Problem Solving Environments for Parallel Scientific Computation

Jack Dongarra
Univ. of Tenn./Oak Ridge National Lab
dongarra@cs.utk.edu

History

- 1960s - First Organized Collections
- 1970s - Advent of Libraries Plotting Packages, Statistical Packages, "Prototype" Interactive Packages,
- 1980s - Continued Development of Libraries, Emergence of Computational Packages, Emergence of Graphical Systems (data visualization)
- 1990s - Development of libraries and packages, Integration, inter-product links
- 2000+ - A "Software Parts" Industry

History

- 1960s - Little or no standardization
- 1970s - Uniform subprogram interfaces, Prototype Command-line systems, Move toward portability.
- 1980s On-line documentation, emerging graphical standards, Prototype WIMP.
- 1990s Hypertext, Point and Click, Move towards Inter-operability

Problems with Current HPCC Software and Technologies

- Incompatible tools use different data formats, programming models, protocols, and user interfaces.
- Users may be unable to differentiate between large number of functionally different but superficially similar tools or software modules.
- Use of tools and software often requires in depth knowledge of parallel programming or numerical analysis.
- Application scientist may be at early stage of problem solving process and be unsure how to proceed to next stage.
- Large scale Grand Challenge and National Challenge problems are multidisciplinary and involve both information processing and computation.
- User may not have appropriate hardware and software or may lack expertise to install software.

Desirable Properties of Problem Solving Environments

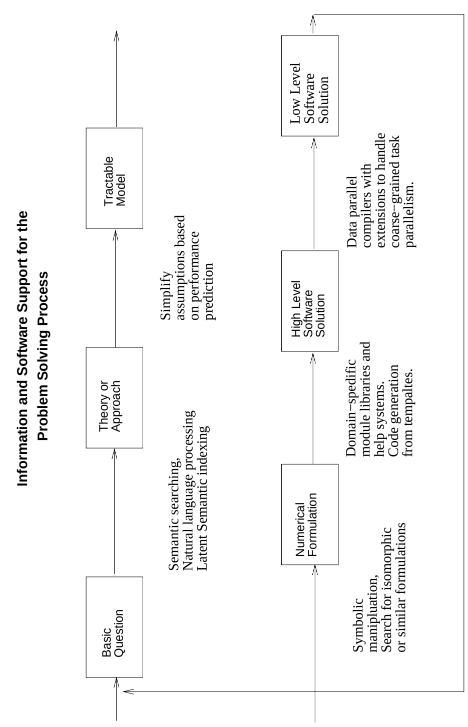
- Support for all stages of problem solving process
- Integration of different tools and process stages
- Interface with user in terms of user's language and level of abstraction
- Domain-specific adaptation and customization of tools
- Support for collaborative work

Enabling Technologies for Problem Solving Environments

- Taxonomic and conceptual domain models and ontologies
- Semantic information retrieval with relevance feedback
- Expert and experienced-based systems
- Network-accessible computational servers
- Agent and applet technologies
- Safe execution environments for mobile code

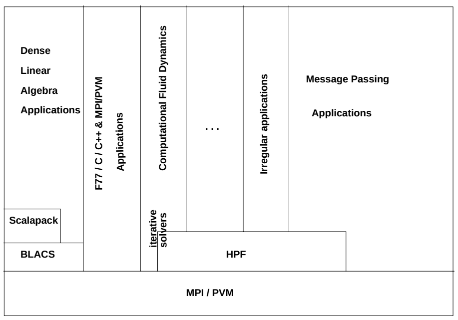
Proposed Research on PSEs for High Performance Computing

- Increased support for information retrieval and information processing aspects of complex problem solving
- Development of taxonomic and conceptual domain models for subdomains within HPCC
- Integration and domain-specific adaptation of problem solving tools and software
- Resolution of security issues for and development of agent and applet technologies
- Support for collaborative solution of complex multi-disciplinary problems



