Future Trends in Distributed Applications and PSEs

José C. Cunha (jcc@di.fc.ul.pt)
CITI – Centre for Informatics and Information Technologies
Dep. Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa


Contents

1. Background:
   ■ look at the past to understand current trends
2. Main dimensions in Distributed PSEs:
   ■ From an integration perspective
3. Challenges and difficulties:
   ■ Posed by new application scenarios
4. Conclusions: An optimistic view
Problem-Solving Environments (PSE)

- Integrated environments for solving a class of related problems in an application domain:
  - Easy-to-use by the end-user
  - Based on state-of-the-art algorithms

Impact of PSEs in many areas

- Fully developed PSE in the Industry, e.g. Automotive, Aerospace
- Many applications in Science and Engineering:
  - Design optimisation
  - Application behavior studies
  - Rapid prototyping
  - Decision support
  - Process control
- Emerging areas: Education, Environment, Health, Finance
- A new profile of end-user, beyond the scientist and engineer
PSE Functionalities

- Support for problem specification
- Support for resource management
- Support for execution services

Distributed and Parallel PSEs

- Distributed Computing
- Parallel Computing
- Grid Computing

Their influences upon PSEs.
Distributed Computing

- Physically distributed computations and data
- Goals:
  - Adapt to geographical application distribution
  - Provide appropriate levels of transparency

- Geographical distribution (LAN or WAN)
  - Users / Access / Processing / Archiving Sites

- Availability and Reliability
  - Fault tolerance / Redundancy
Transparency

- Depends on the layer
- Failure
- Communication (message, RPC, memory)
- Design choices can be revised
- Interactions: events, uncertainty, causality
- Loose / tight interactions / collaboration
- Pessimistic / Optimistic Choices: (DBs)
- Sometimes there is no choice:
  - mobility, disconnected operation

Transparency

- Transparency and Awareness
- The art is to make the right choices at each layer of the computing system
- The concept of transparency can / must be revised as time passes:
  - Raw hardware, Assembly, High-Level Languages, etc....Operating Systems,...., Text editors and processing tools....

- The Grid is the current revision....
Parallel Computing

- Goal: to reduce execution time, compared to sequential execution.
- Computer System Architectures:
  - Supercomputers
  - Shared / Distributed memory multiprocessors
  - LANs and Clusters of PCs
- Parallel Programming requires:
  - Decompose application in parts
  - Launch tasks in parallel processes
  - Plan the cooperation between tasks

Influences of Parallel and Distributed Computing

- Parallel and distributed computing in the 1980s and 90s had great influence upon Application and PSE developments.
Developing Parallel Applications

- Costs of task decomposition and cooperation depend critically on the system layers:
  - Application
  - Algorithm
  - Programming Language
  - Operating System
  - Computer Architecture
- How to evaluate the overall result?
  - Correctness
  - Performance
- Long term research on Models, Tools and Environments

Some Application Characteristics

- Complex models – simulations
- Large volumes of input / generated data
- Difficult interpretation and classification
- High degree of User interaction:
  - Offline / online data processing / visualisation
  - Distinct user interfaces
  - Computational steering
- Multidisciplinary:
  - Heterogeneous models / components
  - Interactions among multiple users, collaboration
- Require parallel and distributed processing
Heterogeneous Components

- Sequential, Parallel, Distributed Problem Solvers (simulators, mathematical packages, etc.)
- Tools for data / result processing, interpretation and visualisation
- Online access to scientific data sets and databases
- Interactive (online) steering

- Mapped onto a parallel and distributed platform e.g. Based on PVM or MPI
**More complex cycle of user activities**

1. Problem specification
2. Configuration of the environment:
   - Component selection (simulation, control, visualisation) and configuration
3. Component activation and mapping
4. Initial set up of simulation parameters
5. Start of execution, possibly with monitoring, visualisation and steering
6. Analysis of intermediate / final results

**Parallel and Distributed PSE**

- Try to meet more complex applications requirements
- Integrate heterogeneous components into an environment
- Transparent access to distributed resources
- Collaborative modeling and simulation
- Web-accessed
Influences of Grid Computing

