GPULib: GPU Acceleration of Scientific Applications in (Very) High-Level Languages

Peter Messmer
messmer@txcorp.com

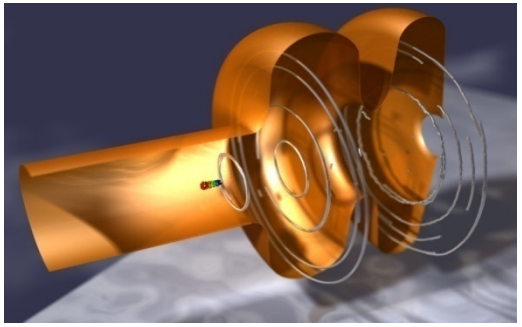
Tech-X Corporation
5621 Arapahoe Ave., Boulder, CO 80303
www.txcorp.com

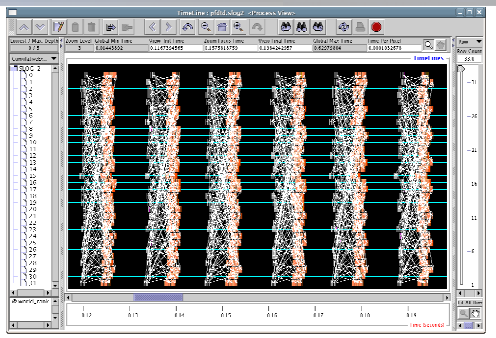
Paul J. Mullowney, Dan Karipides, Keegan Amyx, Nate Sizemore, Brian Granger, Mike Galloy, David Fillmore

This work is supported by NASA SBIR Phase-II Grant #NNG06CA13C

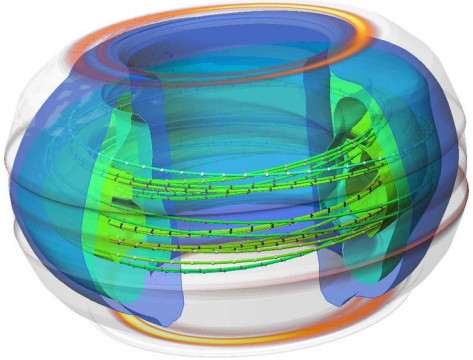
Los Alamos Computer Science Symposium, October 14-15 2008, Santa Fe, NM

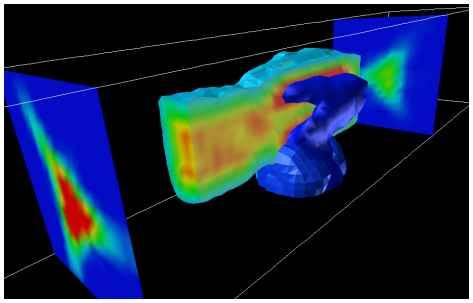
Who are we? What is Tech-X?



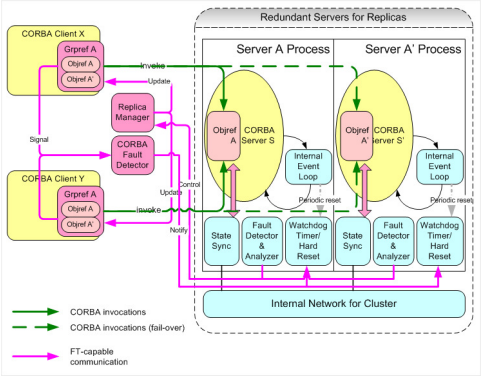


Connecting Physics and HPC

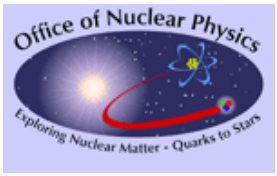




Boulder, CO
~55 employees, 45 PhD
Physicis, CS, Math



And who is paying for that?

