High Performance Computing with Accelerators

Volodymyr Kindratenko

Innovative Systems Laboratory @ NCSA

Institute for Advanced Computing Applications and Technologies (IACAT)

National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

Presentation Outline

- **Recent trends in application accelerators**
  - FPGAs, Cell, GPUs, …
  - Accelerator clusters
- **Accelerator clusters at NCSA**
  - Cluster management
  - Programmability Issues
- **Production codes running on Accelerator Clusters**
  - Cosmology, molecular dynamics, electronic structure, quantum chromodynamics

HPC on Special-purpose Processors

- **Field-Programmable Gate Arrays (FPGAs)**
  - Digital signal processing, embedded computing

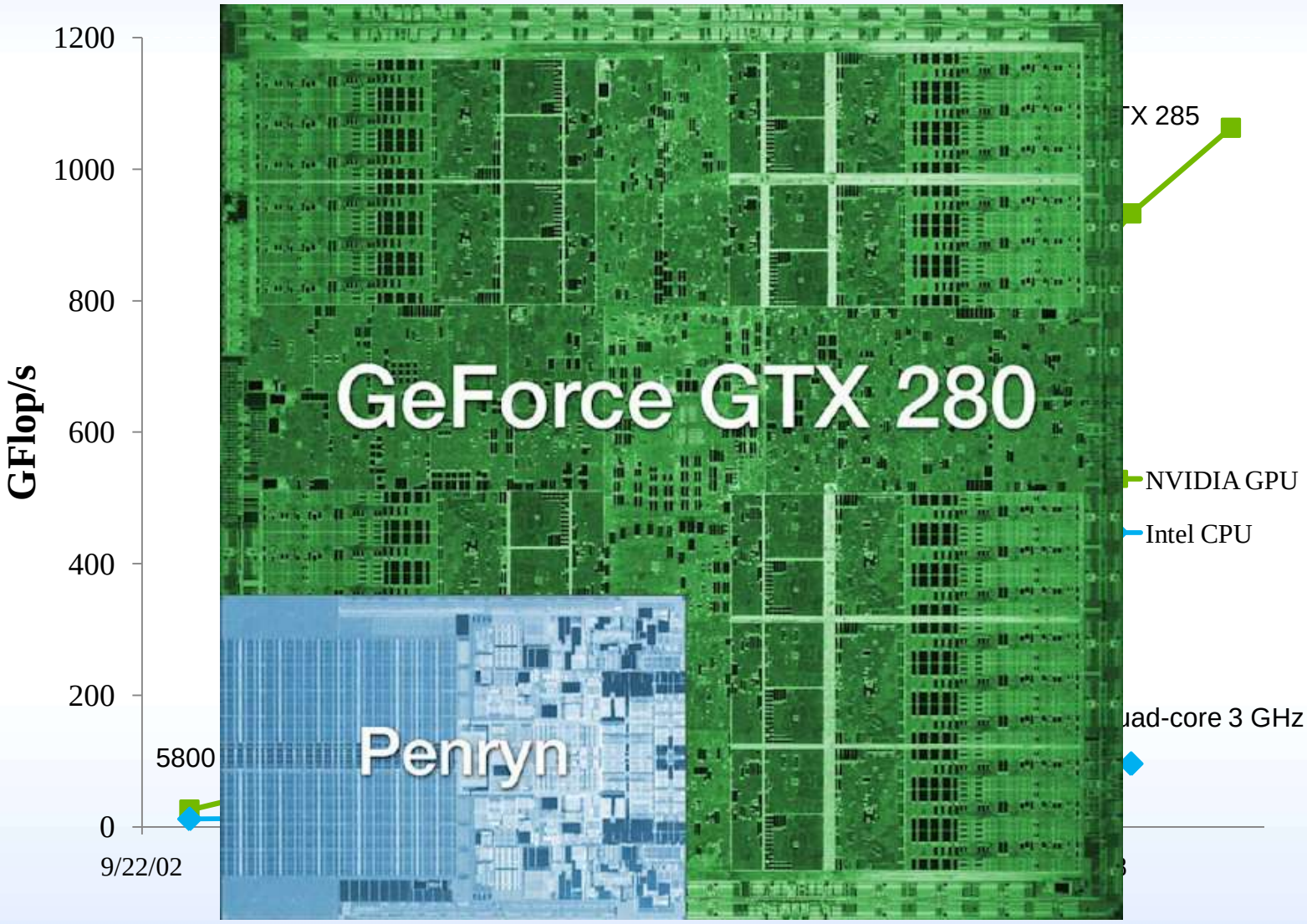
- **Graphics Processing Units (GPUs)**
  - Desktop graphics accelerators

