2nd Supercomputing and Distributed Computing Camp

Turrialba, Costa Rica, 10-16 de julio del 2011

Computación distribuida y sistemas multiagente

Prof. Dr. Alvaro de la Ossa
delaossa@cenat.ac.cr

Contents

- Motivation, terminology
- Distributed computing
- The problem
- Problem features and approaches
- Discussion

Motivation

- Use of parallel computing to model complex processes, structures or problems. E.g.,
  - Social cognition
  - Biological species interactions
  - Physical particles system behavior
  - Economy
  - ...

Motivation

- Complex problems and complexity
  - Problems with a large number of diverse, dynamic and interdependent elements
  - Complexity: use of space and time; model's effectiveness

Motivación

- Problema complejo
  - Fuzzy boundaries
  - Some parameters are unstable or unpredictible
  - Experimentation is difficult or expensive
  - A solution set is not known, only non-satisfactory approximations
  - Many stakeholders are involved, who have distinct viewpoints, interests and objectives
  - There is no unique, optimal solution for all
  - There is no well-defined halt condition

- Other problems
  - Well-defined boundaries; can be abstracted in families or universal types
  - Parameteres are stable or predictible
  - Experimentation is easy
  - There exists a well-defined solution set
  - Stakeholders participate that share a common viewpoint, interests and objectives on the problem
  - An optimal solution for all can be found
  - That solution is easily recognizable

Motivation

- Goal → Build a description, explanation or prediction of the behavior of a complex process, structure or problem
  - A computational model
  - based on distributed computing
  - that describes, explains or predicts the behavior of a set of individuals (objects or subjects) that are part of the process, structure or problem

Terminology

- Problem: process, structure, problem, etc. (whatever it is that you want to model)
  - Object — anything involved in the problem
  - Agent — a computational model of any (agentive) object involved in the problem
  - Environment — the set of objects that represent all the agentive and non-agentive objects in the problem (also domain, millieu or society)

Distributed computing

- Some advantages
  - Adaptability — changes in the individuals result in changes in the model
  - Redundance — the ability for continuous operation and recovery after failures in one or more elements (individuals) of the model
  - Specialization — agents dedicated to the resolution of specific parts or aspects of the problem
  - Efficiency — gain due to parallel processing
  - Economy — (of the individuals): don't need every agent to know all the relevant information

Distributed computing

- Some disadvantages
  - Reliability — it is not always possible to buold a system in such a way that it recovers from the failure of one or more agents;
  - Diagnóstico — the task of determining which agent or agents contribute with error to a solution is non-trivial
  - Comunicación — some problems might require intensive inter-agent communication; with large number of connections among agents, communication needs can eliminate or even turn into loss, the expected win in efficiency due to parallelization

Distributed computing

- Principles
  - Load distribution:
    - In N simpler machines
  - Massive processing:
    - N machines operating simultaneously to solve each one pieces of aspects of the problem
  - Interconnection or communication system:
    - The N machines share information through communication channels or protocols

Distributed computing

- Issues
  - Balanced load distribution
  - Control of the problem solving process: which agents, in which situations, should assume control?
  - Concurrency: access and use of shared resources
  - Organization or “social” structure: decision making regime of a group or society of agents

Distributed computing

- Multiagent systems as models of distributed computing

  Agents ↔ Processors, nodes

  Societies ↔ Networks

  Communication system ↔ Protocols

  Organization ↔ Architecture

The problem

- How to use multiagent systems to model complex problems?
  - Focus on the system complex problem or system dynamics, rather than on the objects involved
  - Model by agent type (roles, functions, responsibilities, actions, etc.)
  - Define a set of parameters to describe a possible situation or state of the problem, and define its initial value and how each of them changes due to agent action and agent-agent interaction

The problem

- The model should make inferences about
  - The causes of certain behavior of the system or its agents
  - The properties of the system or of the problem from agent properties and interactions
  - The individual differences
  - The interdependence relationships among agents

The problem statement

- Build a distributed system, based on the notion of agent, that simulates a complex process or problem
  - Design decisions:
    - Asign task types to agent types (who does what)
    - Define an information distribution scheme (who knows and has access to what)
    - Design the communication system (types of interactions, participants, message exchange, protocol interpretation)

