BERKELEY LAB

Bringing Science Solutions to the World

NWChemEx, Current State and Solver Needs

Bert de Jong wadejong@lbl.gov

Just coming off a summer school and workshop hosted at Berkeley last week

MolSSI

BERKELEY LAB

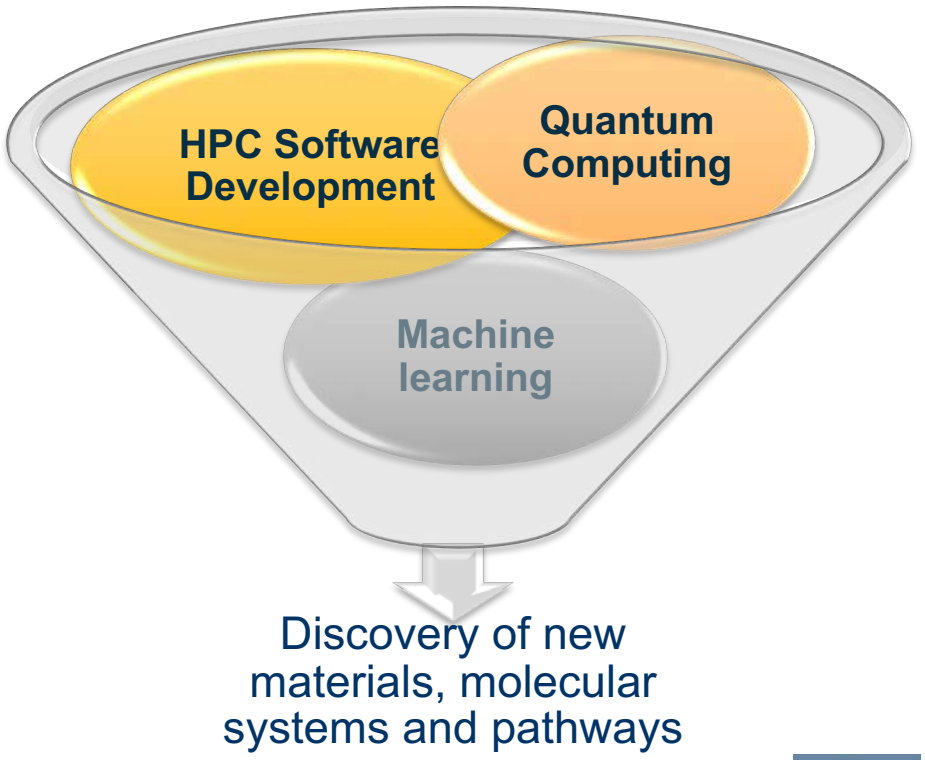
iACS
INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

ECP
EXASCALE COMPUTING PROJECT

BERKELEY LAB

My group pursues multiple solutions to scientific discovery

- Next-generation HPC code for exascale (NWChemEx)
- Using machine learning for discovery and computing
- Quantum computing to tackle exponential complexity



Looking back at NWChem

NWChem v1.0 developed in the EMSL Project (1992-1997)

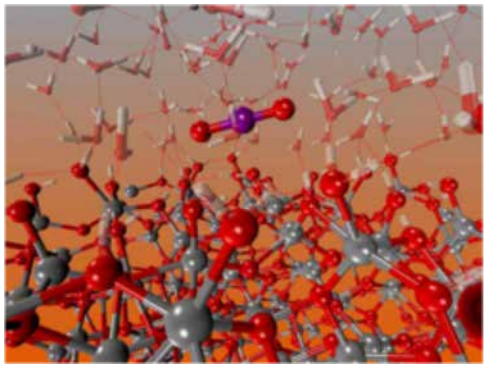
- NWChem v6.6 now available
- Designed for parallel computers from "scratch"

Open-source, High-performance Computational Chemistry Package, Implementing Broad Range of Theoretical Methods

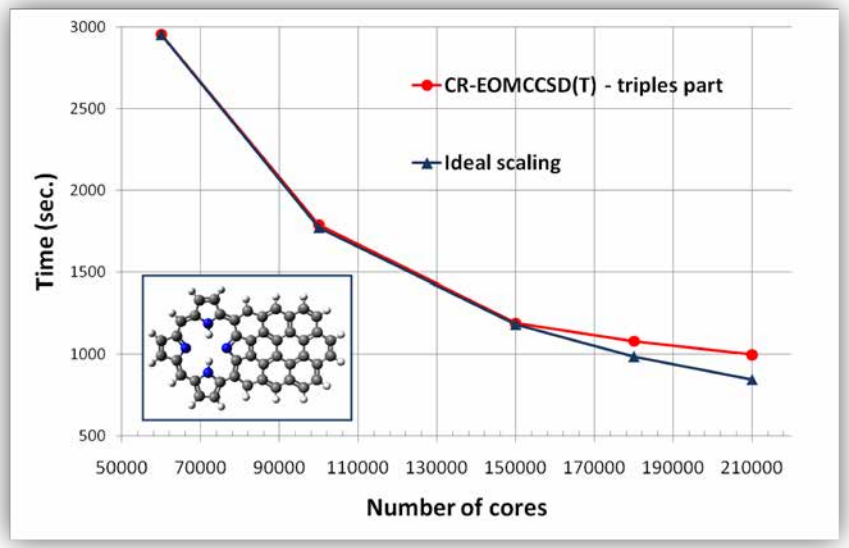
- Downloaded over 70,000 times (includes updates and patches)
- Most recent release (v6.6) downloaded over 4,000 times (10/15/2017)
- More than 3,000 citations to papers describing NWChem (Google Scholar)

