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)

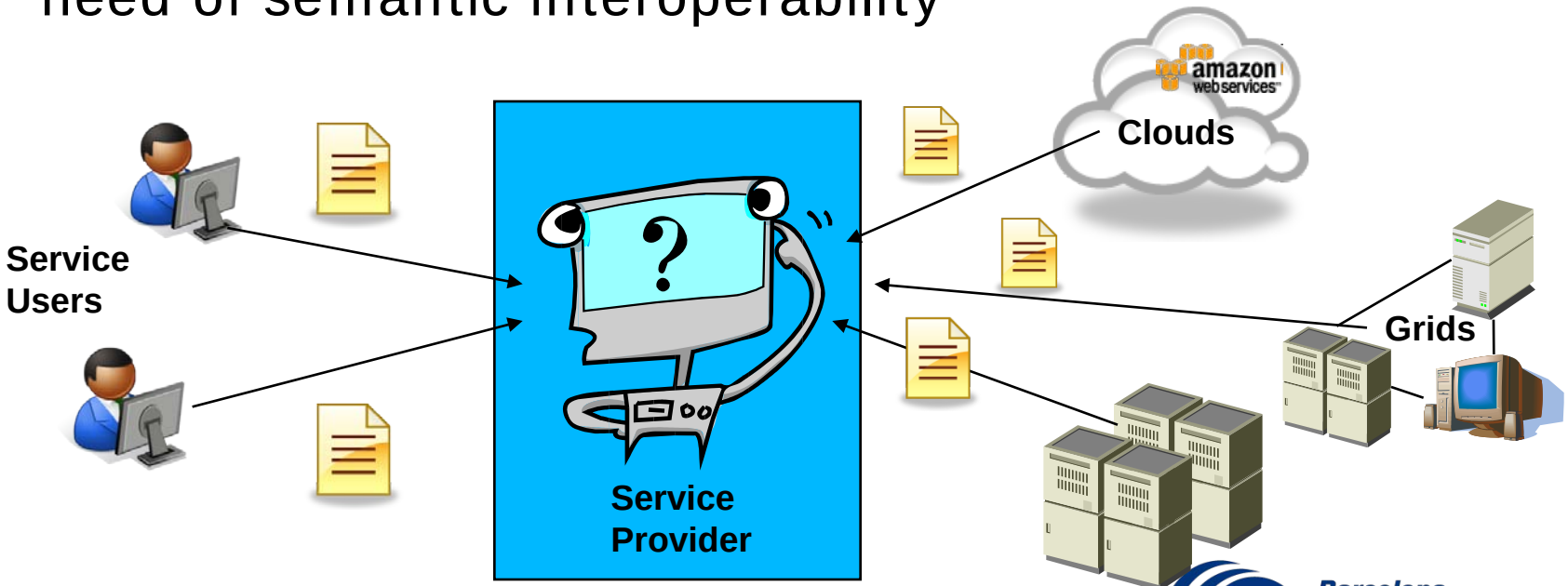
IEEE CloudCom 2010
December 3, 2010 Indianapolis

