
CUDA Toolkit 5.0
Performance Report
January 2013
NVIDIA

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

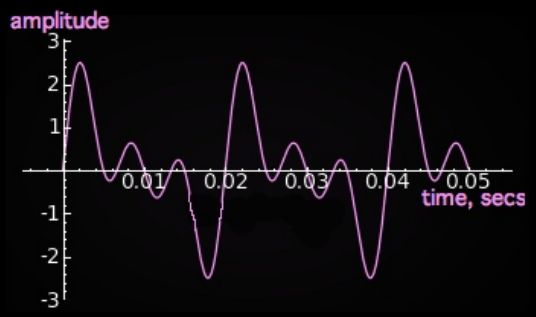
Included in the CUDA Toolkit: www.nvidia.com/getcuda (free download)

For more information on CUDA libraries:

http://developer.download.nvidia.com/GTC/PDF/GTC2012/PresentationPDF/S0629-Monday-CUDA-Accelerated.pdf

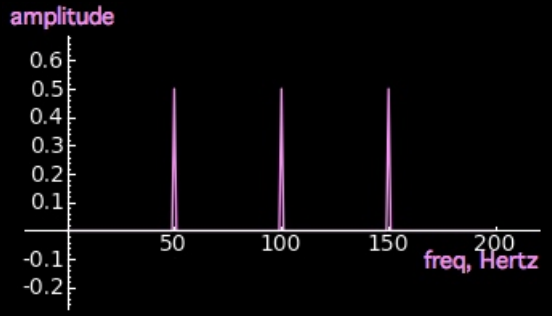
cuFFT: Multi-dimensional FFTs

- Real and Complex data types
- Single- and double-precision
- 1D, 2D and 3D batched transforms
- Flexible input and output data layouts
  - Similar to the FFTW “Advanced Interface”



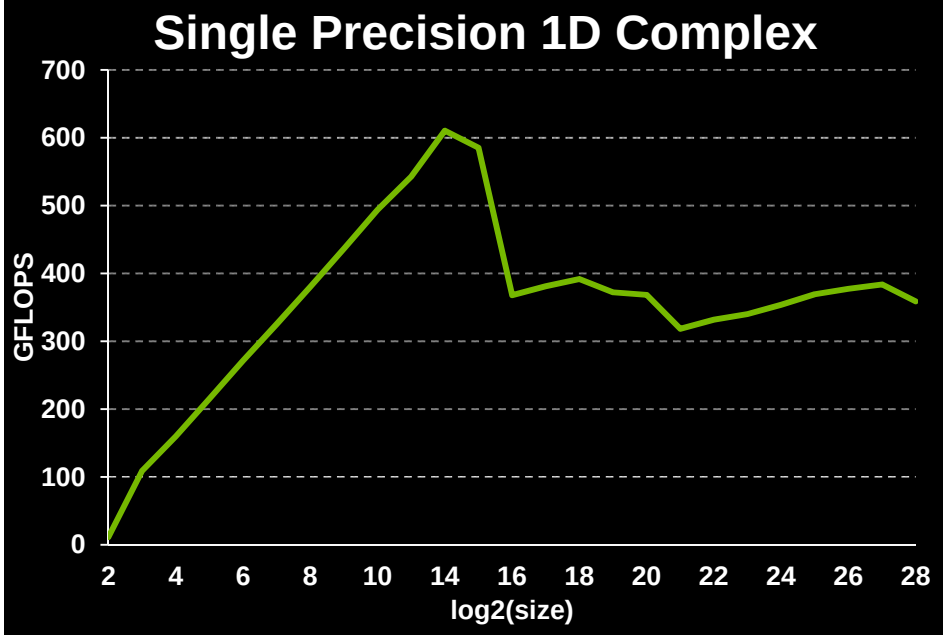
$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(x\frac{n}{N})}$$