- **Physics Processing Units (PPUs)**
  - Desktop games accelerators

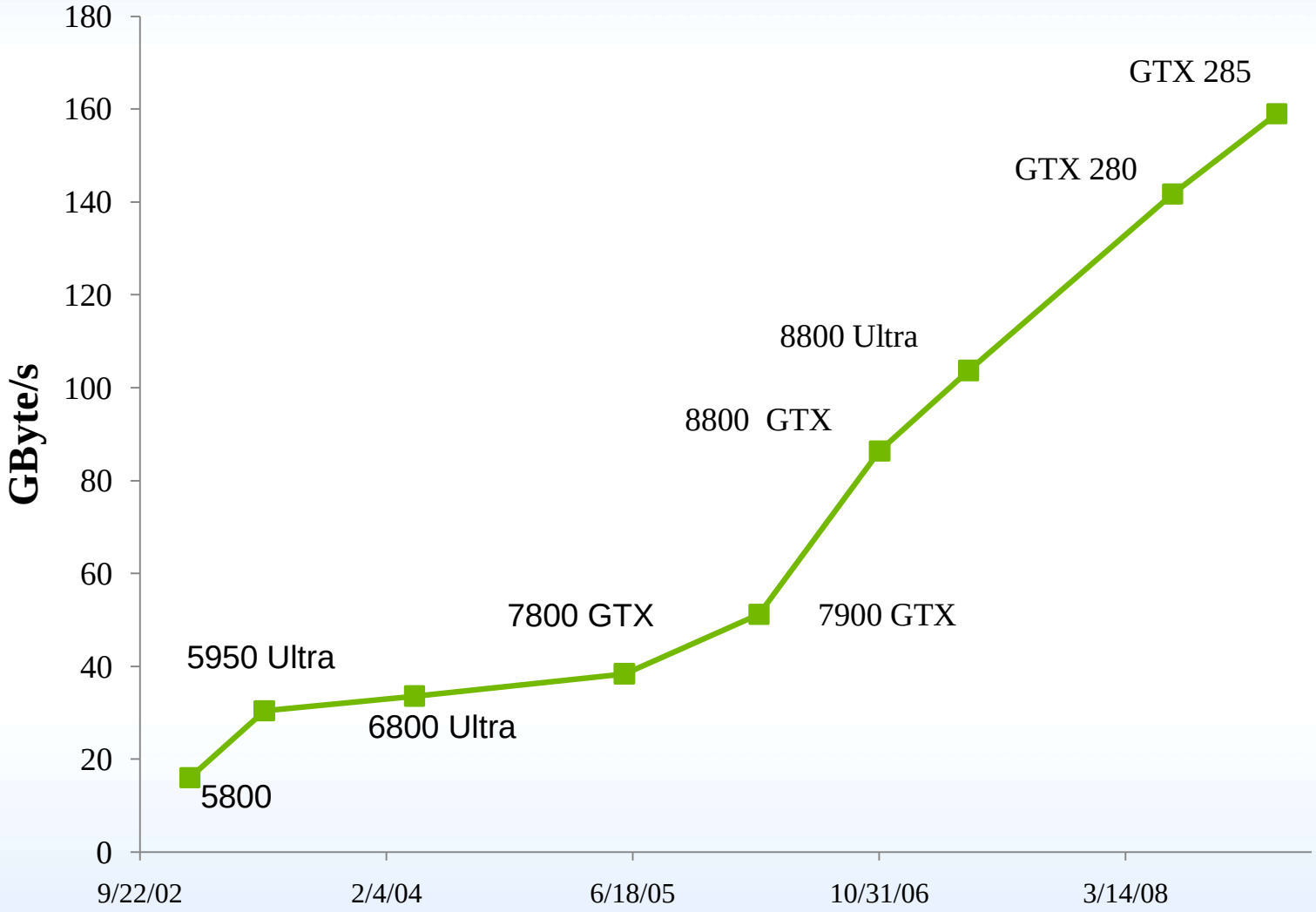
- **Sony/Toshiba/IBM Cell Broadband Engine**
  - Game console and digital content delivery systems

- **ClearSpeed accelerator**
  - Floating-point accelerator board for compute-intensive applications

