# Application Case Studies: NWChem and MADNESS

Jeff Hammond (jhammond@anl.gov)

Argonne Leadership Computing Facility

https://wiki.alcf.anl.gov/parts/index.php/User:Jhammond

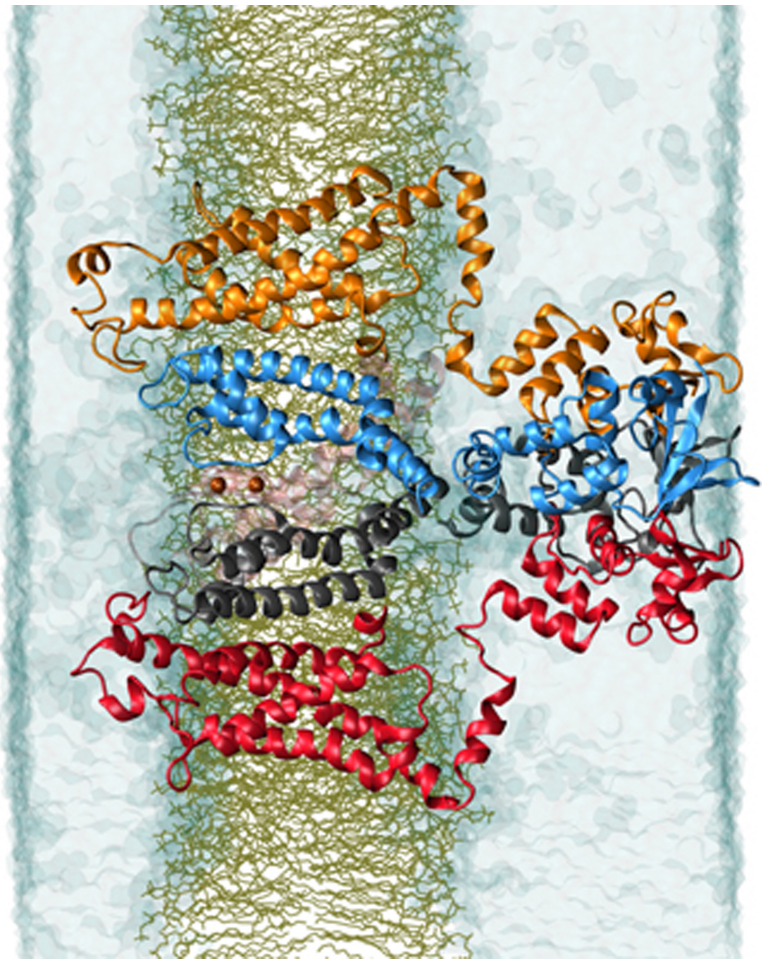
# Outline

- Overview of quantum chemistry
- NWChem: lessons learned
- MADNESS: lessons learned
- Challenges of MPI+X

# Atomistic simulation in chemistry

1. classical molecular dynamics (MD) with empirical potentials
2. ab initio molecular dynamics based upon density-function theory (DFT)
3. quantum chemistry with wavefunctions e.g. perturbation theory (PT), Coupled-Cluster (CC) or Quantum Monte Carlo (QMC).

# Classical molecular dynamics



- Solves Newton's equations of motion with empirical terms and classical electrostatics.
- Size: 100K-10M atoms
- Time: 1-10 ns/day
- Scaling: $\sim N_{atoms}$

Data from K. Schulten, et al. "Biomolecular modeling in the era of petascale computing." In D. Bader, ed., *Petascale Computing: Algorithms and Applications*.

Image courtesy of Benoît Roux via ALCF.

