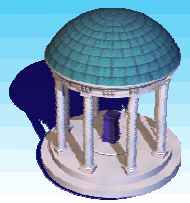
LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware

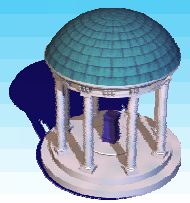
Nico Galoppo, Naga K. Govindaraju, Michael Henson, Dinesh Manocha

http://gamma.cs.unc.edu/LU-GPU



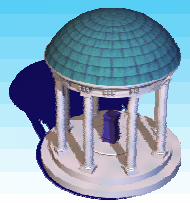
Goals

- Demonstrate advantages of mapping linear algebra routines to graphics hardware:
  - Performance
  - Growth rate
- LAPACK compliant set of linear algebra algorithms on graphics hardware



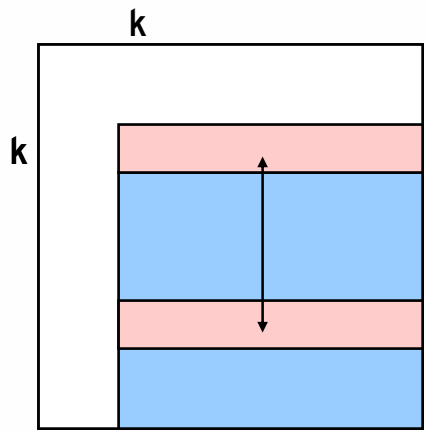
Outline

- LU Decomposition & Related Work
- The potential of GPUs
- LU-GPU algorithm
- Results
- Conclusions & Ongoing Work

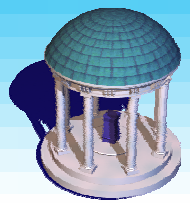


LU decomposition

- Sequence of row eliminations:
  - Scale and add: A(i,j) = A(i,j) – A(i,k) A(k,j)
  - Input data mapping: 2 distinct memory regions
  - No data dependencies within a row elimination

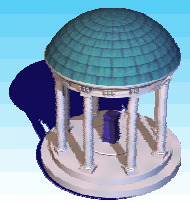


- Pivoting
  - Pointer-swap vs. data copy



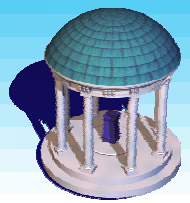
LU decomposition

- Theoretical complexity (partial pivoting):
  $(2/3)\ n^3 + O(n^2)$
- Performance ↔ Architecture
  - Order of operations
  - Memory access (latency)
  - Memory bandwidth



Outline

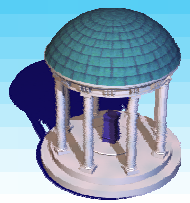
- LU Decomposition & Related Work
- The potential of GPUs
- LU-GPU algorithm
- Results
- Conclusions & Ongoing Work



Commodity CPUs

- LINPACK Benchmark:
  - Intel Pentium 4, 3.06 GHz: 2.88 GFLOPs/s

(Jack Dongarra, Oct 2005)

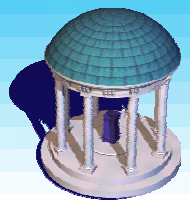


Streaming architectures

- Specialized hardware
- High bandwidth/compute ratio

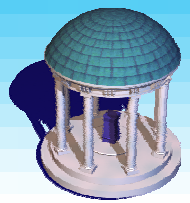
- Merrimac [Erez04]
  - Molecular modeling: 38 GFLOPs vs. 2.7 GFLOPs (P4)
  - $1,000/node
- Imagine [Ahn04]
  - 10.46 GFLOPs/s on QR-decomposition

- Research

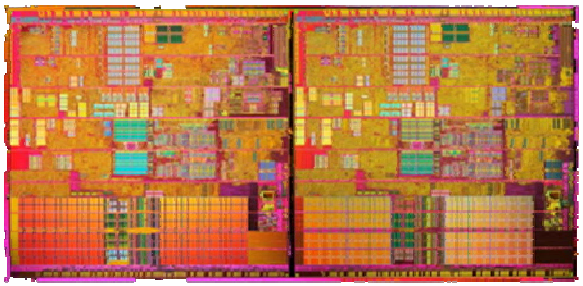


Outline

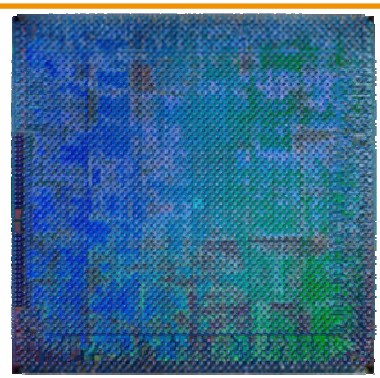
- LU Decomposition & Related Work
- The potential of GPUs
- LU-GPU algorithm
- Results
- Conclusions & Ongoing Work



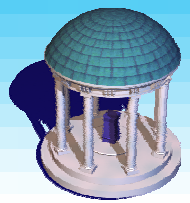
CPU vs. GPU



- Pentium EE 840
  - 3.2 GHz Dual Core
  - 230M Transistors
  - 90nm process
  - 206 mm^2
  - 2 x 1MB Cache
  - 25.6 GFLOPs
  - Price: $ 1,040



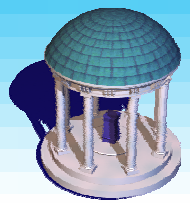
- GeForce 7800 GTX
  - 430 MHz
  - 302M Transistors
  - 110 nm process
  - 326 mm^2
  - 512MB onboard memory
  - 313 GFLOPs (shader)
  - 1.3 TFLOPs (total)
  - Price: $ 450



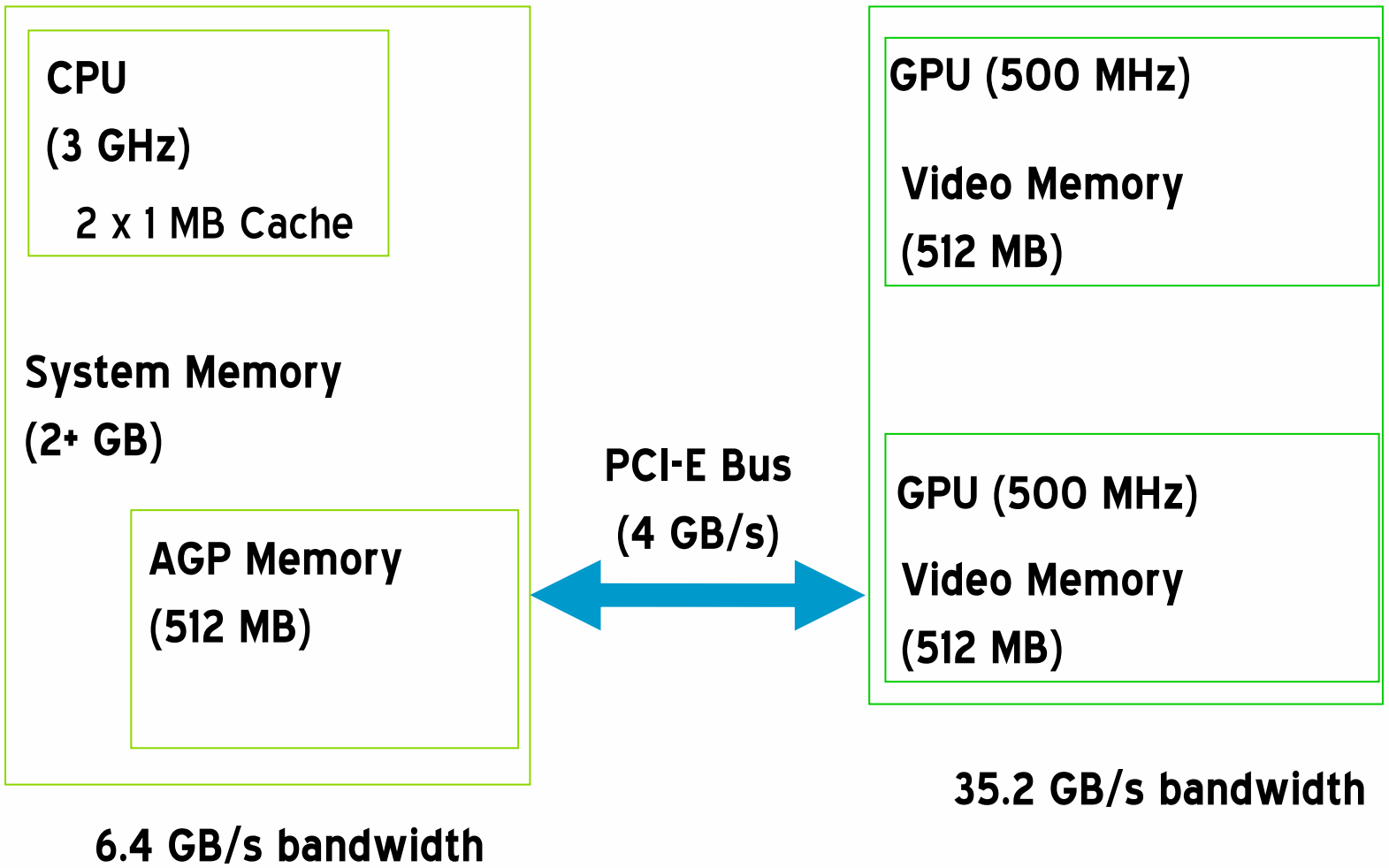
CPU vs. GPU

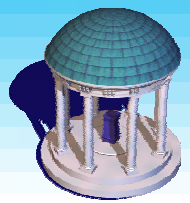
(Henry Moreton: NVIDIA, Aug. 2005)

	PEE 840	7800GTX	GPU/CPU
Graphics GFLOPs	25.6	1300	50.8
Shader GFLOPs	25.6	313	12.2
Die area (mm2)	206	326	1.6
Die area normalized	206	218	1.1
Transistors (M)	230	302	1.3
Power (W)	130	65	0.5
GFLOPS/mm	0.1	6.0	47.9
GFLOPS/tr	0.1	4.3	38.7
GFLOPS/W	0.2	20.0	101.6