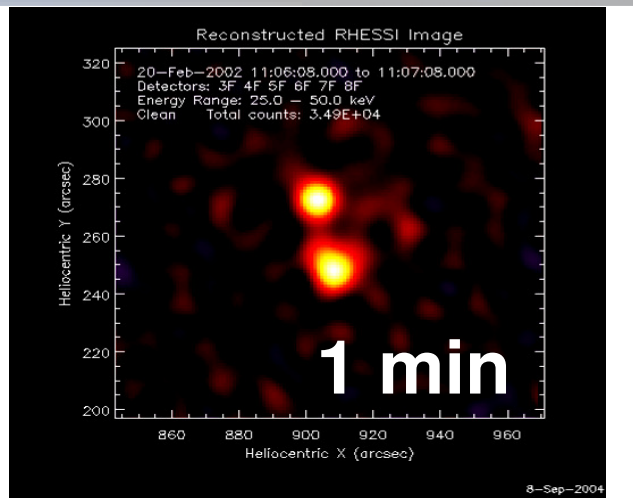


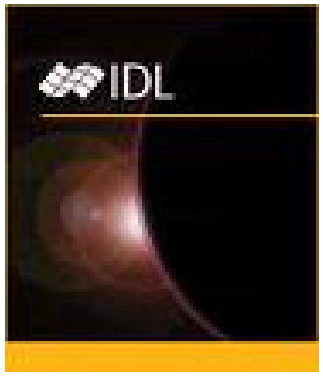
SciDAC
Scientific Discovery through
Advanced Computing

Industrial customers
Aerospace
Plasma processing
Vacuum electronics

The year is 2005..

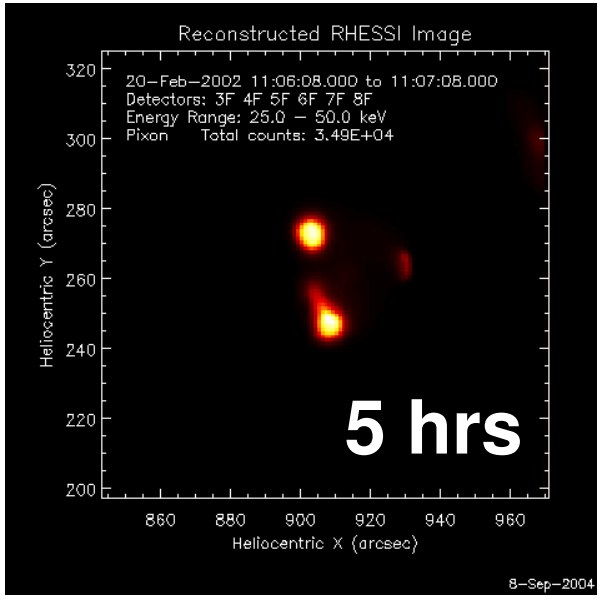
NASA mission is facing a data analysis problem





IDL (Interactive Data Language by ITT VIS) is the tool of choice for data analysis

"People are starved for cycles"



Scientists like to develop in very high-level languages

- **Here "VHLL": IDL (Interactive Data Language), MATLAB, Python**
- **Want to spend their time doing research, not code development**
- **Sociology: Communities "lock-in" on languages**
  - Solar Physics, hyper-spectral imaging: IDL
  - Neuro-Biology, financial modelling: MATLAB
- **Languages offer large collections of domain relevant algorithms**
- **Increasing data volumes: Analysis has to scale as well**
- **=> Conventional cluster computing too cumbersome**
  - Not always access to cluster
  - No desire to write MPI code
  - "Can't you give me something I can plug into my computer and it makes things 10x faster?"

