Integration of Software Systems: Applying Harness, CUMULVS and CCA

James Kohl
Computer Science and Mathematics
Oak Ridge National Laboratory

Monday, December 15, 2003
Background

• Computer Science and Mathematics Division
  – Tools and Software Infrastructure to Support High-Performance Scientific Simulations
  – Variety of Scientific Domains:
    • Astrophysics, Biology/Genomics, Chemistry, Climate Modeling, Fusion, Material Science…
  – All Based on Parallel Computing
    • Monitoring and Control of Concurrent Processes
    • Management of Large-Scale Scientific Datasets
Parallel Computing Concepts

Scalar Computing

Get Me a Coffee!
Parallel Computing Concepts

SuperScalar Computing
Parallel Computing Concepts

Pipelined Computing
Parallel Computing Concepts

SIMD (Single Instruction Multiple Data) Computing
Parallel Computing Concepts

SPMD
(Single Program Multiple Data) Computing
Where Our Work Fits…

Software

Hardware
Where Our Work Fits…

Software Tools

Operating System

E.g. Linux, Windows XP, SGI Irix… (Embedded…?)

Hardware
Where Our Work Fits…

Our Arena of Focus…

Software Tools

Operating System

Hardware
Where Our Work Fits…

Software Tools

Operating System

Massively Parallel Computer Hardware
Where Our Work Fits…

Massively Parallel Computer Hardware
End-to-End Infrastructure Organization

Front-End User Interface and Display (PowerWall, CAVE, Big Bertha…)

Run-Time Monitoring and Control Middleware Visualization, Steering, Fault Tolerance

Concurrent Programming Model (PVM, MPI, OpenMP…)

High-Performance Parallel/Distributed Execution Environment
End-to-End Infrastructure Organization

Scalable High-Resolution Visual Display Facility

CUMULVS:
Collaborative Interactive System For Visualization, Steering, Coupling and Fault Tolerance

Harness:
Adaptive Reconfigurable Distributed Virtual Machine Environment built on High-Performance Clusters

Front-End User Interface and Display (PowerWall, CAVE, Big Bertha…)

Run-Time Monitoring and Control Middleware Visualization, Steering, Fault Tolerance

Concurrent Programming Model (PVM, MPI, OpenMP…)

High-Performance Parallel/Distributed Execution Environment
Harness:
Flexible Robust Parallel Operating Environment

- **Parallel** “Plug-In” Software Components
  - Dynamically Adapt to Changing Program Needs
    - Not Forced to “Fit” into Monolithic Infrastructure
- **Distributed Peer-to-Peer Control**
  - No Single Point of Failure, Robust Protocols
    - Consistent Global State/Database Updated in Parallel
- **Share Resources and Migrate Applications**
  - Merge and Split Virtual Machine Sets
    - Owner Controls Access to CPU, Storage and Network
Harness Features

Self-Assembling Virtual Machine

Parallel Plug-in Software Components
Provide the Capabilities:
Network, Programming Model,
Numerical Libraries,
Front-End Tools/GUIs.

Fault-Tolerant MPI Extensions
~ “FT-MPI”

Robust Fault-Tolerant Environment

Multiple Tools and Parallel Paradigms
Used Simultaneously

Long-Running Simulations
Often Exceed the MTBF for Large
High-Performance Cluster Computers!

→ Distributed Control Algorithms Maintain
Global State in the Presence of Faults and Failures
CUMULVS
Interacting with Simulations On-The-Fly
(Collaborative, User Migration, User Library for Visualization and Steering)

• Interactive Visualization of Intermediate Data
  – Monitor the Ongoing Progress of a Simulation

• Coordinated Computational Steering
  – Teams “Close the Loop” to Control the Simulation
CUMULVS Features

Visualization
Dynamic Attachment of Multiple Independent Viewers

Model Coupling
Natural Extension of Protocols
→ Models Share Parallel Data
→ Parallel Data Redistribution (CCA “MxN”)

Steering
Multiple Simultaneous “Steerers”
→ Updates Applied in Parallel Unison
→ Algorithmic or Model Parameters, or Truncate Runs Gone Awry…

What If?
Explore Non-Physical Effects
CUMULVS
Application-directed Fault Tolerance

• Simulation Defines Key Program State
  – Scientist Decides What & When to Save
• Run-Time Daemon Watches Simulation Tasks
  – Automatic Restart or Rollback on Failure…
• “Semantic” Checkpoints ~ Migration / Reconfig
Science Visualization Corridor

