BREAKING PERFORMANCE PORTABILITY BOTTLENECKS IN NWCHEM

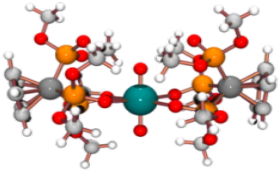
Jeff Hammond
Exascale Co-Design Group
Intel Corporation

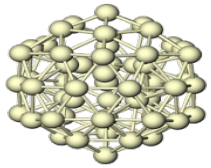
2 April 2019

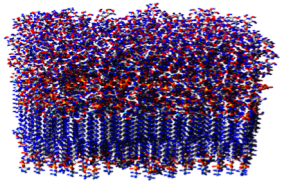
Based on collaboration with:
- Min Si, Jim Dinan, Pavan Balaji
- Karol Kowalski, Edo Apra
- Michael Klemm

NWCHEM

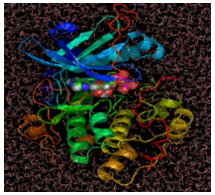
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

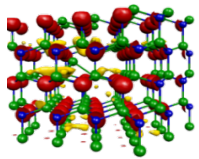


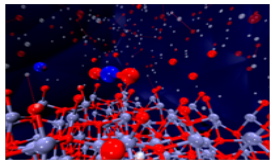




- Suite of computational chemistry functionality:
  - From classical MD to AO DFT … MP2 to CCSD…
  - Multi-scale: QM/MM, embedding
  - NWPW: AIMD code based on MPI
- Massively parallel design for HPC systems circa ~2000.
  - Process-based parallelism in Global Arrays
  - Modular design to enable reuse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







History

Several goals were set for the NWChem software package. These included:

- NWChem would be based on algorithms that:
  - scale to hundreds, if not thousands, of processors
  - use the minimum number of flops consistent with the above.
- NWChem would be affordable to develop, maintain, and extend.
- NWChem would be independent of computer architecture to the maximum extent possible:
  - a hardware abstraction layer would be used to isolate the most of the program from the details of the hardware, but
  - it would be possible to tune the code to some extent to optimize performance on the specific computer being used.

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

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).

See Wikipedia and the introduction to the MPI standard for details.

On the hardware side…

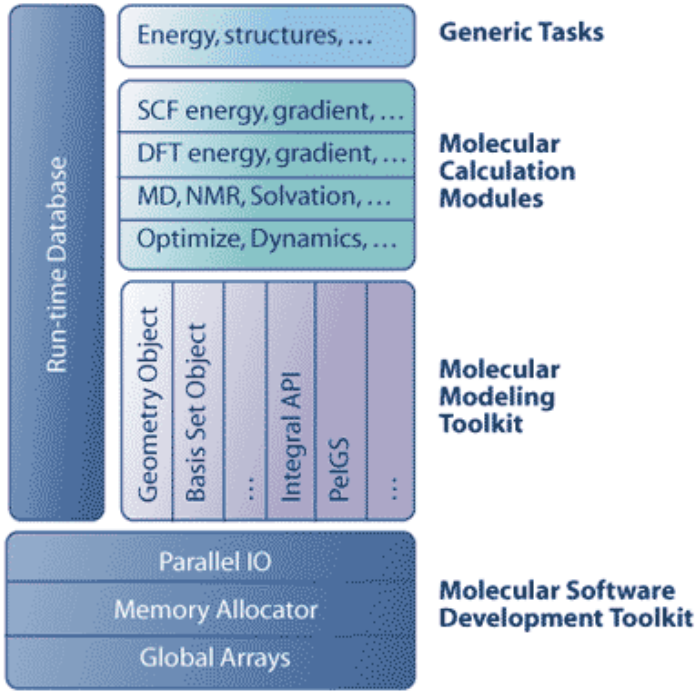
- Beowulf tools introduced in 1994
- Thinking Machines filed for bankruptcy in 1994
- Cray Computer Corporation (spinoff) went bankrupt in 1995
- Cray Research Inc. bought by SGI in 1996
- MasPar exited the hardware business in 1996
- Meiko folded into Quadrics in 1996

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

https://www.microsoft.com/en-us/research/wp-content/uploads/2015/01/MSR-TR-2015-2_Supercomputers-The_Amazing_Race_Bell.pdf

