# Advanced CUDA: Overview of GPU Hardware

John E. Stone

Theoretical and Computational Biophysics Group

Beckman Institute for Advanced Science and Technology

University of Illinois at Urbana-Champaign

**http://www.ks.uiuc.edu/Research/gpu/**

GPGPU2: Advanced Methods for Computing with CUDA,

University of Cape Town, April 2014

# GPU Computing

- Commodity devices, omnipresent in modern computers (over a **million** sold per **week**)
- Massively parallel hardware, hundreds of processing units, **throughput oriented architecture**
- Standard integer and floating point types supported
- Programming tools allow software to be written in dialects of familiar C/C++ and integrated into legacy software
- GPU algorithms are often multicore friendly due to attention paid to **data locality** and **data-parallel work decomposition**

# Benefits of GPUs vs. Other Parallel Computing Approaches

- Increased compute power per unit volume
- Increased FLOPS/watt power efficiency
- Desktop/laptop computers easily incorporate GPUs, no need to teach non-technical users how to use a remote cluster or supercomputer
- GPU can be upgraded without new OS license fees, low cost hardware

# What Runs on a GPU?

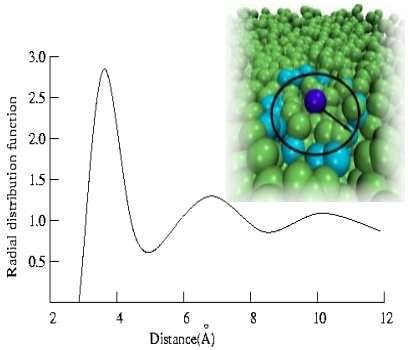
- GPUs run data-parallel programs called **“kernels”**
- GPUs are managed by a host CPU thread:
  - Create a CUDA context
  - Allocate/deallocate GPU memory
  - Copy data between host and GPU memory
  - Launch GPU kernels
  - Query GPU status
  - Handle runtime errors

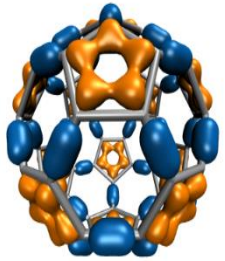
# What Speedups Can GPUs Achieve?

- Single-GPU speedups of **3x** to **10x** vs. one multi-core CPU are very common
- Best speedups can reach **25x** or more, attained on codes dominated by floating point arithmetic, especially native GPU machine instructions, e.g. expf(), rsqrtf(), …
- **Amdahl's Law** can prevent legacy codes from achieving peak speedups with shallow GPU acceleration efforts

# CUDA GPU-Accelerated Trajectory Analysis and Visualization in VMD

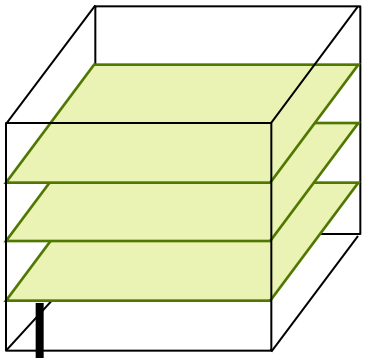
| VMD GPU-Accelerated Feature or Kernel | Typical speedup vs. multi-core CPU (e.g. 4-core CPU) |
|---|---|
| **Molecular orbital display** | **30x** |
| **Radial distribution function** | **23x** |
| **Molecular surface display** | **15x** |
| **Electrostatic field calculation** | **11x** |
| **Ray tracing w/ shadows, AO lighting** | **7x** |
| Ion placement | 6x |
| MDFF density map synthesis | 6x |
| Implicit ligand sampling | 6x |
| Root mean squared fluctuation | 6x |
| Radius of gyration | 5x |
| Close contact determination | 5x |
| Dipole moment calculation | 4x |



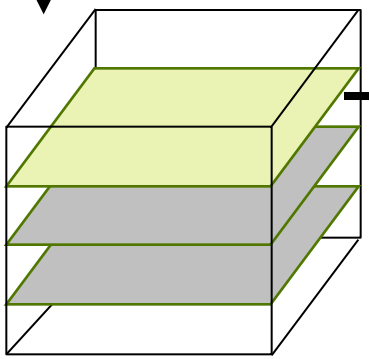



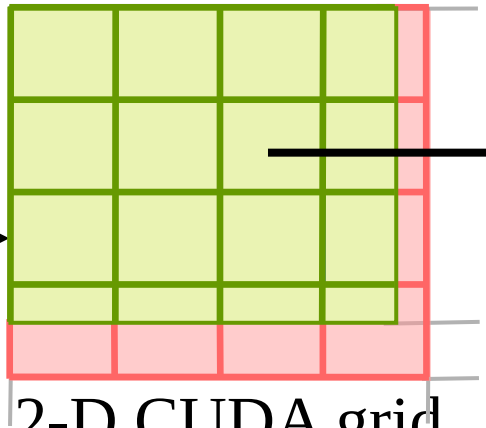
# GPU Solution: Computing $C_{60}$ Molecular Orbitals

3-D orbital lattice: millions of points

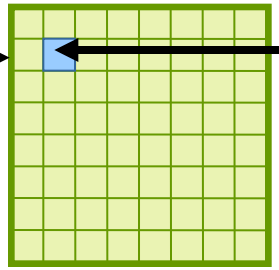


Lattice slices computed on multiple GPUs





2-D CUDA grid on one GPU

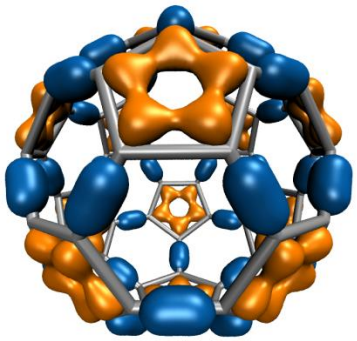


CUDA thread blocks

GPU threads each compute one point.

