

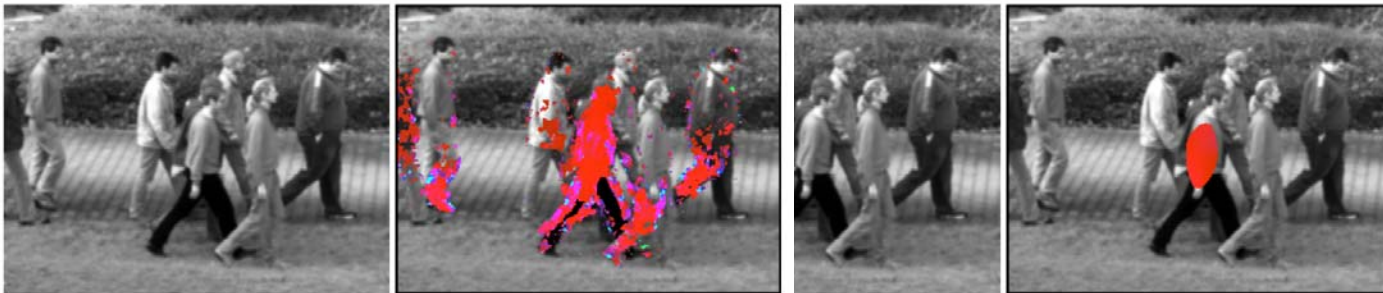
GPGPU Mathematical Primitives

Aaron Lefohn

Neoptica

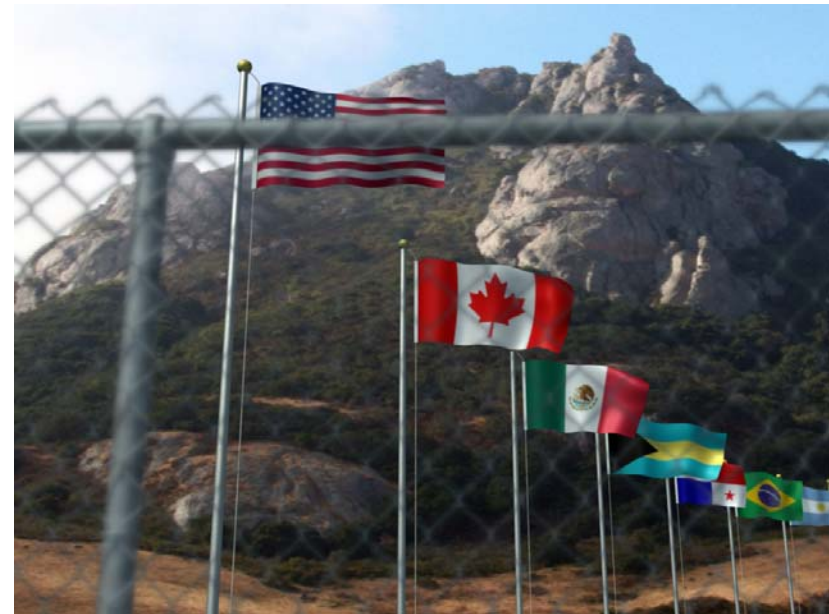


GPGPU Non-Linear PDEs



- Strodka, Garbe, "Real-Time Motion Estimation and Visualization on Graphics Cards," *IEEE Visualization*, 2004

GPGPU Direct Tridiagonal Solver



- 1000, 1000-element tridiagonal linear systems
 - *Kass, Lefohn, Owens, "Interactive Depth-of-Field Using Simulated Diffusion on the GPU," Pixar Technical Report, 2006*

