

Raising the Level of Abstraction in Scalable Programming Models

David E. Bernholdt Oak Ridge National Laboratory
Robert J. Harrison Oak Ridge National Laboratory
Jarek Nieplocha Pacific Northwest Natl. Laboratory
P. Sadayappan The Ohio State University

Research supported by U.S. Department of Energy and National Science Foundation

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

Two Example Approaches Addressing Complexity

- **Global Arrays (GA) and Extensions**
 - Provision of global shared view of arrays and efficient access to array subsections that may be physically distributed across multiple processors
- **The Tensor Contraction Engine (TCE)**
 - A high-level language and optimizing compiler specialized to the needs of a class of applications in quantum chemistry and atomic physics; “telescoped” over the GA library
- **Common Features:**
 - Originated as domain-specific tools (quantum chemistry)
 - Designed to be as general as possible (beyond domain)
 - Tools and ideas expanding into other domains

Global Arrays

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

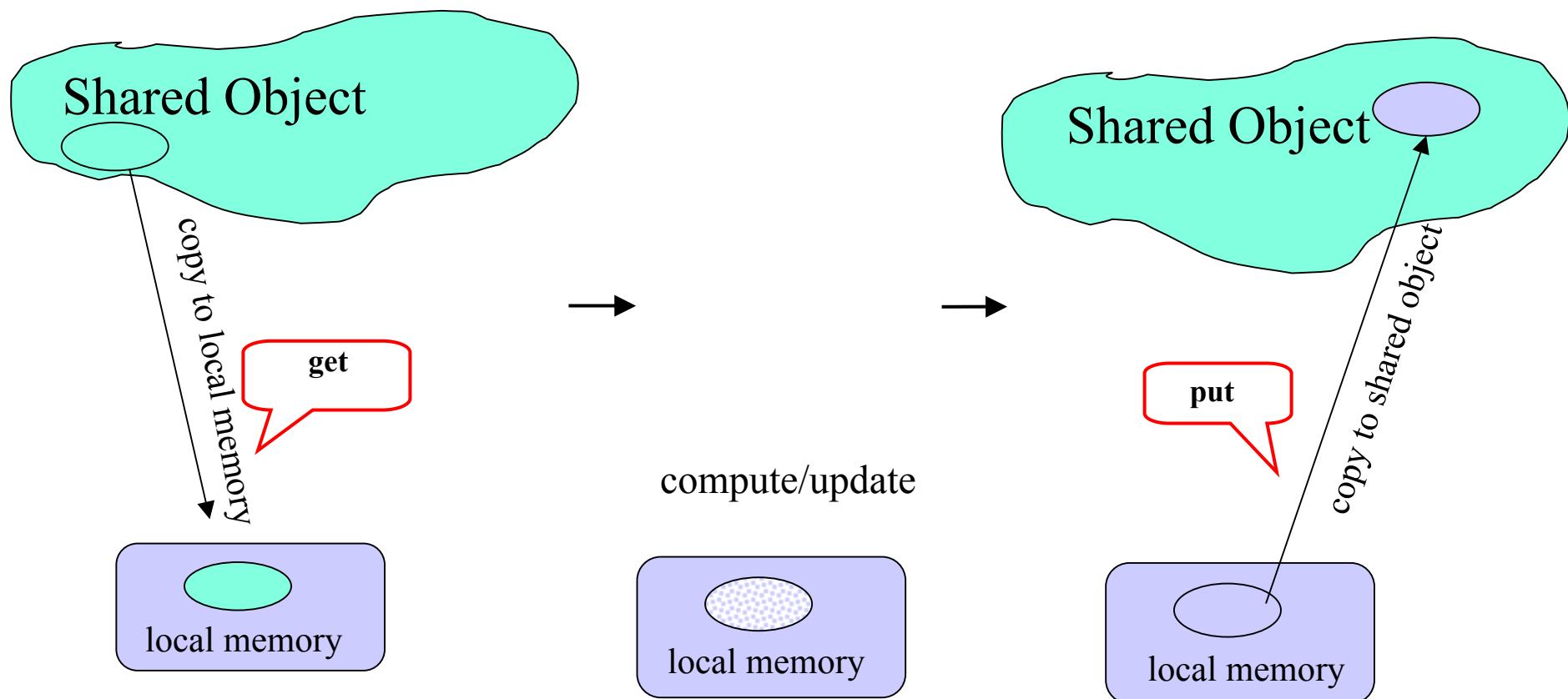
Jarek Nieplocha

Manoj Kumar Krishnan, Bruce Palmer

Vinod Tipparaju, Harold Trease

Robert Harrison, George Fann, David Bernholdt

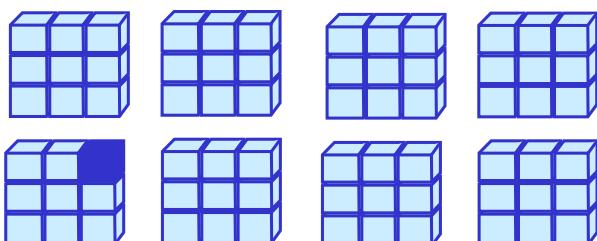
Global Array Model of Computations



Global Arrays

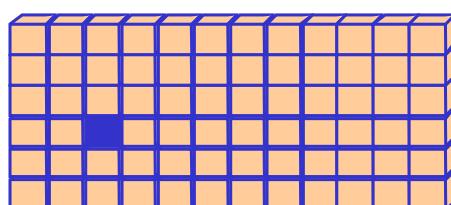
Distributed dense arrays that can be accessed through a shared memory-like style

Physically distributed data



single, shared data structure/
global indexing

e.g., access $A(4,3)$ rather than
buf(7) on task 2

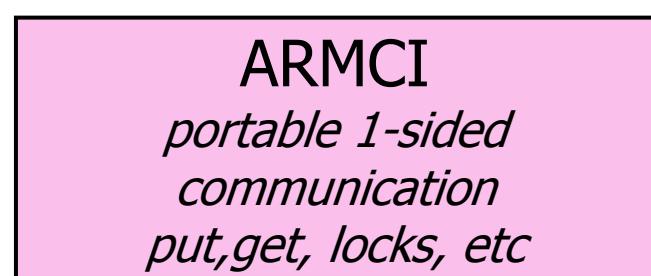
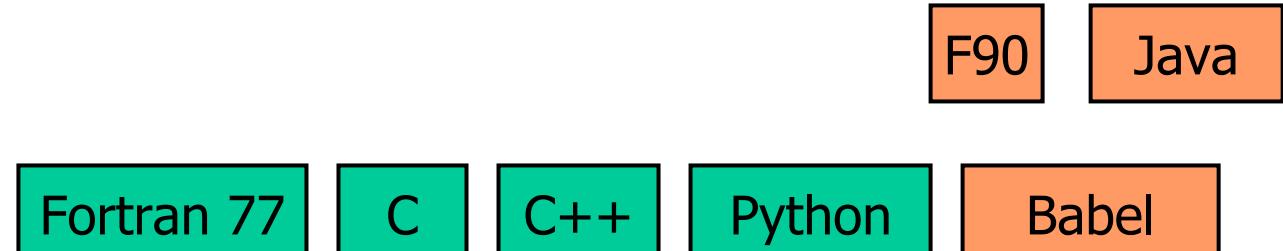