- 35 Mega Pixels
- 7.5’ high by 27’ wide
- 27 Projectors (3000 Lumens each)
  - 3-chip DLP with Dark Metal
  - A Billion Colors
- 64 Opteron nodes with Gigabit Ethernet, Quadrics Interconnect
- nVidia FX5900 Ultra or ATI Radeon 9800 Pro Graphics
Simulation results of core collapse supernovae mechanisms from John Blondin and Tony Mezzacappa as a part of the SciDAC Terascale Supernova Initiative (TSI).
Climate Applications

John Drake discussing the results of bio-chemistry simulation using the LANL POP code.
The Common Component Architecture (CCA)

• Component-Based Framework to Manage the Complexity of Scientific Simulation Software
  – Successful Business Software ~ Corba, DCOM, EJB
  – Add Performance, Languages, Science Data Models

• National Forum (Open) / DOE SciDAC Center
  – Several Major National Labs and Universities

http://www.cca-forum.org/
Why Components?

The task of the software development team is to engineer the illusion of simplicity [Booch].
CCA Components & Frameworks

- Strict Component Boundaries
  - Enforces Interfaces Better than Plain OO
- High-Performance Solutions
  - “Direct-Connect” Invocation Short-Circuiting
    - Virtual Function Call Overhead Within Process
  - “MxN” Parallel Data Redistribution
    - High-Level Parallel Data Exchanges & Operations
    - Substrate for Inter-Framework Interactions
CCA Concept: Ports

- Components interact through well-defined interfaces, or *ports*
  - In OO languages, a port is a class or interface
  - In Fortran, a port is a bunch of subroutines or a module
- Components may *provide* ports – implement the class or subroutines of the port
- Components may *use* ports – call methods or subroutines in another component’s port
- Links denote a caller/callee relationship, *not dataflow!*
  - e.g., FunctionPort could contain: `evaluate(in Arg, out Result)`
Ports, Interoperability, and Reuse

• “Ports” represent “Interfaces”
  – They define how components *interact* …

• Generality, Quality, Robustness of Ports:
  – Up to designer/architect… *The CCA Cannot Do It For You!*
  – “Any old” interface is easy to create, but…
  – Developing a good “standard” interface requires thought, effort, and cooperation!

• General “plug-and-play” interoperability requires components to *conform* to the same interface.

• Interoperability & reuse requires “standard” interfaces
  – “Standard” may merely mean “widely used”…
General Interface Example

Get Beverage()

Get Coffee;
Add Sugar;
Add Cream;
Top With Whipped Cream;

Add Extra Flavoring()

Get Tea;
Add Sugar;
Add Lemon;
Insert Sprig of Mint;

Do Final Presentation()
General Interface Example

- Get Beverage()
  - Get Coffee;
    - Add Sugar;
      - Add Cream;
        - Top With Whipped Cream;
  - Add Extra Flavoring();
    - Do Final Presentation();
  - Open Bottle;
- Get Beer;
General Interface Example

Get Beverage()

Get Coffee;
Add Extra Flavoring()
Add Sugar;
Add Cream;
Do Final Presentation()
Top With Whipped Cream;

Get Gin;
Get Vermouth;
Add Ice (?);
Shake Well;
Drain Off Ice;
Add Olive;

Kohl-2003/30
CCA 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)

(CUMULVS is the “Glue”!)
Scalable Visualization Cache Architecture

- Increasingly Massive Scientific Data Sets
  ⇒ Too Large to Fully Explore / Visualize Interactively
- Modular, Layered Viz Cache Framework
  ⇒ Parallel Storage, Analysis & Reduction Per Layer
  ⇒ Independent Memory & Disk Cache Per Layer
  ⇒ Navigate & Zoom Through Hierarchy

CCA / MxN CUMULVS
SDM & ASPECT Data Reduction

Terabyte+ Data
Reorganized Data For Viz

R₃()  R₂()  R₁()  R₀()

100s of GBs 10s of GBs GBs 100s of MBs

Parallel Rendering Kohl-2003/32

Zoom Source
Multi-resolution Hierarchy
Display

Reduction & Filtering
Summary

• Harness  (geistgaii@ornl.gov ; http://www.csm.orl.gov/harness )
  – Adaptable Parallel Operating Environment

• CUMULVS  (kohlja@ornl.gov ; http://www.csm.orl.gov/cs/cumulvs.html )
  – Visualization, Steering, Coupling, Robustness

• Visualization Facility  (fanngi@ornl.gov ; http://www.csm.orl.gov/viz )
  – High-Resolution and Immersive Displays

• CCA ~ Common Component Architecture
  – High-Performance Components and Interfaces