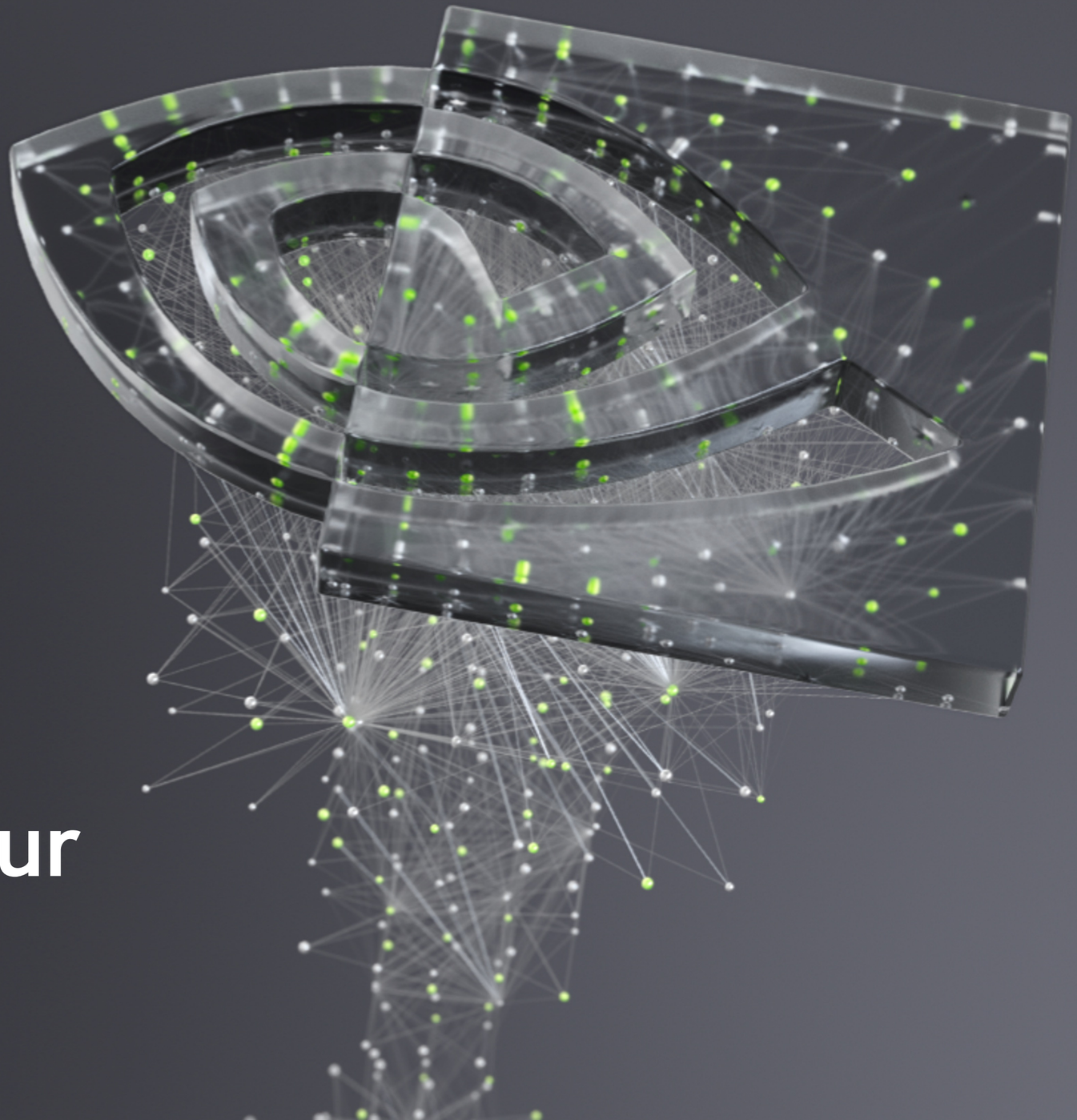


NWChem Guided Tour

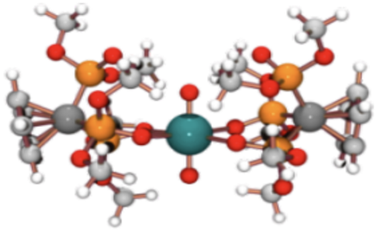
Jeff Hammond
NVIDIA HPC Group

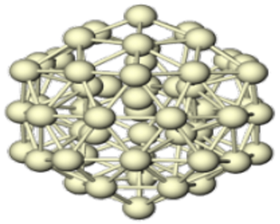
Outline

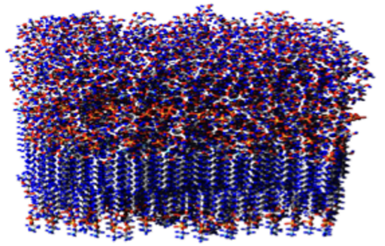
- What is NWChem?
- NWChem on ARM history
- Experiments with different configurations on Ampere Altra Q80-30
- Comparison of Ampere Altra Q80-30 with Intel Xeon 6148
- Conclusions and future work

NWCHEM

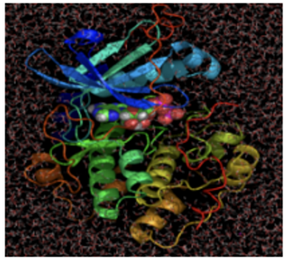
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