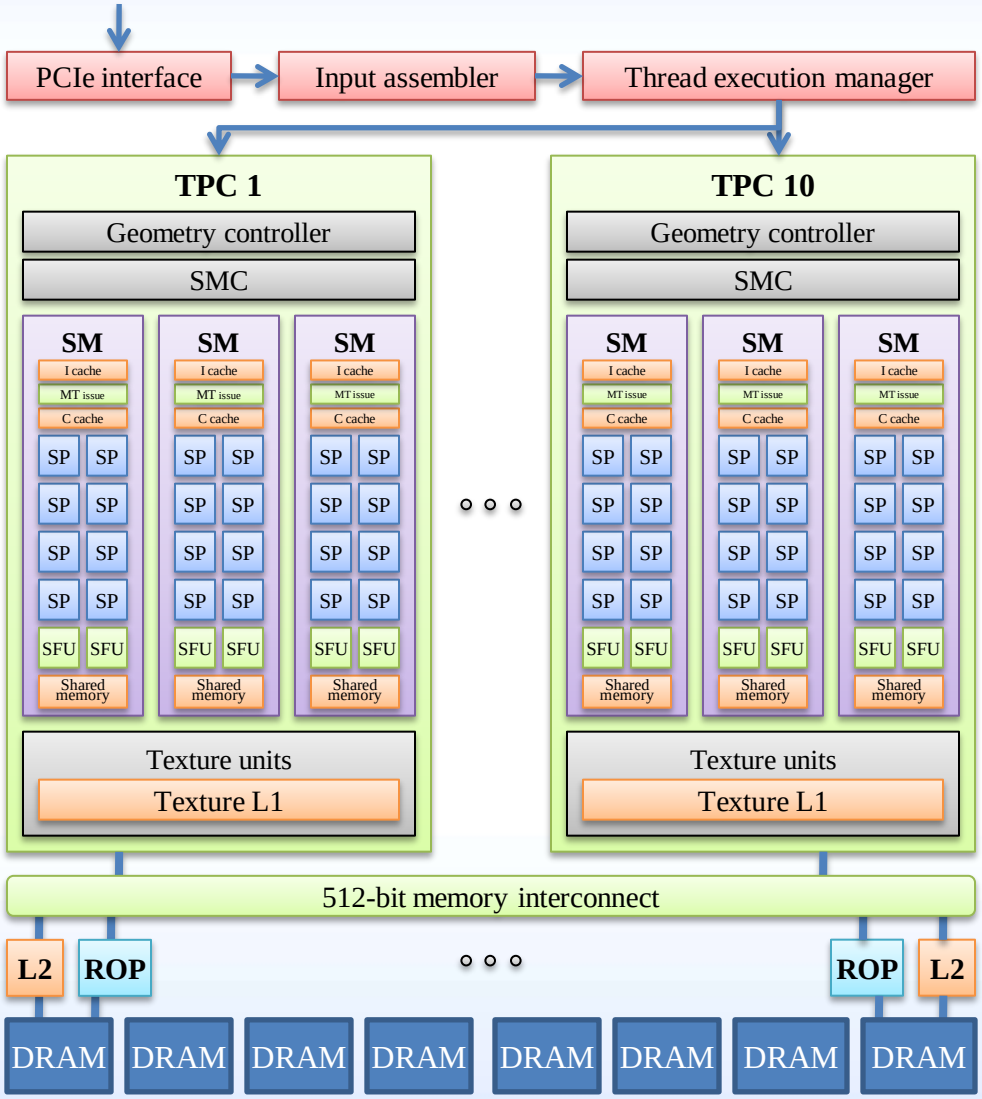
GPU Performance Trends: FLOPS



GPU Performance Trends: Memory Bandwidth



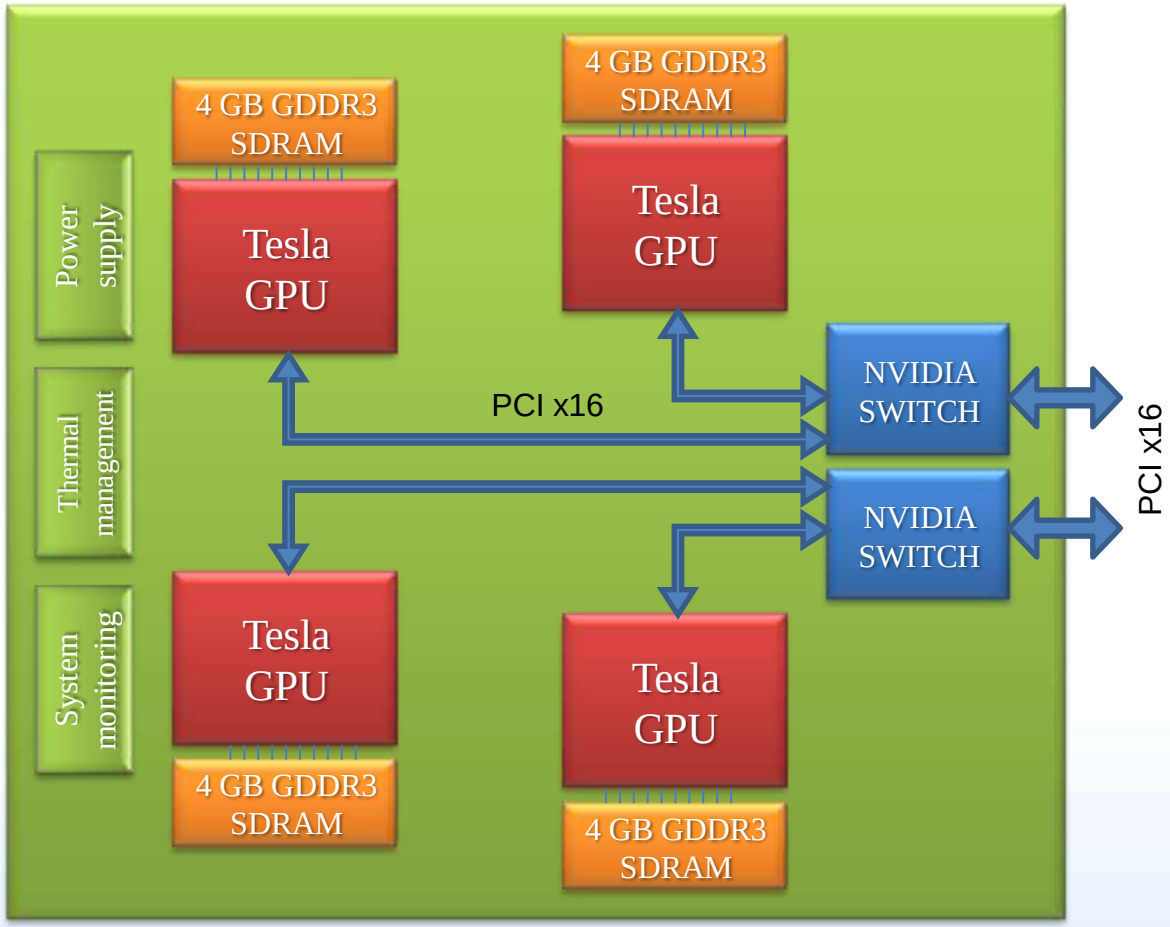
NVIDIA Tesla T10 GPU Architecture



- **T10 architecture**
  - 240 streaming processors arranged as 30 streaming multiprocessors
  - At 1.3 GHz this provides
    - 1 TFLOPS SP
    - 86.4 TFLOPS DP
  - 512-bit interface to off-chip GDDR3 memory
    - 102 GB/s bandwidth