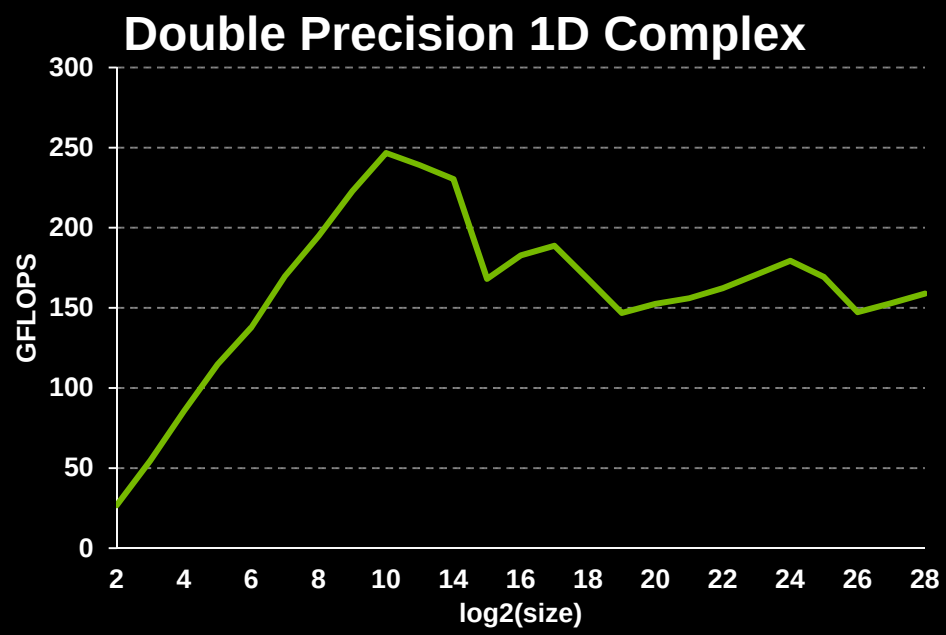
$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(x) e^{j2\pi(x\frac{n}{N})}$$



cuFFT: up to 600 GFLOPS

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





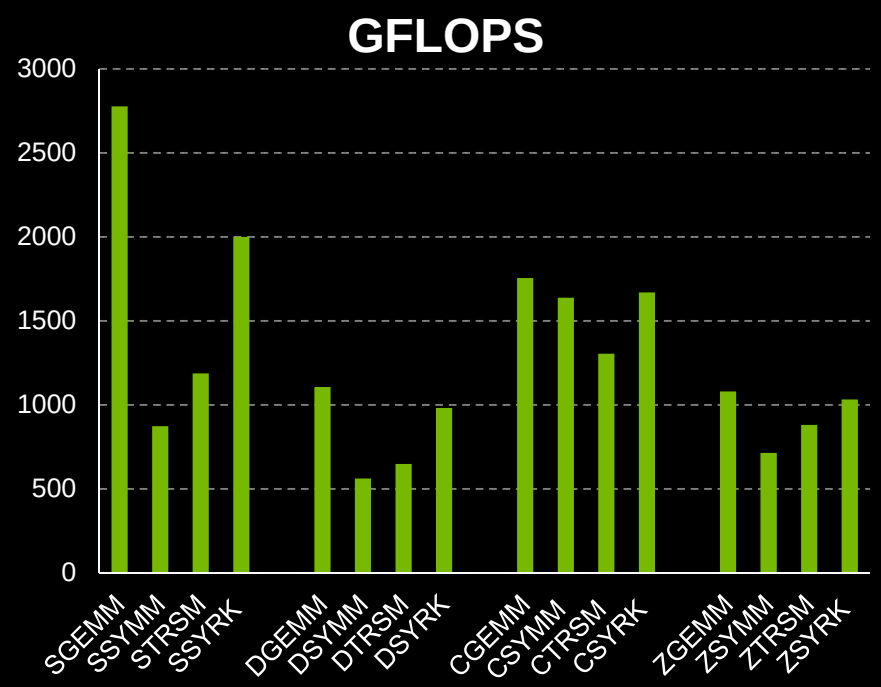
Performance may vary based on OS version and motherboard configuration