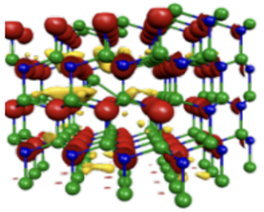


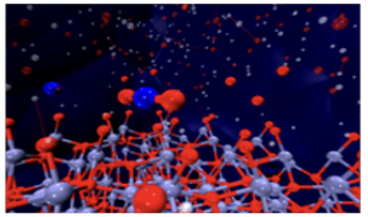




- Suite of computational chemistry functionality:
  - From classical MD to AO DFT … MP2 to CCSD…
  - Multi-scale: QM/MM, embedding
  - NWPW: AIMD code based on MPI
- Massively parallel design for HPC systems circa ~2000.
  - Process-based parallelism in Global Arrays
  - Modular design to enable reuse of integrals, SCF, etc.
  - Object-oriented design in legacy Fortran
  - Threading from BLAS/LAPACK (until recently)







https://www.nersc.gov/assets/Uploads/Hammond-NERSC-OpenMP-August-2019-1.pdf

The world into which NWChem was born

1992

- POSIX released in 1988 as IEEE POSICE and ISO POSIX in 1990.
- Fortran 90 was released as an ISO (ANSI) standard in 1991 (1992).
- MPI 1.0 was released at Supercomputing in November of 1993.
- First production version of the Linux kernel was released in In March 1994.
- OpenMP for Fortran 1.0 published October 1997.
- C++ first released as an ISO standard in 1998.

NWChem 3.2.1 released October 1998 (oldest release I can see in Git)

On the hardware side…

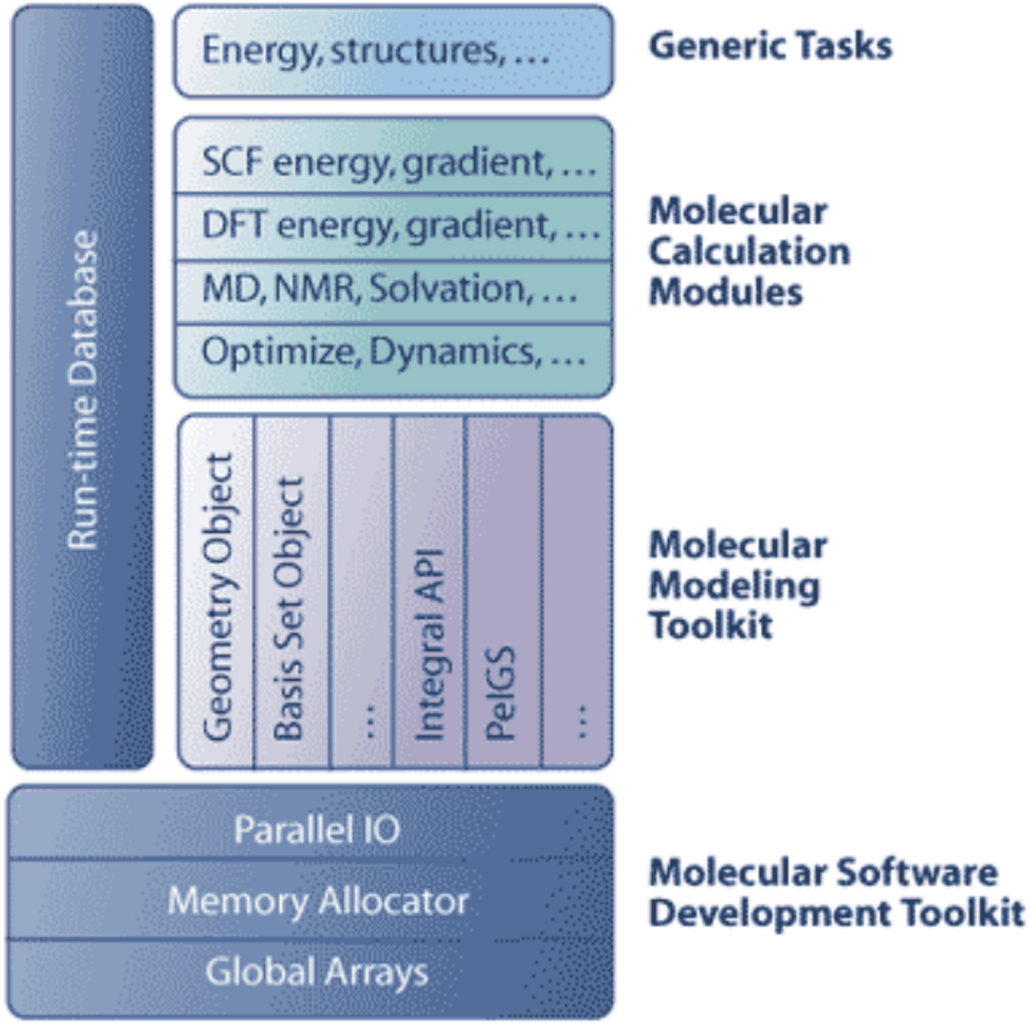
1992-1996 was rough on vendors

- Beowulf tools introduced in 1994
- Thinking Machines filed for bankruptcy in 1994
- Cray Computer Corporation (spinoff) went bankrupt in 1995
- Cray Research Inc. bought by SGI in 1996
- MasPar exited the hardware business in 1996
- Meiko folded into Quadrics in 1996