⇒ **Accelerator hardware (focus here on GPUs)**

⇒ CUDA a great architecture, but still requires understanding of the hardware

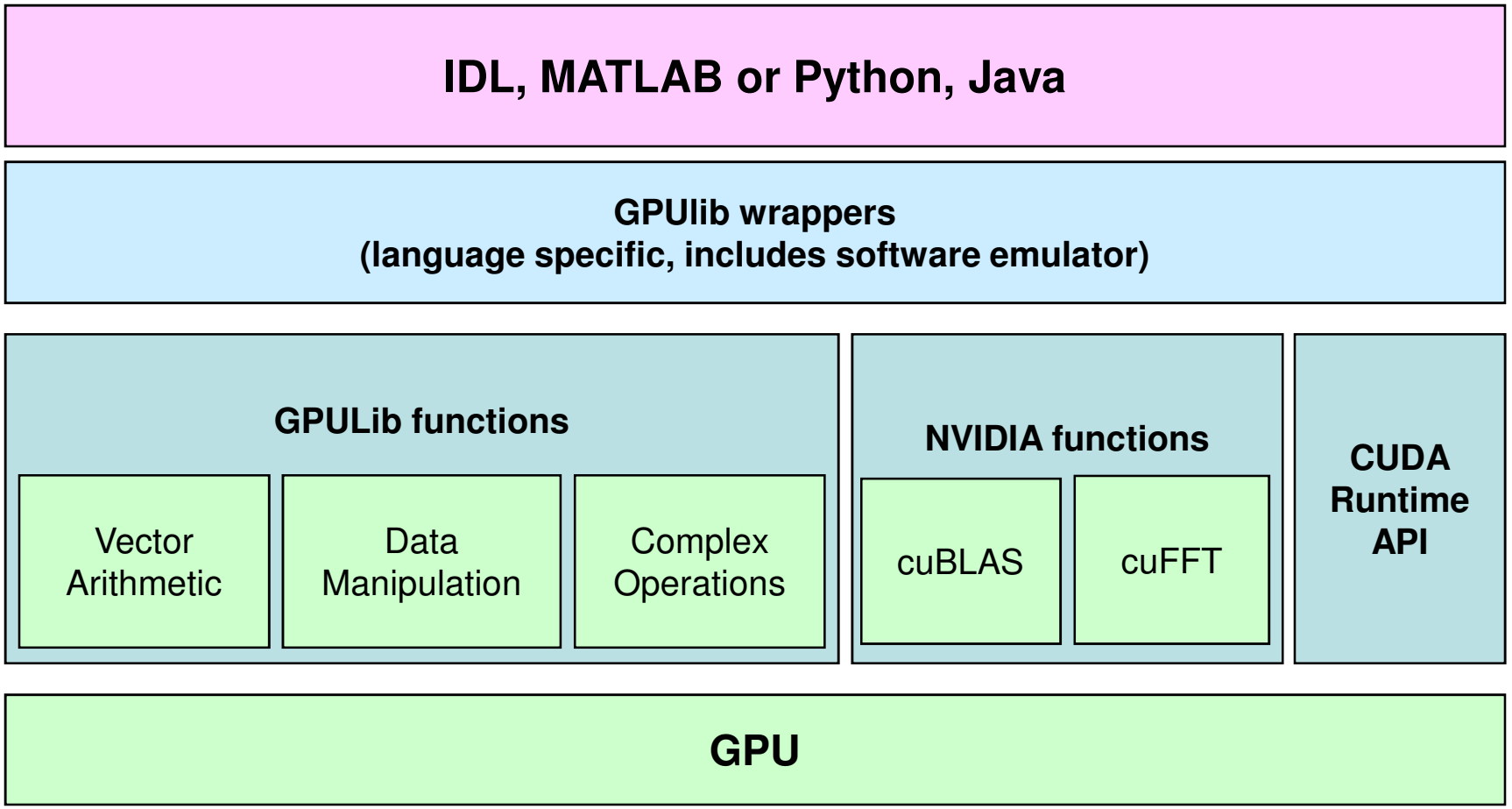
Goal of the project:
Provide acceleration without turning scientists into hardware experts

GPULib design goals: Get speedup from accelerator in a transparent way

- **Accelerators directly usable from within VHLL**
  - Users chose the high-level languages for a reason!
  - Many 4th generation languages vector oriented -> Beneficial to GPU
- **Intuitive for users**
  - Use host language features to make use of accelerators intuitive
- **Code has to remain portable**
  - Key!
  - Provide emulation, but do not incur overhead
- **Take advantage of accelerator**
  - Obtain as high a performance as possible
  - Less than peak is acceptable
- **Provide as many operations as possible on accelerator to reduce data motion**
- **Take advantage of available libraries**
  - cuBLAS, cuFFT
- **Be abstract enough to enable porting to other accelerators**

Messmer, Mullowney, Granger, *"GPULib: GPU computing in High-Level Languages"*, Comuters in Science and Engineering, 10(5), 80, 2008.

