Parallel and Distributed Processing in a Problem-Solving Environment for Environmental Science

José C. Cunha
Parallel and Distributed Processing Group
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Portugal
(jcc@di.fct.unl.pt)

June 1999
Index

- 1\textsuperscript{st} part: PSE: A Global Perspective
- 2\textsuperscript{nd} part: Work at UNL Towards a PSE for Environmental Science
- 3\textsuperscript{rd} part: Conclusions
Problem–Solving Environments

- Integrated environment supporting:
  - entire life cycle
    * development and execution steps
    * to solve a given problem
    * with easy access by an end–user,
    * a scientist or engineer from a given domain
Development Steps

- Tools to help problem specification; design, analysis, verification, evaluation:
  - Rapid prototyping
  - Dependent on a specific domain
  - Expert assistance
Execution Steps

- To support online/offline observation and control of
- scientific experiments / simulation processes
- Activities performed on multiple heterogeneous components (application–specific and generic tools):
  - selection, evaluation and testing
  - configuration, activation, interconnection
  - monitoring, controlling
Hetereogeneous Collection of Interconnected Components

- Sequential, Parallel, Distributed Problem Solvers
- Tools for data and result processing, interpretation, visualization
- Interactive steering: user and agent driven
- Online access to large databases
Requirements

- Complex (simulation) models
- Large volume of input or generated data
- Difficult interpretation and classification
- Reuse of components
- Dynamic configuration
- Dynamic modification of interaction patterns and operation modes according to the needs and evolution of each experiment
Multidisciplinary nature:

- Heterogeneous / hybrid components / models
- Interactions among multiple users, collaborative environments
● Component Integration
  – Statically specified
  – Dynamically inserted and removed from an existing configuration
  – Multiple dynamically changing interaction patterns

● High Degree of User Interaction
  – Distinct operation modes (offline/online data interpretation or visualization)
  – Distinct user interfaces
  – User driven control (steering) of an ongoing computation
  – Agent driven control
• Dynamic Reconfiguration
  – Components dynamically enter / leave the environment
  – Component coordination
  – Multiple users concurrently join ongoing experiments with distinct roles (observers, controllers)
  – Consistent views
Conceptual Layers
Figure 1: Conceptual Layers
Formalisms for Software Architectures

- High-level specification of components, their composition, their interactions, for a given problem
- Modeling and reasoning on the global structure and behavior
- Semantics of interactions through the component connectors
- Specification languages for:
  - Description of system structure and analysis of system behavior
  - Incremental refinement and composition of architectures
• Coordination Models
  – Represent and manage patterns of interaction among components
  – Define cooperation and communication models
  – Guarantees of consistency

• Resource Management Services
  – Configuration of parallel and distributed heterogeneous virtual machines
  – Activation of component instances
  – Mapping and load balancing
  – Local scale and large scale operations
  – Management of metacomputing resources
• Interconnection Services
  – Models and infrastructures for heterogeneous components

• Monitoring and Control
Research Approach

- Short term
  - Build PSE for specific domains
    * Cooperation with scientists / engineers
    * Identification of user/application requirements
    * Early and incremental development of prototypes
    * Quick user feedback
  - Make them evolve towards advanced PSE to ease development and execution of complex applications
• Medium / Long term

  – Generic PSE to be tailored to specific problem domains

  – Tools for the more/less automatic generation of application-specific PSE

  – Advisoring/explaining tools to assist the user

    * During development time (correctness/performance)

    * During execution time (impact of parameter modification upon system behavior)

  – Integration of numeric, symbolic, multimedia, intelligent knowledge processing and discovery, database components
Further Challenges?

Abstract Specification of a PSE. To be submitted to a Meta–Environment that will generate a specific working PSE.

Layers. From a formal specification to the runtime support:

1. Formal specification of software architecture: components and connectors; structure and semantics of interactions.
2. Tools to reason about global system properties.
3. Tools to support transformation between software levels.
5. Specific PSE: working collaborative environment and tools.
Open Issues

- How to generate "Simple PSE", i.e. with only a few components?
- How to build and validate the above mentioned Meta–environments? Through intensive experimentation.
- How to achieve "suitable" component–based middleware and supporting infrastructures?
  - Standards
  - Expressiveness (e.g. how to express coordination issues?)
  - Efficiency (e.g. how to interact with parallel components?)
Work at UNL/Lisboa Towards a PSE for Environmental Science

- Multidisciplinary Project
  - Framework to support Parallel and Distributed PSE (Parallel and Distributed Processing Group headed by Prof. José C. Cunha, Department of Computer Science, UNL)
  - Tridimensional Optimal Layout of WasteWater Treatment Plants (WWTP) (Group headed by Prof. David Pereira, Department of Environmental Sciences and Engineering, UNL)
- Integration of separate/distributed/heterogeneous components
  * distinct programming / computational models
  * distinct / hybrid problem–solving strategies
- Parallel and distributed processing
- Interactive / adaptive control
- Easy access by the end–user in problem specification, development and execution control
- Dynamic reconfiguration
- Multiple cooperative users
Integrated Environment

Global View. Several sub–models (unitary operations, hydraulic, economic) are coordinated by a central model, resulting in a complete computer aided design tool.

