
CUDA Toolkit 4.0
Performance Report
June, 2011
NVIDIA

CUDA Math Libraries

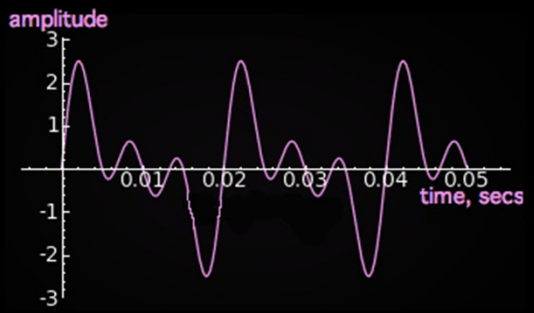
High performance math routines for your applications:

- 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 (free download)
  - www.nvidia.com/getcuda
- For more information on CUDA libraries:
  - http://www.nvidia.com/object/gtc2010-presentation-archive.html#session2216

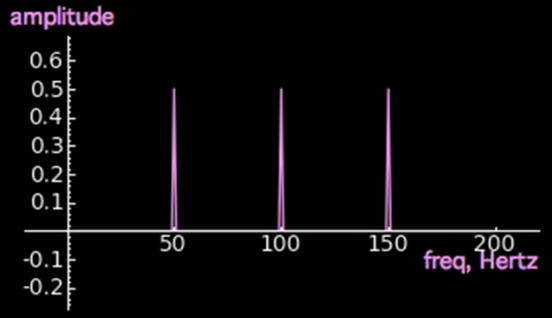
cuFFT: Multi-dimensional FFTs

- New in CUDA 4.0
  - Significant performance improvements in:
    - double precision radix 2, 3, 5 and 7
    - 2D/3D sizes that contain prime factors larger than 7
  - Flexible input and output data layouts*
    - Similar to the FFTW "Advanced Interface"
    - Eliminates extra data transposes and copies



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

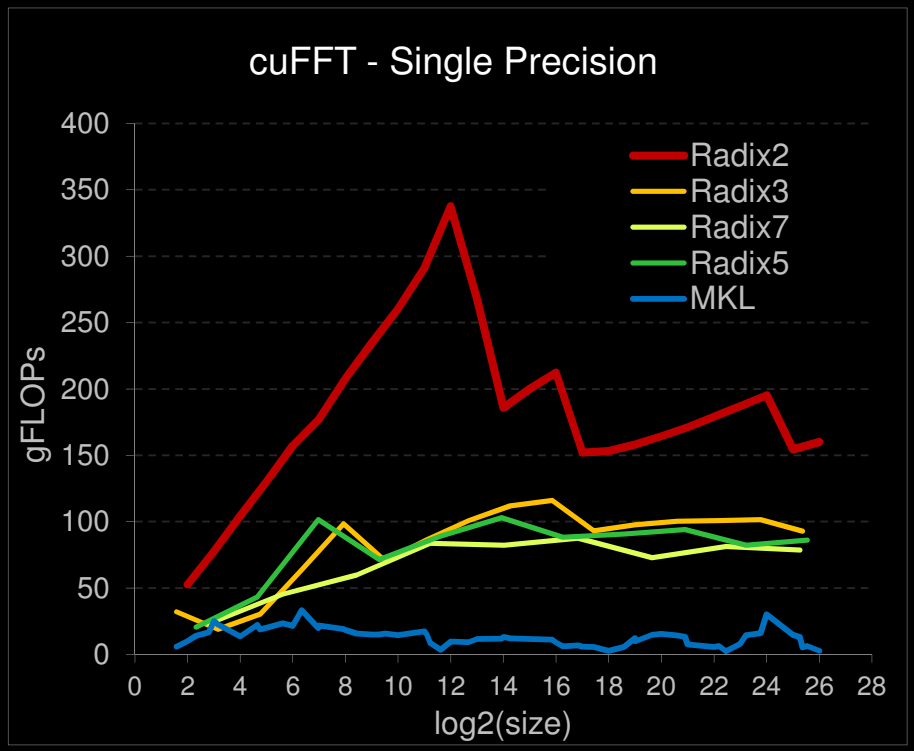
$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(x) e^{j2\pi(x\frac{n}{N})}$$