# Car-Parrinello molecular dynamics

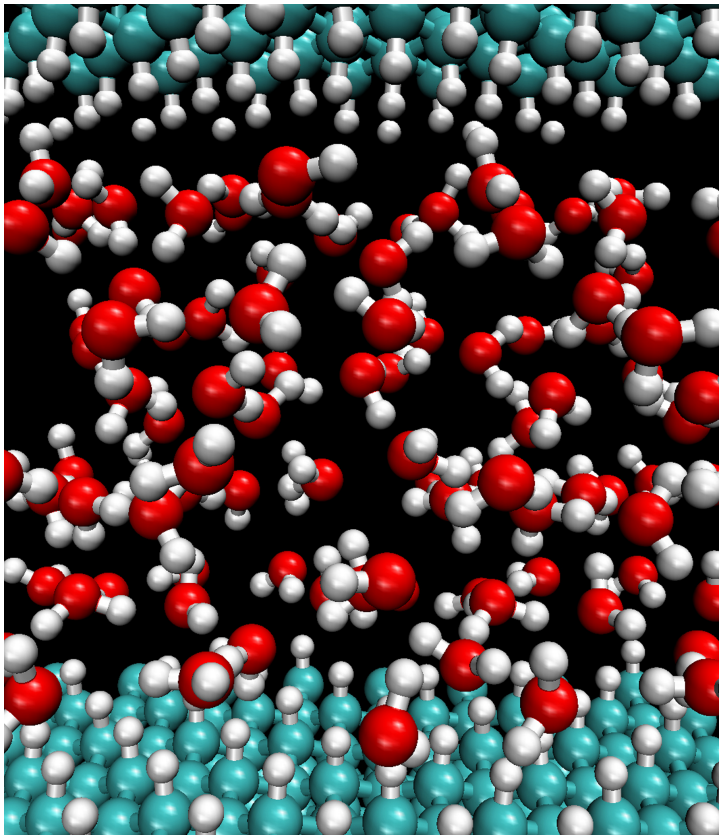
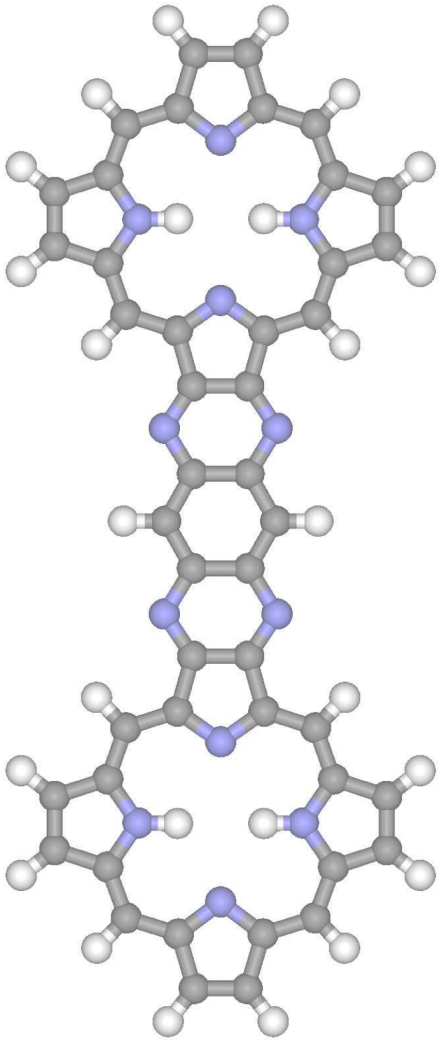


Image courtesy of Giulia Galli via ALCF.

- Forces obtained from solving an approximate single-particle Schrödinger equation; time-propagation via Lagrangian approach.
- Size: 100-1000 atoms
- Time: 0.01-1 ns/day
- Scaling: $\sim N_{el}^{x}$ ($x$=1-3)

F. Gygi, *IBM J. Res. Dev.* **52**, 137 (2008); E. J. Bylaska et al. *J. Phys.: Conf. Ser.* **180**, 012028 (2009).

# Wavefunction theory



- MP2 is second-order PT and is accurate via magical cancellation of error.
- CC is infinite-order solution to many-body Schrödinger equation truncated via clusters.
- QMC is Monte Carlo integration applied to the Schrödinger equation.
- Size: 10-100 atoms, maybe 100-1000 atoms with MP2.
- Time: N/A
- Scaling: $\sim N_{bf}^{x}$ ($x$=2-7)

Image courtesy of Karol Kowalski and Niri Govind.

