Resource Management on Computational Grids

Paolo Palmerini

Dottorato di ricerca di Informatica
(anno I, ciclo II)
email: palmeri@dsi.unive.it
Outline

• Introduction to Grid Technology
  ✩ What is a “Grid”
  ✩ Tools and projects

• The Globus toolkit Box
  ✩ Globus services

• Resource Information Infrastructure
  ✩ The globus MDS
  ✩ GRIS & GIIS

• A Grid Resource Broker
  ✩ State of the Art
  ✩ Architecture of a broker
General Motivation

- HPC applications are becoming complex and multidisciplinary
  - High Energy Physics
  - Aircraft design
- Resources are inherently distributed
- Need for new models and approaches to HPC, distributed computing, resource management.
What is a “Grid”?  


- Starting from the analogy with electric power grid
- Gathering diverse and large scale distributed resources in a unique and coherent view
- Based on the concept of Virtual Organizations
Grid Architecture

- Fabric layer
  - Computational resources
  - Storage resources
  - Network resources
  - Code repositories
  - Catalogs

- Connectivity layer
  - Single sign on
  - Delegation
  - Integration with local services
Grid Architecture (cont’d)

• Resource layer
  - Information Protocols
  - Management protocols

• Collective layer
  - Directory Service
  - Scheduling, co-allocation
  - Monitoring and diagnosing
  - Grid-enabled programming systems
  - Workload management systems
  - Community authorization servers
  - Commodity accounting
Grid Middleware

- Globus
  - http://www.globus.org

- Legion
  - http://www.legion.org

- Condor
  - http://www.cs.wisc.edu/condor

- Harness
  - http://www.epm.ornl.gov/harness
The Globus Toolkit

A software toolkit addressing key technical problems in the development of Grid enabled tools, services, and applications

- Offer a modular “bag of technologies”
- Enable incremental development of grid-enabled tools and applications
- Implement standard Grid protocols and APIs
- Make available under liberal open source license
Basic Globus Services

- Security
- Communication
- Fault Detection
- Information Infrastructure
- Resource Management
- Portability
- Data Management
Globus Layered Architecture
Globus Resource Management
Resource Specification Language

Example: 3 stages pipe

+ ( &\texttt{(resourceManagerContact=\texttt{prosecco.cnuce.cnr.it})} \\
  (count= 1) \\
  (label=\texttt{subjob 0}) \\
  (environment=(\texttt{GLOBUS_DUROC_SUBJOB_INDEX 0})) \\
  (arguments= "pipe.conf" "128" "128" "20" "10") \\
  (directory=\texttt{/home/prosecco/palmeri/ASI-PQE/MPI/src}) \\
  (executable=\texttt{/home/prosecco/palmeri/ASI-PQE/MPI/src/pipe-g}) \\
) \\
( &\texttt{(resourceManagerContact=\texttt{barbera.cnuce.cnr.it})} \\
  (count= 4) \\
  (label=\texttt{subjob 1}) \\
  (environment=(\texttt{GLOBUS_DUROC_SUBJOB_INDEX 1})) \\
  (arguments= "pipe.conf" "128" "128" "20" "10") \\
  (directory=\texttt{/home/prosecco/palmeri/ASI-PQE/MPI/src}) \\
  (executable=\texttt{/home/prosecco/palmeri/ASI-PQE/MPI/src/pipe-g}) \\
) \\
( &\texttt{(resourceManagerContact=\texttt{prosecco.cnuce.cnr.it})} \\
  (count= 1) \\
  (label=\texttt{subjob 5}) \\
  (environment=(\texttt{GLOBUS_DUROC_SUBJOB_INDEX 2})) \\
  (arguments= "pipe.conf" "128" "128" "20" "10") \\
  (directory=\texttt{/home/prosecco/palmeri/ASI-PQE/MPI/src}) \\
  (executable=\texttt{/home/prosecco/palmeri/ASI-PQE/MPI/src/pipe-g}) \\
)
First problem: which RSL level?