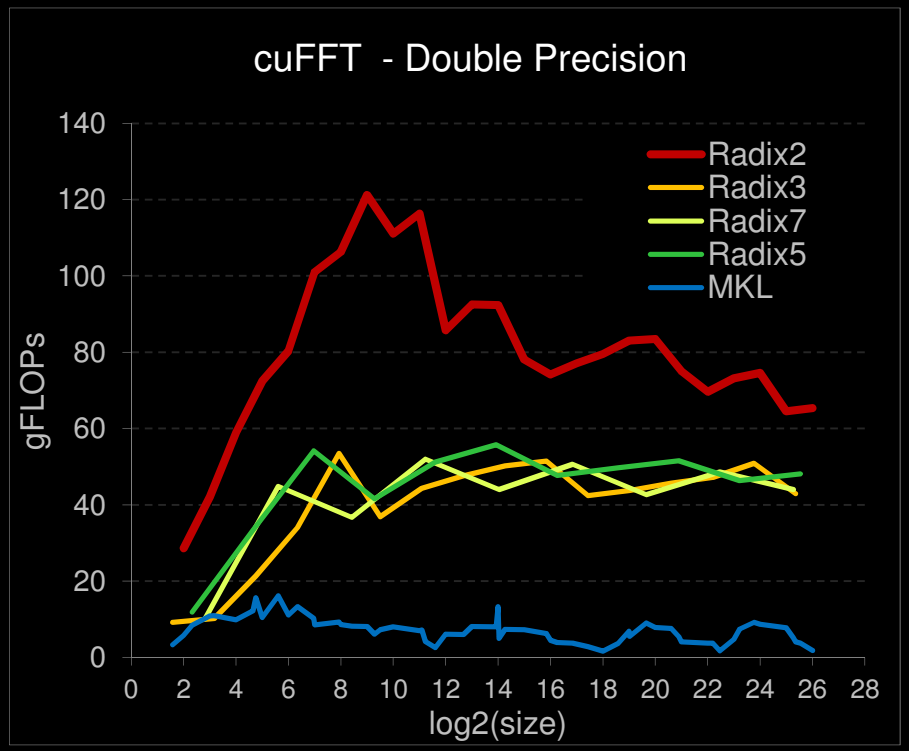


* Only supported for complex-to-complex transforms in this release

FFTs up to 10x Faster than MKL

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

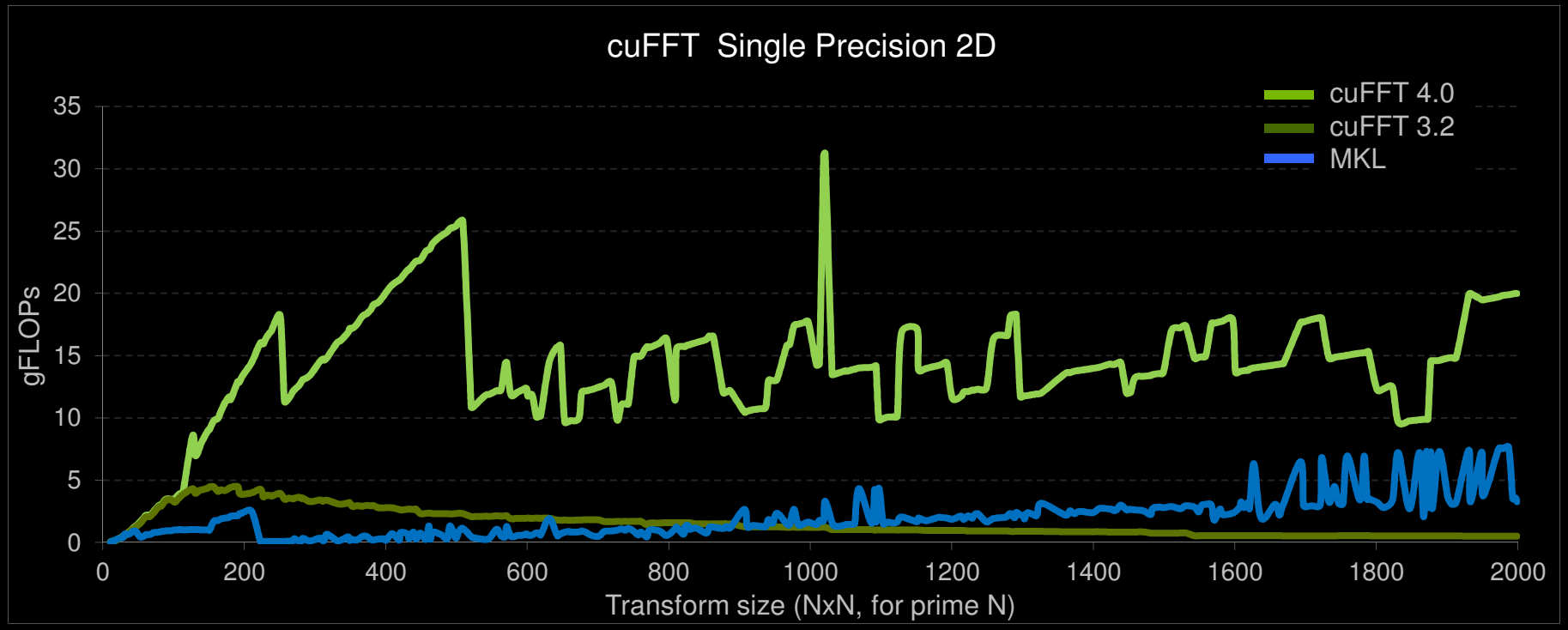




- MKL 10.1r1 on Intel Quad Core i7-940 1333, 2.93Ghz
- cuFFT 4.0 on Tesla C2070, ECC on
- Performance measured for ~16M total elements, split into batches of transforms of the size on the x-axis