Agents

- Agent

Ag := (Sit(Ag), Act(Ag), Dat(Ag)), where

- Sit(Ag) is a description of a situation or a problem for agent Ag
  - Basic representation: attribute-value vectors
- Act(Ag) is the set of actions that can be carried out by agent Ag in situation Sit(Ag)
  - Basic representation: forward-chaining rules, from situations to actions (and their effects)
- Dat(Ag) is the contents "owned" by agent Ag (its internal knowledge)

Agents

- Agent actions according to their scope

  Act(Ag) := Act_own(Ag) + Act_pub(Ag), where

  - Act_own(Ag) contains the actions of agent Ag that are invisible to others in the environment
    - Reasoning, decision making, ...
  - Act_pub(Ag) contains the actions of agent Ag that are visible to others, and thus are called “public”
    - Communication actions, actions on the environment, ...

Agents

- Agent actions according to their trigger

Act(Ag) := Demand(Ag) + Proactive(Ag), where

- Demand(Ag) := Sit(Ag) x Act(Ac)
  - Actions that take place whenever necessary
- Proactive(Ag) := Sit(Ag) x Dat(Ag) x Act(Ag)
  - Proactive actions, i.e., started at agent Ag's initiative

Agents

- The multiagent system's environment

Env := (A, C, M, S), donde

- A : the set of agents
- C : the context (partial representation of the world or domain of the multiagent system)
- M : the memory shared by all agents
- S := (S1, ..., Sn), where each Si is a family of subsets of A, and all families are organized in a tree structure (called the system's social organization or structure)

Agents

- Properties of the environment
  - Accesible — agents may or may not know the current state of the environment
  - Deterministic — each agent action has or has not a predictible effect; degree of certainty about that effect
  - Episodic — in an episodic environment, each agent's performance can be observed and assessed on a number of discrete episodes
  - Dynamic — the environment is characterized by being dynamic if its state changes with time and the agents' actions

Agents

- 2 kinds of agents
  - Human
  - Computational
- Desired properties
  - Autonomy
  - Efficient organization

What's in an agent?

- Identity
  - Name
  - Attribute descriptions
- Roles and responsibilities
  - Methods
  - Action rules
  - Interpretation rules

What's in an agent?

- Previous knowledge
  - Information about facts, object descriptions, concepts, beliefs
- Objective function
  - Preferences
  - Methods
  - Acceptance / rejection criteria for solutions

What's in an agent?

- Knowledge about others
  - Models about the identity, roles, responsibilities, previous knowledge, objective function, etc., of other agents
- Knowledge about the environment
  - Social organization rules
  - Communication / interaction protocols
  - Facts, norms, laws, regulations, etc.

Social Organization

- Social organization :=
  Social structure + Social action system
  - Social structure
    - Hierarchical: master-slave, contract networks, etc.
    - Horizontal: alliances, coalitions, etc.
  - Social action system
    - Social roles, affiliation
    - Communication and interpretation (BDI)
    - Interaction
    - Negotiation-in-context
    - Command and report, ...

Development Tools

- Ascape – framework and runtime environment, Java API, available as Eclipse plugin, SDK, stand-alone Jar, Applet Jar (http://ascape.sourceforge.net/)
- Cougaar – Java-based architecture, focused on scalability (http://www.cougaar.org/)
- JADE – Java Agent Development Framework (http://jade.tilab.com/)
- Repast – Java-based modeling and simulation platform; Repast Simphony 2.0 beta; Repast for HPC 1.0.1 beta, Dec. 2010 (http://repast.sourceforge.net/)
- NetLogo – MAS simulator, education-oriented (http://ccl.northwestern.edu/netlogo/)

Examples

- Ant lines – followers follow a leader by going directly to its position as perceived in each clock tic
- Fireflies – example of synchronization in distributed systems; fireflies perceive other fireflies' flashes and reset their own “cycles” to match those of the fireflies in their nearest vicinity
- Wolf-Sheep Predation – predator-prey ecosystem stability analysis
- Dining Philosophers – synchronization of concurrent processes (several independent processes coordinate the use of shared resources)

¡¡ Muchas gracias !!