Computational chemistry at the petascale: tools in the tool box

Robert J. Harrison
harrisonrj@ornl.gov
robert.harrison@utk.edu

The DOE funding

- This work is funded by the U.S. Department of Energy, the division of Basic Energy Science, Office of Science, under contract DE-AC05-00OR22725 with Oak Ridge National Laboratory. This research was performed in part using
  - resources of the National Energy Scientific Computing Center which is supported by the Office of Energy Research of the U.S. Department of Energy under contract DE-AC03-76SF0098,
  - and the Center for Computational Sciences at Oak Ridge National Laboratory under contract DE-AC05-00OR22725
    .

SciDAC

Scientific Discovery through Advanced Computing

Chemical Computations on Future High-end Computers
Award #CHE 0626354, NSF Cyber Chemistry

- NCSA (T. Dunning, PI)
- U. Tennessee (Chemistry, EE/CS)
- U. Illinois UC (Chemistry)
- U. Pennsylvania (Chemistry)
- Conventional single processor performance no longer increasing exponentially
  - Multi-core
- Many “corners” of chemistry will increasingly be best served by non-conventional technologies
  - GP-GPU, FPGA, MD Grape/Wine

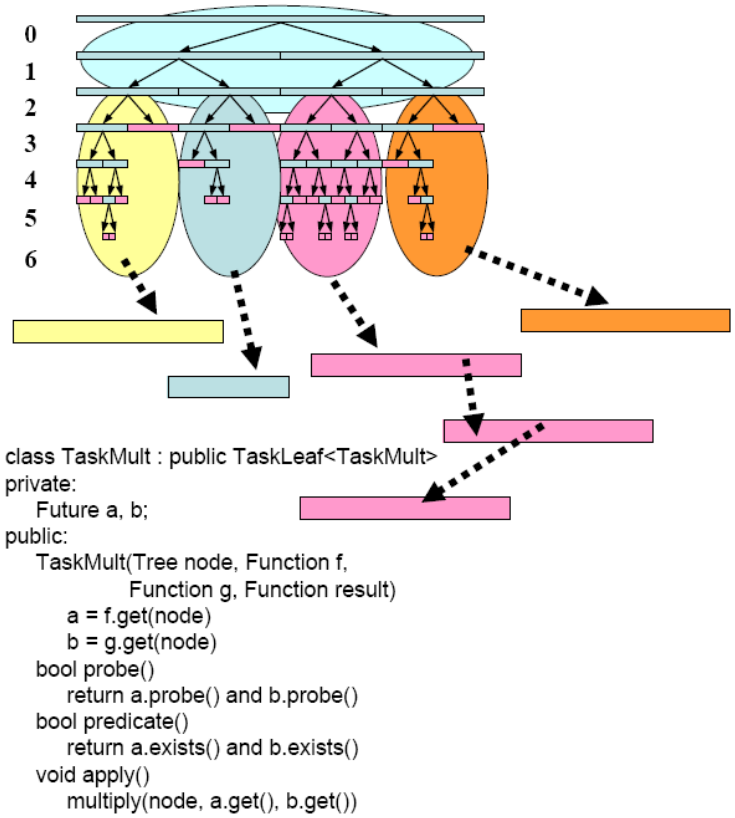
Award No:
NSF CNS-0509410

Project Title:
Collaborative research: CAS-AES: An integrated framework for compile-time/run-time support for multi-scale applications on high-end systems

Investigators:
P. Sadayappan (PI) G. Baumgartner, D. Bernholdt, R.J. Harrison, J. Nieplocha, J. Ramanujam and A. Rountev

Institution:
The Ohio State University
Louisiana State University
University of Tennessee, Knoxville

Website:
http://www.cse.ohio-state.edu/~saday/



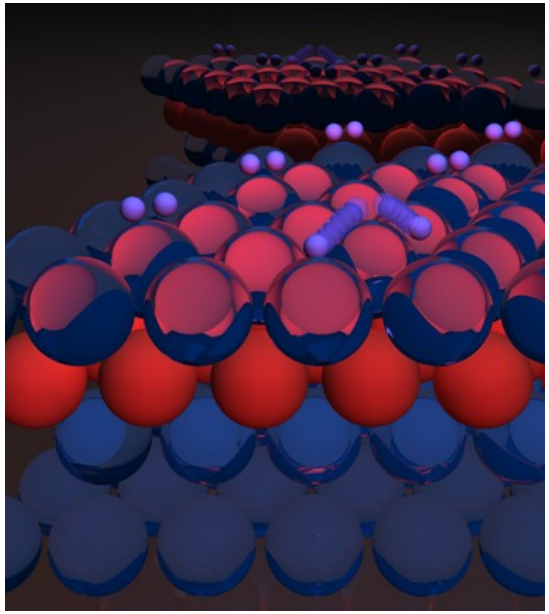
Description of Graphic Image:
A tree-based algorithm (here in 1D, but in actuality in 3-12D) is naturally implemented via recursion. The data in subtrees is managed in chunks and the computation expressed very compactly.

Science Opportunities

- The chemical industry represents 10% of all U.S. manufacturing, employing more than one million Americans.
- These fields require petascale computing coordinated with experimental programs for significant innovation
- Clean energy innovations for automobiles and industry (catalysis, fuel cells, combustion).
  - Catalysts design is presently a misnomer … it is an Edisonian process
  - Only theory+advanced computing can enable rational design
  - Catalytic processes are directly involved in the synthesis of 20% of all industrial products.
- Molecular models for biological processes (protein and membrane function).
  - Modeling excited electronic states in photochemical systems and averaging over dynamics of the macromolecular structure potentially over very long timescales.
- Heavy element chemistry for advanced fuel cycles and environmental restoration.
  - These are special responsibilities of the DOE and petascale chemistry can replace many expensive experiments, and shorten timescales from decades to years.
- Dynamics of atoms and molecules interacting with electrons and powerful laser fields.
  - These are fundamental and defining challenges in physics and chemistry for the 21st century, and have been seeking for a solution for more than 50 years.
  - All major advances in this field are a result of new computational capabilities

Computational catalysis

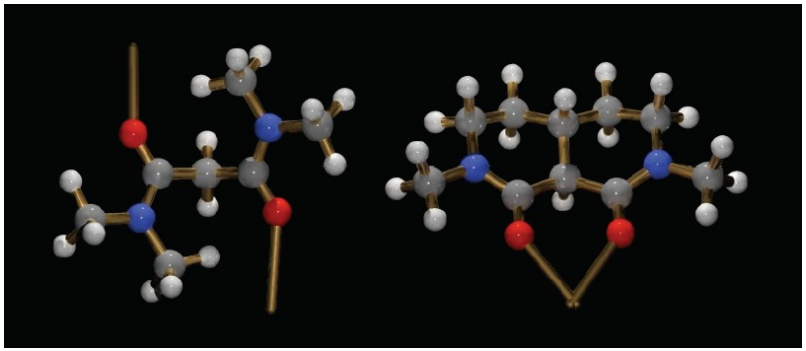
- Currently allocated 6+M hours between NCCS, NERSC, EMSL
  - Need a sustained 10-100x increase
- Approach
  - Large systems accurately described with *modern* hybrid and meta DFT functionals
    - Chemistry codes have the advantage here
    - Full petascale simulation only for the largest runs
  - Higher-accuracy necessary for quantitative rates and for calibration on smaller systems
    - Many-body methods – demonstrated scaling to 10K processors and predicted to go to 100+K
    - 75% of peak speed on EMSL HP cluster 1700 cpu
- Outcomes
  - Rational design of novel catalyst(s)
  - Future savings of $B in various industries
  - Cleaner energy sources



