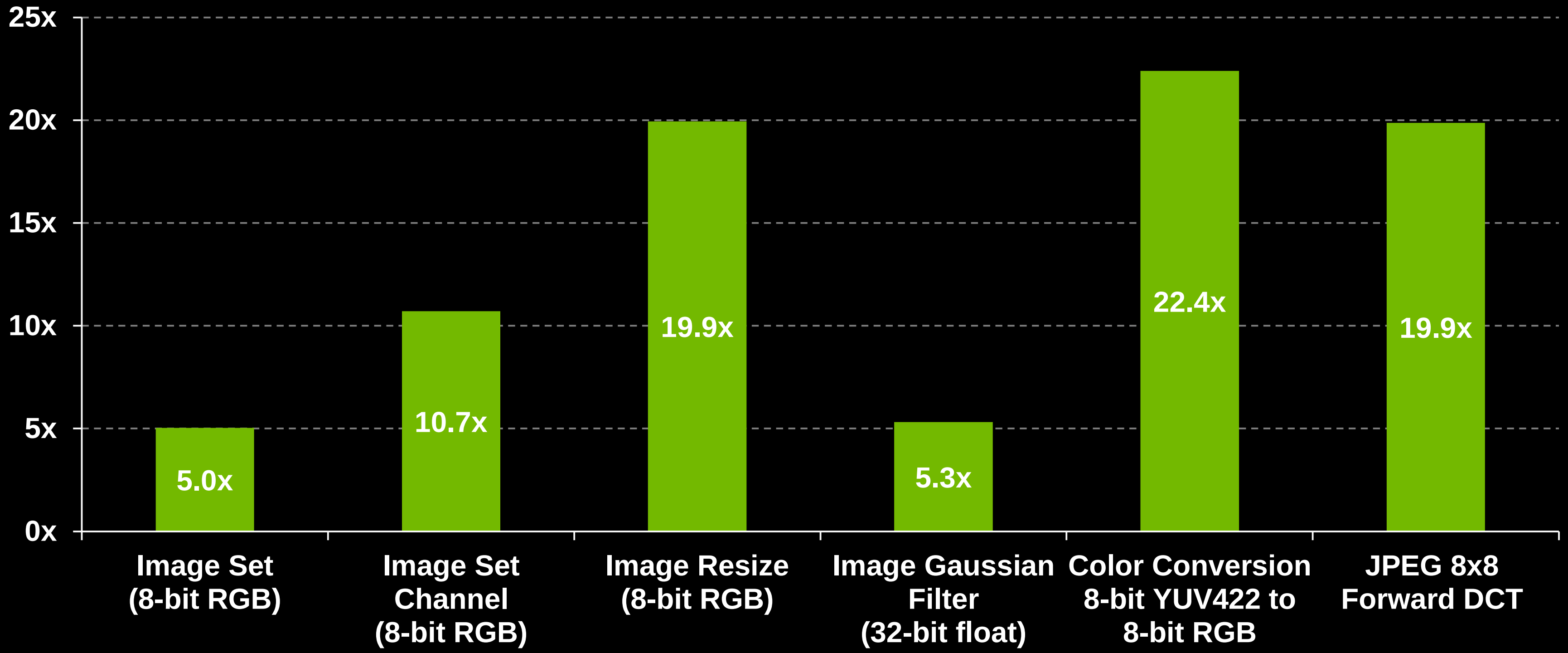
NPP: NVIDIA Performance Primitives

- 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

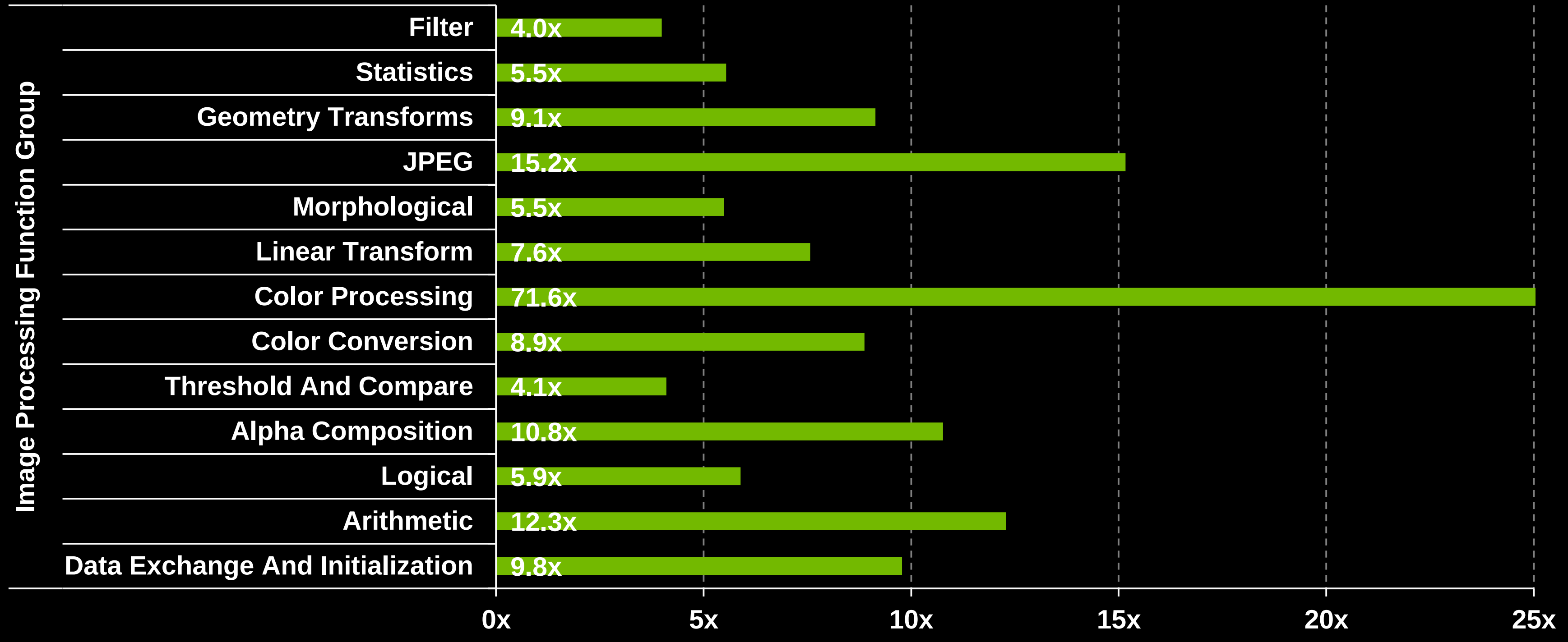
NPP Speedup vs. Intel IPP



Performance may vary based on OS and software versions, and motherboard configuration

- NPP 7.0 on K40m, ECC ON, Base clocks, input and output data on device
- IPP 7.0 on Intel Xeon Haswell single-socket 16-core E5-2698 v3 @ 2.3GHz, 3.6GHz Turbo

NPP Speedup vs. Intel IPP



- NPP 7.0 on K40m, ECC ON, Base clocks, input and output data on device
- Each bar represents the average speedup over all routines in the function group
- IPP 7.0 on Intel Xeon Haswell single-socket 16-core E5-2698 v3 @ 2.3GHz, 3.6GHz Turbo

Performance may vary based on OS and software versions, and motherboard configuration

math.h: C99 floating-point library + extras

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