Fujitsu, HP(E) and IBM may be the only continuously operating HPC system vendors over the lifetime of NWChem

NWChem software architecture

- System interfaces
  - basic I/O (e.g. command-line arguments)
  - timers and many other OS wrappers
- Memory allocation
  - fast (no syscalls, stack pattern)
  - shared-memory (IPC)
  - pinned for network (if necessary)
- Global Arrays and TCGMSG
  - process management (job launch)
  - one-sided communication
  - collective operations
  - load-balancer



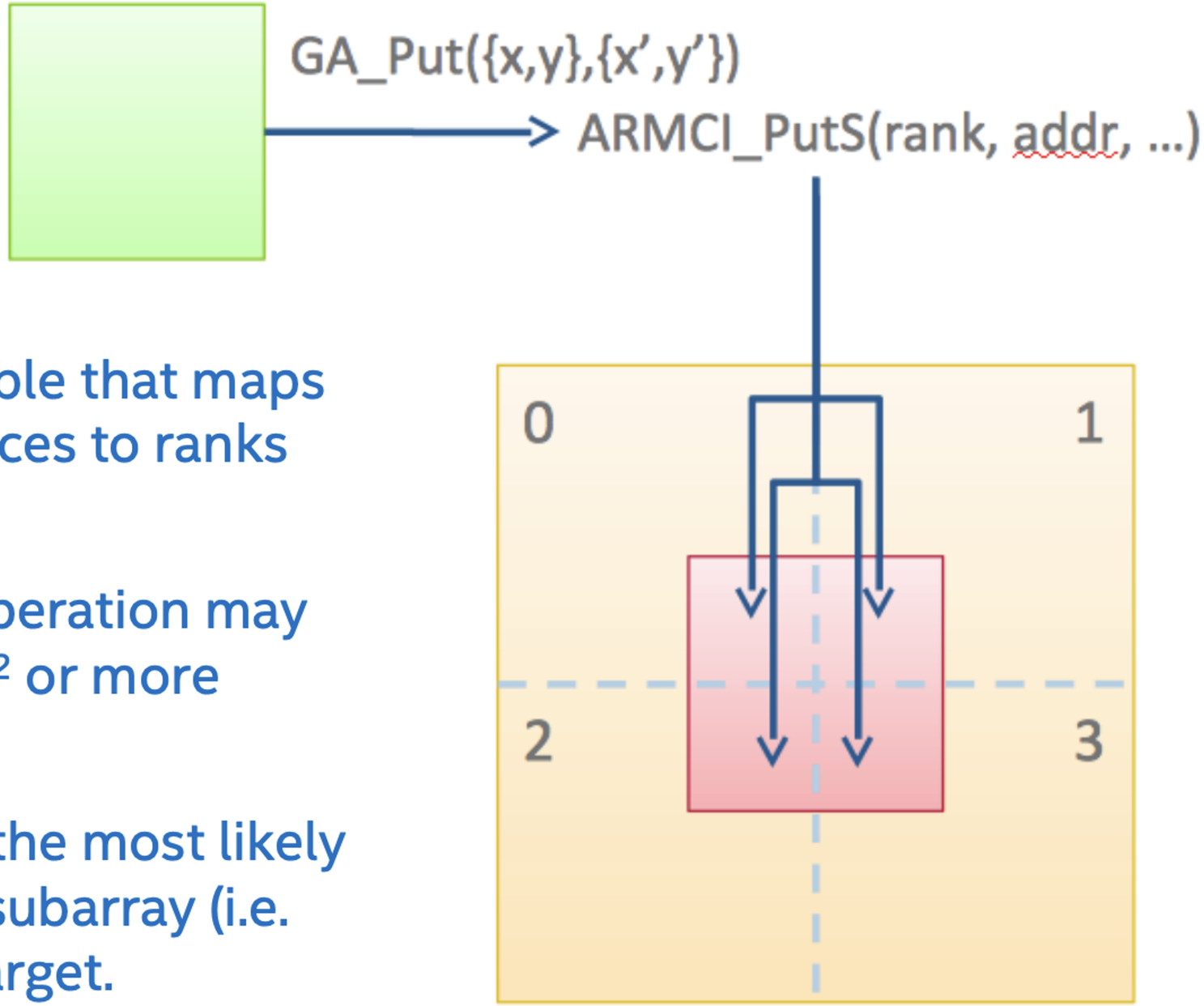
https://icl.utk.edu/ctwatch/quarterly/articles/2006/05/nwchem-development-of-a-modern-quantum-chemistry-program/

Global Arrays

- Distributed array abstraction layer that supports communication primitives and numerical linear algebra methods.
- **User observes** one-sided communication with strong asynchronous progress.
- ARMCI is the one-sided communication abstraction layer inside of GA…

```
double precision buf(100,100)
ga_initialize()
ga_create(MT_DBL, 100, 100, ‘matrix’, 1, 1, g_m)
ga_create(MT_INT, 1, 1, ‘counter', 1, 1, g_c)
ga_zero(g_m); ga_zero(g_c); ga_sync()
buf = 100.0
ga_put(g_m, 1, 100, 1, 100, buf, 100)
buf = 1.0
ga_acc(g_m, 1, 100, 1, 100, buf, 100, 1.0)
ga_get(g_m, 1, 100, 1, 100, buf, 100)
! buf = 101.0 (if nproc=1)
do j=1,100 k = ga_read_inc(g_c, 1, 1, 1)
! k = 99 (if nproc=1)
ga_destroy(g_m); ga_destroy(g_c)
ga_terminate()
```