- Data exchange between sub–models and central model: central model sends data (partial input) and gets results (partial output).
- Interaction may use a subroutine style or communication between independent processes.
- Parallelization is necessary for the optimization problem.
Types of Blocks. Three classes:

- Input Models
- Design and Optimization
- Output Models
Optimization

Combinatorial problem needs a mixture of heuristic rules and methods to reach acceptable solutions

- Optimization as a best solution for a given layout or choose the best design
- Use several techniques, depending on the case, e.g. Dynamic Programming or Parallel Genetic Algorithms
Parallel Genetic Algorithms Environments

- GA Approaches:
  - Sequential
  - Parallel
  - Hybrid; Co-evolutionary computing

- Parallel GA Approaches:
  - Fine grain; Coarse grain
  - Shared–memory; Distributed–memory
  - Master–slave; Island models
• GA Support Environments
  – Application–Oriented
  – Algorithm–Oriented
  – Toolkit–Oriented

• Trend: Heterogenous Component–Based PSE for GA
Fundamental Requirements to Support the Experimentation

- Data visualization: online evolution of the GA computation
- Interactive steering
- Adaptive control
Components of the PSE
Figure 2: Computation, Visualization and Control
Experimentation: built several prototypes

- for each separate component
- for their interconnection
The GA Component

- Basis: Simple Sequential GA from David Goldberg (SGA)

- Parallel Models:

  **Single Population.** The population is managed by the master
  - subdivides it in slices, distributed to slaves
  - each slave evaluates GA function and sends results back
  - SGA–Shared Memory on NT using threads, UNL
  - Similar model used by PGAPack on MPI, by David Levine, Argonne, US
**Island Model.** The population is scattered as independent islands that evolve autonomously

- each island with distinct evolution rules/parameters
- migrations of individuals between islands
- SGA–Island on PVM, Linux LAN and Alpha Cluster, UNL
- SGA–Island on MPI, Linux LAN, UNL
The Visualization Component

**Goal.** To support online or offline visualization of the evolution of the GA objective function

**Approach.** To reuse an existing GA visualization tool, extracted from a monolithic implementation of a sequential GA (SUGAL, A. Hunter, Univ. Sunderland)

- Encapsulated as a PVM or MPI task
- Allows visualization of the evolution of multiple islands
- With dynamic integration into the environment under user control
The Interactive Control Component

**Goal.** Command and steering console

- To inspect the status of the GA computation
- To modify the GA parameters on each island and the migration
Experiments. Several implementations of the control component:

- SGA–Island PVM. The console uses PVM to interact with GA.
- Use of a Distributed Monitor (DAMS) on PVM, to support:
  - the configuration/activation of island tasks
  - the command/steering console
- Use of a Distributed Debugger (DDBG) as a steering console to dynamically modify GA parameters

Further requirements. Support user and agent driven steering
Interaction Among Components

- Using PVM
- Using MPI
- Using a group based model for interconnection of heterogeneous GA components: PVM and MPI. (PHIS model, UNL)
Conclusions

So far: Experiments on tools and mechanisms

- To test and evaluate
- Several parallel GA prototypes
- Existing GA visualization tool
- Use of a flexible monitoring and control architecture
- Use a distributed debugging tool for steering
- Use of a group based interconnection model
Drawbacks:

- Not standard tools / interfaces
- Only local area network / cluster

Current Work:

- Standard interfaces for Computational Steering
- Evaluation / use of CORBA
- Evaluation / use of GLOBUS
Challenges

- Dynamic component integration
- Distinct patterns of component interaction
- Increased flexibility in user and component interaction
- Component and tool coordination
- Multiple cooperative tools and users, sharing the state and controlling an ongoing experiment
Summary of Open Issues

- Dynamic Configuration
- High Interactivity
- Education
- User and Agent Based Observation and Control
- Computational Steering
- Coordination Issues
- Software Architectures
- Generation of PSE’s
EuroTools Special Interest Group on PSE

- EuroTools ESPRIT Working Group
- Main objective: To help end-users and tool developers to communicate and exchange ideas

URL: http://www.irisa.fr/EuroTools

- Coordinator: Jean-Louis Pazat, INRIA
- EuroTools Special Interest Groups:
– HPF/OpenMP
– PVM/MPI
– Object–Orientation
– JavaGrande EU
– Metacomputing

– **Problem–Solving Environments:** Coordinator: José C. Cunha, Universidade Nova de Lisboa, Contact: jcc@di.fct.unl.pt