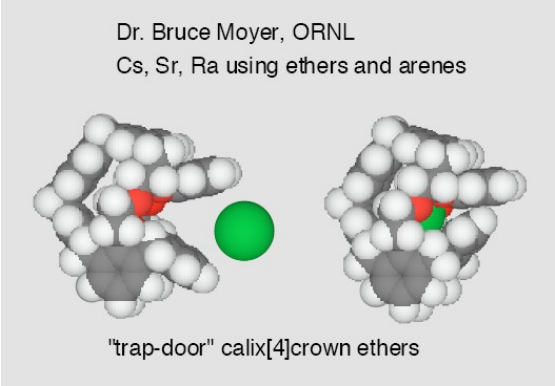
Mavrikakis, Wisconsin. H2 dissociation path on a bimetallic NSA surface.

The role of simulation in heavy element chemistry for advanced fuel cycles

- Molecular-scale knowledge is vital to enable the rational design of new/enhanced agents
  - Reduced cost & risk with increased efficiency
  - Current experimental approach can generate only a fraction of required data over many years
    - The rest are guesstimated.
  - We can compute much of this
    - Need higher precision than currently feasible
  - Combinatorial methods use thermodynamics for screening, but this is not reliable enough
- Approach
  - Mixed MM/QM Gibbs-free energy computations of partition coefficients
  - Simulation of select liquid-liquid, gas-gas interfaces
  - Accurate thermo-chemistry and spectroscopy
    - Many-body methods incorporating relativistic effects
- Outcomes
  - Design of new separation chemistries on a timescale relevant to engineering requirements (months to years rather than decades)



B. Hay, EMSP project 73759



Issues

- Eliminate gulf between theoretical innovation in small groups and realization on high-end computers
- Eliminate the semantic gap so that efficient parallel code is no harder than doing the math
- Hardware/software architecture and programming models for 20xx
- Feasibility, support and maintenance

Major Molecular Electronic Structure Packages

- MOLPRO, MOLCAS, Turbomol, NWChem, GAMESS-US, GAMESS-UK, Gaussian, Jaguar, MPQC, ACES, QChem
- Wide range of functionality and algorithms
- Applicable to much of chemistry
- Vary greatly in speed, robustness, scalability, methods, …
- Several codes often used within any one project

- circa 20M lines of code …

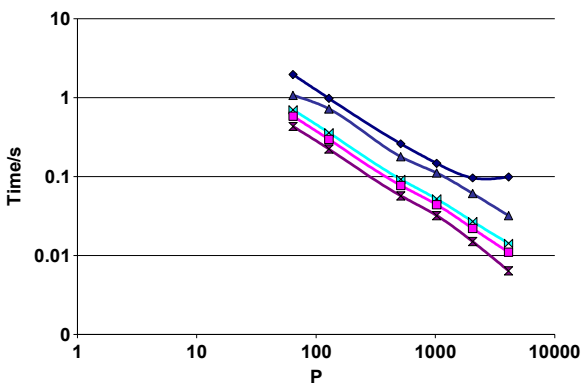
Computational Chemistry Endstation

Open, international collaboration: 8 universities & 5 national labs

- Led out of UT/ORNL
- Focus
  - Actinides, Aerosols, Catalysis
- ORNL Cray XT3, ANL BG/L

Capabilties:

- Chemically accurate thermochemistry
  - Many-body methods required
- Mixed QM/QM/MM dynamics
  - Accurate free-energy integration
  - Simulation of extended interfaces
- Families of relativistic methods



Scaling of MADNESS 64-4096 cpu on XT3

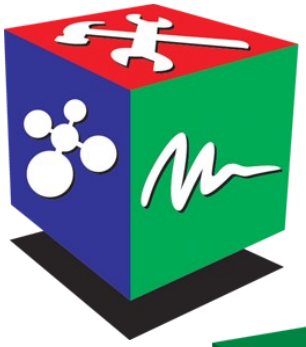
NWChem: Largest CCSD(T) calculation

- Pollack, PNNL/EMSL, 2005.
- 1960 processor Itanium2 cluster
- 1468 basis functions (aug-cc-pVQZ)
- Perturbative triples (T)
  - 23 hours on 1400 processors
  - 75% of peak = 6.3 TFlops.

Trends in Chemistry Codes - I

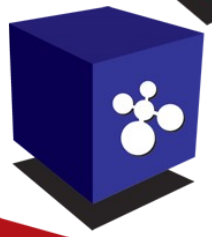
- *All* scalable codes use one-sided communication in various forms
  - Ease of programming; increased scalability
- QM/MM, multi-scale methods, direct dynamics, …
  - Multi-physics methods present scalability challenges
- Multi-level methods (FMM, multi-resolution, multi-grid, mixed-bases) in both space and time
  - A path to (near) linear or optimal scaling without sacrificing accuracy
  - Fully numerical real space methods
- Gaussian bases becoming attractive alternative to plane waves at low/medium precision? (Hütter)

Molecular Science Software Project



MS3
MOLECULAR SCIENCE SOFTWARE SUITE

ECCE
EXTENSIBLE COMPUTATIONAL CHEMISTRY ENVIRONMENT



NWCHEM
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

GA TOOLS
PARALLEL COMPUTING LIBRARIES AND SOFTWARE TOOLS

PNNL
Yuri Alexeev,
Eric Bylaska,
Bert deJong,
Mahin Hackler,
Karol Kowalski,
Lisa Pollack,
Tjerk Straatsma,
Marat Valiev,
Theresa Windus