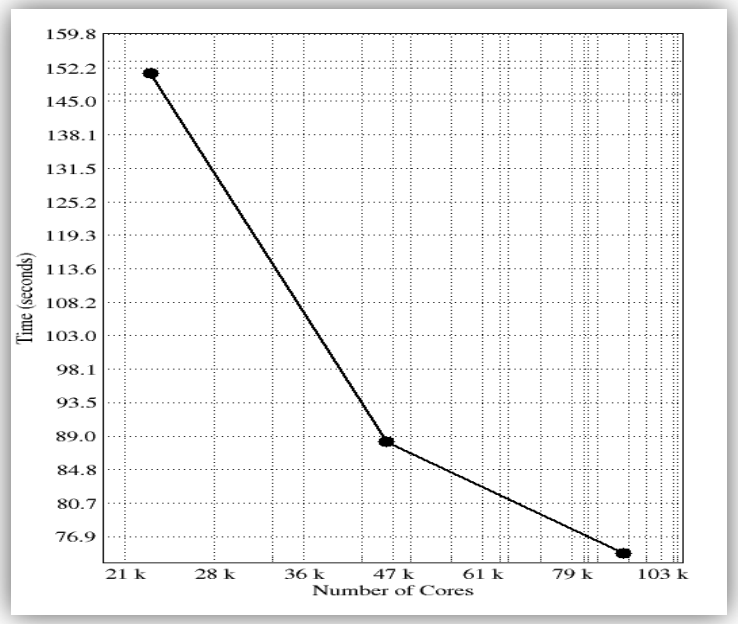
Pioneered Software Technologies Incorporated in Other Codes

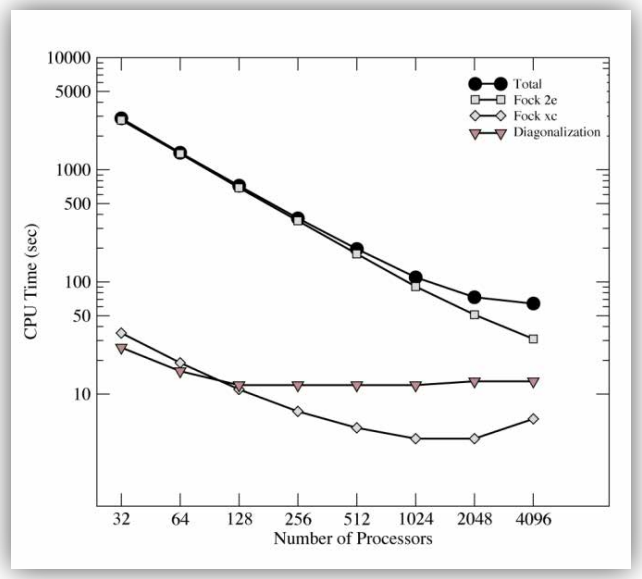
- Global Arrays
- Tensor Contraction Engine