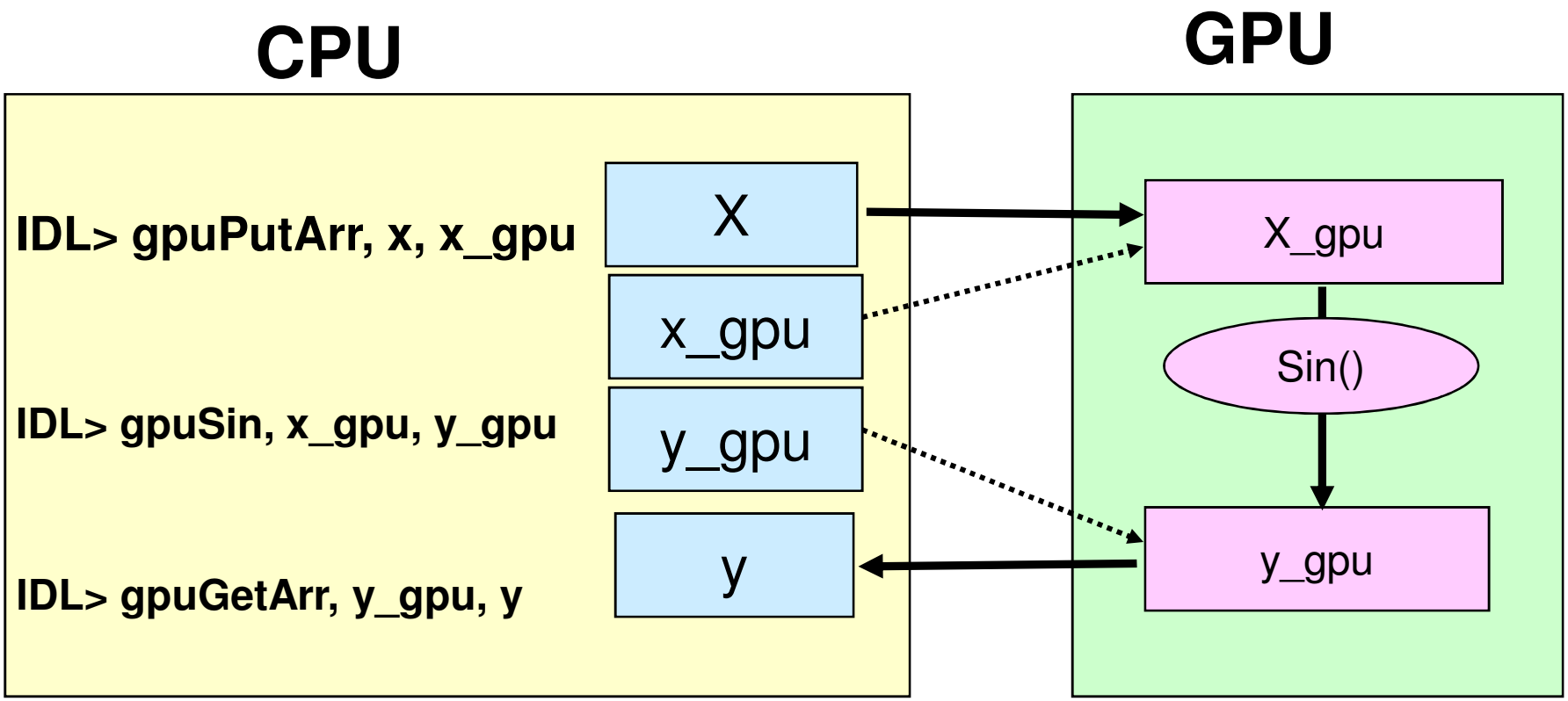
GPULib layered architecture is easily extensible



GPUlib: One way to simplify GPU development

- **GPULib provides a large set of vector operations**
  - Data transfer GPU/CPU, memory management
  - Arithmetic, transcendental, logical functions
  - Data parallel primitives (prefix-sum)
  - Array operations (reshaping, interpolation, range selection, type casting)
  - NVIDIA's cuBLAS, cuFFT
- **Data objects on GPU represented as structure on CPU**
  - Contains size information, dimensionality and pointer to GPU memory
- **Library can be run without the library**
- **Download from http://gpulib.txcorp.com**
  (free for non-commercial use)

A GPULib example in IDL



Can you get all the performance with a vector library?