GA-to-ARMCI

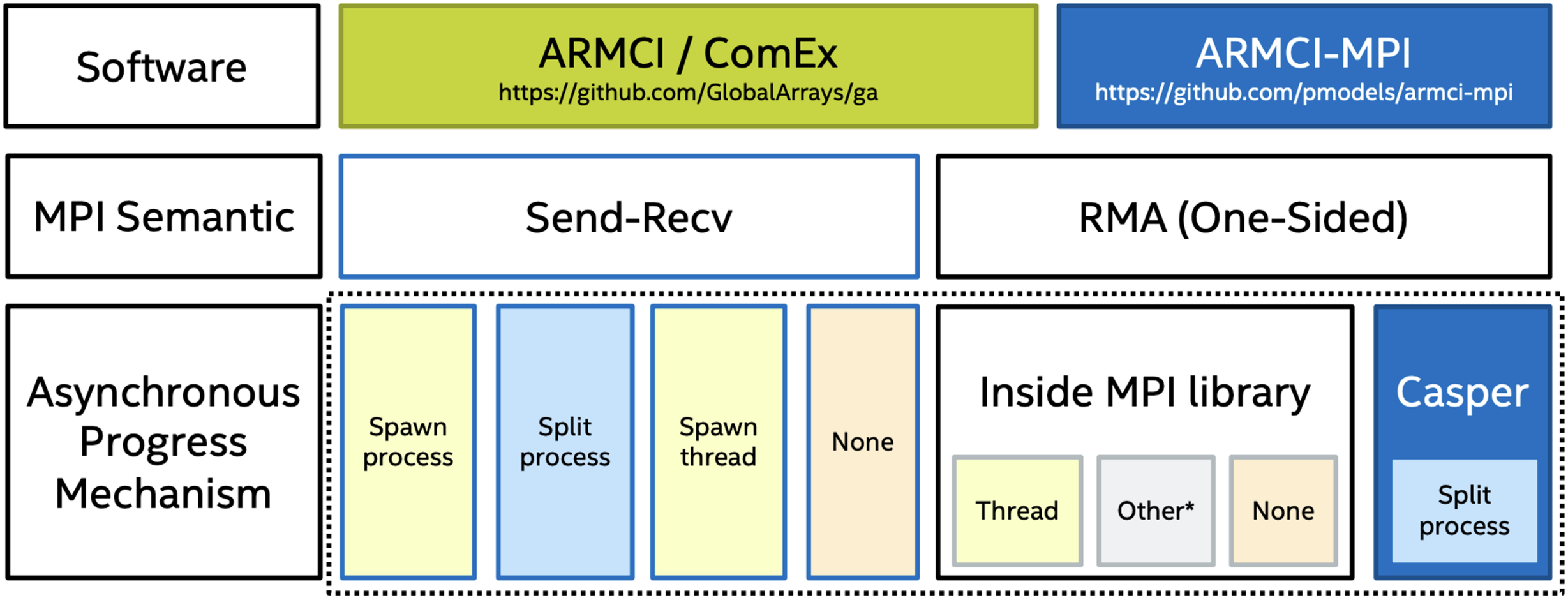


- GA maintains a translation table that maps array handles and global indices to ranks and base pointers.
- A single N-dimensional GA operation may need to communicate with N^2 or more remote ranks.
- For N>1 dimensional arrays, the most likely scenario is a noncontiguous subarray (i.e. vector of vectors) for every target.

Why do we need MPI-based ARMCI?

- All HPC systems support MPI and all actively maintained implementations of MPI support MPI-3, including RMA. No reason to not take advantage of this.
  - DOE procurements now require one-sided and multithreaded MPI…
- Low-level networking APIs vary in usability. Reimplementing page-registration cache and flow-control is painful. Interoperability with MPI (required by ScaLAPACK) is not assured. Some HPC systems do not document or even expose low-level networking interface.
- Allows the computer scientists to focus on the more pressing problems like heterogeneous execution.