Global Address Space

Interoperability of GA

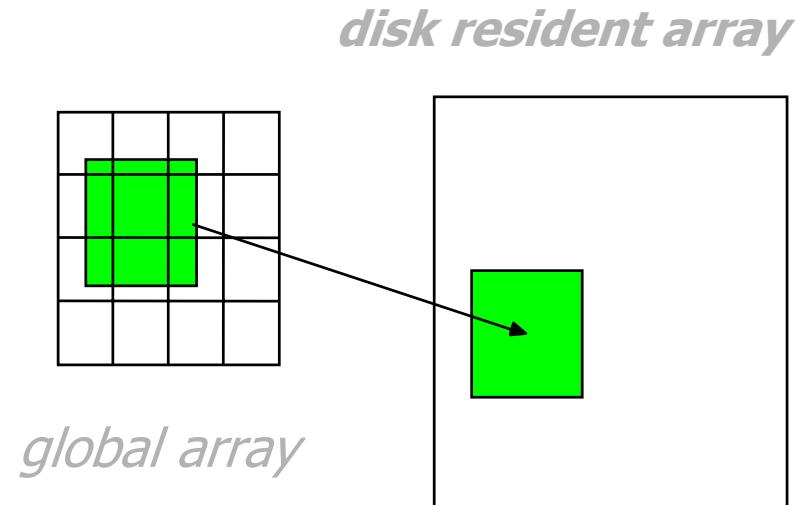
Application
programming
language interface

Global Arrays
and MPI are
completely
interoperable.
Code can
contain calls
to both
libraries.

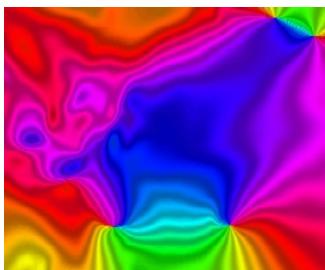


Disk Resident Arrays

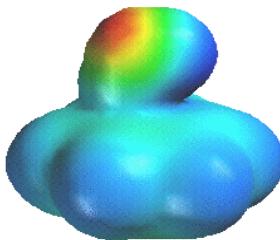
- Extend GA model to disk
- Provide easy transfer of data between N-dim arrays stored on disk and distributed arrays stored in memory
- Use when
 - Arrays too big to store in core
 - checkpoint/restart
 - out-of-core solvers