"Scientists want the control to increase performance as necessary but won't sacrifice everything to performance"

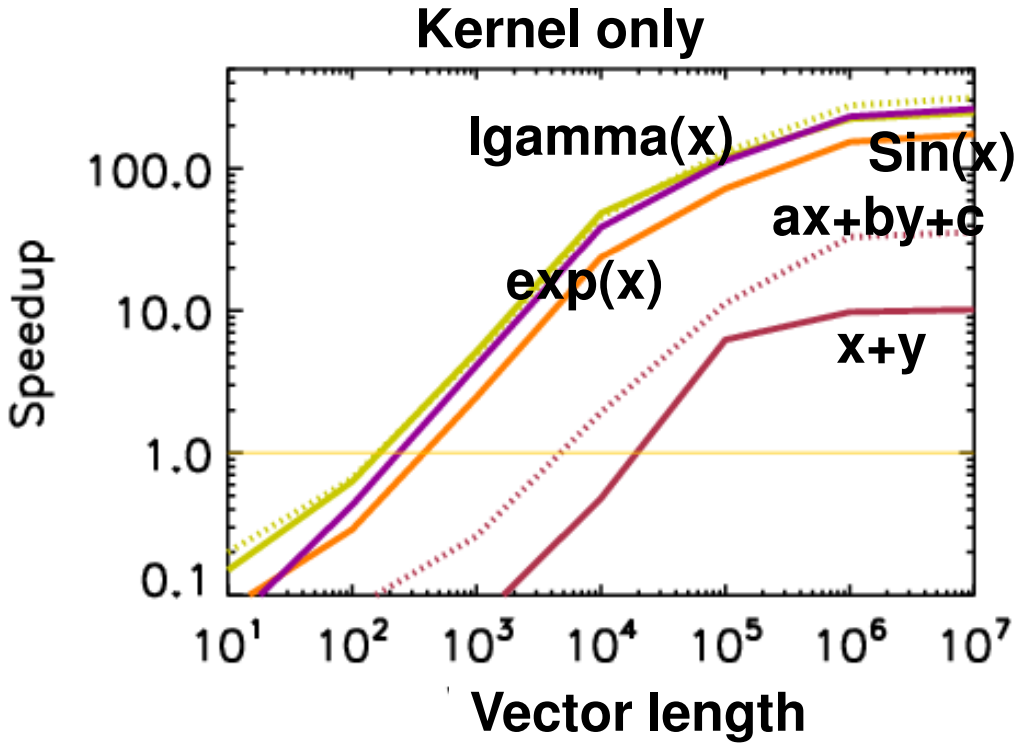
Basili et al, "Understanding the High-Performance Computing Community", IEEE Computer, July '08.

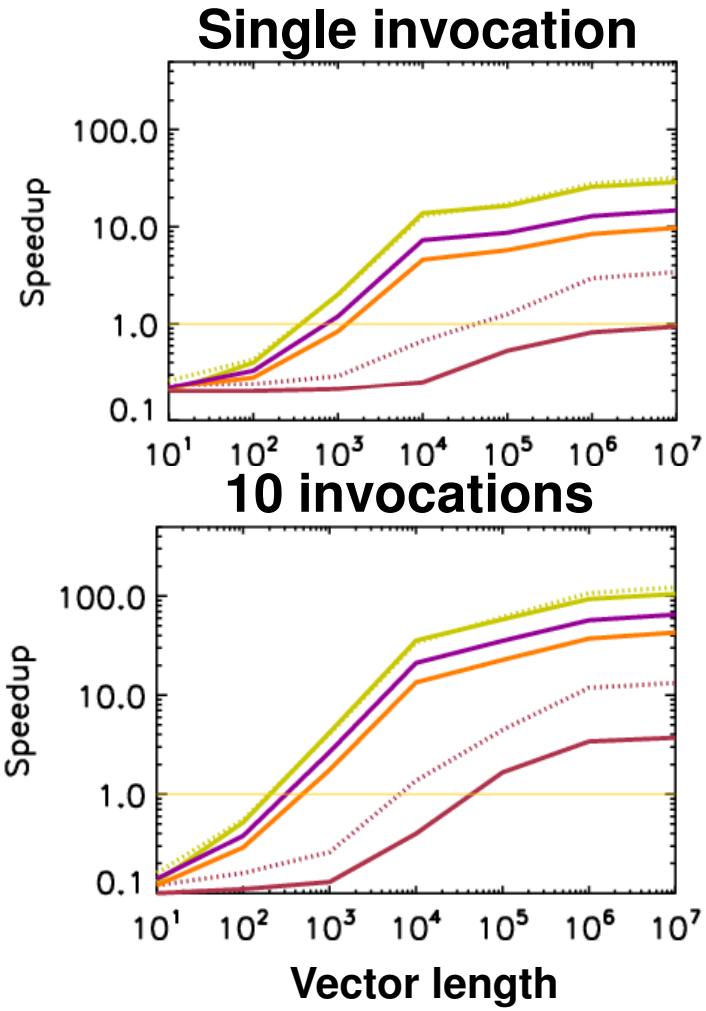
⇒ Vector operations with higher compute density (affine transform of arguments)
z = a **x** + b **y** + c
z = a exp(b **y** + c) + d