NVIDIA Tesla S1070 GPU Computing Server

- **4 T10 GPUs**

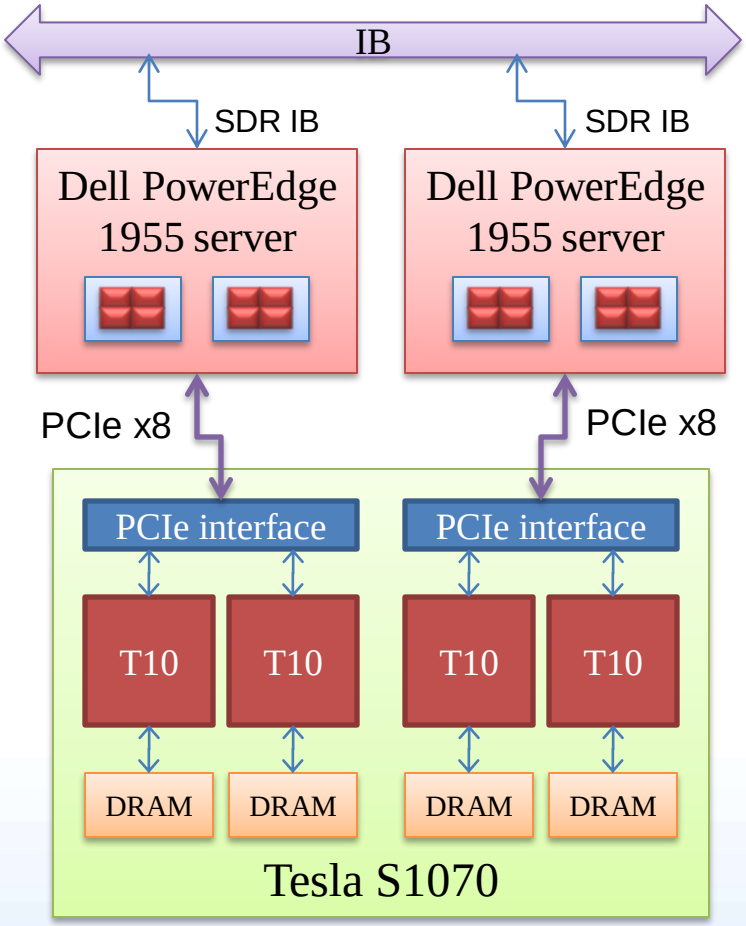


GPU Clusters at NCSA

- **Lincoln**
  - Production system available the standard NCSA/TeraGrid HPC allocation
- **AC**
  - Experimental system available for anybody who is interested in exploring GPU computing