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

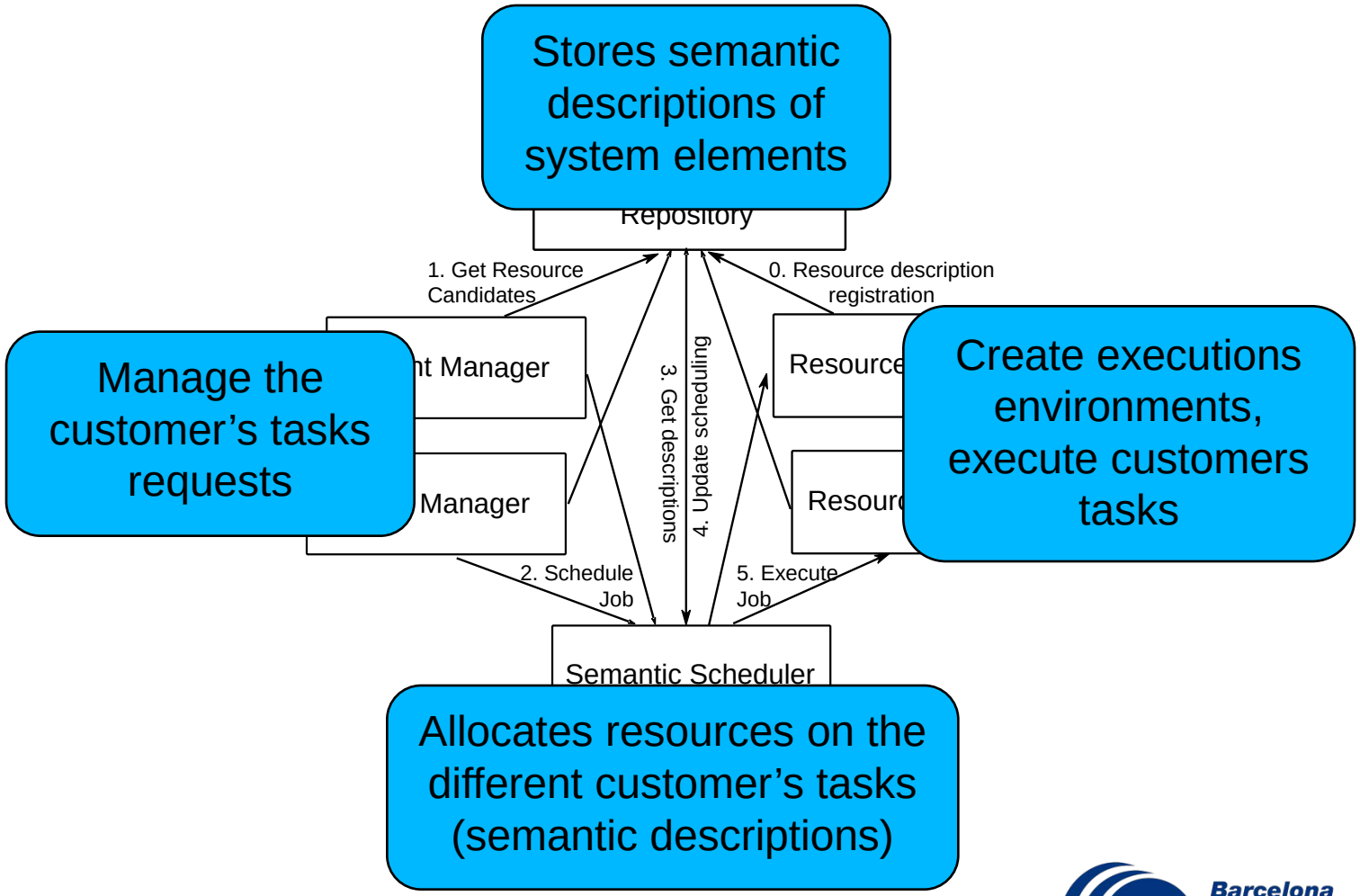


Motivation