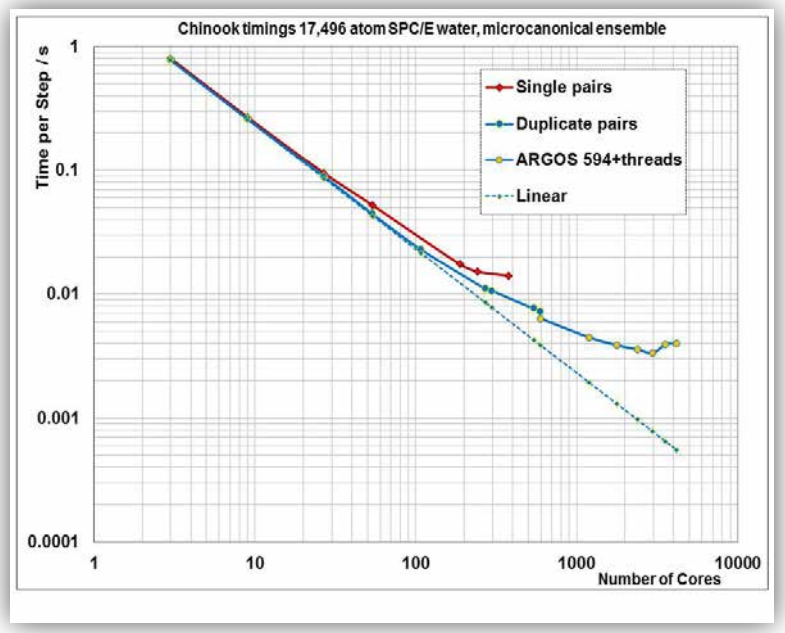


For 20 years NWChem scaled as supercomputers grew…





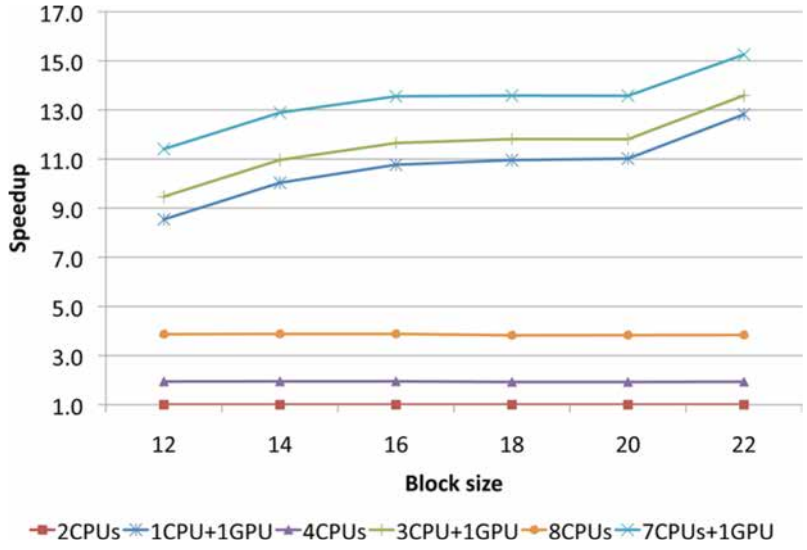


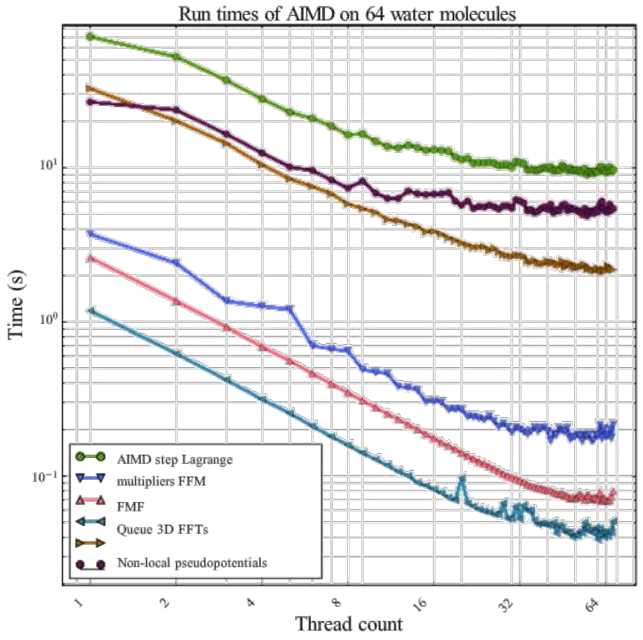


BERKELEY LAB

…but new architectures have become more revolutionary!

GPUs and multicore architectures are showing aging design and concurrency bottlenecks…





…and exascale architectures are likely even more exotic!

BERKELEY LAB

Looking at NWChem with exascale architectures in mind

- Software architecture challenges
- Limits on concurrency
- Reliance on hand rewriting of code for new hardware
- Lack of adaptability and portability to the hardware environment (e.g., memory and I/O)
- Limited approaches to dynamically balance the workload

NWChem's challenges beyond exascale

NWChem's roots lie in scalable code for dense algorithms

- Transition to reduced scaling methodologies needed
- While it has an object oriented design, it was build with Fortran
- Monolithic code

Cutting the cord, starting from scratch with NWChemEx

NWChemEX will enable broad range of new chemistry research with robust predictive capabilities

Redesign NWChem for Exascale Computing Technologies

- Address limitations (new design, state-of-the-art algorithms, take advantage of massive levels of concurrency, …)
- Update code base to C++17 to enable more flexible design

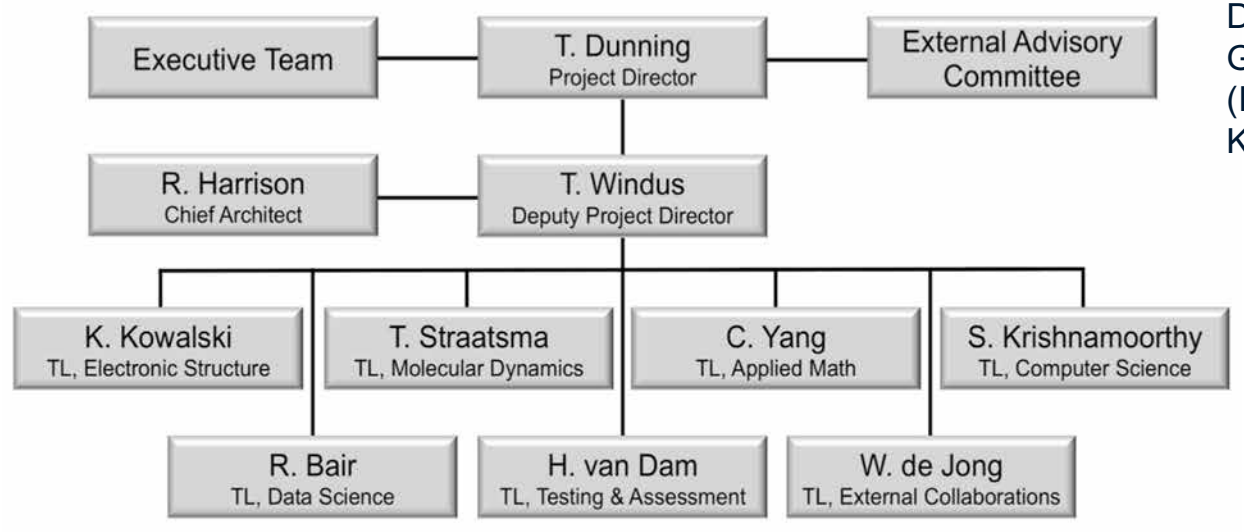
Creating New Computational Framework

- Developing "Tensor Algebra for Many-Body Methods" (TAMM) to replace TCE
- Working closely with XGA team on Global Arrays
- Exploring other software technologies in the ECP

NWChemEx, focus on reduced scaling

- Reduce computational cost with system size; increase accuracy for a given system size
- Reducing the memory footprint and communication through compact representation and low rank approximations
- Improved control of the basis set error in large systems
- Sparse and low-rank representations with robust error control
- Efficiently explore the potential energy surface
- Utilize flexible and efficient frameworks for partitioning the molecule and environment
- Using different quantum chemical representations
- Better convergence of iterative methods for nonlinear problems

Organization of NWChemEx Project



D. Crawford (Virginia Tech), M. Gordon (Iowa State), S. Hirata (Illinois), E. Chow (Georgia Tech), K. Yelick (UC Berkeley)

Edo Aprà (PNNL)
Eric Bylaska (PNNL)
Niri Govind (PNNL)
Colleen Bertoni (Iowa)

Ed Valeev (VATech)
Kris Keipert (ANL)
Jeff Boschen (Iowa)

Ryan Richard (Iowa)
Ajay Panyala (PNNL)
Erdan Mutlu (PNNL)

Targeted Science Challenges

DOE 2014-2018 Strategic Plan

- Development of high performance computing models that demonstrate that biomass can be a viable, sustainable feedstock for biofuels, hydrogen and other products