2D/3D primes now use Bluestein Algorithm

Significant performance improvement for 2D and 3D transform sizes



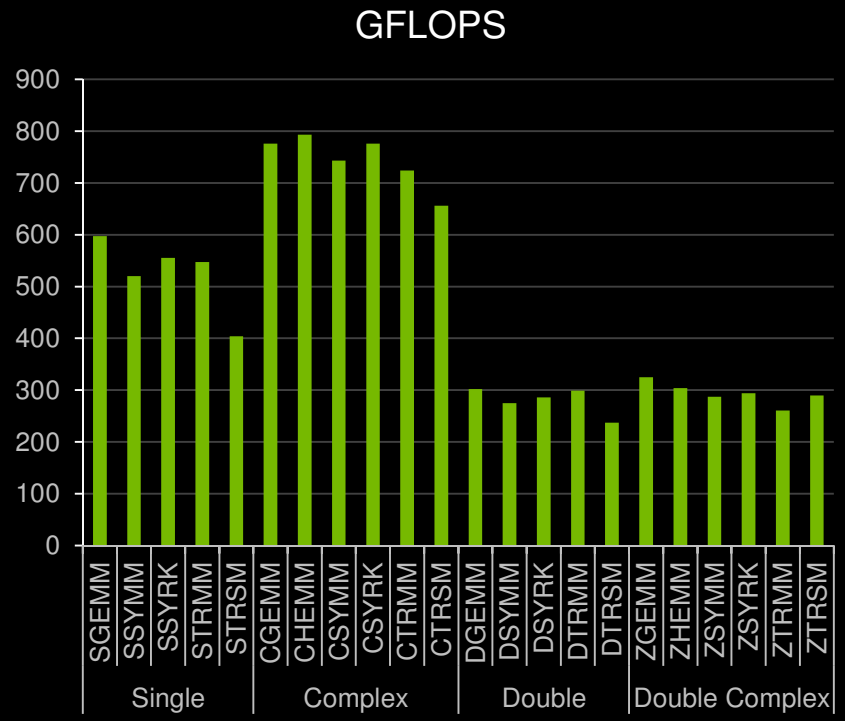
* MKL 10.1r1 on Intel Quad Core i7-940 1333, 2.93Ghz
* cuFFT4.0 on C2070, ECC on

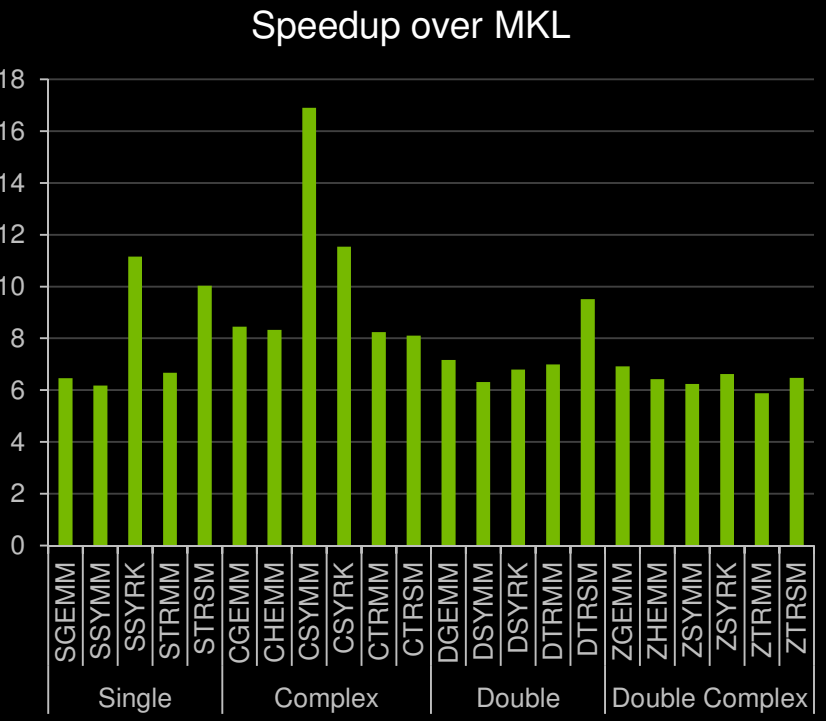
cuBLAS: Dense Linear Algebra on GPUs

- Complete BLAS implementation plus useful extensions
  - Supports all 152 standard routines for single, double, complex, and double complex
- New in CUDA 4.0
  - New API
    - Facilitates multi-GPU programming
    - Thread-safe
    - More routines provide parallelism using streams
    - Previous “legacy” API still supported out-of-the-box
  - Rewrote documentation from scratch
  - Performance improvements
    - Ex: ZGEMM performance improved 10% on Fermi (325 GFLOPS peak on C2050)