- 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

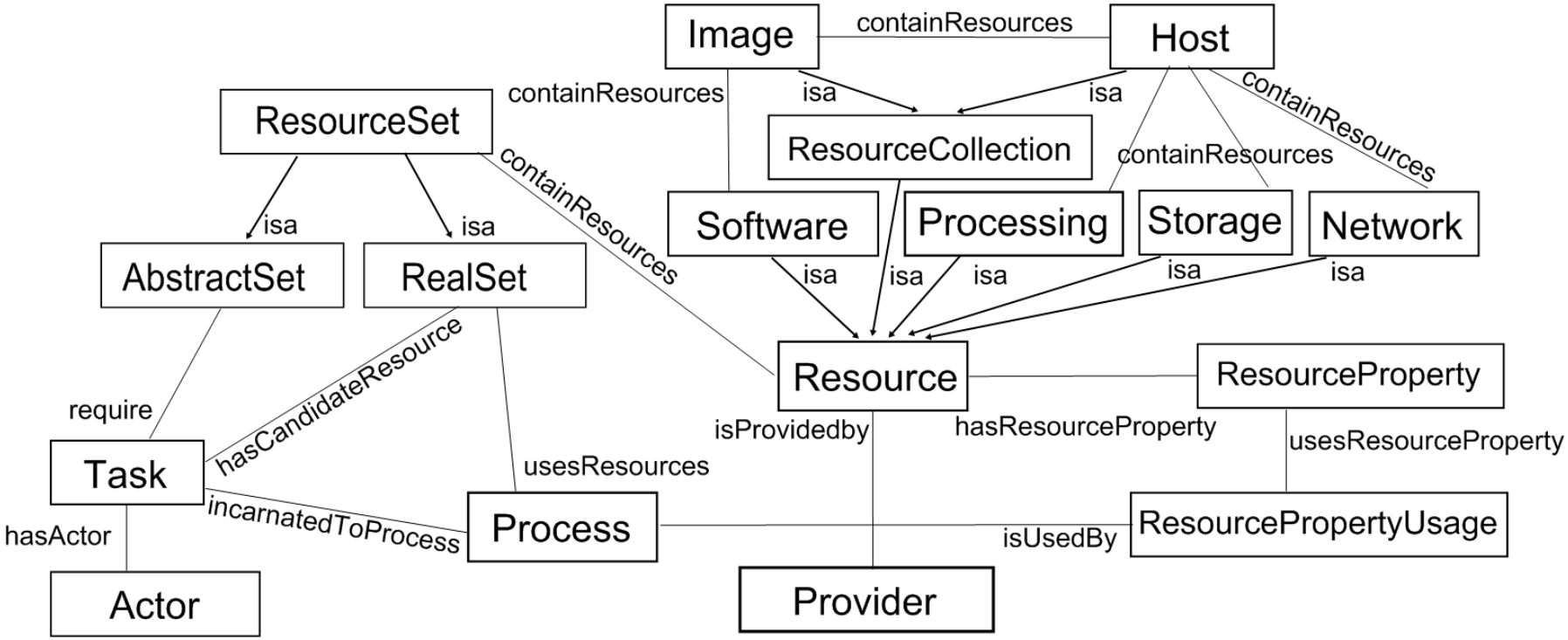
Previous work

- Centralized approach