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/

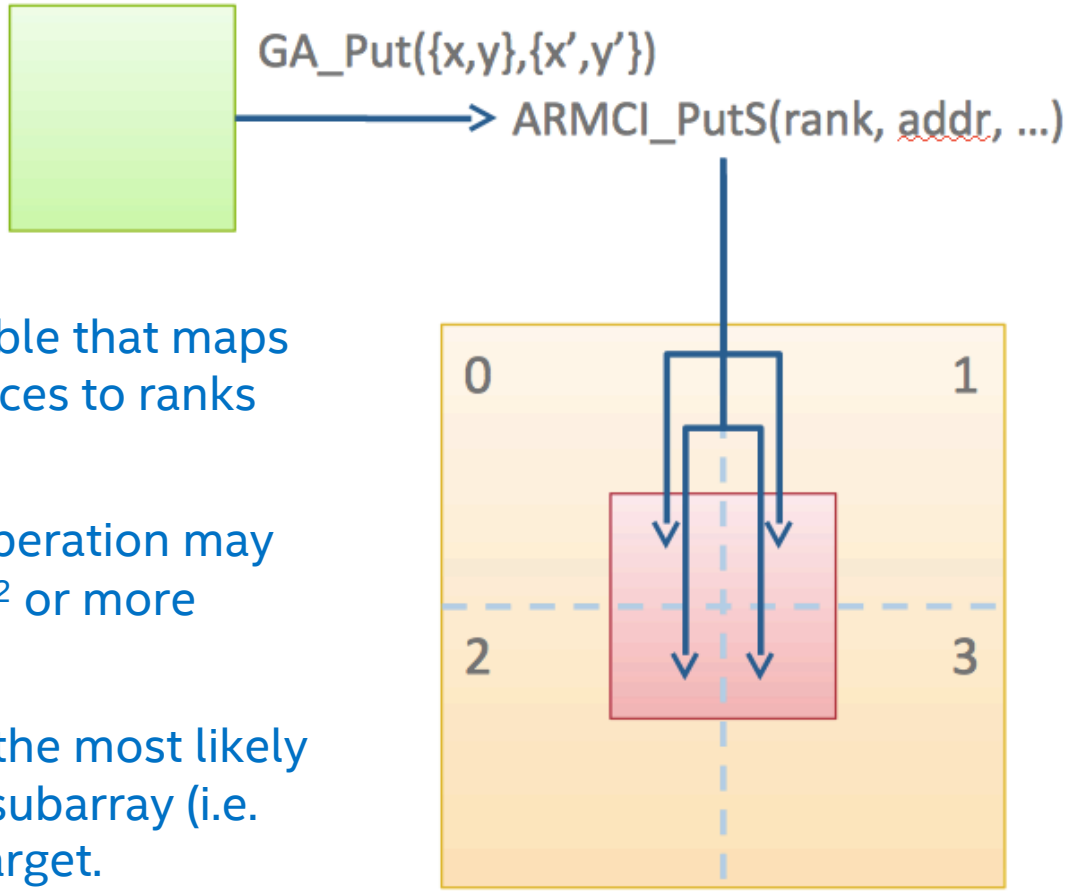
MPI-3

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…

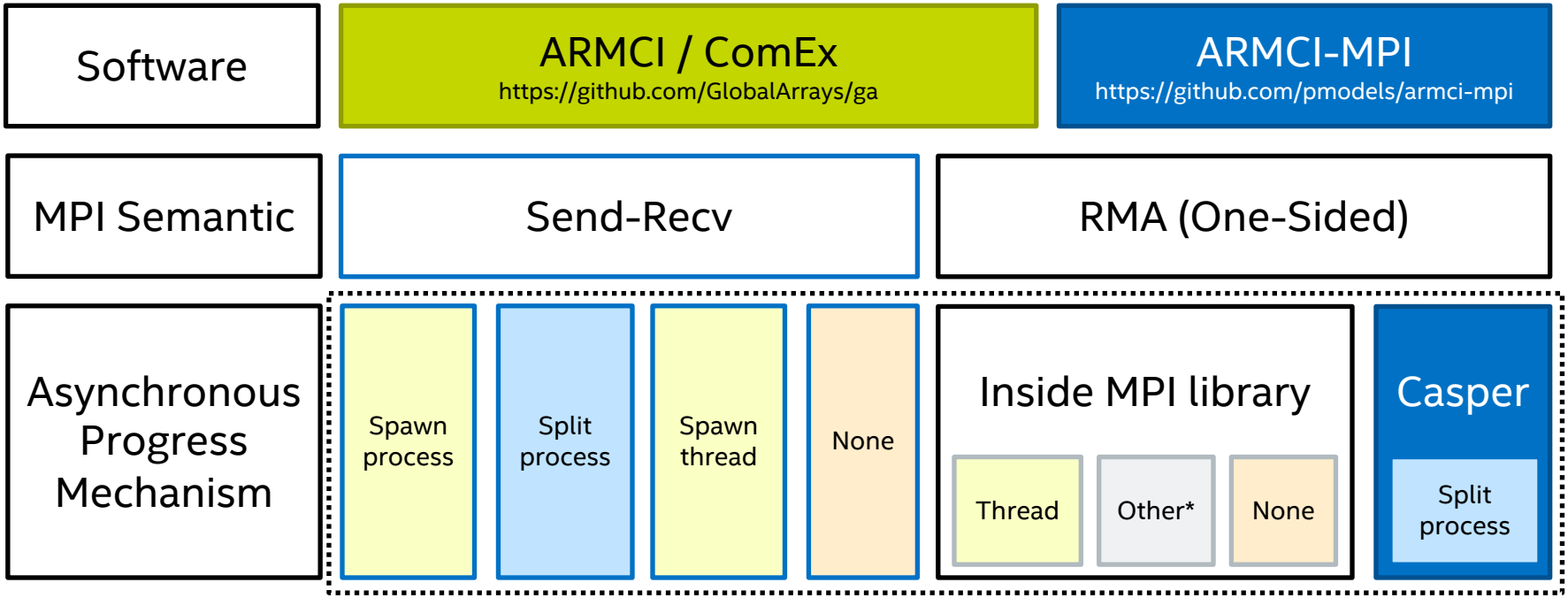
```
double precision buf(100,100)
ga_initialize()
ga_create(MT_DBL, 100, 100, 'matrix', 1, 1, g_m)
ga_create(MT_INT, 1, 1, 'counter', 1, 1, g_c)
ga_zero(g_m); ga_zero(g_c); ga_sync()
buf = 100.0
ga_put(g_m, 1, 100, 1, 100, buf, 100)
buf = 1.0
ga_acc(g_m, 1, 100, 1, 100, buf, 100, 1.0)
ga_get(g_m, 1, 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.

Mapping ARMCI to MPI



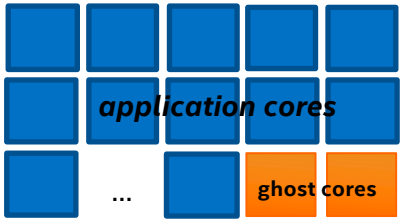
* Interrupts on Blue Gene, hardware offload mechanism (e.g. smart NIC).

