NVIDIA CUDA Libraries

Ujval Kapasi*, Elif Albuz*, Philippe Vandermersch*, Nathan Whitehead*, Frank Jargstorff*
San Jose Convention Center | Sept 22, 2010

*NVIDIA
NVIDIA CUDA Libraries

- CUFFT
- CUBLAS
- CUSPARSE (Separate talk: Th 11AM)
- math.h
- CURAND
- NPP
- Thrust (Separate talks: Th 11AM, Th 2PM)
- CUSP
Goal: World Class Performance

- Accelerate building blocks required by algorithms widely used in GPU computing
  - Our team consists of algorithm experts and CUDA experts

- Heavily optimize the most commonly used routines

- Support all CUDA-capable hardware
  - Optimized libraries with hardware launch

- Incorporate best practices from the field
  - Published papers, open source software, academic partners, etc.
Further information

- http://www.nvidia.com/getcuda

- Questions can be posted to the “CUDA Programming and Development” Forum

- Directly approach our CUDA Library engineers right after this talk
CUFFT Library
San Jose Convention Center | Sept 22, 2010
Introduction

NVIDIA CUDA Fast Fourier Transform Library is a GPU based FFT library computing parallel FFTs on NVIDIA GPUs.

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

\[
f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(x) e^{j2\pi \left(\frac{x}{N}\right)n}
\]
CUFFT Library Features

- Algorithms based on Cooley-Tukey and Bluestein
- Simple interface similar to FFTW
- Streamed asynchronous execution
- 1D, 2D and 3D transforms of complex and real data
- Double precision (DP) transforms
- 1D transform sizes up to 128 million elements
- Batch execution for doing multiple transforms
- In-place and out-of-place transforms
Use CUFFT in 3 easy steps

Step 1 - Allocate space on GPU memory

Step 2 - Create plan specifying transform configuration like the size and type (real, complex, 1D, 2D and so on).

Step 3 - Execute the plan as many times as required, providing the pointer to the GPU data created in Step 1.
Performance of Radix-2 (ECC on)

- Up to 8.8x performance advantage over MKL in both single- and double-precision

* MKL 10.1r1 on quad-Corei7 Nehalem @ 3.07GHz
* FFTW single-thread on same CPU
* CUFFT on Fermi C2050
New in 3.2 Release

- Optimized performance of Radix-3, -5, and -7
  - Hence, acceleration of sizes \((2^a \cdot 3^b \cdot 5^c \cdot 7^d)\)

- Bluestein algorithm improves performance and accuracy for large prime transform sizes
  - Up to 100,000x improvement in accuracy for large prime transforms
  - Motivated by customer request

- Support large batches up to the available GPU memory
  - i.e., up to 6GB on C2070
Radix-3 Performance in 3.2

- Up to 18x for single-precision and up to 15x for double-precision
- Similar acceleration for radix-5 and -7

* MKL 10.1r1 on quad-Core i7 Nehalem @ 3.07GHz
* FFTW single-thread on same CPU
* CUFFT on Fermi C2050
Future Releases of CUFFT

Multi-GPU scaling?
Further performance improvements?

Suggestions?
Cublas Features

- Implementation of BLAS (Basic Linear Algebra Subprograms)
- CUBLAS first release in Toolkit2.0 in 2008
- Divided in three categories
  - Level1 (vector, vector):
    - AXPY: \( y = \alpha x + y \)
    - DOT: \( \text{dot} = x.y \)
  - Level 2 (matrix, vector),
    - Vector multiplication by a General Matrix: GEMV
    - Triangular solver: TRSV
  - Level3 (matrix, matrix)
    - General Matrix Multiplication: GEMM
    - Triangular Solver: TRSM
Cublas Features

- Support of 4 types:
  - Float, Double, Complex, Double Complex
  - Respective Prefixes: S, D, C, Z
- Example: SGEMM
  - S: single precision (float)
  - GE: general
  - M: multiplication
  - M: matrix output
- Contains 152 routines: S(37), D(37), C(41), Z(41)
CUBLAS Applications

- Building block for CUDA port of LAPACK
  - CULA from EM Photonics
  - MAGMA from University of Tennessee
- MATLAB acceleration
  - Parallel Computing Toolbox from The Mathworks
  - Jacket from AccelerEyes
- ANSYS, CAE simulation software
- LS-DYNA, developed by Livermore Software Technology, FEA simulation
CUBLAS DGEMM Performance

- CUBLAS is more than 7 times faster than MKL
- 3.2 has only 8% performance variation versus 300% for 3.1
- 30% speedup vs 3.1

*NVIDIA C2050, ECC on
*MKL 10.2.3, i7 4 cores CPU @ 2.66Ghz
Performance GEMM summary

- **SGEMM**: 636 GFLOPS (CUBLAS 3.2), 78 GFLOPS (MKL 4 THREADS)
- **CGEMM**: 775 GFLOPS (CUBLAS 3.2), 80 GFLOPS (MKL 4 THREADS)
- **DGEMM**: 301 GFLOPS (CUBLAS 3.2), 39 GFLOPS (MKL 4 THREADS)
- **ZGEMM**: 295 GFLOPS (CUBLAS 3.2), 40 GFLOPS (MKL 4 THREADS)

---

* NVIDIA C2050, ECC on
* MKL 10.2.3, i7 4 cores CPU @ 2.66Ghz
Future plan

- Optimize TRSM, SYMM
- BLAS1 results returned in Device memory
- Scalar parameters alpha/beta passed by reference, residing on host or device memory.