cuBLAS Level 3 Performance

Up to ~800GFLOPS and ~17x speedup over MKL





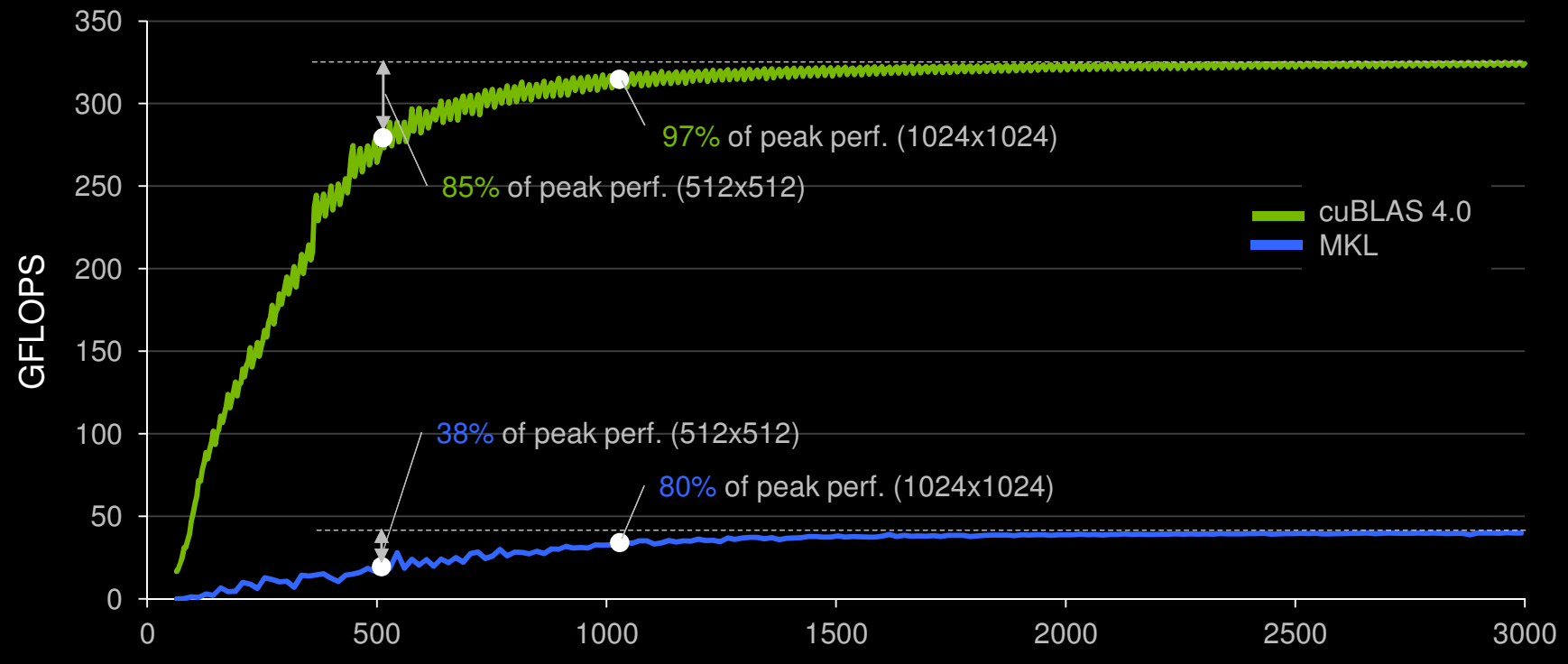
* 4Kx4K matrix size

* cuBLAS 4.0, Tesla C2050 (Fermi), ECC on

* MKL 10.2.3, 4-core Corei7 @ 2.66Ghz

ZGEMM Performance vs. Matrix Size

Up to 8x speedup over MKL



* cuBLAS 4.0, Tesla C2050 (Fermi), ECC on
* MKL 10.2.3, 4-core Corei7 @ 2.66Ghz