ORNL
Edo Apra,
Robert Harrison
Vincent Meunier
Ricky Kendall

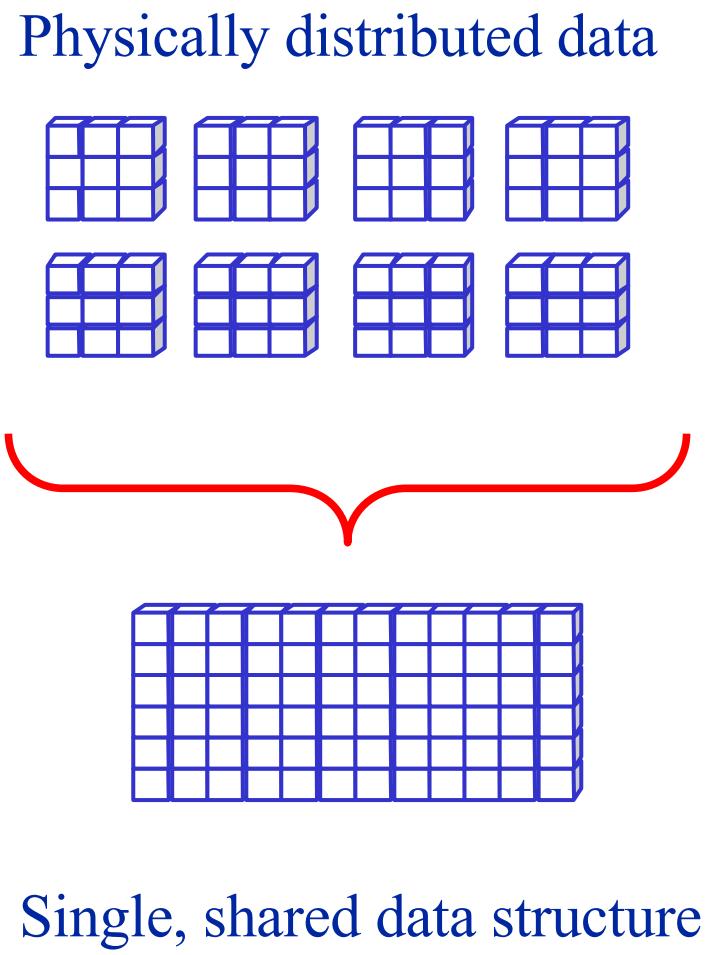
Ames
TL Windus

Manoj Krishnan, Jarek Nieplocha,
Bruce Palmer, Vinod Tipparaju

Gary Black,
Brett Didier,
Todd Elsenthagen,
Sue Havre,
Carina Lansing,
Bruce Palmer,
Karen Schuchardt,
Lisong Sun
Erich Vorpagel

http://www.emsl.pnl.gov/docs/nwchem/nwchem.html

Global Arrays

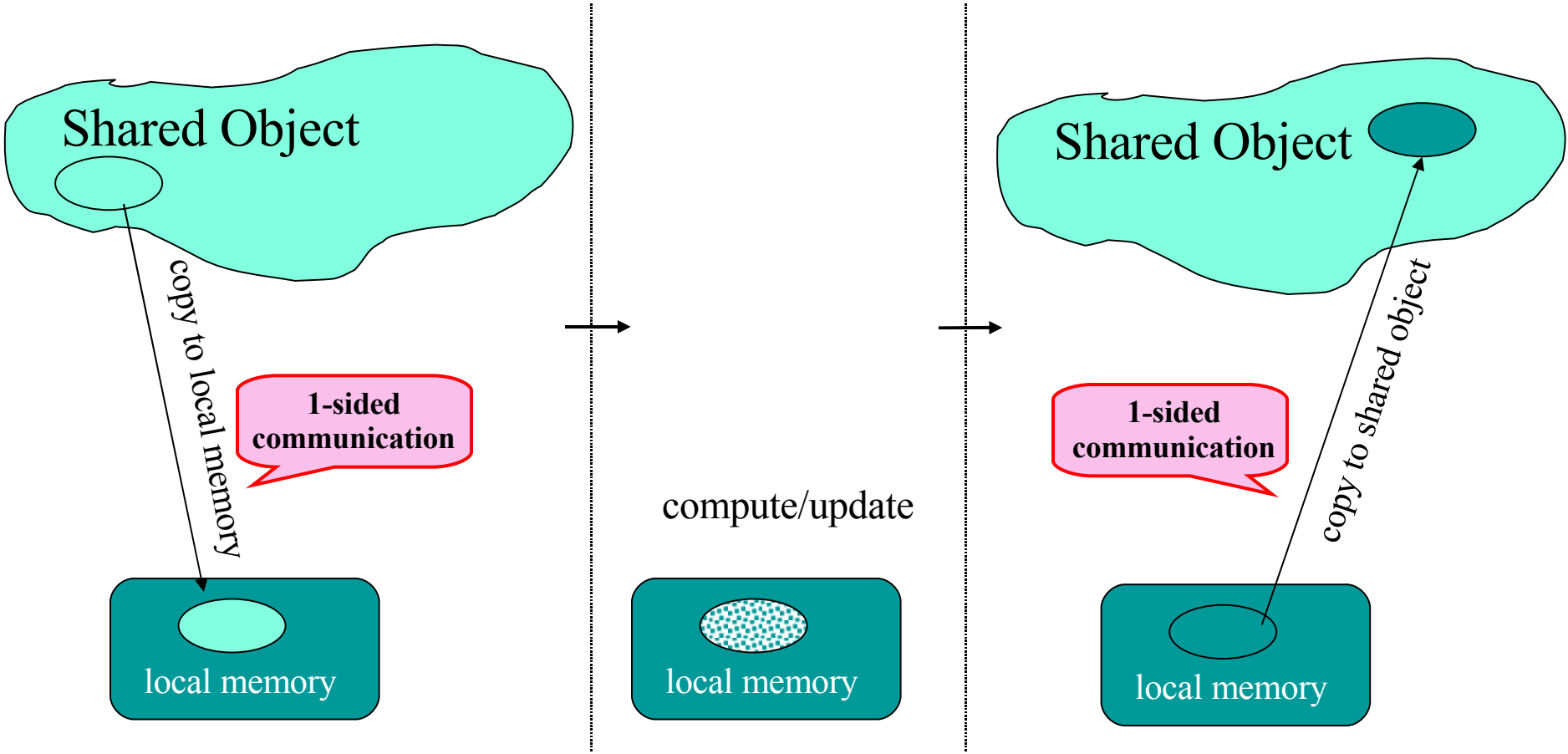


- Shared-memory-like model
  - Fast local access
  - NUMA aware and easy to use
  - MIMD and data-parallel modes
  - Inter-operates with MPI, …
- BLAS and linear algebra interface
- Ported to major parallel machines
  - IBM, Cray, SGI, clusters,...
- Originated in an HPCC project
- Used by most major chemistry codes, financial futures forecasting, astrophysics, computer graphics
- Supported by DOE