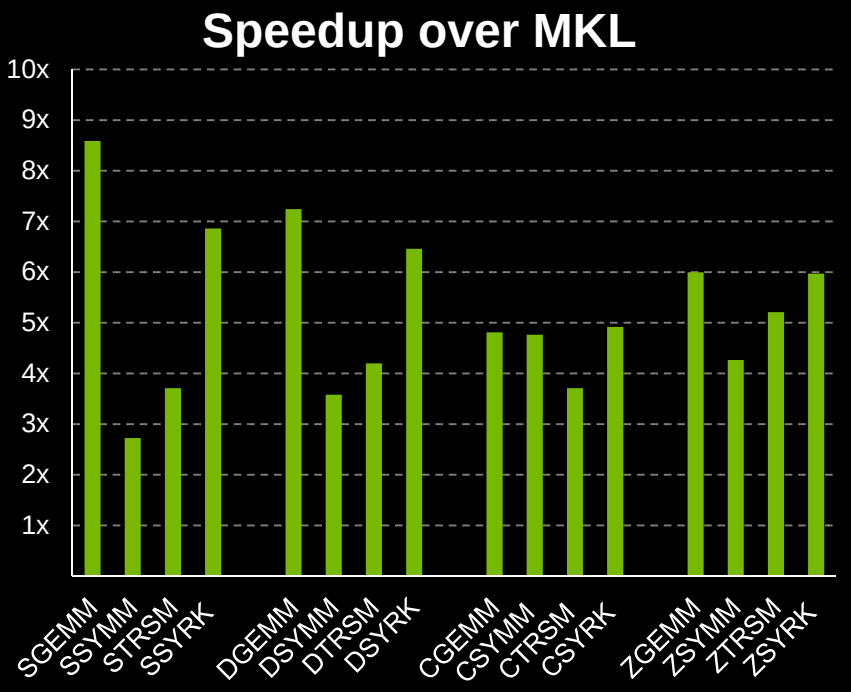
• cuFFT 5.0 on K20X, input and output data on device

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 5.0:
  - cuBLAS interface callable from device kernels on K20, K20X
  - Full list of new features and optimizations:
    http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#cublas
    http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#cublas-performance-improvements

