Adapting Scientific Software and Designing Algorithms for Next Generation GPU Computing Platforms

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/

http://www.ks.uiuc.edu/Research/namd/

http://www.ks.uiuc.edu/Research/vmd/

Novel Computational Algorithms for Future Computing Platforms I

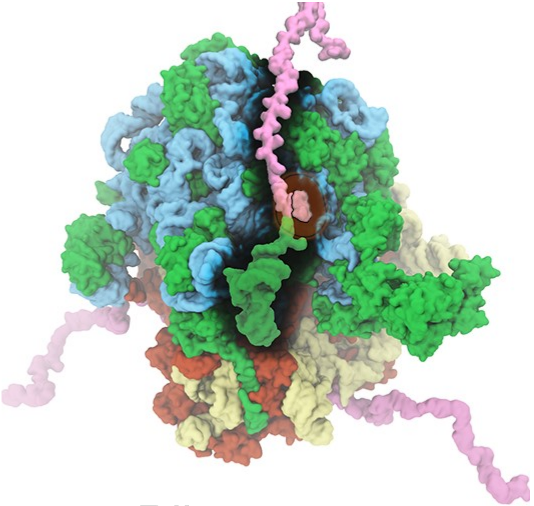
SIAM CSE 2019, Wednesday, February 27th, 2019

NAMD & VMD: Computational Microscope

Enable researchers to investigate systems described at the atomic scale

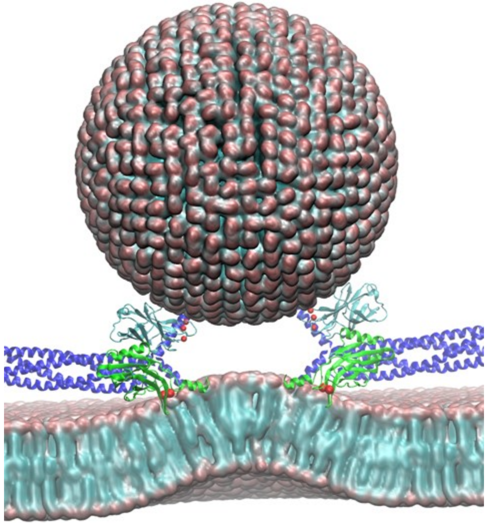
NAMD - molecular dynamics simulation

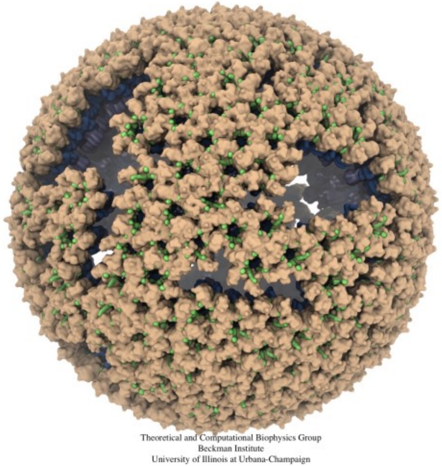
VMD - visualization, system preparation and analysis