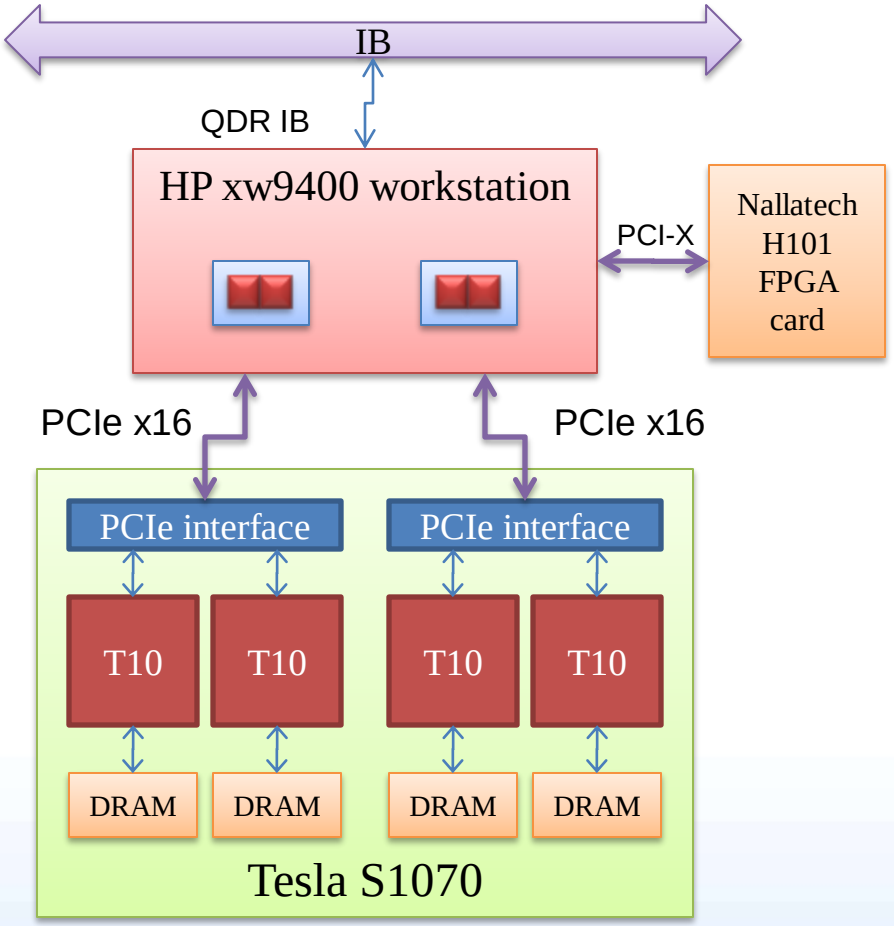
Intel 64 Tesla Linux Cluster *Lincoln*

- **Dell PowerEdge 1955 server**
  - Intel 64 (Harpertown) 2.33 GHz dual socket quad core
  - 16 GB DDR2
  - InfiniBand SDR
- **S1070 1U GPU Computing Server**
  - 1.3 GHz Tesla T10 processors
  - 4x4 GB GDDR3 SDRAM
- **Cluster**
  - Servers: 192
    - CPU cores: 1536
  - Accelerator Units: 96
    - GPUs: 384



AMD AMD Opteron Tesla Linux Cluster *AC*

- **HP xw9400 workstation**
  - 2216 AMD Opteron 2.4 GHz dual socket dual core
  - 8 GB DDR2
  - InfiniBand QDR
- **S1070 1U GPU Computing Server**
  - 1.3 GHz Tesla T10 processors
  - 4x4 GB GDDR3 SDRAM
- **Cluster**
  - Servers: 32
    - CPU cores: 128
  - Accelerator Units: 32
    - GPUs: 128



AC Cluster: 128 TFLOPS (SP)

NCSA GPU Cluster Management Software

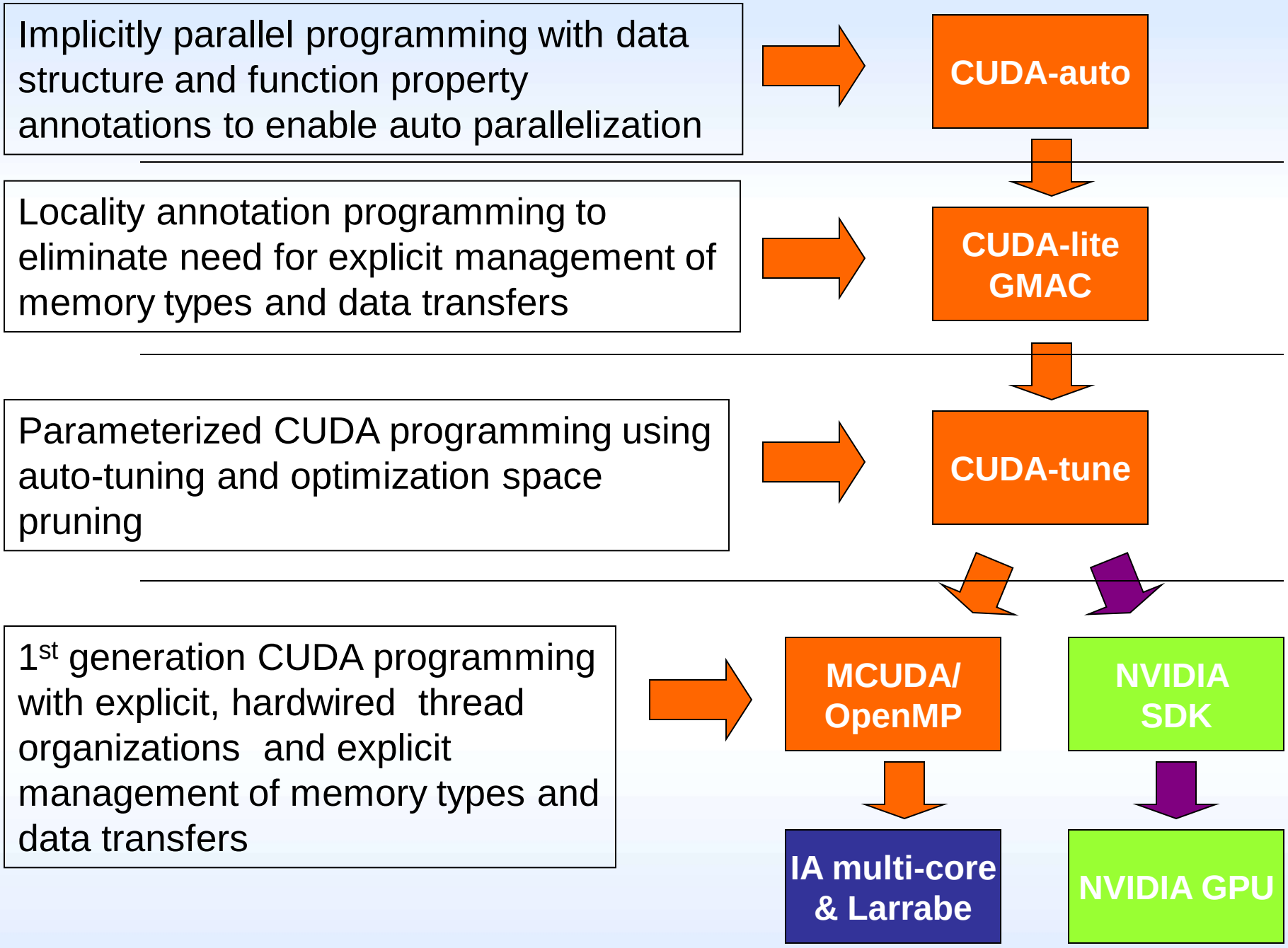
- **CUDA SDK Wrapper**
  - Enables true GPU resources allocation by the scheduler
    - E.g., **qsub -l nodes=1:ppn=1** will allocate 1 CPU core and 1 GPU, fencing other GPUs in the node from accessing
  - Virtualizes GPU devices
  - Sets thread affinity to CPU core "nearest" the the GPU device
- **GPU Memory Scrubber**
  - Cleans memory between runs
- **GPU Memory Test utility**
  - Tests memory for manufacturing defects
  - Tests memory for soft errors
  - Tests GPUs for entering an erroneous state