cuBLAS: >1 TFLOPS double-precision

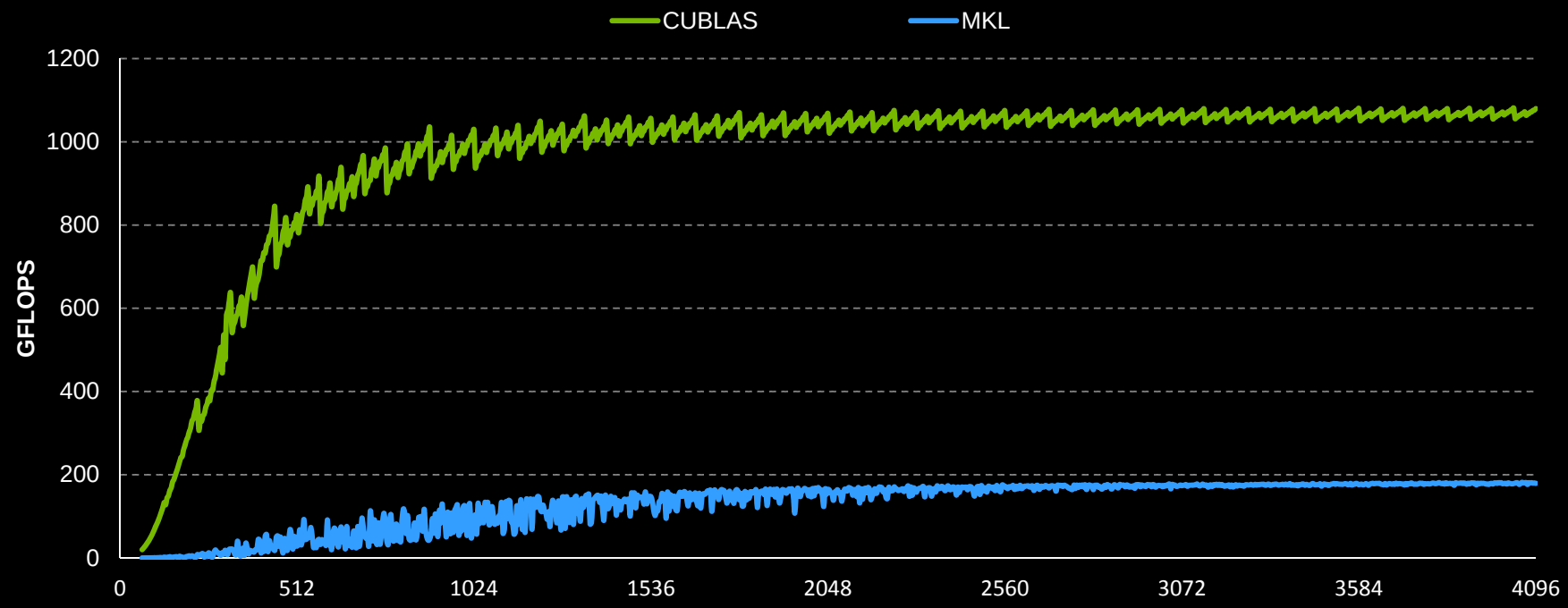




- cuBLAS 5.0 on K20X, input and output data on device
- MKL 10.3.6 on Intel SandyBridge E5-2687W @ 3.10GHz

Performance may vary based on OS version and motherboard configuration

ZGEMM Performance vs. Matrix Size



- cuBLAS 5.0 on K20X, input and output data on device
- MKL 10.3.6 on Intel SandyBridge E5-2687W @ 3.10GHz

Performance may vary based on OS version and motherboard configuration

cuSPARSE: Sparse linear algebra routines

- Format conversion: dense, COO, CSR, CSC, HYB, BlockCSR
- Optimized sparse linear algebra for CSR and HYB formats
- New in CUDA 5.0:
  - Incomplete-LU & -Cholesky preconditioners (ilu0 and ic0)
  - BlockCSR storage format (bsr)
  - Complete list of new feature and optimizations:

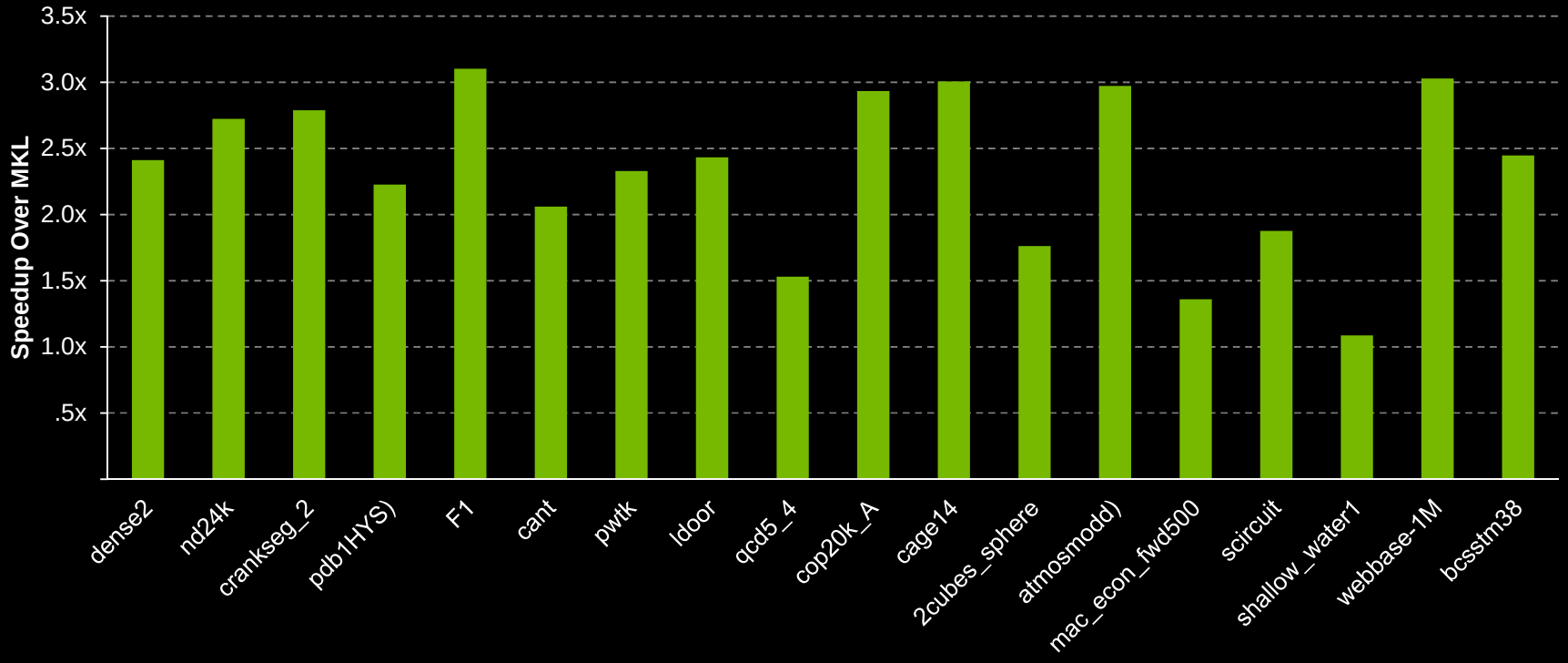
    http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#cusparse

