A Client-Server Approach for Simulation over the Grid

Frédéric Desprez
LIP ENS Lyon
INRIA Rhône-Alpes
GRAAL Research Team

Join work with
A. Su, R. Bolze, E. Caron,
H. Dail, J.-Y. L’Excellent
LIP, Lyon
P. Amestoy, M. Pantel, C. Puglisi
ENSEEIHT/IRIT, Toulouse
Introduction

• One simple (and efficient) paradigm for grid computing
  • Offering (or renting) computational power and/or storage capacity through the Internet
  • Providing access to existing applications to thin clients

😊 Very high potential
  • Need of Problem Solving and Application Service Provider Environments
  • Installation difficulty for some libraries and applications
  • Some libraries or codes need to stay where they have been developed
  • Some data need to stay in place for security reasons

→ Using computational servers through a simple interface

😊 But
  • Always hard to use for non-specialists
  • Often application dependent PSEs
  • No sophisticated scheduling
RPC and Grid-Computing: GridRPC

- **One simple idea**
  - Implementing the RPC programming model over the grid
  - Using resources accessible through the network
  - Mixed parallelism model (data-parallel model at the server level and task parallelism between the servers)

- **Features needed**
  - Load-balancing (resource localization and performance evaluation, scheduling),
  - IDL,
  - Data and replica management,
  - Security,
  - Fault-tolerance,
  - Interoperability with other systems,
  - ...

- **Design of a standard interface**
  - within the GGF (GridRPC WG, C. Lee)
  - www.ggf.org, forge.gridforum.org/projects/gridrpc-wg
  - Existing implementations: NetSolve, Ninf, DIET, XtremWeb
RPC and Grid Computing: Grid RPC

Client

Request

AGENT(s)

S2 !

S1

S2

S3

S4

A, B, C

Answer (C)

Op(C, A, B)
RPC and Grid Computing: Grid RPC

- Adaptable grain
- Simple RPC API
- Libraries and applications integrated in Grid components
- IDL for the client interface, minimal information
- Task parallelism at the client/server level (using asynchronous calls), Data-parallelism at the server level ⇒ mixed parallelism

```c
double A[n][n], B[n][n], C[n][n];                /* data declaration */
dmmul(n, A, B, C);                                   /* local function call */
GRPC_call("dmmul", n, A, B, C);                     /* remote function call */
```
Hidden parallelism to the user

- One sequential call in the client code
  - Data transfer to the target server (maybe parallel one)
  - Resource reservation on the server
  - Distribution for the target parallel routine chosen by the server (agent)
  - Execution of the parallel code on the server (with or without check-pointing for fault-tolerance)
  - Gathering of the result and send to the client (pipeline?)

- Transparent for the client code!
Five fundamental components:

- **Client**
  Offers several user’s interface and submit requests to servers

- **Server**
  Receive clients request and executes software modules on behalf of them

- **Data-base**
  Contains both static and dynamic information about hardware and software resources

- **Scheduler**
  Gets clients requests and takes decisions to map tasks on servers depending of data stored in the database

- **Monitor**
  makes observations about resources status and stores information in the database
AGENT

- Central component of GRID-RPC systems
- Choose servers able to solve a request on behalf of clients
- Main task: load-balancing between servers
  - Gets information about available servers
  - Asks the performance database for information
  - Applies some scheduling heuristics
  - Can take care of
    - Some security (access authorization)
    - Fault tolerance
- 'Smart' localization mandatory
- Some scalability problems may occur

- Centralized (or duplicated) in NetSolve or Ninf
- Distributed in DIET
Agent Behavior

**Scheduling Unit**
- Scheduler
  - Query predictions
  - Query available servers
- NetworkMonitor
  - Inquire suitable server
  - Sends the task
- ServerMonitor
  - Monitor periodically
- ResourceDB
  - Store observed information
- Predictor
  - Perform Predictions
  - Returns scheduling info
- NetworkPredictor
- ServerPredictor

**Global Computing Environment**
- Client
  - Invoke task
  - Sends the task
- Server
  - Execute task
  - Return results
- Network
  - NetworkMonitor
  - Scheduler
  - ResourceDB
  - Predictor
  - NetworkPredictor
  - ServerPredictor

The diagram illustrates the flow of interactions among the components in a global computing environment.

- The Client initiates the process by invoking a task.
- The Scheduler queries predictions and available servers.
- NetworkMonitor checks for suitable servers periodically.
- ServerMonitor monitors the system and returns scheduling information.
- ResourceDB stores observed information.
- Predictor performs predictions and returns scheduling info.
- NetworkPredictor and ServerPredictor contribute to the prediction process.