⇒ Domain-specific algorithms

How to get performance?

- **Kernels are very fast, GPU<->CPU data transfer is slow**





Example: Image Deconvolution

- Image is convolved with detector point-spread function:

$$I_{obs}(x, y) = \int I_{true}(x-u, y-v)P(u, v)dudv$$

- Clean image by (complex) division in Fourier space:

$$I_{true}(x, y) = FFT^{-1}(FFT(I_{obs}) / FFT(P))$$

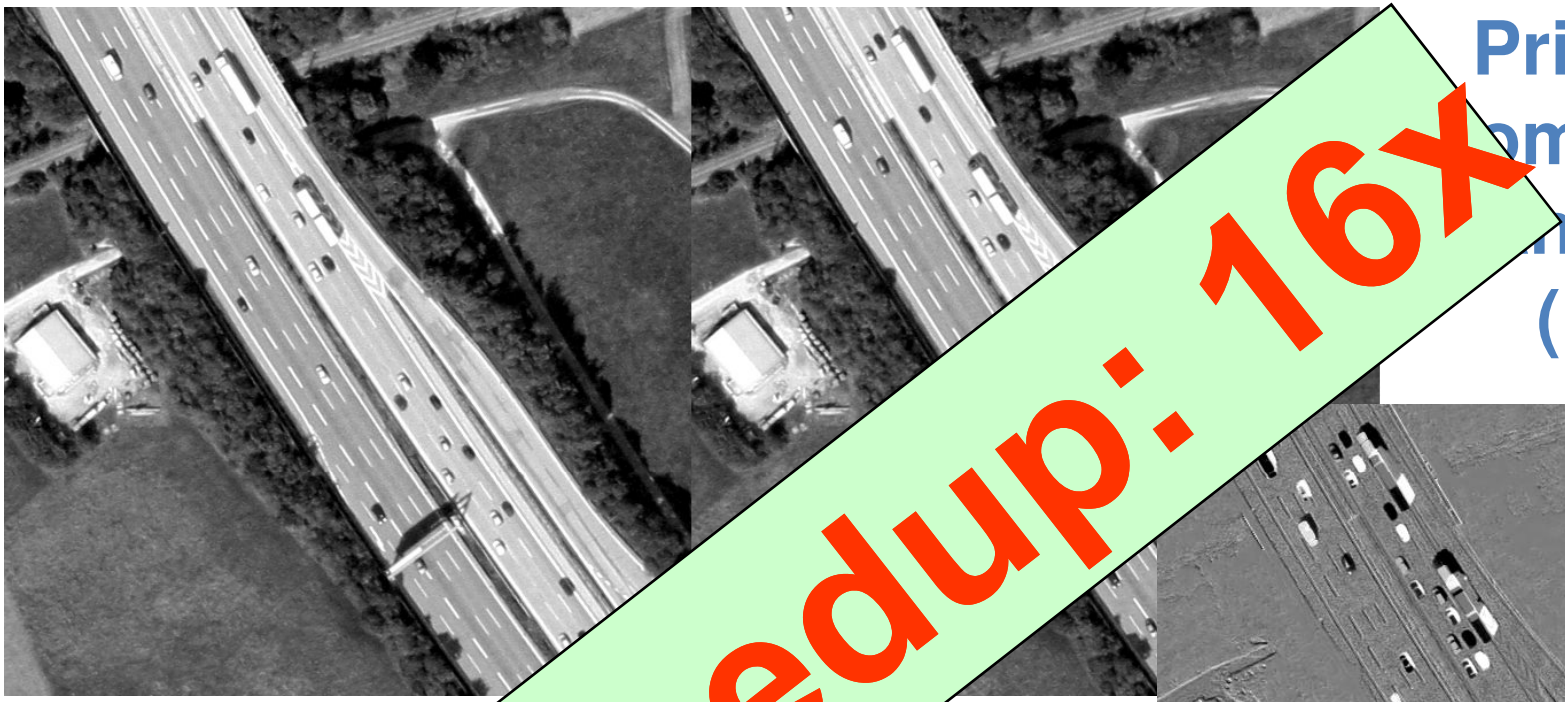
- Large computational load per CPU-GPU data transfer
- Real world problem
- Speedup ranging from 5x – 28x for 256x256 – 3kx3k images

