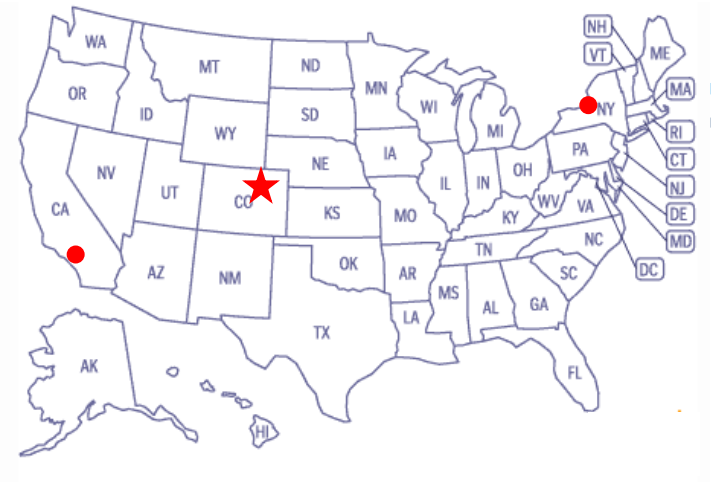
SIMULATIONS EMPOWERING
YOUR INNOVATIONS

GPULIB 1.6

Jon Rood
Tech-X Corporation

TECH-X



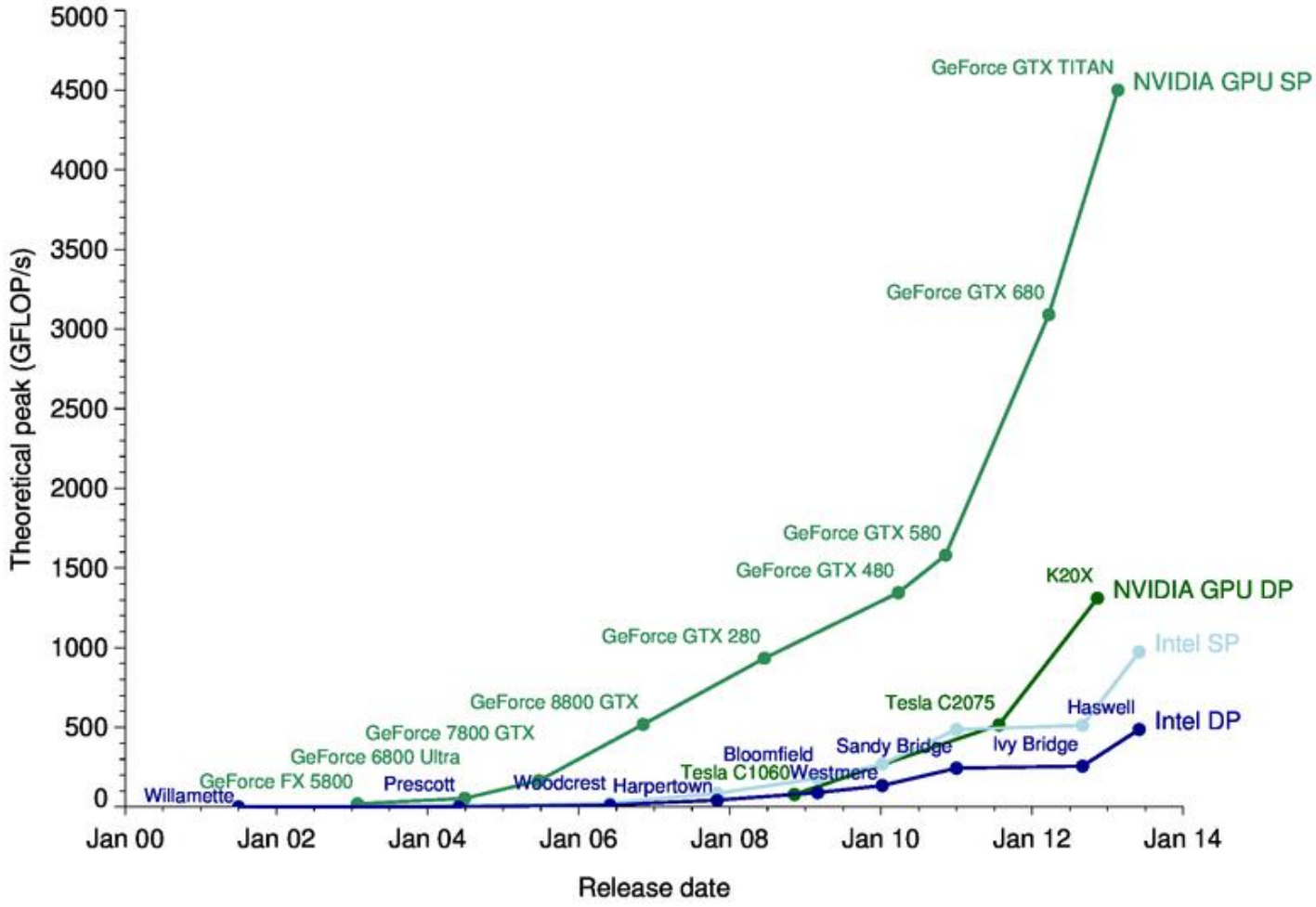
Tech-X's mission is to provide customers with the best computational software and engineering services to enable their breakthroughs in research, development, design, and operations