# Quantum chemistry — standard model

1. Separate molecule(s) from environment (closed to both matter and energy)
2. Ignore relativity, QED, spin-orbit coupling
3. Separate electronic and nuclear degrees of freedom

$\longrightarrow$ non-relativistic electronic Schrödinger equation in a vacuum at zero temperature.

# Quantum chemistry — standard model

$$\begin{aligned}
\hat{H} &= \hat{T}_{el} + \hat{V}_{el-nuc} + \hat{V}_{el-el} \\
\hat{H} &= -\frac{1}{2}\sum_{i=1}^{M}\nabla_i^2 + \sum_{n=1}^{N}\sum_{i=1}^{M}\frac{Z_n}{R_{ni}} + \sum_{i<j}^{M}\frac{1}{r_{ij}}
\end{aligned}$$

$$\Psi\left(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{x}_{n+1},\ldots,\mathbf{x}_N\right) = -\Psi\left(\mathbf{x}_1,\ldots,\mathbf{x}_{n+1},\mathbf{x}_n,\ldots,\mathbf{x}_N\right)$$

The electron coordinates include both space ($r$) and spin ($\sigma$). We will integrate spin out wherever possible.

# Quantum chemistry — standard model

Wavefunction antisymmetry is enforced by expanding in determinents, which we now capture in second quantization.

1. project physical operators (e.g. Coulomb) into one-electron basis — usually atom-center Gaussians
2. generate mean-field reference and expand many-body wavefunction in terms of excitations out of that reference

$\longrightarrow$ Full configuration-interation (FCI) ansatz.

1. truncate exponentially-growing FCI ansatz (CI=linear generator, CC=exponential generator)
2. solve CC (or CI) iteratively
3. add more correlation via perturbation theory

$\longrightarrow$ CCSD(T), as one example.

# Quantum chemistry — standard model

Correct for missing physics using perturbation theory (a posteriori error correction):

1. relativistic corrections
2. non-adiabatic corrections
3. solvent corrections
4. open BC corrections (less common)

# Motivation for HPC

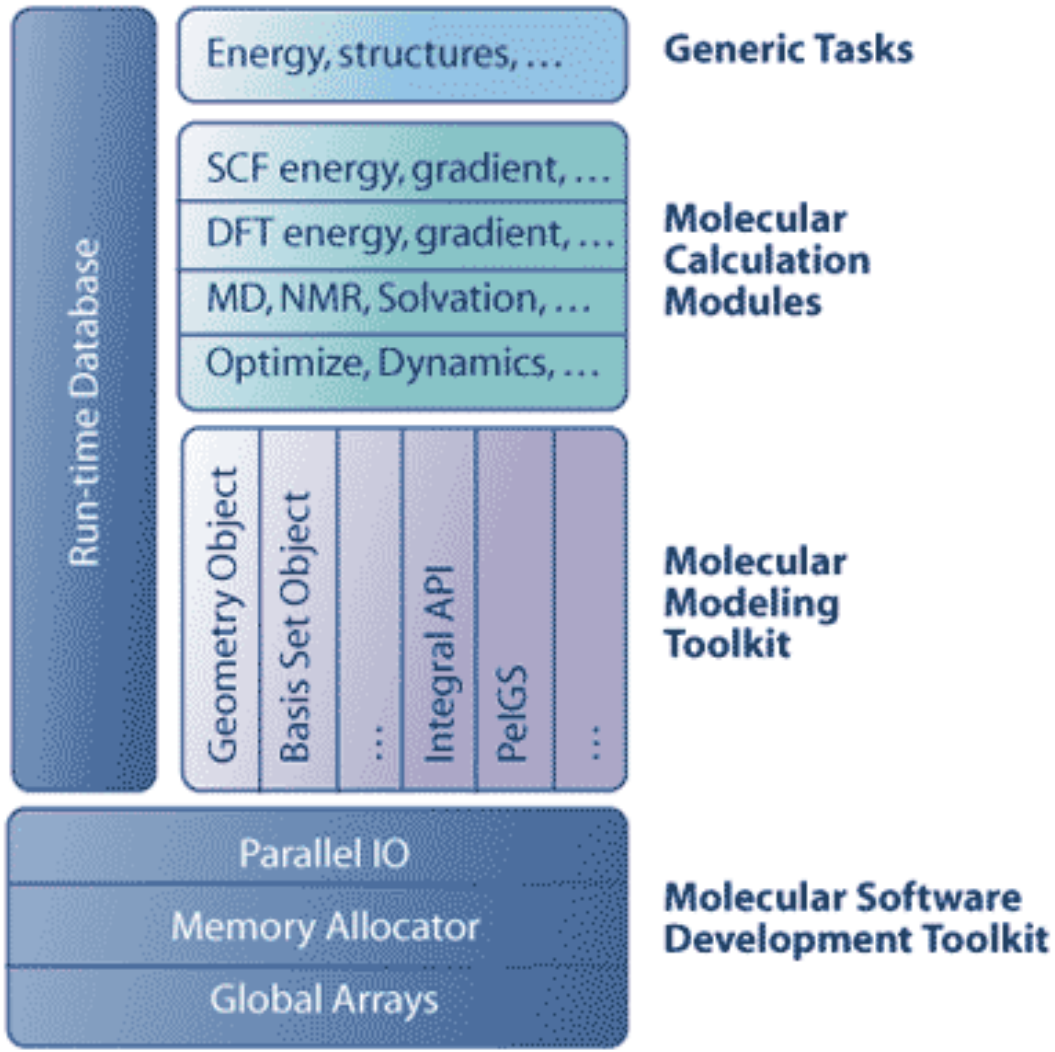
- Electronic excited-states and electric-field perturbations push the limits of conventional approximations in DFT and are outside the scope of classical methods.
- Interesting chemical processes in biology and material science require model systems too large for a conventional computational resources.
- Answering many chemical questions requires large data sets which cannot be obtained in a reasonable amount time if done sequentially.