Performance may vary based on OS version and motherboard configuration

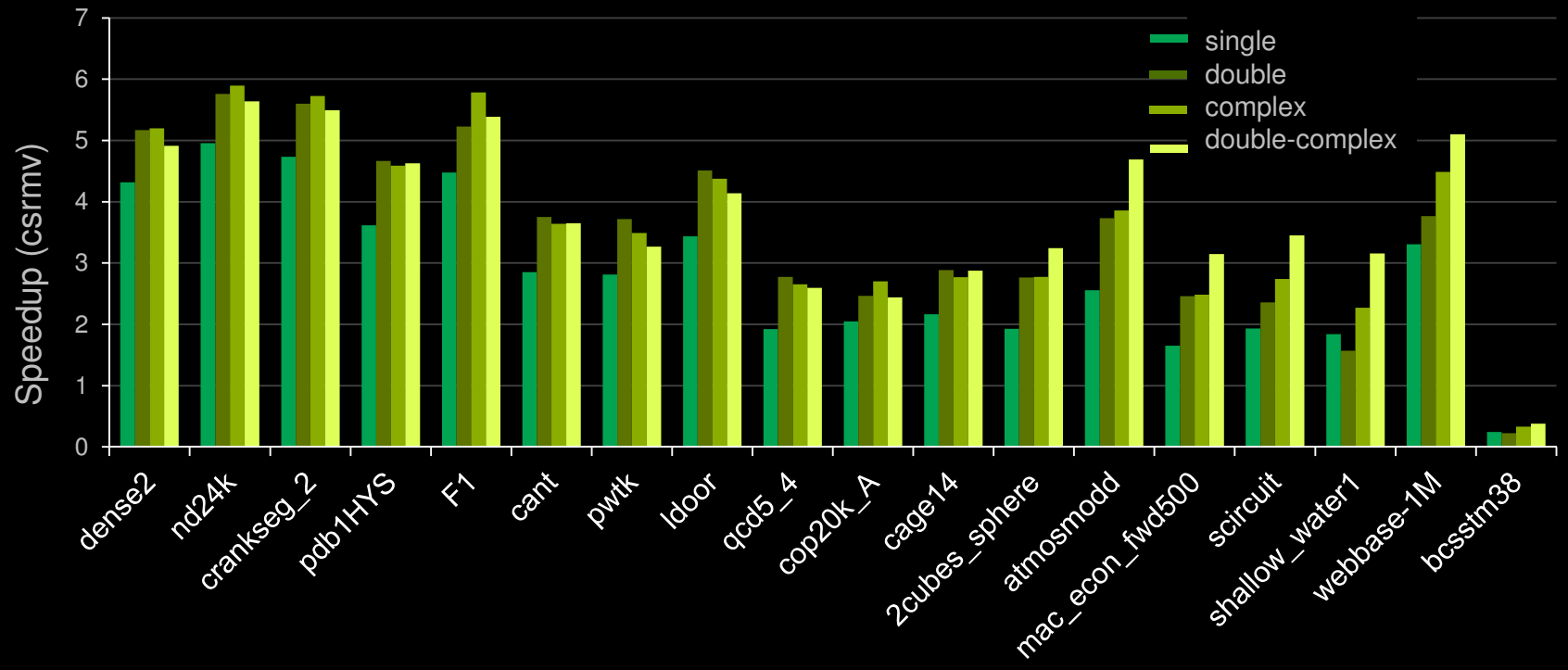
cuSPARSE: Sparse linear algebra routines

- Conversion routines for dense, COO, CSR and CSC formats
- Optimized sparse matrix-vector multiplication for CSR
- New Sparse Triangular Solve CUDA 4.0
  - API optimized for common iterative solve algorithms

