Abstractions for Distributed and Grid Computing Systems
José C. Cunha (jcc@di.fct.unl.pt)
CITI – Centre for Informatics and Information Technologies
Dep. Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

NeSC Workshop on
Distributed Programming Abstractions
Edinburgh, 30 May-1 June 2007
Contents

1. Characteristics of distributed applications
2. Need of new abstractions and models
3. Research issues for exploiting groups in distributed and grid computing environments
Layers of Abstractions

- Application-level Abstractions:
  - For each Application Domain / Classes of Applications
  - Problem-Solving Environments
- Programming Abstractions, Models, and Tools
- Intermediate Frameworks and APIs
  - To hide lower-level concerns
  - To promote incremental development
- Service and Resource Abstractions
Problem-solving perspective

Integrated environments for solving a class of related problems in an application domain
Problem-Solving Environments

• Approach:
  -- specific methods for each problem domain are encapsulated in components (libraries, packages, class OO repositories)
  -- development and runtime support tools are also made available.

• Application components and computational tools are integrated into a single environment (PSE)
  – Easy-to-use by the end-user
PSE Functionalities

- Abstractions for problem specification and for problem solving (relevant to the application domain)
- Resource management
- Execution support services
Application Characteristics and Requirements

- Complex models – simulations
- Large volumes of input / generated data
- Difficult interpretation and classification
- Flexibility in the 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
Parallel Computing

- Goal: to reduce execution time, compared to sequential execution.
- Parallel Programming requires abstractions for:
  - Decomposing application in parts – tasks, processes
  - Scheduling the tasks in parallel processors
  - Coordinating the cooperation between tasks
Distributed Computing

- Physically distributed computations and data
- Goals:
  - Adapt to geographical application distribution of Users / Access / Processing / Archiving Sites:
    - To enable communication
    - Access to remote services
  - Availability and Reliability
    - Fault tolerance / Redundancy
  - Try to provide appropriate levels of transparency
Abstractions

Depend on the layer:

- Failure
- Communication (message, RPC, memory)
- Coordination: events, uncertainty, causality, consistent cuts and snapshots
- Mobility

Abstractions allow different approaches, depending on the application profile:

- Pessimistic / Optimistic
- Strong / Weak guarantees
Grids and ´´Conventional´´ Distributed Systems?

What are the differences?

The distinctive aspects:

- Higher levels of the transparency for the end-user
- Higher levels of integration of services
- Virtualisation of resources

Zsolt Nemeth and Vaidy Sunderam:
User Abstraction
Resource Abstraction
A pool of virtual resources

application services
computation services
dataset, data repositories
information services
storage services
physical resources and devices
Question

- Real application needs?
  - Larger scale of resources for bigger, longer experiments: more accurate results
  - Easier access to remote resources
  - Increased levels of interaction with remote entities for increased productivity
  - Increased capability to analyse and react
Applications using Grids

- **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 Collaboration Spaces:**
  - Access to virtual environments for resource sharing, user interaction and collaboration
  - Access large geographically distributed data repositories, e.g. for data mining

- **Interaction and Sensor Grids (?)**
  - Real-time interactions for decision support

---

Information and Knowledge
More Complex Applications

- Large number of components
- Complex interactions
- Heterogeneity
- Dynamic characteristics
Dynamic characteristics

Looking at the past:
- Fault tolerance, Load balancing, Task spawning

At present and in future:
- Changes in the configuration and availability of resources, variations of characteristics and behaviour
- Changes at the application level: user control of experiments
- Flexibility to build PSEs
- Mobility of agents and devices
To meet new application challenges

- New problem-solving strategies with adaptive behaviour
- Awareness to Quality of Service factors
  - Management by the intermediate layers
  - By agents – planners
  - Contract negotiation
  - Dynamic revision of plans and reconfiguration
Research dimensions

- To provide
  - Abstractions and models for distributed application design and development
  - Associated tools and environments
  - Support infrastructures
- Ongoing developments in distributed - grid computing
  - Towards more robust, stable and standard infrastructures and support architectures
  - Setting up the basis to build higher level abstractions, models and environments
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
Programming Abstractions ?
Grid Programming Abstractions

Grid programs are distributed programs: they are composed of logical individual *nodes* or components which interact in a coordinated way:

*nodes* - encapsulate computation or data access
components performing individual parts of a whole application

*integration* and global coordination of the individual nodes and their interactions, as a distributed computation

(In the mid-90s, Metacomputing was a keyword)
Abstractions for:

- node composition and structure
- coordination: concurrency, parallelism, distribution; communication and interaction
- node aggregation and hierarchies
- meta-level information
- node code instantiation
- node registration, search and discovery
- node execution management and control
- specification of QoS
To conclude
Application Characteristics

- Large volumes of data, requiring
  - Efficient management and search
  - Parallel and distributed processing
- Integration of distributed, heterogeneous components in highly dynamic and interactive environments
- Dynamic, distributed, and mobile application entities, requiring appropriate management of
  - Structure, interaction, and coordination
- Dynamic organisation of small, medium, or large scale collections of distributed entities
Distributed and Grid Computing Systems

- Increasing levels of interaction among components
- New forms of dynamic behavior
  - Due to mobility
  - Due to more frequent changes in system and application configurations
  - Due to changes in interaction and behavior
- Increasing scale in terms of system and application components
Ongoing work

- Design Patterns
- Dynamic Groups
Design Patterns

- As a way to abstract commonly occurring structures and behaviors in distributed and grid applications
- And how they can be integrated in software development and execution support environments
- Allowing their manipulation during software development, e.g. to ease the building of PSEs
- And during application execution, to support coordination and autonomic behavior

- But sometimes new patterns of behavior emerge dynamically and need to be identified, and requiring decisions to be made dynamically...
Design Patterns for Grid Workflows

Omer Rana, Cecília Gomes, and José Cunha

- 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
Dynamic Groups: not a new idea
Long-term research on Process Groups: Kenneth Birman (Virtual Synchrony models) and many others.

- As organisation and cooperation paradigm to support
  - Scale, dynamism, and mobility, eg for local or ad-hoc communities of mobile entities, and for dynamic environments
- Appropriate forms of interaction and coordination in small, medium, or large scale organisations, possibly hierarchical. Exploit forms of shared knowledge, and information, and trust relationships among group members, and for specialisation of services and cooperation
- As units of system or application composition to help build and manage complex and dynamic organisations
distributed processes within a group

internal interaction via
messages or
via shared space

group interface
Benefits from using Groups

- 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:
  - collective communication
  - parallel / load balancing / fault tolerance
  - access to a shared logical state (or a way to have structured tuple spaces...)
Groups for structure and organisation

- Collections of agents which share common attributes
- Common logical characteristics shared by group members
  - Common computational or communication behaviors
  - Common goals in a society of agents
  - Need of sharing common resources and information
  - Cooperation towards providing common service functionalities with specific constraints
    - Performance
    - QoS
    - Cost parameters
Groups for scalability(2)

- By allowing hierarchies of entities where a group member can be an individual entity or another group
- Important in large-scale and complex organisations
- Allowing confinement of local and global policies
- And more flexible and efficient forms of communication and information dissemination
Groups for modelling dynamic systems

- By providing consistency of views among the group members
- By supporting forms of cooperation among group members, including a shared group state
- Or to manage components with common properties
- By allowing dynamic change of group membership
Groups as units of system composition

- Groups can appear at distinct abstraction levels:
  - At application level
  - At programming level
  - At system level

- Groups can be considered as programming units and used to build hierarchies
  - From its outside, a group can be viewed as an object, an agent, or a service, through an well-defined interface (like a set of methods, or ports), and with an internal behavior, hidden from the outside
  - Separation between the group interface and its internal behavior allows implementing local policies, internal to a group, in a transparent way
Groups as units of system composition

- A group can support a reactive or a pro-active, and goal-oriented behavior.
- It is possible to organise a distributed application or system in terms of collections of multiple groups, each responsible for a local service and policy, and globally managed by having global coordination and policies for overseeing and deciding on global strategies.
Groups as structuring units may include processes or other groups
Another perspective(1)

- Many distributed applications require the ability to capture and identify common attributes and their changes related to distributed and dynamic entities.

- The need to identify such attributes and their changes can become a critical concern, for example:
  - For dynamic strategies for resource management, depending on changing cost and resource usage.
  - To dynamically form ad-hoc groups:
    - As spontaneous identification of communities of interests (e.g., proximity between mobile users).
    - As dynamic definition of common interests, in reaching common goals, sharing common knowledge and functionalities, and contributing to common tasks.
Another perspective(2)

- The dynamic identification of groups as emerging from dynamically identified patterns of behavior or from the intention of pursuing common goals
- This can become a powerful mechanism to guide strategies for autonomic management of complex distributed systems and applications
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.