- Jarek Nieplocha, PNNL

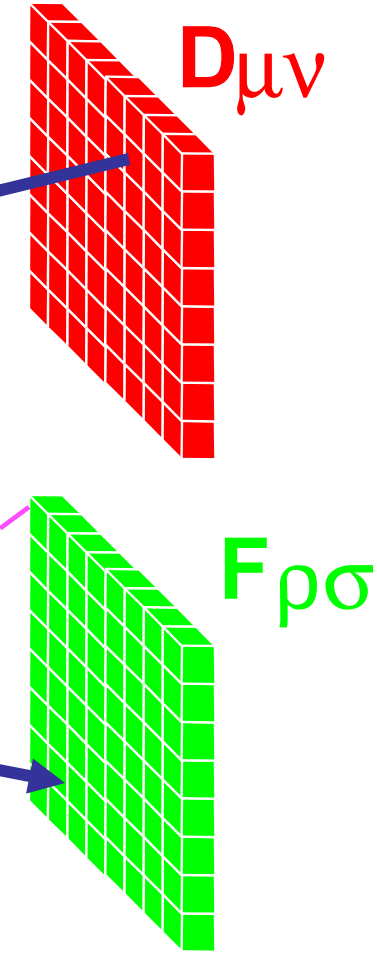
http://www.emsl.pnl.gov/docs/global/

Non-uniform memory access model of computation



Dynamic load balancing

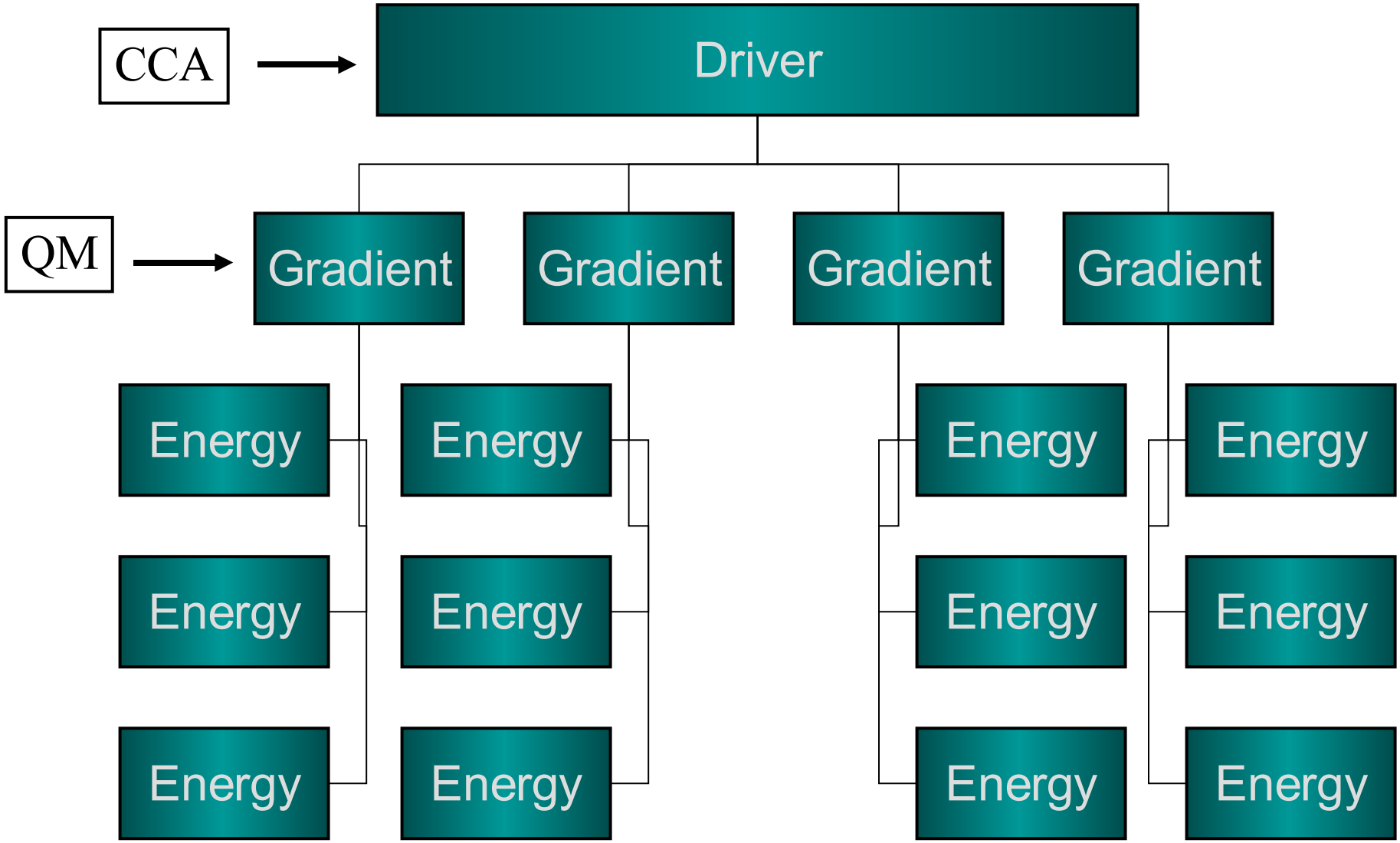
```
While ((task = SharedCounter()) < max)

    call ga_get(                    )

    (do work)

    call ga_acc(                    )
End while
Barrier()
```



Trends in Chemistry Codes - II

- Multi-level parallelism
  - An effective path for most applications to scale to 100K processors without vast effort
    - Coarse grain over vibrational degrees of freedom in numerical hessian, or geometries in a surface scan or parameter study
    - Conventional distributed memory within each subtask
    - Fine grain parallelism within a few processor SMP (multi-threads, OpenMP, parallel BLAS, …)
  - Efficient exploitation of fine grain parallelism is a major concern on future architectures
- MADNESS side steps some of these issues
- Community effort to increase interoperability and leverage new capabilities between codes
  - CCA being adopted

Numerical Hessians



TL Windus, formerly PNNL

Synthesis of High Performance Algorithms for Electronic Structure Calculations

http://www.cis.ohio-state.edu/~gb/TCE

- Collaboration between DOE/SciDAC, NSF/ITR and ORNL/LDRD
- Objective: develop a high level programming tool that translates many-body quantum theory into efficient massively parallel codes. This is anticipated to revolutionize the rate of progress in this field by eliminating man-years of programming effort.
- NSF Project:
- Sadayappan (PI), Baumgartner, Cociorva, Pitzer (OSU)
  Bernholdt, Harrison (unfunded) (ORNL)
  Ramanujam (LSU)
  Nooijen (Waterloo)
- DOE SciDAC: Harrison (PI), Hirata (PNNL)
- DOE ORNL/LDRD: Bernholdt (PI, 2002-3)
- Other SciDAC projects adopting this tool: Piecuch, Gordon
- Also being applied to nuclear physics (Bernholdt and Dean)