| Device | CPUs, GPUs | Runtime (s) | Speedup |
|---|---|---|---|
| Intel X5550-SSE | 1 | 30.64 | 0.14 |
| Intel X5550-SSE | 8 | 4.13 | 1.0 |
| GeForce GTX 480 | 1 | 0.255 | 16 |
| GeForce GTX 480 | 4 | 0.081 | 51 |

# Molecular Orbital Inner Loop, Hand-Coded x86 SSE
# Hard to Read, Isn't It? (And this is the "pretty" version!)

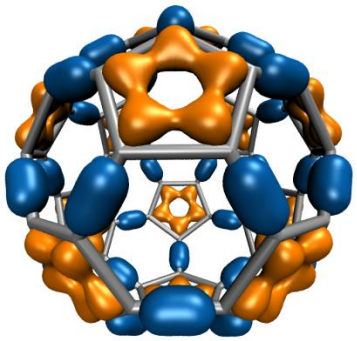


```
for (shell=0; shell < maxshell; shell++) {
  __m128 Cgto = _mm_setzero_ps();
  for (prim=0; prim<num_prim_per_shell[shell_counter]; prim++) {
    float exponent       = -basis_array[prim_counter    ];
    float contract_coeff =  basis_array[prim_counter + 1];
    __m128 expval = _mm_mul_ps(_mm_load_ps1(&exponent), dist2);
    __m128 ctmp = _mm_mul_ps(_mm_load_ps1(&contract_coeff), exp_ps(expval));
    Cgto = _mm_add_ps(contracted_gto, ctmp);
    prim_counter += 2;
  }
  __m128 tshell = _mm_setzero_ps();
  switch (shell_types[shell_counter]) {
    case S_SHELL:
      value = _mm_add_ps(value, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), Cgto));   break;
    case P_SHELL:
      tshell = _mm_add_ps(tshell, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), xdist));
      tshell = _mm_add_ps(tshell, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), ydist));
      tshell = _mm_add_ps(tshell, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), zdist));
      value = _mm_add_ps(value, _mm_mul_ps(tshell, Cgto));      break;
```

Writing SSE kernels for CPUs requires assembly language, compiler intrinsics, various libraries, or a really smart autovectorizing compiler **and lots of luck...**

# Molecular Orbital Inner Loop in CUDA

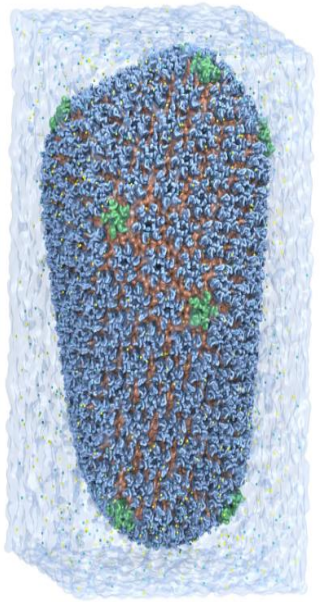
```
for (shell=0; shell < maxshell; shell++) {
    float contracted_gto = 0.0f;
    for (prim=0; prim<num_prim_per_shell[shell_counter]; prim++) {
      float exponent          = const_basis_array[prim_counter     ];
      float contract_coeff = const_basis_array[prim_counter + 1];
      contracted_gto += contract_coeff * exp2f(-exponent*dist2);
      prim_counter += 2;
    }
    float tmpshell=0;
    switch (const_shell_symmetry[shell_counter]) {
      case S_SHELL:
        value += const_wave_f[ifunc++] * contracted_gto;    break;
      case P_SHELL:
        tmpshell += const_wave_f[ifunc++] * xdist;
        tmpshell += const_wave_f[ifunc++] * ydist
        tmpshell += const_wave_f[ifunc++] * zdist;
        value += tmpshell * contracted_gto;   break;
```



Aaaaahhhh….

Data-parallel CUDA kernel looks like normal C code for the most part….

# HIV-1 Parallel HD Movie Rendering on Blue Waters Cray XE6/XK7



New “TachyonL-OptiX” on XK7 vs. Tachyon on XE6:
K20X GPUs yield **up to eight times** geom+ray tracing speedup