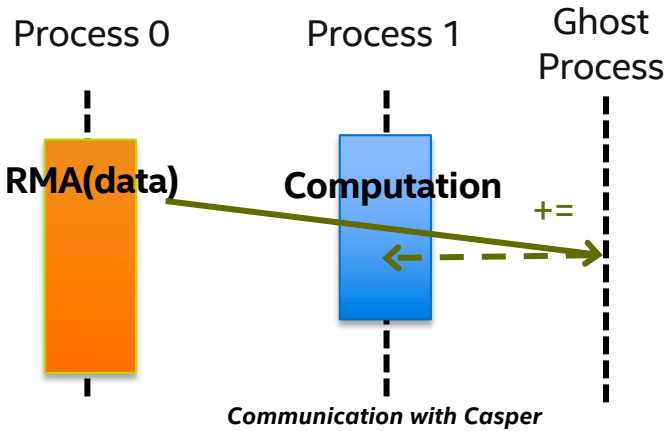
Casper: Asynchronous progress for MPI RMA

Process-based asynchronous progress

- Assign arbitrary number of cores to "ghost processes"
  - Dynamically reconfigurable during application execution to adapt to varying communication intensity (e.g. four-index vs. perturbative triples)
- Ghost process intervenes in all RMA operations to support out-of-band (w.r.t. application) async. progress.
- Works via PMPI interposition i.e. LD_PRELOAD at runtime to opt-in without recompile or relink.

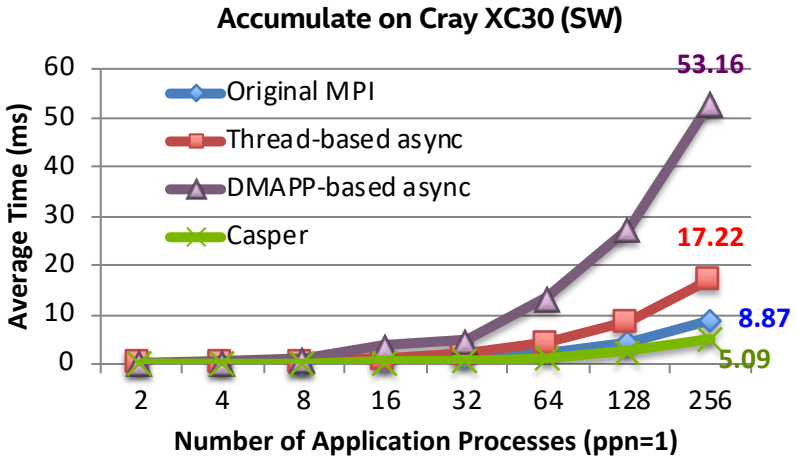
Dr. Min Si (Argonne) is the lead developer and PI of the Casper project.

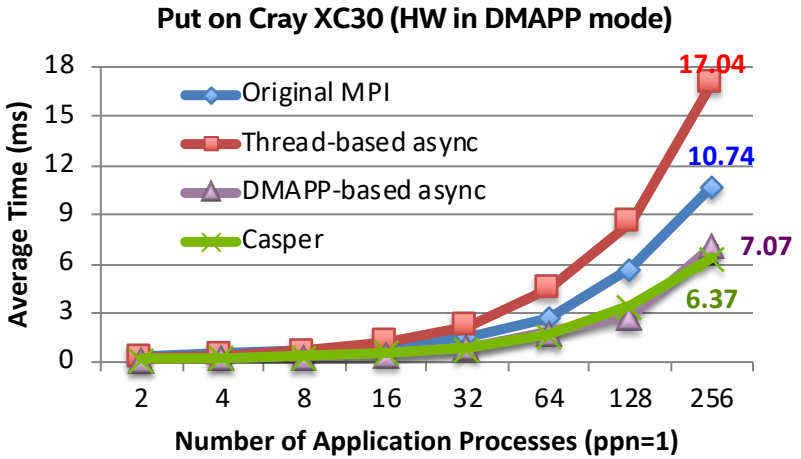




Evaluating Casper


