UT-BATTELLE

The Common Component Architecture (CCA) and Automated Code Synthesis (LDRD)

David E. Bernholdt

Computer Science and Mathematics Division, Oak Ridge National Laboratory

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

What is the Common Component Architecture?

- Component models help manage the complexity of large-scale software systems
  - Facilitate OO design methodologies
  - Promote reuse and interoperability
  - Facilitate interdisciplinary development teams
- CCA is a component model specifically designed for the needs of the HPC scientific simulation community
  - Explicitly supports parallel computing
  - Small/zero performance cost (for local & parallel environment)
  - Minimalist approach to simplify componentization of existing software
- Not a silver bullet for all aspects of software development
  - Algorithms, programming still hard, have to be done
  - Software architecture/design still important

Groups Behind the CCA

CCA Forum

- Set specifications for CCA components & frameworks
- Promote and facilitate development of domain-specific “standard” interfaces
- Meeting quarterly since 1998
- Open membership
  - Superset of CCTTSS
- Next meeting 10-12 April, Valley View Lodge, Townsend, TN (hosted by ORNL)

CCTTSS (SciDAC ISIC)

- Develop CCA technology from current prototype stage to full production environment
- Increase understanding of how to use component arch. effectively in HPC environment
- Participants:
  - Argonne, Livermore, Los Alamos, Oak Ridge, Pacific Northwest, and Sandia National Laboratories;
  - Indiana University, and University of Utah

ORNL's participation in CCTTSS

(highlighted in red)

- **Scientific Components** *(Lead: Lois McInnes, ANL)*
  - Abstract interfaces and component implementations
  - Scientific data; Linear, nonlinear, and optimization solvers; Steering and visualization; Multi-threading and load redistribution; Fault tolerance
  - Quality of service research

ORNL participation in CCTTSS (cont)

- **MxN Parallel Data Redistribution** *(Lead: Jim Kohl, ORNL)*
  - Basic MxN interface and component implementation
  - Higher-level coupling (units, grid interpolation, etc.)
  - "Automatic" MxN via framework, parallel RMI
- **Outreach and Applications Integration** *(Lead: David Bernholdt, ORNL)*
  - General education and awareness
  - Strong liaison with adopting groups
  - Chemistry and Climate work within CCTTSS

Scientific Data Components

- CCA Forum Working Group lead by Lori Freitag,ANL
- Common approach to basic data types to facilitate reuse and interoperability
  - Emphasis on *descriptions* of data in existing codes – rather than imposing our type definitions

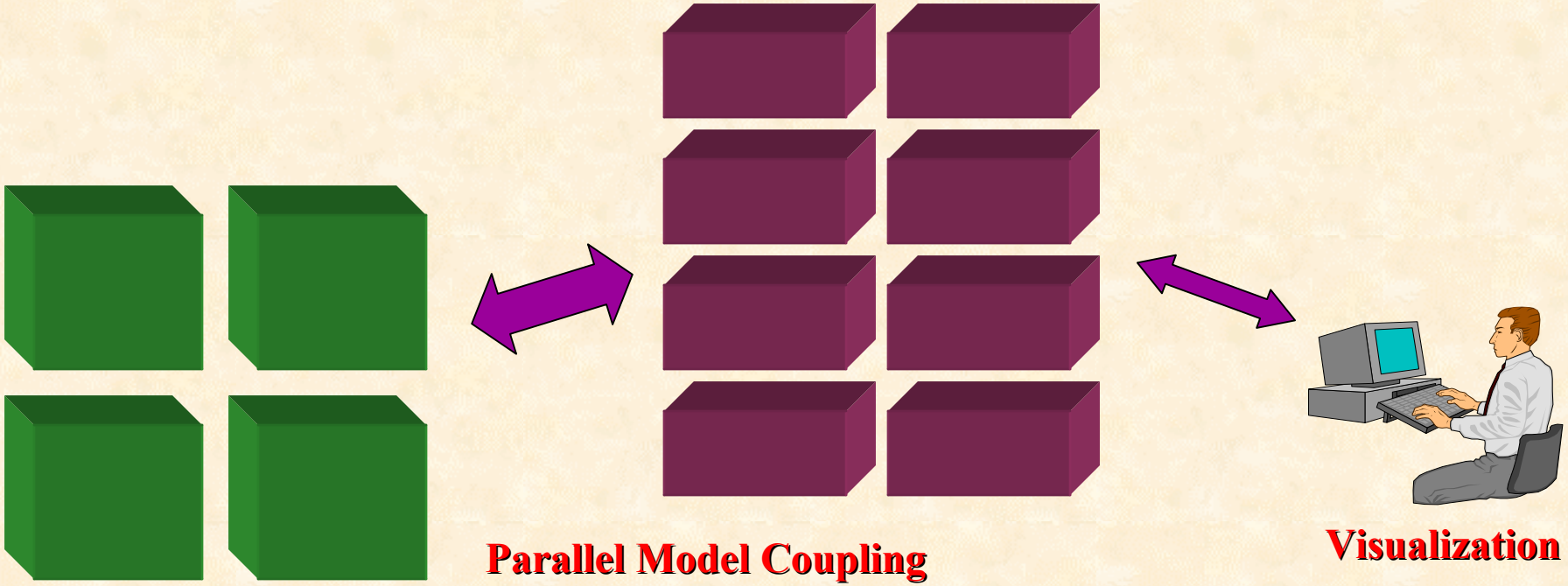
- Basic Scientific Data Objects (Bernholdt, ORNL)
- Unstructured Mesh (Freitag, ANL)
  - Now primarily in TSTT
- Structured Mesh (McInnes, ANL)
  - Requirements factored into basic array, distributed array
  - Structured AMR (new, w/ APDEC and others?)