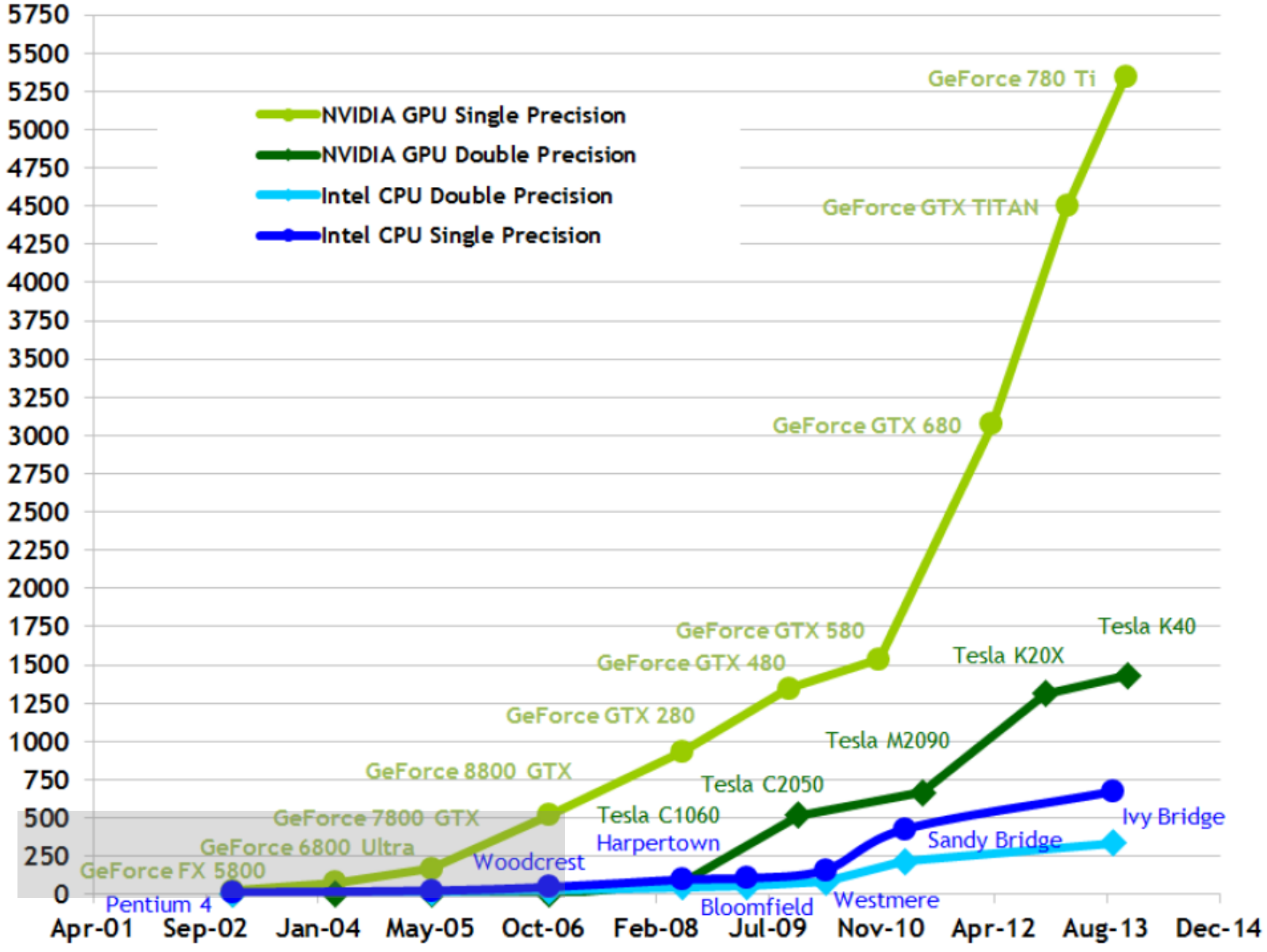
Cray XE6: 2x Opteron 62xx CPUs (32-cores)

Cray XK7: 1x Opteron 62xx CPU (16-cores) + NVIDIA Tesla K20X

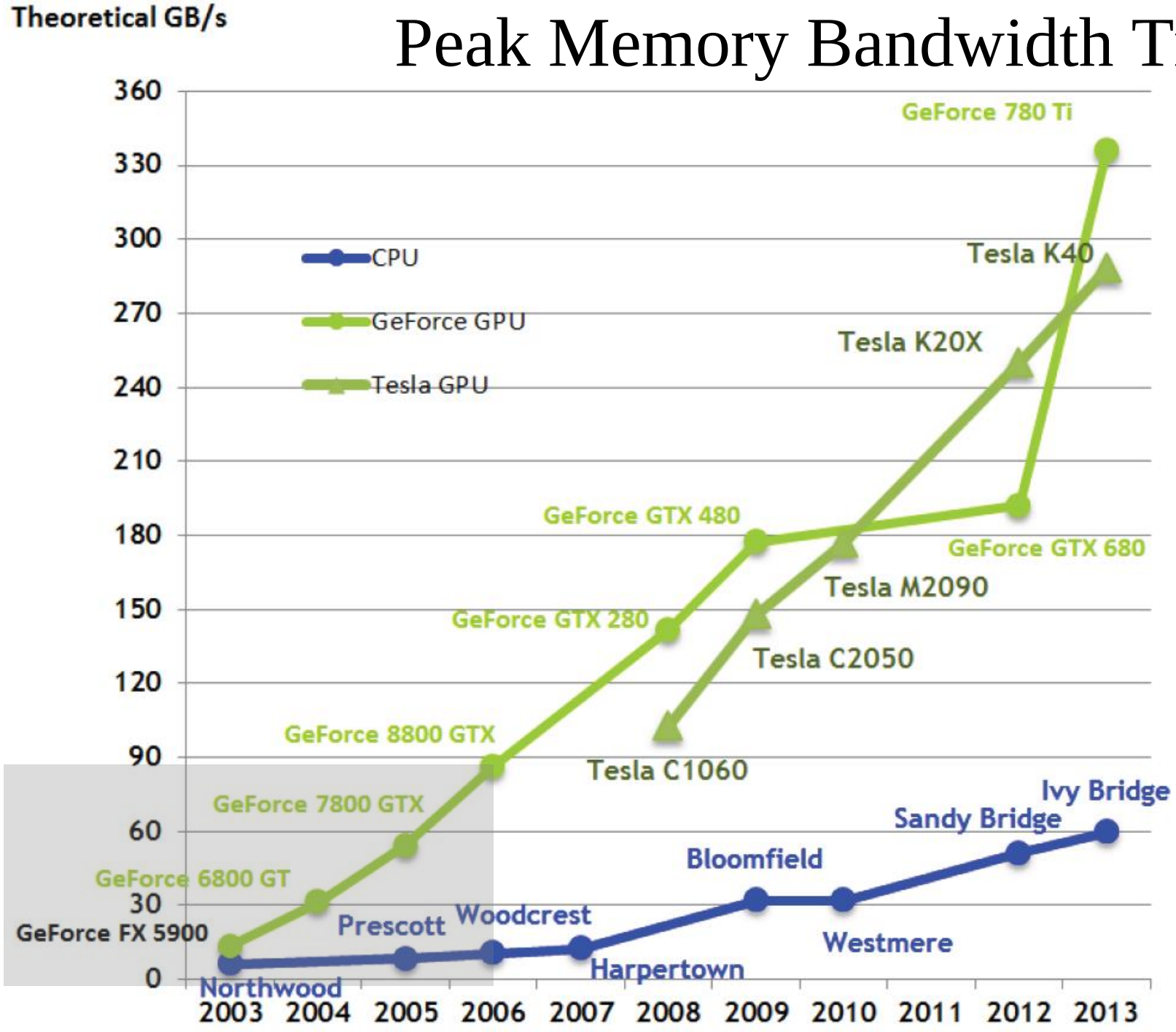
| **Node Type and Count** | **Script Load Time** | **State Load Time** | **Geometry + Ray Tracing** | **Total Time** |
|---|---|---|---|---|
| **256 XE6 CPU nodes** | 7 s | 160 s | **1,374 s** | **1,541 s** |
| 512 XE6 CPU nodes | 13 s | 211 s | 808 s | 1,032 s |
| 64 XK7 Tesla K20X GPUs | 2 s | 38 s | 655 s | 695 s |
| 128 XK7 Tesla K20X GPUs | 4 s | 74 s | 331 s | 410 s |
| **256 XK7 Tesla K20X GPUs** | 7 s | 110 s | **171 s** | **288 s** |

**GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms.**
Stone et al. In UltraVis'13: Eighth Workshop on Ultrascale Visualization Proceedings, 2013.

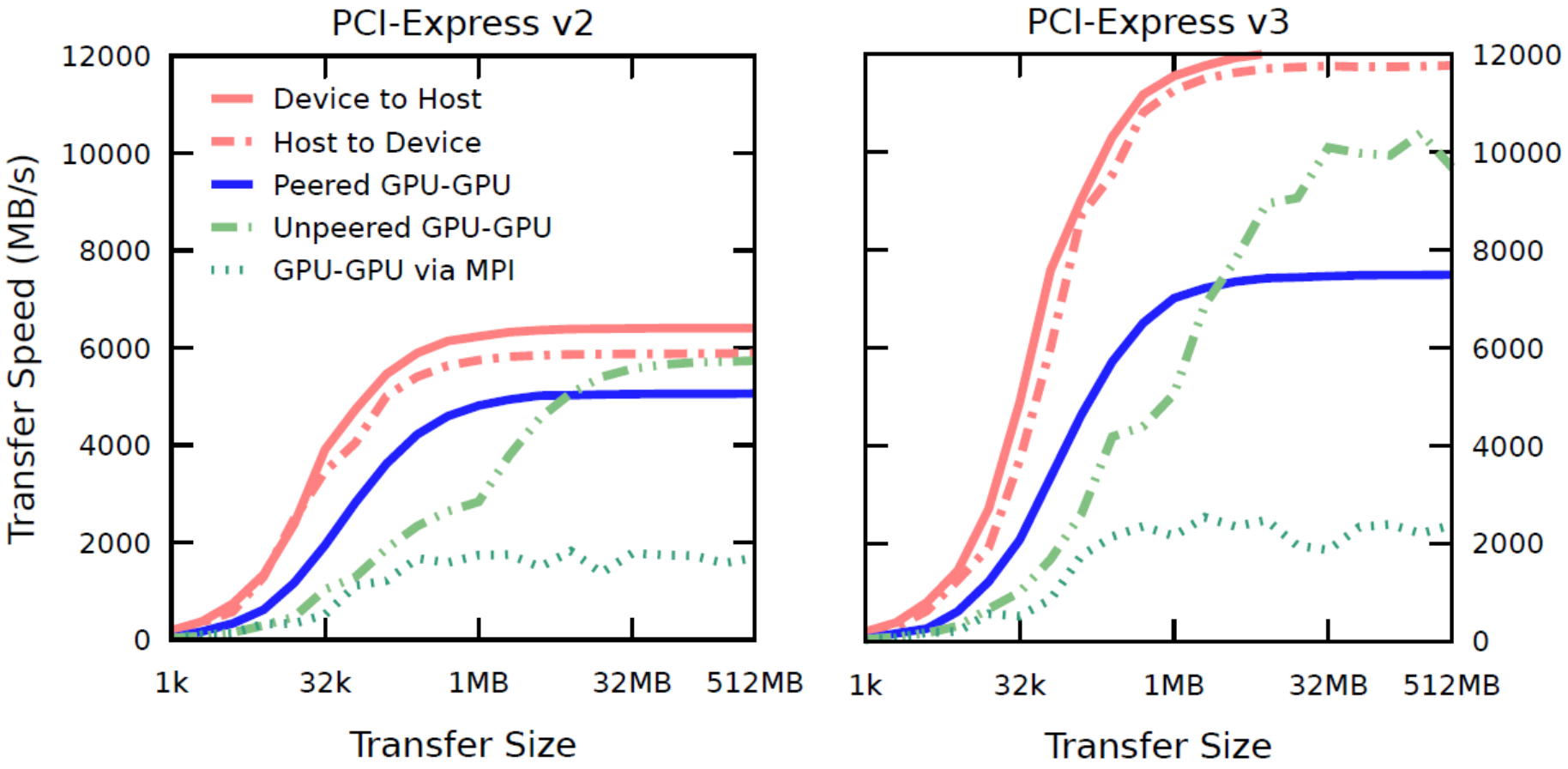
# Peak Arithmetic Performance Trend



# Peak Memory Bandwidth Trend



# GPU PCI-Express DMA



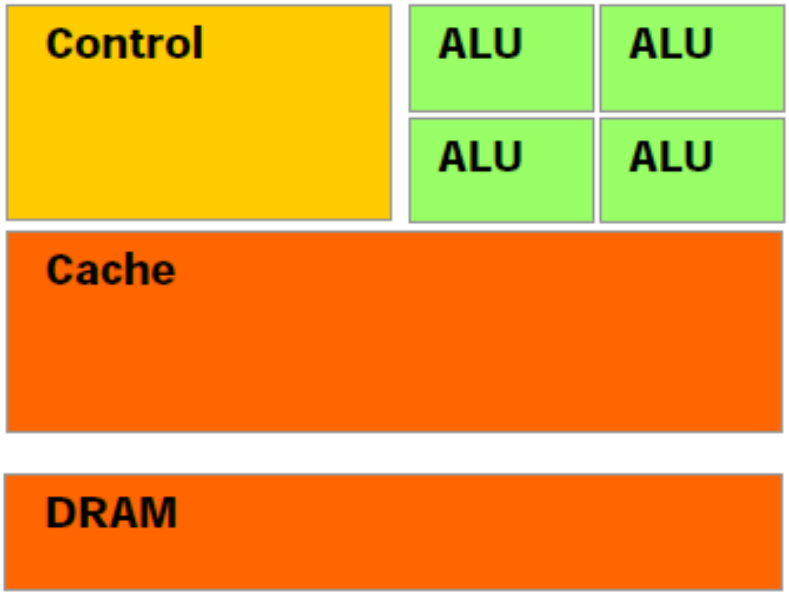

**Simulation of reaction diffusion processes over biologically relevant size and time scales using multi-GPU workstations**
Michael J. Hallock, John E. Stone, Elijah Roberts, Corey Fry, and Zaida Luthey-Schulten.
Journal of Parallel Computing, 2014. **(In press)**