CCSD Doubles Equation

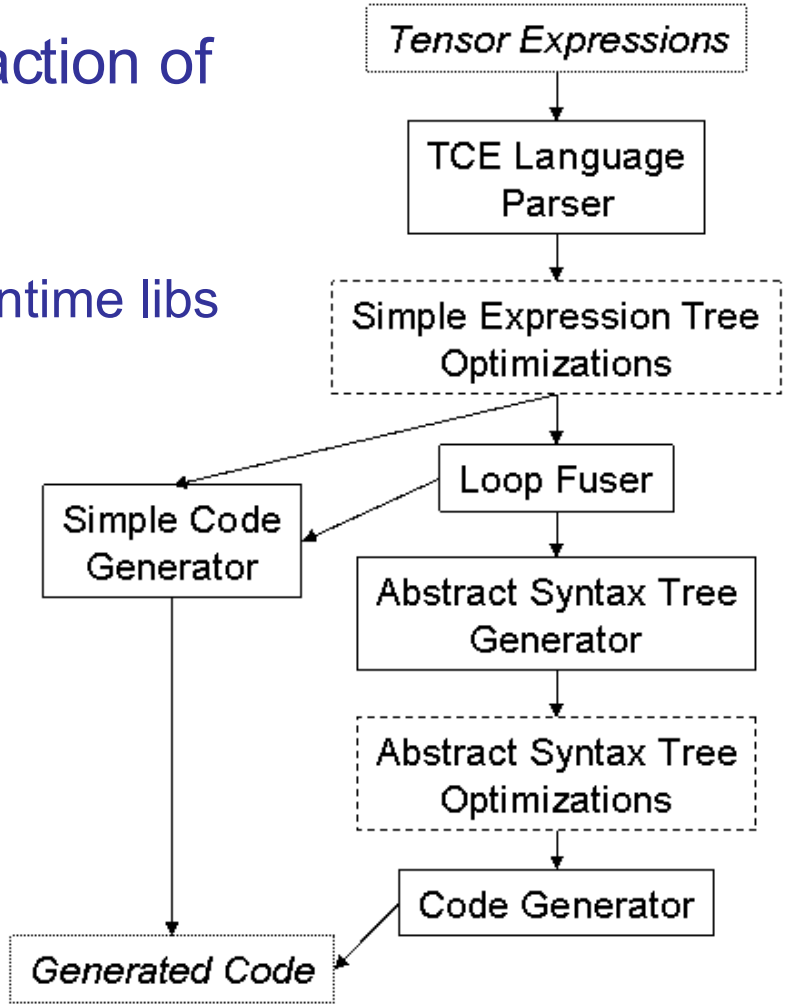
```
hbar[a,b,i,j] == sum[f[b,c]*t[i,j,a,c],{c}] -sum[f[k,c]*t[k,b]*t[i,j,a,c],{k,c}] +sum[f[a,c]*t[i,j,c,b],{c}] -sum[f[k,c]*t[k,a]*t[i,j,c,b],{k,c}]
    -sum[f[k,j]*t[i,k,a,b],{k}] -sum[f[k,c]*t[j,c]*t[i,k,a,b],{k,c}] -sum[f[k,i]*t[j,k,b,a],{k}] -sum[f[k,c]*t[i,c]*t[j,k,b,a],{k,c}]
    +sum[t[i,c]*t[j,d]*v[a,b,c,d],{c,d}] +sum[t[i,j,c,d]*v[a,b,c,d],{c,d}] +sum[t[j,c]*v[a,b,i,c],{c}] -sum[t[k,b]*v[a,k,i,j],{k}]
    +sum[t[i,c]*v[b,a,j,c],{c}] -sum[t[k,a]*v[b,k,j,i],{k}] -sum[t[k,d]*t[i,j,c,b]*v[k,a,c,d],{k,c,d}] -sum
    [t[i,c]*t[j,k,b,d]*v[k,a,c,d],{k,c,d}] -sum[t[j,c]*t[k,b]*v[k,a,c,i],{k,c}] +2*sum[t[j,k,b,c]*v[k,a,c,i],{k,c}] -sum
    [t[j,k,c,b]*v[k,a,c,i],{k,c}] -sum[t[i,c]*t[j,d]*t[k,b]*v[k,a,d,c],{k,c,d}] +2*sum[t[k,d]*t[i,j,c,b]*v[k,a,d,c],{k,c,d}] -sum
    [t[k,b]*t[i,j,c,d]*v[k,a,d,c],{k,c,d}] -sum[t[j,d]*t[i,k,c,b]*v[k,a,d,c],{k,c,d}] +2*sum[t[i,c]*t[j,k,b,d]*v[k,a,d,c],{k,c,d}] -sum
    [t[i,c]*t[j,k,d,b]*v[k,a,d,c],{k,c,d}] -sum[t[j,k,b,c]*v[k,a,i,c],{k,c}] -sum[t[i,c]*t[k,b]*v[k,a,j,c],{k,c}] -sum[t[i,k,c,b]*v[k,a,j,c],{k,c}]
    -sum[t[i,c]*t[j,d]*t[k,a]*v[k,b,c,d],{k,c,d}] -sum[t[k,d]*t[i,j,a,c]*v[k,b,c,d],{k,c,d}] -sum[t[k,a]*t[i,j,c,d]*v[k,b,c,d],{k,c,d}]
    +2*sum[t[j,d]*t[i,k,a,c]*v[k,b,c,d],{k,c,d}] -sum[t[j,d]*t[i,k,c,a]*v[k,b,c,d],{k,c,d}] -sum[t[i,c]*t[j,k,d,a]*v[k,b,c,d],{k,c,d}] -sum
    [t[i,c]*t[k,a]*v[k,b,c,j],{k,c}] +2*sum[t[i,k,a,c]*v[k,b,c,j],{k,c}] -sum[t[i,k,c,a]*v[k,b,c,j],{k,c}]
    +2*sum[t[k,d]*t[i,j,a,c]*v[k,b,d,c],{k,c,d}] -sum[t[j,d]*t[i,k,a,c]*v[k,b,d,c],{k,c,d}] -sum[t[j,c]*t[k,a]*v[k,b,i,c],{k,c}] -sum
    [t[j,k,c,a]*v[k,b,i,c],{k,c}] -sum[t[i,k,a,c]*v[k,b,j,c],{k,c}] +sum[t[i,c]*t[j,d]*t[k,a]*t[l,b]*v[k,l,c,d],{k,l,c,d}] -2*sum
    [t[k,b]*t[l,d]*t[i,j,a,c]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[k,a]*t[l,d]*t[i,j,c,b]*v[k,l,c,d],{k,l,c,d}]
    +sum[t[k,a]*t[l,b]*t[i,j,c,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[j,c]*t[l,d]*t[i,k,a,b]*v[k,l,c,d],{k,l,c,d}] -2*sum
    [t[j,d]*t[l,b]*t[i,k,a,c]*v[k,l,c,d],{k,l,c,d}] +sum[t[j,d]*t[l,b]*t[i,k,c,a]*v[k,l,c,d],{k,l,c,d}] -2*sum
    [t[i,c]*t[l,d]*t[j,k,b,a]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,c]*t[l,a]*t[j,k,b,d]*v[k,l,c,d],{k,l,c,d}]
    +sum[t[i,c]*t[l,b]*t[j,k,d,a]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,k,c,d]*t[j,l,b,a]*v[k,l,c,d],{k,l,c,d}]
    +4*sum[t[i,k,a,c]*t[j,l,b,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,k,c,a]*t[j,l,b,d]*v[k,l,c,d],{k,l,c,d}] -2*sum
    [t[i,k,a,b]*t[j,l,c,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,k,a,c]*t[j,l,d,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,k,c,a]*t[j,l,d,b]*v[k,l,c,d],{k,l,c,d}]
    +sum[t[i,c]*t[j,d]*t[k,l,a,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,j,c,d]*t[k,l,a,b]*v[k,l,c,d],{k,l,c,d}] -2*sum
    [t[i,j,c,b]*t[k,l,a,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,j,a,c]*t[k,l,b,d]*v[k,l,c,d],{k,l,c,d}] +sum[t[j,c]*t[k,b]*t[l,a]*v[k,l,c,i],{k,l,c}]
    +sum[t[l,c]*t[j,k,b,a]*v[k,l,c,i],{k,l,c}] -2*sum[t[l,a]*t[j,k,b,c]*v[k,l,c,i],{k,l,c}] +sum[t[l,a]*t[j,k,c,b]*v[k,l,c,i],{k,l,c}] -2*sum
    [t[k,c]*t[j,l,b,a]*v[k,l,c,i],{k,l,c}] +sum[t[k,a]*t[j,l,b,c]*v[k,l,c,i],{k,l,c}] +sum[t[k,b]*t[j,l,c,a]*v[k,l,c,i],{k,l,c}]
    +sum[t[j,c]*t[l,k,a,b]*v[k,l,c,i],{k,l,c}] +sum[t[i,c]*t[k,a]*t[l,b]*v[k,l,c,j],{k,l,c}] +sum[t[l,c]*t[i,k,a,b]*v[k,l,c,j],{k,l,c}] -2*sum
    [t[l,b]*t[i,k,a,c]*v[k,l,c,j],{k,l,c}] +sum[t[l,b]*t[i,k,c,a]*v[k,l,c,j],{k,l,c}] +sum[t[i,c]*t[k,l,a,b]*v[k,l,c,j],{k,l,c}]
    +sum[t[j,c]*t[l,d]*t[i,k,a,b]*v[k,l,d,c],{k,l,c,d}] +sum[t[j,d]*t[l,b]*t[i,k,a,c]*v[k,l,d,c],{k,l,c,d}]
    +sum[t[j,d]*t[l,a]*t[i,k,c,b]*v[k,l,d,c],{k,l,c,d}] -2*sum[t[i,k,c,d]*t[j,l,b,a]*v[k,l,d,c],{k,l,c,d}] -2*sum
    [t[i,k,a,c]*t[j,l,b,d]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,c,a]*t[j,l,b,d]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,a,b]*t[j,l,c,d]*v[k,l,d,c],{k,l,c,d}]
    +sum[t[i,k,c,b]*t[j,l,d,a]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,a,c]*t[j,l,d,b]*v[k,l,d,c],{k,l,c,d}] +sum[t[k,a]*t[l,b]*v[k,l,i,j],{k,l}]
    +sum[t[k,l,a,b]*v[k,l,i,j],{k,l}] +sum[t[k,b]*t[l,d]*t[i,j,a,c]*v[l,k,c,d],{k,l,c,d}] +sum[t[k,a]*t[l,d]*t[i,j,c,b]*v[l,k,c,d],{k,l,c,d}]
    +sum[t[i,c]*t[l,d]*t[j,k,b,a]*v[l,k,c,d],{k,l,c,d}] -2*sum[t[i,c]*t[l,a]*t[j,k,b,d]*v[l,k,c,d],{k,l,c,d}]
    +sum[t[i,c]*t[l,a]*t[j,k,d,b]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,j,c,b]*t[k,l,a,d]*v[l,k,c,d],{k,l,c,d}]
    +sum[t[i,j,a,c]*t[k,l,b,d]*v[l,k,c,d],{k,l,c,d}] -2*sum[t[l,c]*t[i,k,a,b]*v[l,k,c,j],{k,l,c}] +sum[t[l,b]*t[i,k,a,c]*v[l,k,c,j],{k,l,c}]
    +sum[t[l,a]*t[i,k,c,b]*v[l,k,c,j],{k,l,c}] +v[a,b,i,j]
```