Application Development Methodology

- Adhere to accepted standards wherever possible
- Tool supported application development
- Develop tools and libraries to render parallelization more effective and less error-prone



End-User Requirements on Parallel Application Development

- Efficient parallelization
- Portability
- Minimization of software development efforts

- Development of an application engineering environment for parallel and distributed systems
- User-centered and application-driven
- Easy-to-use in scientific and engineering domains

- PSEs have been developed for certain areas of mathematics, for example Matlab for linear algebra computations, and commerce, sophisticated PSEs for computational scientists are generally lacking.
- Most current PSEs are designed to run on PCs and workstations, rather than on massively parallel computers or networks of workstations.
- PSE proposed will encapsulate expertise, enhance scientific productivity, and lead to a more effective use of computational resources.
- The PSE will be a complete environment, assisting scientists in developing applications, formulating input, and executing programs.
- The compilation, job control, and execution components of the proposed PSE will be based on a meta-computing environment.

Library Advisory System

- Purpose
  - Advising users about which algorithms, libraries, and/or tools are most appropriate for their specific problems
  - World Wide Web (WWW) interface to a knowledge base holding information about software libraries
  - Applicable in the fields of scientific computing and commercial applications
- Two different expert systems proposed for investigation
  - Driven by manually entered evaluation functions
  - Driven by supervised learning

Ongoing Activities and Future Directions

- Parallelization of Algorithms and Applications
  - Parallelization of Templates for systems and eigenvalue problems
  - Take advantage of international collaboration in HPC
  - integrate and coordinate between academic institutes, software and hardware vendors, and business enterprises
  - Library related issues
  - Evaluation of tools and applications
  - Tools to support the software development cycle
- Tool Environment Development
  - NetSolve
  - Performance modeling and estimation
  - User interface
  - Incorporate in production software tools

Information Structuring Toolkit

- Purpose
  - Browsable information systems like the WWW are very labor intensive to install and maintain
  - We propose development of a web based groupware toolkit that help communities of people to create structured information systems
- Features
  - Basic units of information are the object and object attribute value
  - Tools for transforming/filtering/merging information
  - Use of related techniques from IR and conceptual data analysis
  - Facilities for developing custom groupware IS applications
  - Written in Java, hence portable, and easily extendible

Research Questions Addressed

- How can easy programming be achieved (in particular for non-experts)?
- How can a powerful software reuse mechanisms be realized?
- How can good portability, scalability, and parallel efficiency be ensured?

Examples Heterogeneous Networks
Even on IEEE machines results may differ between machines, compilers and compiler switches.

- An iteration where the stopping criterion depends on the machine precision. Stopping criteria for iterative methods – may not be satisfied on each processor simultaneously
- Processors sharing a distributed vector v compute its two-norm, and depending on that either scale v by a constant much different from 1, or do not.
- Bisection for finding eigenvalues of symmetric matrices.
- Eigenproblem for a tridiagonal matrix – run QR on each processor and each processor finds k eigenvectors. But each processor may compute a different QR sequence
- Adaptive quadrature – $[a, b + \epsilon_1], [b + \epsilon_2, c]$

Heterogeneous Networks

- Challenges associated with writing reliable numerical software on networks containing heterogeneous processors
- Processors which may do floating point arithmetic differently
- Even supposedly identical machines running with different compilers or even just different compiler options
- The basic problem lies in making *data dependent branches* on different processors

Heterogeneous Networks

- Defensive programming
- Machine parameters
- Communication and representation between processors
- Controlling processor
- Additional communication
- Testing strategies

Summary

- The PSE is evolutionary in terms of the computing resources used.
- The proposed PSE will also allow incremental additions to the software resources of the environment.
- As new numerical methods are developed it will be a simple matter to incorporate them into the software resources accessible by the PSE.
- The PSE will feature not only complete applications, but also an application editor that allows a user to graphically modify an existing application, or to build a new application from scratch.
- The application editor will be fully integrated with an on-line documentation system, and context-sensitive help.