- Looking for feedback on
  - Workloads that don’t fit within a single GPU
  - Workloads that operate on small matrices
Features

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

- C99 compatible math library, plus extras
- Basic ops: x+y, x*y, x/y, 1/x, sqrt(x), FMA (IEEE-754 accurate in single, double)
- 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, erfinv, erfcinv, ...
Improvements

- Continuous enhancements to performance and accuracy
- Changes based on customer feedback

**CUDA 3.1**  
**erfinvf (single precision)**
- accuracy  
  5.43 ulp → 2.69 ulp  
- performance  
  1.7x faster than CUDA 3.0

**CUDA 3.2**  
**1/x (double precision)**
- performance  
  1.8x faster than CUDA 3.1

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt
\]
CURAND

- New library for random number generation (CUDA 3.2)
- Applications
  - Physical sciences
    - particle physics
    - physical chemistry
  - Finance
    - risk analysis
    - derivatives pricing
Features

• Library interface
  – Pseudorandom generation
  – Quasirandom generation
  – Bits, uniform, normal, floats, doubles

• Kernel interface
  – Inline generation, avoid memory altogether
Pictures

Sobol’ dimension mismatch

Original memory layout
× y × y × y × y × y × y × y × y
× y z × y z × y z × y z × y z

Revised memory layout
× × × × × × y y y y y y
XORWOW Pseudorandom Number Generator

Single thread

\[
t = (x \ ^\ (x \ >> \ 2));
\]
\[
x = y;
y = z;
z = w;
w = v;
\]
\[
v = (v \ ^\ (v \ <<< \ 4)) \ ^\ (t \ ^\ (t \ <<< \ 1));
\]
\[
d += 362437;
\]
\[
return d + v;
\]

\[
\begin{bmatrix}
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x\langle 0 \rangle \\
x\langle 1 \rangle \\
x\langle 2 \rangle \\
x\langle 3 \rangle \\
x\langle 4 \rangle
\end{bmatrix}
\]

\[
f^n(x) = A^n x
\]

\[A^n \text{ is } O(\log n)\]
XORWOW Pseudorandom Number Generator

Parallel threads

\[
\begin{align*}
  f^0(s_0) &\rightarrow f^1(s_0) &\rightarrow \cdots \\
  f^{2^{1.67}}(s_0) &\rightarrow f^{2^{1.67}+1}(s_0) &\rightarrow \cdots \\
  f^{2^{2.67}}(s_0) &\rightarrow f^{2^{2.67}+1}(s_0) &\rightarrow \cdots \\
  \vdots & \vdots & \vdots \\
  f^{2^{4095.67}}(s_0) &\rightarrow f^{2^{4095.67}+1}(s_0) &\rightarrow \cdots
\end{align*}
\]
Customer Feedback To Drive New Features

• More base generators
  – LCG, Mersenne Twister, rand48, ...
  – XOR-256

• More distributions
  – Log-normal, exponential, binominal, ...

• Performance optimizations
NVIDIA Performance Primitives (NPP)

- What is NPP?
- Performance
- Applications
- Roadmap
What is NPP?

• C library of functions (primitives)
  – well optimized
  – low level API:
    • easy integration into existing code
    • algorithmic building blocks
  – actual operations execute on CUDA GPUs
• Approximately 350 image processing functions
• Approximately 100 signal processing functions
Image Processing Primitives

• Data exchange & initialization
  – Set, Convert, CopyConstBorder, Copy, Transpose, SwapChannels

• Arithmetic & Logical Ops
  – Add, Sub, Mul, Div, AbsDiff

• Threshold & Compare Ops
  – Threshold, Compare

• Color Conversion
  – RGB To YCbCr (& vice versa), ColorTwist, LUT_Linear

• Filter Functions
  – FilterBox, Row, Column, Max, Min, Dilate, Erode, SumWindowColumn/Row

• Geometry Transforms
  – Resize, Mirror, WarpAffine/Back/Quad, WarpPerspective/Back/Quad

• Statistics
  – Mean, StdDev, NormDiff, MinMax, Histogram, SqrIntegral, RectStdDev

• Segmentation
  – Graph Cut
NPP Performance

• NPP vs highly optimized Intel CPU code (IPP)
• Majority of primitives 5x to 10x faster
• Up to 40x speedups
• HW:
  – GPU: NVIDIA Tesla C2050
  – CPU: Dual Socket Core™ i7 920 @ 2.67GHz
Applications

• NPP’s image processing primitives accelerate video or still-image processing tasks.

• AccelerEyes’ Matlab Plug-in:
  – “Jacket 1.4 provides direct access to the NVIDIA Performance Primitives or NPP enabling new Image Processing functionality such as ERODE and DILATE.”
NPP Roadmap

• NPP releases in lockstep with CUDA Toolkit:
  – grow number of primitives (data initialization, conversion, arithmetic, ...)
  – complete support for all data types and broad set of image-channel configurations
  – Asynchronous operation support

• NPP 3.2 adds 167 new functions:
  – Mostly data-initialization/transfer and arithmetic
  – New basic signal processing
Additional Information

• On the web:
  – developer.nvidia.com/npp

• Feature requests:
  – npp@nvidia.com
THANK YOU
Q&A Session