# NWChem

Initiated by Thom Dunning, early developers include Robert Harrison, Ricky Kendall, Jeff Nichols, Jarek Nieploch (GA), ...

Home page: `http://www.nwchem-sw.org/`

# NWChem Software Architecture




- Think in terms of cakes (mmm, cake).
- Design for the most complex workflow.
- Make sure you cover the complex cartesian space of MxNxK.
- Firewall the CS bits to make the code friendly to science developers.

# NWChem Epochs

1. Prototype and development stage. Get GA working. Figure out software best practices. The Osterhout phase.
2. Get the core features working in parallel. Few new method but new applications thanks to increased scale. The meat and potatoes phase.
3. First parallel X for a lot of X; new capability due to integration of X+Y. The 'Half Baked' phase: "have you ever done . . . *in parallel*?"
4. Transition to a robust community code with thousands of users; users not just in it for the parallelism.
5. Breakthrough capability (TCE) and adoption as a research platform.
6. To petascale and beyond . . .

# Challenges

(repeated from last week)

- Hard to find talented parallel programmers with domain expertise.
- F77 rather than C++ for OO design.
- GA started before MPI; ARMCI portability requires huge effort.
- Lack of compiler optimizations cause unfortunate compromises in abstraction/encapsulation vs. performance.
- The world of FOSS was different then: look up state-of-the-art in version control in 1994.
- Funding sources get complacent.

# Evil!

```
nwdft/coulomb/dft_mem3c.F:#include "basP.fh"
nwdft/coulomb/dft_n3c.F:#include "basP.fh"
nwdft/rt_tddft/init/rt_tddft_init_geoms.F:#include "geomP.f
nwdft/rt_tddft/input/rt_tddft_input_excite.F:#include "geom
nwdft/so_dft/grad_force_so.F:#include "apiP.fh"
ddscf/fast/pot_shell.F:#include "spcartP.fh"
ddscf/int_1e_ga.F:#include "apiP.fh"
ddscf/print_1eints.F:#include "apiP.fh"
ccsd/ccsd_aux.F:#include "apiP.fh"
ccsd/ccsd_aux.F:#include "basP.fh"
ccsd/ccsd_aux.F:#include "basdeclsP.fh"
ccsd/ccsd_aux.F:#include "geomP.fh"
ccsd/ccsd_aux.F:#include "geobasmapP.fh"
```

# Successes

- One of a handful of original gangsters of application codesign; intensive collaboration between Chem, CS and Math people.
- NWChem rode the wave of MPPs from the Intel Delta and KSR to Cray XT/XE/XC and Blue Gene/P and /Q.
- Global Arrays is the basis for the majority of parallel quantum chemistry codes today. The community recognized that GA was not an NWChem-specific solution.
- NWChem integrated parallelism, best methods and fast algorithms at the same time; in many cases it was the first to do this.
- Lots of science. This is the primary goal, after all.