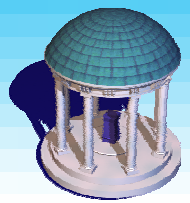
CPU vs. GPU: Bandwidth



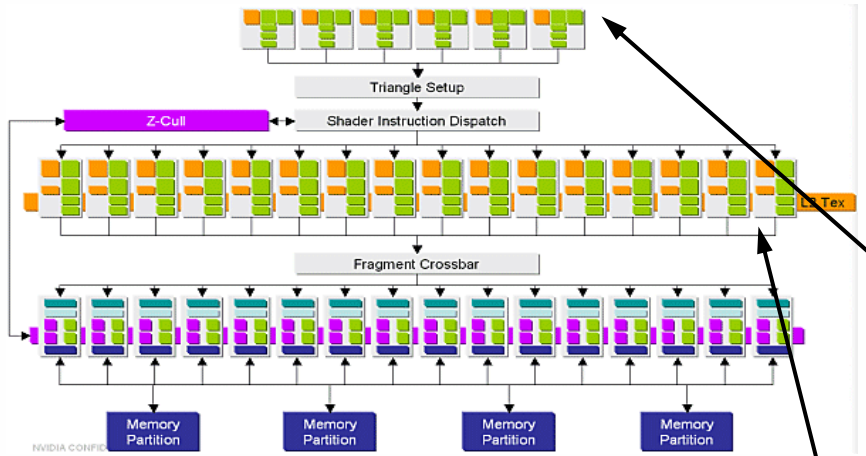


Bandwidth

- Large high bandwidth memory
  - 512 MB video memory vs. 2 MB L2 cache on CPUs
- High memory to compute clock ratio – reduces memory stalls

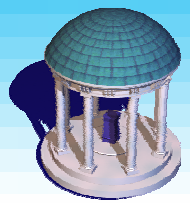


Graphics pipeline



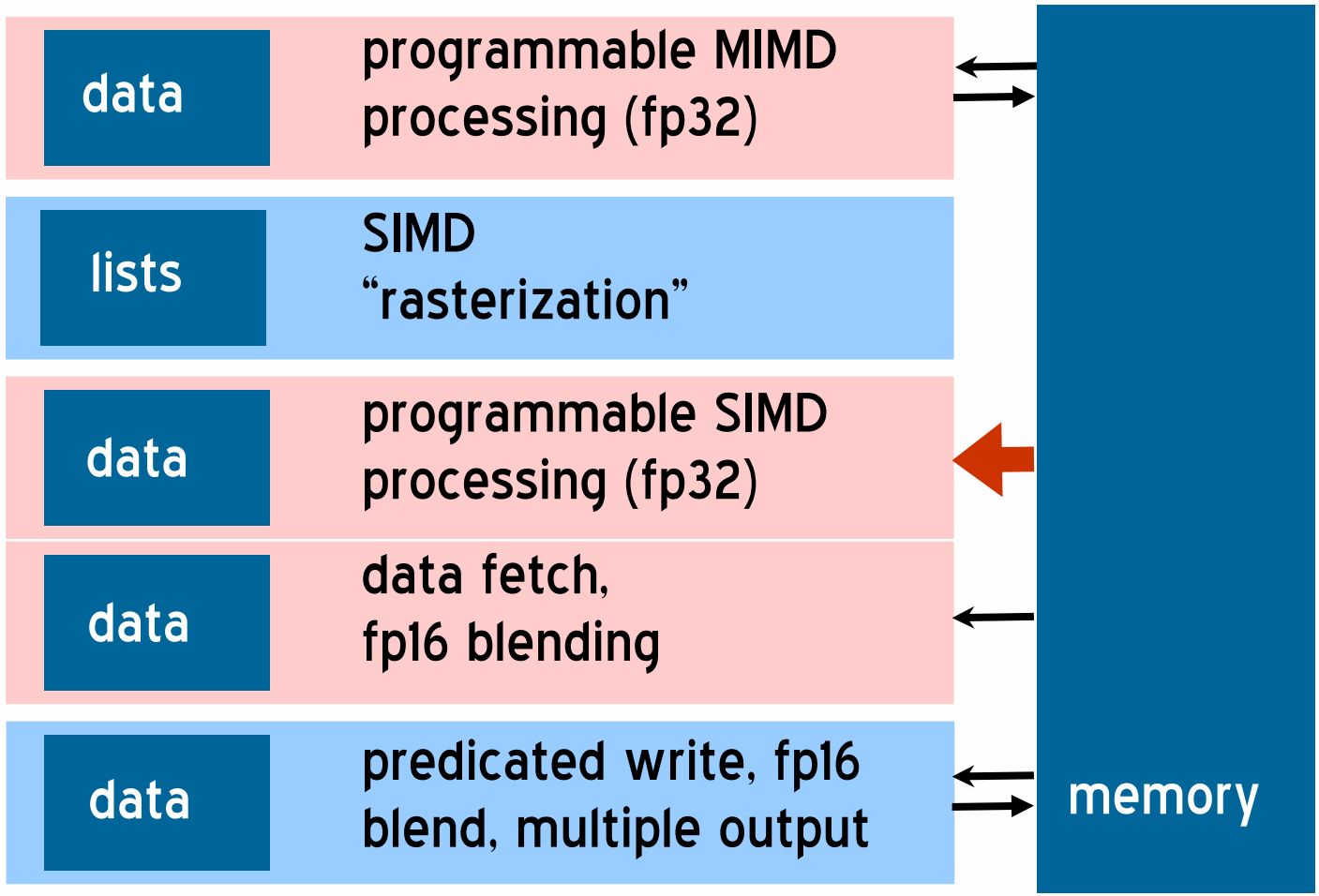
vertex	programmable vertex processing (fp32)
polygon	polygon setup, culling, rasterization
pixel	programmable per-pixel math (fp32)
texture	per-pixel texture, fp16 blending
image	Z-buf, fp16 blending, anti-alias (MRT)

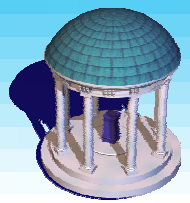
memory



Stream processor (non-graphics)

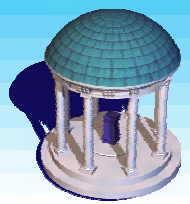
(David Kirk, NVIDIA, May'05)





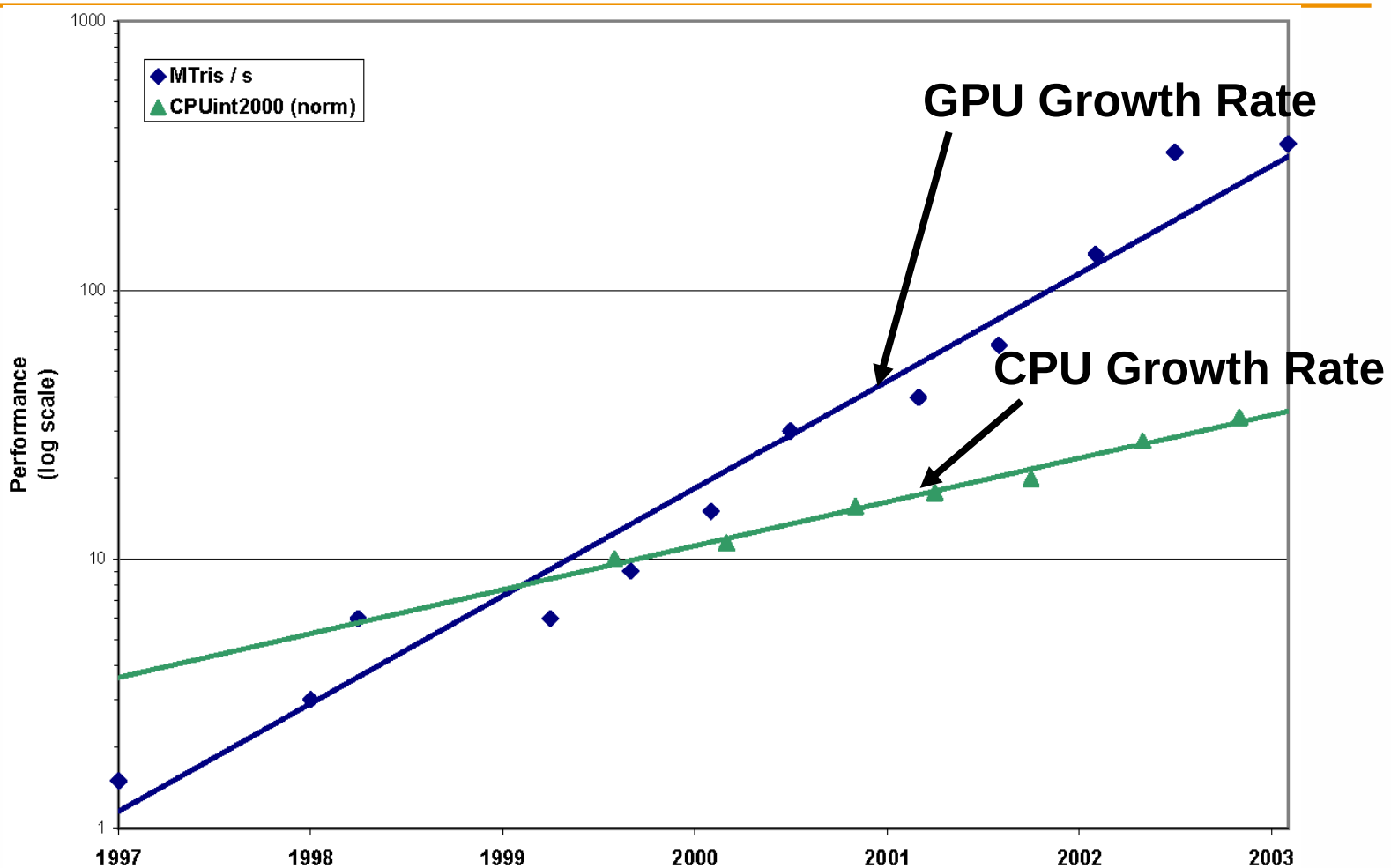
Potential of graphics processors

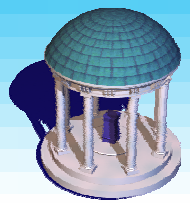
- Commodity horsepower
  - Parallel computation
  - Bandwidth
- Programmable graphics pipeline
  - Stream processor
- Exploit large growth rate