- High-performance computational software
- High-performance visualization and HPC application front ends
- Enhancing code performance
- Other development including quality and portability

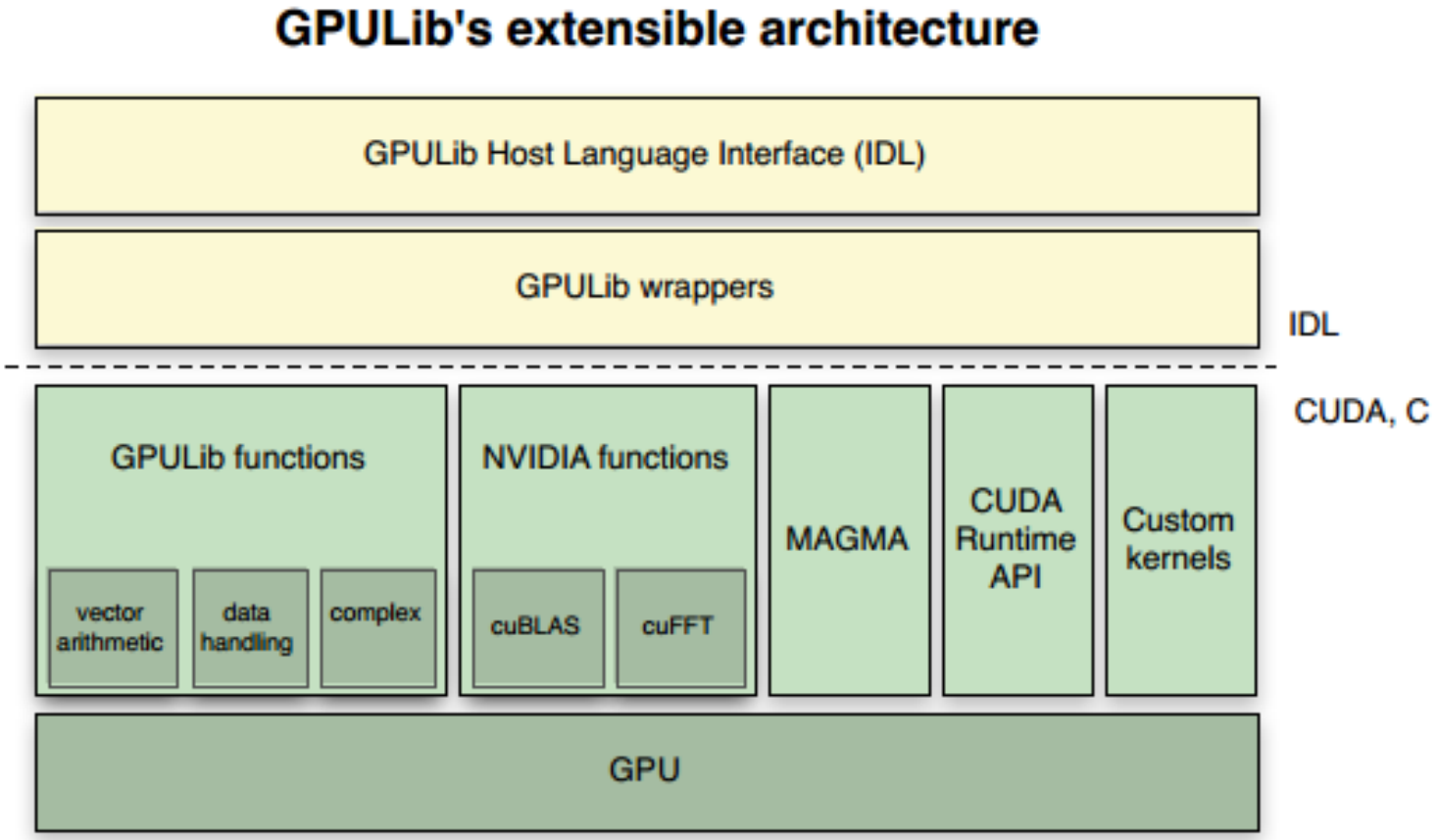
What is GPULib?

