A Grid for process control

Fabrice Sabatier, Supélec, Fabrice.Sabatier@metz.supelec.fr
Amelia De Vivo, Universita di Salerno, amedev@unisa.it
Stéphane Vialle, Supélec, Stephane.Vialle@supelec.fr

Long term goals / Why to use a Grid for process control ?

• To connect a physical process to “computing power” like to electrical power !

• Physical processes are installed where they are needed:
  • can be far of computing centers,
  • can be in computer hostile environment,
  • can be far of computer maintenance people,
  • ….

• Embedded large computing power can be:
  • too much power consuming
  • too much expensive
  • too constraining for the physical process mission
Long term goals / Why to use a Grid for process control?

- Access to:
  - large computing power
  - redundant computing (for fault tolerance)
  - remote control and maintenance
  - up to date process control libraries (grid services)
  - unlimited history saving mechanisms

Small embedded computing units & connection to the Grid

Project Road Map

“Step by Step” integrated project
Incremental development and deployment with frequent performance measurements
Approximate road map

Phase 1 – 2002-2003:
- P2P connection France-Italy across “ssh-link”
- Experiment remote control across Internet (robot server + client applications)
- Performance measurements
- Optimization of the robot control algorithms (serial optimization, multitreading, hyperthreading, MPI, computation-communication-mechanical move overlapping)

Phase 2 – 2003-2004:
- Deployment of a light Grid environment across Internet (Internet/VPN/Corba/GridRPC)
- High-level services implement complex robot commands
- Low-level services support redundant and concurrent calls
- User friendly API development
- Grid service semantic definition (beginning)
- Performance measurements
- Fault tolerance experiment and achievement

Grid soft. architecture
- Application
- RobGrid API
- DIET-GridRPC
- CORBA
- VPN-IPSEC
- Int/Ethernet
Approximate road map

Phase 3 – 2004-2005:

• Extension of the Grid (still VPN based):
  – more sites, with different “internet distances”
  – several physical processes to control
• Redundancy management policy &
  Redundancy manager Grid services
• Improvement of socket comm.: TCP → UDT (?)
• Performance measurements

Join us!

Approximate road map

Phase 4 – 2004-?:

• Deployment of a **Globus based** Grid environment
• Grid service portage: VPN/Corba/GridRPC → “Globus/XXX”
• API improvement: RobGrid API → ProCtrlGrid API
• Monitoring and accounting
• Performance measurements

Electrical power
(electrical grid)

Computing power
(process control grid)
Details on phase 2

2003-2004
Using DIET on a VPN
Real deployment across France and Italy

Phase 2
Short term goals

- To support special applications needing extra CPU
- To efficiently process embarrassingly parallel applications
- To dynamically switch to unloaded machines, avoiding to devote machines
- To be fault tolerant
- To share our robotic system with our (distant) partners
**Phase 2**

**Robot & Grid testbed**

- Robotic environment
- Grid of computing resources
  - multiprocessors
  - DHCP
  - DNS
  - LDAP
  - firewall
  - gateway
- Eth. Gigabit
- router
- LAN
- Internet
- France
- Italy
- Grid middleware
  - Application
  - RobGrid API
  - DIET-GridRPC
  - CORBA
  - VPN-IPSEC
  - Int/Ethernet

**Phase 2**

**Software Grid Architecture**

- High level services
- Low level services
- Robot server
  - Buffer ctrl
  - Serial link
  - TCP
- Robotic applications on the Grid
- High level robot commands
  - Low-level robot commands
    - Buffer control
    - TCP sockets
  - DIET API (GRidRPC)
  - DIET middleware (based on Corba)
  - VPN (IPSEC)
  - Int/Ethernet
  - Grid middleware services
Phase 2
Secure VPN

IPSEC based

Needs:
- Port UDP-500 to be opened
- Protocols ESP (50) and AH (51) to be authorized
- Firewall: to reject msg from PCs without VPN certificate
- Gateway:
  - to establish authenticated connections
  - to encapsulate TCP msg in ESP msg

---

Phase 2
Grid deployment & Chain of services

1. VPN-Corba-DIET
2. VPN-Corba-DIET
3. VPN-TCP
4. VPN-TCP
Phase 2
High Level Grid Interface: RobGrid

RobGrid main features:
- C++ library, based on GridRPC
- Client objects for easy access to high level Grid services
- Manages redundant calls to high level Grid services
- Hides communication initializations with any service

High level Grid services
(localization, navigation, lightness, …)

High level DIET interface
(Session, LocClient, NavClient, …)

Low level robot services
DIET & Grid middleware services

Programming new high level Grid services:
Need a high-level Grid service for robot control?
Implement quickly a new one (calling RobGrid internal objects)

One high level Grid service = a set 4 of sub-services:
- Connection to the related service of the robot server
- Reset of the result buffers on the robot server
- Robotic operation (ex: navigation, localization, …)
- Disconnection from the robot server
Adding a new high level Grid service for robot control:

```
loc->Call();
Res = loc->GetResult();

Nav->AsyncCall(x,y,theta);
While(!nav->Probe()) {
    light->Call();
    ...
}
loc->Call();
Res = loc->GetResult();
```

“Lightness measurement”
Service has been:
- quickly developed
- quickly included in the Grid

Application code example:

```
Session *session = new Session();
NavClient *nav = new NavClient(2);
LightClient *light = new LightClient(1);
LocClient *loc = new LocClient(2);

Session->Start();
loc->Connect();
nav->Connect();
light->Connect();
nav->AsyncCall(x,y,theta);
while(!nav->Probe()) {
    light->Call();
    ...
}
loc->Call();
Res = loc->GetResult();
```

delete loc;
delete nav;
delete light;
delete session;
Phase 2

Performance measurement

Benchmark of localization routine on the Supelec sub-Grid:

- Frequently called (strongly optimized)

- **Local Grid performances:**
  - no sensible overhead:
  - local redundant computation
  - hide variations:

  ![Graph showing performance comparison](chart.png)

  - Unloaded computing server alone
  - Unloaded computing server across the sub Grid
  - Sub-Grid with not unloaded servers

---

Phase 2

Performance measurement

Benchmark on 24h for localization operation across Internet:

![Graph showing execution time](chart.png)

**20h-9h:**
- localization across Internet is OK
- slow down < 2
- regular execution time

Usable for redundant computating to achieve fault tolerance …
Phase 2
Fault tolerance experiment

Running the complete application:
« Localization + navigation + lightness measurement »

The faster localization service stops
The redundant localization service drives the camera
The faster localization service re-start

• Application don’t stop, and go on.
• Slow down is limited to the parts using a slower service.
?
Fault tolerance is achieved.

Phase 2
Fault tolerance experiment
Phase 2: main results

- **Design and deployment of a computing resource Grid:**
  - [Internet – VPN – Corba – DIET – API-RobGrid – Appli]
  - Low level service support concurrent and redundant calls

- **Design and implementation of a high-level API:**
  - “Easy-to-use” high-level API (RobGrid)
  - High-level Grid service definitions
  - Standard Grid service contains and actions (Grid semantic)

- **Experiment of autonomous robot control across internet:**
  - Overlapping communications, computations and mechanical moves
  - Fault tolerance achievement (slow-down but go on)

Phase 3 …

Scale the number of sites
Scale the number of processes to control

Phase 4 …

… Install on Globus

… to be continued !
A Grid for process control

Questions?