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

cuSPARSE is up to 6x Faster than MKL

Sparse Matrix x Dense Vector

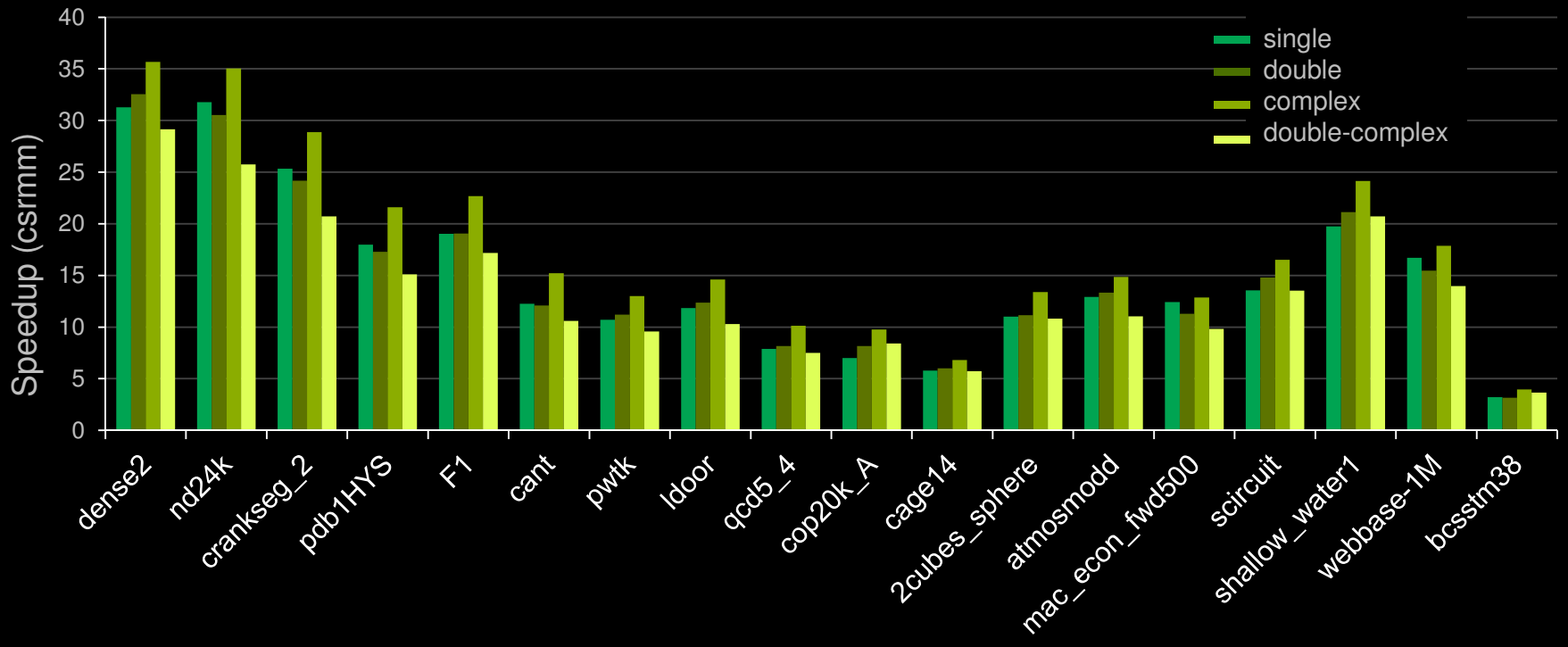


* cuSPARSE 4.0, NVIDIA C2050 (Fermi), ECC on
* MKL 10.2.3, 4-core Corei7 @ 3.07GHz

Performance may vary based on OS version and motherboard configuration

Up to 35x faster with 6 Dense Vectors

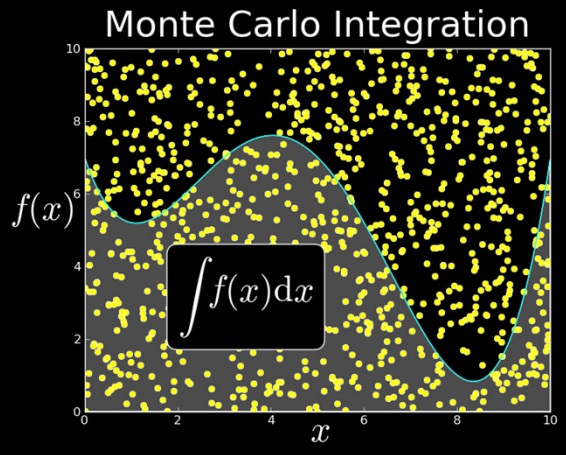
Useful for block iterative solve schemes



* cuSPARSE 4.0, NVIDIA C2050 (Fermi), ECC on
* MKL 10.2.3, 4-core Corei7 @ 3.07GHz