Basic Scientific Data Objects

- Raw Data Interface
  - Tentative definition, no implementation
- Local Array Interface
  - Tentative definition, no implementation
- **Distributed Array Descriptor Interface**
  - Fairly mature definition (Bernholdt + Data Working Group)
    - Supports HPF 2.0 capabilities and more
    - Compatible w/ CUMULVS, Global Arrays, SCALAPACK, PETSc, ESI, …
  - Implemented for SC01 demo applications (Bernholdt)
    - C++, ~3500 LOC (commented), production est. 14,000 LOC
  - Developing interface revisions based on experience, SIDLization (Elwasif)
- "Native" CCA Distributed Array Interface
  - Planned (Bernholdt, Nieplocha), based on DAD

MxN Parallel Data Redistribution

- Share Data Among Coupled Parallel Models
  - Disparate Parallel Topologies (M processes vs. N)
  - e.g. Ocean & Atmosphere, Solver & Optimizer…
  - e.g. Visualization (Mx1, increasingly, MxN)

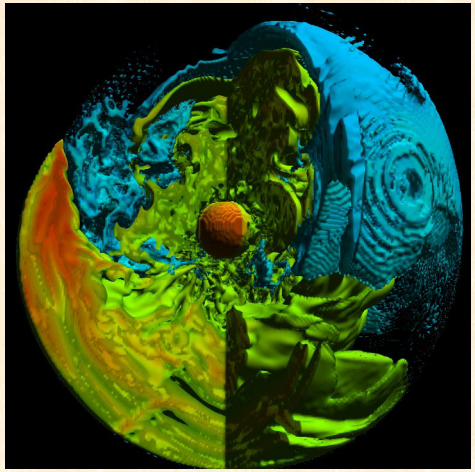


Recent MxN Activities

- MxN port (Kohl + MxN Working Group)
  - Draft specification completed
  - Extension of concepts in CUMULVS, PAWS
  - Uses distributed array descriptors to represent data
  - Minimal intrusion to “instrument” component, third-party control possible
  - Multiple data fields, exchange in either direction
  - One-shot or periodic transfer
- MxN components
  - Implemented for SC2001 demo applications
  - Kohl, based on CUMULVS
    - SC01: Mx1 for visualization w/ CUMULVS, AVS, VTK for optimization, heat eqn, and PDE demo applications
    - C++ ~3000 LOC (commented), production est. 6000 LOC + CUMULVS extensions, est. +10,000 LOC
  - Rasmussen et al, based on PAWS
    - SC01: MxN data ping-pong application
    - C++ ~2800 LOC (commented) + PAWS library

Outreach and Applications Integration

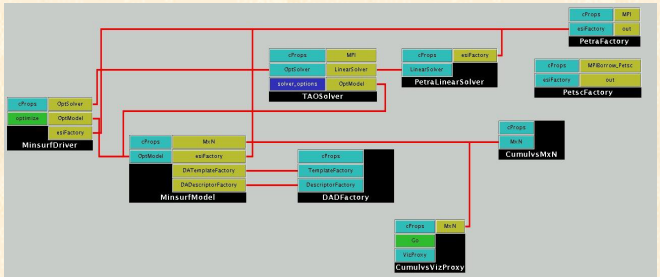
- General education and awareness
  - Tutorial at PNNL (Bernholdt)
  - Tutorial at January (Bernholdt, Rasmussen, Kohn, Freitag) and April CCA Forum Meetings
  - CCA presentation ORNL SciDAC seminar series (Bernholdt)
- Strong liaison with adopting groups
  - Bernholdt attended Terascale Supernova Initiative Collaboration Meeting
  - Kohl initiated contact with GATech SciDAC group interested in CCA & MxN
- "In-house" work on Chemistry, Climate integration
  - Climate kick-off in November @ NCAR (ANL, NCAR, ORNL)
  - Chemistry kick-off in December @ PNNL (ANL, ORNL, PNNL, SNL)
  - Recent discussions with Al Wagner (ANL) on activities aligned with Chemistry work