**http://dx.doi.org/10.1016/j.parco.2014.03.009**

# GPU: Throughput-Oriented Hardware Architecture

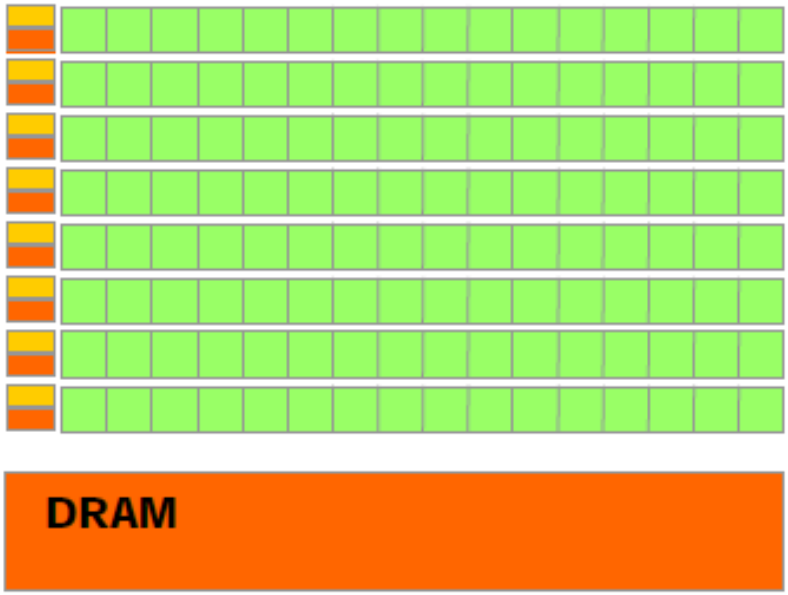
- GPUs have very small on-chip caches
- Main memory latency (several hundred clock cycles!) is tolerated through hardware multithreading – **overlap memory transfer latency with execution of other work**
- When a GPU thread stalls on a memory operation, the hardware **immediately switches** context to a ready thread
- Effective latency hiding requires saturating the GPU with lots of work – **tens of thousands** of **independent** work items

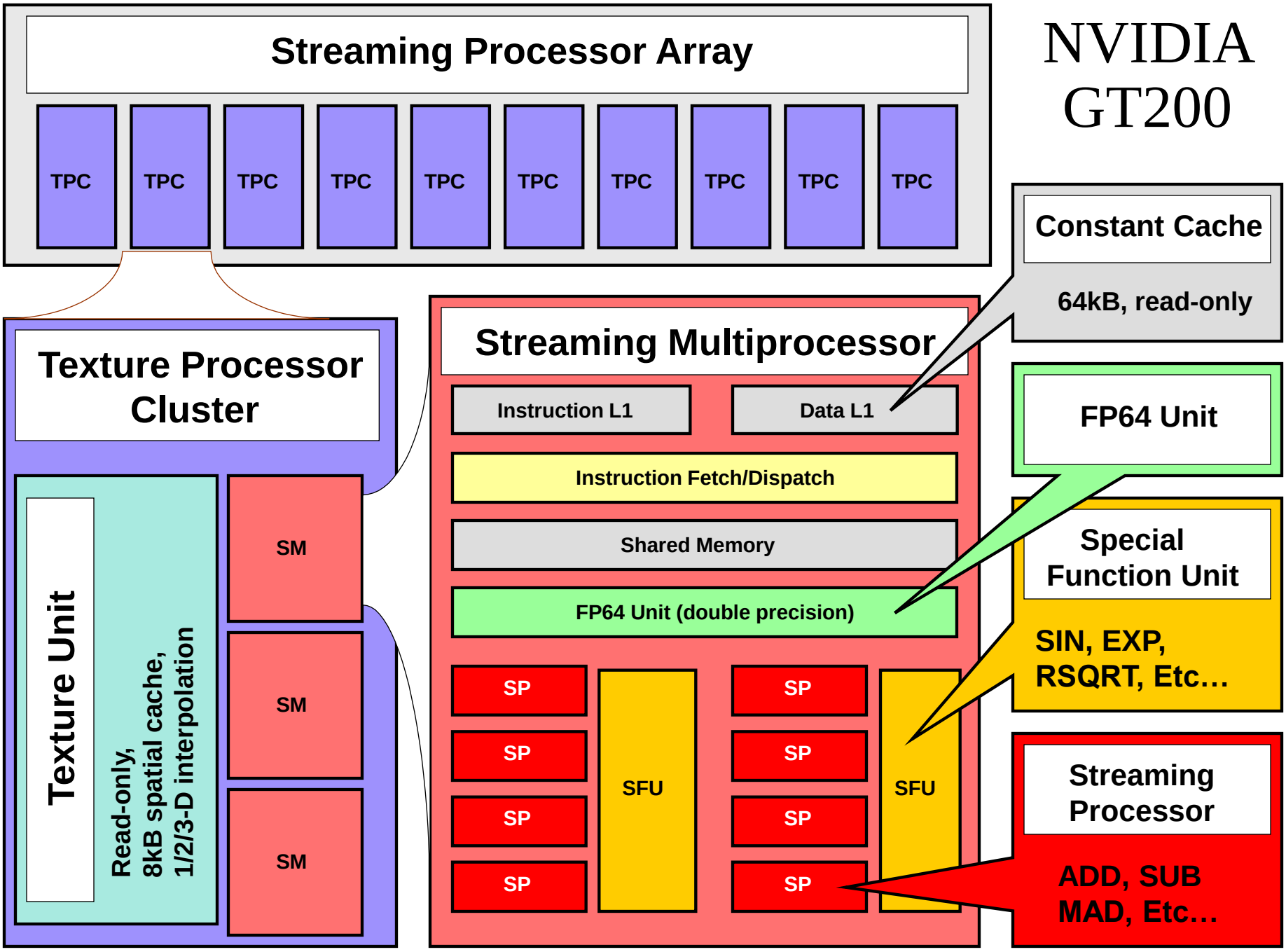
# Comparison of CPU and GPU Hardware Architecture