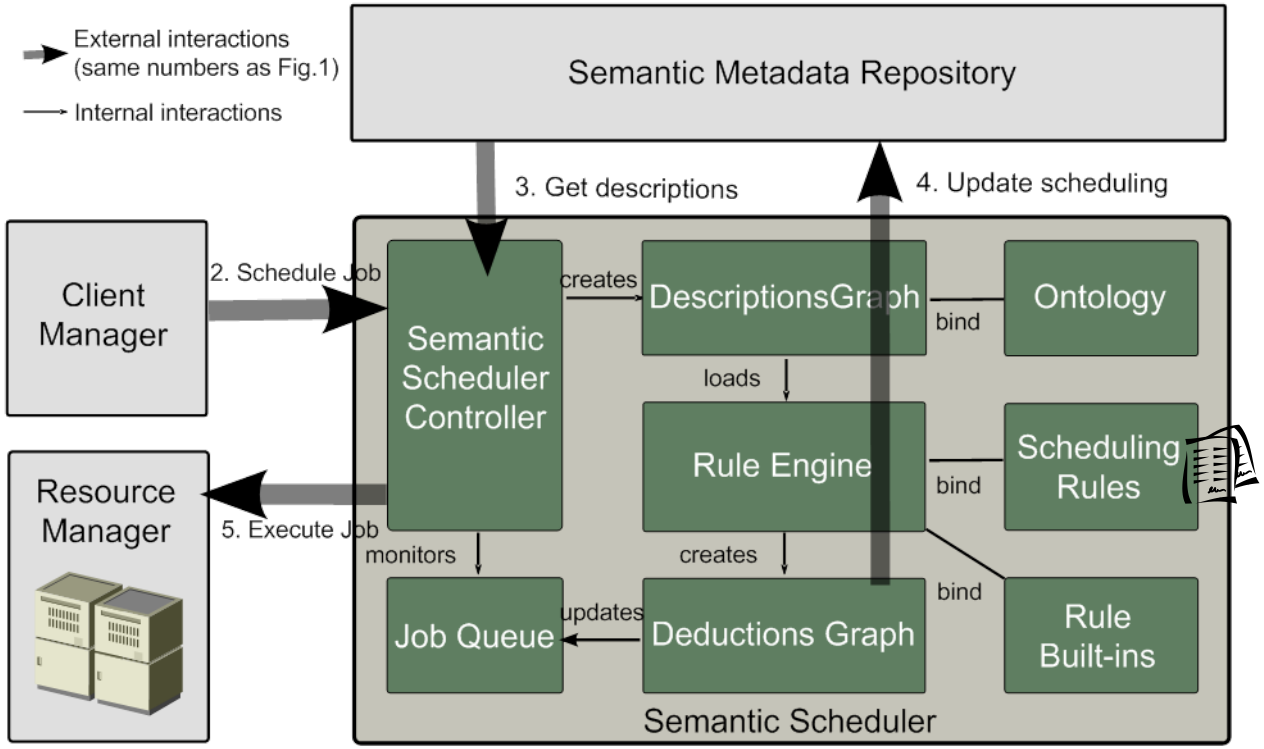
Resource Allocation Ontology

- 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.