    http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#cusparse-performance-improvements

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

cuSPARSE performance

Sparse Matrix x Dense Vector

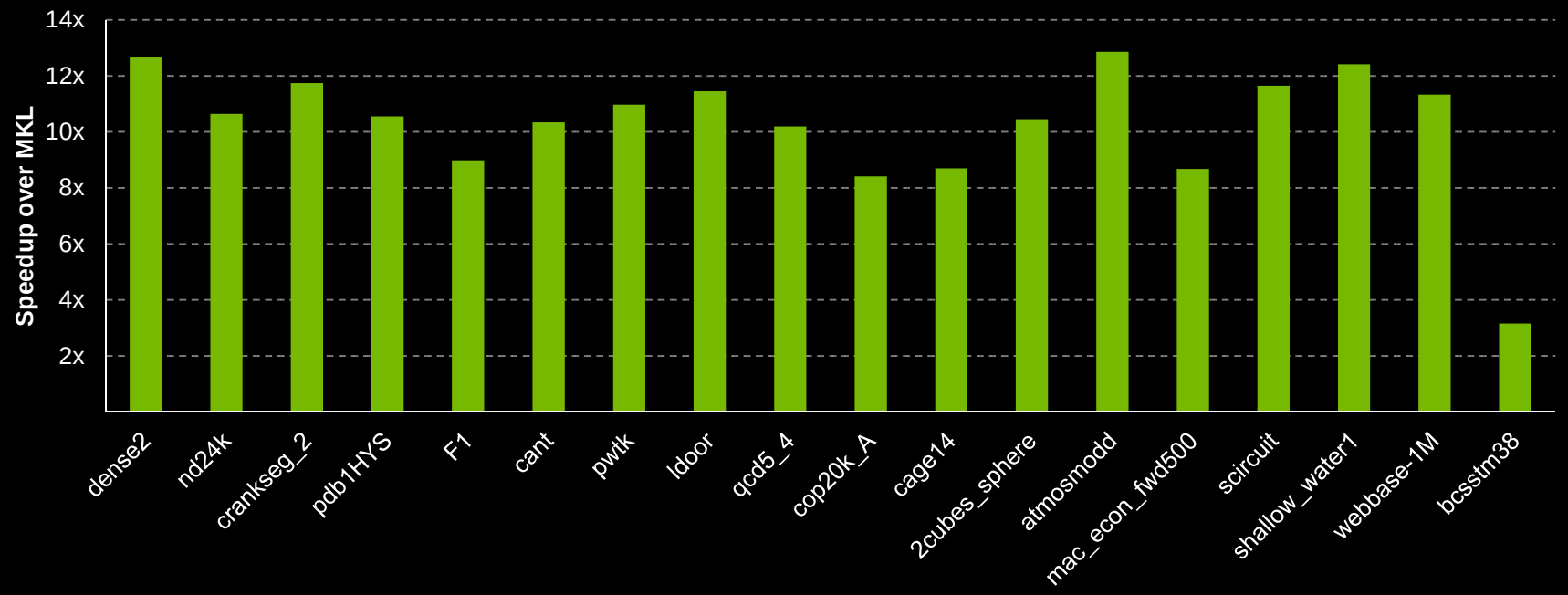


- Average of s/d/c/z routines
- cuSPARSE 5.0 on K20X, input and output data on device
- MKL 10.3.6 on Intel SandyBridge E5-2687W @ 3.10GHz

Performance may vary based on OS version and motherboard configuration

cuSPARSE: up to 12x Faster than MKL

Sparse Matrix x 6 Dense Vectors
(useful for block iterative solvers)