*ground* RSL is too complex:

```rsl
((resourceManagerContact="barbera")
  (count= 4)
  (label="subjob 1")
  (environment=(GLOBUS_DUROC_SUBJOB_INDEX 1))
  (arguments= "pipe.conf" "128" "10")
  (directory=".")
  (executable="/pipe-g")
)
```

High level request

run pipe-g on 6 nodes with 128 MBytes memory each

We need an *entity* that translate high level requests to *ground* RSL.
Metacomputing Directory Services

- Basic elements are Virtual Organizations (VO)
- Scalability achieved through a distributed model
- Soft-state registration mechanism
MDS Implementation

- Grid Information Service (GRIS)
  - Provides resource description
  - Modular content gateway
- Grid Index Information Service (GIIS)
  - Provides aggregate directory
  - Hierarchical groups of resources
- Lightweight Dir. Access Protocol (LDAP)
  - Standard with many client implementations
  - Used for resource inquiry and registration
Lightway Directory Access Protocol

- Structural information
  - Resource hierarchy maps to objects
  - Named positions in LDAP DIT
- Merged information
  - Some parents join child data
  - Simplifies common query patterns
- Auxiliary information
  - Uniform representation of leaf/parent data
  - Uses LDAP auxiliary objectclasses
Second Problem: resource discovery

- Which resources do we need?
- Where do we find them?
- How do we use them?
Second Problem: resource discovery

- Which resources do we need?
- Where do we find them?
- How do we use them?

The entity should:

- Understands our needs
- Looks for the best resources available
- Executes the application on them
Nimrod/G: a Grid Resource Broker

- A specialized system for the execution of parametric jobs on distributed resources.
- Uses a simple language to express the experiment and provides machinery that automates the task of formulating, running and monitoring the jobs, and collecting the results.
- Uses Globus Services
- Four phases scheduling algorithm
  - Discovery. Based on MDS
  - Allocation. Evaluate a cost foreach job.
  - Monitoring
  - Refinement
Limitations of Nimrod/G

- Narrow application domain. Parametric problems are only a part of Grid Applications.
- No communications among components. Network influence is not properly considered.
- Cost model assumes constant job behavior. In many cases the cost of a job cannot be known in advance (e.g. datamining).
Sun Grid Engine Software

- Revenue from grid computing solutions is expected to grow from $1.5 billion (2001) → $4 billion 2003;
- A set of tools for managing distributed resources;
- Aimed at optimizing resource usage within an organization;
- Based on a centralized queue where jobs are submitted and scheduled to the most adequate resource;
- Jobs are described in terms of resource requirements, (hardware and software licenses) and priorities;
- It’s an open source project!

http://www.sun.com/gridware
http://www.gridengine.sunsource.net
Resource Management of Structured Applications

- We want to manage the configuration and execution of structured parallel applications on computational grids.
  - Describe the application and its requirement
  - Discover the resources that match the requirements
  - Schedule the components onto the discovered nodes (co-allocation)

- Important issues
  - heterogeneous architecture
  - executable staging
  - maintain up-to-date information on grid status
Describing Structured Applications

- The application can be described by means of a configuration file, built in a semi-automatic way.
  - static info: application structure, component names and implementations
  - dynamic info: parallelism degree, mapping of components onto machines.

- Coded in XML, but Conceptually equivalent to ground RSL.

```xml
<global_config>
  <structure/>
  <granularity/>
  <mapping/>
</global_config>
```
Configuration file

Components of the XML file

- **structure**
  Specify the general structure of the application, by means of a generic graph. Component names and mutual *dataflow* dependencies. Also the name and location of the modules (i.e. files) that implements the components.

- **granularity**
  The parallelism degree and the replication of each component. For example the number of workers in a farm, or the number of processes for a parallel module.

- **mapping**
  The physical machines the components will run on.
Resource Discovery

Varius approaches are possible:

• **Sol. 1.** Query a statically known list of other GRIS or GIIS.
  ✧ Many queries during configuration.

• **Sol. 2.** Build an ad-hoc *root* GRIS that covers a statically known list of other GRIS or GIIS.
  ✧ Queries are resolved locally.
  ✧ Use the globus mechanism for updating information.

• **Sol. 3.** Search engine for grid resources
  ✧ Off-line spidering of all the GRIS and indexing them
  ✧ On-line query engine that finds resources that match user queries.
Scheduling

• Once the application is configured (in its granularity and mapping), we have the low level RSL specification and the application could be started.
Scheduling

- Once the application is configured (in its granularity and mapping), we have the low level RSL specification and the application could be started.
- But the grid is a highly heterogeneous environment, whose status rapidly changes during time.
Scheduling

• Once the application is configured (in its granularity and mapping), we have the low level RSL specification and the application could be started.

• But the grid is a highly heterogeneous environment, whose status rapidly changes during time.

• We can introduce a centralized scheduler that is in charge of managing the queue of requests for the computational grid.
Scheduling

The Grid Scheduler should:

- Co-schedule application’s component, when possible.
- Optimize resource utilization (i.e. overlapping communication and computation).
- Apply application specific (hence more accurate) cost models.
- Implement trading and negotiation for resource contention.
A Grid Resource Broker for Structured Parallel Applications
References