GPULib enables users to access high performance computing with minimal modification to their existing programs. By providing bindings between IDL and large function libraries, GPULib can accelerate new applications or be incorporated into existing applications with no knowledge of GPU programming or memory management required.

GPU Performance



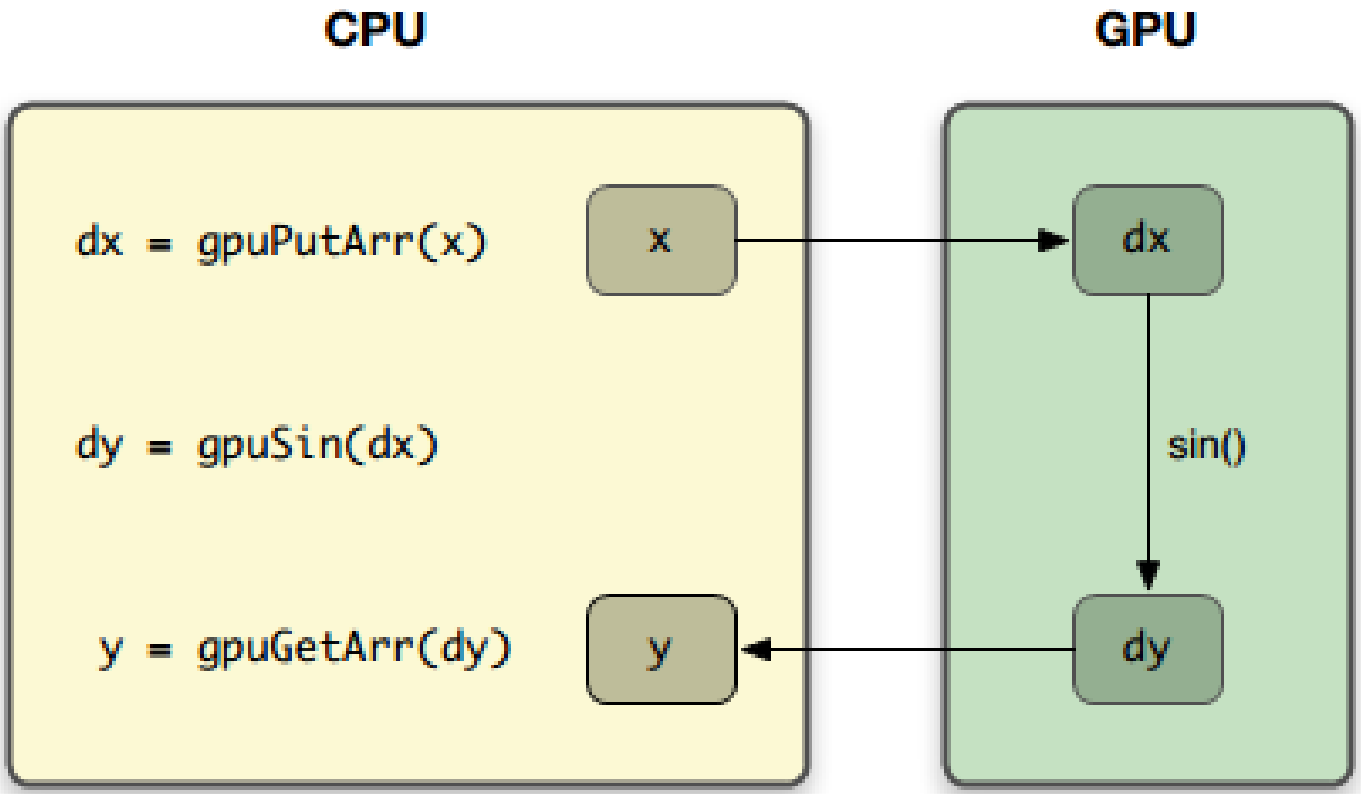
What is GPULib? (continued)



NVIDIA CUDA

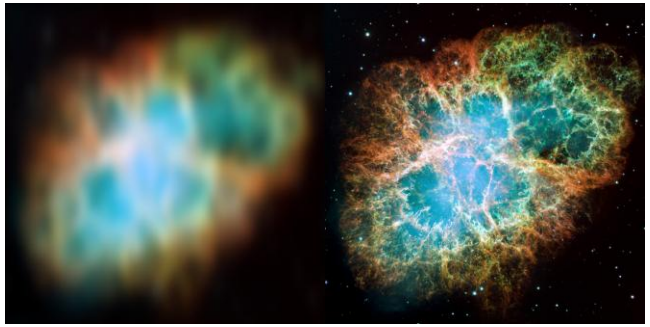
- General purpose graphical processing unit (GPGPU) programming
- Single-instruction multiple-data (SIMD) model
- Well-supported by NVIDIA with libraries for:
  - FFT
  - BLAS
  - Image processing
  - Sparse matrices
  - Etc.