Mapping ARMCI to MPI

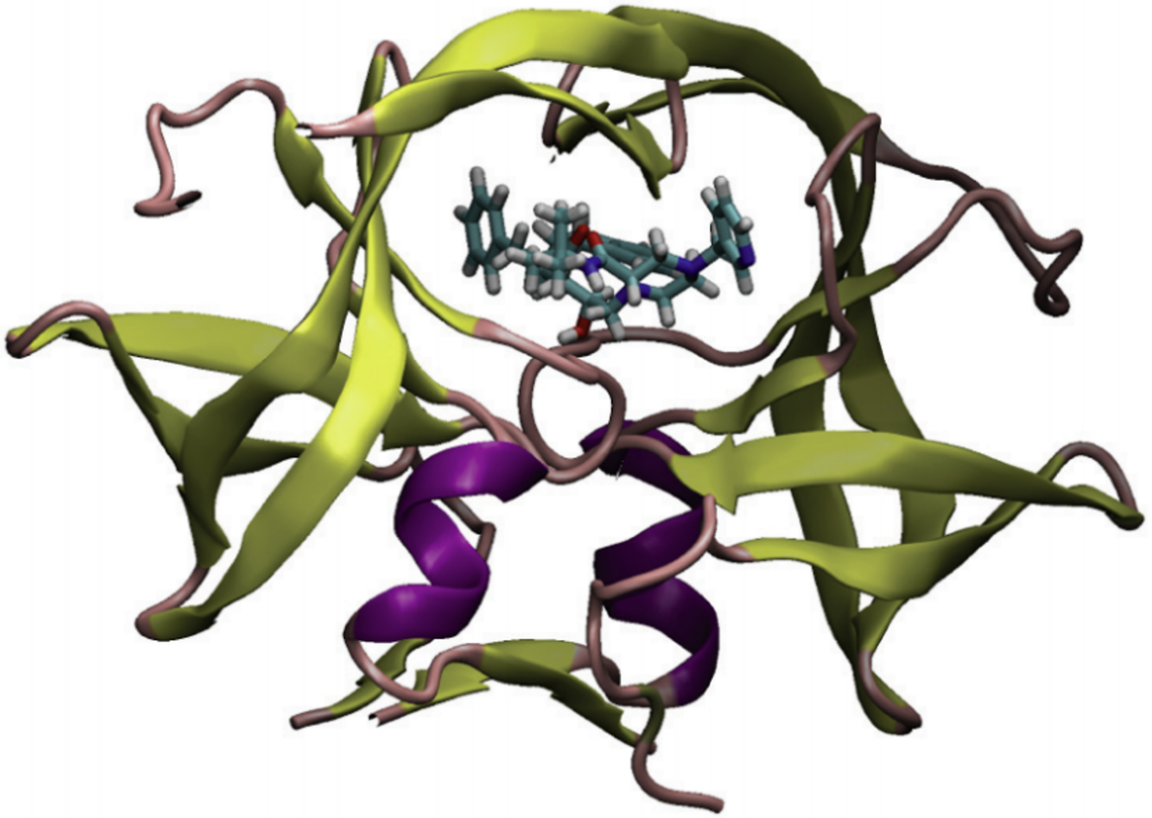


https://github.com/pmodels/casper

* Interrupts on Blue Gene, hardware offload mechanism (e.g. smart NIC).

NWChem Evaluation

- 1hsg_28 benchmark system
- 122 atoms, 1159 basis functions
- H,C,N,O w/ cc-pVDZ basis set
- Semidirect algorithm
- Closed shell (RHF)



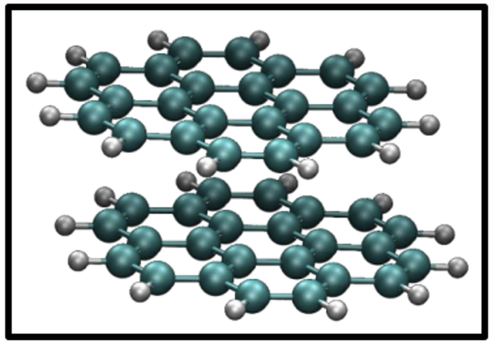
E. Chow, X. Liu, S. Misra, M. Dukhan, M. Smelyanskiy, J. R. Hammond, Y. Du, X.-K. Liao and P. Dubey. *International Journal of High Performance Computing Applications*. "Scaling up Hartree–Fock Calculations on Tianhe-2." http://dx.doi.org/10.1177/1094342015592960
(GTFock used GA/ARMCI-MPI and MPICH-Glex for these petascale runs.)

NWChem SCF performance (new)

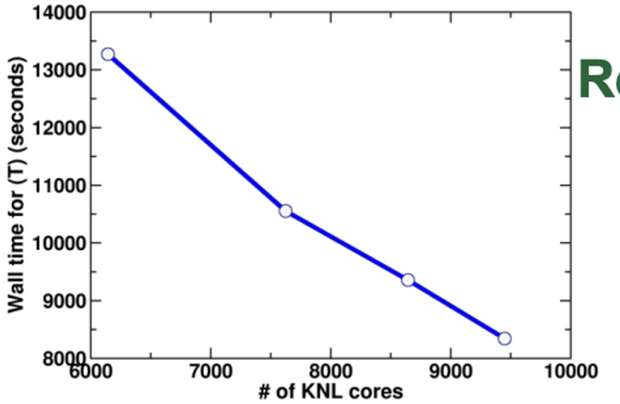
NWChem 6.3/ARMCI-MPI3/Casper			NWChem Dev/ARMCI-MPIPR (built by NERSC, Sept. 2015)		
iter	energy	time	iter	energy	time
1	-2830.4366669990	**69.3**	1	-2830.4366669999	**61.4**
2	-2831.3734512499	77.1	2	-2831.3734512509	69.3
3	-2831.5712604368	84.6	3	-2831.5713109521	77.8
4	-2831.5727804428	93.0	4	-2831.5727856618	87.3
5	-2831.5727956927	107.3	5	-2831.5727956974	103.9
6	-2831.5727956977	**128.0**	6	-2831.5727956980	**125.7**

Running on 8 nodes with 24 ppn. **Both** use 2 ppn for comm.

