# 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

# 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.

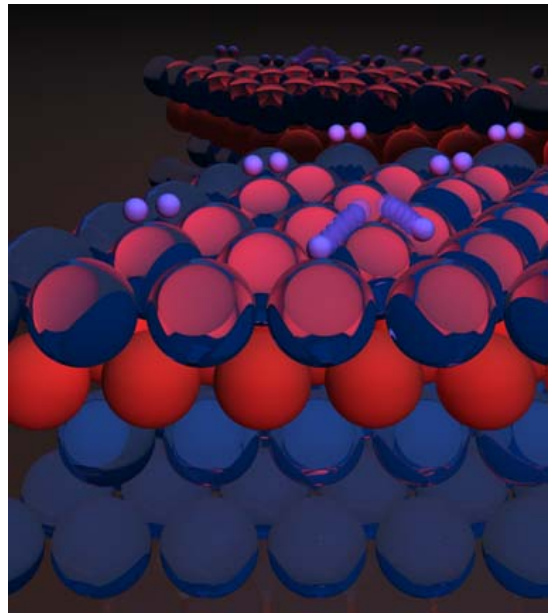
- Molecular models for biological processes (protein and membrane function).
  - Modeling excited electronic states in photochemical systems and averaging over dynamics of the macromolecular structure potentially over very long timescales.

- 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

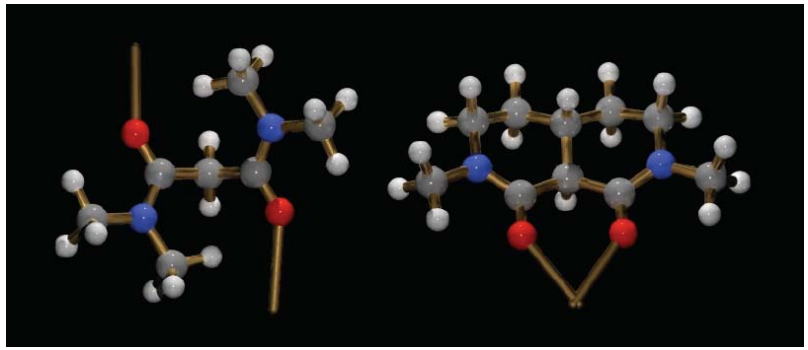
- Currently allocated 5+M hours between NCCS, NERSC, EMSL
  - Need a sustained 10-100x increase
- Approach
  - Large systems accurately described with *modern* hybrid and meta DFT functionals
    - Chemistry codes have the advantage here
    - Full petascale simulation only for the largest runs
  - Higher-accuracy 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



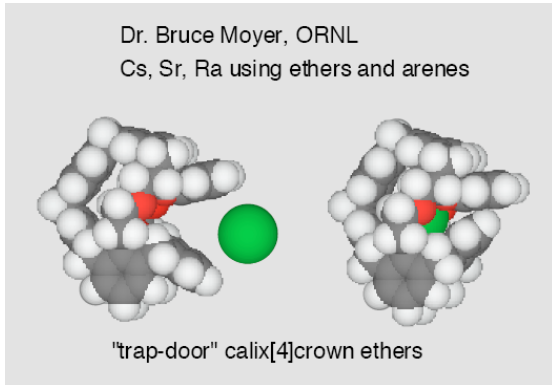
*Mavrikakis*, Wisconsin. H2 dissociation path on a bimetallic NSA surface.

# 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



# Computing now …

- The death of sequential computing
- Does anyone in the room still have a single cpu desktop machine?
- Laptop?
- Cell phone?

# Computing in 2022 - I

- Looking back from 2007 to 1992
  - About 500x increase in desktop performance
    - 100MHz to 2x2 3GHz Core2
    - 30x from clock, 8x from parallelism
  - About 2500x increase supercomputer speed
    - 100GF to 250TF
    - 30x from clock, 40x from parallelism

# Computing in 2022 - II

- Looking forward to 2022
  - Expect same performance increases
  - Almost entirely from increased parallelism
  - Custom devices with much higher speed
  - Memory and I/O hierarchy much deeper

  - 20K * 2500 = 500M “processors”

# Parallel scalability of NWChem - I

- Typical parallel efficiency of current largest jobs running on circa 1K processors
  - DFT – 50% ?
  - CCSD(T) - 80% ?
- Assume Amdahl's law
  - Severe assumption
  - DFT r=.001
  - CCSD(T) r=0.00025
- Efficiency on 10K cpu
  - DFT 9%, CCSD(T) 29%

$$T(P) = t_{seq} + \frac{t_{par}}{P} \quad \text{Amdahl's law}$$

$$r = \frac{t_{seq}}{t_{par}}$$

$$\varepsilon(P) = \frac{T(1)}{PT(P)} \quad \text{Definition}$$

$$= \frac{1+r}{1+Pr}$$

$$r = \frac{1-\varepsilon(P)}{P\varepsilon(P)-1}$$

# Parallel scalability of NWChem - II

- Actual results running on 10K processors

# Parallel scalability of NWChem - III

- Actual results running on 100K processors

# 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
- Software architecture and programming models for 2022
- Becoming a real community code
- Feasibility, support and maintenance