Overview

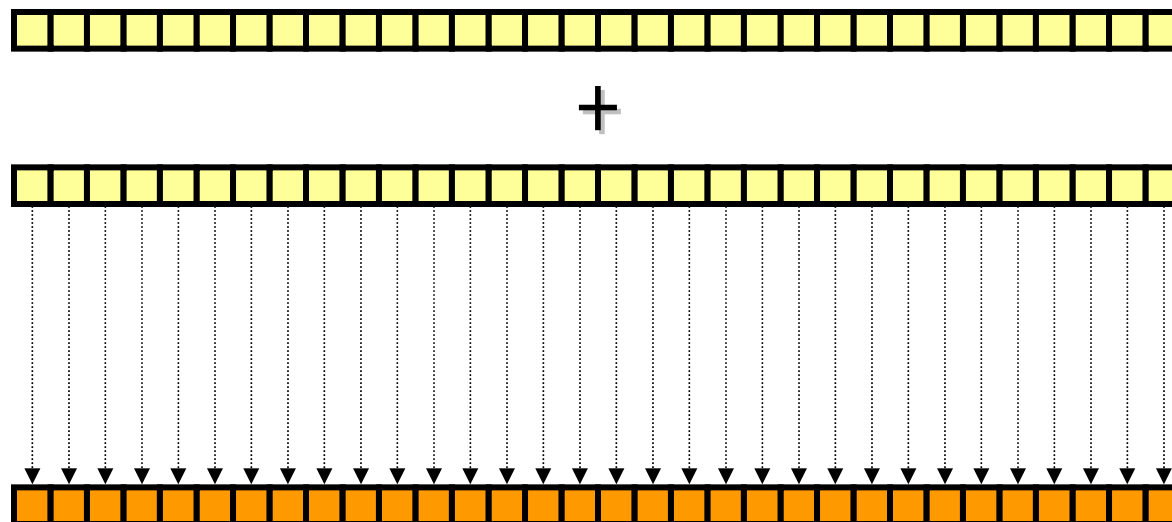
- Linear Algebra
- Differential Equations
- Performance Results
- Summary