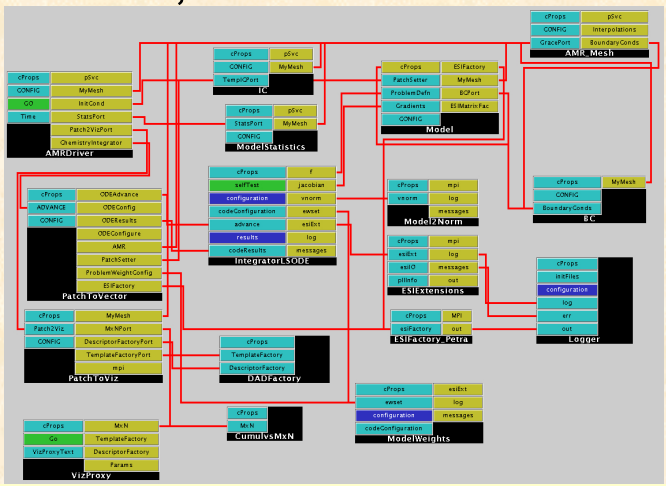
Shockwave from 3d supernova explosion simulation. *Image by Ross Toedte based on data from John Blondin, SciDAC TSI*

SC2001 CCA Software Distribution

- Four different model applications
  - 31 distinct components,
  - up to 17 in any single application,
  - 6 used in more than one application
- Developed by ANL, LANL, ORNL, SNL



CCA application for solution of an unconstrained minimization problem using the TAOSolver optimization component



CCA application for solution of a two-dimensional heat equation on a square domain using an adaptive structured method.

- Components leverage and extend parallel software tools including CUMULVS, GrACE, LSODE, MPICH, PAWS, PETSc, PVM, SUMAA3d, TAO, and Trilinos.
- **Released for early adopters and evaluation purposes**
- All pre-release testing by ORNL (Elwasif, Wilde)

CCA @ ORNL Summary

- Well-integrated into CCTTSS team
- We are leading two of the four CCTTSS thrust areas, active in three
- Driving development of key specifications promoting interoperability and extensibility
- Outreach and Application Integration activities can be time-consuming, but are very important, especially at this moment

UT-BATTELLE

Automated Synthesis of High Performance Algorithms
for Electronic and Nuclear Structure Calculations

Principal Investigators:

David E. Bernholdt	CSM, ORNL
James B. White III	CSM, ORNL
David J. Dean	Physics, ORNL
Michael R. Strayer	Physics, ORNL

Co-Investigators:

Gerald Baumgartner	CIS, Ohio State U
Robert J. Harrison	PNNL
Marcel Nooijen	Chem., Princeton
Russell M. Pitzer	Chem., Ohio State U
Jagannathan Ramanujam	CS, Louisiana State U
Ponnuswamy Sadayppan	CIS, Ohio State U

Sponsored by the Director's R&D Fund

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

Goal and Motivation

- Goal of this project is to develop application-oriented synthesis framework with significant optimization capabilities
  - Concrete initial targets: quantum chemistry, nuclear physics
    - Significant consumers of DOE computing resources
    - Chemistry work targets NWChem environment
  - Design for generality to facilitate extension to other domains

- Complex mathematical expressions are important in HPC simulation in many domains, much code required
- Implementation involves many human design decisions before a compiler sees any code
- Design optimization too complex for humans – handled heuristically based on experience, intuition
- Modern parallel architectures are very complex