Ribosome

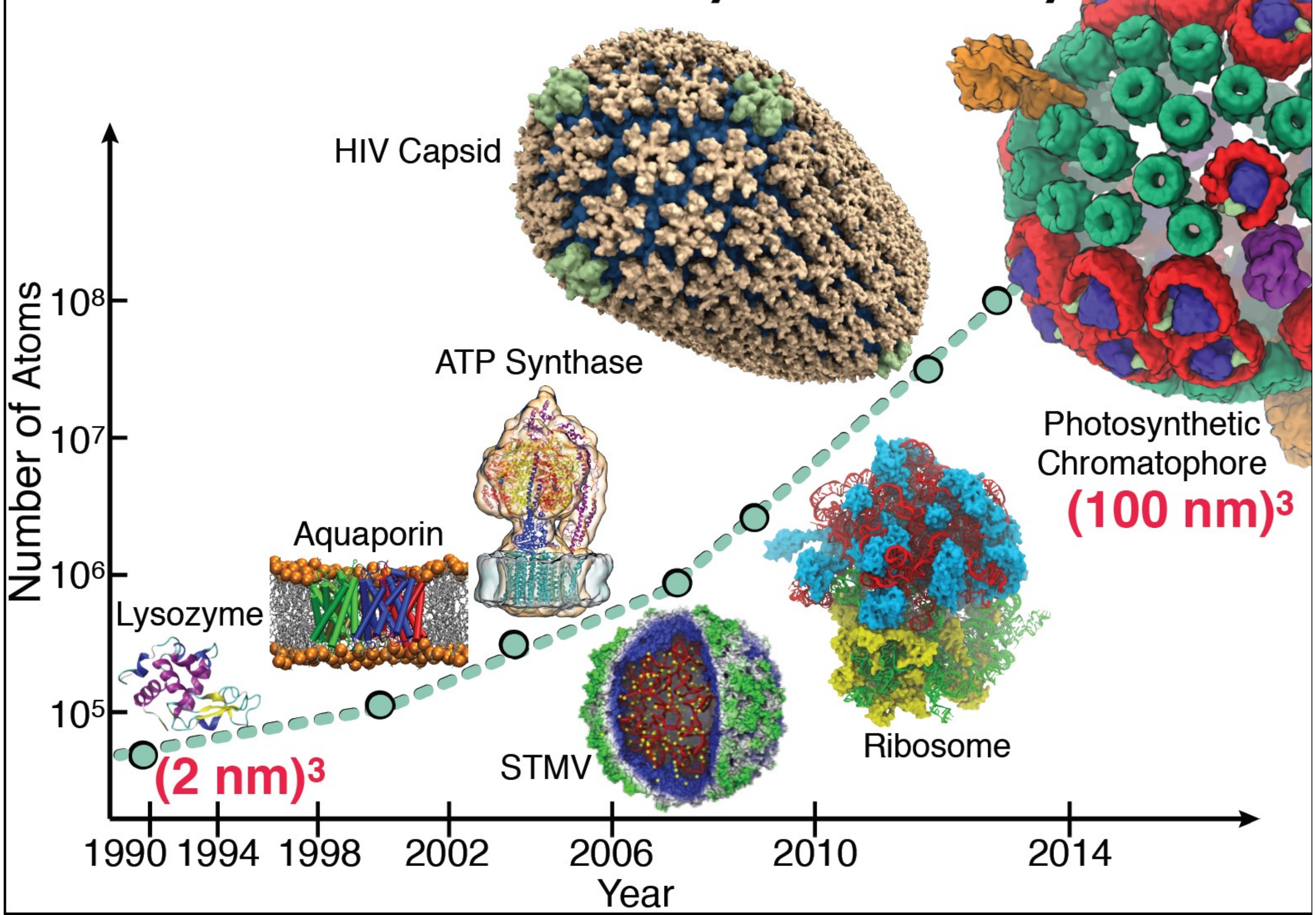
Neuron





Virus Capsid

All-Atom Molecular Dynamics Today



Major Approaches For Programming Hybrid Architectures

- Use **drop-in libraries** in place of CPU-only libraries
  - **Little or no code development**
  - Examples: MAGMA, BLAS-variants, FFT libraries, etc.
  - **Speedups limited by Amdahl's Law** and overheads associated with data movement between CPUs and GPU accelerators
- Generate accelerator code as a variant of CPU source, e.g. using OpenMP and **OpenACC directives**, and similar