What happened next?

- People downloaded GPULib with interests in
  - Medical Imaging
  - Image Rectification
  - Remote sensing
  - Signal processing
  - Wildlife tracking
  - and many more …

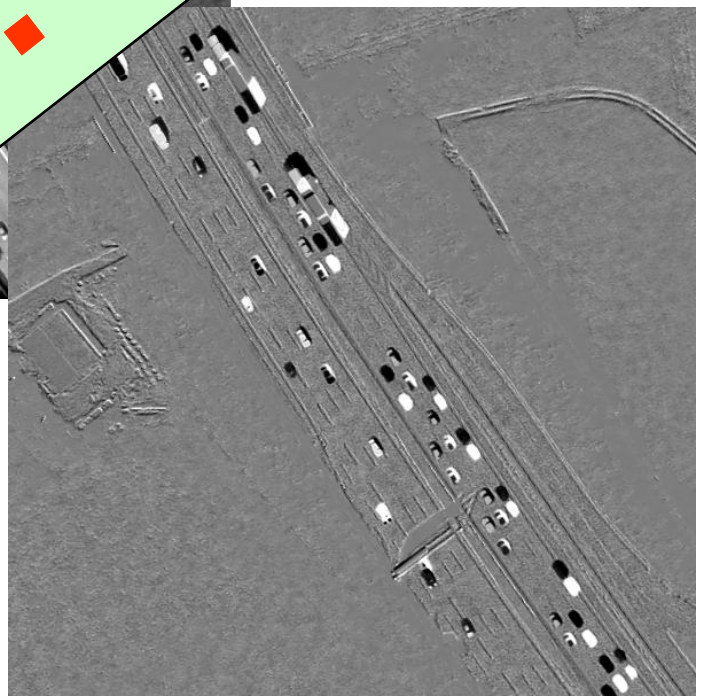
- Customers and evaluations include
  - NASA
  - US AFRL
  - Rutherford Appleton Lab
  - Leiden University, NL
  - Laboratory for Atmospheric and Space Physics (LASP)
  - Many universities …