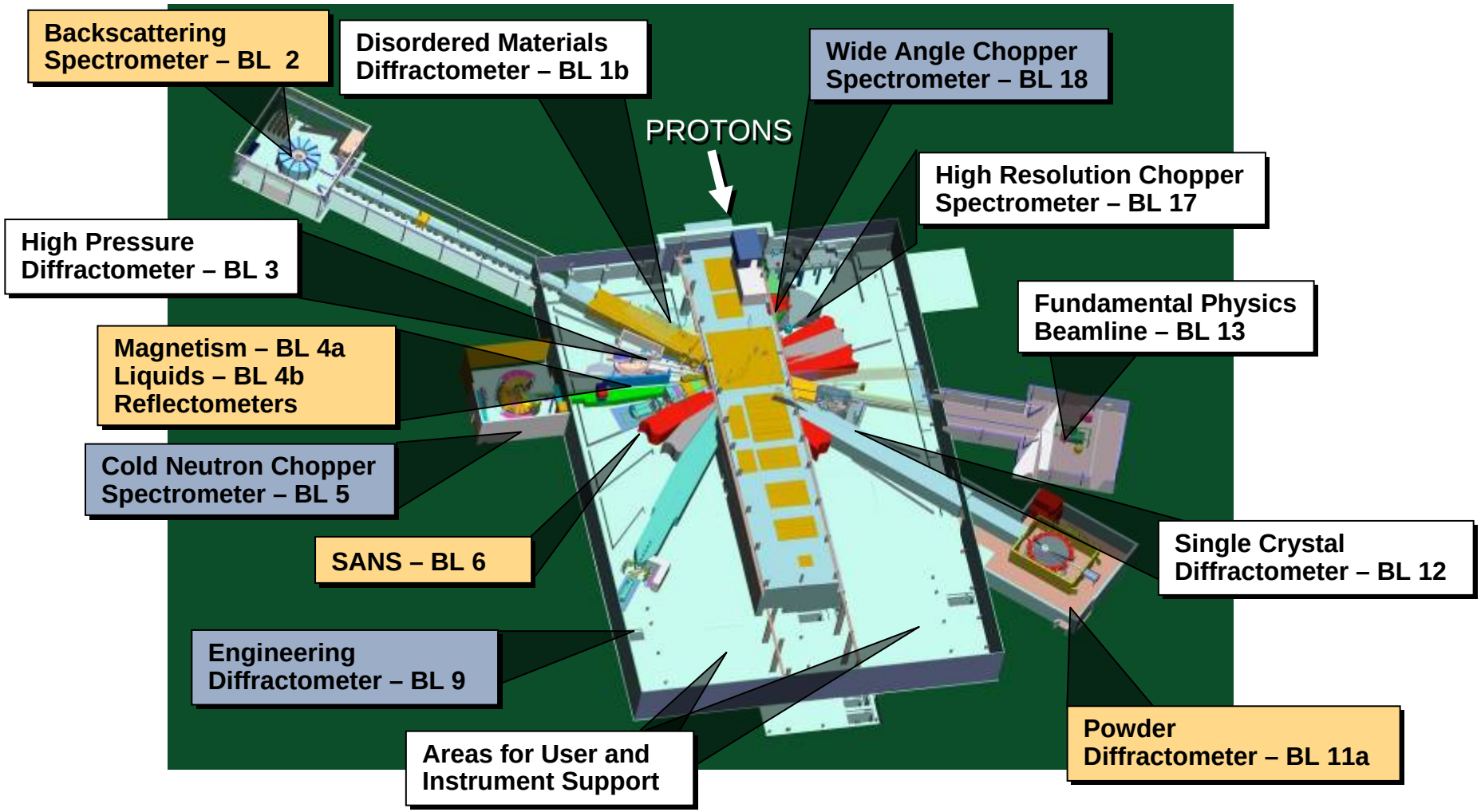
# The way forward demands a change in paradigm

## - by us chemists, the funding agencies, and the supercomputer centers

- A communal effort recognizing the increased cost and complexity of code development for modern theory at the petascale

- Re-emphasizing basic and advanced theory and computational skills in undergraduate and graduate education

# An analogy:
# Experimental End Stations at SNS

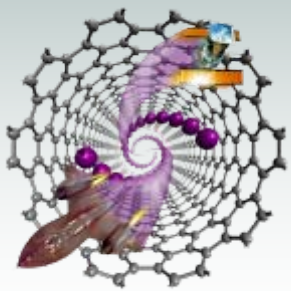


# Definition:
# Computational End Station

**NLCF deploys a fundamentally new approach for long-term engagement of research communities modeled on the "end station" concept through which major experimental facilities provide specialized instruments to specific user groups**