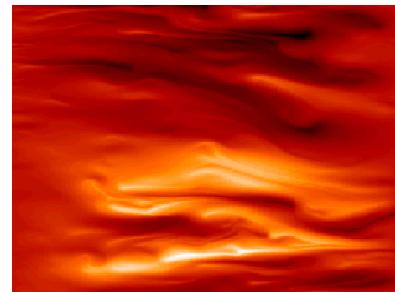
Application Areas



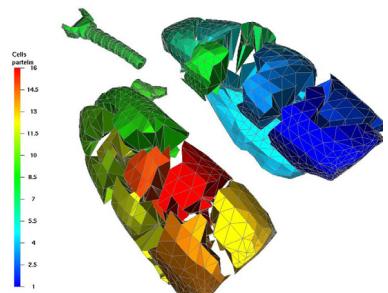
Visualization and image analysis



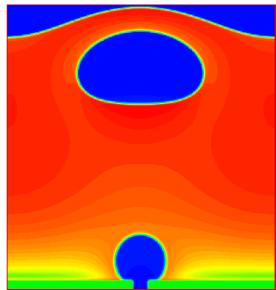
electronic structure



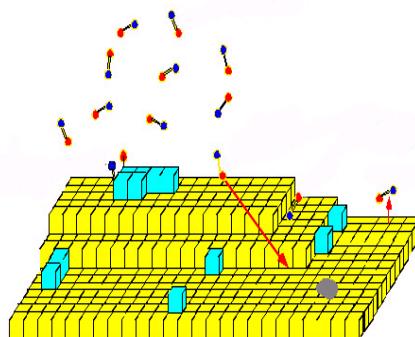
glass flow simulation



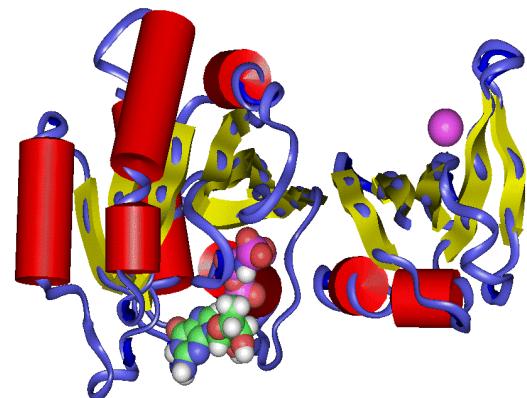
biology



thermal flow simulation



material sciences



molecular dynamics

Others: financial security forecasting, astrophysics, geosciences

The Tensor Contraction Engine: A Tool for Quantum Chemistry

Oak Ridge National
Laboratory

*David E. Bernholdt,
Venkatesh Choppella, Robert
Harrison*

Pacific Northwest National
Laboratory
So Hirata

Louisiana State University
J Ramanujam

Ohio State University

*Gerald Baumgartner, Alina
Bibireata, Daniel Cociorva,
Xiaoyang Gao, Sriram
Krishnamoorthy, Sandhya
Krishnan, Chi-Chung Lam,
Quingda Lu, Russell M.
Pitzer, P Sadayappan,
Alexander Sibiryakov*

University of Waterloo
*Marcel Nooijen, Alexander
Auer*

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

Research at ORNL supported by the Laboratory Directed Research and Development Program. Research at PNNL supported by the Office of Basic Energy Sciences, U. S. Dept. of Energy. Research at OSU, Waterloo, and LSU supported by the National Science Foundation Information Technology Research Program

Problem Domain: High-Accuracy Quantum Chemical Methods

- Coupled cluster methods are widely used for very high quality electronic structure calcs.
- Typical Laplace factorized CCSD(T) term:

$$A3A = \frac{1}{2} (X_{ce,af} Y_{ae,cf} + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}} + X_{\bar{ce},\bar{af}} Y_{\bar{ae},cf} \\ + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}} + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}} + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}})$$

$$X_{ce,af} = t_{ij}^{ce} t_{ij}^{af} \quad Y_{ae,cf} = \langle ab | ek \rangle \langle cb | fk \rangle$$

Typical methods will have tens to hundreds of such terms

- Indices i, j, k $O(O=100)$ values, a, b, c, e, f $O(V=3000)$
- Term costs $O(OV^5) \approx 10^{19}$ FLOPs; Integrals ~ 1000 FLOPs each
- $O(V^4)$ terms ~ 500 TB memory each

CCSD Doubles Equation

Theory	Terms
CCD	11
CCSD	48
CCSDT	102
CCSDTQ	183
...	...

In the coupled cluster method with single and double excitations (CCSD) the “singles” and “doubles” equations are iterated until convergence and that solution is used to evaluate the molecular energy

The “Tensor Contraction Engine” Addresses Programming Challenges

- User describes computational problem (tensor contractions, a la many-body methods) in a simple, high-level language
 - Similar to what might be written in papers
- Compiler-like tools translate high-level language into traditional Fortran (or C, or...) code
 - Currently targeting GAs as parallel programming model
- Generated code is compiled and linked to libraries providing computational infrastructure
- **Productivity**
 - User writes simple, high-level code
 - Code generation tools do the tedious work
- **Complexity**
 - Significantly reduces complexity visible to programmer
- **Performance**
 - Perform optimizations prior to code generation
 - Automate many decisions humans make empirically
 - Tailor generated code to target computer
 - Tailor generated code to specific problem