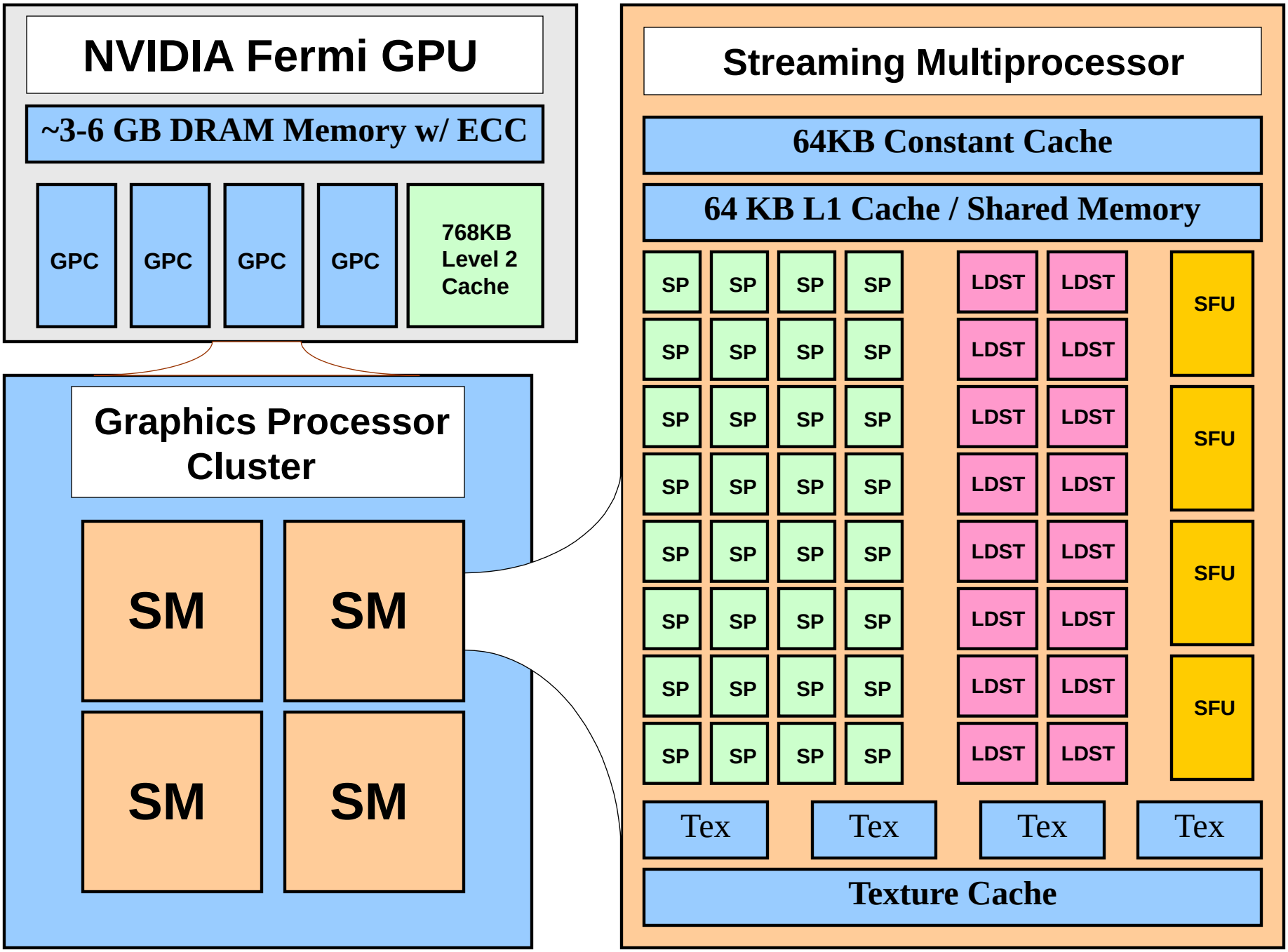
**CPU**: Cache heavy, focused on individual thread performance