Exploiting technology moving faster than Moore's law

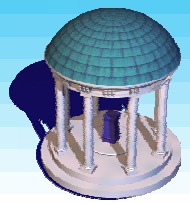
Source: Anselmo Lastra





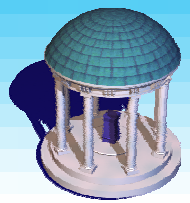
General purpose computing on GPUs

- Physical Simulation
  - Fluid Flow [Fan et al. 2004]
  - FEM [Rumpf and Strzodka 2001]
  - Cloud Dynamics [Harris et al. 2003]
- Sparse Linear Algebra
  - Operators [Krüger and Westermann 2003]
  - Iterative Solvers [Bolz et al. 2003]



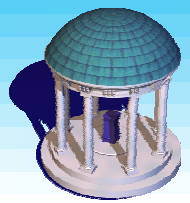
General purpose computing on GPUs

- Matrix-Matrix Multiplication
  - Fixed graphics pipeline, fixed-point arithmetic [Larsen and McAllister 2001]
  - Floating-point (SP) [Fatahalian et al. 2004]
- High-level API
  - BrookGPU [Buck et al. 2004]
  - Sh [McCool et al. 2004]



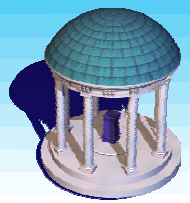
Outline

- LU Decomposition & Related Work
- The potential of GPUs
- LU-GPU algorithm
- Results
- Conclusions & Ongoing Work



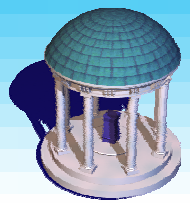
Motivation for LU-GPU

- LU decomposition maps well:
  - Stream program
  - Few data dependencies
- Pivoting
  - Parallel pivot search
  - Exploit large memory bandwidth

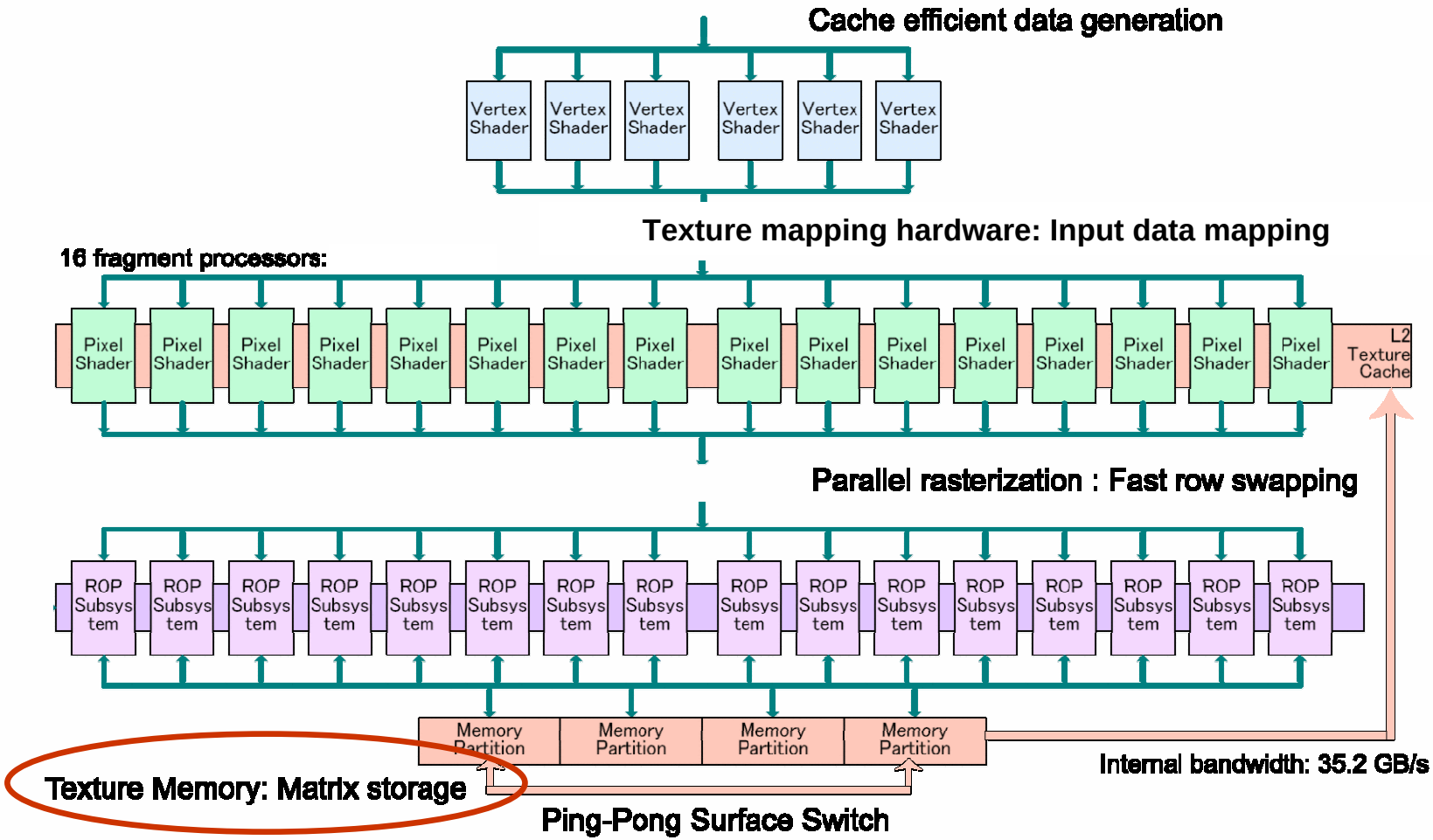


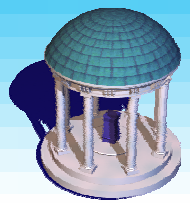
GPU based algorithms

- Data representation
- Algorithm mapping



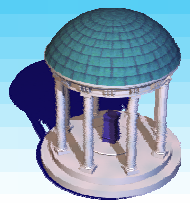
Data representation





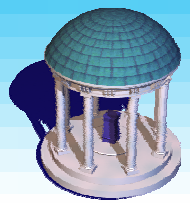
Data representation

- Matrix elements
  - 2D texture memory
  - One-to-one mapping
- Texture memory = on-board memory
  - Exploit bandwidth
  - Avoid CPU-GPU data transfer

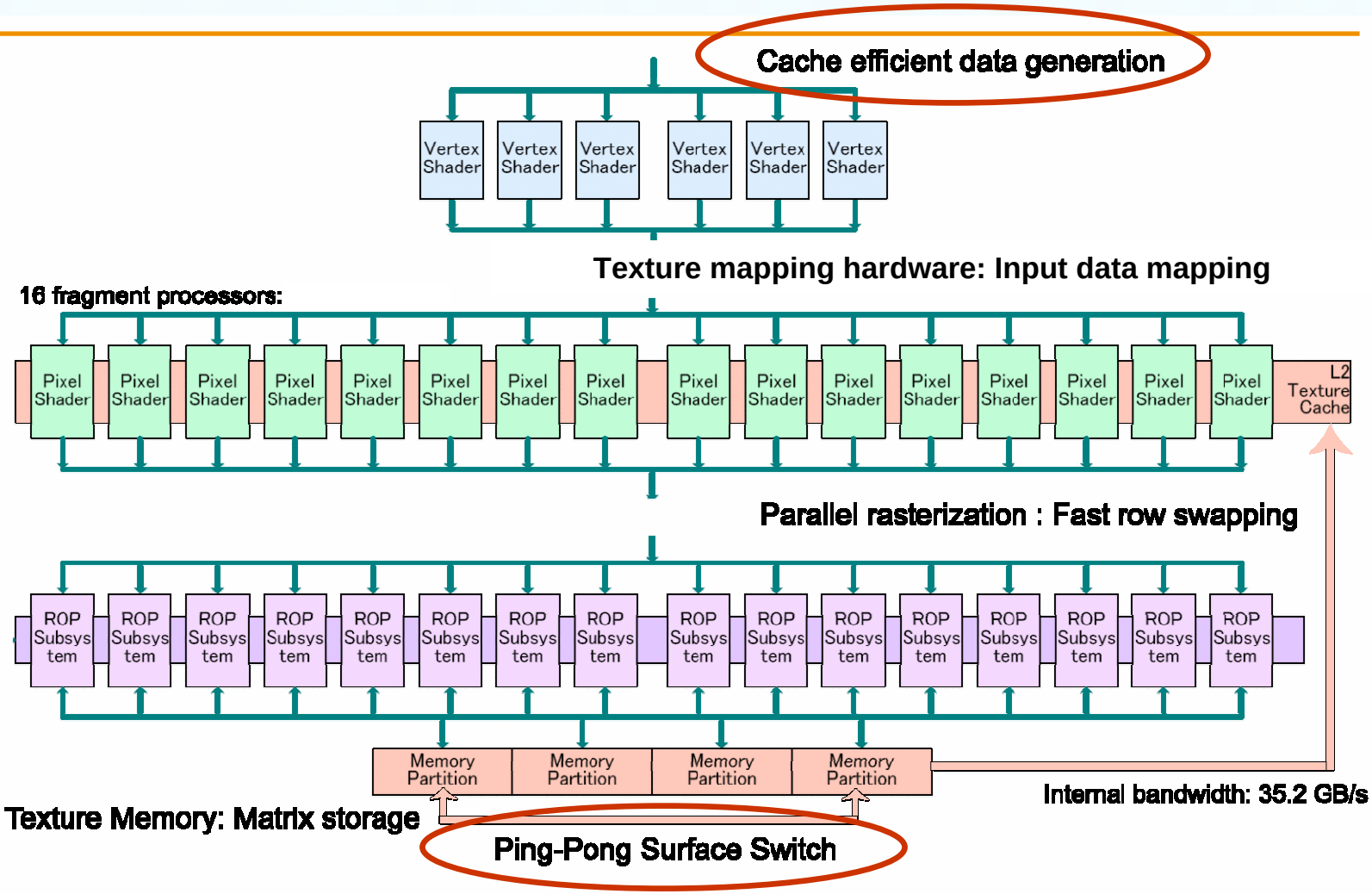


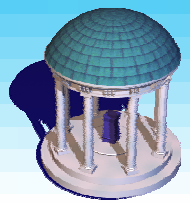
GPU based algorithms

- Data representation
- Algorithm mapping
  - Stream computation
  - Input data mapping
  - Fast row swaps