Tensor Contraction Engine (TCE)

- High-level domain-specific language for a class of problems in quantum chemistry/physics based on contraction of large multi-dimensional tensors
- Specialized optimizing compiler
  - Produces F77+GA code, linked to runtime libs

```
range V = 3000;
range O = 100;

index a,b,c,d,e,f : V;
index i,j,k,l : O;

mlimit = 100GB;

procedure P(in A[V,V,O,O], in B[V,V,V,O],
            in C[V,V,O,O], in D[V,V,V,O],
            out S[V,V,O,O])=
begin
  S[a,b,i,j] == sum[ A[a,c,i,k] * B[b,e,f,l]
                   * C[d,f,j,k] * D[c,d,e,l],
                     {c,e,f,k,l}];
end
```

$$S_{abij} = \sum_{cefkl} A_{acik} B_{befl} C_{dfjk} D_{cdel}$$



NWChem CCSD(T) performance

- Performed with NWChem by EMSL staff
- Closed-shell CCSD(T) calculations
  - Hand-written code
  - Octane (1468 basis functions, aug-cc-pVQZ) [Pollack, 2005].
    - Perturbative triples (T) took 23 hours on 1400 processors, yielding 75% CPU efficiency and a sustained performance of 6.3 TFlops. Fourteen iterations were required for convergence of the CCSD, which took approximately 43 hours on 600 processors.
  - Water cluster
    - Perturbative triples (T) took 28 hours on 1840 processors, yielding 84% CPU efficiency and a sustained performance of 11.04 TF.
- Large open-shell CCSD(T)
  - TCE machine-generated code
  - CF3CHFO* (605 basis functions, aug-cc-pVQZ)
  - 19 hours with 1024 processors
- Example target system for petaflop
  - $W_3O_9^-$ which has 678 basis functions with the aug-cc-pVTZ/ECP(W) basis set and close to 1400 basis functions with the aug-cc-pVTZ/ECP(W) basis set
  - These are single point calculations and geometry optimizations and frequency calculations are even more costly yet necessary.