# Failures

- Developer community is not open; repo is private and locked down.
- Suffering from sacred code disease. Dead code is not removed.
- Broken abstractions have led to spaghetti. Linking NWChem's library components is nearly impossible.
- Neglected multicore challenges; no significant use of threads. Refactoring for thread-safety is still not happening.
- GA programming model template is latency sensitive: blocking Get is the worst form of one-sided, yet the easiest to use.
- Centralized load-balancer was designed for 100-way parallelism – is still used on 100,000-way parallel systems.

# What are we doing about it?

(This list is by no means complete!)

- Version 6+ is OSS; you can at least hack on the releases.
- Performance tuning for multicore (LBNL).
- Designing new load-balancers inspired by work-stealing and static partitioning (PNNL and ALCF).
- GPU offload (PNNL); OpenMP and MIC offload (ALCF).
- Lots of science work still happening (PNNL, etc.).

In spite of all of this, without DI$RUPTIVE development, we're not making it to exascale.

# MADNESS

Led by Robert Harrison and George Fann.

https://code.google.com/p/m-a-d-n-e-s-s/

# What is MADNESS?

*Multiresolution Adaptive Numerical Environment for Scientific Simulation*

General purpose numerical framework for reliable and fast scientific simulations: Chemistry, nuclear physics, atomic physics, material science, climate, fusion...

A general purpose parallel programming environment designed for the petascale and beyond.

A research platform that provides global namespace, futures, active messages, etc.

# MADNESS Math

E.g., with guaranteed precision of 1e-6 form a numerical representation of a Gaussian in the cube $[-20,20]^3$, solve Poisson's equation, and plot the resulting potential (all running in parallel with threads+MPI)

Let

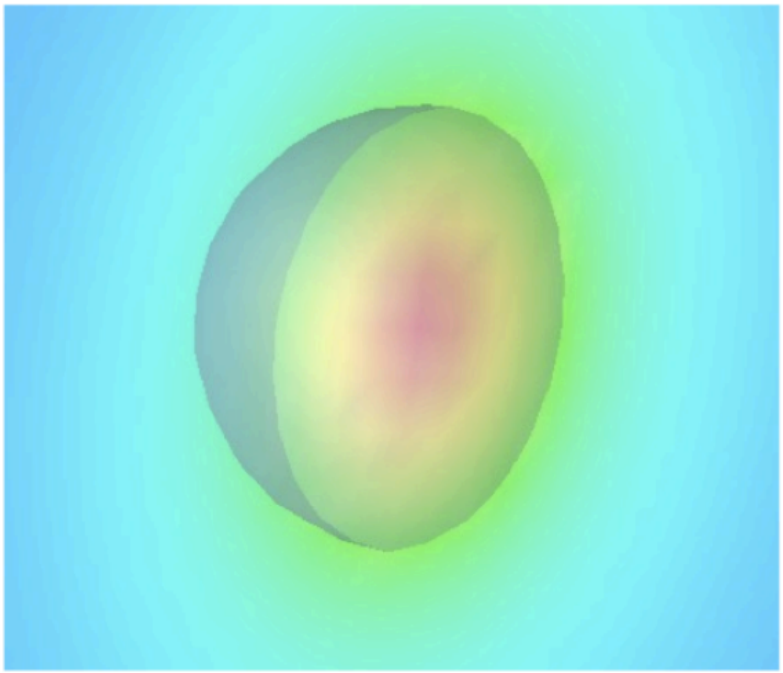
$$\Omega = [-20, 20]^3$$
$$\epsilon = 1e-6$$
$$g = x \rightarrow \exp\left(-\left(x_0^2 + x_1^2 + x_2^2\right)\right) * \pi^{-1.5}$$