- Grid computing is having great influence in modern Applications and PSEs

Layers of a Grid Architecture

- User Interfaces, Applications, PSEs
- Development Tools and Environments
- Grid Middleware: Services and Resource Management
- Heterogeneous Resources and Infrastructure
Elements of a Grid Architecture

- User interfaces and grid portals
- Applications and PSEs
- Development environments and tools
- Grid middleware:
  - Resource management and scheduling
  - Information registration and discovery
  - Authentication, Security
  - Storage access, and communication
- Heterogeneous and physical resources, and network infrastructure

Very complex systems

- Aim at providing unifying abstractions to the end-user
- Large-scale universe of distributed, heterogeneous, and dynamic resources
- Critical aspects:
  - Distributed
  - Large-scale
  - Multiple administrative domains
  - Security and access control
  - Heterogeneity
  - Dynamic
Grids enable more ambitious goals

- Enable ‘heavy’ applications in science and engineering
  - Complex simulations with visualisation and steering
  - Access and analysis of large remote datasets
  - Access to remote data sources and special instruments
- distributed in wide-area networks, and
- accessed through collaborative and multi-disciplinary PSE, via Web Portals.

![Diagram showing a user interface with access control and a global shell, leading to a pool of virtual resources including application services, computation services, dataset, data repositories, information services, storage services, and physical resources and devices.]
Question
- Just an academic exercise? No!
- Real application needs
  - Larger scale of resources for bigger, longer experiments: more accurate models
  - Easier access to remote resources
  - Increased levels of interaction for increased productivity
  - Increased capability to analyse and react

Applications and User Profiles
- Computational Grids:
  - provide a single point of access to a high-performance computing service
- Scientific Data Grids:
  - Access large datasets with optimized data transfers and interactions for data processing
- Virtual Organisations and Interactions:
  - Access to virtual environments for resource sharing, user interaction and collaboration
  - Real-time interactions for decision support
- Information and Knowledge services:
  - Access large geographically distributed data repositories, e.g. for data mining applications
Enabling more ambitious goals - 1

Access is globally unified through virtual layers:
- solve new or larger problems by aggregating available resources
- access a large diversity of computation, data and information services
- enable coordinated resource sharing and collaboration across virtual organisations

Enabling more ambitious goals - 2

- Towards uniform and standard large-scale computing environments
- Virtual resources:
  - Transient: to support experiments (computation, data, scientific instruments)
  - Persistent (databases, catalogues, archives)
  - Collaboration spaces
More ambitious application requirements

- Distinct operation modes (offline/online)
- Distinct user interfaces
- User / Agent driven control
- Dynamic modification of operation modes and interactions
- Multiple users concurrently join ongoing “experiments” with distinct roles (observers, controllers)

Dynamic characteristics

- Looking at the past:
  - Fault tolerance, Load balancing, Task spawning
- At present and in the future:
  - Changes in the configuration and availability of resources, variations of characteristics and behaviour
  - Changes at the application level: user control of a dynamic experiment
  - Flexibility to build PSEs
  - Mobility of agents and devices
More Complex Applications and PSEs

- Large number of components
- Complex interactions
- Dynamic configuration

Component integration

- Reuse of components
- Diversity of components
- Diversity of component interactions
- Components dynamically enter/leave the environment
- Component interactions change dynamically
Global conceptual layers

1. Software architectures
2. Coordination models
3. Resource management
4. Execution, monitoring and control
5. Support infrastructures
1 - Software Architectures

- Specification of components, their composition and interactions
- Modeling and reasoning on global structure and behavior
- Specification languages:
  - for structure and behavior
  - incremental refinement and dynamic composition

2 - Coordination models

- Represent and manage interaction patterns among components
- Communication and cooperation models
- Consistency guarantees
- Abstract, logical, dynamic organisation models
  - Dynamic application structure, interaction patterns and operation modes
3 - Resource management

- Configuration of parallel and distributed virtual machines
- Resource discovery, scheduling, and reservation
- Execution and monitoring at local and large scale
- Quality of service

Application Challenges

- New problem-solving strategies with adaptive behaviour
- Awareness to Quality of Service factors
  - Management at intermediate layers
  - By agents – planners
  - Contract negotiation
  - Dynamic revision of plans
  - Reconfiguration
- Specify, compose, develop, understand dynamic distributed large-scale applications: models, languages, and tools
Software Engineering Challenges