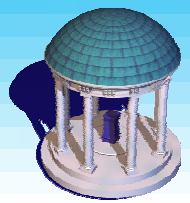
Algorithm mapping



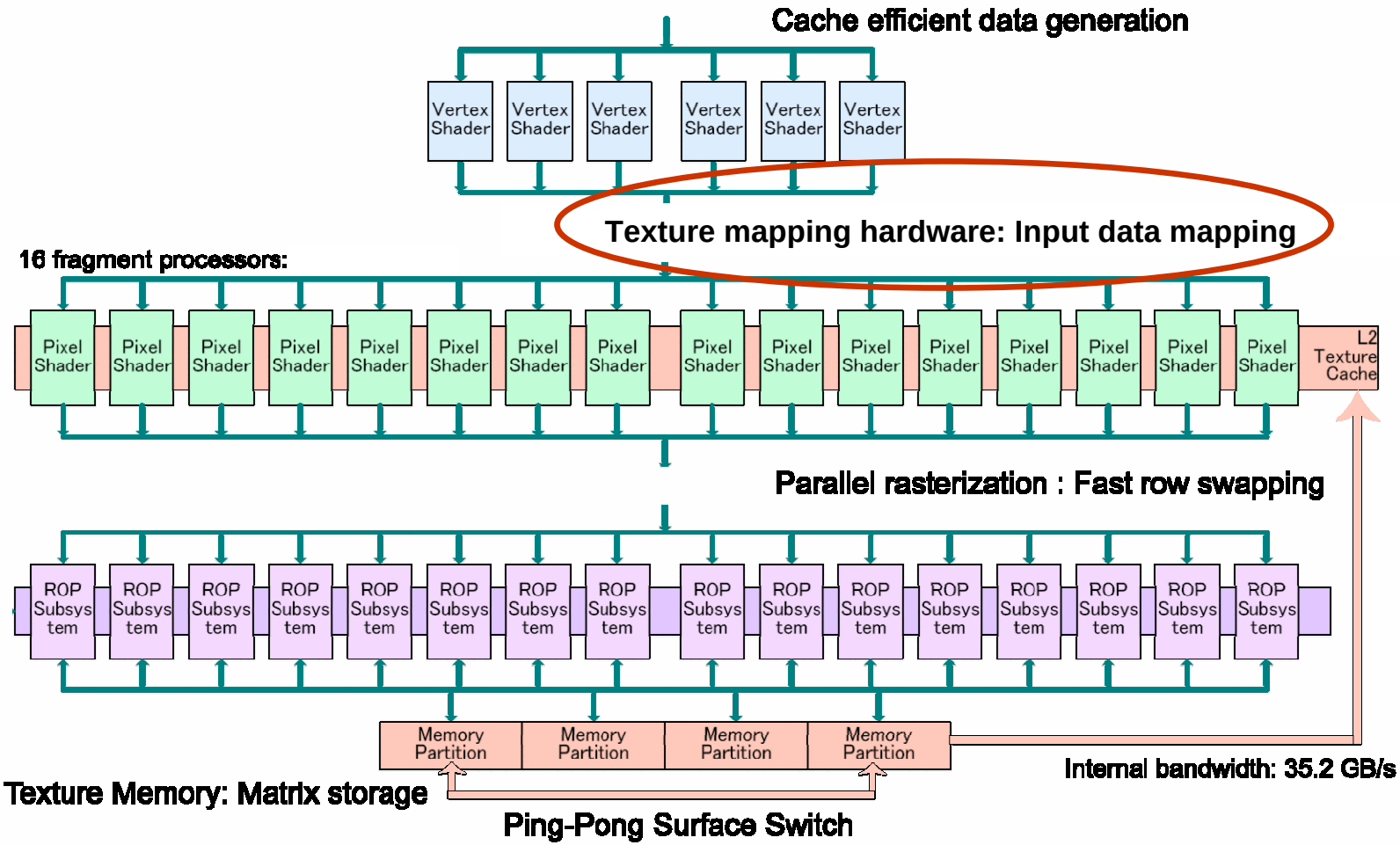


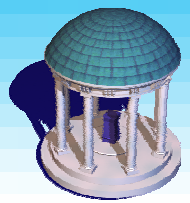
Stream computation

- Rasterize quadrilaterals
  - Generates computation stream
  - Invokes SIMD units
  - Rasterization simulates blocking
- Rasterization pass = row elimination
- Alternating memory regions



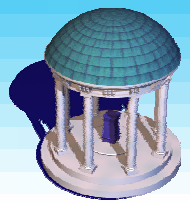
Input data mapping





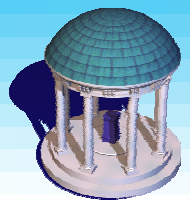
Input data mapping

- Dedicated **texture mapping hardware**
  - Traditionally for color interpolation
  - **Map** input matrix elements to output elements
  - Eliminates computation of memory locations
- 25% performance improvement

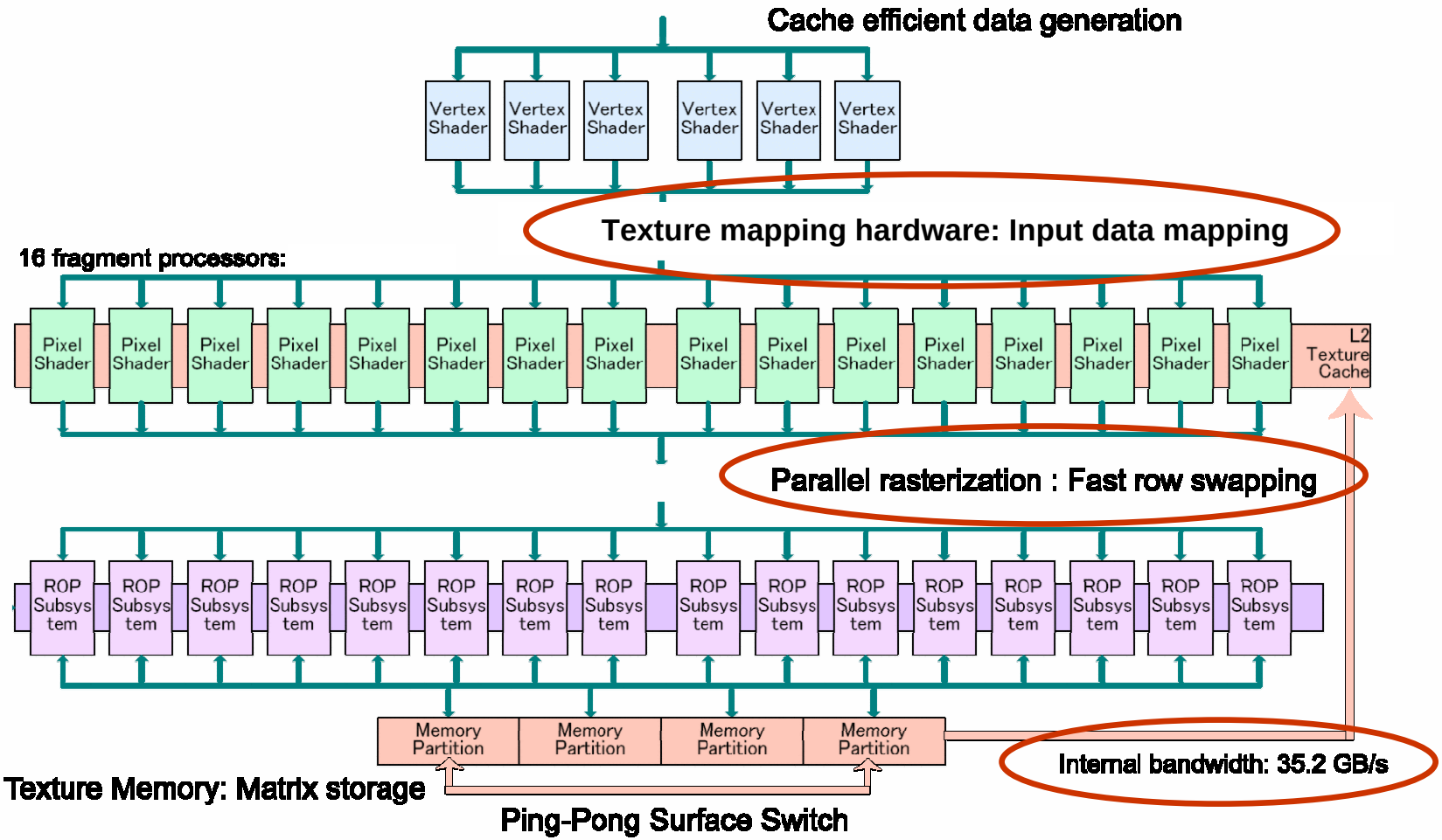


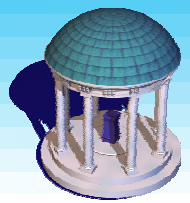
Pivoting

- Main issues:
  - Pivot search
  - Row/column swapping



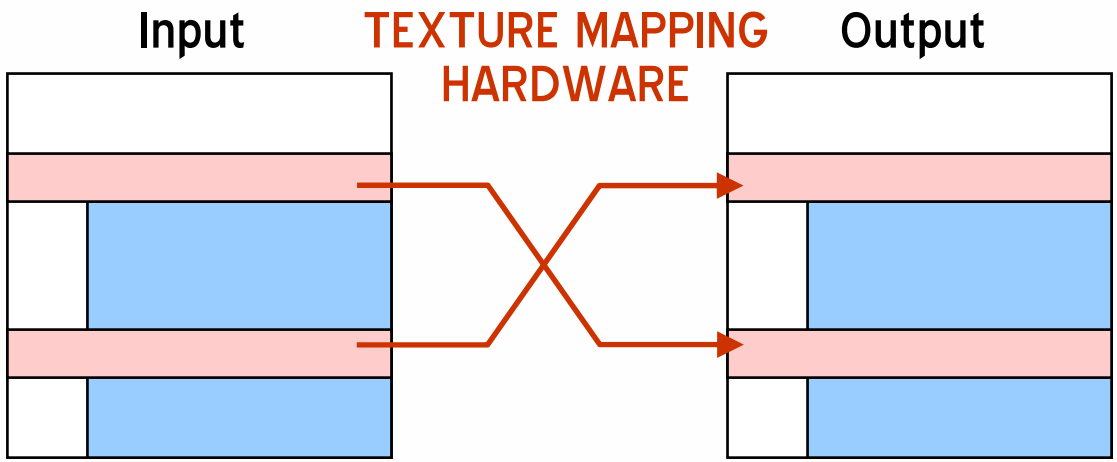
Pivoting

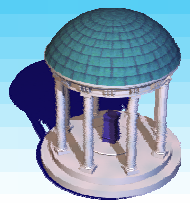

Cache efficient data generation
Vertex Shader
Vertex Shader
Vertex Shader
Vertex Shader
Vertex Shader
Vertex Shader
Texture mapping hardware: Input data mapping
16 fragment processors:
Pixel Shader
L2 Texture Cache
Parallel rasterization : Fast row swapping
ROP Subsystem
Memory Partition
Internal bandwidth: 35.2 GB/s
Texture Memory: Matrix storage
Ping-Pong Surface Switch