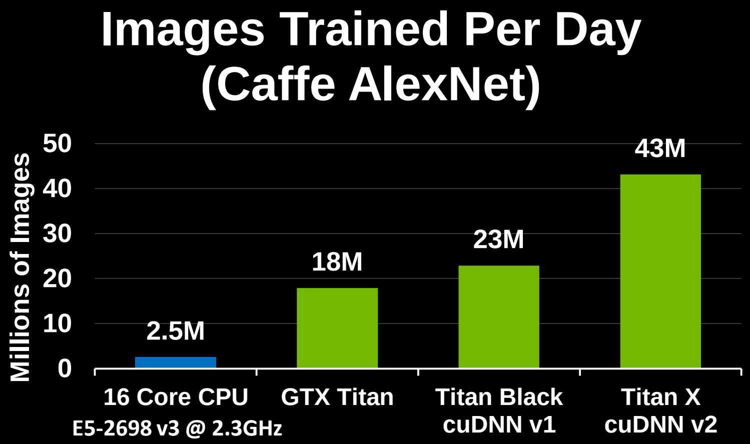
- Basic: +, *, /, 1/, sqrt, FMA (all IEEE-754 accurate for float, double, all rounding modes)
- Exponentials: exp, exp2, log, log2, log10, ...
- Trigonometry: sin, cos, tan, asin, acos, atan2, sinh, cosh, asinh, acosh, ...
- Special functions: lgamma, tgamma, erf, erfc
- Utility: fmod, remquo, modf, 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]

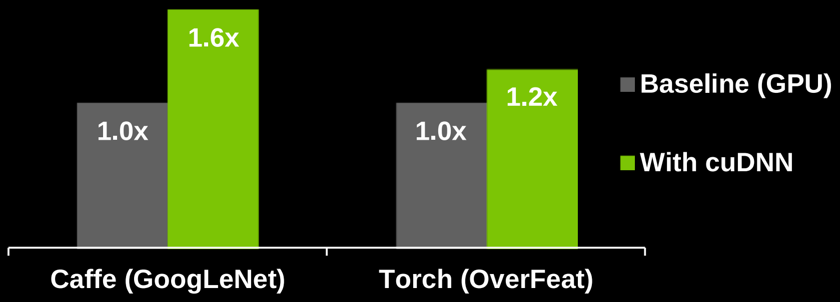
New in CUDA 7.0

- Significantly optimized double precision reciprocal
  - `rcp()`
- 3D/4D Euclidean Norms:
  - `[r]norm3d[f], norm4d[f]`

NVIDIA releases cuDNN Version 2

- Accelerates key routines to improve performance of neural net training
  - Up to 1.8x faster on AlexNet than a baseline GPU implementation
- New support for 3D convolutions
- Integrated into all major Deep Learning frameworks: Caffe, Theano, Torch





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