Linear Algebra on GPUs

- The basics
 - Vector-vector
 - Matrix-vector
 - Matrix-matrix

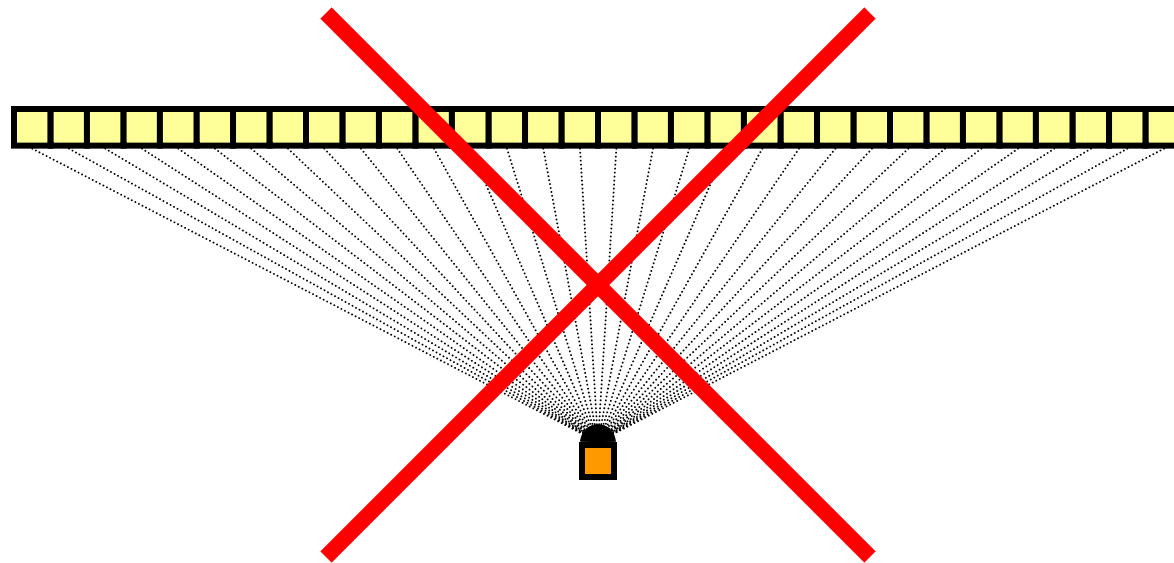
Basics: Vector-Vector Operations

- Add / subtract
 - Trivial parallel `foreach` operation



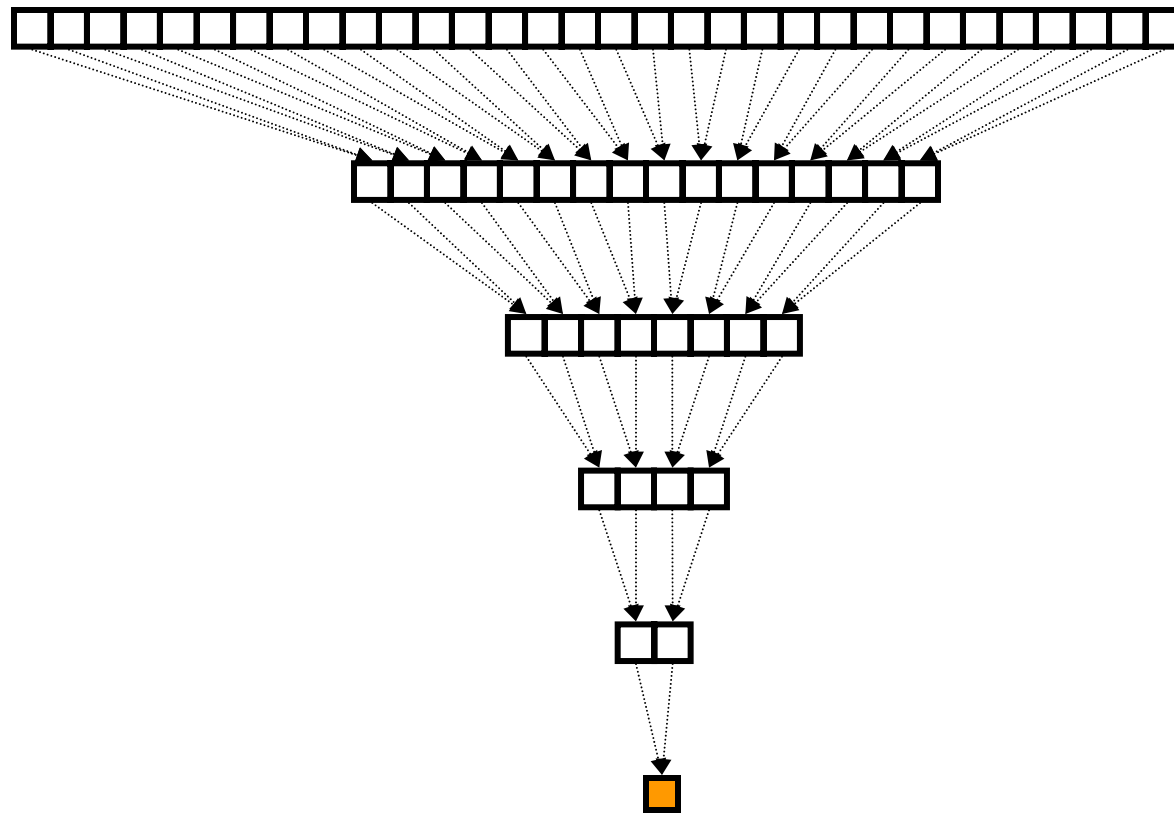
Basics: Vector-Vector Operations

- Inner product / normalize
 - Trivial implementation might render a single fragment (1 thread) and perform serial computation
 - No parallelism



