On The Future of Problem-Solving Environments

Elias Houstis and John R. Rice, Purdue
Geoffrey Fox, Florida State University
Randall Bramley, Indiana University

Overview

- Some specific applications for PSEs - and implications
- Design principles for future PSEs
- Directions for future PSE development
- Some outstanding research problems for PSE work

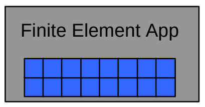
Existing Systems

- Four framework flavors, with most systems a blend
- **Workbenchs**: give sense of work arena to user
  - Matlab, Maple, PELLPACK, PetSc, NetSolve, Cumulvs, ... ,
- **Component composition**: user wires together modules to create application
  - SCIRun, Component Architecture Toolkit, WISE, COG Kits, ...
- **Code composition/generation**: end product is code in a programming language, which is then compiled and run
  - POOMA/PAWS, SciNapse, Falcon, ...
- **Collaborational**: targets multiple users sharing work
  - Tech Talk, Shastra, Intelligent Archive, ...

Motivating Examples

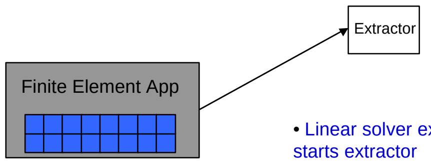
- What do users need in future from PSEs?
- What features are current applications pointing us towards?

Application 1: Mold Filling



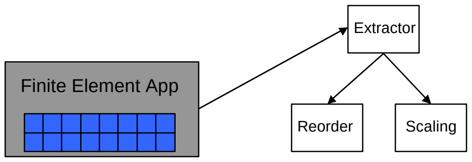
- Engineer starts parallel mold filling finite element simulation in Montréal
- Five days later, failures in sparse linear solver occur
- Engineer contacts linear solver expert in Indiana

Application 1: Mold Filling



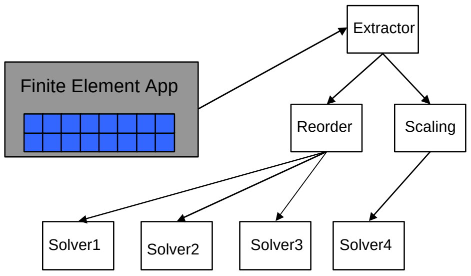
- Linear solver expert starts extractor process on his local machine

Application 1: Mold Filling



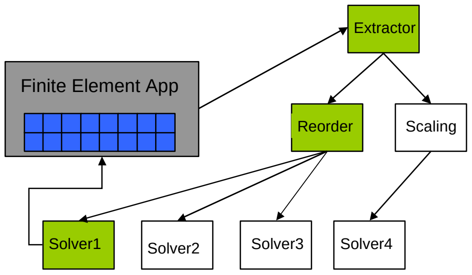
• LS expert starts testing various solution strategies

Application 1: Mold Filling



• LS expert identifies effective solution strategy

Application 1: Mold Filling



• Most effective LS solution is fed back to Montreal

Application 1: Mold Filling

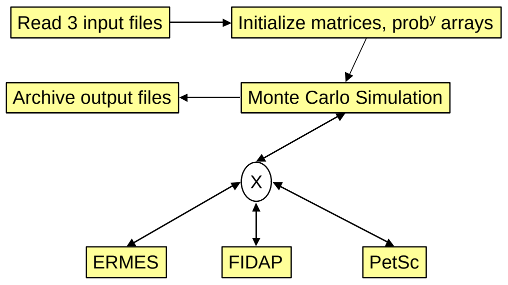
Crucial aspects:

- Remote collaboration with area experts
- Solution strategies hot-wired into running code
- Engineers may not be able to share their code or the model data with remote linear solver expert

