How to bring the power of Cloud and Distributed Infrastructures to Scilab?
1 Motivations

2 The middleware
   - Overview
   - DIET architecture

3 Scilab: Integration architecture
   - Server side
   - Client side: Scilab module

4 Example

5 Conclusions and Future work
Motivations

- Increase Scilab computational power
- Use Scilab in a distributed environment
- Brings the Scilab capabilities to a middleware
- Brings the middleware capabilities to Scilab

What we learn? What we show?

- How to interface a middleware with Scilab?

Former approaches

- Message-passing (PVM)
- Parallel libraries (ScaLAPACK)
DIET: a Grid and Cloud middleware

- **Distributed Interactive Engineering Toolbox**

- **Context:** Development of a toolbox for deploying application services providers with a hierarchical architecture for scalability

- A middleware designed for HPC in heterogeneous and distributed environments (i.e. Grid and Clouds)

- One simple approach: RPC programming model for large distributed infrastructure
  - DIET is compliant with the Grid-RPC standard

- GRAAL research team and SysFera company starting 2000

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

**Client**
- Application which uses DIET to solve problems
- Diversity – Web page/PSE/compiled program
- Transparency – only knows the Master Agent

**Master Agent**

**Local Agent**

**Server Daemon**
DIET components

**Client**

**Master Agent**
- Platform entry point
- Finds the best server instance
- Returns a reference to the client

**Local Agent**

**Server Daemon**
DIET components

Client

Master Agent

Local Agent
- Transmits information and data between MA and servers
- Stores information for each subtree
- Used to create a scalable hierarchy

Server Daemon
DIET components

Client

Master Agent

Local Agent

Server Daemon

- Computational server – manages a processor or a cluster
- Offers services and information about itself
DIET architecture

- **MA**: Master Agent
- **LA**: Local Agent
- **SeD**: Server Daemon

The middleware for DIET architecture.
Integration

- Where does Scilab fit in a DIET architecture?
  - Client
  - Server
- What APIs to use?
Server side: Multiple Scilab instances for multiple clients

One instance per request

- ✓ Own environment for request
- ✓ Serving multiple clients at once
- ✗ Requests overlap
- ✗ Variables name conflict
- ✗ Start and terminate – time consuming functions
Server side: Single Scilab instance for multiple clients

Single instance with mutex

- ✔ Faster – no need to restart for each client
- ✔ No overlapping
- ✔ No conflicts
- ✗ Blocking calls
- ✗ Requires clean-up

Single start/ single terminate
Server side

- Choice – single instance with mutex lock
- Use `call_scilab` interface
  - Start/Stop a Scilab instance
  - Send commands
- Access for non-Scilab users
Client side: Scilab module

Scilab module
- ATOMS (AuTomatic mOdules Management for Scilab)
- Fast and easy integration
- Well defined structure
- Load and start in two simple steps
What is it?
The packaging system of Scilab external modules

What does it offer?
- Multi-platform
- Guidelines
- Skeleton to get started
- Calls to external C/C++/Fortran programs
Data management

Data types

- **DIET_FILE** – easy transfer of scripts
- **DIET_CONTAINER** – ALL other types of data
  - double, integer, string, boolean
  - scalars and matrices
  - sparse matrices
  - polynomials
  - real and complex, signed or unsigned numbers
Four simple steps

1. Create a script / Declare variables
2. Load the sci2diet module
3. Call DIET from within Scilab
4. Get results!
Example

Example - client side view

```
-->exec loader.sce;
Start DIET
  Load macros
  Load gateways
  Load help
  Load demos

-->chdir /home/adelaneacsu/Documents/script_modul

-->clear

-->n = 4;

-->DIET_call("myScript.sce", "C ", "n");
```
Example - Client side view

```scilab
--> clear
--> n = 4;
--> DIET_call("myScript.sce", "C ", "n");
--> disp(C);

4

310.  404.  498.
340.  446.  552.
370.  488.  606.
400.  530.  660.
430.  572.  714.
```
Example - Server side view

```matlab
---A = matrix(1:3:60, n+1, n);
---for i = 1:n
---   for j = 1:(n-1)
---     B(i,j) = i+j-1;
---   end;
---end;
---C = A * B;
---hi = ["Hello", "World"];
---disp(C);

310.  404.  498.
340.  446.  552.
370.  488.  606.
400.  530.  660.
430.  572.  714.
---clear
```
Conclusions and Future work

Conclusions

- Successful integration
- Two-parts architecture
  - Run Scilab tasks on server
  - Call DIET from a Scilab instance
- Easy to use
- Code available here: http://graal.ens-lyon.fr/DIET
Conclusions and Future work

Future work

Data persistency
- Store intermediary data on DIET platform
- Save time and memory
- Requires pre-knowledge about data

Improve scheduling
- Plug-in scheduler for DIET
- Collect information from the Scilab engine directly
- Better and faster selection of servers
Thank you for your attention!