Adaptive Task Scheduling in Grid Computing Environments

Angelos Michalas\textsuperscript{1}, Malamati Louta\textsuperscript{2}

\textsuperscript{1}Technological Educational Institute of Western Macedonia
\textsuperscript{2}Harokopio University of Athens
What is the Grid?

- The **World Wide Web** provides seamless access to information that is stored in many millions of different geographical locations.

- The **Grid** is an emerging infrastructure that provides seamless access to computing power and data storage capacity distributed over the globe.
Open Grid Services Architecture (OGSA)

- OGSA has enhanced Web Services properties by defining the concept of Grid Services
- Architectural features of Grid Services:
  - service factories
  - Registries
  - naming and referencing standards for service instances
  - stateful services
  - event notification mechanisms and versioning support.
Paper Contribution

- To propose an adaptive task scheduling mechanism based on an Ant Colony Optimization (ACO) algorithm.
- To present a Grid Services Architecture (GSA) supporting the aforementioned mechanism.
- A set of grid services are defined conforming to the OGSA standards.
- Semantically enhanced so as to provide a common semantic understanding between the components involved in the scheduling process.
Task Assignment Framework

➢ GIVEN

✓ the set of candidate Computing Resources and their layout,
✓ the set of tasks constituting the required services
✓ the resource requirement of each task in terms of CPU utilization
✓ the current load conditions of each computing resource and of the network links

➢ FIND

✓ find the best assignment pattern of tasks to computing resources subject to the capabilities of the computing resources

➢ Our approach uses an Ant Colony Optimization algorithm (ACO) for task allocation.

➢ ACO algorithms are based in a behavioral pattern exhibited by ants and more specifically their ability to find shortest paths using pheromone, a chemical substance that ants can deposit and smell across paths.

➢ ACO is used to solve many NP-hard problems including routing, assignment, and scheduling.
Mapping between the ant system and the grid system.
BACKGROUND (1)

- (Yan, 2005) uses the basic idea of MMAS ACO. The pheromone deposited on a trail includes:
  - an encouragement coefficient when a task is completed successfully and the resource is released,
  - a punishment coefficient when a job failed and returned from the resource
  - a load balancing factor related to the job finishing rate on a specific resource.

- (Chang, 2009) uses a balanced ACO which performs job scheduling according to resources status in grid environment and the size of a given job.
  - Local pheromone update function updates the status of a selected path after job assignment.
  - Global pheromone update function updates the status of all existing paths after the completion of a job.
(Dornermann, 2007) presents a metascheduler which decides where to execute a job in a Grid environment consisting of several administration domains controlled by different local schedulers.

- AntNests offer services to users based on the work of autonomous agents called Ants.
- A grid node hosts one running AntNest which receives, schedules and processes Ants as well as sends Ants to neighboring AntNests.
- State information carried by Ants is used to update pheromones on paths along AntNests.
Scheduling Architecture (OGSA)

- **Scheduling Service (SS):** selecting on behalf of the Grid resource provider the best task assignment pattern.
- **Job Submit Service (JSS):** promoting the service request to the appropriate SS.
- **The Resource Agent (RA):** promoting the current load conditions of a candidate computing resource (CCR).
- **The Network Provider Agent (NPA):** providing current network load conditions (i.e., bandwidth availability) to the appropriate SS.
- **A semantic description using Web Ontology Language (OWL) for resources and tasks is adopted, so as to provide a common semantic understanding to be shared between the components involved in the scheduling process.**
OWL for tasks

```xml
<owl:Class rdf:ID="TaskRequest">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasJobType"/>
      <owl:someValuesFrom rdf:resource="#jobTypes"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#needsCPUs"/>
      <owl:someValuesFrom rdf:resource="#CPUSList"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#inputFiles"/>
      <owl:someValuesFrom rdf:resource="#inputSandbox"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#outputFiles"/>
      <owl:someValuesFrom rdf:resource="#outputSandbox"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#constraints"/>
      <owl:someValuesFrom rdf:resource="#constraintList"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```
Scheduling Architecture (OGSA)
The ACO Algorithm

- The initial pheromone value of each CCR is given by the formula:

\[ \tau_j(0) = CPU\_Speed_j \cdot (1 - CPU\_Load_j) \]  \hspace{1cm} (1)

- Pheromone trails are updated upon assignment of a task on a CCR and upon termination of a task according to the formula:

\[ \tau_j^{post} = \rho \cdot \tau_j^{pre} + \Delta \tau_{ij} \]  \hspace{1cm} (2)

  - When task \( i \) is assigned to CCR \( j \), \( \Delta \tau_{ij} = -M \), while when task \( i \) is completed and CCR \( j \) is released \( \Delta \tau_{ij} = M \).

  - \( M \) is a positive value relevant to the computation workload of the task.

- The desirability of assigning task \( i \) to CCR \( j \) is defined by the following formula:

\[ des_{i,j} = \tau_j - Com\_Cost_{i,j} \]  \hspace{1cm} (4)

  - The factor \( Com\_Cost_{i,j} \) is the cost of migration to CCR \( j \).
Task Scheduling Modules

- Simulated grid environment composed of six computing resources reside on a 100Mbit/sec LAN, running the Linux Redhat OS.

- The overall Task Scheduling Mechanism has been implemented in Java.

- Voyager mobile agent platform used for the realisation of the software components as well as for the inter-component communication.

- JSS, SS and monitoring modules RAs, NPAs implemented as fixed agents

- The task is implemented as intelligent mobile agent, which can migrate and execute to remote computing resources.
Experimental Results(1)

- To evaluate the efficiency of our service task allocation method the following experimental procedure has been followed which is similar to (Chang, 2009).

- We consider 1500 simple tasks each performing matrix multiplication of real numbers. The matrix sizes are varying from 400x400 up to 1000x1000.

- The task size depends on its matrix size and is about $n \times n \times 4$ bytes (each real number is represented by 4 bytes).

- The number of instructions that the task contains, can be drawn from task’s complexity.

- Since matrix multiplication has $O(n^3)$ complexity, $2n^3$ instructions are estimated for a $n \times n$ matrix multiplication.

- Communication cost is similar for all hosts only the computation workload of tasks is considered.
Experimental Results (2)

- The same experiments have been conducted with and without using the ACO service task allocation scheme.

- In the latter case, service tasks are assigned in a round robin fashion to service nodes.

- In order to measure the efficiency of both methods we use the standard deviation of CPU load of CGNs.

- The load of each CGN is sampled after each task assignment and the standard deviation of each method is computed per 100 samples from 100 to 1500 tasks.

- The standard deviation is computed as:

\[
\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2}
\]

- Where \( \sigma \) is the standard deviation, \( x_i \) is the CPU load of resource \( i \) and \( \bar{x} \) is the average load of all resources.
Experimental Results(3)

- Decrease in the standard deviation $\Leftrightarrow$ the load of CCRs is better balanced.

- In case more tasks exist ACO performs better than the Least Loaded and Load Minimum (that need state information e.g., CPU load) as well as the Random and the Round Robin.

![Standard Deviation of Load](chart.png)
Conclusions

• In this study the task assignment problem in Grid computing environments has been addressed using an Ant Colony Optimization algorithm (ACO).

• Our objective was to design a scheduling architecture conforming to the OGSA standards, for Grid computing environments.

• Future work includes, to experiment with the applicability of the framework presented herewith in case of tasks requiring interactions with other resources so as to accomplish their goals.
THANK YOU