A Multi-agent Approach for Semantic Resource Allocation

Jorge Ejarque, Raül Sirvent and Rosa M. Badia

Grid Computing and Clusters Group - Barcelona Supercomputing Center (BSC)
Artificial Intelligence Research Institute - Spanish National Research Council (IIIA-CSIC)

IEEE CloudCom 2010
December 3, 2010 Indianapolis
Outline

- Motivation
- Previous Work.
  - Resource Allocation Ontology
  - Centralized Semantic Scheduling
- Distributed Approach
  - Multi-agent distribution
- Implementation and Evaluation
- Conclusions and Future Work
Motivation

- SP has to allocate the user’s tasks in their available resources.
- Resources are heterogeneous and can be supplied by different providers.
- SP users specify requirements in different SLA terms which must be satisfied.
- Users and providers use their own terms. There is a need of semantic interoperability.
Motivation

- **SP Infrastructure can change**
  - New resources can be added or removed from the SP infrastructure

- **Resource can fail**
  - Performance Degradation

- **Resource Allocation must be adapted according to customer and provider preferences**

- **Agent technologies provide:**
  - Adaptation to infrastructure changes
  - Distributed resource allocation according to several customer and provider preferences
Centralized approach

- Stores semantic descriptions of system elements
- Manage the customer’s tasks requests
- Create executions environments, execute customers tasks
- Allocates resources on the different customer’s tasks (semantic descriptions)
- Centralized approach

1. Get Resource Candidates
2. Schedule Job
3. Get descriptions
4. Update scheduling
5. Execute Job

Client Manager

Resource Manager

Semantic Scheduler

Repositories

0. Resource description registration

Manage the customer’s tasks requests

Store semantic descriptions of system elements

Create executions environments, execute customers tasks

Allocate resources on the different customer’s tasks (semantic descriptions)
Resource Allocation Ontology

- Based on Grid Resource Ontology
- Extension to describe basic resource allocation concepts
  - Resources properties, Collections, Abstract task (resource requirements, time constraints) and Process (abstract task allocation), Actors, Providers, etc.
Rule-based semantic scheduling
Decentralized motivation

- Scalability
- Reduce single point of failure
- Customer and providers have different interests and they should be taken into account in the allocation process.
  - Different rules depending on the customers and providers or resource types
Distributed Architecture

- Distributed approach

  - Resource Allocation Rules
  - Resource Allocation Ontology
  - Semantic Metadata Repository

  - Agent Platform
    - Job Agent Container
      - Scheduler
      - Jobs
    - Res. Agent Container
      - Scheduler
      - Res. Man

  - Private Clouds
  - Clusters
  - Grids
  - Public Clouds (EC2,...)

- Distributed approach

  - Resource Allocation Rules
  - Resource Allocation Ontology
  - Semantic Metadata Repository

  - Agent Platform
    - Job Agent Container
      - Scheduler
      - Jobs
    - Res. Agent Container
      - Scheduler
      - Res. Man

  - Private Clouds
  - Clusters
  - Grids
  - Public Clouds (EC2,...)
System Agents

- **Distributed resource allocation**
  - Scheduling functionalities distributed across the system agents
  - Based on a negotiation between the system agents. (CNP)
  - Agents platforms can be distributed across multiple hosts

- **Two parts**
  - Client Management (Job Agents)
    - Management of jobs
    - Resource selection
  - Resource Provisioning (Resource Agents)
    - Management of resources
    - Resource allocation proposals

- **Belief-Desire-Intention agent modeling**
  - Define data, goals and plans for each agent
Job Agent

- Semantic annotation of tasks.
- Initiate the negotiation and select the best proposal for the customer interests.
- Adapt the task allocation to the system events. (Rescheduling)
- React to status changes (Fault tolerance)

<table>
<thead>
<tr>
<th>Belief (Status)</th>
<th>Goal</th>
<th>Plans</th>
</tr>
</thead>
</table>
| Requested       | Get Resources         | Annotate task description
                     || Negotiate allocation |
| Scheduled       | Find Better Allocation| Negotiate allocation                           |
| Running         | Monitor execution     | Check performance
                     || Update requirements
                     || Increase resources |
| Suspended       | Recover Suspended     | Evaluate work done
                     || Update requirements
                     || Negotiate allocation |
| Stopped         | Recover Stopped       | Negotiate allocation                           |
| Non Scheduled   | Recover Non Scheduled | Negotiate allocation                           |
| Cancelled       | Cancel Job            | Remove allocation                              |
Resource Agent

- Semantic annotation of resource descriptions
- Make proposal according to resource interests
- Create execution environments and execute tasks by means of a resource management interface
-React to resource events (Failures)

<table>
<thead>
<tr>
<th>Belief</th>
<th>Goal</th>
<th>Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduled Jobs</td>
<td>Monitor Scheduled Jobs</td>
<td>Check scheduled job</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perform execution</td>
</tr>
<tr>
<td>Running Jobs</td>
<td>Monitor Running Jobs</td>
<td>Check running jobs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notify status changes</td>
</tr>
<tr>
<td>Status Failed</td>
<td>Recover Failure</td>
<td>Try local rescheduling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notify affected jobs agents</td>
</tr>
<tr>
<td>Status Running</td>
<td>Register Resource</td>
<td>Annotate resource description</td>
</tr>
</tbody>
</table>
Resource allocation negotiation sequence

1. Select candidate resources
2. Send call for resource allocation to selected resource agents
3. Evaluate requirements, discard impossible allocations (resource full, license terms) and apply other provider’s rules
4. Send back proposed resource allocations
5. Apply customer selection rules
6. Accept selected proposal

SELECT DISTINCT ?id ?host WHERE {
  ?host tech:isProvidedBy ?rm .
  ?rm biz:actor_name ?id .
  ?memory rdf:type tech:MemoryCapacity .
  ?memory tech:hasValue ?mem_cap .
  FILTER (?mem_cap >= 1024**xsd:int) .
}
Implementation

- System agents: Jadex BDI Engine
- Semantic Capabilities: Jena 2 Framework
  - RDF, OWL APIs
  - Jena Rule Engine (Scheduler Module)
  - SPARQL queries
Evaluation

- Qualitative evaluation
  - BDI Agents facilitate coordination of complex tasks (job and resource management)
  - Resource Allocation Classes mapped to different schemas
    - EC2
      -Instances -> Host Class
      -Instance Storage, Memory, ECU -> Storage, Processing Resource
    - GLUE schema
      - ComputeElementType -> Host Class
      - CE properties -> Storage, Processing Resource properties
    - Ganglia schema
      - HostType -> Host Class
      - Ganglia metrics -> Resource properties
Evaluation

- Centralized vs Decentralized allocation time
Evaluation

- Centralized vs. Distributed
Evaluation

- Different agent deployment configuration
Conclusions

- Resource allocation framework which combines several technologies
  - Semantics for unifying data sources
  - Multi-agents for adapting to a changing environment
  - Rule scheduling promising approach
- Negotiation combines customer and providers preferences
  - Different rule examples
- Evaluation
  - Annotations from different schemas
  - Agents introduce a negotiation overhead
  - Trade-off between number of resources per agent and agents per machine.
Future Work

- Improve performance reducing overheads
- Exploit the benefits of rule-based resource allocation
  - Dynamic scheduling policies with rules
- Introduce semantics in the resource management interface interoperability
  - Not only in the resource description
Thank you for your attention.
Questions?

jorge.ejarque@bsc.es