```
Lock_all (win);
for (i=0; i<nproc; i++) {
        OP(i, double, cnt = 1);
        Flush(i);
        busy wait 100us;
}
Unlock_all (win)
```





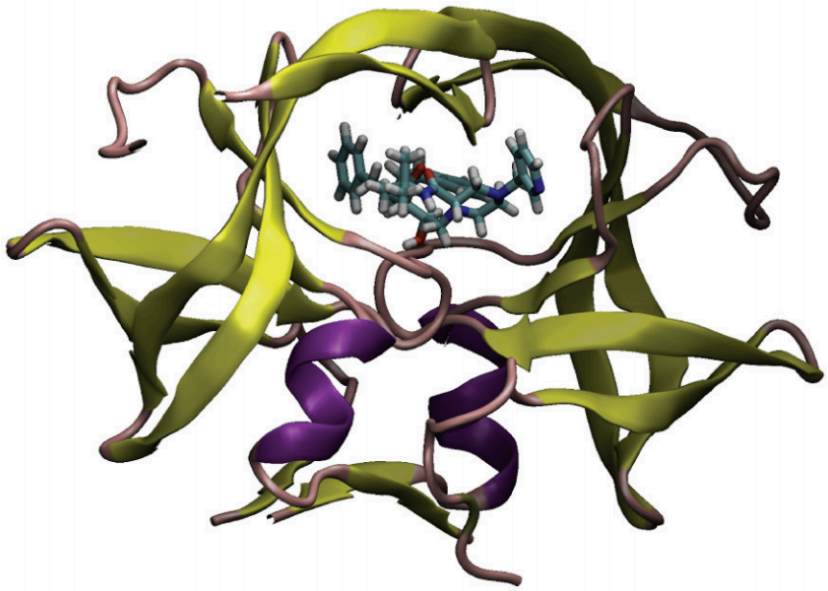
http://dx.doi.org/10.1109/IPDPS.2015.35 / https://www.mcs.anl.gov/papers/P5221-1014.pdf

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 LLNA.
- Allows the computer scientists to focus on the more pressing problems like heterogeneous execution.

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 (old)

NWChem 6.3/ARMCI-MPI3/Casper

iter	energy	time
1	-2830.4366669992	69.6
2	-2831.3734512508	78.8
3	-2831.5712563433	86.9
4	-2831.5727802438	96.1
5	-2831.5727956882	110.0
6	-2831.5727956978	**127.8**

NWChem 6.5/ARMCI-DMAPP
(built by NERSC, Nov. 2014)

iter	energy	time
1	-2830.4366670018	67.6
2	-2831.3734512526	85.5
3	-2831.5713109544	105.4
4	-2831.5727856636	126.6
5	-2831.5727956992	161.7
6	-2831.5727956998	**190.9**

Running on 8 nodes with 24 ppn. Casper uses 2 ppn for comm.

NWChem SCF performance (new)

NWChem 6.3/ARMCI-MPI3/Casper

iter	energy	time
1	-2830.4366669990	**69.3**
2	-2831.3734512499	77.1
3	-2831.5712604368	84.6
4	-2831.5727804428	93.0
5	-2831.5727956927	107.3
6	-2831.5727956977	**128.0**

NWChem Dev/ARMCI-MPIPR
(built by NERSC, Sept. 2015)

iter	energy	time
1	-2830.4366669999	**61.4**
2	-2831.3734512509	69.3
3	-2831.5713109521	77.8
4	-2831.5727856618	87.3
5	-2831.5727956974	103.9
6	-2831.5727956980	**125.7**

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

"The best performance improvement is the transition from the nonworking state to the working state”

– John Ousterhout

NWChem

Science:

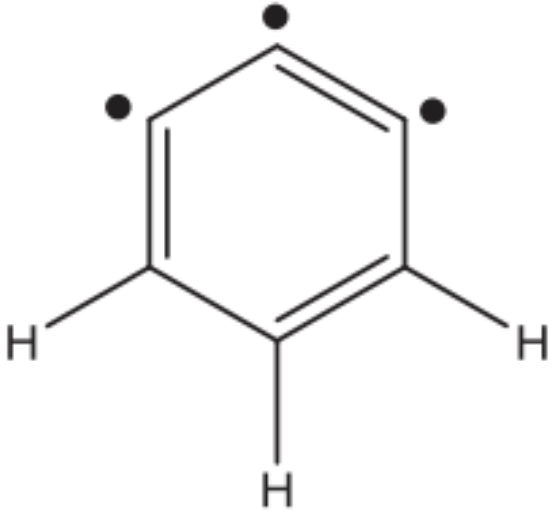
CCSD(T)/cc-pV5Z (very high accuracy) to compare against Quantum Monte Carlo (near-exact) for very challenging open-shell molecule. Such molecules are common in combustion.

Performance:

Run on up to 560 cores of Argonne InfiniBand* cluster and up to 38400 cores of NERSC Edison. On 3840 cores, the calculation took 1.2 hours. IB Verbs implementation crashed 100% of the time.

Citation:

Lucas Koziol and Miguel A. Morales, *J. Chem. Phys.* **140**, 224316 (2014). “A Fixed-Node Diffusion Monte Carlo Study of the 1,2,3-Tridehydrobenzene Triradical”



Not just NWChem

Science:

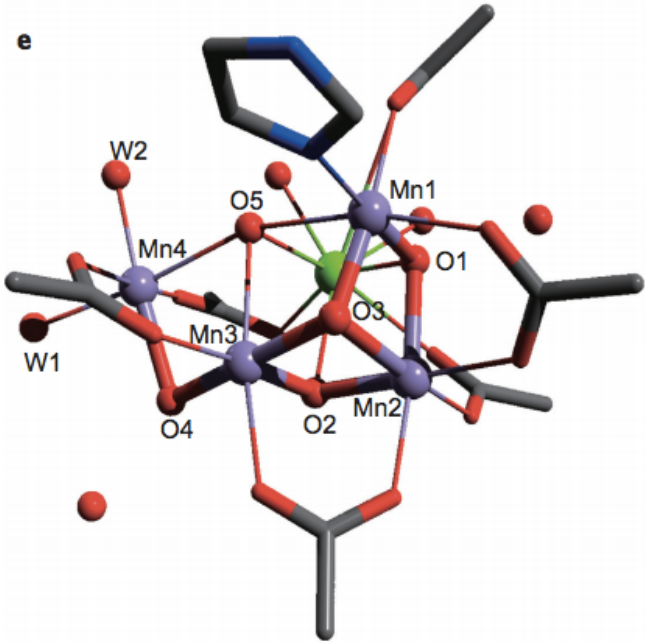
Understanding basic chemistry of catalytic metalloenzymes using highly correlated DMRG wavefunction.

Performance:

Running on group-scale InfiniBand clusters. Not crashing is sufficient performance ☺

Citation:

Yuki Kurashige, Garnet Kin-Lic Chan and Takeshi Yanai. *Nature Chemistry* **5**, 660–666 (2013). “Entangled quantum electronic wavefunctions of the Mn_4CaO_5 cluster in photosystem II”



FORTRAN

Modern Fortran is modern

- Fortran 90 has modules, types, dynamic memory allocation…
- Fortran 2003 exposes all the essential system interfaces (e.g. argc, argv) and supports a standard interface to C features.

Aside: Fortran 2008 supports good MPI bindings for the first time.

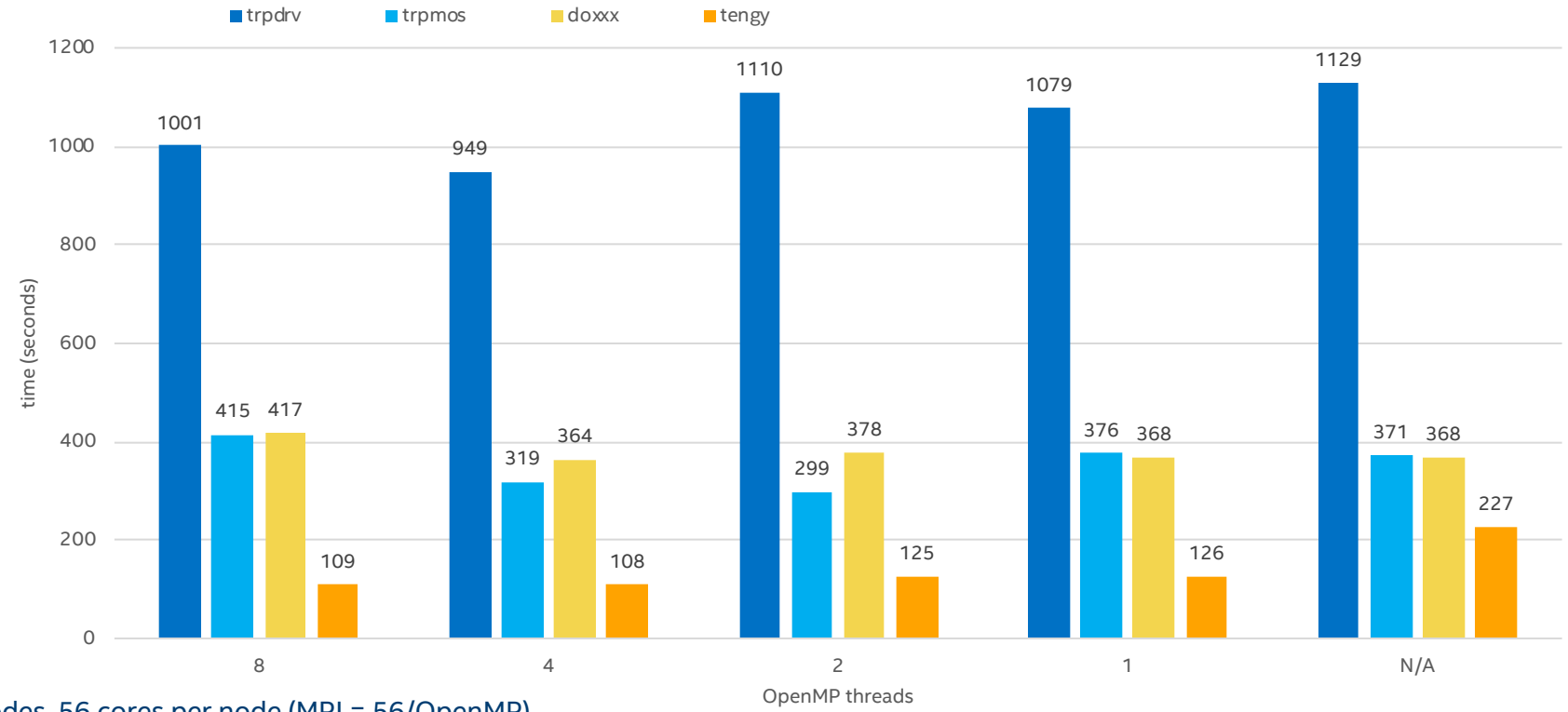
OpenMP

Why (not) OpenMP?

- Only serious option for Fortran
- Path to heterogeneity
- Reduce process count
  - Increase computational intensity
  - Decrease communication frequency
  - Better memory utilization
  - Reduce contention on mutexes
- Amdahl's Law favors processes over threads due to coverage
- Essential subroutines are thread-unsafe (atomic integrals)
- Multi-level parallelism increases design space, especially for load-balancing
- Compiler support for OpenMP target and tasks isn't great

CCSD(T) triples performance

$(H_2O)_7$ with cc-pVTZ (406 basis functions)
Intel® Xeon™ Platinum 8180 processors (2x28)
Omni Path interconnect, local SSD scratch
Intel Fortran, C/C++, MKL (2018.2.199)



4 nodes, 56 cores per node (MPI = 56/OpenMP)