## End Station defined by three characteristics:

1. **Addresses problems that are of national importance (e.g., nanotech)**



National Problem

+

2. **Scientific team willing to create and maintain end station**

Scientific team

+

3. **Suite of scientific codes in area tuned to NLCF resources**

Application suite

# Molecular Science Software Project

**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**

MS3
MOLECULAR SCIENCE SOFTWARE SUITE

ECCE
EXTENSIBLE COMPUTATIONAL CHEMISTRY ENVIRONMENT

NWCHEM
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

GA TOOLS
PARALLEL COMPUTING LIBRARIES AND SOFTWARE TOOLS

**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

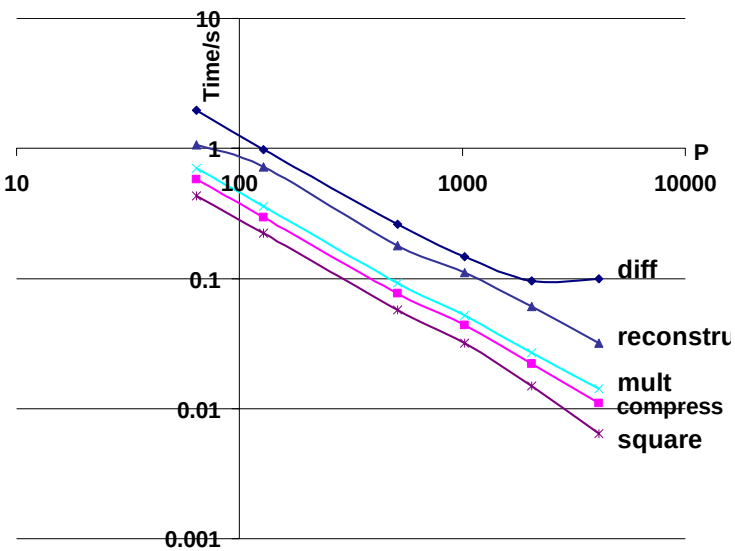
# Computational Chemistry Endstation

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

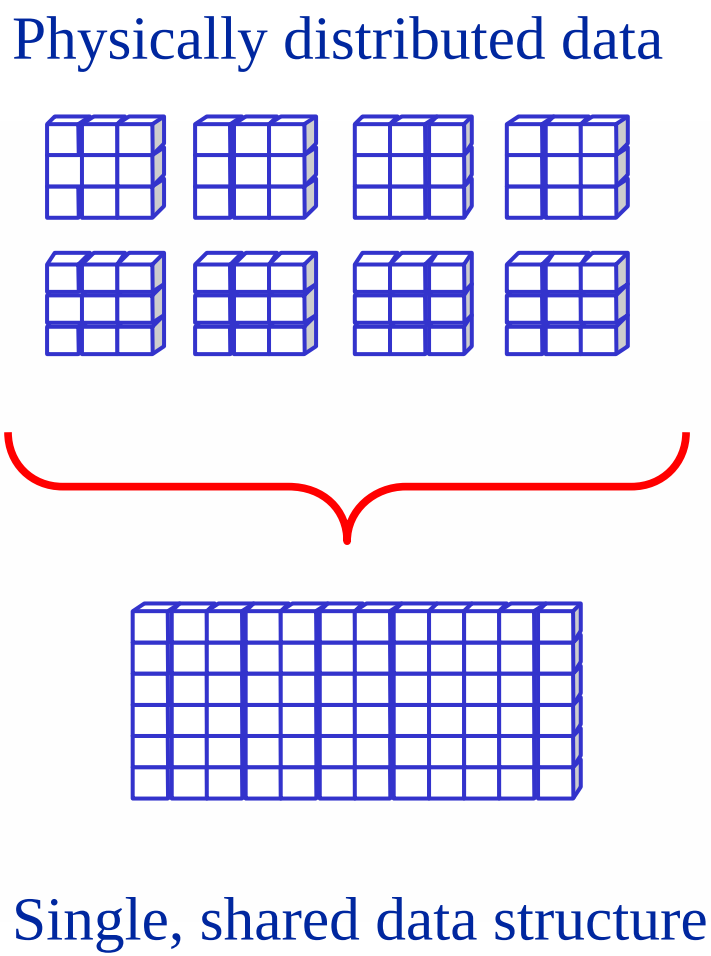
NWChem: Largest CCSD(T) calculation

- 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)

# Global Arrays

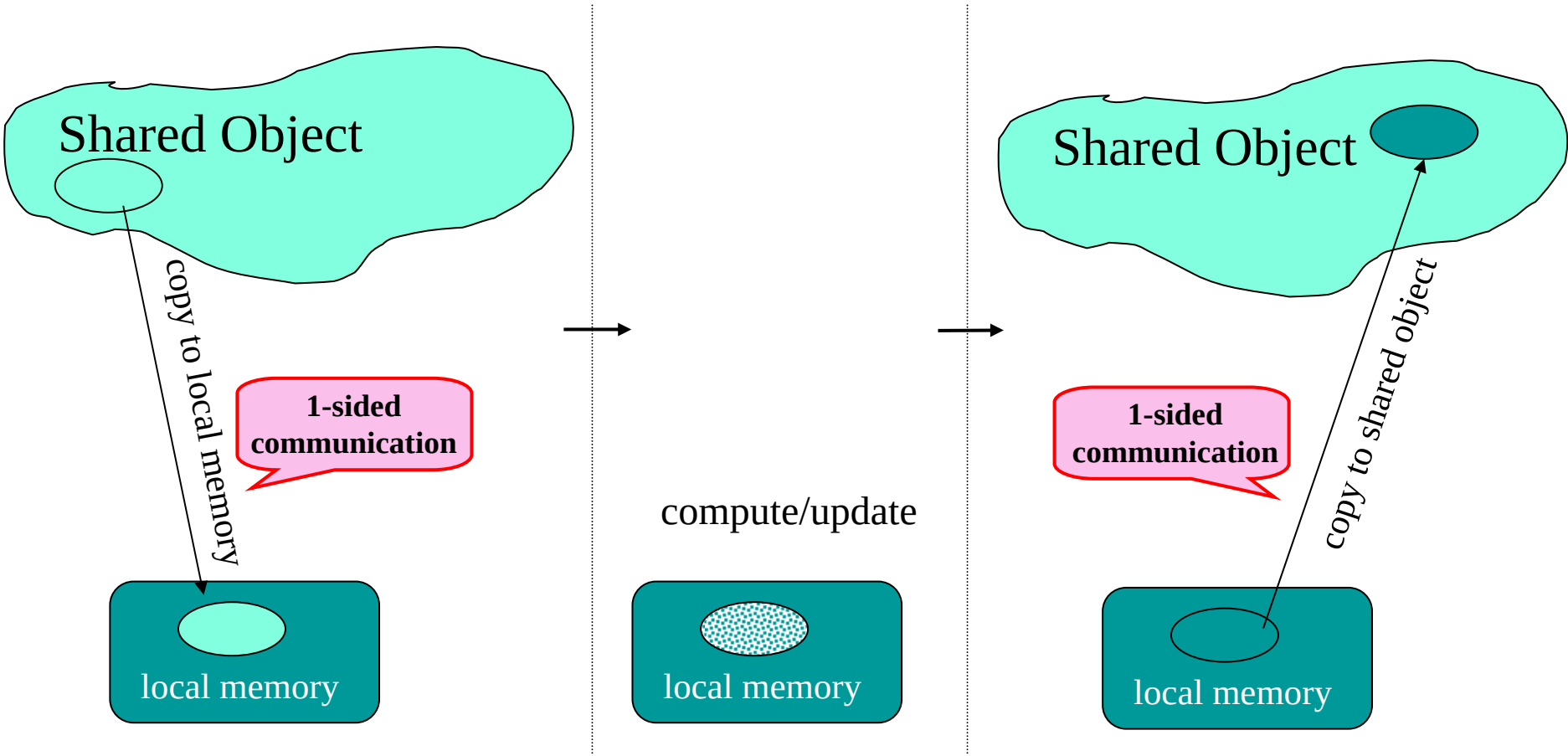



- Shared-memory-like model
  - Fast local access
  - NUMA aware and easy to use
  - MIMD and data-parallel modes
  - Inter-operates with MPI, …
- BLAS and linear algebra interface
- Ported to major parallel machines
  - IBM, Cray, SGI, clusters,...
- Originated in an HPCC project
- Used by most major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