- Write **lower-level** accelerator-specific code, e.g. using **CUDA, OpenCL**, other approaches

Exemplary Hetereogeneous Computing Challenges

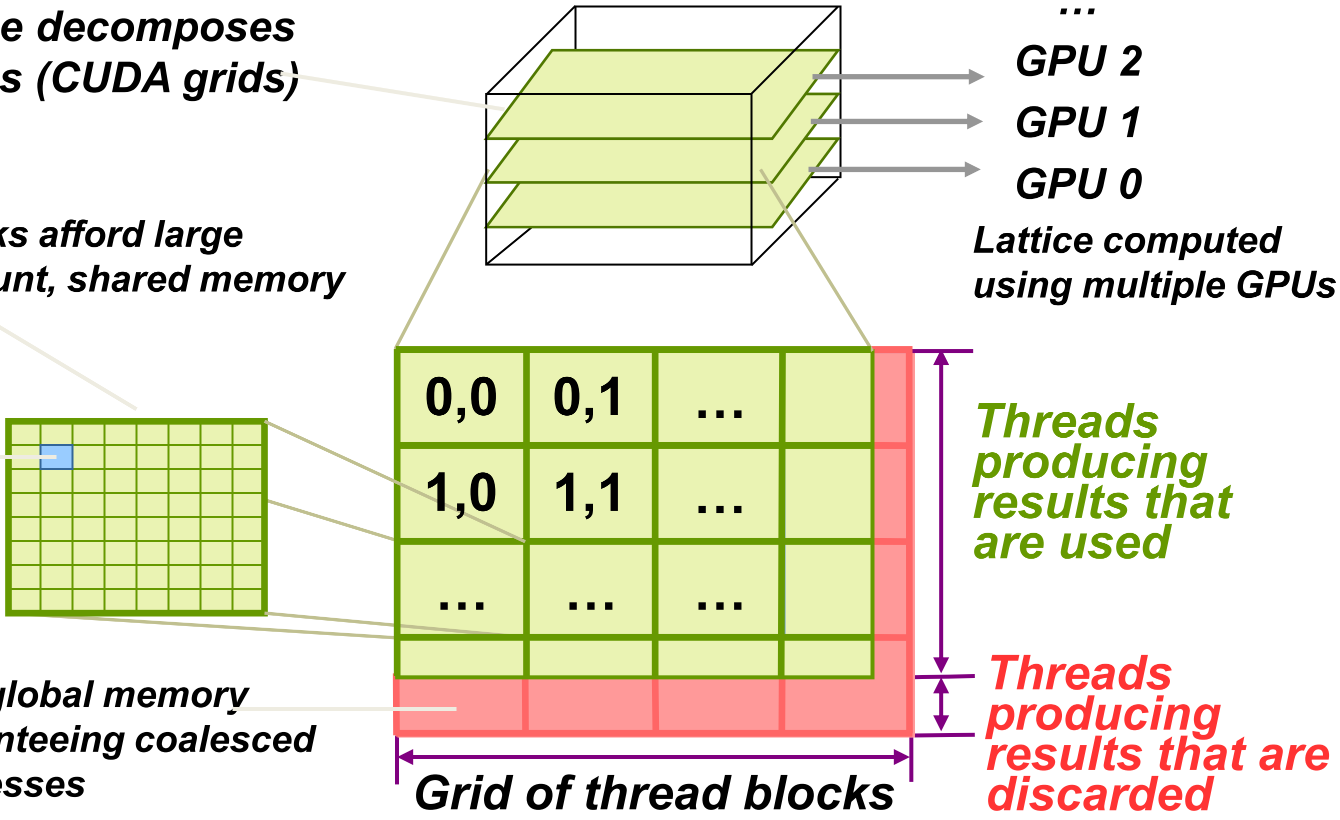
- **Tuning, adapting, or developing software for multiple processor types**
- Decomposition of problem(s) and load balancing work across heterogeneous resources for **best overall performance and work-efficiency**
- **Managing data placement** in disjoint memory systems with varying performance attributes
- **Transferring data** between processors, memory systems, interconnect, and I/O devices