GPU Node Pre/Post Allocation Sequence

- **Pre-Job (minimized for rapid device acquisition)**
  - Assemble detected device file unless it exists
  - Sanity check results
  - Checkout requested GPU devices from that file
  - Initialize CUDA wrapper shared memory segment with unique key for user (allows user to ssh to node outside of job environment and have same gpu devices visible)
- **Post-Job**
  - Use quick memtest run to verify healthy GPU state
  - If bad state detected, mark node offline if other jobs present on node
  - If no other jobs, reload kernel module to “heal” node (for CUDA 2.2 bug)
  - Run memscrubber utility to clear gpu device memory
  - Notify of any failure events with job details
  - Terminate wrapper shared memory segment
  - Check-in GPUs back to global file of detected devices

GPU Programming

- **3rd Party Programming tools**
  - CUDA C 2.2 SDK
  - OpenCL 1.2 SDK
  - PGI x64+GPU Fortran & C99 Compilers
- **IACAT's on-going work**
  - CUDA-auto
  - CUDA-lite
  - GMAC
  - CUDA-tune
  - MCUDA/OpenMP


Implicitly parallel programming with data structure and function property annotations to enable auto parallelization
CUDA-auto
Locality annotation programming to eliminate need for explicit management of memory types and data transfers
CUDA-lite GMAC
Parameterized CUDA programming using auto-tuning and optimization space pruning
CUDA-tune
1st generation CUDA programming with explicit, hardwired thread organizations and explicit management of memory types and data transfers
MCUDA/ OpenMP
NVIDIA SDK
IA multi-core & Larrabe
NVIDIA GPU

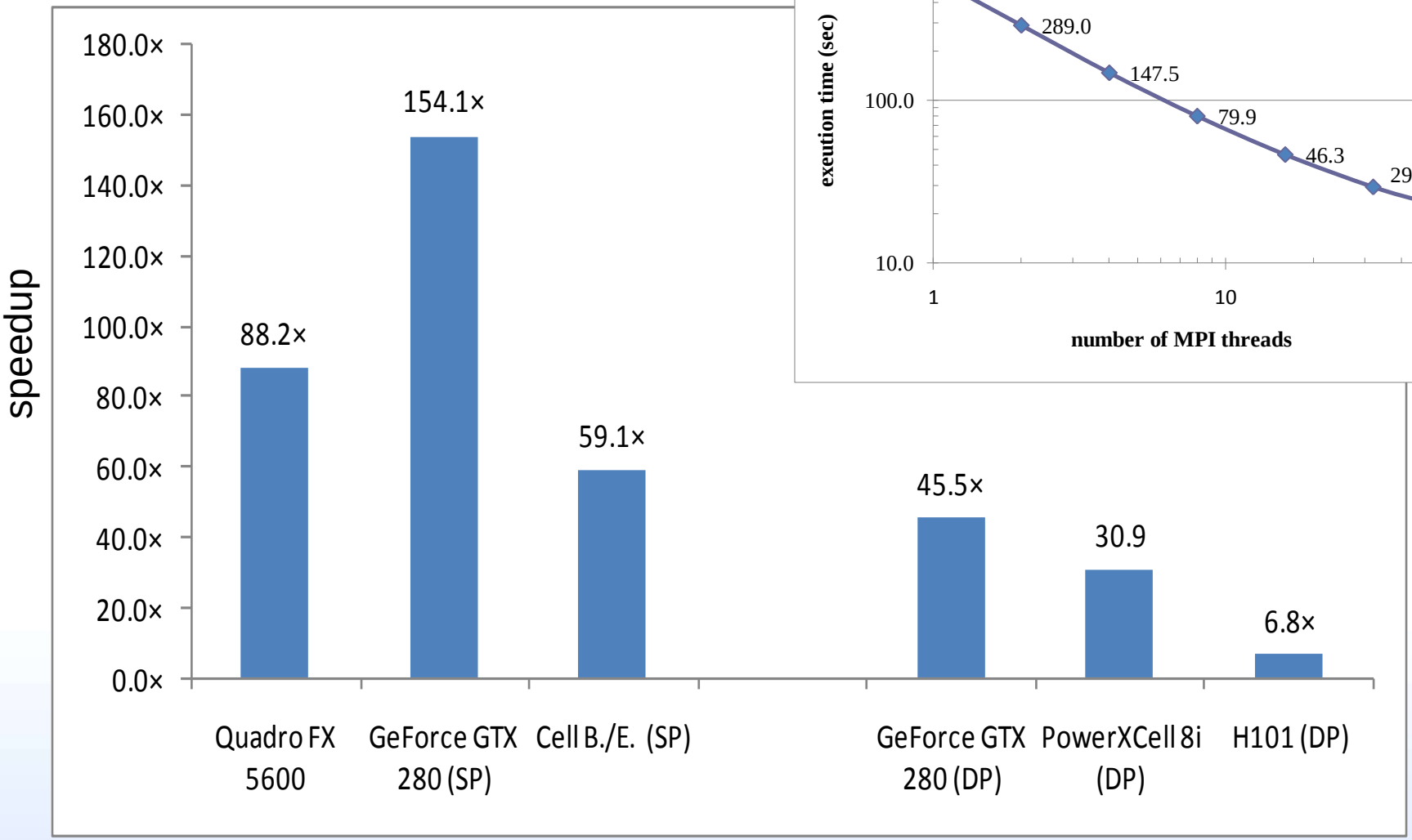
GMAC – Designed to reduce accelerator use barrier