Application 2: Chemical Engineering

- NCSA Alliance application for “Science Portals”
- Electrochemical deposition and dissolution with corrosion
- Sequence of applications, tied together via files/scripts
- Continuum model (PDE) for transport and reaction in electrolyte bulk solution
- Monte Carlo for molecular scale events (surface region)
- Monte Carlo decides needs new flux updates, writes out concentration files
- Continuum model reads those for its boundary conditions, computes, writes out new fluxes

Application 2: Chemical Engineering



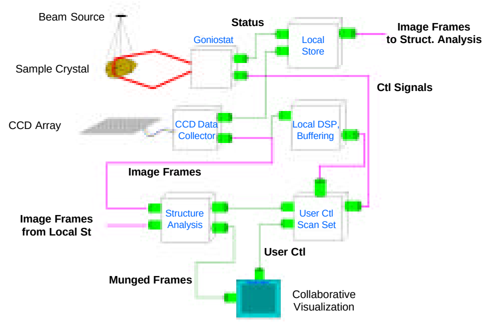
Application 2: Chemical Engineering

- Crucial aspects:
  - Transfer is via files ... and users want to keep it that way (or at least have files for archiving).
  - Modules are commercial code ... and just *try* to get the source for FIDAP. Means we cannot recompile code.
  - Mixing and matching of components is desired (choose among three solvers for continuum model). PDE solve is cheap w/r to MC solver, so may want multiple ones simultaneously

Application 3: X-Ray Crystallography

- X-Ray crystallography to determine molecular structure
- Better resolution from high-energy, high-flux beam lines at LBL and ANL
- Procedure
  - University scientist ships sample to LBL
  - Scientist watchs mounting of sample via teleconferencing
  - Scientist sends initial set of scan commands
  - Preliminary frames are processed by local/remote components, get sent to scientist who manipulates images to check crystal, mounting, ...
  - Scientist sends new scan parameters - or terminates beam-line session
  - At any time, scientist can allow remote colleagues to view partial results and confer

Application 3: X-Ray Crystallography



Application 3: X-Ray Crystallography

- Crucial aspects:
  - Network aware computational/data store modules required
    - Dynamic negotiation of protocols and capabilities, on a per-message basis if possible
    - Ability to restore lost links, re-establish state
    - Co-allocation of multiple QoS streams with differing service levels
  - Human in the loop ... at every stage and phase of this application, computation is ***not*** the bottleneck.

Requirements Suggested by Apps

- Multilanguage systems are common: each of the three uses Perl, Java, Fortran, C, C++.
- Must be able to deal with modules which are executables with no access to source code
  - Cannot recompile the module to include PSE framework libraries, utilities.
- Must assure security for sensitive code providers
- Noncomputing resources must be integrated with traditional computing capabilities
  - remote instruments
  - mass storage systems
  - pesky humans.

Design Principles

- Significant, widespread agreement on several design principles for future PSEs
- Two major categories
  - Human centered design
  - Component-based, modular design

Design Principles: 1

- **Human-centered design**: the user is the design focus.
  - Interaction systems structured around how end-users work, instead of the underlying computational architectures that support the work
  - Problem statements natural to application, not a mathematical abstraction (e.g. Matlab's pdetool, SciNapse interfaces)
  - Make tailoring and customization easy and optional
  - Provide hierarchical access
    - modules and utilities are blackboxes unless user chooses to open them
    - multiple levels of abstraction

Design Principles: 2

- **Component-based architecture**
  - Base subsystems, algorithms, utilities, user interaction systems are interchangeable plug-and-play components. Should be able to access PSE from GUI, Perl, Python, programming language, or another PSE
  - Language interoperability.
  - Separate macroscopic from microscopic "components"
  - Allows wider range of capabilities and dynamic updating of a PSE
  - Lets area experts in different fields contribute, maintain, and improve capabilities without requiring similar expertise from users.
  - Modern scientific computing entails huge data sets; be able to move the component to the data instead of other way around. Data may be immobile or sensitive.