Semantic Scheduler

- Rule-based semantic scheduling

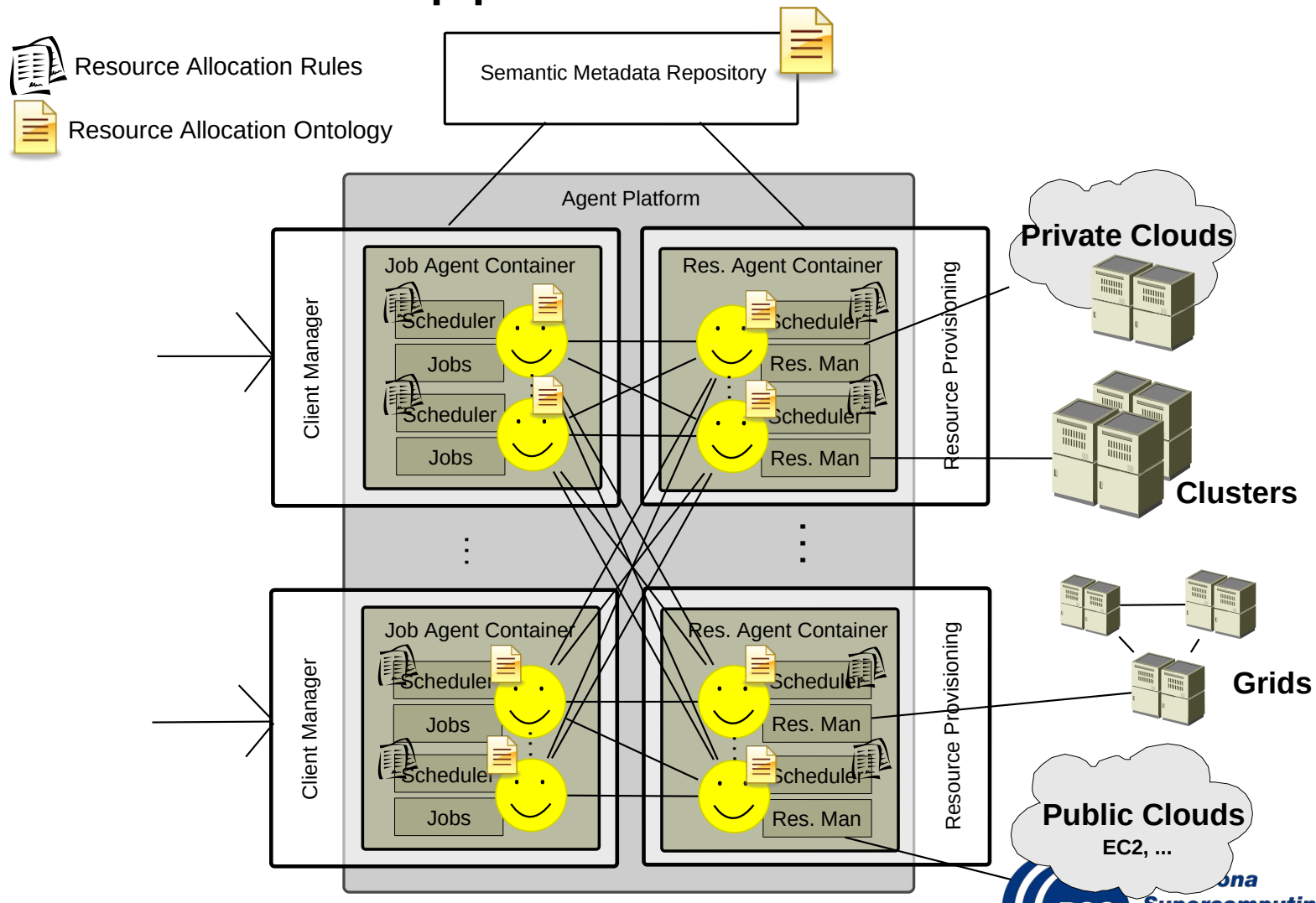


Decentralized motivation

- Scalability
- Reduce single point of failure
- Customer and providers has 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



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)

Belief (Status)	Goal	Plans
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)

Belief	Goal	Plans
Scheduled Jobs	Monitor Scheduled Jobs	Check scheduled job Perform execution
Running Jobs	Monitor Running Jobs	Check running jobs Notify status changes
Status Failed	Recover Failure	Try local rescheduling Notify affected jobs agents
Status Running	Register Resource	Annotate resource description

Distributed Resource Allocation

- Resource allocation negotiation sequence

```
SELECT DISTINCT ?id ?host WHERE {
  ?host tech:isProvidedBy ?rm .
  ?rm biz:actor_name ?id .
  ?host tech:containsResource ?proc .
  ?proc rdf:type gro:Processing .
  ?proc gro:hasResourceProperty ?memory .
  ?memory rdf:type tech:MemoryCapacity .
  ?memory tech:hasValue ?mem_cap .
  FILTER (?mem_cap >= 1024^^xsd:int) .
  ?host tech:containsResource ?image .
  ?image rdf:type tech:Image .
  ?image tech:hasResource gro:Software_XX }
```

```
[Earliest_Start_Date:
  (?job rdf:type gro:Execute_Task),
  (?job gro:hasActionState tech:requested),
  (?job gro:incarnatedToProcess ?incarnationA),
  (?incarnationA gro:hasPlannedStart ?stDateA),
  (?job gro:incarnatedToProcess ?incarnationB),
  (?incarnationB gro:hasPlannedStart ?stDateB),
  lessEqual(?stDateA, ?stDateB),
  ->
  remove(4,5)]

[Priority_AtoB:
  (?job rdf:type gro:Execute_Task),
  (?job gro:hasActionState tech:requested),
  (?job gro:incarnatedToProcess ?incarnationA),
  (?incarnationA gro:usesResource ?candidateA),
  (?candidateA gro:containsResources ?resourceA),
  (?resourceA tech:isProvidedBy biz:ResourceProviderA),
  (?job gro:incarnatedToProcess ?incarnationB),
  (?incarnationB gro:usesResource ?candidateB),
  (?candidateB gro:containsResources ?resourceB),
  (?resourceB tech:isProvidedBy biz:ResourceProviderB)
  ->
  remove(6,7)]
```