Mol. Orbital GPU Parallel Decomposition

MO 3-D lattice decomposes into 2-D slices (CUDA grids)

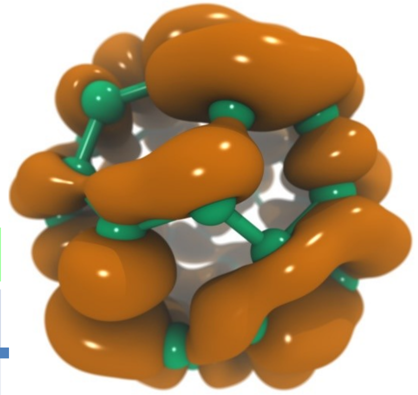
Small 8x8 thread blocks afford large per-thread register count, shared memory

Each thread computes one MO lattice point.



Padding optimizes global memory performance, guaranteeing coalesced global memory accesses

VMD Tesla V100 Performance for C_{60} Molecular Orbitals, 516x519x507 grid



Hardware platform	Runtime,	Speedup
IBM Power8 (ORNL 'crest') + 1x Tesla K40 [1]	3.49s,	**1.0x**
Intel Xeon E5-2697Av4 + 1x Tesla V100 [2]	**0.610s,**	**5.7x**
Intel Xeon E5-2697Av4 + 2x Tesla V100 [2]	**0.294s,**	**11.8x**
Intel Xeon E5-2697Av4 + 3x Tesla V100 [2]	**0.220s,**	**15.9x**
IBM Power9 "Newell" + 1x Tesla V100	**0.394s,**	**8.8x**
IBM Power9 "Newell" + 2x Tesla V100	**0.207s,**	**16.8x**
IBM Power9 "Newell" + 3x Tesla V100	**0.151s,**	**23.1x**
IBM Power9 "Newell" + 4x Tesla V100	**0.130s,**	**26.8x**

NVLink perf. boost

[1] **Early Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms.** J. E. Stone, A.-P. Hynninen, J. C. Phillips, K. Schulten. International Workshop on OpenPOWER for HPC (IWOPH'16), LNCS 9945, pp. 188-206, 2016.

[2] **NAMD goes quantum: An integrative suite for hybrid simulations.** Melo et al., Nature Methods, 2018.

Challenges Adapting Large Software Systems for State-of-the-Art Hardware Platforms

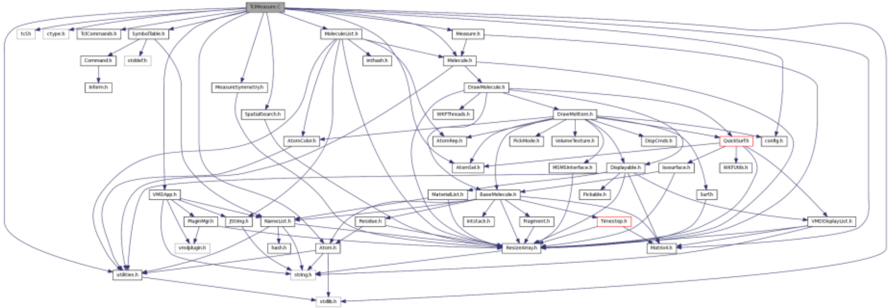
- Initial focus on key computational kernels eventually gives way to the need to optimize an **ocean of less critical routines**, due to observance of **Amdahl's Law**
- Even though these less critical routines might be easily ported to CUDA or similar, the **sheer number of routines often poses a challenge**
- Need a low-cost approach for **getting "some" speedup** out of these second-tier routines
- In many cases, it is completely **sufficient to achieve memory-bandwidth-bound GPU performance with an existing algorithm**

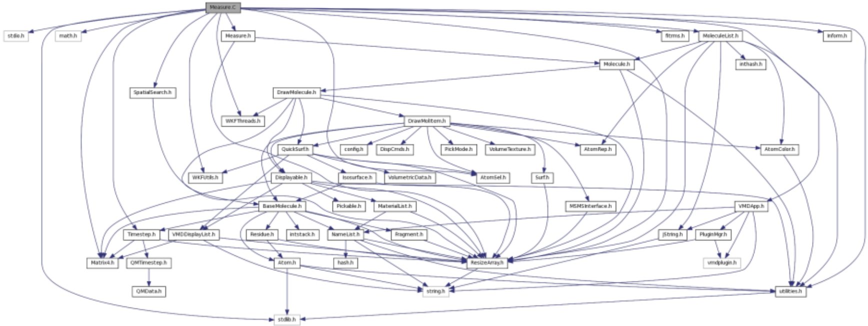
Amdahl's Law and Role of High Abstraction Accelerator Programming Approaches