The interactions ensure effective resource management and task execution.
Distributed Interactive Engineering Toolbox
DIET’s Goals

• Our goals
  ❖ To develop a toolbox for the deployment of environments using the Application Service Provider (ASP) paradigm with different applications
  ❖ Use as much as possible public domain and standard software
  ❖ To obtain a high performance and scalable environment
  ❖ Implement and validate our more theoretical results
    ▪ Scheduling for heterogeneous platforms, data (re)distribution and replication, performance evaluation, algorithmic for heterogeneous and distributed platforms, …

• Based on CORBA, NWS, LDAP, and our own software developments
  ❖ FAST for performance evaluation,
  ❖ LogMgr for monitoring,
  ❖ VizDIET for the visualization,
  ❖ GoDIET for the deployment

• Several applications in different fields (simulation, bioinformatic, …)
• Release 1.1 available on the web
• ACI Grid ASP, RNTL GASP

http://graal.ens-lyon.fr/DIET/
Client Interface

- As simple as possible
- Multi-interfaces (C, C++, Fortran, Java, Matlab, Mathematica, Scilab, Web, ...)
- Proposition of a standard interface within the Global Grid Forum (DIET, Ninf, and Netsolve)
DIET Architecture

- **Client**
- **Server front end**
- **Master Agent**
- **Local Agent**
- **JXTA**
- **MA**
- **FAST library**
- **Application Modeling**
- **System availabilities**
- **LDAP**
- **NWS**
Request Management
Some Research Topics

• Scheduling
  ◆ Distributed scheduling
  ◆ Software platform deployment with or without dynamic connections between components
  ◆ Plug-in schedulers

• Data-management
  ◆ Scheduling of computation requests and links with data-management
  ◆ Replication, data prefetching
  ◆ Workflow scheduling

• Performance evaluation
  ◆ Application modelization
  ◆ Dynamic information about the platform (network, clusters)

• Applications
  ◆ Bioinformatic, geology, physic, chemical engineering, sparse solvers evaluation, …
Data Management
Data/replica management

• Two needs
  - Keep the data in place to reduce the overhead of communications between clients and servers
  - Replicate data whenever possible

• Two approaches for DIET
  - DTM (LIFC, Besançon)
    - Hierarchy similar to the DIET’s one
    - Distributed data manager
    - Redistribution between servers
  - JuxMem (Paris, Rennes)
    - P2P data cache

• NetSolve
  - IBP (Internet Backplane Protocol) : data cache
  - Request Sequencing to find data dependences

• Work done within the GridRPC Working Group
  - Relations with workflow management
- Persistence at the server level
- To avoid useless data transfers
  - Intermediate results (C, D)
  - Between clients and servers
  - Between servers
  - “transparent” for the client
- Data Manager/Loc Manager
  - Hierarchy mapped on the DIET hierarchy
  - modularity
- Proposition to the Grid-RPC WG (GGF)
  - Data handles
  - Persistence flag
  - Data management functions
- A peer-to-peer architecture for a data-sharing service in memory
- Persistence and data coherency mechanism
- Transparent data localization

**Project JXTA**
- Toolbox for the development of P2P applications
  - Set of protocols
- One peer
  - Unique ID
  - Several communication protocols (TCP, HTTP, …)
JuxMEM Architecture

- A peer-to-peer architecture for data-sharing
Experimentations
Target Platform

4 - 128 Clients
LS - Lyon

Local Agent
PARACI - Rennes

1-32 SeDs

Master Agent
LS - Lyon

LA
Cristal - Rocquencourt

LA
Cobalt - Grenoble

Local Agent
LS-Lyon
DIET Scalability with # clients. Size = 10.

![Graph showing scalability with number of clients]

- **Latency**
- **Server Calls**
- **Diet Calls**

*Axes:*
- Y-axis: Time (seconds)
- X-axis: Number of clients
DIET Scalability with # clients. Size = 1000
“My team has created a very innovative solution, but we’re still looking for a problem to go with it.”
• Stereoscopic processing:
  • Maximal matching between the spots of both pictures.
  • Elevation computation.

View angles information and coordinates of initial corresponding points

• Geometrical constraints
• Optical disparities

MNT Binary files
Digital Elevation Models (MNT), cont.
Tests for Large Systems of Equations
Coordinated by ENSEEIHT-IRIT, Toulouse
Funded by ACI GRID

• Goal
  - Provide a friendly test environment for expert and non-expert users of sparse direct linear algebra software
  - Easy access to software and tools, a wide range of computer architectures, matrix collections
  - On a user’s specific problem, compare execution time / accuracy / memory usage / … of various sparse solvers
    - public domain … as well as commercial,
    - sequential … as well as parallel
    - Find best parameter values / reordering heuristics on a given problem

http://www.enseeiht.fr/lima/tlse
Request Examples

- Memory required to factor a matrix
- Error analysis as a function of the threshold pivoting value
- Minimum time on a given computer to factor a given unsymmetric matrix
- Which ordering heuristic is the best for solving a given problem
### Results for solver MUMPS