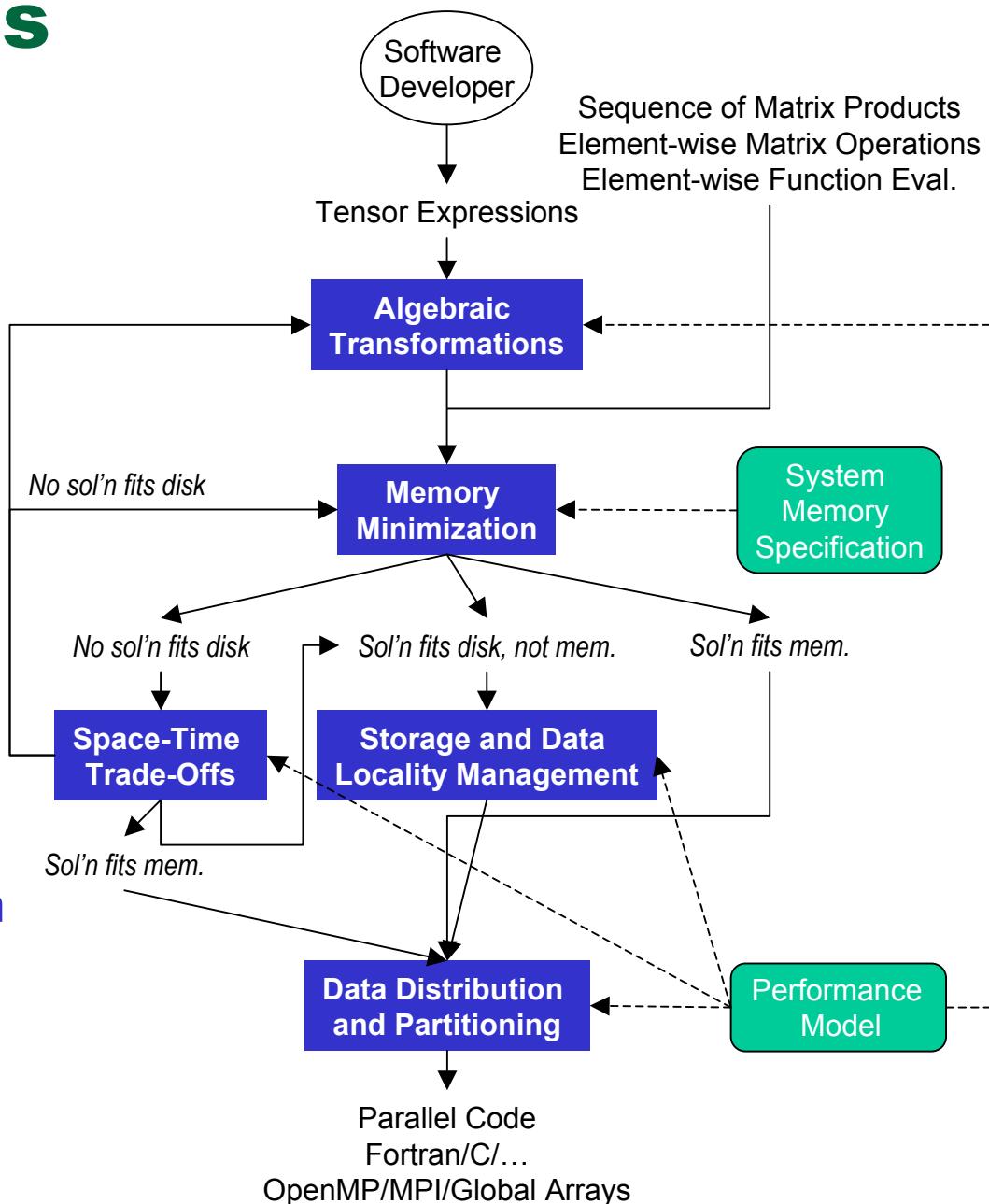
A High-Level Language for Tensor Contraction Expressions

```
range V = 3000;  
range O = 100;  
  
index a,b,c,d,e,f : V;  
index i,j,k : O;  
  
mlimit = 1000000000000;  
  
function F1(V,V,V,O);  
function F2(V,V,V,O);  
  
procedure P(in T1[O,O,V,V], in T2[O,O,V,V], out X)=  
begin  
    X == sum[ sum[ F1(a,b,f,k) * F2(c,e,b,k), {b,k}]  
              * sum[ T1[i,j,a,e] * T2[i,j,c,f], {i,j}],  
              {a,e,c,f}];  
end
```

$$A3A = \frac{1}{2} \left(X_{ce,af} Y_{ae,cf} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},cf} \right. \\ \left. + X_{ce,\bar{a}\bar{f}} Y_{ae,\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{ae,\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{ae,\bar{c}\bar{f}} \right)$$
$$X_{ce,af} = t_{ij}^{ce} t_{ij}^{af} \quad Y_{ae,cf} = \langle ab \| ek \rangle \langle cb \| fk \rangle$$