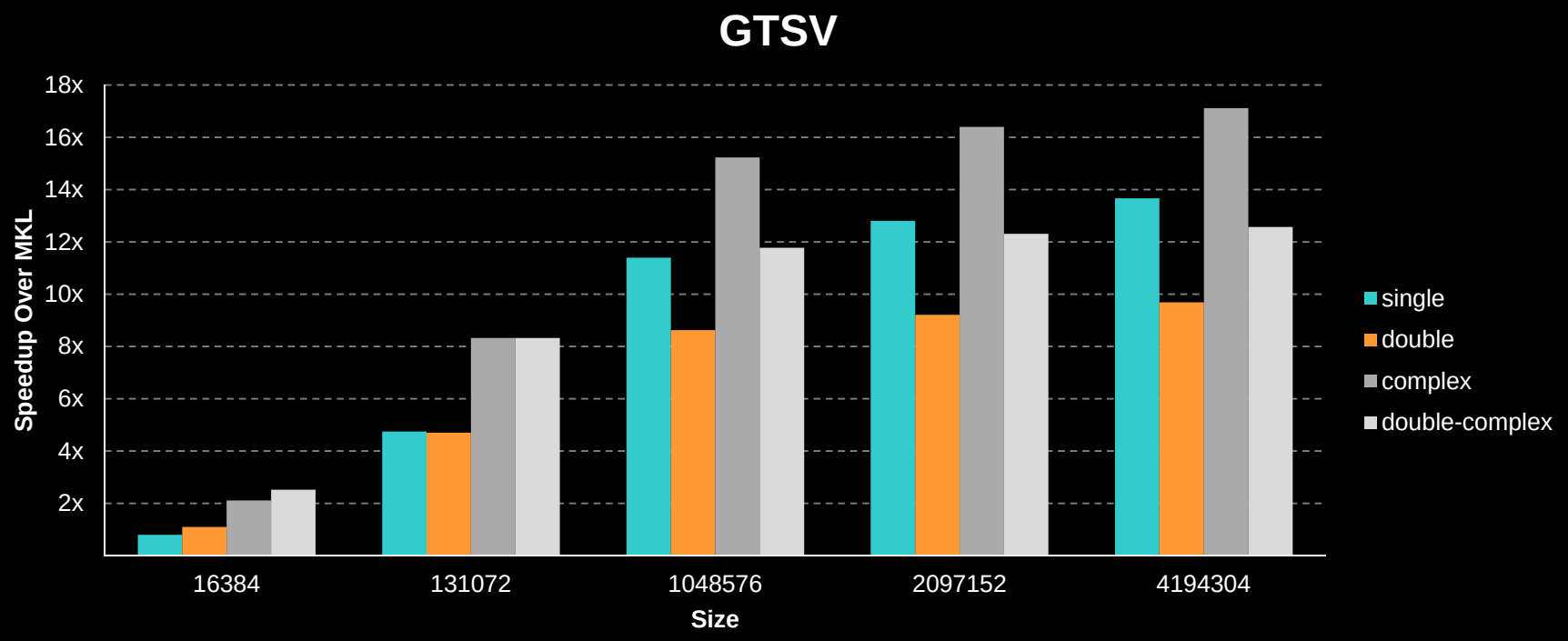


- Average of s/d/c/z routines
- cuSPARSE 5.0 on K20X, input and output data on device
- MKL 10.3.6 on Intel SandyBridge E5-2687W @ 3.10GHz

Performance may vary based on OS version and motherboard configuration

Tri-diagonal Solver Performance vs. MKL

Speedup for Tri-Diagonal solver (gtsv)

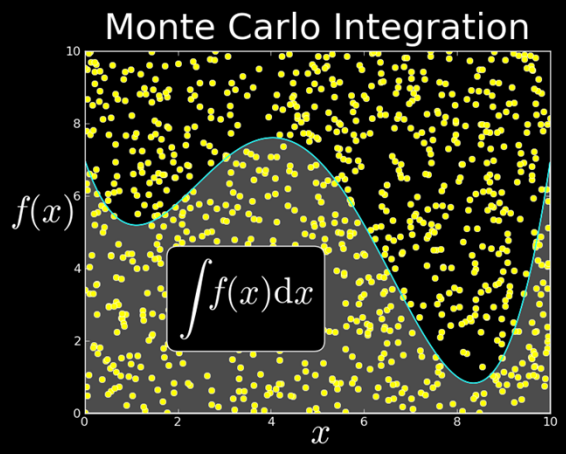


- cuSPARSE 5.0 on K20X, input and output data on device
- MKL 10.3.6 on Intel SandyBridge E5-2687W @ 3.10GHz