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Realisation</th>
<th>Ordering</th>
<th>EstimatedMemory</th>
<th>EffectiveMemory</th>
</tr>
</thead>
<tbody>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>AMD</td>
<td>123000</td>
<td>123000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>AMD</td>
<td>118000</td>
<td>115000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>PORD</td>
<td>92000</td>
<td>92000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>METIS</td>
<td>93000</td>
<td>90000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>QAMO</td>
<td>123000</td>
<td>123000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>MMDx</td>
<td>131000</td>
<td>127000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>MMD +</td>
<td>123000</td>
<td>121000</td>
</tr>
<tr>
<td>/local/share/matrices/DISK0/ex11/rua</td>
<td>MUMPS</td>
<td>COLAMD</td>
<td>138000</td>
<td>137000</td>
</tr>
</tbody>
</table>

### Expertise results

**GRID-TLSE Project**

- **EIM**: Estimated Memory
- **EM**: Effective Memory

![Bar Chart](image)

### Expertise comments

There are no comments on this expertise.

- Make a comment
Why using a grid?

- Sparse linear algebra software makes use of sophisticated algorithms for (pre/post)-processing the matrix
- Multiple parameters interfere for the efficient execution of sparse linear solvers
  - Ordering
  - Amount of memory
  - Architecture of the target computer
  - Available libraries
  - Determining the best combination of parameter values is multi-parametric problem
  - Combinatorial nature of these parameters
- The installation of any sparse solver library on a new architecture can be a nightmare!
- Testing different architectures
- Always using the latest version of each library
Is it realistic?

- Time to send the data can be more important than the computation itself!

- But
  - Large number of independent requests
  - Time to answer is not critical
  - Data persistency between elementary requests easy to express

- Clear need for the users!
  - Managing software and hardware testing from a PSE
Architecture

- **Client** sends experiment requests.
- **Expert** writes scenarios, deploys new software.
- **Expert Site Grid TLSE** connects clients and experts.
- **Websolve** requests expertise.
- **Weaver** runs synthetics requests.
- **DIET** requests results.
- **Database** consults and modifies.
- **Scenarios Services** consults.
- **History Log Files** modifies.
- **Matrix Collections** extracts.
- **Static** and **Dynamic** services.
- **Stats** and **Client Provided Matrix** services.
Research Issues

- **Sparse Linear Algebra**
  - Automatically choosing the right parameters, the correct sequence of operations
  - Help the user as much as possible

- **Scenarii Management**
  - Generation and management of workflows
  - Need to be connected to the scheduling of requests/data management

- **Interoperability**
  - Connecting different libraries with different data formats
  - Meta-data

- **Data Management**
  - Leaving data in place as much as possible
  - Matrix collections

- **DIET Research issues**
  - Managing requests of different sizes with data dependences
Conclusion and Future Work
We did not talk about …

- **Automatic deployment**
  - Depending of the target architecture, the location of servers, clients, …

- **Distributed scheduling**
  - Plugin schedulers, relations with batch schedulers, application dependent scheduling

- **Other applications**
  - Simulation (physic, chemical eng., …), robotics, bioinformatic, geology, …

- **Adding services**
  - Registering new applications

- **Performance evaluation**
  - Routine/application cost, data (re)distribution, computation of the optimal number of processors used on the servers

- **Fault tolerance**
  - Agent, servers, checkpointing

- **Platform monitoring**
  - Distributed log management (LogService), post-mortem visualization (VizDIET)

- **Security**
  - Authentication, communications, firewalls, …
Conclusions and future work

• GridRPC
  ✷ Interesting approach for several applications
  ✷ Flexible and efficient
  ✷ Many interesting research issues (scheduling, data management, resource discovery and reservation, deployment, fault-tolerance, …)

• DIET
  ✷ Scalable, open-source, and multi-application platform
  ✷ Concentration on several issues like resource discovery, scheduling (distributed scheduling and plugin schedulers), deployment (GoDIET), performance evaluation (FAST and Freddy), monitoring (LogService and VizDIET), data management and replication (DTM and Juxmem)
  ✷ Large scale validation on the Grid5000 platform
  ✷ Demo @ SC2004

• TLSE
  ✷ Help for the development of high performance sparse direct solvers
  ✷ Opening the whole platform in 2005 (CSC 2005 ?)
  ✷ Demo @ SC2004

http://www.grid5000.org/

http://graal.ens-lyon.fr/DIET (online demo)
Questions ?

http://graal.ens-lyon.fr/DIET