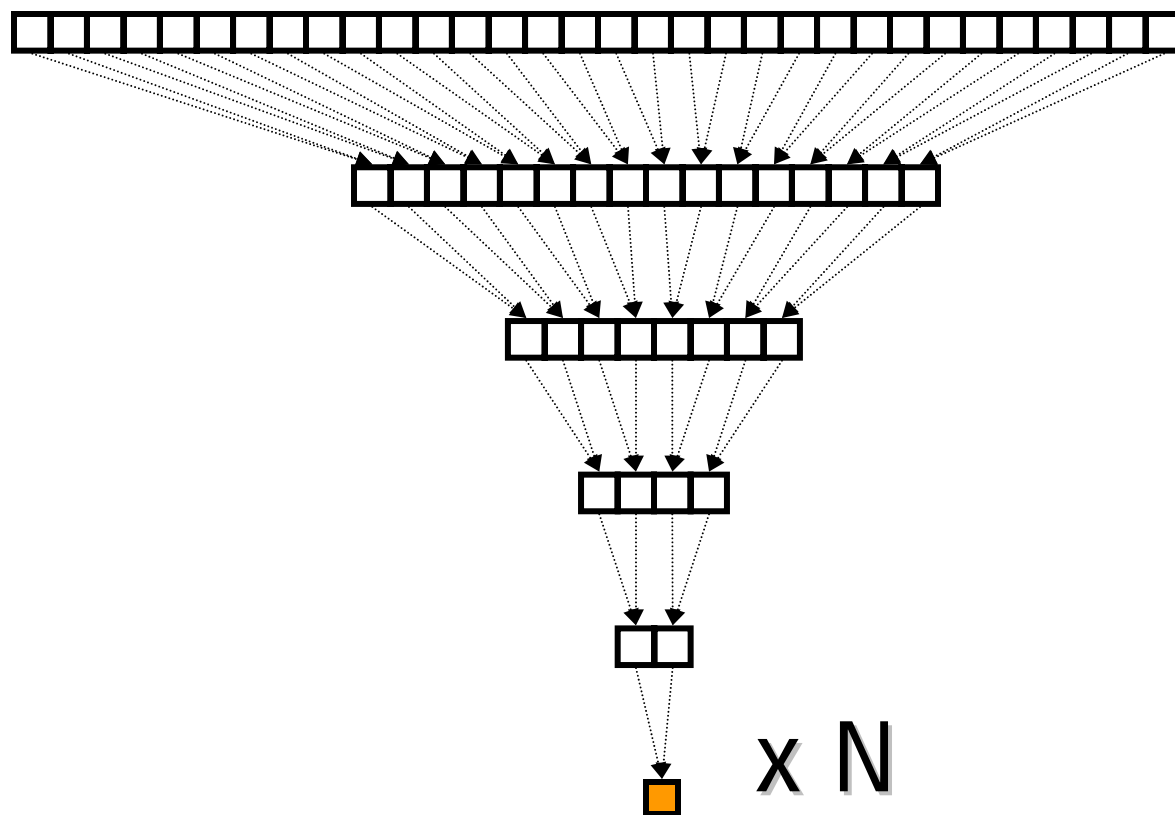
Basics: Vector-Vector Operations

- Inner product / normalize
 - Parallel reduction



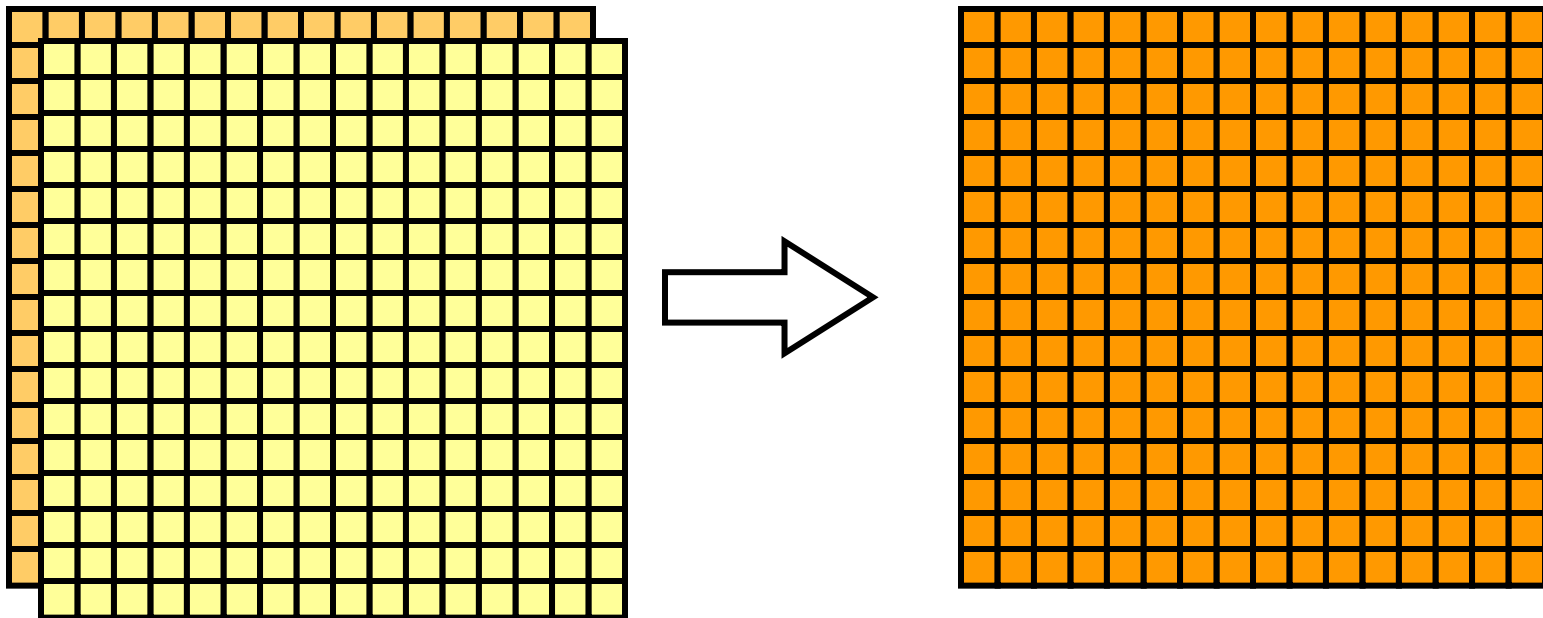
Basics: Matrix-Vector Multiplication

- N inner products in parallel



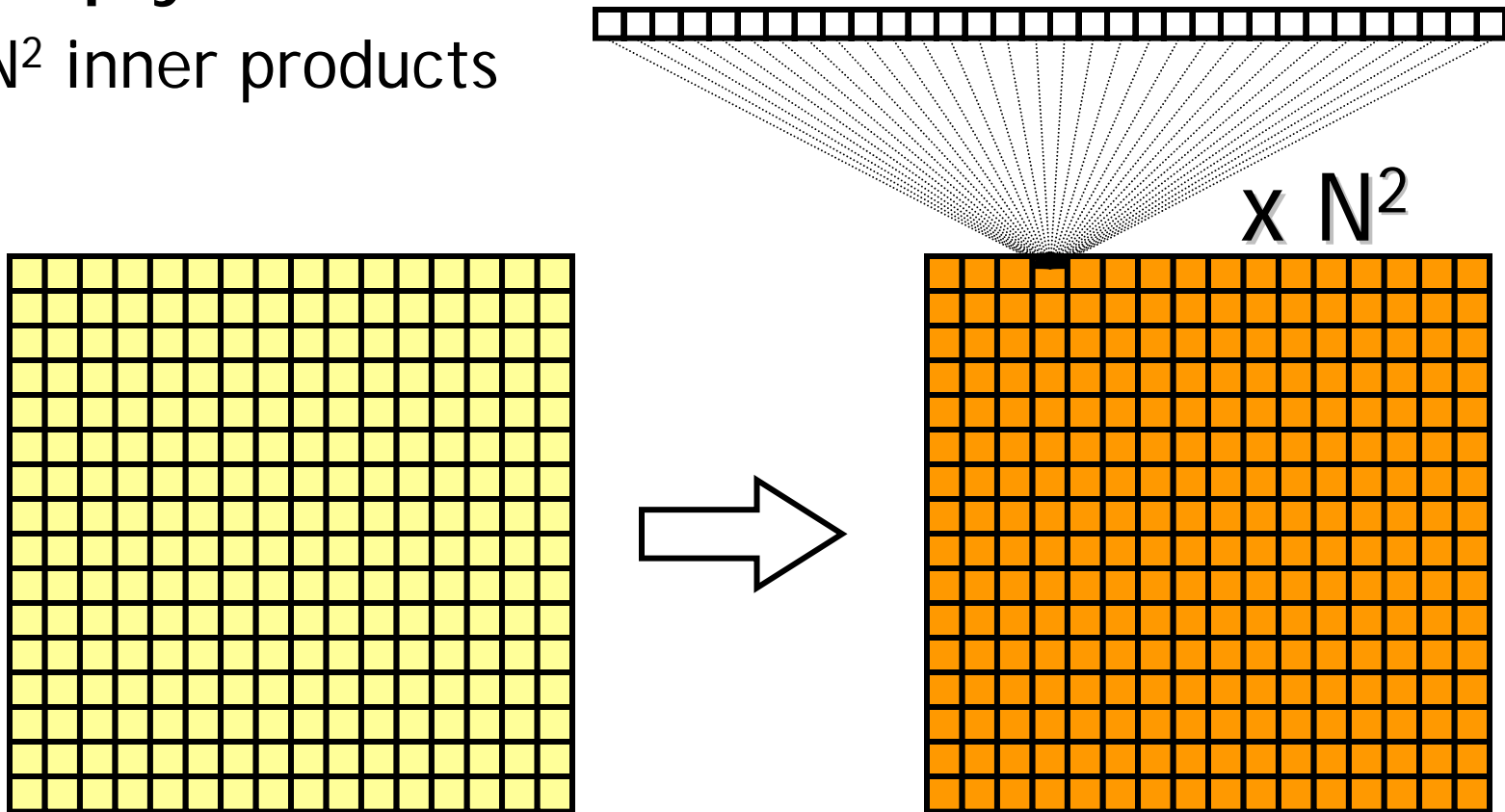
Basics: Matrix-Matrix Operations

- Add / transpose
 - Parallel `foreach`



Basics: Matrix-Matrix Operations

- Multiply
 - N^2 inner products



Basics: Matrix-Matrix Multiply

- **Interesting parallelism note**
 - N^2 inner products provide enough parallelism that it is OK to perform each one in a single pass
 - This differs from computing a single inner product, which must be parallelized for good performance
- **Performance challenges**
 - No writeable cache to capture reuse
 - (Until NVIDIA G80...)
 - Cache-to-register pathway is bottleneck on many GPUs