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

- Jarek Nieplocha, PNNL

# Non-uniform memory access model of computation



# 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

# 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

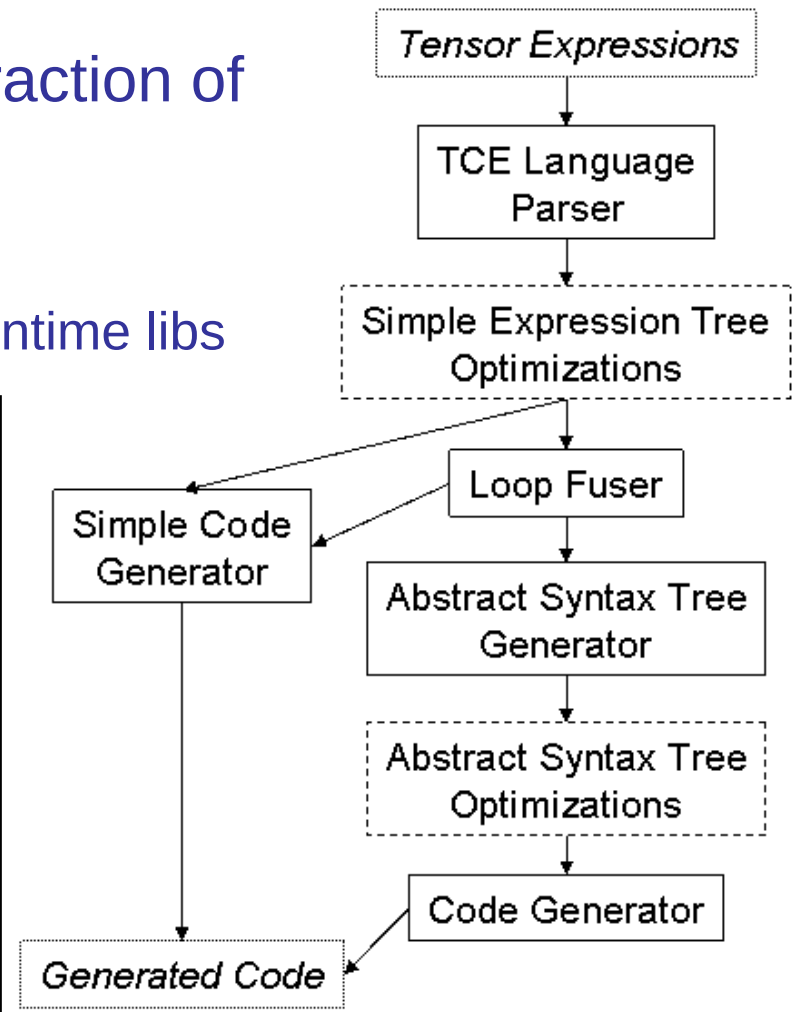
```
range V = 3000;
range O = 100;

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
```

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



# 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

$$V_0 \subset V_1 \subset \cdots \subset V_n$$

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

- Low-separation rank

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

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

- Low-operator rank

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

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

# How to “think” multiresolution

- Consider a ladder of function spaces

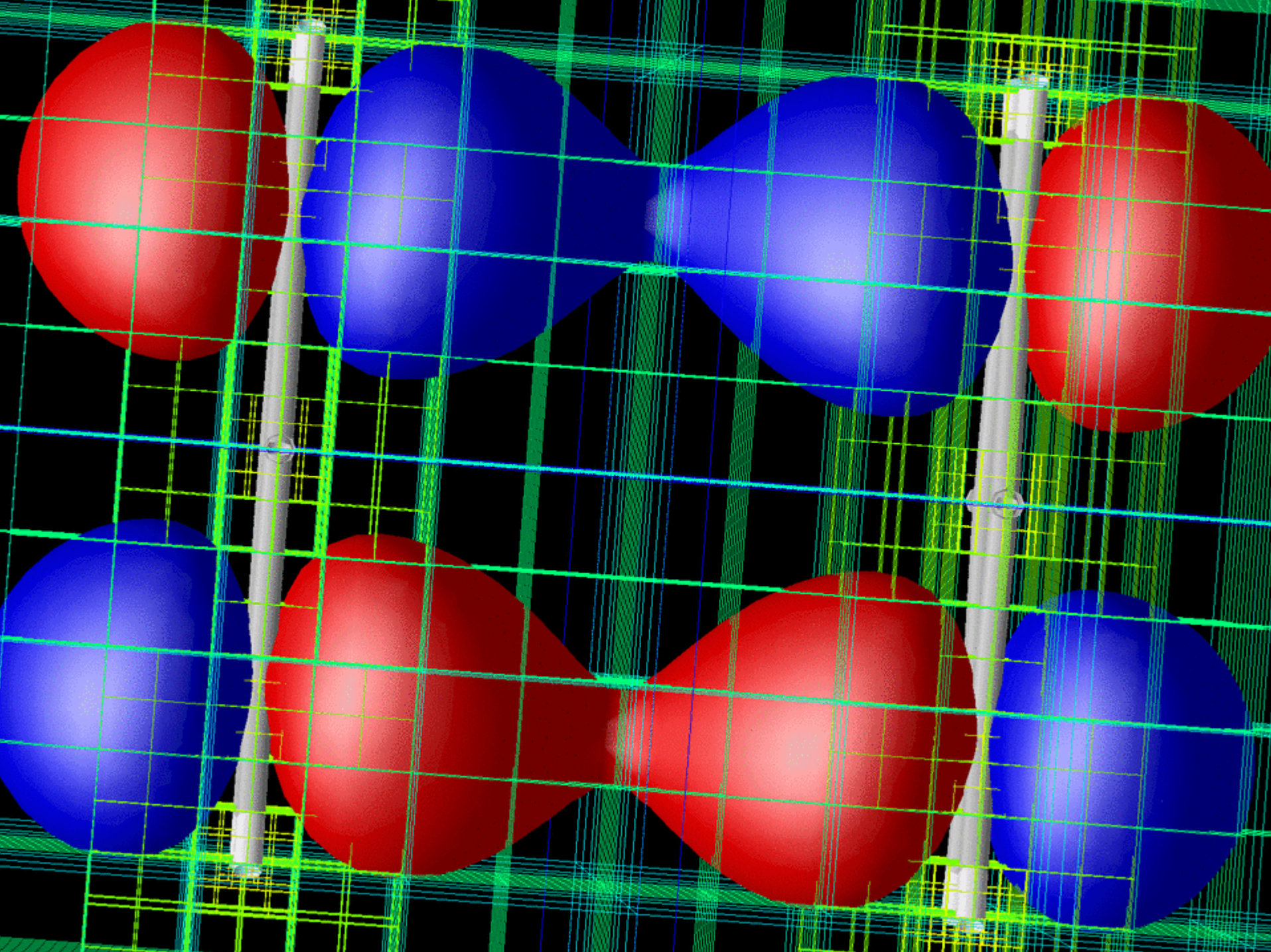
$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_n$$