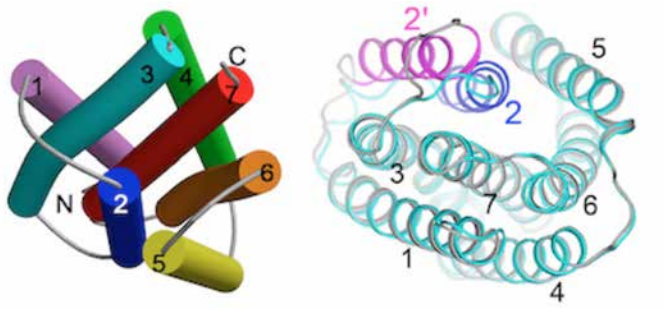
Two Inter-related Science Challenges

- Efficient Production of Biomass

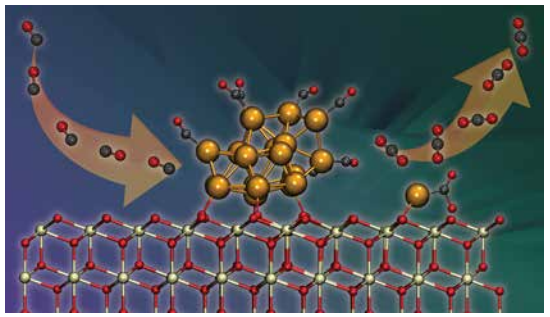
  Development of detailed molecular model of transport processes across cellular membranes that control stress responses to aid in the development of stress-resistant crops

- Efficient Conversion of Biomass to Biofuels

  Development of detailed molecular model of catalytic conversion of biomass-derived alcohols to biofuels to aid in the discovery of energy efficient conversion processes