Partial pivoting

- Fast row swap
  - Data copy: mapped rasterization
    - Texture mapping hardware
    - High memory bandwidth
  - Improvement over pointer swapping

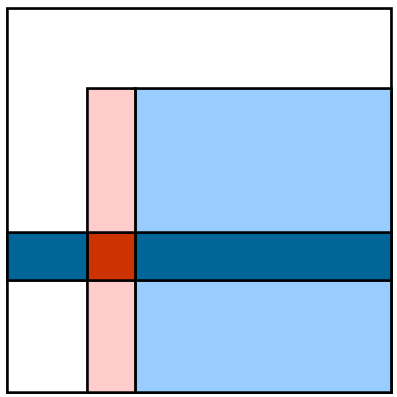




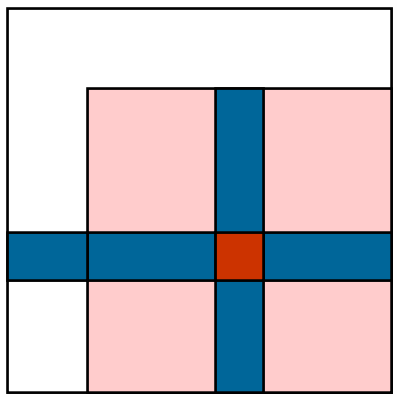
Full pivoting

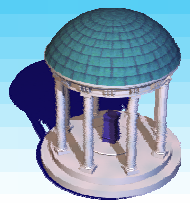
- Fast column/row swap
- Parallel pivot search
  - Divide and conquer approach

Partial pivoting



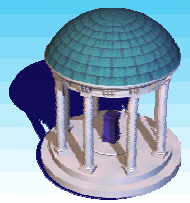
Full pivoting





Outline

- LU Decomposition & Related Work
- The potential of GPUs
- LU-GPU algorithm
- Results
- Conclusions & Ongoing Work



Benchmarks

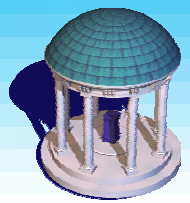
Commodity CPU

3.4 GHz Pentium IV with Hyper-Threading, 1 MB L2 cache

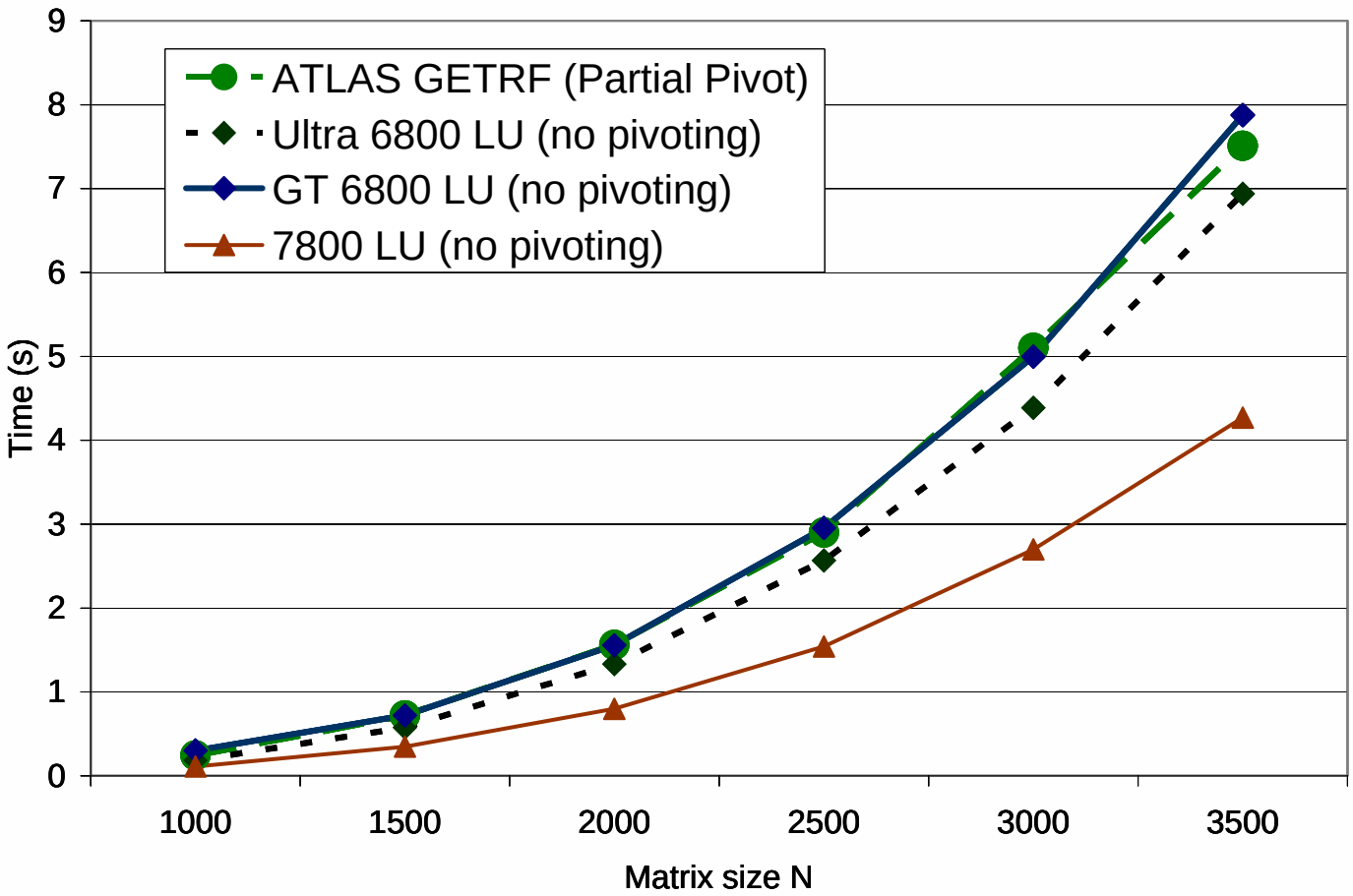
LAPACK sgetrf() (blocked algorithm, ATLAS library)

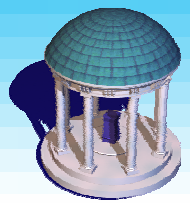
LAPACK sgetc2() (SSE-optimized IMKL library)

GPU	SIMD units	Core clock	Memory	Memory clock
6800 GT	12	350 MHz	256 Mb	900 MHz
6800 Ultra	16	425 MHz	256 Mb	1100 MHz
7800 GTX	24	430 MHz	256 Mb	1200 MHz