  – E.g., increasing quality atomic basis sets, or finer resolution grids, …

- Telescoping series

$$V_n = V_0 + (V_1 - V_0) + (V_2 - V_1) + \cdots + (V_n - V_{n-1})$$

  – Instead of using the most accurate representation, use the difference between successive approximations
  – Representation on $V_0$ small/dense; differences sparse
  – Computationally efficient; many possible insights



# Integral Formulation

- Solving the integral equation
  - Eliminates the derivative operator and related “issues”
  - Converges as fixed point iteration *with no preconditioner*

$$\left(-\tfrac{1}{2}\nabla^2 + V\right)\Psi = E\Psi$$

$$\Psi = -2\left(-\nabla^2 - 2E\right)^{-1} V\Psi$$

$$= -2G*\left(V\Psi\right)$$

$$\left(G*f\right)(r) = \int ds \frac{e^{-k|r-s|}}{4\pi|r-s|} f(s) \text{ in 3D ; } k^2 = -2E$$

Such Green’s Functions (bound state Helmholtz, Poisson) can be rapidly and accurately applied with a single, sparse matrix vector product.

# Bases

- Polynomials
  - The good
    - Familiar with much knowledge to draw upon
    - Good for discontinuities located at nodes
  - The bad
    - Over-sampling circa $3^d$
    - Bad condition number, not band limited
- Approximate prolate spheroidal functions
  - The good
    - Asymptotically over sample by only circa $1.2^d$
    - Strong band limit
  - The bad
    - Unfamiliar with many holes in current knowledge
    - Not yet fully tested in adaptive scheme
    - Multiresolution just now being demonstrated