Resource Agent

```
[Evaluate_Deadline:
  (?job rdf:type gro:Execute_Task),
  (?job gro:hasActionState tech:requested),
  (?job tech:hasDeadline ?deadline),
  (?job gro:incarnatedToProcess ?incarnation),
  (?incarnation gro:hasPlannedEnd ?job_end),
  lessThan(?deadline, ?job_end),
  ->
  drop(3,4)]

[Used_Hosts_First:
  (?job rdf:type gro:Execute_Task),
  (?job gro:hasActionState tech:requested),
  (?job gro:incarnatedToProcess ?incarnationA),
  (?incarnationA gro:usesResource ?candidateA),
  (?candidateA gro:containsResources ?HostA),
  hasValue(?HostA gro:isUsedBy),
  (?job gro:incarnatedToProcess ?incarnationB),
  (?incarnationB gro:usesResource ?candidateB),
  (?candidateB gro:containsResources ?HostB),
  noValue(?HostB, gro:isUsedBy)
  ->
  drop(6,7)]
```

Implementation

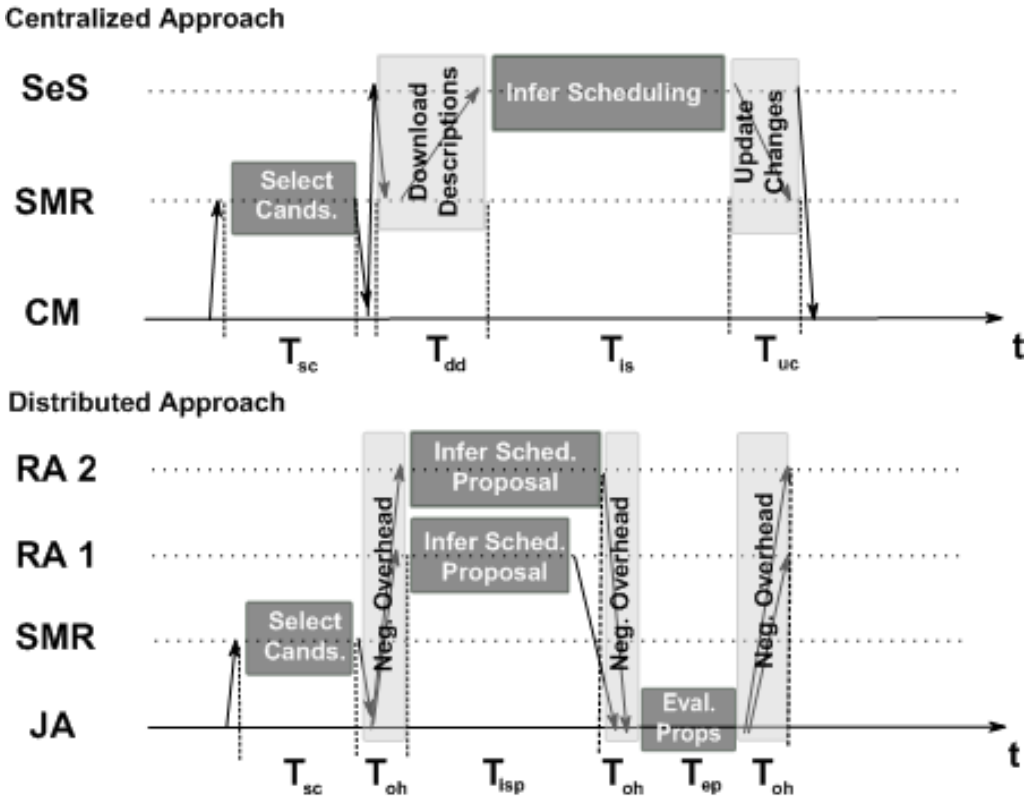
- System agents: Jadex BDI Engine
- Semantic Capabilities: Jena 2 Framework
  - RDF, OWL APIs
  - Jena Rule Engine (Scheduler Module)
  - SPARQL queries