- Initial partitioning of algorithm(s) between host CPUs and accelerators is typically based on **initial performance balance point**
- **Time passes and accelerators get MUCH faster, and/or compute nodes get denser…**
- Formerly harmless CPU code ends up limiting overall performance!
- Need to address bottlenecks in increasing fraction of code
- **High level programming** tools like **Kokkos** and **OpenACC directives** provide **low cost, low burden**, approach to **improve incrementally** vs. status quo
- **Complementary to lower level approaches** such as CPU intrinsics, CUDA, OpenCL, and they all need to coexist and interoperate very gracefully alongside each other

Example of VMD Module Connectivity

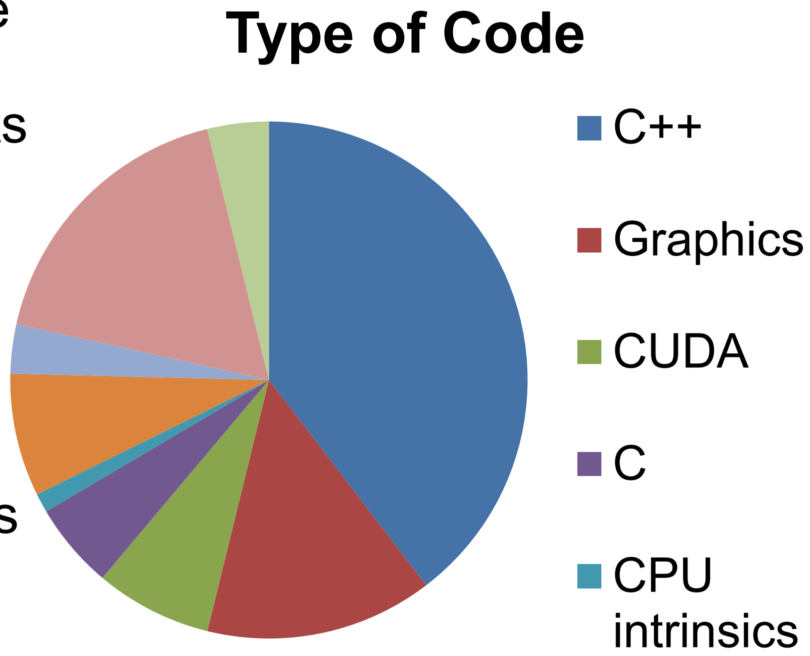
- Early progress focused acceleration efforts on handful of high level analysis ***kernels*** that were the most computationally demanding
- Future hardware requires **pervasive acceleration**
- Top image shows script interface links to top level analytical routines
- Bottom image shows links among subset of data analytics algorithms to **leaf-node functions**



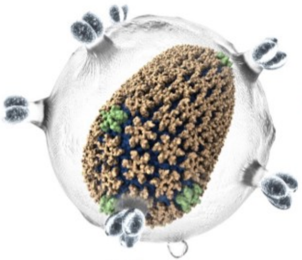


VMD Software Decomposition

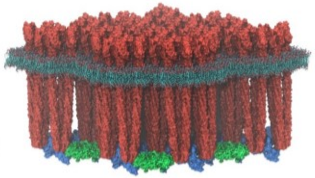
- Computational code is 50% of VMD core
- Hand-written accelerator + vectorized code (CUDA + CPU intrinsics) represents only **14% of core computational code**
  - **20,000 lines of CUDA**
  - **3,100 lines of intrinsics**
- Percent coverage of leaf-node analytical functions is lower yet
- Need to evolve VMD toward high coverage of performance-critical analysis code with fine-grained parallelism on accelerators and vectorization



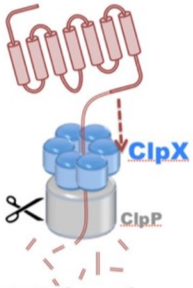
Petascale Simulations Driving NAMD/VMD Development