# Current applications

- DFT & HF
  - 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

# 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)
```

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

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

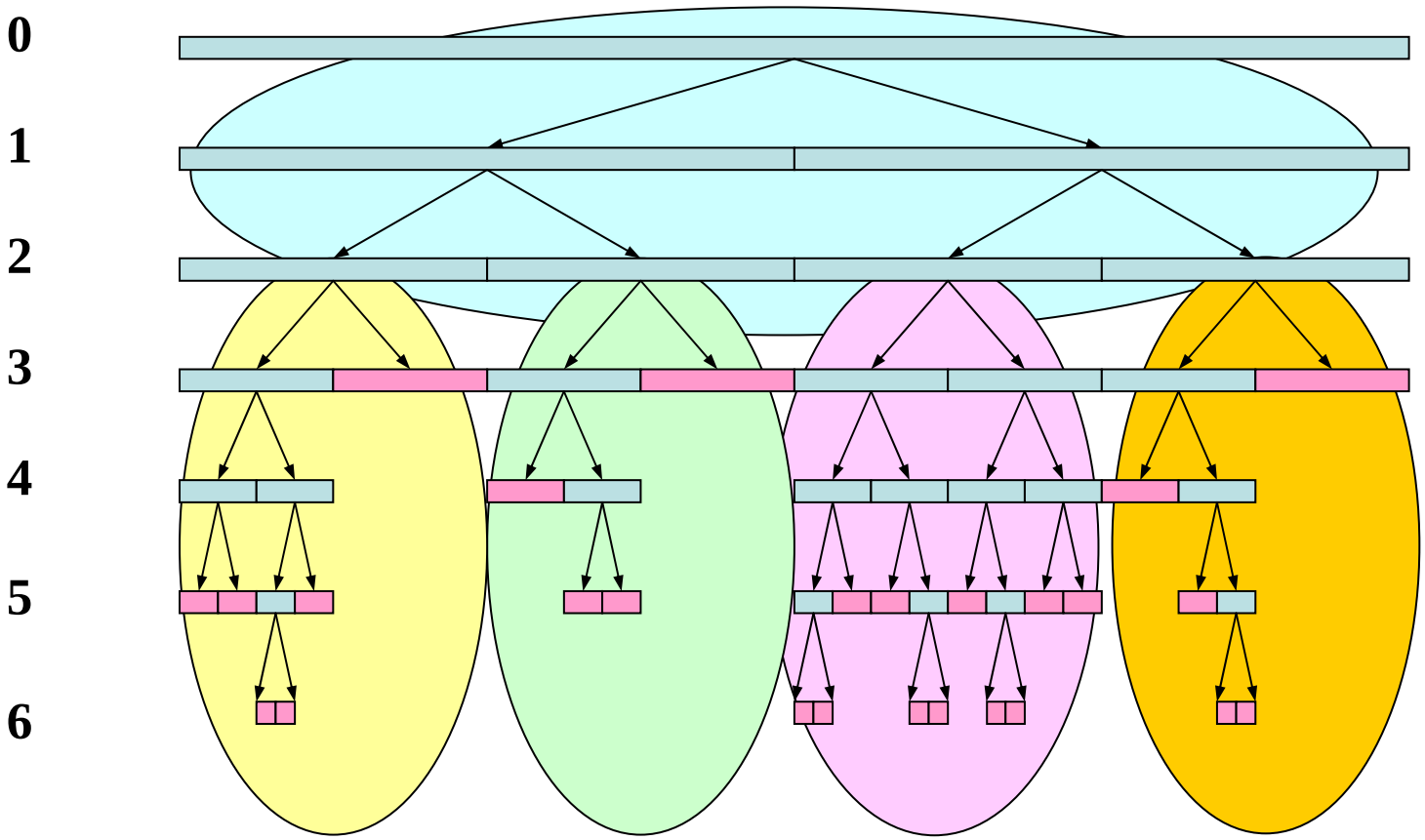
$$= \int \frac{\rho(s)}{|r-s|} ds$$

- All with guaranteed speed and precision

# New MADNESS solver

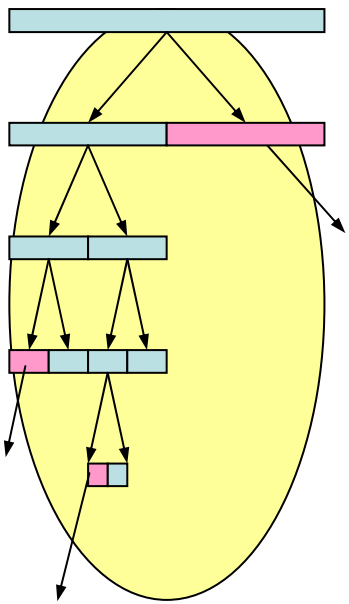
- Total rewrite in C++
  - Three levels of parallelism targeting massively parallel computer using multi-processor nodes
  - In anticipation of highly-threaded processors
  - Ideally targets low latency AM+MPI+threads
  - Portable implementation polling+MPI+threads
- Core math functionality is now running
  - 3D functions, real and complex (1-6D functions will be added this FYI)
  - Scaling demonstrated up to 4096 processors – designed for 1+M.

# 1-D Example Sub-Tree Parallelism




Both sub-trees can be done in parallel.
In 3-D nodes split into 8 children … in 6-D there are 64 children

# Distributed-memory Cilk-like model



Parameter:
- MPI rank
- probe()
- set()
- get()

Task:
- Input parameters
- Output parameters
- probe()
- run()

```
Compress(tree,result):
        Parameter left, right
        if (tree.left) Compress(tree.left, left)
        if (tree.right) Compress(tree.right, right)
        AddTask(Op, left, right, result)
WaitTasks()
```

**Benefits:** **Most receives pre-posted greatly increasing scalability**
**Communication latency & transfer time largely hidden**
**Much simpler composition than explicit message passing**
**Positions code to use “intelligent” runtimes with work stealing**
**Positions code for efficient use of multi-core chips**

# Abstraction overheads

- If you are careful you win
  - *Increased performance and productivity*
  - This is the lesson of Global Arrays, Charm++, …
- Modern C++ is just as fast as Fortran
  - Templates, inlining, restrict qualifier
  - And don’t forget the productivity of Python!
- Overhead of virtualization measured in nano-sec.
  - Less than *80ns* additional latency to create, execute and reap a new user-space thread
  - Less than *30ns* to switch between user-space threads
  - ORNL Cray-XT3, 2.6 GHz opteron, Catamount, my fibers library

# What's in it for the Users? Looking Toward the HPCS Languages and Beyond

David E. Bernholdt, Wael R. Elwasif, and Robert J. Harrison

Oak Ridge National Laboratory

Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the US Dept. of Energy under contract DE-AC-05-00OR22725.

# The HPCS Languages

- Chapel
  - Cray
- Fortress
  - Sun Microsystems
- X10
  - IBM
- ARDA/DARPA/DOE sponsored language evaluation project
  - Lusk (PI), Yellick, Mellor-Crummey, Gropp, Harrison
- *Thanks to vendor groups for spending several days each with us to improve our understanding of their languages!*

# A Revolution in the Making?

- HPCS languages provide significant advances in approach and capability compared to traditional scientific computing languages
  - More alike than different from user perspective
- Major differences from PGAS languages
- High-level (sequential) language features
- Integrated productivity features
- Integrated concurrency

# The Sequential Languages

- Rich array data types
- Strongly typed but with type inference
- Object oriented model
  - Distinction between reference and value types
- Generic programming
- Strongly library oriented
- Extensible language model (more later)

# Productivity Features

- Index sets/regions for arrays, etc.
  - “Array language” (Chapel, X10)
- Safe(r) and more powerful language constructs
  - Atomic sections vs locks
  - Sync variables and futures
  - Clocks (X10)
- Type inference
- Leverage IDE environments
- Units and dimensions (Fortress)
- Component management, testing (Fortress)
- Math/science-based presentation (Fortress)

# Concurrency

- Not SPMD!
  - Initially single thread of control, parallelism through language constructs
- True global view of memory, one-sided access model
- Support for both task and data parallelism
- “Threads” grouped by “memory locality”
  - Explicitly two level (Chapel, X10), or hierarchical (Fortress)
- Rich distributed array capability
  - Programmer-provided distribution details
- Parallel loops
- “Generator” concept used widely for loops, distributions
- Futures
  - Local and remote

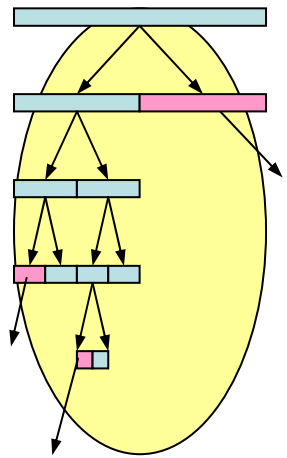
# Something for the Tinkerer

- Object orientation, generic programming, and programmer-written data distributions
- All three languages also intend to offer an unprecedented flexibility to extend the language
  - Libraries
  - Compiler optimizations and specializations
  - Language syntax (Fortress, X10)
- Extremely powerful tools to support high-level domain-specific abstractions

# MADNESS Example – III

- X10, 1-D equivalent (ran in parallel, in one place)