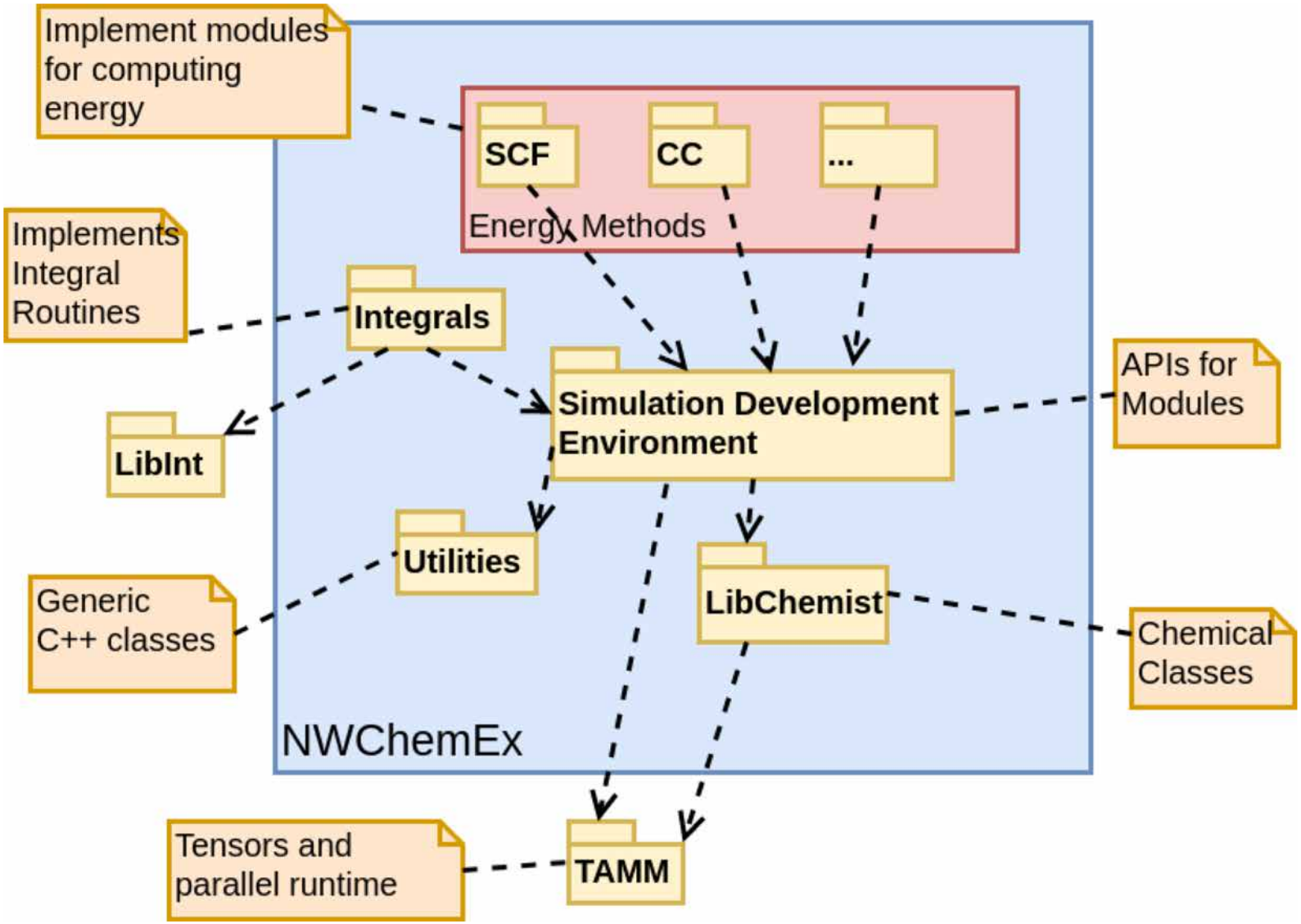


Transport of Ca^{2+} ions across membranes in Bax Inhibitor-1



Catalytic dehydration of 2-propanol in H-ZSM-5

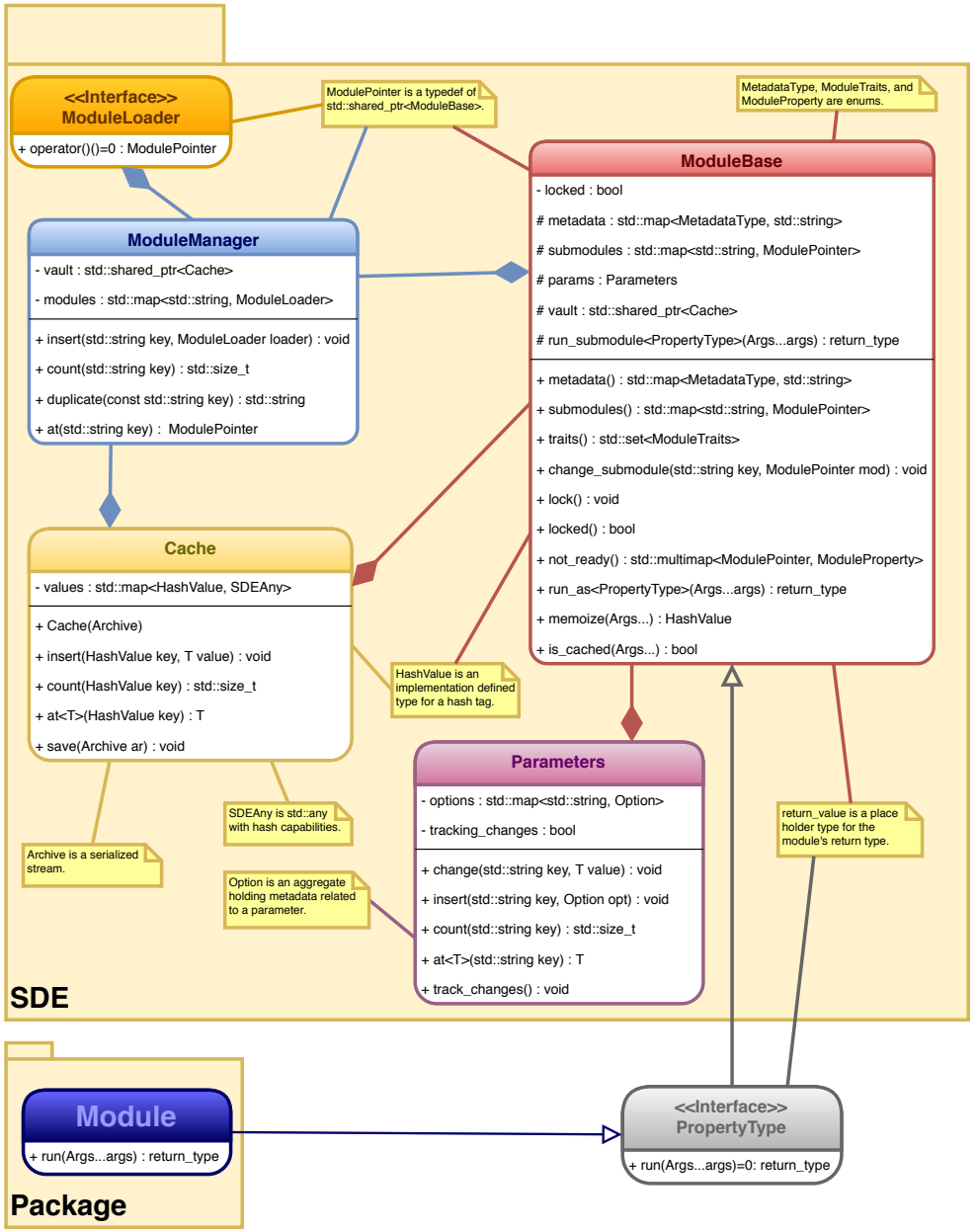
Rough view of NWChemEx infrastructure



Heart of NWChemEx will be SDE and TAMM

Simulation development environment (SDE)

- Well defined classes
- Learning from other efforts
- Designed for
  - Flexibility
  - Extendibility



BERKELEY LAB

Tensor Algebra for Many-body Methods (TAMM)

- TAMM builds on ideas from TiledArray (Valeev, Virginia Tech)
- Build on the notion of index spaces, with large flexibility on how to define and distribute those
  - Key for sparse, low-rank algorithms
- Integrated runtime and scheduler to minimize communication and maximize load-balanced computation
  - Will support various schedulers