In

$$f = \mathcal{F}\, g$$
$$u = \nabla^{-2}\left(-4 * \pi * f\right)$$
print "norm of f", $\langle f \rangle$, "energy", $\langle f | u \rangle * 0.5$
plot $u$

End

*output:* norm of f 1.00000000e+00 energy 3.98920526e-01



# MADNESS Math

We can write in an integral form the solution for an electronic wavefunction.

$$\hat{H}\varphi(r) = (\tfrac{1}{2}\nabla^2 + \hat{V})\varphi(r) = \varepsilon\varphi(r)$$

Integral equation from:

$$\varphi(r) = -2(\nabla^2 + 2\varepsilon)^{-1}\hat{V}\varphi(r)$$

– Higher accuracy achievable in integral form
– Correct asymptotics by design
– Computationally efficient: LSR and local refinement
– MRA provides fast algorithms with guaranteed precision

# Learning from NWChem

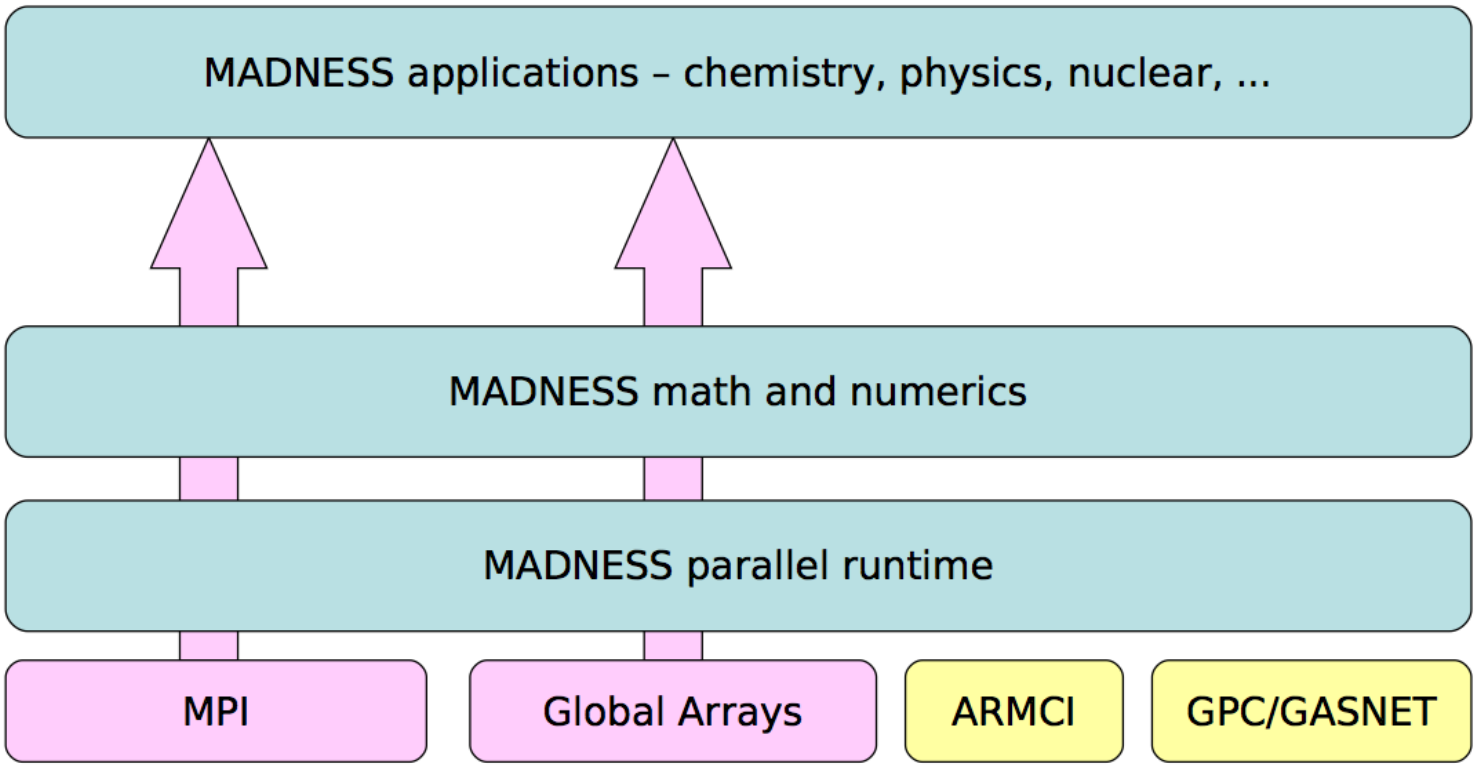
- Collaboration between Domain (Chemistry, Physics, etc.), CS and Math people.
- Asynchronous programming model and always-on implicit (global namespace not process-centric) parallelism.
- Portable runtime via MPI, Pthreads, C++. Non-portable elements limited to threaded runtime, which is giving way to Intel TBB.
- GPUs are a problem but that's NVIDIA's problem, not ours. MIC works just fine.