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

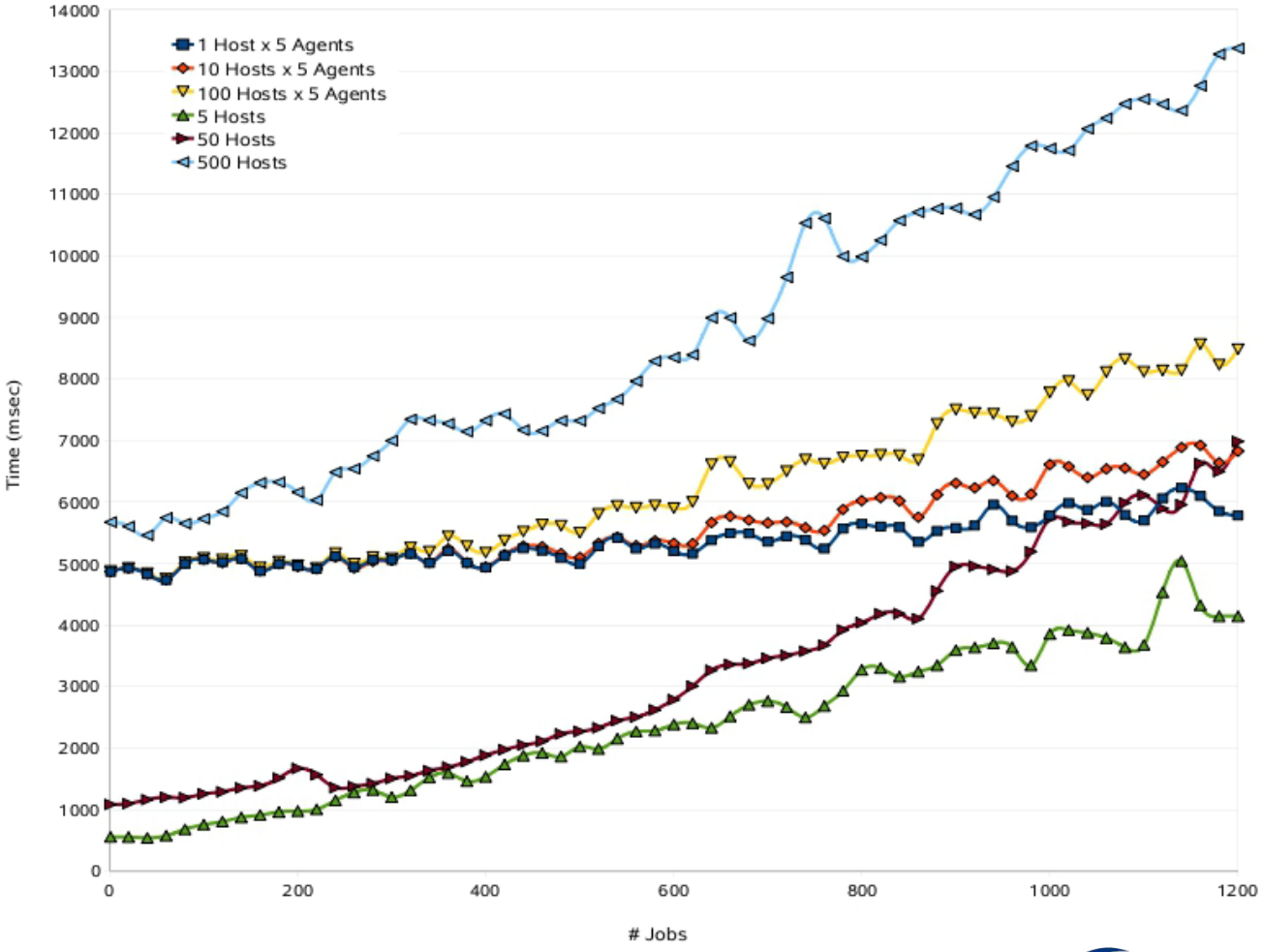
Evaluation

- Centralized vs Decentralized allocation time