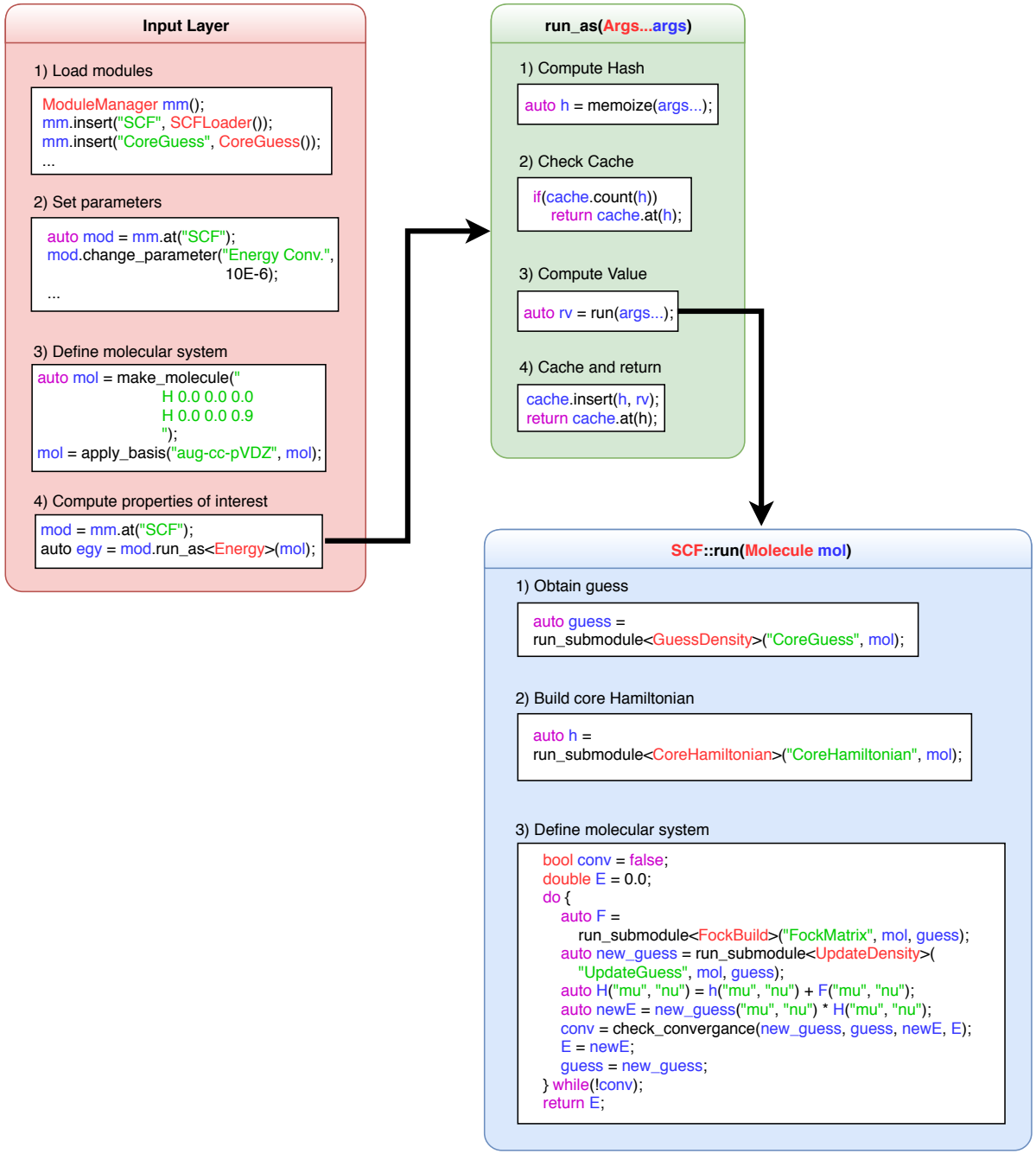
What does code look like with TAMM

```
template<typename T>
void hartree_fock(const TiledIndexSpace& AO,
                  const Tensor<T>& C,
                  Tensor<T>& F) {
  const TiledIndexRange& N = AO.range("all");
  const TiledIndexRange& O = AO.range("occ");

  TiledIndexLabel a,b,c;
  TiledIndexLabel ao,bo,co;
  std::tie(a,b,c) = AO.range_labels("all", 1, 2, 3);
  std::tie(ao,bo,co) = AO.range_labels("occ", 1, 2, 3);

  // compute overlap integrals
  //Tensor<T> S{N,N};
  //S = compute_1body_ints(shells, Operator::overlap);
  Tensor<T> S{N,N, one_body_overlap_integral_lambda};
  // compute kinetic-energy integrals
  Tensor<T> T{N,N,one_body_kinetic_integral_lambda};
  //T = compute_1body_ints(shells, Operator::kinetic);
  // compute nuclear-attraction integrals
  //Tensor<T> V{N,N};
  //V = compute_1body_ints(shells, Operator::nuclear, atoms);
  Tensor<T> V{N,N, one_body_nuclear_integral_lambda};
  // Core Hamiltonian = T + V
  Tensor<T> H{N, N};
  H(a,b) = T(a,b);
  H(a,b) += V(a,b);

  Tensor<T> D{N, N};
  compute_soad(atoms, D);

  const auto maxiter = 100;
  const auto conv = 1e-12;
  auto iter = 0;
  Tensor<T> ehf{},ediff{},rmsd{};
  Tensor<T> eps{N,N};

  do {
    ++iter;
    // Save a copy of the energy and the density
    Tensor<T> ehf_last{};
    Tensor<T> D_last{N,N};
```

What does code look like with TAMM for SCF

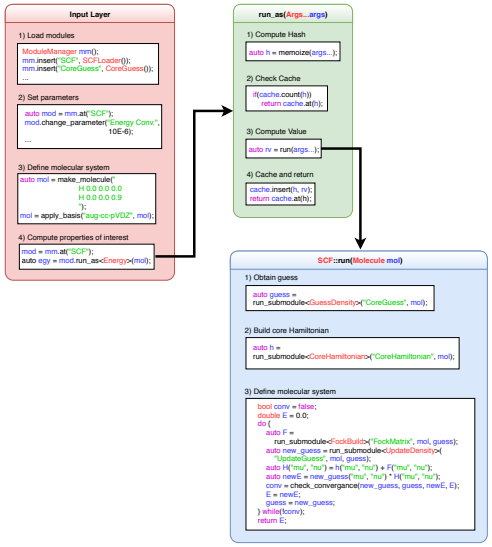
```{c++}
void compute_2body_fock(const TiledIndexSpace& AO,
            const std::vector<libint2::Shell> &shells,
            const Tensor<T> &D, Tensor<T> &F) {
  const TiledIndexRange& N = AO.range("all");
  TiledIndexLabel s1, s2, s3, s4;
  std::tie(s1,s2, s3, s4) = AO.range_labels("all", 1, 2, 3, 4);
  const auto n = nbasis(shells);
  Tensor<T> G{N,N};
  //TODO: construct D from C
  // construct the 2-electron repulsion integrals engine
  Tensor<T> ERI{N, N, N, N, coulomb_integral_lambda};
  Scheduler()
  .fuse(PermGroup(,,,).iterator(),
      G(s1, s2) += D(s3, s4) * ERI(s1, s2, s3, s4),
      G(s3, s4) += D(s1, s2) * ERI(s1, s2, s3, s4),
      G(s1, s3) -= 0.25*D(s2,s4) * ERI(s1,s2,s3,s4),
      G(s2, s4) -= 0.25*D(s1,s3) * ERI(s1,s2,s3,s4),
      G(s1, s4) -= 0.25*D(s2,s3) * ERI(s1,s2,s3,s4),
      G(s2, s3) -= 0.25*D(s1,s4) * ERI(s1,s2,s3,s4)
      ).execute();


  // symmetrize the result and return
  //Tensor<T> Gt{N,N};
  //Gt(a,b) = G(b,a); //G.transpose();
  F(s1,s2) += 0.5 * G(s1,s2);
  F(s1,s2) += 0.5 * G(s2,s1);
}

template<typename T>
void hartree_fock(const TiledIndexSpace& AO,
                  const Tensor<T>& C,
                  Tensor<T>& F) {
  const TiledIndexRange& N = AO.range("all");
  const TiledIndexRange& O = AO.range("occ");

  TiledIndexLabel a,b,c;
  TiledIndexLabel ao,bo,co;
```