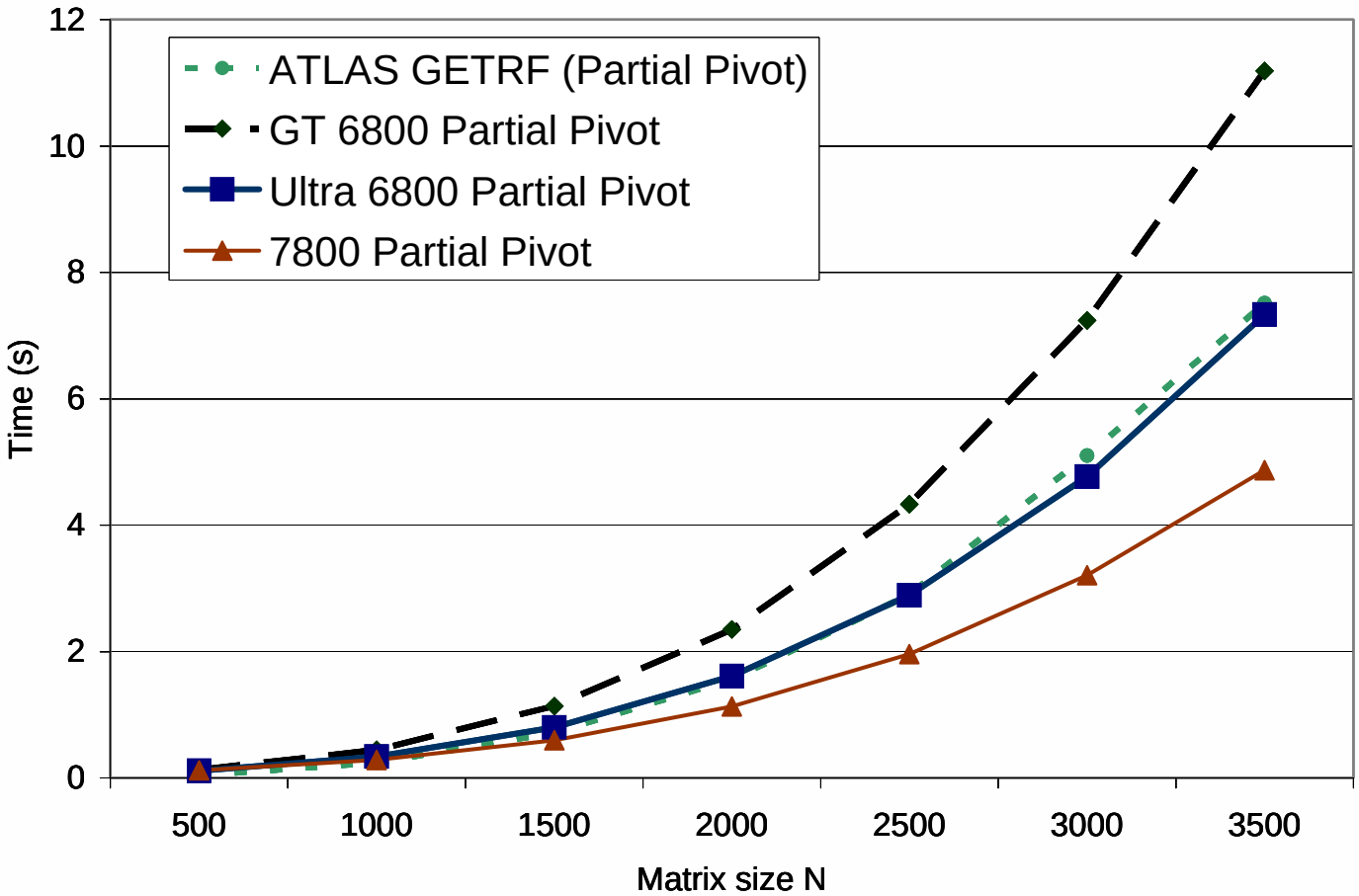


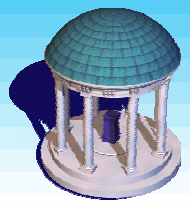
Results: No pivoting



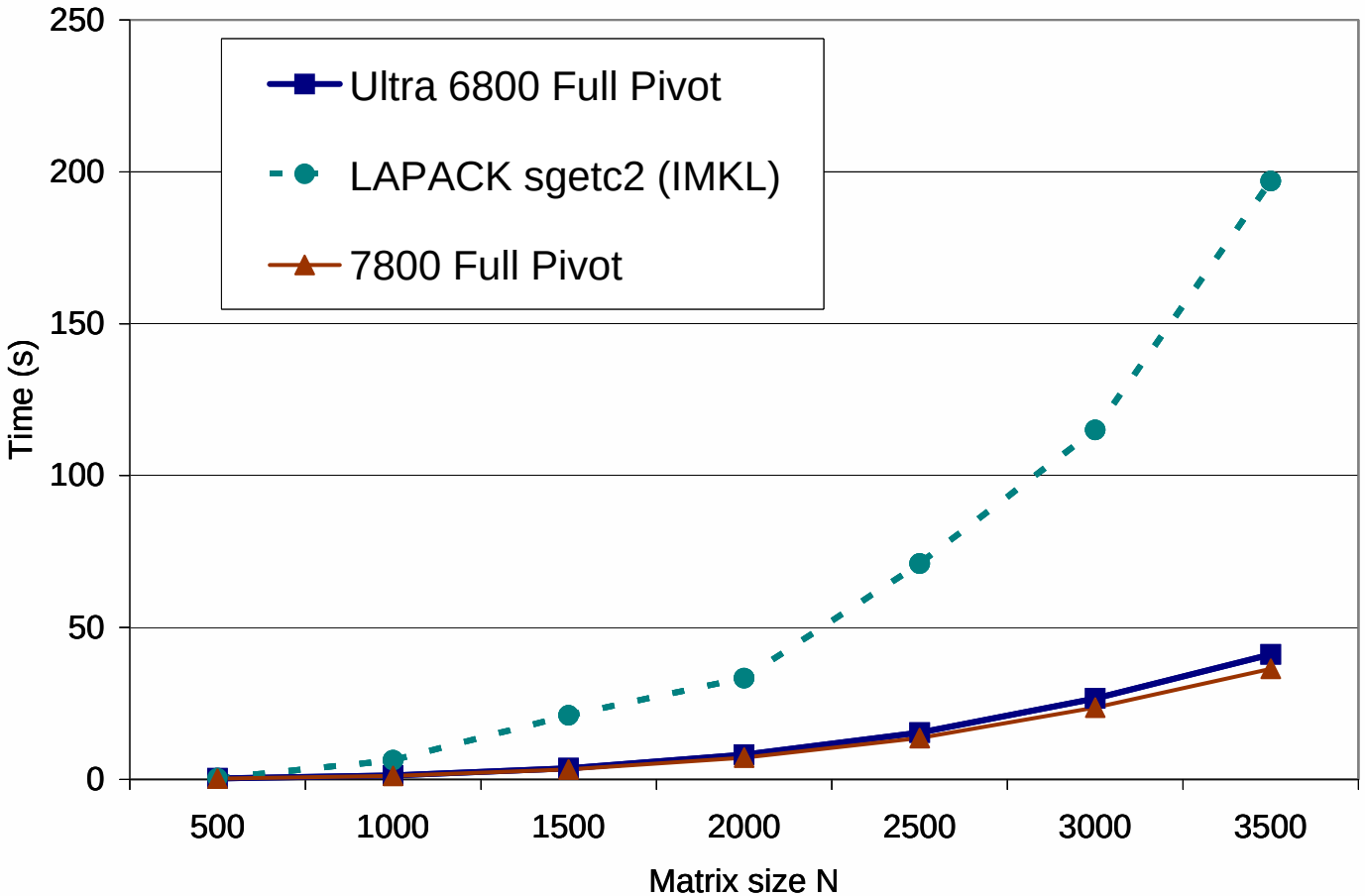


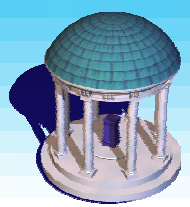
Results: Partial pivoting



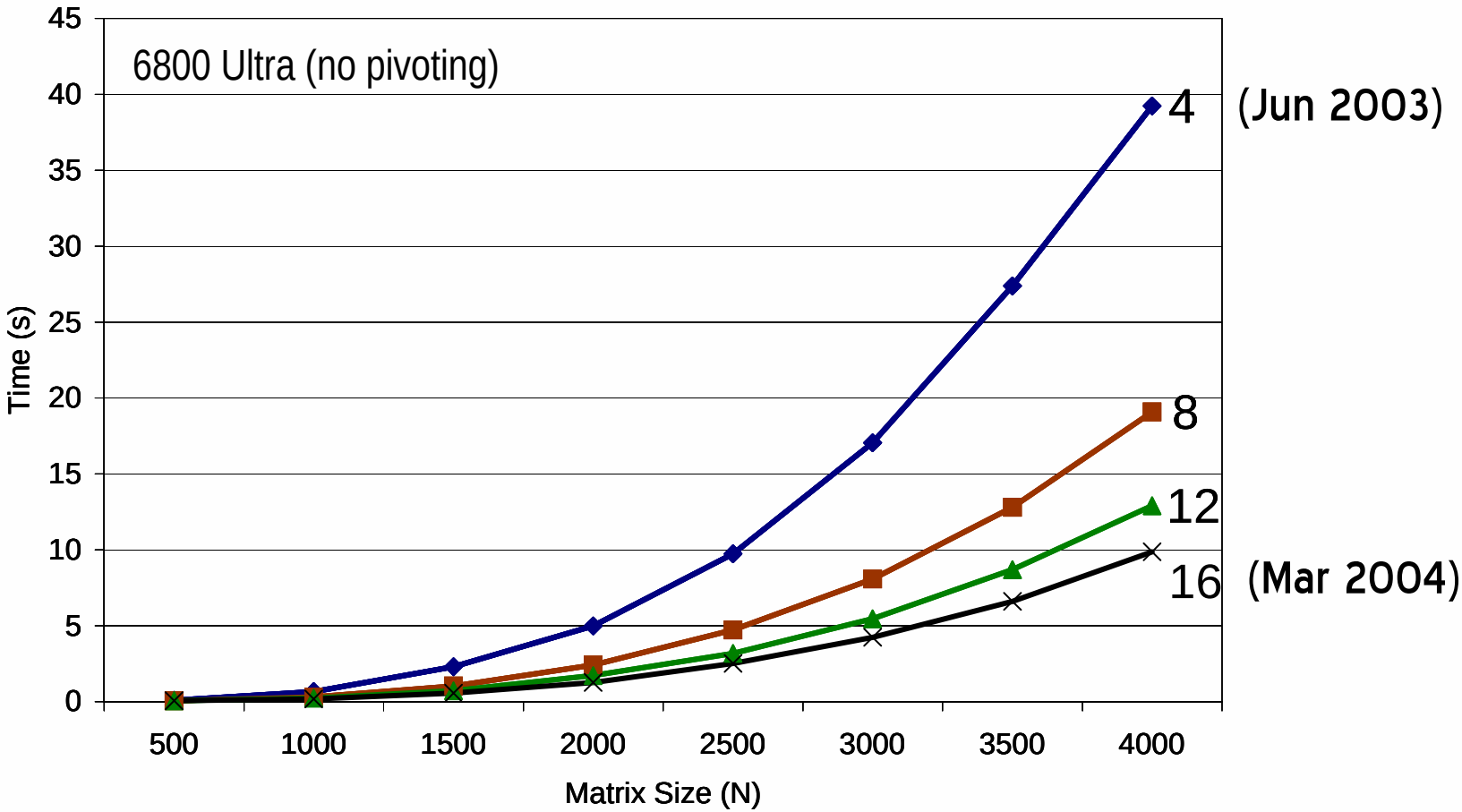


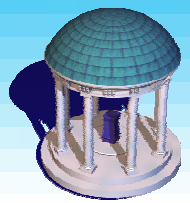
Results: Full Pivoting



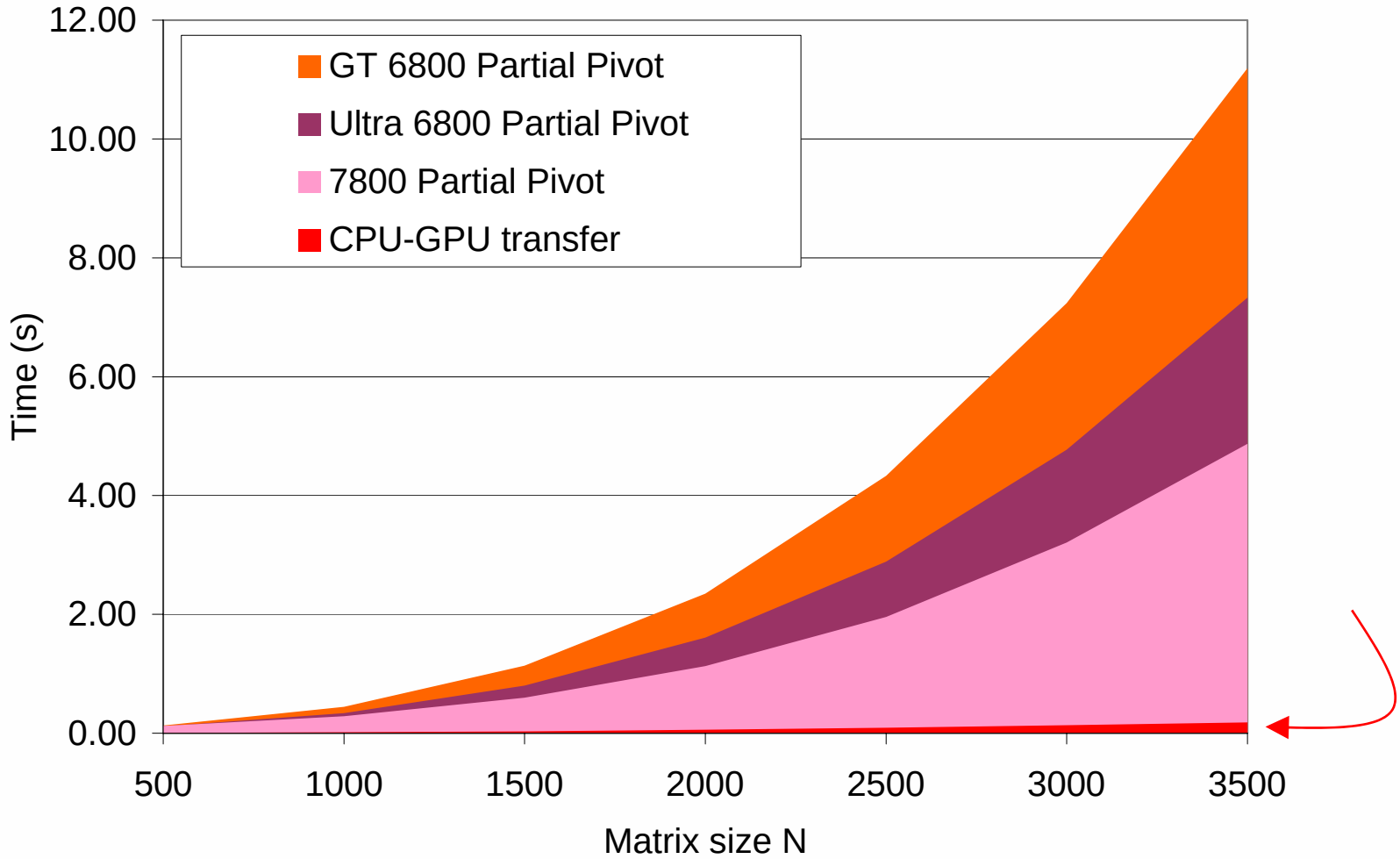


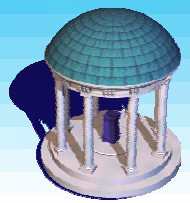
Results: Number of computational units



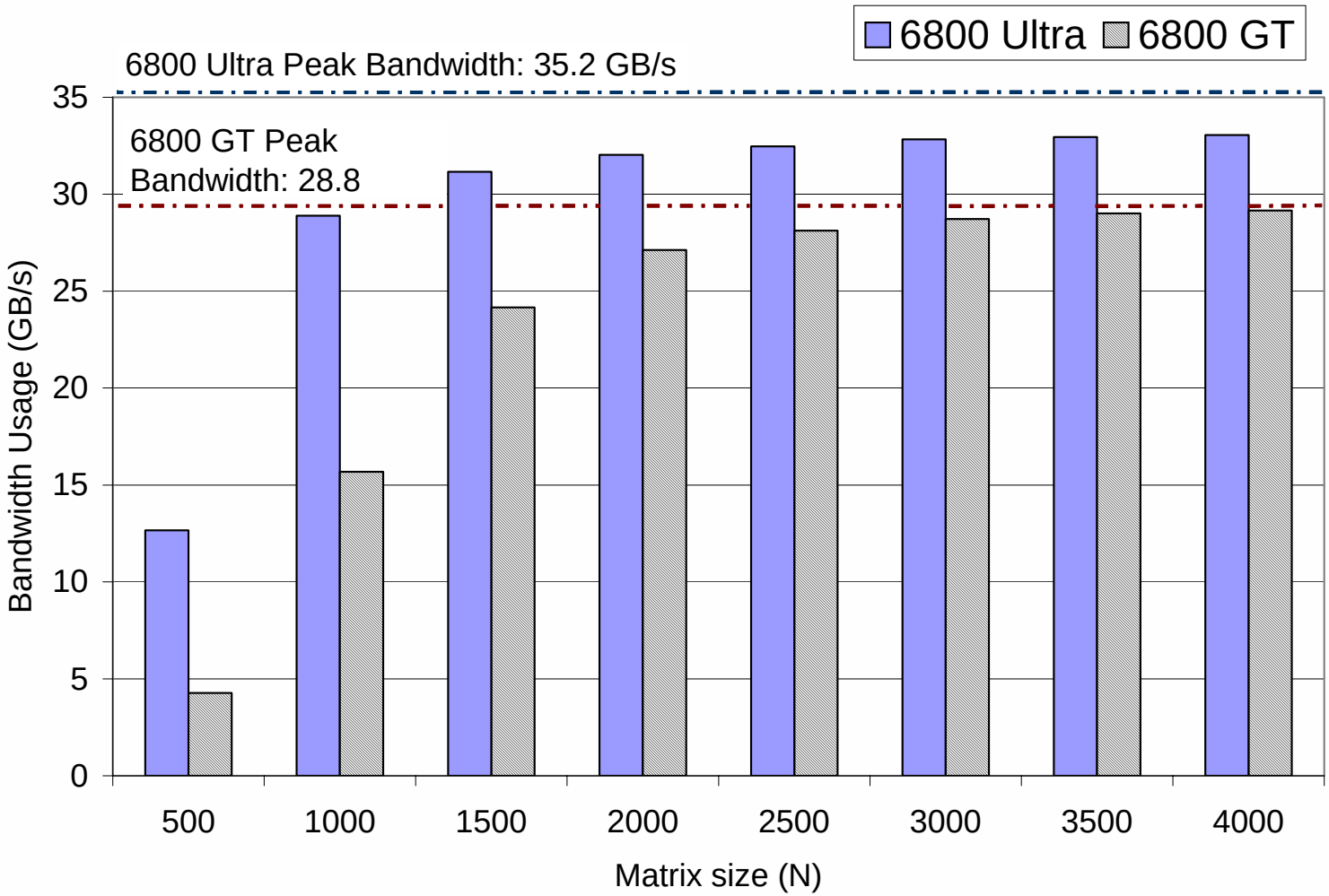


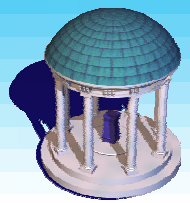
GPU-CPU data transfer overhead



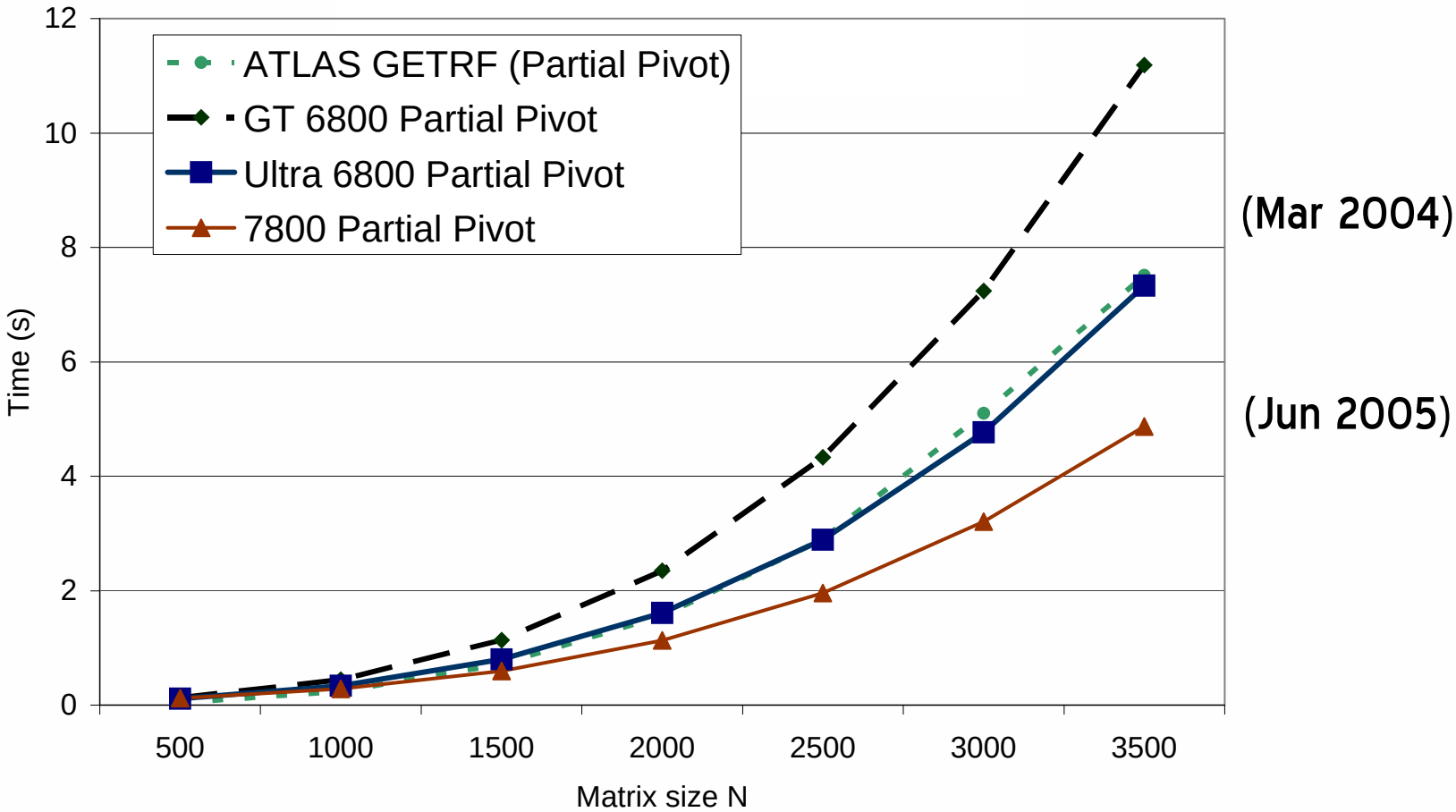


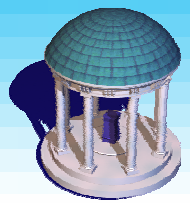
Bandwidth efficiency





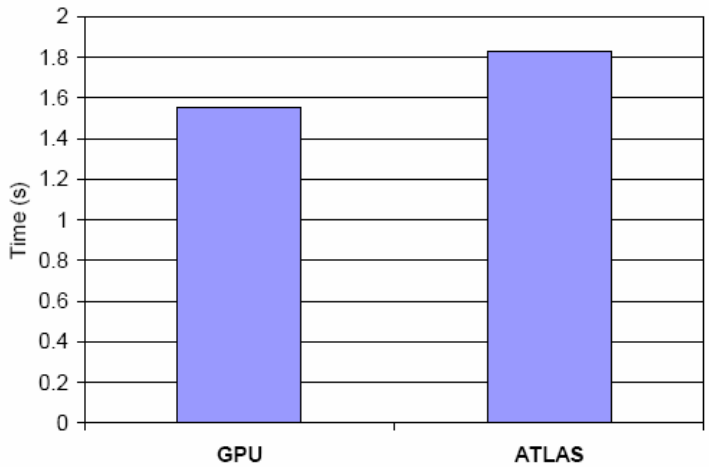
Faster than Moore's law

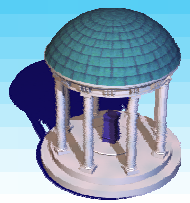




Application: Fluid Simulation

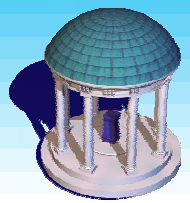
- Solve parallel sub-problems
  - N=2048
  - Diagonally-dominant
  - No pivoting required
- 15% faster than ATLAS on Pentium IV 3.06 GHz





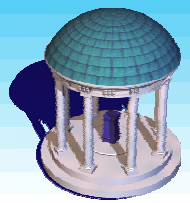
Limitations

- Maximum matrix size: 4096x4096
  - Block-partitioned LU decomposition