Perspectives

- Software reuse mechanisms
- Human-machine interfaces
- Interactive guidance mechanisms
- Distributed computing environments

Methods

- Formal specification languages
- Knowledge-based systems
- Automatic program synthesis techniques

Key Results

- Two problem classes addressed
  - Stencil-based problems
  - Numerical linear algebra problems
- Features of prototype environments developed for both classes
  - Application-class specific problem description formalisms
  - Reusable software components
  - Knowledge-based system to support selection of the most appropriate software components
  - Interactive user guidance mechanism
  - Automatic program synthesis techniques to ensure an effective and transparent coding process
- Portability, Scalability, and parallel efficiency addressed at the level of high-level, reusable software components

An Environment for Stencil-Based Problems

- Graphical user interface to support easy specification of the problem
- Design skeletons used as reusable software components

Three Layers

- The top level is an intelligent graphical user interface for application use and development.
- The lowest level are software libraries and modules for mathematical and scientific computation.
- The intermediate software layer consists of "middleware" for coordinating the upper and lower software layers, for job compilation, execution, and monitoring, and for managing the on-line documentation and help subsystems.
- The PSE will be designed so that creating a new application-specific PSE affects only the upper and lower software levels, leaving the middle layer unchanged.

Intelligent User Interface

- A graphical editor for creating and modifying applications;
- A set of computational templates for rapidly prototyping new applications;
- Tools for composing application modules;
- Job submission and control interface

Application editor

- Also facilitate the building of new codes by graphically editing existing codes and incorporating user-written modules.
- A user can build their own application from the ground up using the software libraries supplied by the PSE or by using their own software.
- The graphical editor graphically displays an application as an hierarchical flow chart.
- At the highest level the flow chart displays the complete application.
- Clicking on a component of the flow chart will display a flow chart for that component.
- At the lowest level of the flow chart hierarchy actual code is displayed.
- A user can edit the application by modifying the flow chart hierarchy at any level.
- The application editor will incorporate extensive online, context-sensitive help.
- Users can query the help system to get a description of what any component of a flow chart does, together with a summary of the input and output variables for that subprogram.
- At the lowest level of the hierarchy, clicking on a variable will display information about it.

Application templates

- An application template is a flow chart hierarchy in which some of the nodes must be supplied by the user subject to certain interface constraints.
- The user may either make use of existing modules supplied by the PSE or insert their own.
- Application templates are provided for the rapid prototyping of new applications.

Complete application programs example

- Many material science application codes are scientifically and computationally complex.
- As part of the proposed PSE, several of these advanced codes will be provided. These codes include:
  - the only first principles local density approximation based (LDA) $O(N)$ (where N is the number of atoms comprising the system), locally self-consistent multiple scattering method (LSMS),
  - the LDA-based linearized muffin-tin orbital method (LMTO),
  - an LDA-based pseudo-potential method that includes relaxation effects,
  - the semi-empirical tight binding molecular dynamics code,
  - a set of classical molecular dynamics codes.
- These will serve not only as useful complete application codes, but will also be made available in template form so that scientists can modify them and/or implement additional modules in order to treat physical phenomenon that are not currently contained in these models.

Middleware Components

- The middle layer of software in the PSE has two main components.
- Architecture for designing and building system services that provide the illusion of a single virtual machine to users, a virtual machine that provides secure shared object and shared name spaces, application adjustable fault-tolerance, improved response time, and greater throughput.
- A system for examining online documentation on the PSE itself and the various applications and modules embedded in the PSE.

Middleware Components

- Parallel object-oriented language and compiler
- Support for PVM/MPI
- Support for legacy and other language codes
- Resource management
- Transparent file and data access
- Fault-Tolerance
- Post-mortem debugger
- Online documentation subsystem

Low-Level Software Components

- Compilers,
- Debuggers,
- Performance
- Analysis tools,
- Application software,
- Libraries, and
- Files accessed by the online help and documentation systems.
- These resources are managed by the higher software levels, and are not directly accessible from by the user.