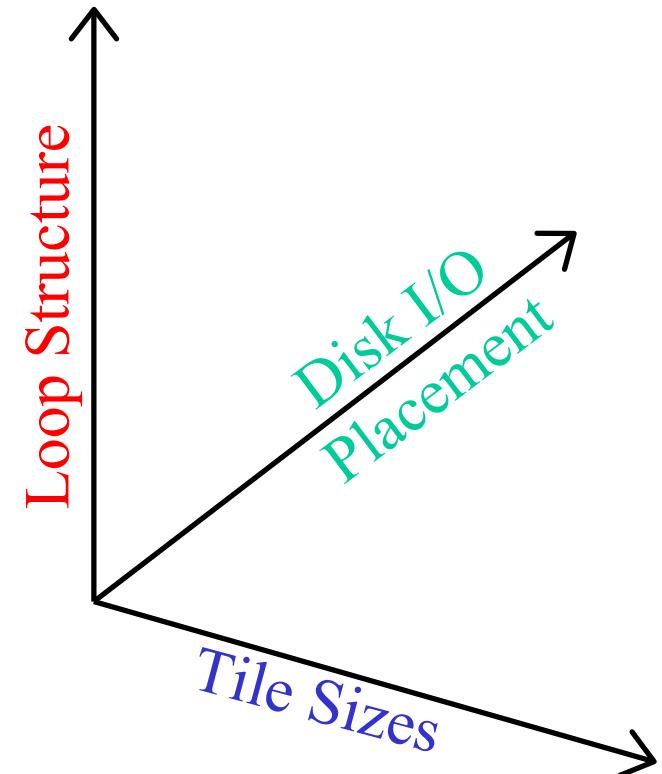
TCE Components

- Algebraic Transformations
 - Minimize operation count
- Memory Minimization
 - Reduce intermediate storage via loop fusion (LCPC'03)
- Space-Time Transformation
 - Trade-offs between storage and recomputation (PLDI'02)
- Data Locality Optimization
 - Optimize use of storage hierarchy via tiling (ICS'01, HiPC'03, IPDPS'04)
- Data Dist./Comm. Optimization
 - Optimize parallel data layout (IPDPS'03)
- Integrated System
 - (SuperComputing'02, Proc. IEEE 05, To Appear)



Search in Multidimensional Space

- Much prior work by the compiler community on loop transformations, but very little work on transformations driven by accurate performance models
- We address synthesis of efficient concrete out-of-core code from an abstract specification of a loop computation
 - Driven by accurate performance model
 - Extensive search among space of parameters
 - Exploit domain-specific information
- Relation to library tuning approaches like ATLAS:
 - Search uses performance model rather than actual code execution
 - Compiler-driven (applies to any computation in the class handled) vs. manually engineered for specific library routines



Effect of Multidimensional Optimization: AO-to-MO Transformation

$$B(a,b,c,d) = \sum_{p,q,r,s} C1(s,d) \times C2(r,c) \times C3(q,b) \times C4(p,a) \times A(p,q,r,s)$$

Range $p,q,r,s=150$, Range $a,b,c,d=140$, Memory = 2GB

Optimizations	Total Disk I/O Time (s)	Total Execution Time (s)
Fusion + Optimized Tiling	248.43	954.87
No Fusion, Optimized Tiling	747.83	1261.95
No Fusion, Fixed Tiling ($4\sqrt{M}/3$)	1240.85	1957.18