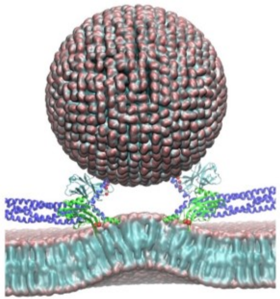
Viral Infection



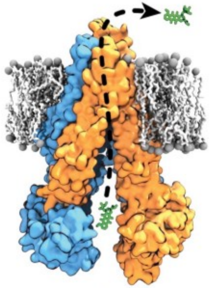
Symbiont Bacteria



Molecular Motors



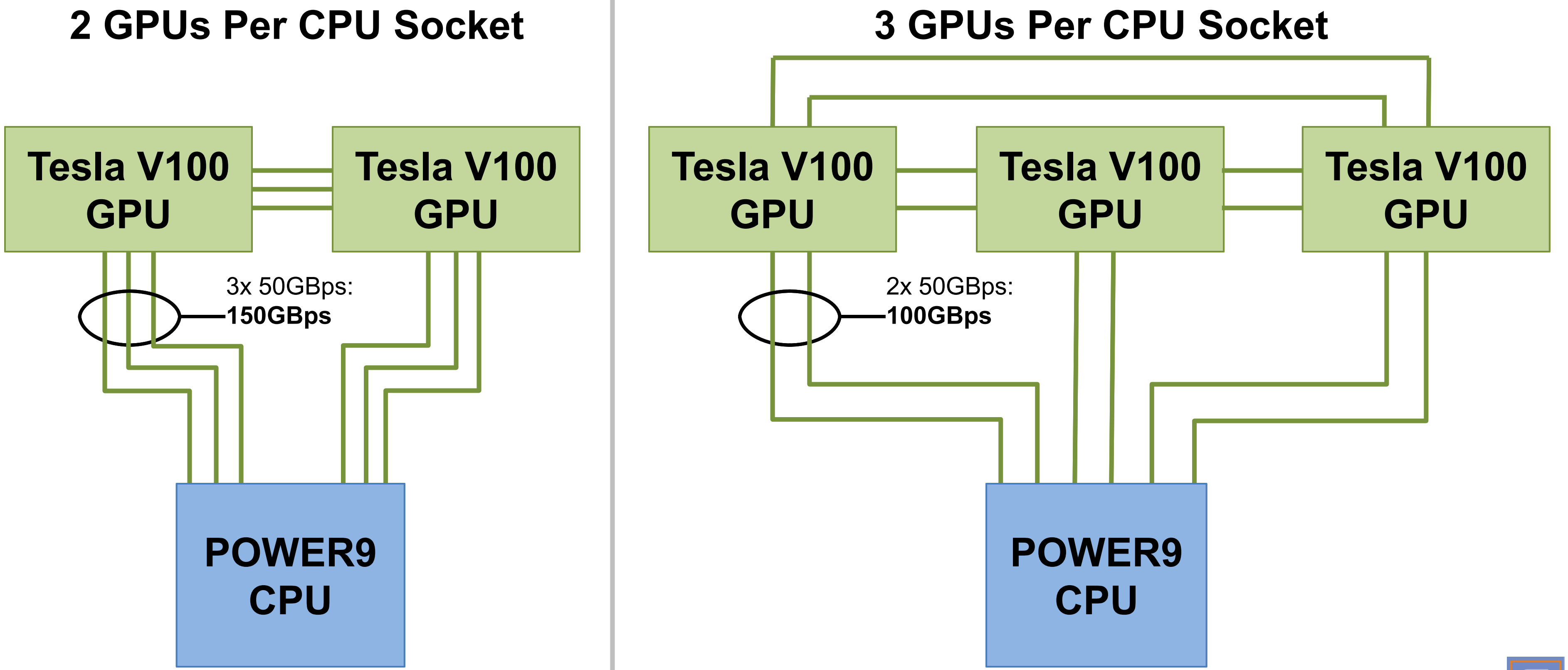
Neurons and Synapses



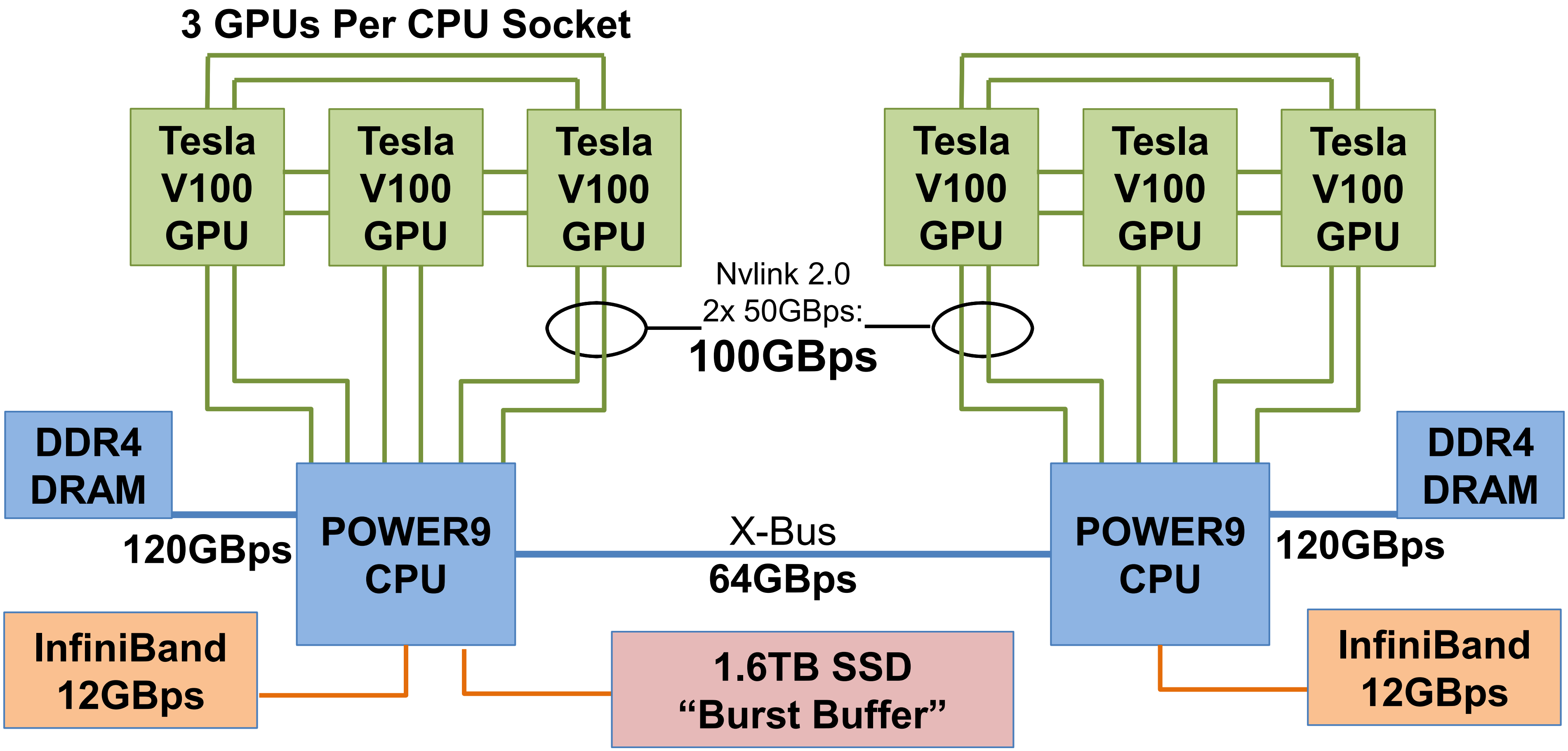
Membrane Transporters

NCSA ORNL	Blue Waters (4,228 XK7 nodes) Titan (18,688 XK7 nodes)	AMD Opteron + K20X Kepler GPU	**16 CPU cores / GPU**
TACC	Stampede 2 (4200 KNL nodes, 1736 Skylake nodes)	Intel Knights Landing Intel Xeon Skylake	68 CPU cores 48 CPU cores
ORNL	**Summit (~4600 nodes)**	**2 IBM Power9 + 6 Tesla V100 GPUs**	**7 CPU cores / GPU**