Basics: Sparse Matrices

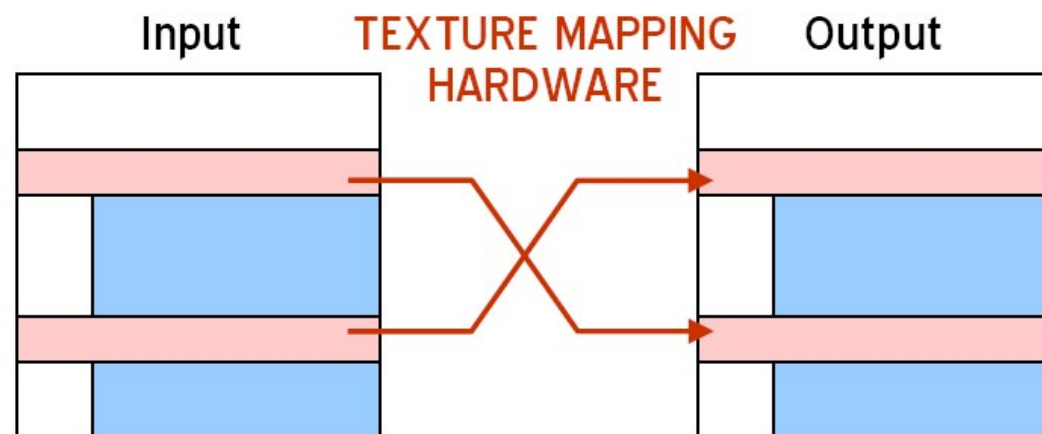
- **N-Diagonal (Banded)**
 - Store each diagonal as vector
 - Special cases: 1-4-Diagonal
 - Store diagonals in quadword elements of single vector
- **Unstructured**
 - ITPACK format (padded compressed row) is attractive (comes with Brook distribution)
 - Same number of non-zero elements in each row
 - Keeps computation SIMD
 - Other formats use more indirection and a varying amount of computation per vector element

GPU Linear Solvers

- Solve $My = x$ for y
 - Use basic linear algebra parallel constructs
 - Data-parallel algorithms
- Examples
 - Conjugate gradient
 - Jacobi
 - Gauss-Seidel
 - Dense LU-decomposition
 - Tridiagonal LU-decomposition

Dense LU-Decomposition

- GPU iterators (rasterization quads)
 - Match access patterns to GPU memory layout
 - Leverage texture/raster engine for pivoting



Galoppo, Govindaraju, Henson, Manocha,

"LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware,"

ACM/IEEE Supercomputing, 2005

Direct Tridiagonal Linear Solver

$$\begin{pmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & a_n & b_n \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_n \end{pmatrix} = \begin{pmatrix} h_1^0 \\ h_2^0 \\ h_3^0 \\ \vdots \\ h_n^0 \end{pmatrix}$$

- Numerical recipes algorithm is sequential
 - Uses forward- and back-substitution
 - Recurrence relation is "impossible" to parallelize

```
for (int j = n - 2; j >= 0; --j) {
    u[j] -= gam[j+1] * u[j+1];
}
```


Tridiagonal LU-Decomposition

- Use parallel scan operation to parallelize
 - Cyclic reduction
 - $O(N)$ computation in $O(\log N)$ passes
 - Can solve 1000s of tridiagonal linear systems (each with 1000s of elements) in parallel

Kass, Lefohn, Owens

*"Interactive Depth of Field Using Simulated Diffusion on the GPU,"
Pixar Animation Studios Technical Report #06-01, 2006*

GPGPU Differential Equations

- Ordinary differential equation example
- Partial differential equation examples

GPGPU ODEs

- N-body particle system

- Brute force solution maps well to GPU

- “Stream all N particles past all N particles”

```
foreach pi in Particles
  foreach pj in Particles
    pi += computeInteraction(pi, pj)
```

- Replace outer loop with parallel GPU **foreach**

- Killer-app

- Folding@Home (see Mike Houston’s upcoming talk)