Measurements were taken on an Itanium 2 System

Productivity of TCE

- The tensor contraction expressions for the higher members of the Coupled Cluster family of models can be generated relatively easily, but the effort to manually generate Fortran code is quite significant
- The code development time for other models of comparable complexity can be reduced from years to days/weeks
- More than 25 methods implemented to date → » 5 years to hand code

Theory	#Terms	#F77Lines	Year
CCD	11	3209	1978
CCSD	48	13213	1982
CCSDT	102	33932	1988
CCSDTQ	183	79901	1992



Result of first hand implementation of CCSDTQ

Lessons Learned (What do application programmers want?)

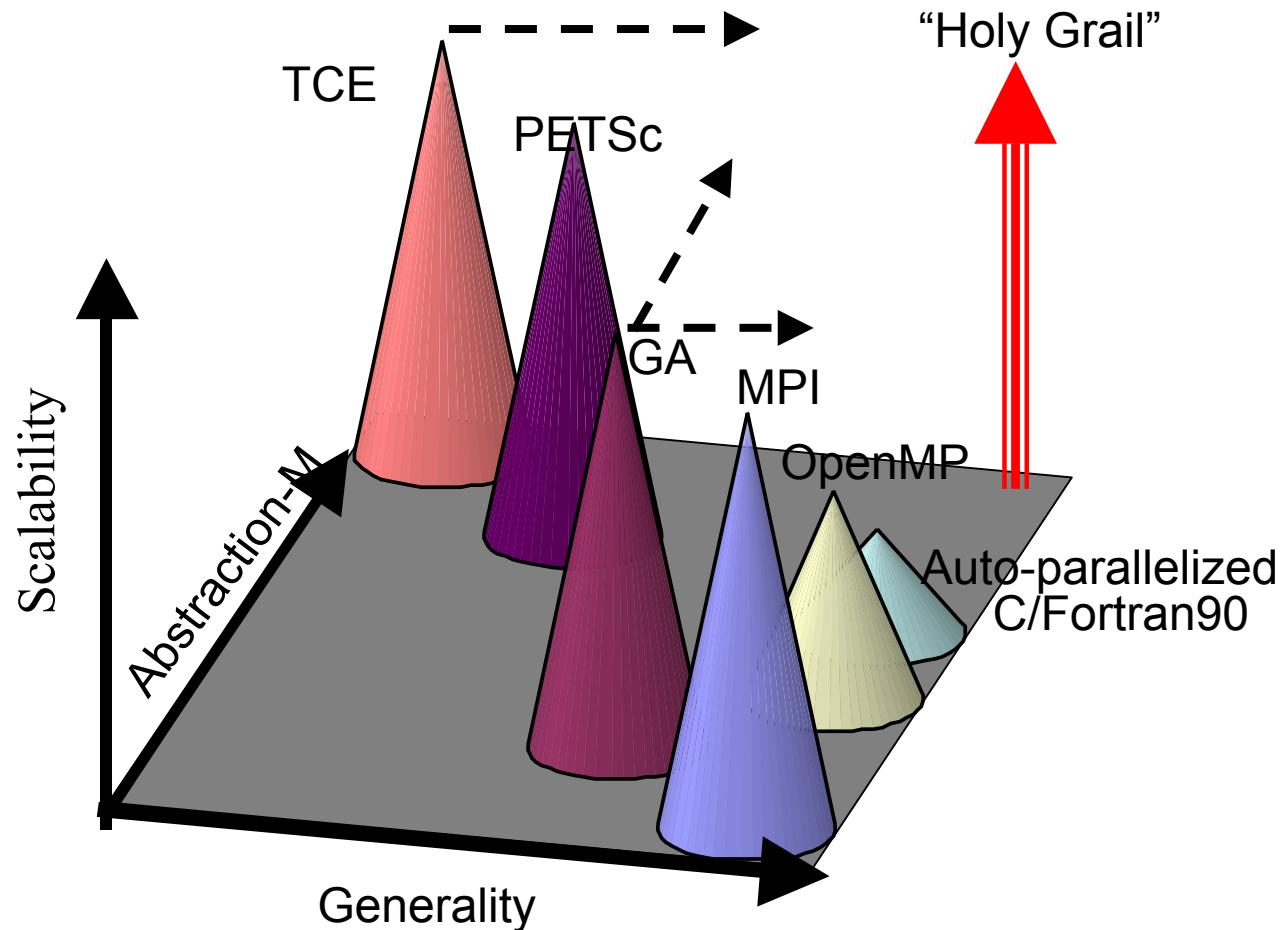
Global Arrays

- One-sided access
- Global view
- The “right” data abstraction
 - For quantum chemistry
 - Other abstractions for other areas could be created with ARMCI layer
- Both explicit and implicit management of memory hierarchy possible
 - Default distributions
 - DRAs

Tensor Contraction Engine

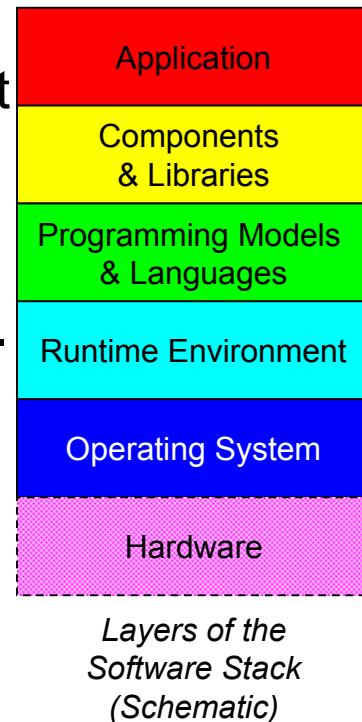
- The “right” abstraction
 - Closes the semantic gap between the application and the implementation language
- Provides performance with productivity
 - Domain-specificity central to optimization
- More reliable than hand implementation

Future Work: Enhancement of Generality/Abstraction/Scalability



The Importance of Vertical Integration

- Success of GA and TCE due in large part to:
 - Systems development driven by application needs
 - Active involvement of applications developers
 - Emphasis on maintaining both high performance and high productivity
- High **Generality/Abstraction/Scalability** very difficult
 - Hence most high-end S/W is done now in MPI