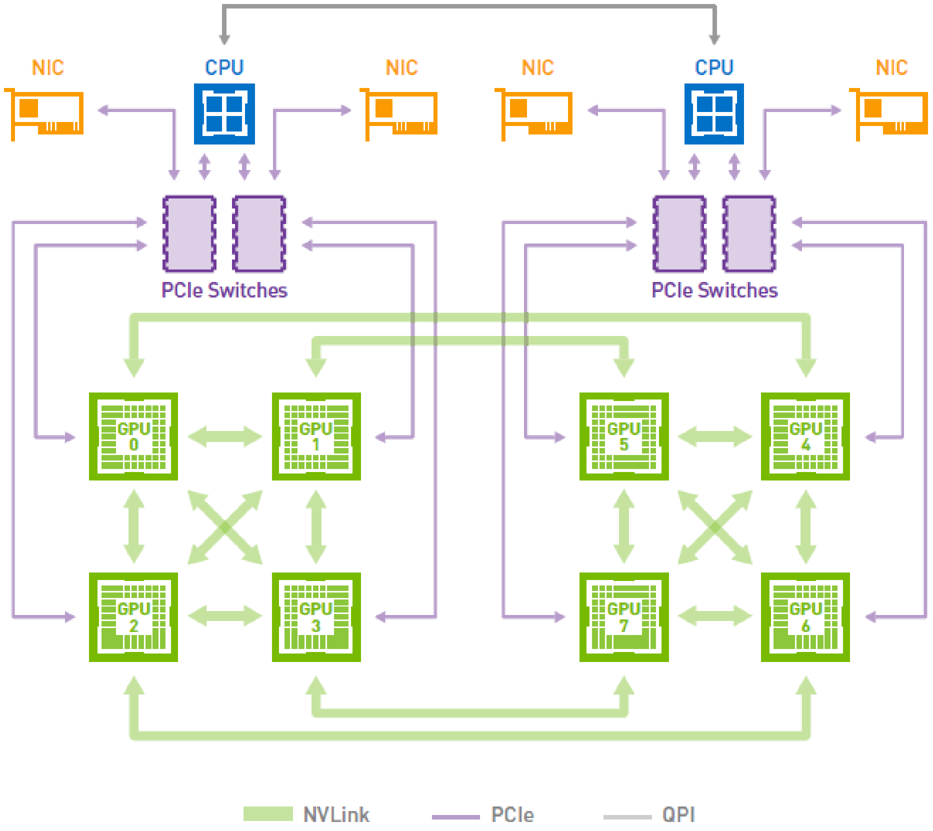
IBM AC922 NVLink 2.0 Topologies



IBM AC922 Summit Node



NVIDIA DGX-1

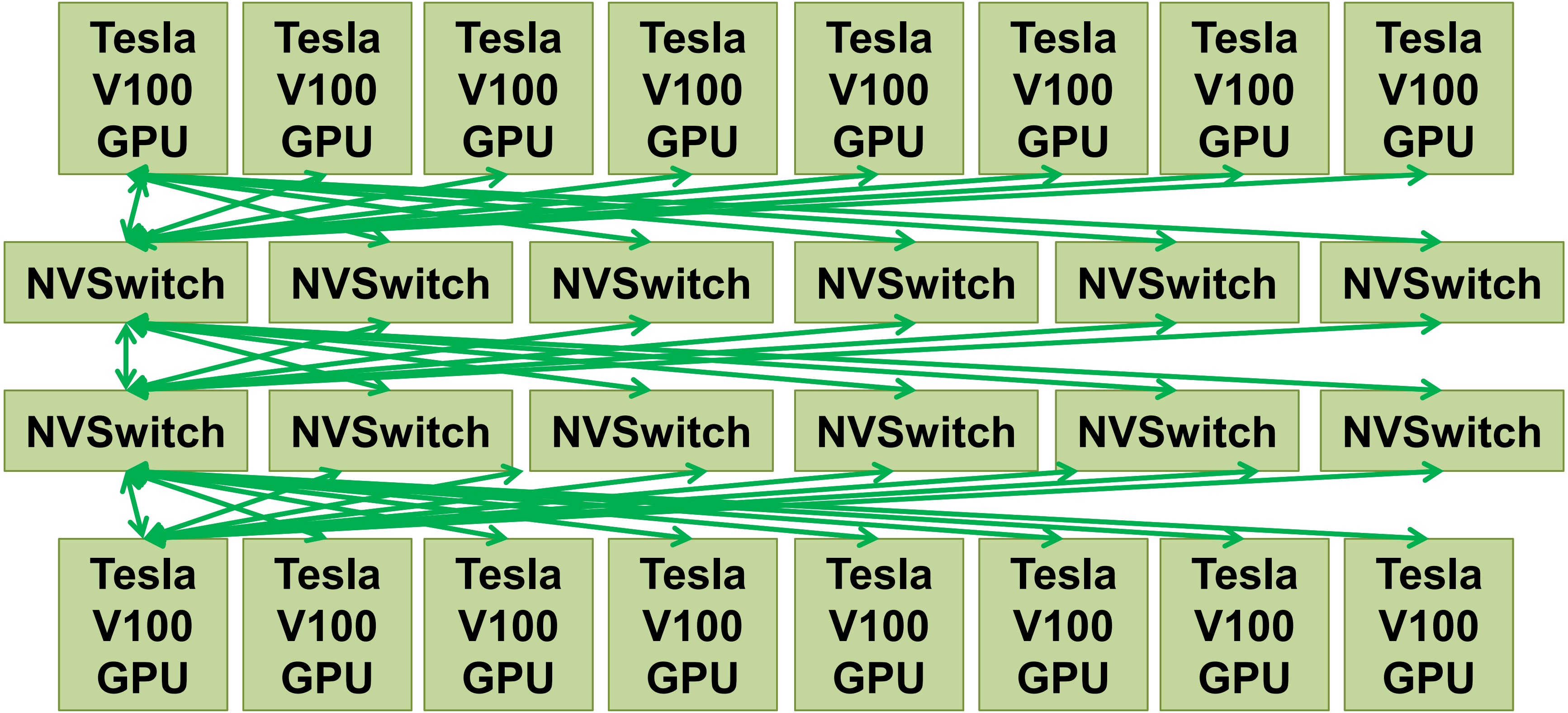


NIC
CPU
NIC
NIC
CPU
NIC
PCIe Switches
PCIe Switches
GPU 0
GPU 1
GPU 5
GPU 4
GPU 2
GPU 3
GPU 7
GPU 6
NVLink
PCIe
QPI

NVIDIA DGX-2

16x 32GB Tesla V100 GPUs w/ 300GB/s NVLink, fully switched

512GB HBM2 RAM w/ **2.4TB/s Bisection Bandwidth, 2 PFLOPS**



Opportunities and Challenges Posed by DGX-2-Like System Designs

- CPUs “oversubscribed” by GPUs
- Unfavorable for algorithm designs that perform “siloed” GPU calculations followed by reductions
- GPU algorithms must dis-involve CPUs to greatest possible extent
- Fully-switched NVLink-connected memory systems permit fine-grained multi-GPU algorithms via direct peer memory load/stores
- Throughput oriented GPU algorithms can hide both local and remote memory latencies gracefully
- Use atomic operations where needed during kernel execution rather than bulk-synchronization and reduction ex post facto
- New levels of algorithm sophistication are possible, but not yet well supported by existing high level programming abstractions

Potential Hardware Evolution

Think of ORNL Summit node and DGX-2 as "entry points" to potential future possibilities…

Questions and observations:

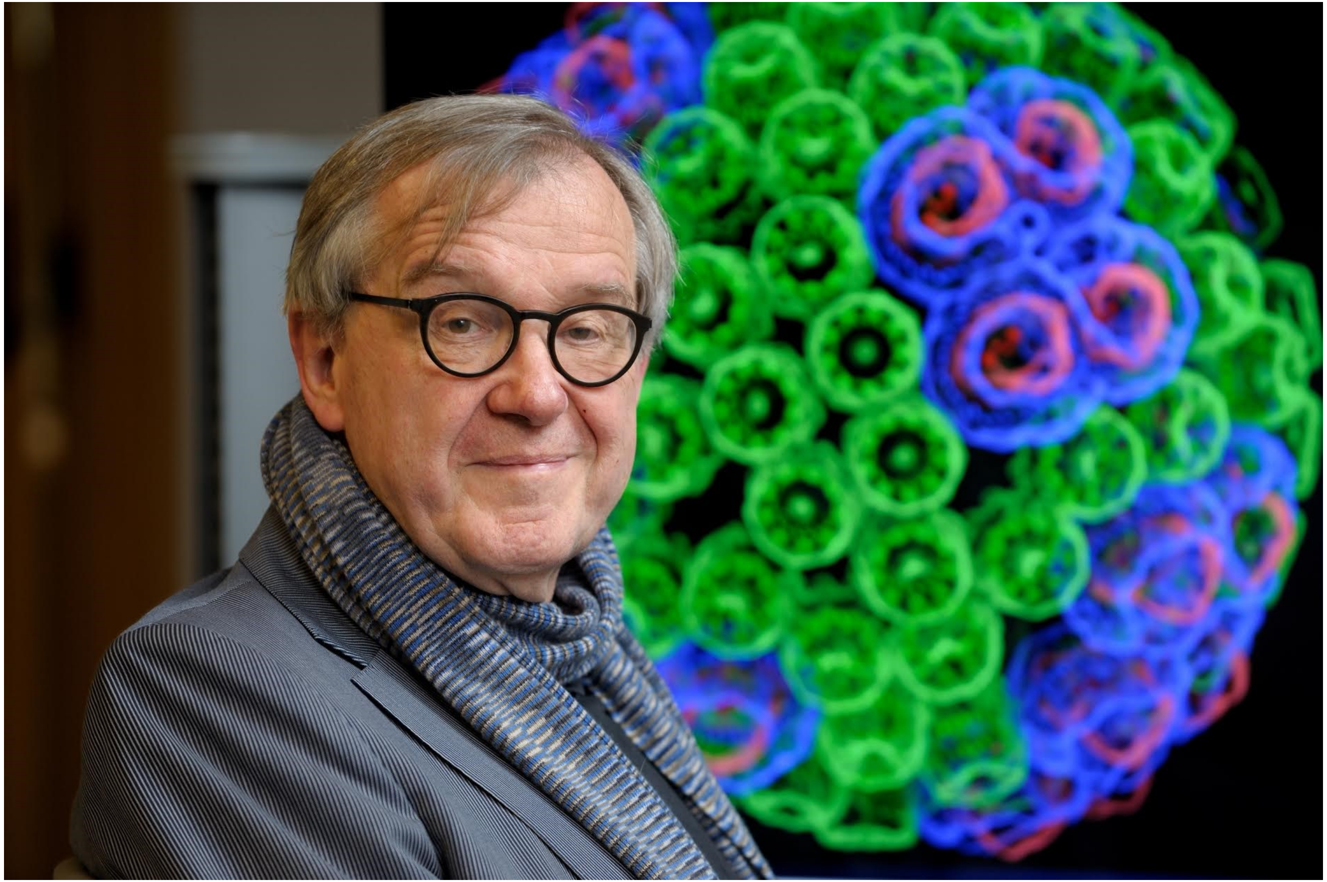
- Would the need for ongoing growth in memory bandwidth among tightly connected accelerators w/ HBM **predict even denser nodes?**
  - **Leadership systems use 6-GPU nodes now. How many in 2022 or thereafter?**
  - **Will future DGX-2 like hardware permit multi-node NVLink-connected load/store memory access?**
- As accelerated systems advance, will directives and other high level programming approaches **encompass peer-to-peer accelerator operations better**?
- Future programming approaches:
  - Support both **existing tightly-coupled Summit- and DGX-2 type systems**, and **future descendants** that may implement multi-node shared memory via switched NVLink-based fabrics.
  - Need to make it much easier to program complex collective operations, reductions, fine-grained distributed-shared-memory data structures among multiple accelerators

Other Challenges and Needs

- **I/O performance growth has largely stalled**, at least as compared to compute performance, **start trading compute for reduced I/O**
- **Scientific reproducibility** benefits from tools that produce completely deterministic output
- Hard to engineer complete determinism in light of floating point non-associativity, increasing asynchrony, and massive fine-grained parallelism, greater need for techniques like **fixed-point summation for greater determinism**
- State-of-the-art hardware to provides a multitude of **bespoke numeric representations and arithmetic** capabilities that make it challenging for algorithm designers and software implementors to have a complete grasp of their achieved end-to-end numerical accuracy and precision
- Software development tools should help algorithm designers safely exploit the custom numerical representations and arithmetic operations on next-generation hardware

Acknowledgements

- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- NVIDIA CUDA and OptiX teams
- Funding:
  - NIH support: P41GM104601
  - ORNL Center for Advanced Application Readiness (CAAR)
  - IBM POWER team, IBM Poughkeepsie Customer Center
  - NVIDIA CUDA, OptiX, Devtech teams
  - UIUC/IBM C3SR
  - NCSA ISL



"When I was a young man, my goal was to look with mathematical and computational means at the inside of cells, one atom at a time, to decipher how living systems work. That is what I strived for and I never deflected from this goal." – Klaus Schulten