# MADNESS Coding Standards

- C++03 with optional C++11; use inline assembly for kernels.
- Ideas but not code from GA, Cilk, Charm++, Chapel.
- MPI+Pthread runtime – moving to TBB for thread runtime now.
- Portable from Amazon EC2 and Macbook to Cray XC and Blue Gene/Q.
- MIC and GPU supported but offload latency and lack of fine-grain task control on GPU is a problem.

# MADNESS Software Architecture

## MADNESS architecture




Intel Thread Building Blocks more scalable; also ported to BGQ

# MADNESS Performance on Blue Gene/Q

| | Elemental | Eigen3 | LAPACK |
|---|---|---|---|
| Water*70 (Full SCF)<br>16 nodes | 1272.4<br>6.7% | 1387.7<br>-2.2% | 1357.3 |
| Water*213 (maxiter=2)<br>256 nodes | 2753.5<br>15.2% | 3546.9<br>-10.6% | 3171.4 |
| eigen solution time per iteration | 1.5 s | | ~700 s |

MADNESS uses an adaptive numerical basis set that cannot be directly compared to localized orbitals or plane-waves...

# Lessons learned from MADNESS

- Original version prototyped in Python to debug ideas.
- Outsource portability to standards (MPI, POSIX).
- Do not reimplement the same algorithm better if there's a better algorithm that has never been implemented!
- Don't prematurely solve problems with known answers if it interferes with agility.
- Open development model. GPLv2 on Google Code.
- Project authority proportion to LOC contributed.
- Annual developer workshops; frequent hackathons.
- Unit tests; working on Jenkins automation.
- Using pro tools e.g. Google Test.

# Challenges with MPI+X

# The future is MPI+X

- MPI+OpenMP is too often fork-join.
- Pthreads scare people; can't be used from Fortran (easily).
- TBB and Cilk come from Intel (FYI: TBB now runs on BGQ).
- OpenCL is an eye chart and has no abstraction for performance variability.
- CUDA is an X for only one type of hardware (ignoring Ocelot).

Never confuse portability with portable performance!

# MPI+Y

- If you use OpenMP libraries built with multiple compilers, you may get multiple thread pools.
- OpenMP, TBB, etc. all use Pthreads. So do many apps and libraries. Oversubscribe much?
- `MPI_THREAD_MULTIPLE` adds overhead; some apps use their own mutex but internal mutexes are invisible to other MPI clients.

The stark reality is that general MPI+Y – i.e. MPI+X for X≠OpenMP – is heavily dependent upon an MPI implementation that is designed to be used in a truly multithreaded way. Today, only Blue Gene/Q as this.

Based on `https://www.ieeetcsc.org/activities/blog/challenges_for_interoperability_of_runtime_systems_in_scientific_applications`

# Acknowledgments

ALCF, Pavan Balaji, Jim Dinan, Robert Harrison, Karol Kowalski, Jack Poulson, Robert van de Geijn, and many others.