Future Features of PSEs

Future Direction: Recommender Systems

- Recommender systems will play a major role in PSE. Already have a confusing and large array of choices:
  - problem formulation
  - algorithm selection
  - software components to choose
  - resources on which to run modules
- RS's will be context sensitive for applications
- RS's will dynamically adapt to a user's background, expertise, current goals
- First must provide the most basic RS: manuals, README files, documentation !

Future Direction: Math Libraries

- Still need improvements in the golden oldies: linear systems, PDE's, ODE's, signal processing, ...
- Tremendous need for libraries that generically handle
  - Dynamic graph and tree algorithms
  - N-body simulations with wide variety of force functions, constraints
  - Integral equations with nonstandard kernel operators
- Robust, reliable, libraries which provide multilevel information to user

Future Direction: Interaction with Resources

- Remote instrument access and control
  - telescopes, x-ray crystallography, ocean buoys, general lab equipment, robotic spacecraft to Mars
- Sophisticated data generation systems
  - on-line data bases, data mining systems, user-supplied simulations
- Presentation tools
  - integrate with standard desktop tools, visualization, production of papers, electronic publication on Web
- Rather than have PSEs *incorporate/integrate* such resources, need to have them *interoperate*. Model of future is not encapsulation, but peer-to-peer interactions.
- PSEs that dynamically attach to PSEs in other domains, work with them, then detach.

Future Direction: Web-based Interfaces

- Scientists increasingly use Web browser as their lab notebook.
- Users will access full range of scientific computing activities via Web (c.f. HotPage)
  - Setting up and launching applications
  - Monitoring and steering
  - Detaching and reattaching to running simulations
  - Recording results, archiving files, controling instruments
- Remote access via a range of portals; view results in different ways from I-Desk, workstation, laptop, handheld

Future Direction: Distributed Computing

- "Grid" paradigm of distributed computing gives users a wide choice of resources to use.
- Constant Moore's Law drumbeat means many sites will provide resources to outsiders (c.f. Web servers)
- Distributed resources will be used without user action
- Laying out a multi-component system onto Grid optimally means solving dynamic network flow problem, which changes on the order of milliseconds.

Research Issues for PSEs

- Bottlenecks increasingly are not computational
  - more sophisticated numerical techniques
  - large data creation, movement, caching - across the Internet
  - user interaction is often the main bottleneck
  - interfacing and connecting disparate systems (hardware, software, human notational)
- Computers and storage systems can now record every keystroke and timestamp it.
  - Which of this massive information is useful for problem solver tracking and logging?
  - What should be kept to later capture a state (of computation, human mindset, ...)

Research Issues for PSEs

- Recommender systems now require hand-crafting, large investment of time/effort by experts.
  - Can a generic recommender architecture relieve the burden?
  - How much can case-based reasoning systems do automatically?
- Collaborational mechanisms are increasingly important
  - Who gets control, and what type of control? Need scalable approach to collaboration as well as algorithms.
  - How to protect sensitive data/codes in a collaboration framework?
- Cost-mitigation mechanisms
  - How to leverage commercial software, desktop tools?
  - How to share software development amongst multiple PSEs?

Research Issues for PSEs

- Component based PSE design
  - How to develop the required high-level interface standards?
  - What interoperability standards to address, when to do it?
  - How to reward the extra effort needed to make a code into an interoperable, reusable component? [Reward may not be $$]
  - How to deal with separate issues in microscopic (e.g., algorithm implementation) versus macroscopic (full program) components?
- Distributed components in PSEs entail
  - QoS contracts, fault tolerance
  - System info that lets components move adaptively
  - Components that provide introspection and reflection
  - How to catalog, "advertise", distribute new components?

Research Issues for PSEs

- Error detection and handling
  - "Errors" have many flavors: catastrophic, accuracy attainment failure, ...
  - Errors can occur at many levels:
    - inside an algorithm
    - when connecting modules
    - in the user interaction system
    - on the network
  - Automatic failure detection, checkpointing, restarting
- Performance evaluation and analysis
- Security in an increasingly bad neighborhood (Internet)