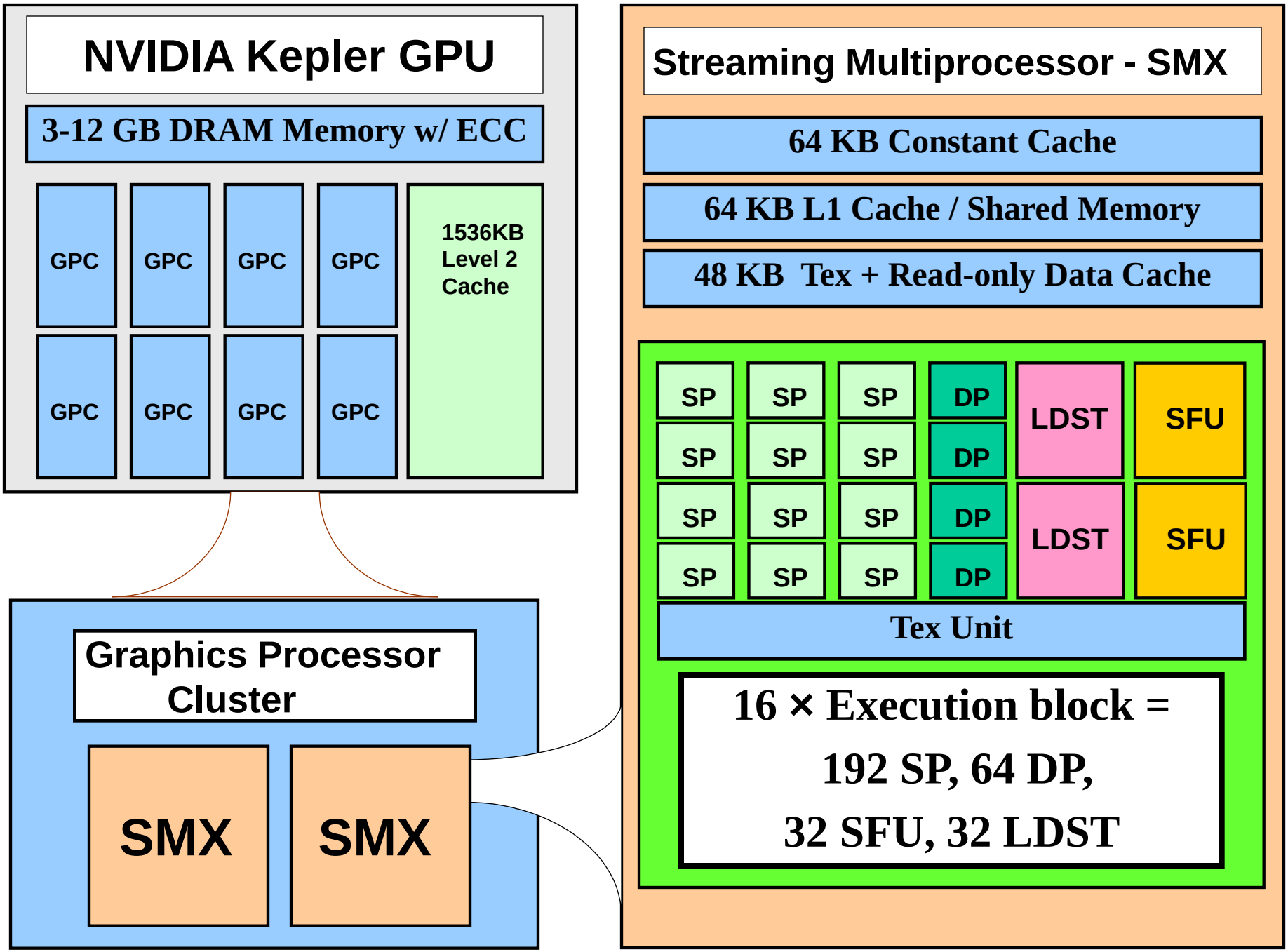



**GPU**: ALU heavy, massively parallel, throughput oriented




Streaming Processor Array
TPC
TPC
TPC
TPC
TPC
TPC
TPC
TPC
TPC
TPC
NVIDIA
GT200
Texture Processor Cluster
Texture Unit
Read-only,
8kB spatial cache,
1/2/3-D interpolation
SM
SM
SM
Streaming Multiprocessor
Instruction L1
Data L1
Instruction Fetch/Dispatch
Shared Memory
FP64 Unit (double precision)
SP
SP
SP
SP
SFU
SP
SP
SP
SP
SFU
Constant Cache
64kB, read-only
FP64 Unit
Special Function Unit
SIN, EXP,
RSQRT, Etc…
Streaming Processor
ADD, SUB
MAD, Etc…


NVIDIA Fermi GPU
~3-6 GB DRAM Memory w/ ECC
GPC
GPC
GPC
GPC
768KB Level 2 Cache
Graphics Processor Cluster
SM
SM
SM
SM
Streaming Multiprocessor
64KB Constant Cache
64 KB L1 Cache / Shared Memory
SP
LDST
SFU
Tex
Texture Cache


NVIDIA Kepler GPU
3-12 GB DRAM Memory w/ ECC
GPC
GPC
GPC
GPC
GPC
GPC
GPC
GPC
GPC
1536KB Level 2 Cache
Graphics Processor Cluster
SMX
SMX
Streaming Multiprocessor - SMX
64 KB Constant Cache
64 KB L1 Cache / Shared Memory
48 KB Tex + Read-only Data Cache
SP
DP
LDST
SFU
Tex Unit
16 × Execution block =
192 SP, 64 DP,
32 SFU, 32 LDST

# Acknowledgements


- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- NVIDIA CUDA Center of Excellence, University of Illinois at Urbana-Champaign
- NVIDIA CUDA team
- NVIDIA OptiX team
- NCSA Blue Waters Team
- Funding:
  - NSF OCI 07-25070
  - NSF PRAC “The Computational Microscope”
  - NIH support: 9P41GM104601, 5R01GM098243-02


NIH BTRC for Macromolecular Modeling and Bioinformatics
1990-2017
Beckman Institute
University of Illinois at
Urbana-Champaign