- Suitable levels of flexibility in all stages of the software lifecycle:
  - Application specification and design
  - Program transformation and refinement
  - Simulation
  - Code generation
  - Configuration and deployment
  - Coordination and control of the execution

Issues - 1

- Clear separation and representation of concepts:
  - Computation and interaction
  - Structural and behavioural properties
- Specification of multiple components:
  - Enabling alternative mappings
  - Varying degrees of automated processing
  - Supported by pattern and template repositories with relevant attributes
Issues - 2

- Mapping the programming models into the underlying computing platforms:
  - Interacting with resource descriptions and discovery services
  - For flexible configuration and deployment
- Coordination of distributed execution:
  - Allowing workflow descriptions
  - With adaptability and dynamic reconfiguration
Summary Issues in Building PSEs

- Models and tools for:
  - Component integration
  - Understanding and evaluating global properties
  - Mapping between software levels

- Component middleware and infrastructures:
  - Standardisation
  - Expressiveness
  - Efficiency
  - Local and large scale operation

Question

- Currently, basic services layers are not completely stable, standardised or fully deployed,
  so, how do we develop and experiment with the higher-lever abstractions?
How to Go From Higher-Level to Low-Level Abstractions

- Higher-Level Application Abstractions:
  - For each Class of Applications
  - For each Application Domain
- PSEs Layer
- Intermediate Frameworks
  - To provide "stable" / extensible APIs
  - To hide lower-level concerns
  - To promote incremental development
- Basic Services Abstractions

TransGrid: a CIFT Research Project

- To improve parallel and distributed environments for complex problem solving, in computational clusters/ grids.
- Several CIFT research stream:
  - Parallel and Distributed Processing
  - Multimedia and Graphics
  - Human Language Technology Streams.
- Dimensions:
  - Applications
  - Abstractions and Models
  - Tools and Environments
  - Distributed Execution
1 - Application Classes

- Parallel Text Mining;
- Collaborative Mobile Multimedia;
- Distributed Simulation, Visualisation and Steering;
- Distributed Intelligent Agent Systems.

Application Characteristics

- large volumes of data (text or images):
  - efficient search, parallel processing and input/output
- dynamic, distributed, mobile application entities:
  - appropriate structuring, interaction, coordination
- integration of distributed heterogeneous components in a highly interactive environment:
  - supporting dynamic reconfiguration of components, and execution at a small or large scale
- organisation, management, coordination in a distributed agent system:
  - dynamic organisation and intelligent agents.
2 - Abstractions and Models

- Design Patterns
- Dynamic Groups
- Distributed Logic Agents

Design Patterns

Cecília Gomes, José Cunha, and Omer Rana

- Patterns as first-class entities
- To use patterns to abstract commonly occurring structures and behaviours in distributed dynamic environments
- To integrate them into grid environments
Groups, as an organisation and cooperation paradigm in distributed systems.
- A large complex system organised in groups, which may be further structured forming hierarchies.
- Interactions among group members are more easily managed due to its smaller scale, thus enabling more appropriate coordination paradigms.

To exploit group concepts, in order to handle scalability, dynamism and mobility.
- A high-level group-oriented model:
  - for the dynamic organisation of distributed agents
  - Integrating point-to-point, multicast, and logical shared-memory interaction models
Challenges in Groups Design

- Geographical location / proximity
- Local and spontaneous communities in mobile worlds
- Structuring units in hierarchies
- More efficient forms of interaction
- Trust relationships
- Specialisation of services
- Cooperation:
  - parallel / load balancing / fault tolerance
  - access to a shared logical state

Distributed Logic Agents

Rui Marques, José Cunha, Terrance Swift (U. Maryland)

- To develop distributed computing models and architectures for logic programs to support distributed intelligent agent systems.
- A basis for:
  - supporting reasoning, planning, intelligent decision support, and intermediate between the user and the system levels
3 - Tools and Environments

- To observe application behaviour for resource management and for system/application dynamic reconfiguration;
- To support integrated testing and debugging;
- To support flexible infrastructures for tool integration

4 - Execution infrastructure

- To evaluate infrastructure support for the selected applications, namely concerning:
  - parallel i/o and file systems
  - support for group abstractions, for collaborative virtual shared spaces
Conclusions on Challenging Requirements

- Higher degrees of user interaction, increased flexibility in observation, control, or modification of application components.
- Multidisciplinary applications, interactions between distinct sub-models, and distributed user collaboration.
- Dynamic applications and environments, as new application components or system resources are dynamically generated, made unavailable, or mobile.
- Spatial distribution of application components and system resources, at small, medium or large scales.