MADNESS

Multiresolution
Adaptive
Numerical
Scientific
Simulation

Multiresolution Adaptive Numerical Scientific Simulation

Ariana Beste[1], George I. Fann[1], Robert J. Harrison[1,2], Rebecca Hartman-Baker[1], Shinichiro Sugiki[1]

[1]Oak Ridge National Laboratory

[2]University of Tennessee, Knoxville

In collaboration with

Gregory Beylkin[4], Fernando Perez[4], Lucas Monzon[4], Martin Mohlenkamp[5] and others

[4]University of Colorado

[5]Ohio University

harrisonrj@ornl.gov

SciDAC Scientific Discovery through Advanced Computing

UT-BATTELLE Oak Ridge National Laboratory

Multiresolution chemistry objectives

- Complete elimination of the basis error
  - One-electron models (e.g., HF, DFT)
  - Pair models (e.g., MP2, CCSD, …)
- Correct scaling of cost with system size
- General approach
  - Readily accessible by students and researchers
  - Higher level of composition
  - Direct computation of chemical energy differences
- New computational approaches
- *Fast algorithms with guaranteed precision*

Essential techniques for fast computation

- Multiresolution

$$V_0 \quad V_1 \quad \mathsf{L} \quad V_n$$

$$V_n = V_0 + \left(V_1 - V_0\right) + \mathsf{L} + \left(V_n - V_{n-1}\right)$$

- Low-separation rank

$$f(x_1, \mathsf{K}, x_d) = \sum_{l=1}^{M} \sigma_l \prod_{i=1}^{d} f_i^{(l)}(x_i) + O(\varepsilon)$$

$$\left\| f_i^{(l)} \right\|_2 = 1 \qquad \sigma_l > 0$$

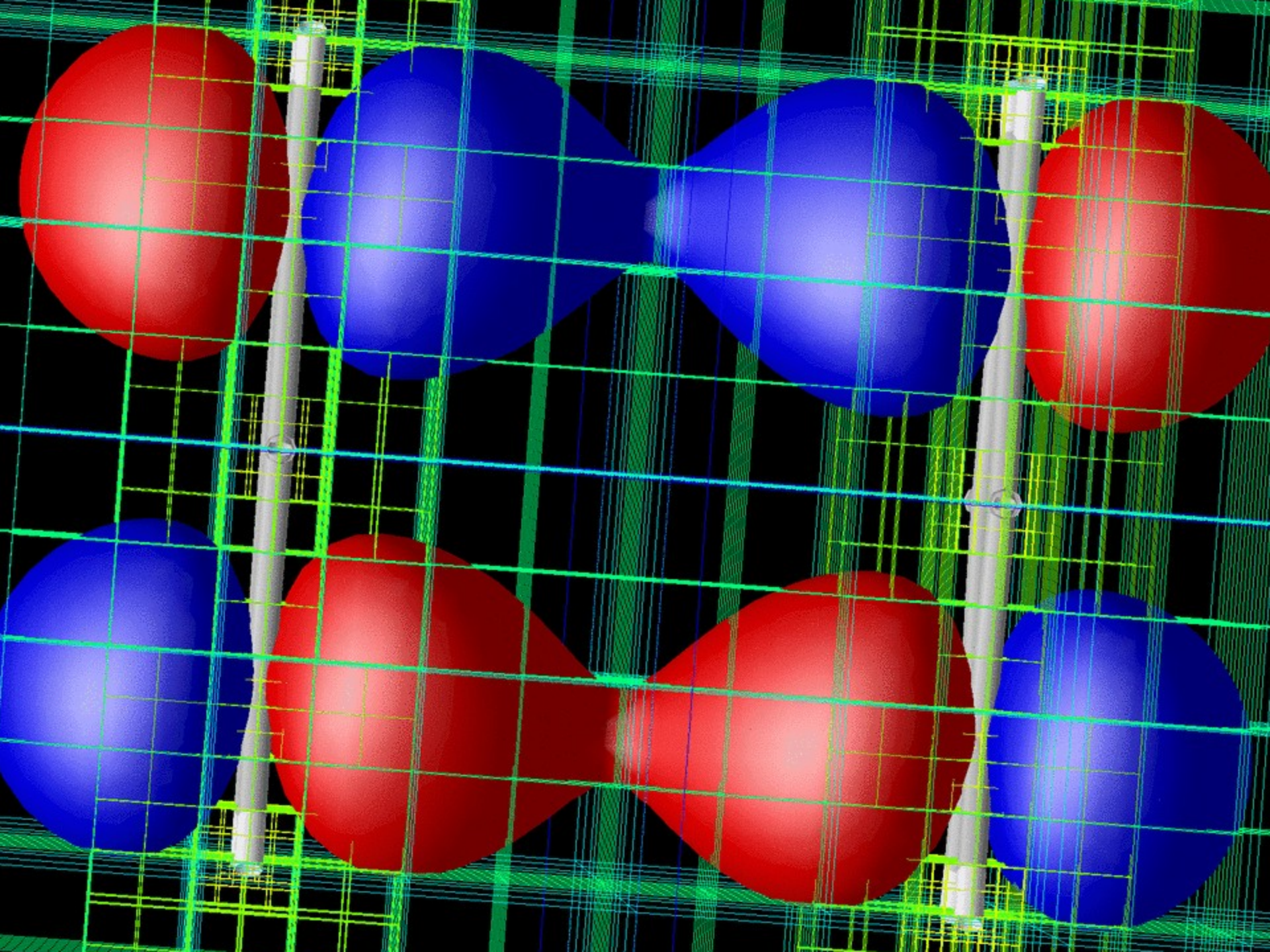
- Low-operator rank

$$A = \sum_{\mu=1}^{r} \underline{u}_\mu \sigma_\mu \underline{v}_\mu^T + O(\varepsilon)$$

$$\sigma_\mu > 0 \qquad \underline{v}_\mu^T . \underline{v}_\lambda = \underline{u}_\mu^T . \underline{u}_\lambda = \delta_{\mu\lambda}$$

Please forget about wavelets

- They are not central
- Wavelets are a convenient basis for spanning V_n-V_{n-1} and understanding its properties
- But you don't actually need to use them
  - MADNESS does still compute wavelet coefficients, but *Beylkin's new code does not*
- Please remember this …
  - Discontinuous spectral element with multi-resolution and separated representations for fast computation with guaranteed precision.



Current applications

- DFT & HF for electrons
  - Energies, gradients, excitation energies, non-linear optical properties, …
  - Catalysis, nano-scale chemistry
- MCSCF
  - Time evolution of few-electron systems
- 6D prototype
  - Exact solution of 2-electron problem without use of any special symmetry
- Nuclear structure

High-level composition using functions and operators

- Conventional quant. chem. uses explicitly indexed sparse arrays of matrix elements
  - Complex, tedious and error prone
- Python classes for Function and Operator
  - in 1,2,3,6 and general dimensions
  - wide range of operations

```
Hpsi = -0.5*Delsq*psi+ V*psi
J = Coulomb.apply(rho)
```