- **Unified CPU / GPU Address Space:**
  - Same CPU and GPU address
- **Customizable implicit data transfers:**
  - Transfer everything (safe mode)
  - Transfer dirty data before kernel execution
  - Transfer data as being produced (default)
- **Multi-process / Multi-thread support**
- **CUDA compatible**

GPU Cluster Applications

- **TPACF**
  - Cosmology code used to study how the matter is distributed in the Universe
- **NAMD**
  - Molecular Dynamics code used to run large-scale MD simulations
- **DSCF**
  - NSF CyberChem project; DSCF quantum chemistry code for energy calculations
- **WRF**
  - Weather modeling code
- **MILC**
  - Quantum chromodynamics code, work just started
- **…**

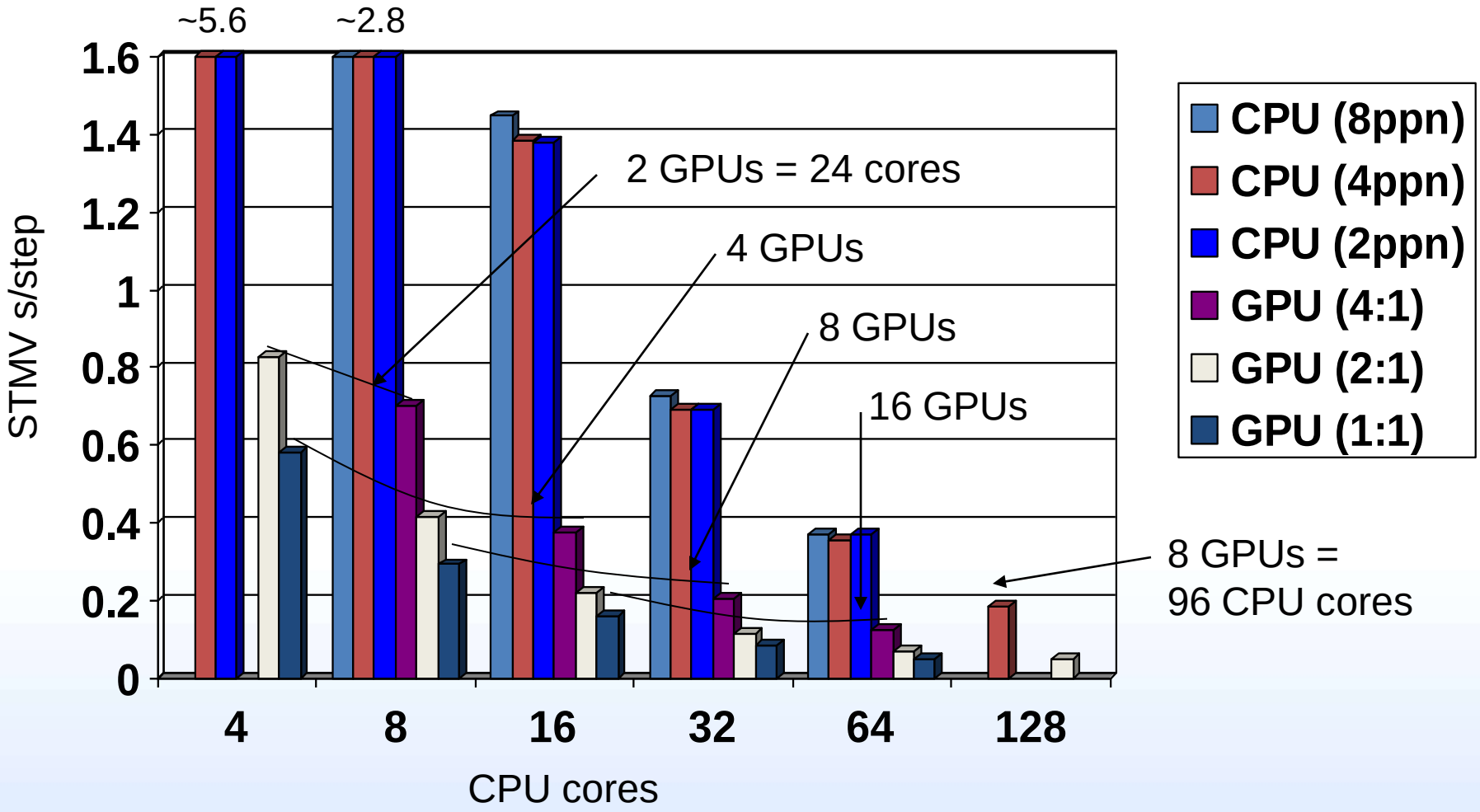
TPACF on *AC*



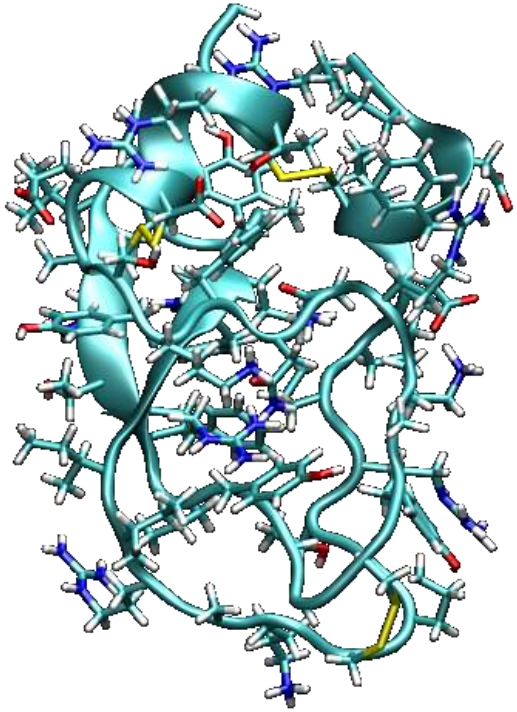
QP GPU cluster scaling

NAMD on *Lincoln*

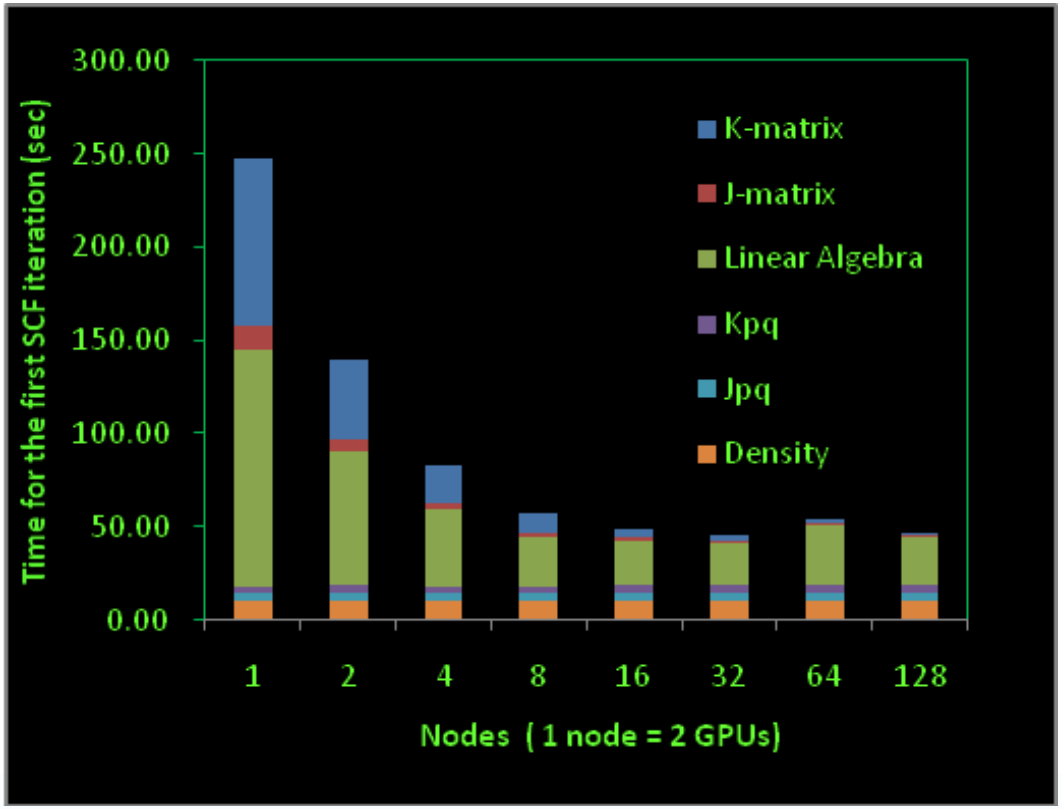
(8 cores and 2 GPUs per node, very early results)



DSCF on *Lincoln*



Bovine pancreatic trypsin inhibitor (BPTI) 3-21G, 875 atoms, 4893 basis functions



MPI timings and scalability

GPU Clusters Open Issues / Future Work

- **GPU cluster configuration**
  - CPU/GPU ratio
  - Host/device memory ratio
  - Cluster interconnect requirements
- **Cluster management**
  - Resources allocation
  - Monitoring
  - CUDA/OpenCL SDK for HPC
- **Programming models for accelerator clusters**
  - CUDA/OpenCL+pthreads/MPI/OpenMP/Charm++/…