Performance may vary based on OS version and motherboard configuration

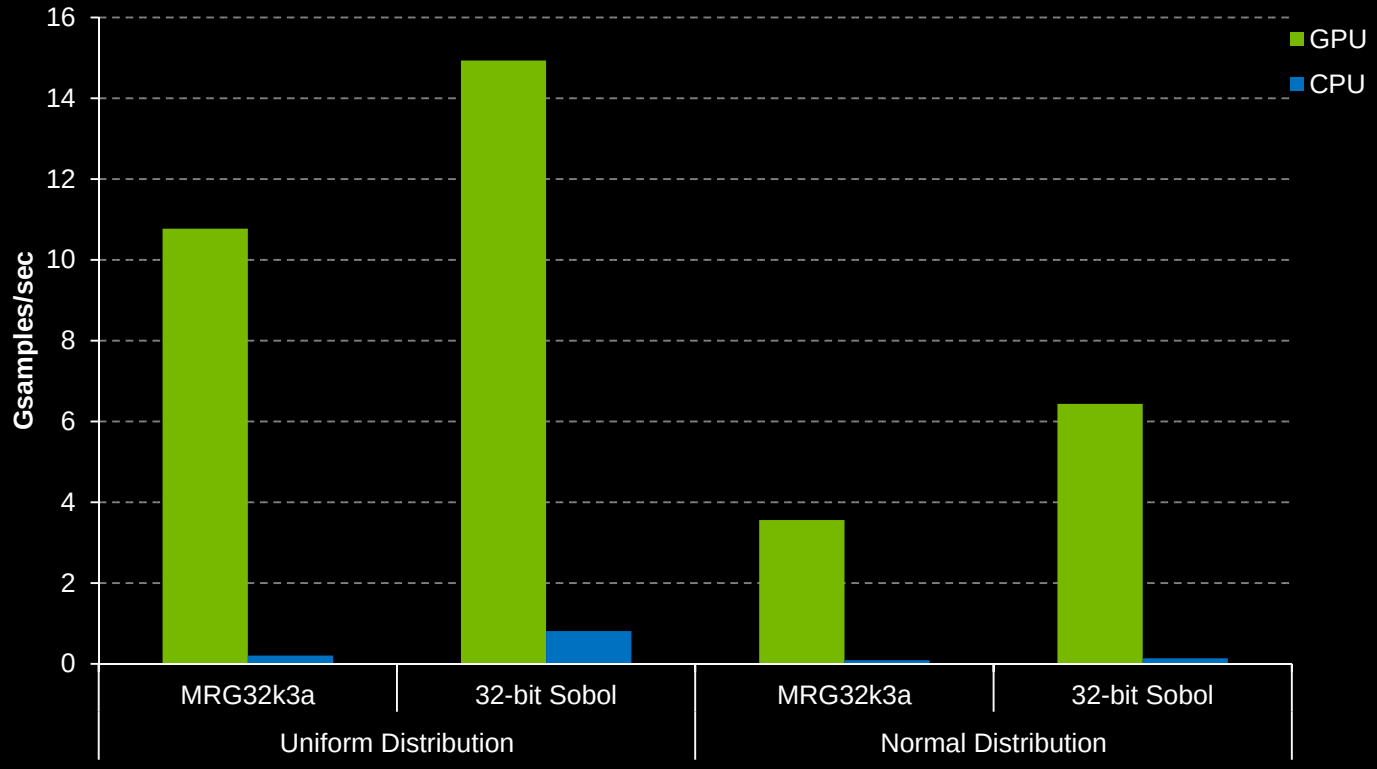
cuRAND: Random Number Generation

- Generating high quality random numbers in parallel is hard
  - Don’t do it yourself, use a library!
- Pseudo- and Quasi-RNGs
- Supports several output distributions
- Statistical test results in documentation
- New in CUDA 5.0: Poisson distribution



cuRAND Performance

Double Precision RNGs



- cuRAND 5.0 on K20X, input and output data on device
- MKL 10.2.3 on Intel SandyBridge E5-2687W @ 3.10GHz

