CCA-PIO & CCA-LSI:

Fang Liu, Yongquan Yuan, and Randy Bramley
{fangliu, yyuang, bramley}@indiana.edu
Extreme Computing Lab, Indiana University, Bloomington, Indiana

Project Overview
This work targets the emerging use of software component technology for high-performance scientific parallel and distributed computing. The use of Common Component Architecture (CCA) framework provides opportunity to encapsulate the complex scientific software and to reuse them easily. BABEL compiler from LNL provides language interoperability among scientific packages.

This poster presents an effort on designing minimal set of interfaces for HPC parallel file I/O libraries and HPC linear solver libraries. Preliminary experiments show that overhead introduced by adding the CCA layer on original software package is very small.

Motivating Simulations
The large-scale scientific simulation often combines use multiple software packages developed by different groups in different computer languages. For example, computational aero/physics, chemistry and fusion, each has challenging resolution and complexity requirements that demand massively parallel computing resources and a range of sophisticated software. CCA provides an application composition model based on reusable components. Common interface needs to be defined to facilitate the interaction between components.

This diagram presents computational phases in typical multidisciplinary scientific applications.

CCA-PIO Design
What CCA-PIO stands for?
A CCA Common Parallel File I/O Interface.

Design Architecture

Why CCA-PIO is needed?
• High-level, easy-to-use parallel I/O interfaces to manipulate structured data such as arrays to facilitate transient objects between components that might be running on different numbers of processes.
• Interface, programming language, and amount and type of metadata required by different I/O libraries will vary.
• Files written by one data library interface are not directly accessible by another.

SIDL Interface
CCAPIO uses SIDL to provide a universal, minimal set of parallel I/O interfaces for data and metadata, spanning multiple data libraries.

package CCAParallelIO {  
  interface ParallelIOM {  
    int openFile (...);  
    int openFile (...);  
    int read(...);  
    int read(...);  
    int write (...);  
    int write (...);  
    interface metadata {  
      int getMetaData (...);  
      ...  
    }  
  }  
  interface SIDL {  
    dataStore in interface  
    Setup array template  
    Open file  
    Manipulate data  
    Prepare Interfacedata metadata  
    Get information about data.  
  }  
}

CCA Parallel I/O Component
• Provides interface compatibility between components
• Provides language interoperability (Fortran, C++, Java, Python)
• Enables parallel I/O data sharing across components
• Maintains data portability over platform
• Allows re-use of existing data libraries

CCA-LSI Design
What CCA-LSI stands for?
A CCA General Sparse Linear Solver Interface.

Design Architecture

Sparse linear solver is the most essential and computationally intensive part of HPC applications. Our design idea is to decouple the HPC applications from the solver libraries they use by adding Sparse-Solver interface between the application and solver libraries. Application user doesn’t have to learn individual underneath solver libraries, but simply use our interface and implementation.
Currently, the implementation is done for the Trilinos, Petsc and SuperLu.

SIDL Interface
CCASLI uses SIDL to provide a minimal set of interface for parallel sparse linear solver libraries.

package SIDL {  
  interface SparseSolver {  
    Setup linear system  
    Setup solve parameters  
    Solve  
    Iterative and direct subinterfaces:  
    Specifications for algorithm, parameters, settings.  
  }  
  interface ZeroSolver {  
    setup solver parameters  
    Solve  
    ...  
  }  
  interface SparseIterativeSolver {  
    Setup linear system  
    Setup solve parameters  
    Solve  
    ...  
  }  
  interface SparseDirectSolver {  
    Setup linear system  
    Setup solve parameters  
    Solve  
    ...  
  }  
  interface SIDL {  
    int setup solver parameters(...);  
    int solve (...);  
  }  
}

CCA Parallel Linear Solver Component
Implementation of the CCASLI provides a reusable CCA linear solver component which relieves the application developer from changing application code whenever changing the solver libraries. Along with BABEL generated client stub, it provides language interoperability.

This work is supported by DOE grant...