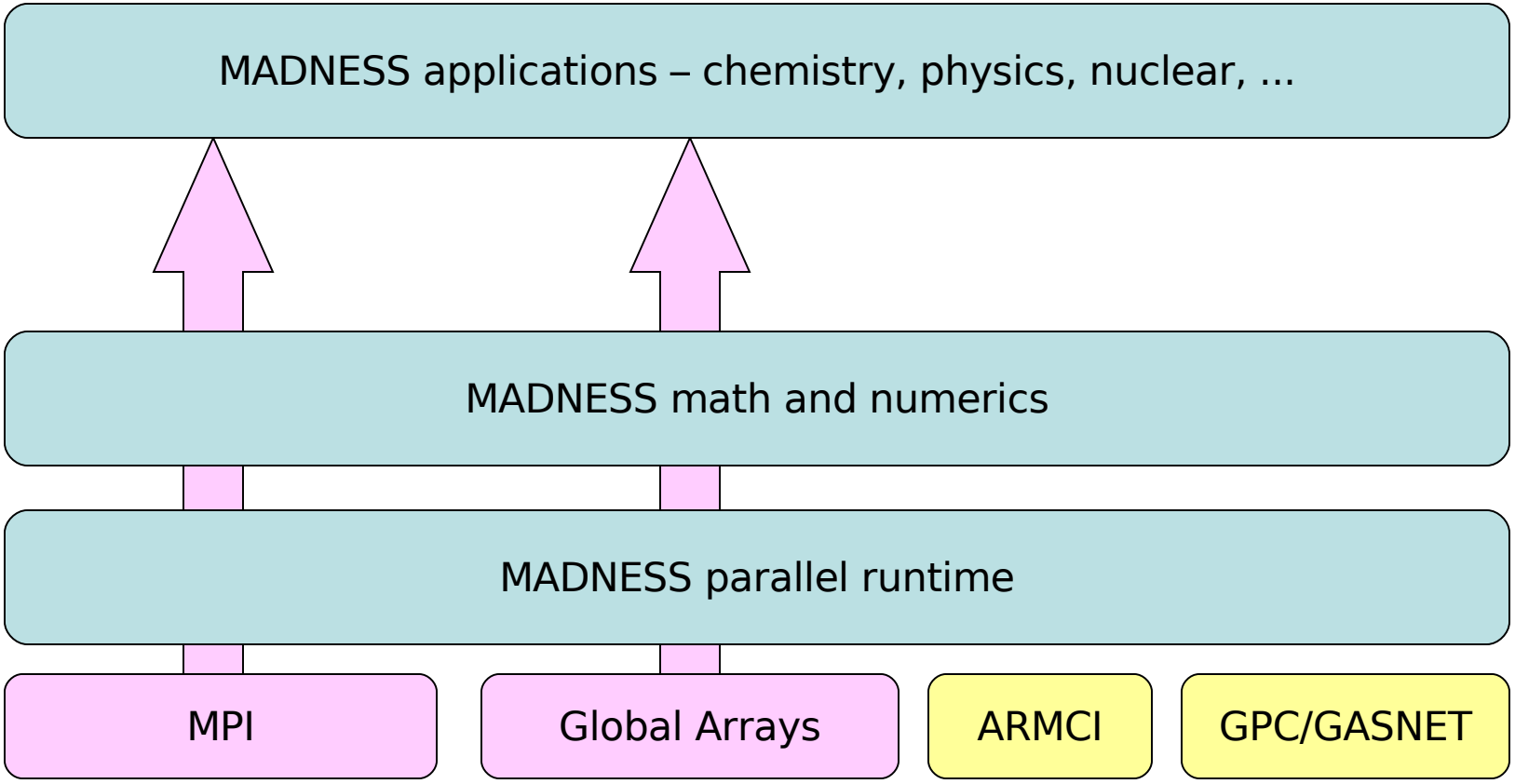
$$H\varphi = -\frac{1}{2} \quad {}^2\varphi + V\varphi$$

$$J(r) = G * \rho$$

$$= \frac{\rho(s)}{|r-s|} ds$$

- All with guaranteed speed and precision

MADNESS architecture



Runtime Objectives

- Scalability to 1+M processors ASAP
- Runtime responsible for
  - scheduling and placement,
  - managing data dependencies,
  - hiding latency, and
  - Medium to coarse grain concurrency
- Compatible with existing models
  - MPI, Global Arrays
- Borrow successful concepts from Cilk, Charm++, Python
- Anticipating next gen. languages

Key elements

- Futures for hiding latency and automating dependency management
- Global names and name spaces
- Non-process centric computing
  - One-sided messaging between objects
  - Retain place=process for MPI legacy
- Dynamic load balancing

Futures

- Result of an asynchronous computation
  - Cilk, Java, HPCLs
- Hide latency due to communication or computation
- Management of dependencies
  - Via callbacks

```
int f(int arg);
ProcessId me, p;

Future<int> r0=task(p, f, 0);
Future<int> r1=task(me, f, r0);

// Work until need result

cout << r0 << r1 << endl;
```

Process “me” spawns a new task in process “p” to execute `f(0)` with the result eventually returned as the value of future `r0`. This is used as the argument of a second task whose execution is deferred until its argument is assigned. Tasks and futures can register multiple local or remote callbacks to express complex and dynamic dependencies.

Global Names

- Objects with global names with different state in each process
  - C.f. shared[threads] in UPC; co-Array

```
class A : public WorldObject<A>{
   int f(int);
};
ProcessID p;
A a;
Future<int> a.task(p,&A::f,0);
```

- Non-collective constructor; deferred destructor
  - Eliminates synchronization

A task is sent to the instance of a in process p. If this has not yet been constructed the message is stored in a pending queue. Destruction of a global object is deferred until the next user synchronization point.

Global Namespaces

- Specialize global names to containers
  - Hash table done
  - Arrays, etc., planned
- Replace global pointer (process+local pointer) with more powerful concept
- User definable map from keys to “owner” process

```
class Index;  // Hashable
class Value {
   double f(int);
};

WorldContainer<Index,Value> c;
Index i,j;  Value v;
c.insert(i,v);
Future<double> =
   c.task(j,&Value::f,666);
```

A container is created mapping indices to values.

A value is inserted into the container.

A task is spawned in the process owning key `j` to invoke `c[j].f(666)`.

Namespaces are a large part of the elegance of Python and success of Charm++ (chares+arrays)

Abstraction Overheads

- If you are careful you win
  - *Increased performance and productivity*
  - This is the lesson of Global Arrays, Charm++, …
- Creating, executing, reaping a local, null task – 350us (100K tasks, 3GHz Core2, Pathscale 3.0, -Ofast) dominated by new/delete
- Chain of 100K dependent tasks with the result of a task as the unevaluated argument of the previous task
  - ~1 us per task
- Creating a remote task adds overhead of inter-process communication which is on the scale of 3us (Cray XT).
  - Aggregation can reduce this.
- Switching between user-space threads <20ns

Work in progress

- Multiple kernel threads for multi-core
- Full integration of Global Arrays
- Port to GPC and/or GASNET
- Interfaces to external solvers
- Dynamic load balance via work stealing
- Planned
  - Multiple user threads per kernel thread to reduce need for manual closures
  - Near-optimal global scheduling via DAG
  - Interfaces from C++ to Python and Octave