What will code look like in NWChemEx



BERKELEY LAB

What will code look like in NWChemEx



Input Layer

1) Load modules

```
ModuleManager mm();
mm.insert("SCF", SCFLoader());
mm.insert("CoreGuess", CoreGuess());
...
```

2) Set parameters

```
auto mod = mm.at("SCF");
mod.change_parameter("Energy Conv.",
                     10E-6);
...
```

3) Define molecular system

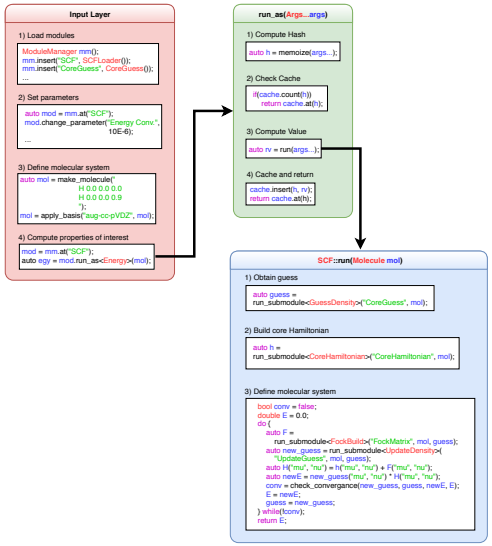
```
auto mol = make_molecule("
                         H 0.0 0.0 0.0
                         H 0.0 0.0 0.9
                         ");
mol = apply_basis("aug-cc-pVDZ", mol);
```

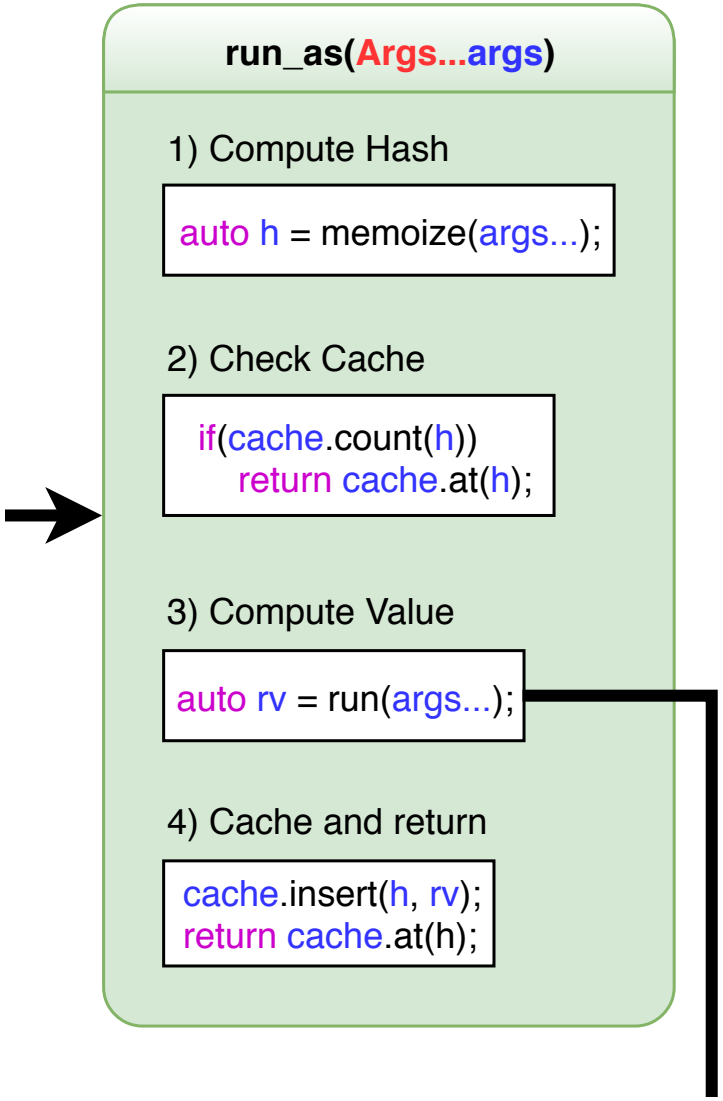
4) Compute properties of interest

```
mod = mm.at("SCF");
auto egy = mod.run_as<Energy>(mol);
```


BERKELEY LAB

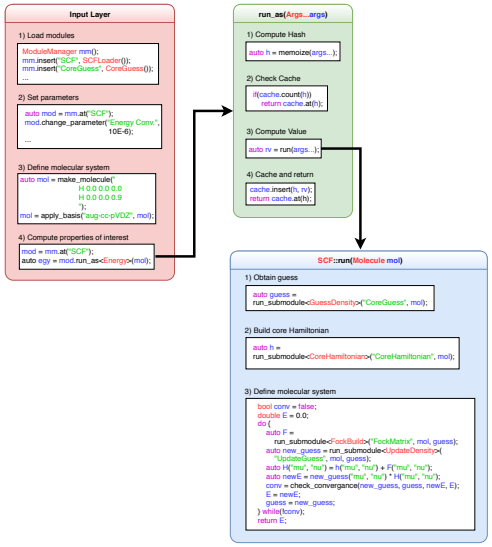
What will code look like in NWChemEx





BERKELEY LAB

What will code look like in NWChemEx



SCF::run(Molecule mol)