- Precision
  - Single precision floating point
  - Not 100% IEEE floating point compliant
- CPU-GPU data transfer overhead
  - Small matrices



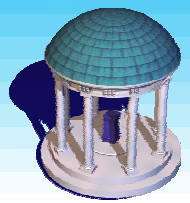
Graphics hardware advancements

- Improved floating point bandwidth
  - 4 component vs. single component
- Floating point blending
  - Use of non-programmable TFLOPs



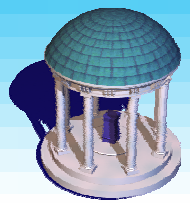
Outline

- LU Decomposition & Related Work
- The potential of GPUs
- LU-GPU algorithm
- Results
- Conclusions & Ongoing Work



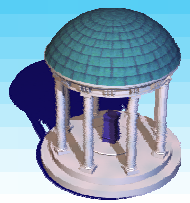
Conclusions

- Algorithm mapped to graphics pipeline
  - Novel mapping of row operations to rasterization
  - Stream computation
  - Blocking



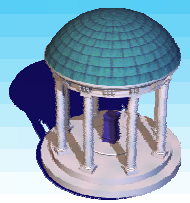
Conclusions

- Optimized with GPU architecture
  - Input data mapping
  - Fast pivoting



Conclusions

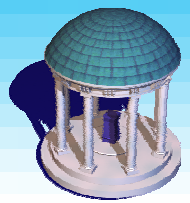
- Performance
  - Comparable to industry-standard libraries
  - Relatively small development effort
- GPU are useful co-processors
  - Scientific computations
  - Many applications



Conclusions