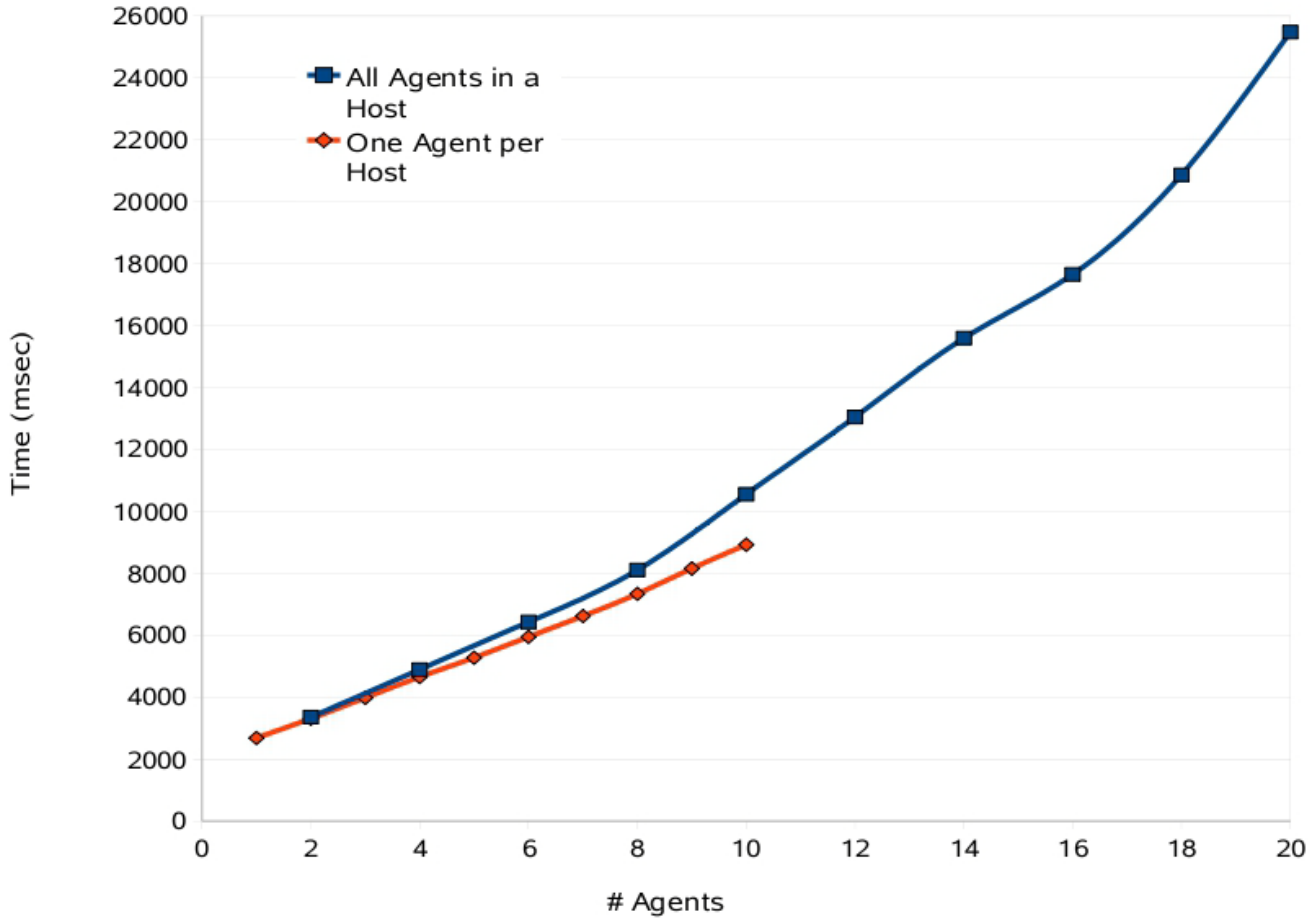
Evaluation

- 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

Barcelona Supercomputing Center
Centro Nacional de Supercomputación