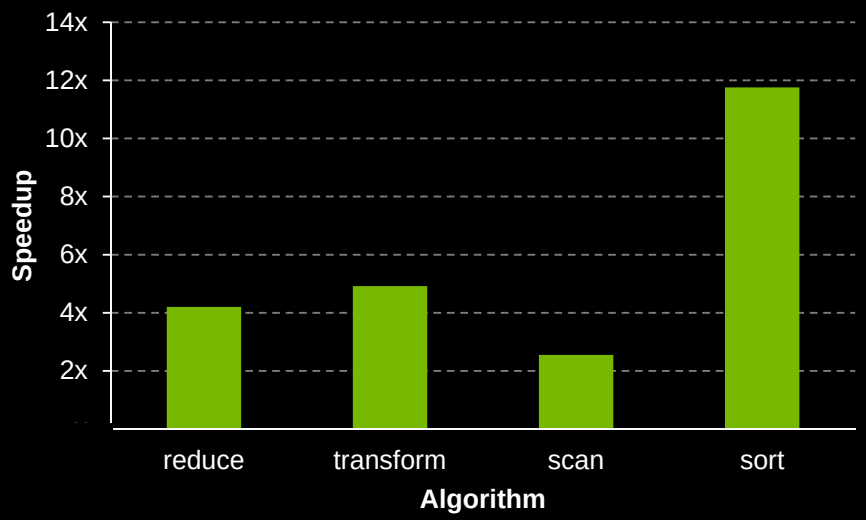
Performance may vary based on OS version and motherboard configuration

CUDA C++ Template Library

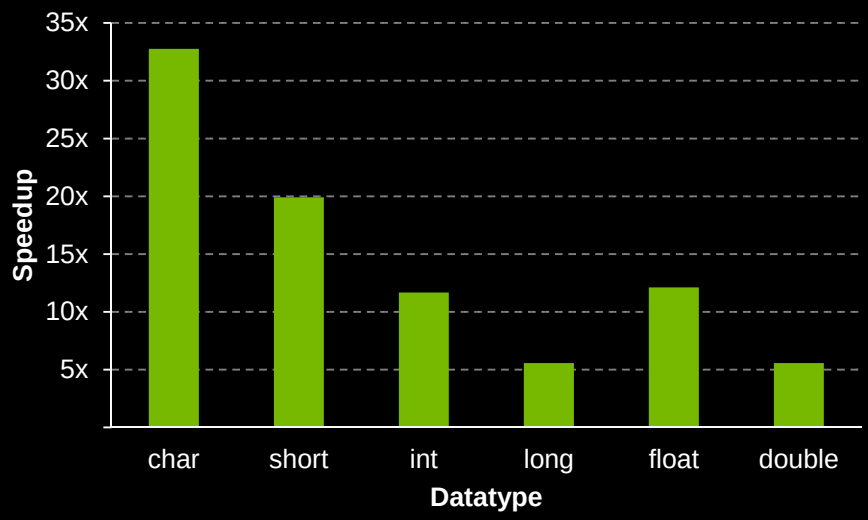
- Template library for CUDA
  - Host and Device Containers that mimic the C++ STL
  - Optimized algorithms for sort, reduce, scan, etc.
  - OpenMP backend for portability
- Also available on github: http://thrust.github.com/
- Allows applications and prototypes to be built *quickly*

Thrust Performance

Various Algorithms (32M int.) Speedup over TBB



Sort (32M samples) Speedup over TBB



- Thrust 5.0 on K20X, input and output data on device
- TBB 4.1 on Intel SandyBridge E5-2687W @3.10GHz

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, …
- Extras: rsqrt, rcbrt, exp10, sinpi, sincos, cospi, erfinv, erfcinv, …

- New in CUDA 5.0
  - sincospi[f]() and normcdf[inv][f]()
  - sin(), cos() and erfcinvf() more accurate and faster
  - Full list of new features and optimizations:

    http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#math

    http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html#math-performance-improvements