 - But the more complex problems get, the more important it is to have models with high **G/A/S**, to achieve high productivity and reliability
- Need more rapid progress in raising **G/A/S** of prog. models
 - Vertically integrated teams of application developers and systems developers across levels of SW/HW stack
 - Many high-level approaches should be vigorously pursued



Next Steps...

Vertically-Integrated HPC Software Development Environments: Performance and Productivity for Next Generation Scientific Applications

David E. Bernholdt, Robert J. Harrison,
Jeffrey S. Vetter, and James B. “Trey” White III ORNL
P. Sadayappan Ohio State U.
Stanley A. Ahalt and Ashok Krishnamurthy OSU and OSC
Jarek Nieplocha PNNL

The “Vertical Integration for Productivity” (VIP) Approach

- Foster increased research & development activity in programming models/environments for ultrascale scientific computing
- Applications-driven
 - Involve application scientists directly in VIP projects
 - Credible application-level demonstrations required before other application groups will take a new approach seriously
- Key elements for simultaneously achieving productivity and portable performance:
 - Raising level of abstraction for programmer
 - Vertical integration of the software stack
 - Levels of software stack should interact intelligently rather than treat each other as black boxes
 - Not a single monolithic system, but standards for interoperability of layers

Moving Forward

- Planning workshop series
 - Build community and obtain input to define/refine the approach
 - Targeting late summer for first workshop
- Need to determine relationships to other efforts
 - DARPA HPCS & High-Productivity Languages
 - Stan Ahalt's "blue collar" HPC environments workshop
 - NSF ST-HEC program
 - Others...?