https://github.com/nwchemgit/nwchem/blob/master/src/ccsd/ccsd_trpdrv_omp.F

From multicore to offload (and back)

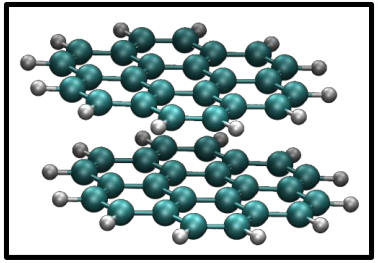
Well-written OpenMP adapts quickly to host and offload execution:

- TCE OpenMP progression: KNC → KNL (Xeon) → TBA
- K-R OpenMP progression: BGP → Xeon (KNL) → KNC → TBA

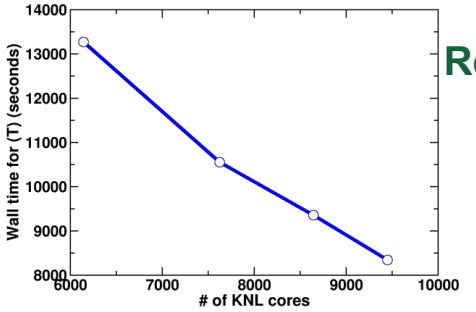
Key requirements for extensible OpenMP:

- Reasonable data structures, explicitly enumerated in data directives
- Long-running parallel execution (minimal fork-join)
- Multilevel or coarse-grain parallelism (e.g. NUMA-friendly)
- Thread-safe subroutines (both comm. and comp.)

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

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**

Cori Supercomputer @ NERSC

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

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

OpenMP Details

NWChem modernization

- Plane-wave DFT code (NWPW)
  - OpenMP: host (holistic)
- Kobayashi-Rendell CCSD(T)
  - Semidirect algorithm (+ints)
  - Non-blocking communication
  - OpenMP
    - host (extensive)
    - offload (triples)
- Tensor Contraction Engine (TCE)
  - Multi-level parallel MRCC
  - Better load-balancing, reduced communication:
    - 4-index, CCSD, (T)
  - OpenMP
    - host (extensive)
    - offload (triples)

NWChem Semidirect CCSD(T)

• SCF iterative solver	$O(r^{2-4})+O(r^3)$	62 seconds
• Four-index transformation (up to 2v terms)	$O(vr^4)$	42 seconds
• CCSD iterative solver	**$O(o^2v^2r^2)$**	
• First iteration saves integral files to local disk		301 seconds
• Subsequent iterations read integral files		94 seconds
• Four-index transformation (only 3v terms)	$O(vr^4)$	632 seconds
• Triples evaluation (non-iterative)	**$O(o^3v^4)$**	200 seconds
	$r=o+v$, $o \ll v$	

NWChem Semidirect CCSD(T)

	Integrals	Array Algebra
• SCF iterative solver	Heavy	Eigensolver