Performance may vary based on OS version and motherboard configuration

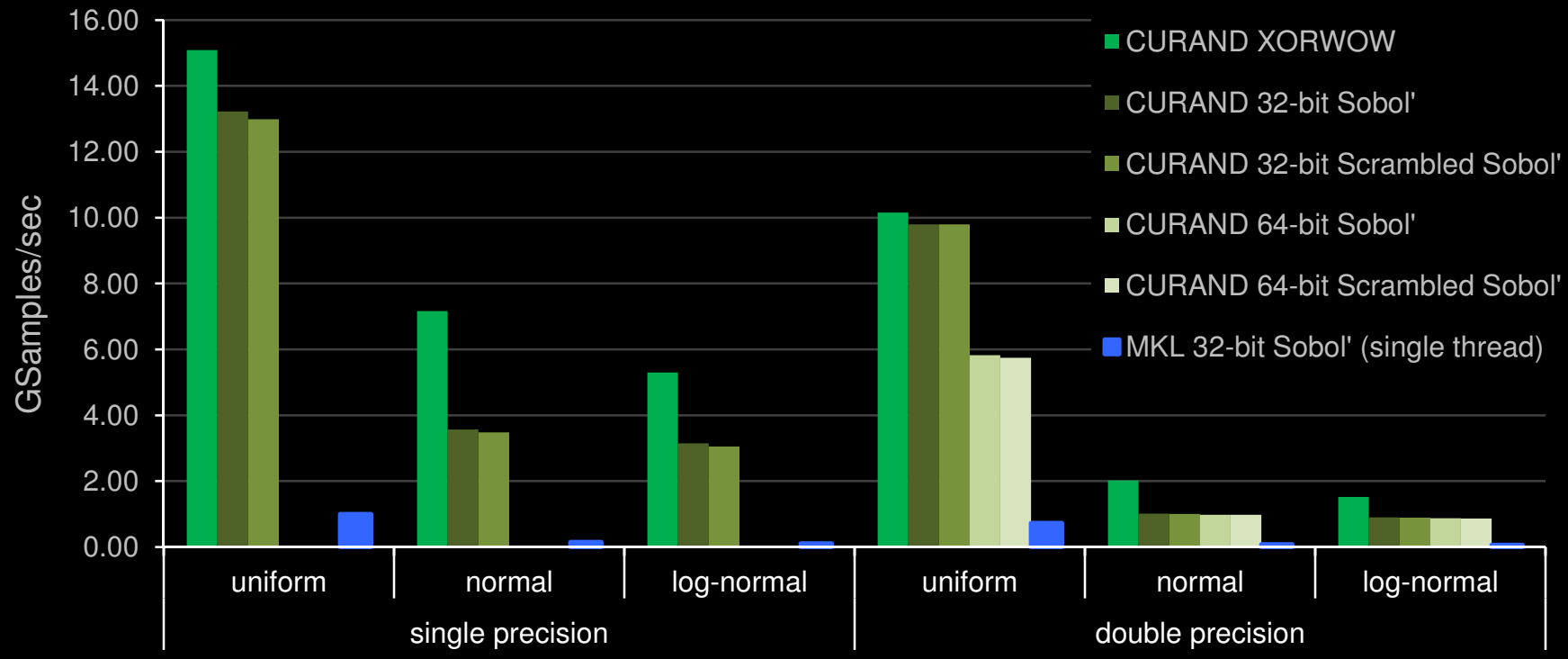
cuRAND: Random Number Generation



- New in CUDA 4.0
  - Scrambled and 64-bit Sobol'
  - Log-normal distribution
  - New parallel ordering supports faster XORWOW initialization
  - Results of CURAND generators against standard statistical test batteries are reported in documentation

cuRAND Performance

cuRAND 64-bit Scrambled Sobol' 8x faster than MKL 32-bit plain Sobol'



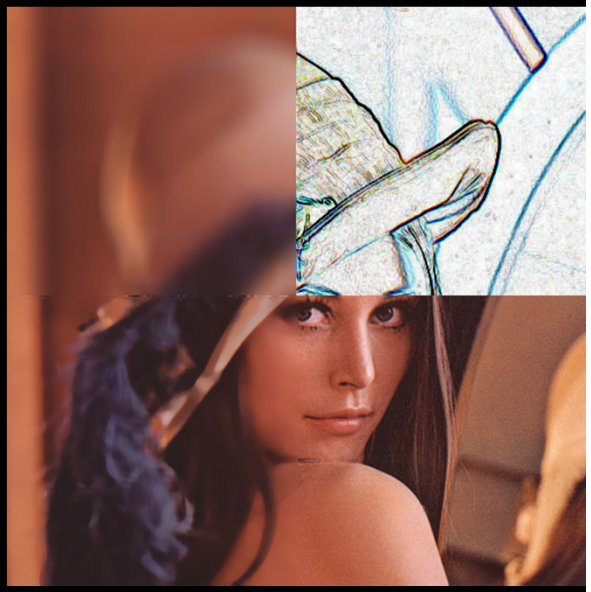
Performance may vary based on OS version and motherboard configuration

* CURAND 4.0, NVIDIA C2050 (Fermi), ECC on

NVIDIA Performance Primitives

Up to **40x** speedups

- Arithmetic, Logic, Conversions, Filters, Statistics, etc.
  - ~420 image functions (+70 in 4.0)
  - ~500 signal functions (+400 in 4.0)
- Majority of primitives 5x to 10x faster than analogous routines in Intel IPP