Scaling of the SPEC CCSD(T) Library on the Full Partition of the KNL Nodes of the Cori Supercomputer at NERSC



Coronene Dimer ($C_{24}H_{12}$)$_2$



Wall time for (T) vs. number of KNL cores

Cori Supercomputer @ NERSC

Team: Aprà, Hammond (Intel), Daily, Palmer, Xantheas

Scientific Achievement

Calculation of the binding energy of the coronene dimer, an archetypal system for graphene

Significance and Impact

Ability to obtain accurate interaction energies of large systems; largest to date CCSD(T) calculation (9.14 PFLOPs) used 538,650 Knight's Landing (KNL) cores (9,450 nodes; 57/68 cores per node)

Research Details

- **216 electrons / 1,776 basis functions** (cc-pVTZ basis set)
- OpenMP for multi-threading in CCSD and CCSD(T)
- Checkpoint restart capability in CCSD
- Improved inter-node parallelization of the (T) correction on the KNL nodes using the Global Arrays (GA) tool

# of KNL nodes/cores	(T) kernel	
	Wall time (sec)	PFLOPs
7,624 / 434,568	10,553	7.65
8,644 / 492,708	9,357	8.13
9,450 / 538,650 (97.5% of full partition)	**8,344**	**9.14**

Work was performed at Pacific Northwest National Laboratory under a NERSC Initiative for Scientific Exploration (NISE) and BES allocation awards

NWChem Software Requirements

- CPU: x86_64, AArch64, PowerPC for sure, others should work.
  - GPU: NVIDIA via CUDA and OpenACC.
  - GPU: OpenMP 4 offload.
- OS: All Linux-like(Linux, MacOS, BSD).
- Compilers: GCC, Intel, NVHPC (PGI) tested extensively. Others should work.
- Math Libraries: Netlib, MKL, OpenBLAS, BLIS, ARMPL, Apple Accelerate, etc.
- Communication: MPI is assumed.
  - MPI-PR works with all known MPI libraries.
  - ARMCI-MPI unsafe with Open-MPI. MPICH-based (Intel, MVAPICH2) should be fine.

Performance Variables

- Compiler optimization
  - Primarily atomic integral computations
- Math Libraries
  - BLAS
  - LAPACK
  - math.h/libm
- GA/ARMCI/MPI
  - ARMCI vs ARMCI-MPI designs
  - MPICH vs Open-MPI RMA designs
  - Interprocess I/O scalability in Linux

Experiments

Dual-socket Ampere Altra Q80-30 (80 cores, 3.0 GHz max)

- Compilers and Math Libraries
  - GCC 8.3.1 and OpenBLAS latest
  - NVHPC 21.5 and included OpenBLAS
- MPI
  - MPICH latest
  - Open-MPI latest
- GA/ARMCI
  - MPI-PR
  - ARMCI-MPI

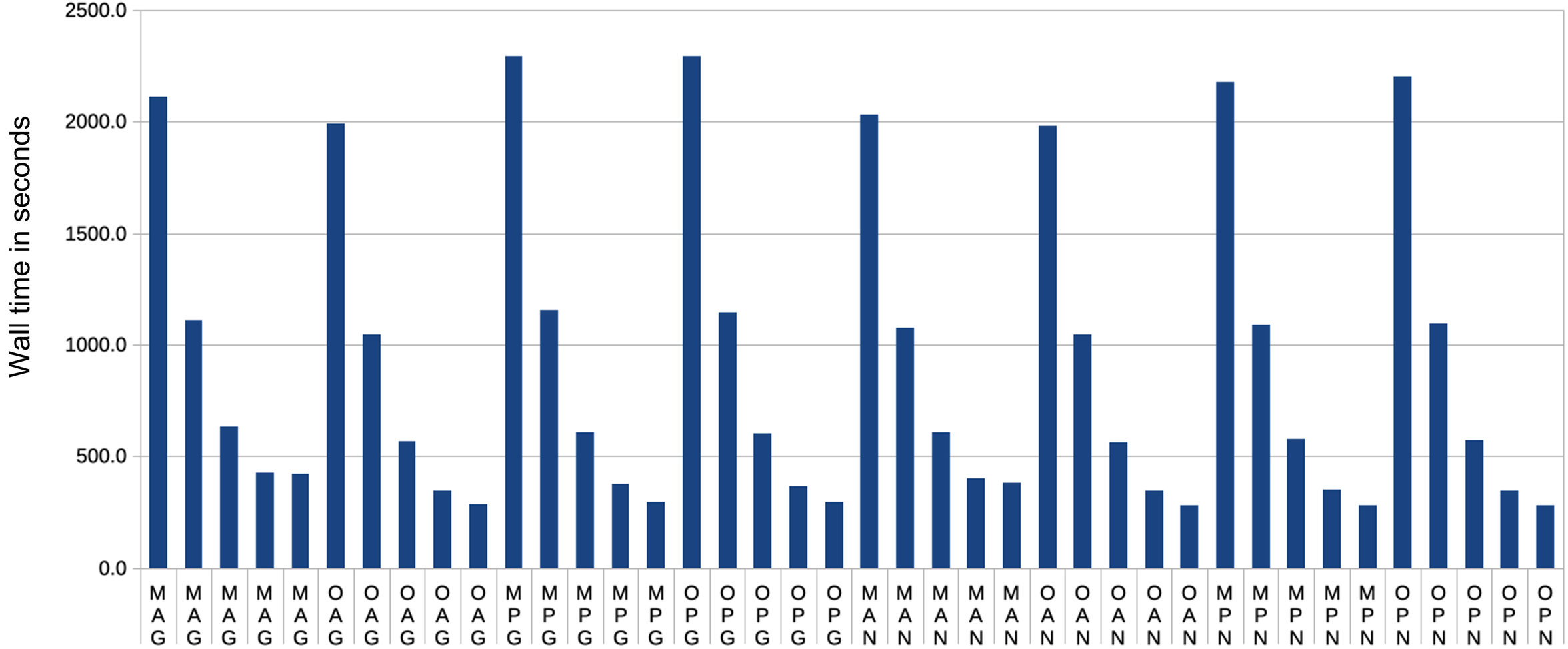
B3LYP/cc-pVTZ is a common simulation type for molecular simulations that is dominated by atomic integral computations, exchange-correlation quadrature, and a small amount of dense linear algebra.