GPGPU ODEs

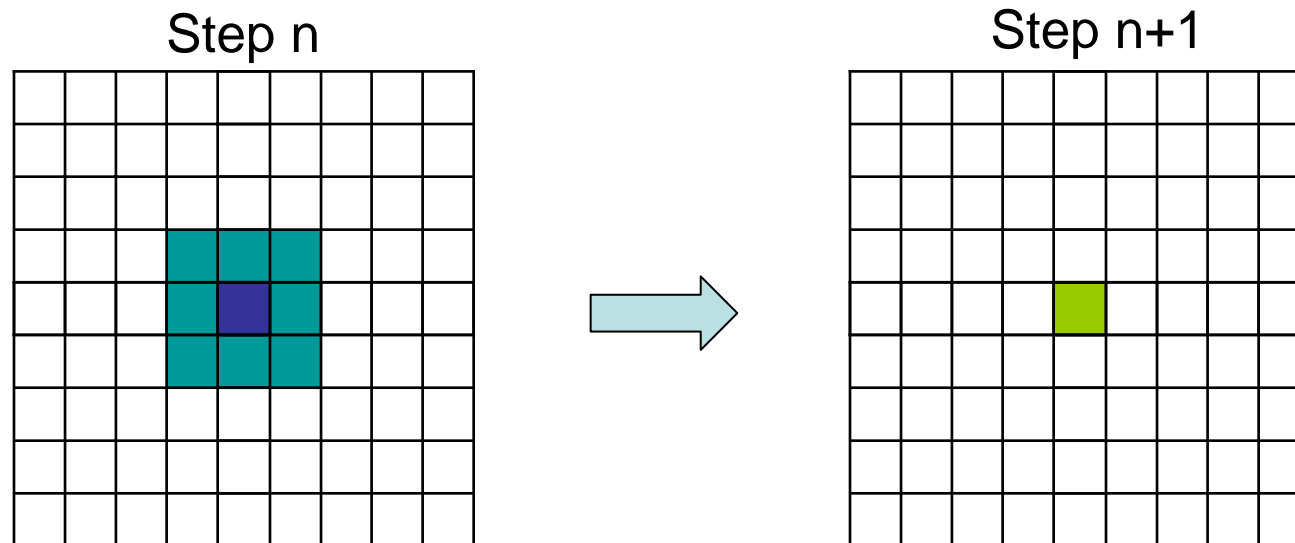
- $O(N \log N)$ optimized algorithms
 - More difficult to map to GPUs
 - Must build irregular data structure each iteration
 - Neighbor lists or hierarchical grid
 - Varying number of interactions per particle
 - Architectural improvements making this easier
 - Scatter
 - More per-thread storage
 - More efficient conditional execution

GPGPU Partial Differential Equations

- **Example GPGPU PDE Applications**
 - Navier-Stokes (incompressible fluids)
 - Level sets (deformable implicit surfaces)
 - Image processing
 - Registration
 - Segmentation
 - Computer vision

GPGPU Partial Differential Equations

- Explicit, finite difference PDE solvers map well to GPUs
 - Gather small number of local neighbors
 - Grid \rightarrow texture



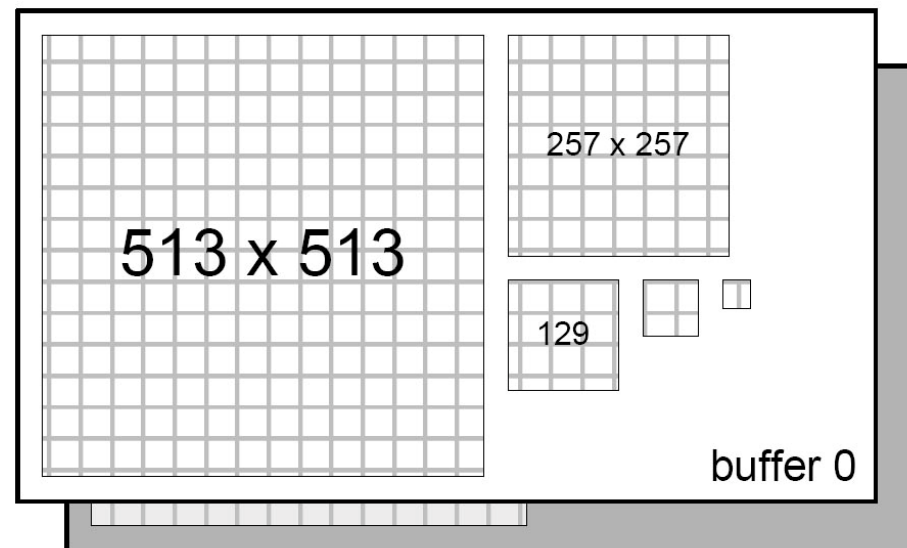
(Figure from Robert Strzodka)

GPGPU PDEs

- Finite difference optimizations
 - Multigrid
 - Banded sparse grids
 - Adaptive grids

GPGPU PDEs

- Multigrid solvers are a good fit for GPUs
 - Multigrid hierarchy \rightarrow Mipmap hierarchy
 - GPU is fast at building dense, hierarchical structures