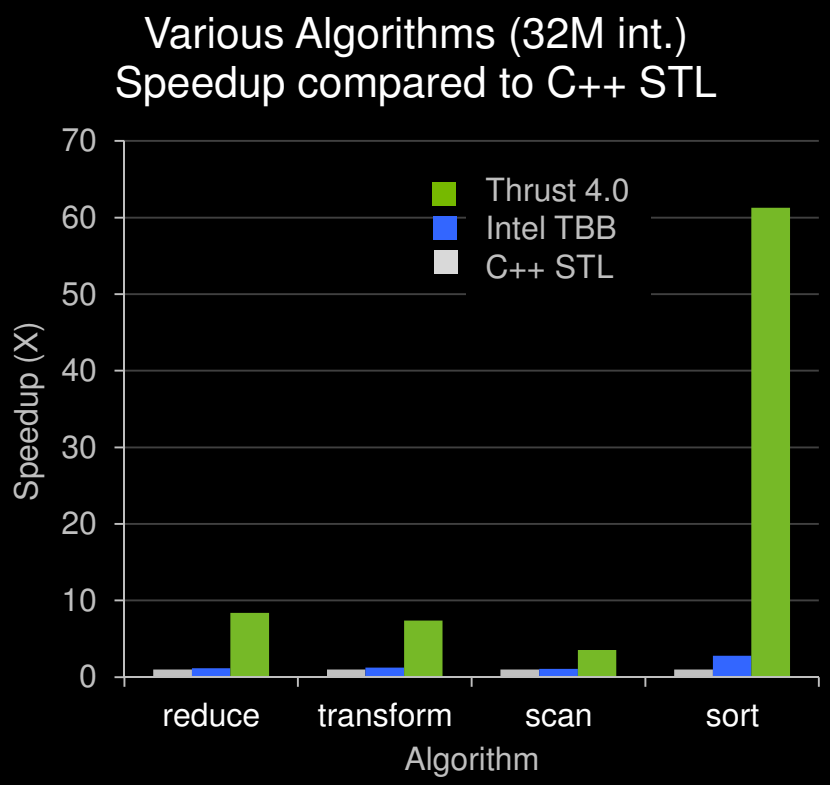


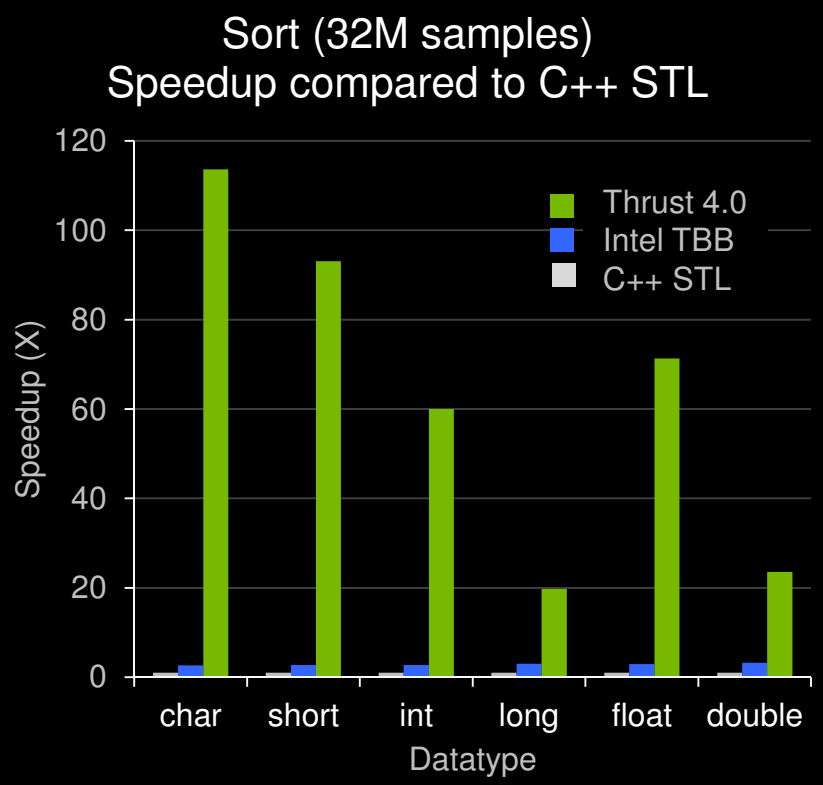
* NPP 4.0, NVIDIA C2050 (Fermi)
* IPP 6.1, Dual Socket Core™ i7 920 @ 2.67GHz

Thrust: CUDA C++ Template Library

- Added to CUDA Toolkit as of CUDA 4.0
  - Also available on Google Code
- Template library for CUDA
  - Host and Device Containers that mimic the C++ STL
  - Optimized algorithms for sort, reduce, scan, etc.
  - OpenMP backend for portability
- Allows applications and prototypes to be built *quickly*

Thrust Algorithm Performance





* Thrust 4.0, NVIDIA Tesla C2050 (Fermi)

* Core i7 950 @ 3.07GHz

math.h: C99 floating-point library + extras

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

- 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, ...

- Performance improvements in CUDA 4.0
  - Double-precision /, rsqrt(), erfc(), & sinh() are all >~30% faster on Fermi
- Added cospi() to CUDA 4.0