These simulations are compute-bound unless something goes wrong.

https://github.com/NWChem/input-generator

$(H_2O)_{21}$ B3LYP/cc-pVTZ

Descending bars are 10, 20, 40, 80, 160 processes for each binary.



M=MPICH
O=Open-MPI

A=ARMCI-MPI
P=MPI-PR

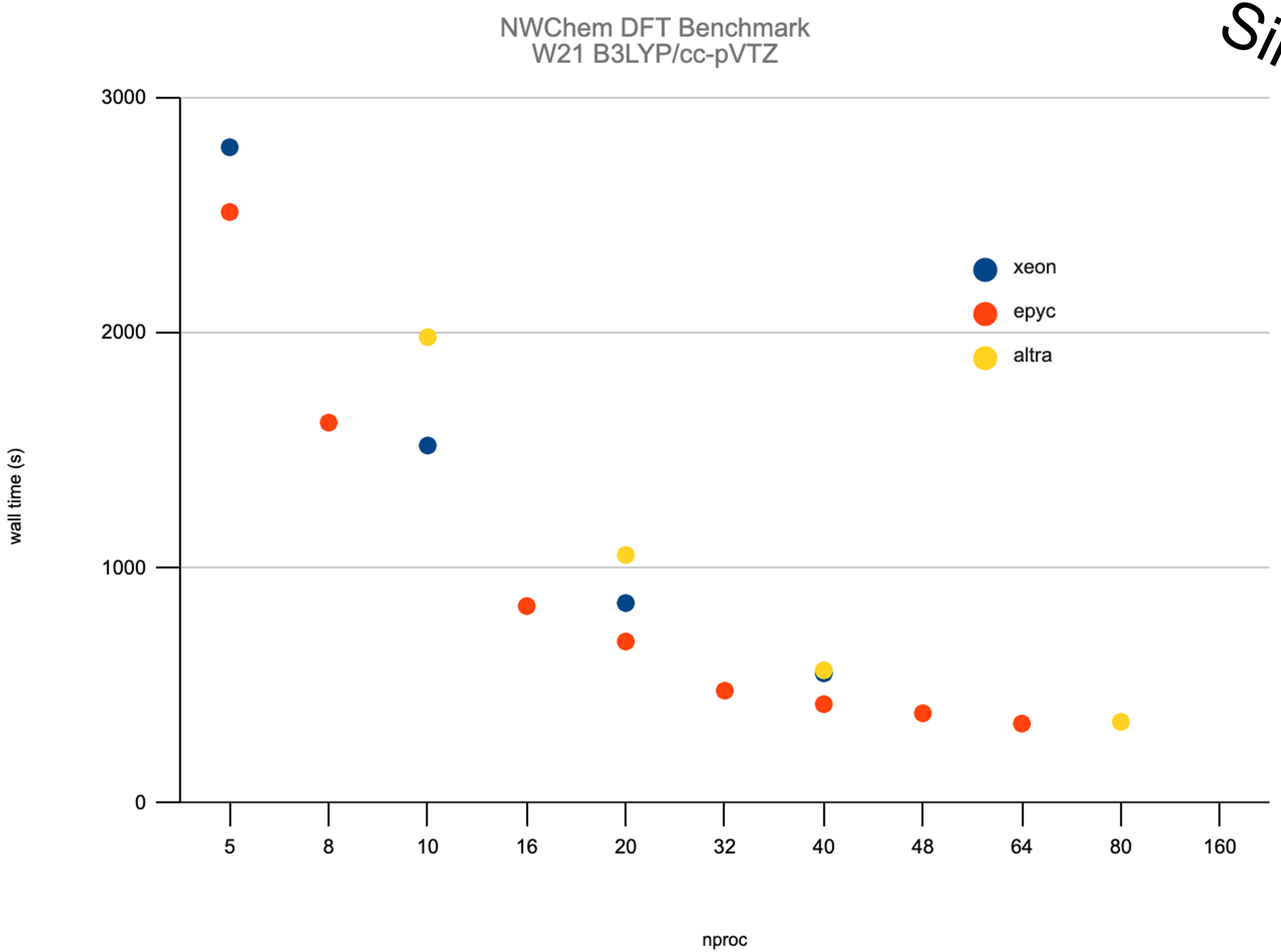
G=GCC
N=NVHPC

NVIDIA

Performance Analysis

- NVHPC binaries are ~2-7% than GCC binaries.
  - This is consistent with the observed impact of compilers in the DFT code with other compilers.
  - GCC 8 may lack important optimizations for the ARM Neoverse N1 core…
- Open-MPI binaries are ~0-20% faster than MPICH binaries.
  - Open-MPI appears to have better shared-memory support in RMA, but we did not explore all the tuning parameters (e.g. MPICH used OFI not UCX).
- ARMCI-MPI versus MPI-PR is ±10%.
  - MPI-PR computes with 1 less core, which matters at low core counts (e.g. 10).
  - MPI-PR will scale better relative to ARMCI-MPI most of the time due to progress.
  - ARMCI-MPI with Casper (not tested here) will also dedicate 1+ cores to progress.