```
void refine(final int n, final int l, final int nmax) {
        left = new Tree(this,2.0*l);
        right =new Tree(this, 2.0*l+1);
        final nullable Tree ll = left, rr=right;
        if (n < (nmax-1)) {
                async {ll.refine(n+1,2*l,nmax);}
                async { rr.refine(n+1,2*l+1,nmax);}
        }
        if (n < nmax) data = null;
}
```

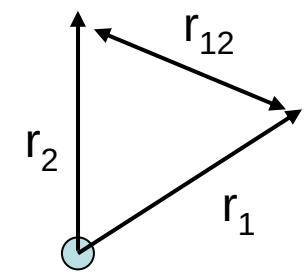


- Issues for all of the languages
  - How to control (or relinquish control) of the initial data placement?
    - One or the other is cumbersome in all of the languages
  - How to express dynamic load balancing between places?
    - Only Fortress seems to provide this

# Summary

- Chemists, driven by necessity, are uniting coding efforts
- Advanced numerical methods
  - Potentially algorithmically faster & more accurate
  - Higher-level composition of applications
  - Well suited to anticipated architectures
- Other parallel programming paradigms
  - Can be portable, efficient and vastly more productive than MPI

# Electron correlation




- All defects in the mean-field model are ascribed to electron correlation
- Consideration of singularities in the Hamiltonian imply that for a two-electron singlet atom (e.g., He)

$$\Psi(r_1, r_2, r_{12}) = 1 + \tfrac{1}{2} r_{12} + O(r_{12}^2) \quad \text{as} \quad r_{12} \to 0$$

- Include the inter-electron distance in the wavefunction