What is GPULib? (continued)



Features of GPULib

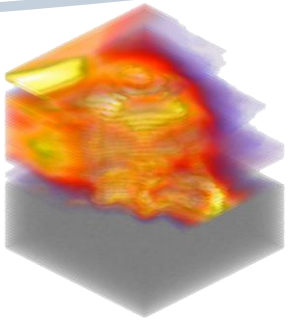
- Basic vector arithmetic
  - Float, double, complex, and double complex
  - Matrix operations (BLAS)
- FFT
  - 1-, 2-, and 3-dimensional, as well as batched
- Interpolation
- Special functions like LGAMMA
- Accelerated versions of common IDL routines like HISTOGRAM and WHERE



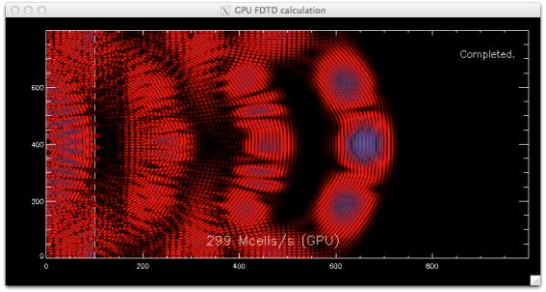
Deconvolution of Hubble Space Telescope image done with GPULib.

Features of GPULib (continued)

- Accelerated special purpose image processing operations
  - Radon transform
- Array indexing and efficient subarray operations
- Use of streams to hide memory transfer times
- Memory transfer/allocation
- Use CPU when CUDA enabled hardware not present



Reconstruction of 3-dimensional object using GPU accelerated Radon transform.



Finite-difference time-domain (FDTD) simulation using GPULib.

Examples (simple)

A simple case calculation:

```
x = findgen(10)
y = findgen(10)
z = x + y
```

In GPULib, this is just:

```
IDL> gpuinit
Graphics card: GeForce GT 330M, compute capability:
  1.2, memory: 112 MB available, 255 MB total
CUDA version: 5.0
MAGMA version: 1.3.0
Checking GPU memory allocation...cudaSuccess
IDL> dx = gpuFindgen(10)
IDL> dy = gpuFindgen(10)
IDL> dz = dx + dy
```

Examples (metadata)

```
IDL> help, dz
DZ              GPUFLOAT  = Array[10]
IDL> print, dz
      0.00000       2.00000       4.00000
      6.00000       8.00000       10.0000
      12.0000       14.0000       16.0000
      18.0000
IDL> print, size(dz)
           1          10          11          10
```

Examples (indexing)

Simple indexing:

```
print, z[0:2]
print, z[[0, 1, 3]]
```

In GPULib, this is straightforward:

```
IDL> print, dz[0:2]
      0.00000      2.00000      4.00000
IDL> print, dz[gpuPutarr([0., 1., 3.])]
      0.00000      2.00000      6.00000
```

Examples (explicit memory)

But we might also do:

```
dx = gpuFindgen(10)
dy = gpuFindgen(10)
dz = gpuFltarr(10, /nozero)
dz = gpuAdd(dx, dy, lhs=dz)
```

New in GPULib 1.6

- Added MAGMA linear algebra routines
  - GPU accelerated LAPACK library
- Added support for up to 8-dimensional arrays
- Installers for Mac OS X, Linux, and Windows users
- Support for CUDA 5.0
- Added optimized scalar/array operations
- Ability to load and execute pre-compiled custom kernels
  - Load and execute custom CUDA code

MAGMA 1.3

Some operations are easy:

```
dinverse = gpuInvert(da, lhs=dinverse)
```

Some are harder:

```
status = gpusgels((byte('N'))[0], $
                  m, n, nrhs, $
                  da->_getHandle(), lda, $
                  db->_getHandle(), ldb, $
                  work, lwork, $
                 info)
```

Custom Kernel Example

```
/* CUDA Kernel Device code
 *
 * Computes the vector addition of A and B into C.
 * The 3 vectors have the same number of elements numElements.
 */
__global__ void
VecAdd_kernel(const float *A, const float *B, float *C, int
  numElements)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;

    if (i < numElements)
    {
        C[i] = A[i] + B[i];
    }
}
```