Automated Synthesis

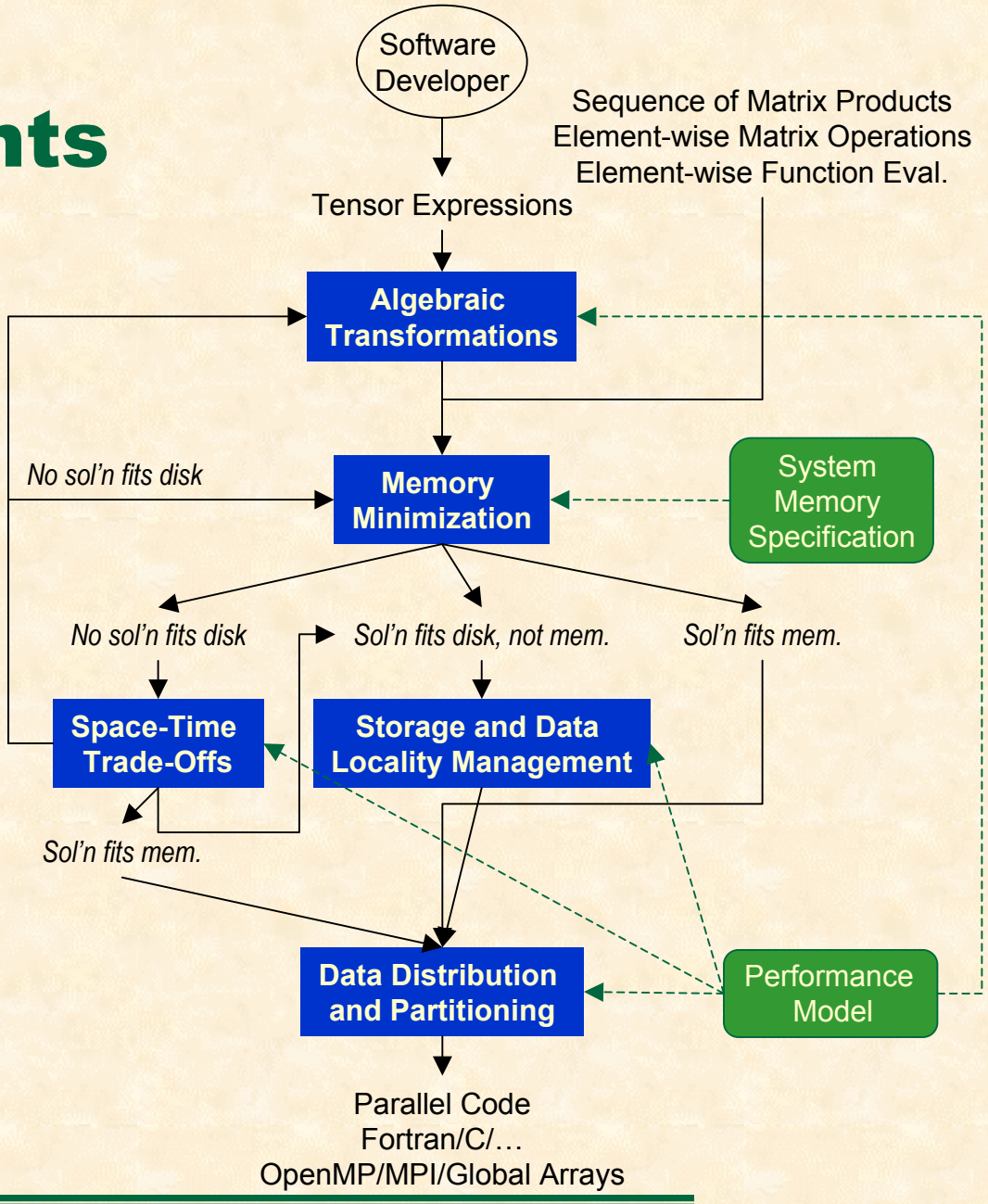
- **Automated source code synthesis** can help address complexity and performance issues
  - Start from high level expressions
  - Optimize equations that are coded
  - Impossible for general-purpose compilers
- Demonstrated in a number of domains
  - TAMPR (linear algebra; ANL, Queen's (Ireland), SNL)
  - SPIRAL (signal processing; CMU, Drexel, MathStar, UIUC, USC)
  - SciFinance (financial modeling; SciComp Inc.)
- Introduction of optimizations
  - ATLAS (linear algebra; UTK)
  - Telescoping Languages Project (Indiana, Rice, UTK)

"Tensor Contraction Engine" (TCE) Concept

- Begin with high-level expression of the equations – "in the language of the domain scientists"
- Use software tools to manipulate/optimize expressions into something suitable for implementation – "global optimization"
  - Use performance models to tailor algorithms to hardware
- Generate source code, compiler can provide further "local" optimizations
- Developer produces more efficient code more easily
- Code is automatically tuned to make best use of hardware

TCE Components

- Algebraic Transformations
  - Minimize operation count
- Memory Minimization
  - Reduce intermediate storage
- Space-Time Transformation
  - Trade-offs between storage and recomputation
- Storage Management and Data Locality Optimization
  - Optimize use of storage hierarchy
- Data Distribution and Partitioning
  - Optimize parallel layout



Algorithm Synthesis Summary

- Automated synthesis of optimized source addresses both performance and productivity
- Use “tensor contraction engine” model with five distinct types of optimizations
  - Applied to high-level expressions – impossible for general-purpose compilers
- Leverage expertise and experience of large collaboration to develop expertise within ORNL
- Focus on particular domains (chemistry, nuclear physics) to develop techniques, then expand to others