1) Obtain guess

```
auto guess =
run_submodule<GuessDensity>("CoreGuess", mol);
```

2) Build core Hamiltonian

```
auto h =
run_submodule<CoreHamiltonian>("CoreHamiltonian", mol);
```

3) Define molecular system

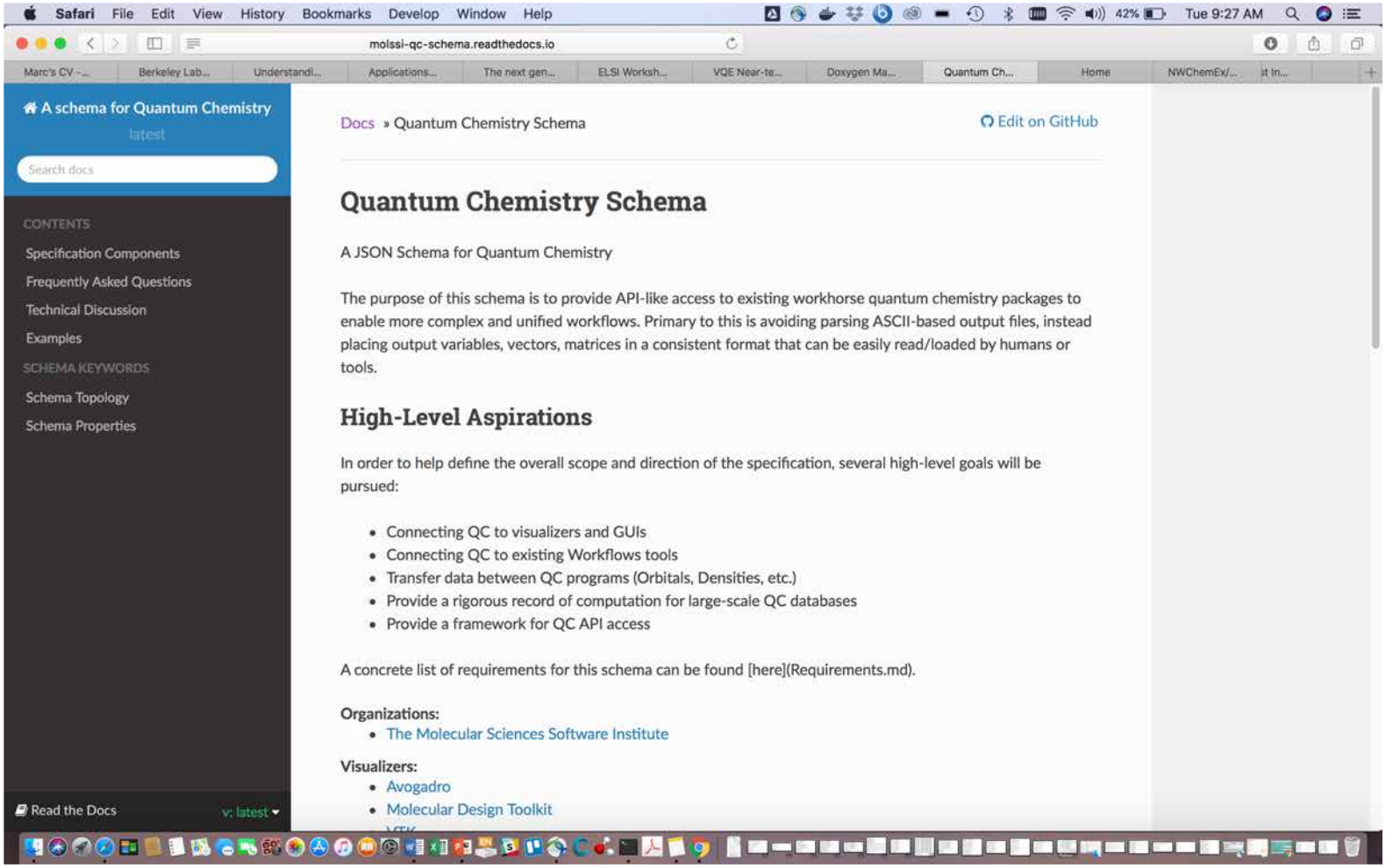
```
bool conv = false;
double E = 0.0;
do {
   auto F =
      run_submodule<FockBuild>("FockMatrix", mol, guess);
   auto new_guess = run_submodule<UpdateDensity>(
      "UpdateGuess", mol, guess);
   auto H("mu", "nu") = h("mu", "nu") + F("mu", "nu");
   auto newE = new_guess("mu", "nu") * H("mu", "nu");
   conv = check_convergance(new_guess, guess, newE, E);
   E = newE;
   guess = new_guess;
} while(!conv);
return E;
```

External interface and data in NWChemEx

- NWChemEx will have a Python front-end
  - Interface with Jupyter Notebooks being developed
  - Interface will be integrated into OpenFermion for quantum computing

- Data will be accessible through HDF5, JSON, and other rich data formats
  - Extensive and scalable user customizable checkpoint restart

Integrating MolSSI effort JSON schema data standard effort



http://molssi-qc-schema.readthedocs.io/en/latest/index.html

BERKELEY LAB

Acknowledgements

The NWChemEx project is supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two organizations – the Office of Science and the National Nuclear Security Administration in the U. S. Department of Energy.

The ECP is responsible for the planning and preparation of a capable exascale ecosystem – including software, applications, hardware, advanced system engineering, and early testbed platforms – to support the nation's exascale computing imperative.

BERKELEY LAB

Bringing Science Solutions to the World

wadejong@lbl.gov

Openings for postdocs in

- **Quantum computing for chemical sciences**
- **Machine learning for biochemical sciences**

Many programs available for undergrad and grad student interns