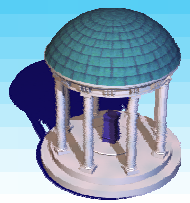
- LU-GPU Open Source library available:

  http://gamma.cs.unc.edu/LUGPULIB/



Ongoing work

- Sorting on GPUs
- Linear algebra:
  - GPU-LAPACK / QR decomposition



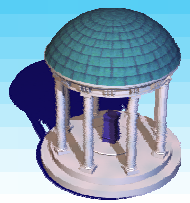
Sorting on GPUs

- Goal: Utilize the high parallelism, and memory bandwidth on GPUs for fast sorting

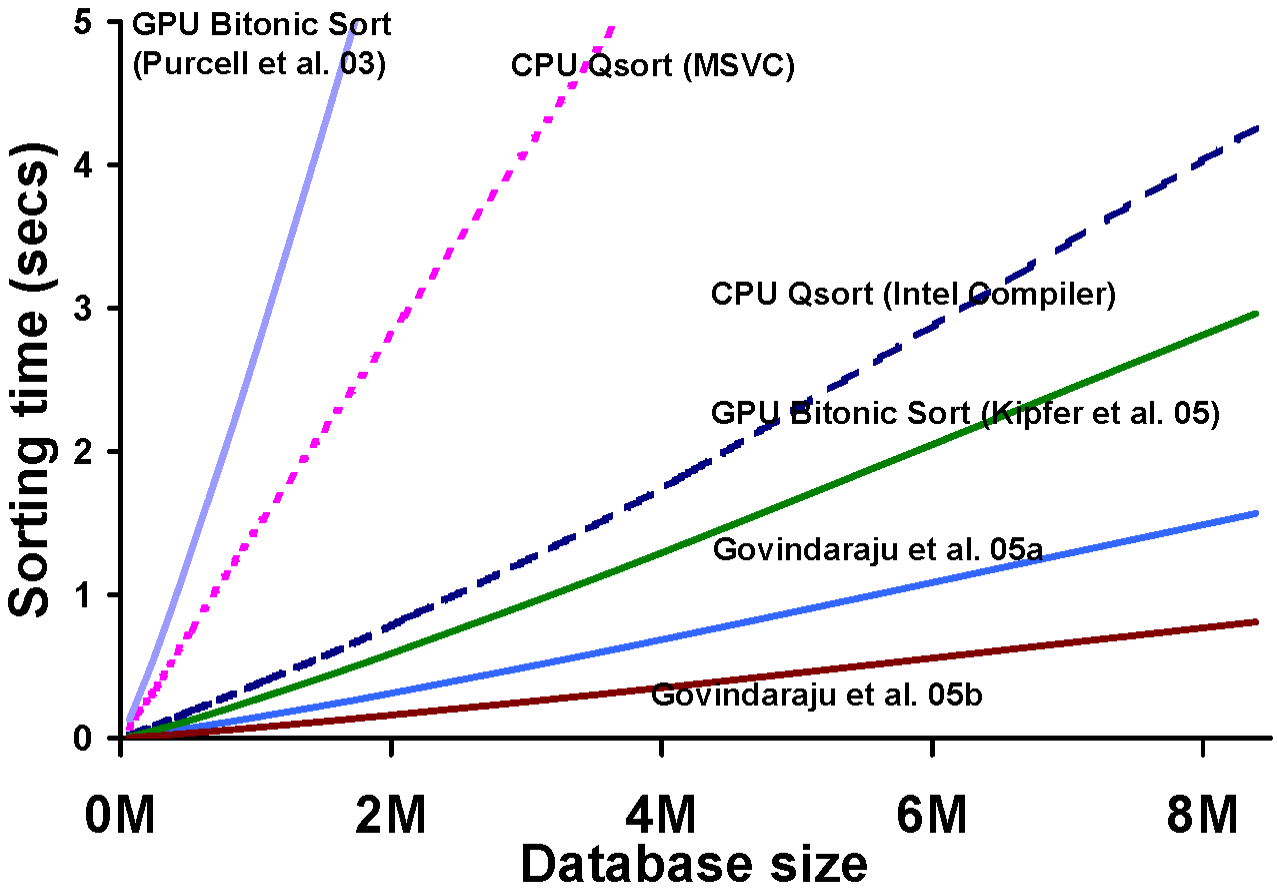
[Govindaraju et al, SIGMOD05]

- GPUSort: Open Source library

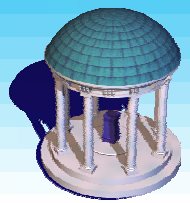
[http://gamma.cs.unc.edu/GPUSORT/]



Sorting on GPUs

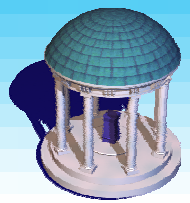


6 times faster than Quicksort on 3.4 GHz Pentium IV PC!



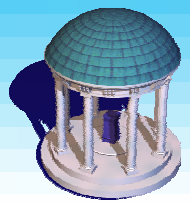
Linear algebra

- LAPACK-compliant library for GPUs
- QR-decomposition in development (LAPACK SGEQRF)



Acknowledgements

- Army Research Office
- DARPA
- National Science Foundation
- Office of Naval Research
- RDECOM
- Intel Corporation
- NVIDIA Corporation
- UNC GAMMA Group



Thank you

- For questions or comments:

nico@cs.unc.edu
naga@cs.unc.edu
henson@cs.unc.edu

http://gamma.cs.unc.edu/
http://gamma.cs.unc.edu/LUGPULIB/