• Four-index transformation (up to 2v terms)	Heavy	TS-GEMM
• CCSD iterative solver		
• First iteration saves integral files to local disk	Heavy	
• Subsequent iterations read integral files	Light (I/O)	
• Four-index transformation (only 3v terms)	Heavy	TS-GEMM
• Triples evaluation (non-iterative)	None	Sq-GEMM
	Not Thread Safe!	Thread Safe!

Threading and atomic integrals (ints)

- NWChem integral code is really fast, but ancient (1990s)
  - Manages cache well, but not vectorizable
  - Fast interpolation (mostly accurate), but using common blocks
- Intel co-developed OptErd for KNC but not significantly faster
- Intel investigating a range of potentially thread-safe options for NWChem
  - Libcint (CalTech)
  - Simint (Georgia Tech) - successor to OptErd
  - ERD (Florida) – original Fortran implementation of OptErd (in GAMESS???)

Detailed profile (and SP)

- 6 MPI, 6 OpenMP on E5-2699v3
- 1.85x benefit from SP in GEMM
- 1.8x benefit from SP in TENGY

  TENGY = reduction over 8 matrices with transposes and scaling
- $(H_2O)_3$ – o=12, v=330
  - 4 GEMM are vvv, 4 GEMM are ovv
- More MPI hits memory + I/O wall

routine	calls	cpu-max	cpu-max
aoccsd	1	0.75	0.75
iterdrv	1	0.45	0.47
pampt	2	1.32	1.45
t2pm	2	2.99	3
sxy	2	39.79	87.21
ints	3934915	304.69	304.51
f_write	1590	11.17	66.04
t2eri	1590	96.97	96.51
idx2	1590	7.25	7.21
idx34	2	0.35	0.35
ht2pm	2	1.39	1.33
itm	2	21.36	11.73
pdiis	2	0.23	0.23
r_read	795	2.38	2.21
triples	1	4	3.72
rdtrpo	1	0.65	0.65
trpmos	1	622.63	457.28
trpdrv	1	17.47	9.63
doxxx	42978	227.99	421.61
tengy	42978	32.05	57.61
Total		1333.91	1426.99

OpenMP coverage

- Triples code is fully threaded
  - 3 loops with comm around 1 region
  - All comms converted to nonblocking
- Parallel regions are as wide as possible within a subroutine
- ~All array algebra is threaded
- ~Half of loops containing GA calls threaded (GA not thread-safe)
- No int loops can be threaded (yet)