GPULib example 1: Image processing



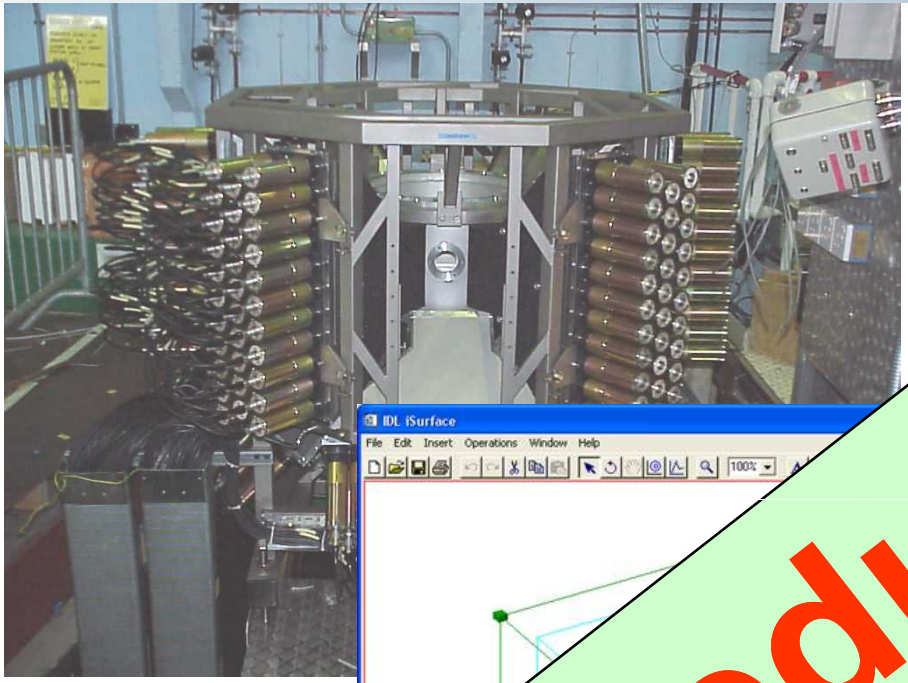
Speedup: 16x

Principal
omponent
nalysis
(PCA)

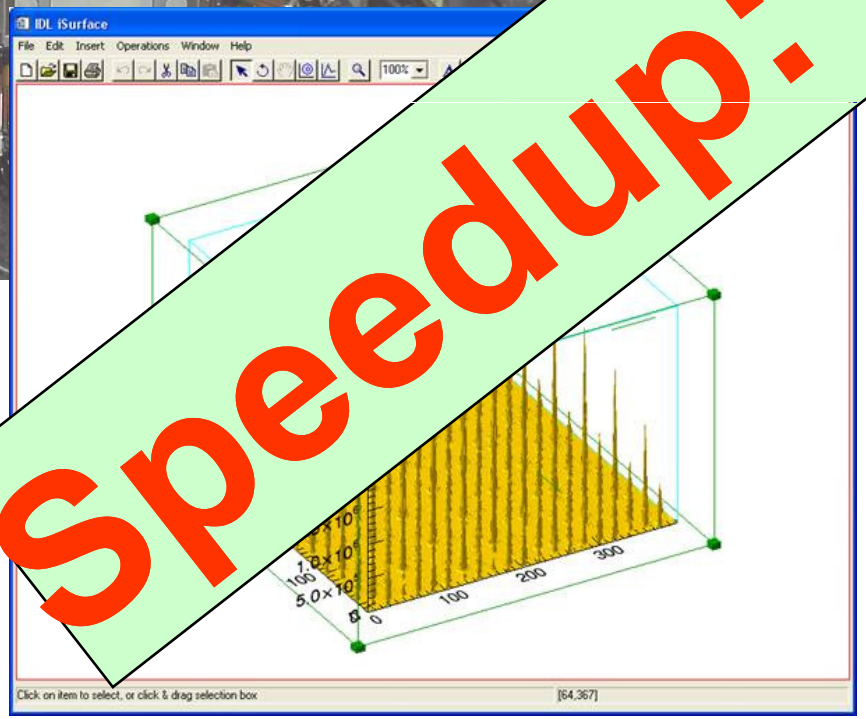


Data courtesy of
Dr. Mort Canty,
FZ Juelich, Germany

GPULib example 3: Simulation



Neutron scattering experiment



Speedup: 10x

Use simulation written in IDL to compute location of scattering maxima (Bragg peaks)

Data courtesy of
Dr. Matthias Gutmann,
Rutherford Appleton
Research Lab, UK

Where we would like to go..

- **More specialized kernels**
  - Collaborate with users to get their performance tuned
  - GPULib enables iterative approach to GPUs/accelerators
- **Performance promising enough that library could act as abstraction for accelerators for "conventional" HPC applications**
  - Unify of C/Fortran interface
- **Develop HPC relevant kernels**
  - Ghost cell exchanges
  - Particle-push kernels
- **Target different accelerators**
  - Portable code for accelerators

Conclusions

- GPUlib offers large set of vector operations on GPU
- Enables users to take advantage of accelerators from within their favourite languages
- One example of accelerator interface that requires no hardware knowledge
- Scientists do not lock in on a particular hardware
- We are happy to collaborate on getting your analysis accelerated on GPUs