Custom Kernel Example (continued)

```
ptx_filename = filepath('vectorAdd_kernel.ptx',root=tx_src_root())
ptx_source = gpu_read_ptxfile(ptx_filename)

module = gpuLoadModule(ptx_source, ERROR=err)
kernel = gpuLoadFunction(module, 'VecAdd_kernel', ERROR=err)
```

Custom Kernel Example (continued)

```
n = 20L
dx = gpuFindgen(n)
dy = gpuFindgen(n)
dz = gpuFltarr(n)
nThreadsPerBlock = 256L
nBlocksPerGrid = (n + nThreadsPerBlock - 1L) / nThreadsPerBlock

gpuExecuteFunction, kernel, $
                    dx->_getHandle(), $
                    dy->_getHandle(), $
                    dz->_getHandle(), $
                    n, $
                    GRID=nBlocksPerGrid, $
                    BLOCKS=nThreadsPerBlock, $
                    ERROR=err

z = gpuGetArr(dz)
```

New in GPULib 1.6 (continued)

- Improved performance and support for complex data
  - GPUHISTOGRAM
  - GPUHIST_2D
  - GPUATAN2 for complex and double complex
  - GPUREAL
  - GPUCONJ

Advantages of GPULib

- Speed up IDL code easily
  - No CUDA knowledge required
  - No build process needed
  - Falls back to CPU if no CUDA capable GPU present
  - Makes use of IDL 8.0+ overloaded operators for simple notation
- Easy installation
  - Installers for Windows, Linux, and OS X
- Fully documented API with examples

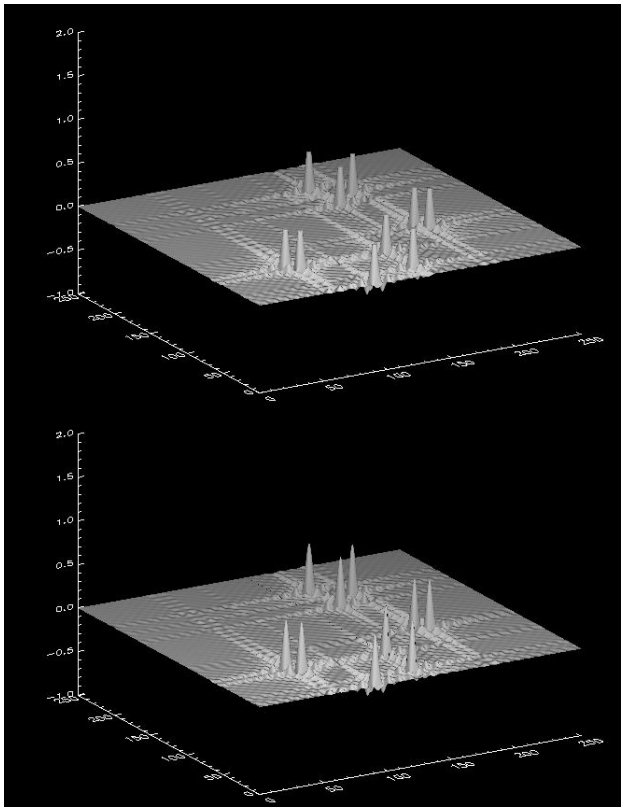
Advantages of GPULib (continued)

Algorithm	Speedup
FFT (250x250 to 10000x10000)	10-60x
FFT based deconvolution	7.7x
LGAMMA (1e6 element arrays)	22.8x
FDTD	25x
Levenberg-Marquardt fitting	6.25x

Using NVIDIA Tesla C2070 vs. Intel Xeon X5640 2.67GHz CPU (12 cores)

Coming in 2013 and beyond...

- Curve fitting
  - Levenberg-Marquardt algorithm
- Runtime kernel creation
  - Evaluate custom expressions in a single kernel
- More image processing routines
- More data types
  - Integer types
- OpenCL support
- CUDA 5.5



Levenberg-Marquardt fitting (bottom) is used to retrieve the value for saturated pixels (top) by fitting a sum of Gaussian functions to the original data. GPU evaluation of the fitting function improves the performance by a factor of 6.25x.

Other Tech-X IDL Products

- We have more IDL products
- HPC IDL/FastDL:
  - GPULib
  - TaskDL
    - Task farming for independent jobs
  - mpiDL
    - Bindings to the MPI interface

TECH-X

SIMULATIONS EMPOWERING
YOUR INNOVATIONS

Michael Galloy: mgalloy@txcorp.com
Jon Rood: rood@txcorp.com