```
ccsd_itm_omp.F:!$omp parallel
ccsd_itm_omp.F:!$omp parallel
ccsd_pampt3.F:!$omp parallel do
ccsd_idx1.F:!$omp parallel do
ccsd_idx2.F:!$omp parallel
ccsd_idx34.F:!$omp parallel
ccsd_trpdrv_omp.F:!$omp parallel
```

Thread-safety of GA/ARMCI:
- ARMCI-MPI thread-safety is “done”
- GA thread-safety is massive effort
  - Sent first changeset to PNNL

Good OpenMP techniques

- **omp parallel for [simd]** is an anti-pattern.
  - Treat **omp parallel** like MPI_Init or MPI_Comm_dup - it's expensive.
  - Same for **omp target teams distribute parallel for simd**…
- Identify ALL of the data that needs to be qualified in both host and target.
  - Use **default(none)** to force yourself to be explicit.
  - Getting it right on the host will make it easy to move to **target data**.
- Fusion is your friend. Maximize available parallelism.

Examples

https://github.com/jeffhammond/nwchem/blob/ccsd_trpdrv_omp_c99_gpu/src/ccsd/ccsd_trpdrv.F

https://github.com/jeffhammond/nwchem/blob/ccsd_trpdrv_omp_c99_gpu/src/ccsd/ccsd_trpdrv_omp.F

and other versions in:

https://github.com/jeffhammond/nwchem/tree/ccsd_trpdrv_omp_c99_gpu/src/ccsd

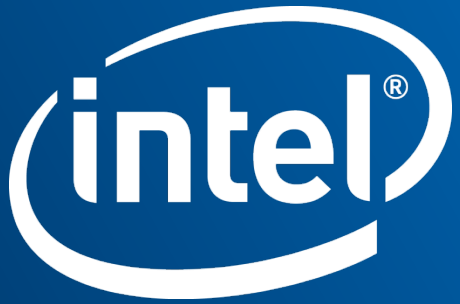
and

https://github.com/nwchemgit/nwchem/tree/master/src/ccsd

Conclusions

Summary

- There are now multiple MPI-based ARMCI implementations that are enabling NWChem to run on “all the supercomputers” with good-to-ideal performance.
- NWChem’s use of MPI RMA is driving the MPI community to do better implementations of MPI-3.
- OpenMP is enabling NWChem coupled-cluster codes to run well on a wide-range of architectures and set the stage for future platforms.
- Fortran 2003 will enable NWChem to remove a bunch of gross utility code that nobody wants to maintain.


intel®