# GPU Computing Publications

## http://www.ks.uiuc.edu/Research/gpu/

- **Runtime and Architecture Support for Efficient Data Exchange in Multi-Accelerator Applications** Javier Cabezas, Isaac Gelado, John E. Stone, Nacho Navarro, David B. Kirk, and Wen-mei Hwu. IEEE Transactions on Parallel and Distributed Systems, 2014. (Accepted)
- **Unlocking the Full Potential of the Cray XK7 Accelerator** Mark Klein and John E. Stone. Cray Users Group, 2014. (In press)
- **Simulation of reaction diffusion processes over biologically relevant size and time scales using multi-GPU workstations** Michael J. Hallock, John E. Stone, Elijah Roberts, Corey Fry, and Zaida Luthey-Schulten. Journal of Parallel Computing, 2014. (In press)
- **GPU-Accelerated Analysis and Visualization of Large Structures Solved by Molecular Dynamics Flexible Fitting** John E. Stone, Ryan McGreevy, Barry Isralewitz, and Klaus Schulten. Faraday Discussion 169, 2014. (In press)
- **GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms.** J. Stone, K. L. Vandivort, and K. Schulten. UltraVis'13: Proceedings of the 8th International Workshop on Ultrascale Visualization, pp. 6:1-6:8, 2013.
- **Early Experiences Scaling VMD Molecular Visualization and Analysis Jobs on Blue Waters.** J. E. Stone, B. Isralewitz, and K. Schulten. In proceedings, Extreme Scaling Workshop, 2013.
- **Lattice Microbes: High-performance stochastic simulation method for the reaction-diffusion master equation.** E. Roberts, J. E. Stone, and Z. Luthey-Schulten. J. Computational Chemistry 34 (3), 245-255, 2013**.**

# GPU Computing Publications

## http://www.ks.uiuc.edu/Research/gpu/

- **Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.** M. Krone, J. E. Stone, T. Ertl, and K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012.
- **Fast Analysis of Molecular Dynamics Trajectories with Graphics Processing Units – Radial Distribution Functions.** B. Levine, J. Stone, and A. Kohlmeyer. *J. Comp. Physics*, 230(9):3556-3569, 2011.
- **Immersive Out-of-Core Visualization of Large-Size and Long-Timescale Molecular Dynamics Trajectories.** J. Stone, K. Vandivort, and K. Schulten. G. Bebis et al. (Eds.): *7th International Symposium on Visual Computing (ISVC 2011)*, LNCS 6939, pp. 1-12, 2011.
- **Quantifying the Impact of GPUs on Performance and Energy Efficiency in HPC Clusters.** J. Enos, C. Steffen, J. Fullop, M. Showerman, G. Shi, K. Esler, V. Kindratenko, J. Stone, J Phillips. *International Conference on Green Computing*, pp. 317-324, 2010.
- **GPU-accelerated molecular modeling coming of age. J. Stone, D. Hardy, I. Ufimtsev, K. Schulten.** *J. Molecular Graphics and Modeling*, 29:116-125, 2010.
- **OpenCL: A Parallel Programming Standard for Heterogeneous Computing. J. Stone, D. Gohara, G. Shi.** *Computing in Science and Engineering*, 12(3):66-73, 2010.

# GPU Computing Publications
## http://www.ks.uiuc.edu/Research/gpu/

- **An Asymmetric Distributed Shared Memory Model for Heterogeneous Computing Systems**. I. Gelado, J. Stone, J. Cabezas, S. Patel, N. Navarro, W. Hwu. *ASPLOS '10: Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 347-358, 2010.
- **GPU Clusters for High Performance Computing**. V. Kindratenko, J. Enos, G. Shi, M. Showerman, G. Arnold, J. Stone, J. Phillips, W. Hwu. *Workshop on Parallel Programming on Accelerator Clusters (PPAC)*, In Proceedings IEEE Cluster 2009, pp. 1-8, Aug. 2009.
- **Long time-scale simulations of in vivo diffusion using GPU hardware**. E. Roberts, J. Stone, L. Sepulveda, W. Hwu, Z. Luthey-Schulten. In *IPDPS'09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Computing*, pp. 1-8, 2009.
- **High Performance Computation and Interactive Display of Molecular Orbitals on GPUs and Multi-core CPUs**. J. Stone, J. Saam, D. Hardy, K. Vandivort, W. Hwu, K. Schulten, *2nd Workshop on General-Purpose Computation on Graphics Pricessing Units (GPGPU-2), ACM International Conference Proceeding Series*, volume 383, pp. 9-18, 2009.
- **Probing Biomolecular Machines with Graphics Processors**. J. Phillips, J. Stone. *Communications of the ACM*, 52(10):34-41, 2009.
- **Multilevel summation of electrostatic potentials using graphics processing units**. D. Hardy, J. Stone, K. Schulten. *J. Parallel Computing*, 35:164-177, 2009.

# GPU Computing Publications

## http://www.ks.uiuc.edu/Research/gpu/

- **Adapting a message-driven parallel application to GPU-accelerated clusters**. J. Phillips, J. Stone, K. Schulten. *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, IEEE Press, 2008.
- **GPU acceleration of cutoff pair potentials for molecular modeling applications**. C. Rodrigues, D. Hardy, J. Stone, K. Schulten, and W. Hwu. *Proceedings of the 2008 Conference On Computing Frontiers*, pp. 273-282, 2008.
- **GPU computing**. J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, J. Phillips. *Proceedings of the IEEE*, 96:879-899, 2008.
- **Accelerating molecular modeling applications with graphics processors**. J. Stone, J. Phillips, P. Freddolino, D. Hardy, L. Trabuco, K. Schulten. *J. Comp. Chem.*, 28:2618-2640, 2007.
- **Continuous fluorescence microphotolysis and correlation spectroscopy**. A. Arkhipov, J. Hüve, M. Kahms, R. Peters, K. Schulten. *Biophysical Journal*, 93:4006-4017, 2007.