Single Node Scaling



Performance Analysis

- NVHPC binaries are ~2-7% than GCC binaries.
  - This is consistent with the observed impact of compilers in the DFT code with other compilers.
  - GCC 8 may lack important optimizations for the ARM Neoverse N1 core…
- Open-MPI binaries are ~0-20% faster than MPICH binaries.
  - Open-MPI appears to have better shared-memory support in RMA, but we did not explore all the tuning parameters (e.g. MPICH used OFI not UCX).
- ARMCI-MPI versus MPI-PR is ±10%.
  - MPI-PR computes with 1 less core, which matters at low core counts (e.g. 10).
  - MPI-PR will scale better relative to ARMCI-MPI most of the time due to progress.
  - ARMCI-MPI with Casper (not tested here) will also dedicate 1+ cores to progress.

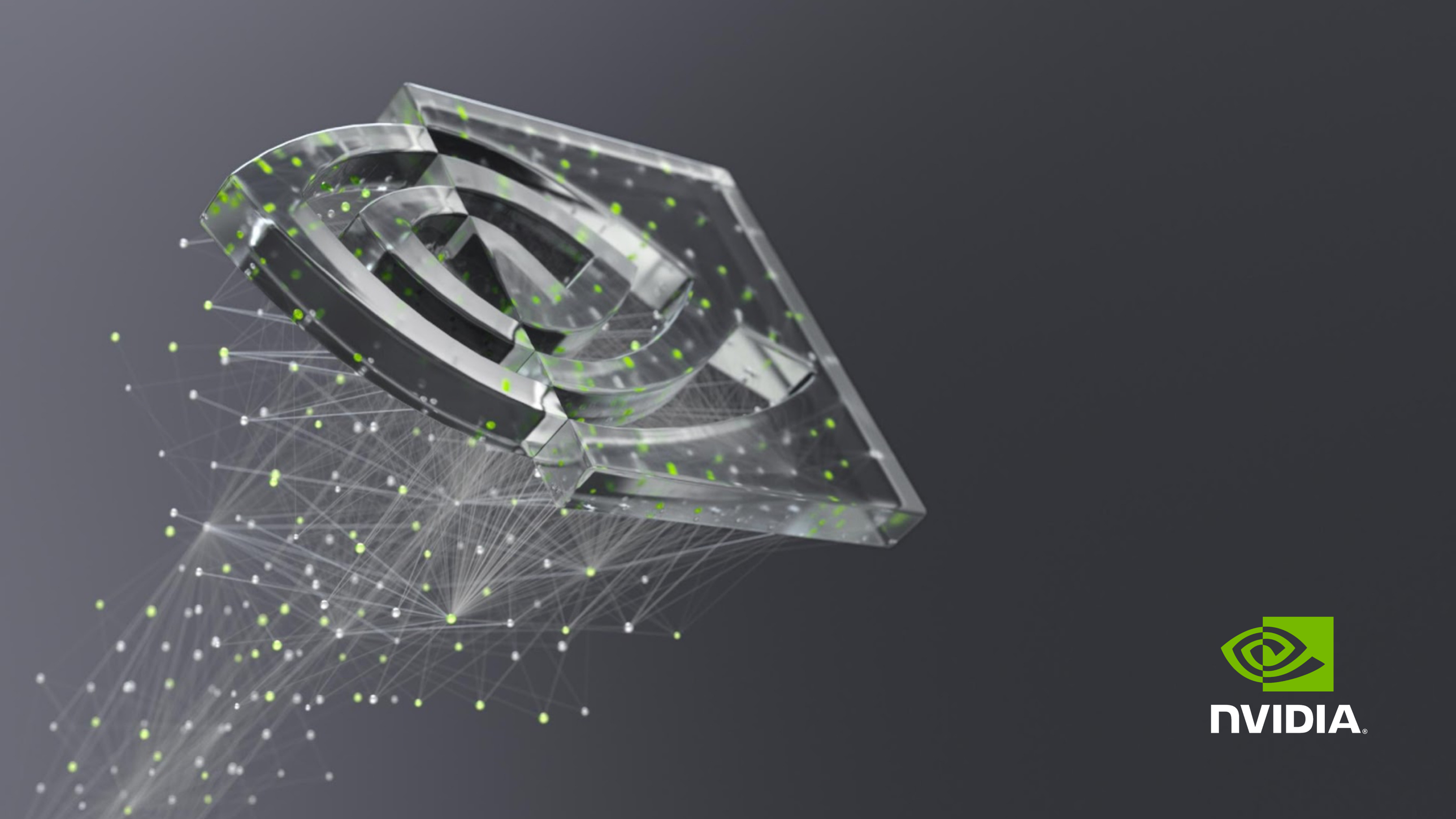
Questions/Comments

https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/2799534081/Getting+Started+with+NWChem+for+ISC22+SCC

Twitter: https://twitter.com/science_dot

Email: jeff_hammond@acm.org

LinkedIn: https://www.linkedin.com/in/jeffhammond/


NVIDIA