Conclusions

- Main challenges:
  - New concepts
  - Models, tools and support environments
- They are driving significant research and development efforts that will have great impact upon many areas
Announcing a New Book

Call For Chapters

"GRID COMPUTING: SOFTWARE ENVIRONMENTS AND TOOLS"
http://www.cs.cf.ac.uk/User/O.F.Rana/grid-se/

Jose C. Cunha and Omer F. Rana (Eds)

To be published by Springer Verlag

Submission Deadline: November 15, 2003

Goals

- The uptake of Grid computing technologies will be restricted by the availability of suitable abstractions, methodologies, and tools.

- Our goal is to identify:
  - software engineering techniques for Grid environments,
  - along with specialist tools,
  - and case studies illustrating their use.

- Specialist software is necessary:
  - to enable the deployment of applications
  - to facilitate software developers in constructing software components for such infrastructure.
Annex 2

- An Introduction to GroupLog

A Coordination Model for Collective Agent Based Systems: GroupLog

Fernanda Barbosa and José C. Cunha
(fb.jcc)@di.fct.unl.pt
Dep. Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Portugal
Motivation
The main objective is

<table>
<thead>
<tr>
<th>structuring the interaction between computational entities</th>
</tr>
</thead>
</table>

* structure the agent space
* dynamic reconfiguration
* coordination of cooperating agents

A GroupLog system: a number of concurrently executing agents, able to:
* Communicate through interface predicates
* Join groups to participate in coordination activities
Agents and Groups

- Agents:
  - Computational entity
  - Internal configuration
  - Guarded communication
  - Remote predicate invocation
  - One-to-one, one-to-many

- Groups:
  - Dynamic organisation of agents
  - Direct and indirect communication between group members:
    - (one-to-one, one-to-many, shared group space)

L₁ - Dynamic structuring units of program entities (Agents)

Structuring communication and synchronisation

- Name
  - Its type
- Clause Context
  - Local knowledge
- Interface Context
  - A well defined interface

Agent

Agent Behaviour
Models actions when it interacts with other agents

Agent Creation
A clause models the initial actions and configuration
**Agent Behaviour**

Each interface is defined by a set of

- **Interface Rule**

- **Current configuration**: the configuration of the agent
- **Interface**: the signature of the interface predicate
- **Pre-Actions**: the actions that the agent needs to execute **before** change to new configuration
- **New Configuration**: the new agent configuration after executing the guarded actions
- **Post-Actions**: the actions that the agent needs to execute **after** change to new configuration

---

**Example: Stack**

Clients can communicate with stack agent through the interface predicates (push, top, pop)

```
Current configuration: stack(id,L)
Interface: push(X)
Guarded Actions: -
New Configuration: stack(id,[X:L])
Executed actions: -
```

```
Current configuration: stack(id,[X:L])
Interface: pop(X)
Guarded Actions: -
New Configuration: stack(id,L)
Executed actions: -
```
L₂ - Dynamic grouping of agents (Groups)
Structuring the set of agents and supporting their cooperation

- Name
- Clause Context
- Interface Context
- Group Behavior
- Group Creation

Group Shared State
A multiset of atoms shared by the group members

Group Membership
The agents and groups belonging to the group

Group Membership
The group is a composition of agents and groups, which changes dynamically

- An entity can enter into a group
- An entity can leave a group

The members of the group are hidden from the outside

The communication is through the group interface predicates

This isn’t allowed
Shared Group State

The members of the group may coordinate by accessing the shared group state.

The group table allows the coordination of the philosophers through the shared state (forks information).

These predicates are blocking.

Dyning Philosopher

Conclusion

The group notion is important to specify:

* a group of agents, with a shared knowledge
* the dynamic evolution of the system
* the coordination of the agents in a group