  - E.g., Hylleraas 1938 wavefunction for He

$$\Psi(r_1, r_2, r_{12}) = e^{-\varsigma(r_1 + r_2)}(1 + a r_{12} + \cdots)$$

  - Potentially very accurate, but not systematically improvable, and (until recently) not computationally feasible for many-electron systems

# Conventional approach

- The two-electron wave function is expanded as a product of one-particle functions (orbitals)

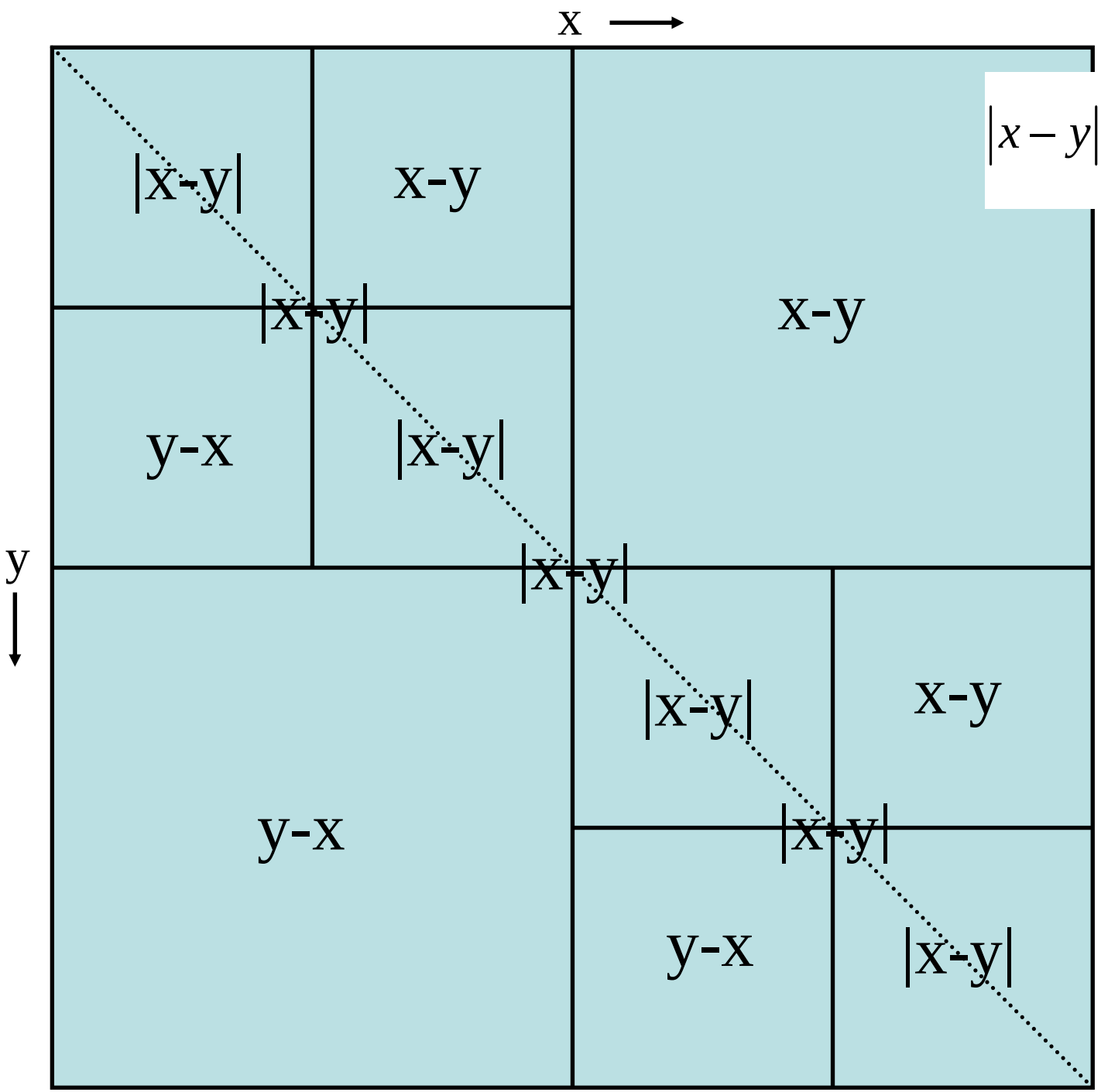
$$\Psi(r_1, r_2) = \sum_{ij} c_{ij} \phi_i(r_1) \phi_j(r_2)$$

- Can prove for atoms, that if saturate the atomic basis up to some angular momentum L, then

$$\Delta E_{corr} \propto (L+1)^{-3}$$

| L | d | f | g | h |
|---|---|---|---|---|
| $(L+1)^{-3}$ | 0.04 | 0.016 | 0.008 | 0.0046 |

- Correlation consistent basis sets (Dunning) are currently the best choice – cost is $O(\varepsilon^{-4})$
- Explicitly correlated wave functions yields $O(\varepsilon^{-2.4})$
- Fully numerical promises $O(\log \varepsilon^{-1})$




$$|x-y| = \sum_{\mu=1}^{r} f_\mu(x) g_\mu(y)$$

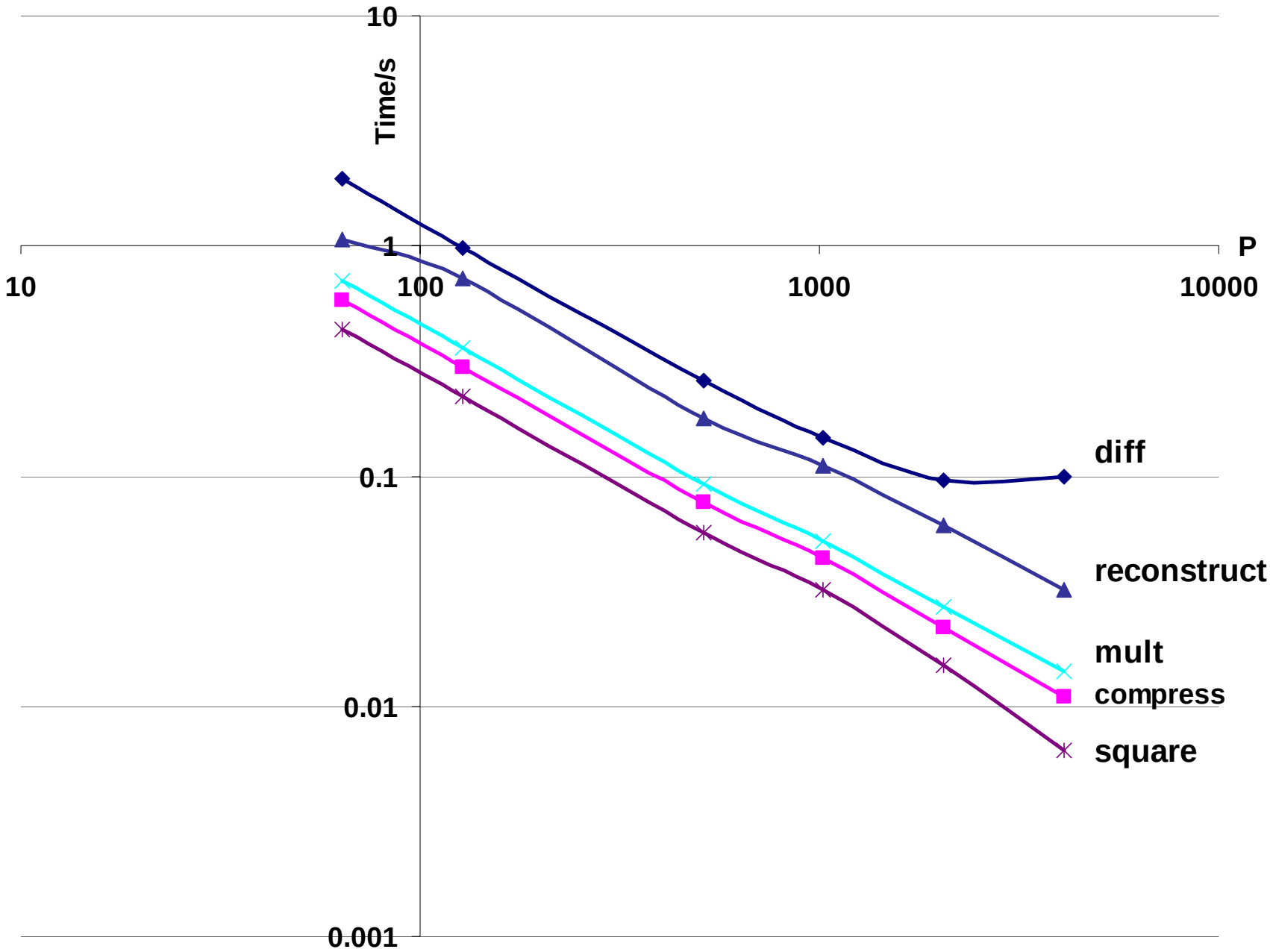
$r$ = separation rank

In 3D, ideally must be one box removed from the diagonal

Diagonal box has full rank

Boxes touching diagonal (face, edge, or corner) have increasingly low rank

Away from diagonal $r = O(-\log \varepsilon)$


10
Time/s
1
0.1
0.01
0.001
10
100
1000
10000
P
diff
reconstruct
mult
compress
square