Goodnight, Wooley, Lewin, Luebke, Humphreys

"A Multigrid Solver for Boundary Value Problems Using Programmable Graphics Hardware,"

ACM/Eurographics Graphics Hardware, 2003

GPGPU PDEs

- Banded sparse grids

- Easy if sparse structure is fixed
- Harder if sparse structure is dynamic
- Examples:

- Mesh deformation uses fixed sparse structure

Bolz, Farmer, Grinspun, Schröder,

"Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid,"

ACM SIGGRAPH, 2003

- Level-set implicit surface deformation has dynamic sparse structure

Lefohn, Kniss, Hansen, Whitaker,

"Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware,"

IEEE Transactions on Visualization and Computer Graphics, 2003/2004

GPGPU PDEs

- Adaptive grids
 - Very little work published
 - GPU must update adaptive data structure each step
 - Coming soon...

Performance Results

- **Dense LU-Decomposition (SC '05)**
 - 15% - 35% faster than ATLAS (partial pivot)
 - Matrix sizes $> 3500^2$
 - NVIDIA GeForce 7800 / Intel Pentium 4 3.4 Ghz
 - Up to 10x faster than LAPACK (full pivot)
 - Intel Math Kernel Lib
 - Matrix sizes $> 3500^2$
 - NVIDIA GeForce 7800 / Intel Pentium 4 3.4 Ghz

Performance Results

- **Matrix-matrix multiply**
 - GPU: 110 GFLOPS (ATI X19K, CTM)
 - CPU: 8-10 GFLOPS (Single Intel P4 3.2 GHz)
 - Cell: > 200 GFLOPS (3.2 GHz)

- **NVIDIA CUDA BLAS**
 - See Ian Buck's talk and/or ask NVIDIA 😊

Summary

- **Techniques**

- Use data-parallel linear algebra algorithms
- Redefine memory access patterns (iterators) for GPU
 - Contiguous output domain
 - Avoid scatter (maintain parallelism)
 - Prefer regular computations
 - Minimize indirections

Summary

- **Challenges**
 - No writeable cache / local store
 - Hard to beat CPU-based block decomposition
 - Good news—NVIDIA G80 has thread-local memory!
 - Must combine multiple operations before reading data back to CPU
- **Iterative solvers work very well!**

GPGPU Math Libraries

- LU-GPU (dense LU-decomposition)
 - <http://gamma.cs.unc.edu/LUGPULIB/>
- Linear algebra framework
 - <http://wwwcg.in.tum.de/Research/Publications/LinAlg>
- GPUFFT
- GPU FFT
 - <http://sourceforge.net/projects/gpufft/>
- NVIDIA CUDA BLAS/FFT
 - <http://developer.nvidia.com/object/cuda.html>
- PeakStream
 - <http://www.peakstreaminc.com/>
- RapidMind
 - <http://www.rapidmind.com>

References

- Kass, Lefohn, Owens, "Interactive Depth of Field Using Simulated Diffusion on the GPU," (Cyclic reduction, direct tridiagonal linear solver), Pixar Technical Report, 2006
- Galoppo, Govindaraju, Henson, Manocha, "LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware," ACM/IEEE Supercomputing, 2005
- Jiang, Snir, "Automatic Tuning Matrix Multiplication on Graphics Hardware," Parallel Architecture and Compilation Techniques (PACT), 2005
- Fatahalian, Sugerman, Hanrahan, "Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication," ACM/EG Graphics Hardware, 2004
- Lefohn, Kniss, Hansen, Whitaker, "Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware," IEEE Transactions on Visualization and Computer Graphics, 2004
- Strodka, Garbe, "Real-Time Motion Estimation and Visualization on Graphics Cards," IEEE Visualization, 2004
- Harris, Baxter, Scheuermann, Lastra, "Simulation of Cloud Dynamics on Graphics Hardware," ACM/EG Graphics Hardware, 2003

References

- Krüger, Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," ACM SIGGRAPH, 2003
- Bolz, Farmer, Grinspun, Schröder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid," ACM SIGGRAPH, 2003
- Hillesland, Molinov, Grzeszczuk, "Nonlinear Optimization Framework for Image-Based Modeling on Programmable Graphics Hardware," ACM SIGGRAPH, 2003
- Harris, Coombe, Scheuermann, Lastra, "Physically-Based Visual Simulation on Graphics Hardware," ACM/EG Graphics Hardware, 2002
- Rumpf, Strzodka, "Using graphics cards for quantized FEM computations," IASTED Visualization, Imaging and Image Processing, 2001