

A Generic Framework for Automatic Deployment on Grids


Journées  Lille
30-31 Octobre 2006

[Areski Flissi](#) (LIFL/CNRS), [Philippe Merle](#) (INRIA/LIFL)

Equipe GOAL / Projet INRIA Jacquard

Laboratoire d'Informatique Fondamentale de Lille, France
UMR CNRS 8022

Outline

- Introduction, problem, scenario
- Our proposition: A Generic Deployment Framework to automatize deployment on grids
- Illustration: Deployment of OpenCCM middleware and CORBA Components-based applications on 
- Evaluation & performances
- Conclusion & perspectives

- Availability of some grid infrastructures as Grid5000, the french grid infrastructure for research in computer science
 - => **Software deployment tends to become a major research topic**
- Deployment of software on large systems as grids is a complex activity
 - Set of heterogeneous tasks to orchestrate, e.g.:
 - reservation of nodes,
 - installation/uninstallation of software, libraries or binaries on remote nodes,
 - configuration of nodes' environment (e.g. shell variables),
 - starting of application servers, middleware services or processes,
 - activation/unactivation of software instances
 - ...
 - Users can spend many hours to prepare and start their experiences!

Current solutions for deployment:

- Don't address the problem of software deployment **independently of underlying technologies**
- Don't cover the **whole deployment process**:
 - Component-based platforms and distributed applications:
 - JOnAS or OpenCCM provide features to respectively deploy J2EE or CCM applications
 - OSGi technology (gateways) offers mechanisms for remotely deploying and managing OSGi bundles (representing end-user services)
 - => **But nothing is telled about the deployment of the middleware itself**
 - ProActive middleware (which is dedicated to grid computing) addresses the deployment of both servers and applications
 - => **But built on top of this technology!**

Current solutions for deployment:

- Are mainly based on "ad hoc" solutions
 - File transfer protocols, e.g. FTP, HTTP, scp, rsync
 - Remote access protocols, e.g. SSH, Telnet, rlogin
 - Shells or batch commands or scripts
 - Programs...
 - etc.
- ⇒ Very difficult to reuse, to automatize
 - i.e. Changing of targeted domain, parallel execution of scripts ??
- Or specific tools
 - Such as Grid5000's OAR (submission of nodes reservation requests) or Kadeploy (deployment of customized operating system images) tools.
- ⇒ Users/applications must be aware of and manage themselves allocated nodes

```
user@grid5000$ ~/my_program my_nodelist
```

- Example : Deployment of OpenCCM middleware and CCM applications on Grid5000 infrastructure
 - Requires execution of several "heterogeneous" tasks

1) *Accessing resources and reserving nodes*

- (i) Connect to the frontal node of one cluster, e.g. using SSH

```
$ ssh frontale.lille.grid5000.fr  
$ ssh oar
```

- (ii) Check for available nodes

```
$ oargridstat
```

- (iii) Submit reservation request, e.g. using OARGrid

```
$ oargridsub lille:nodes=50,gdx:nodes=200,bordeaux:nodes=25 -F -w  
"1:30:00"
```

- (iv) Get *IdJob* and the list of allocated nodes

```
$ oargridsub -l <IdJob> > nodelist
```

2) *Installation of software*

(v) Install OpenCCM application servers on each node, plus all required libraries (JRE, CORBA ORB, etc.)

```
$ for node in nodelist do
  ssh $node wget
    http://openccm.objectweb.org/.../OpenCCM-0.9.0-JacORB-2.1.tar.gz &
  ssh $node tar -xvfz OpenCCM-0.9.0-JacORB-2.1.tar.gz
```

- Id. for the Java Runtime Environment, the CORBA Object Request Broker (e.g. JacORB)... i.e. for all dependencies!

(v') Or deploy your specific (previously built) image using Kadeploy

```
$ kadeploy -f nodelist -e OpenCCM-0.9_JacORB-2.1_JRE-1.5_Ubuntu
```

3) *Configuration*

(vi) Configure environment by setting shell variables, path, etc.

```
$ for node in nodelist do
  ssh $node "export JAVA_HOME=/opt/java/jdk-1.5.0_05 &
    export OpenCCM_HOMEDIR=/opt/OpenCCM-0.9 &
    export PATH=$JAVA_HOME/bin:OpenCCM_HOMEDIR/bin:$PATH"
```

4) *Starting servers*

(vii) Start application servers (that will host the business components) and useful middleware services (e.g. CORBA CosNaming service), **taking care of dependencies and synchronisation problems**

```
$ ssh <first_node> "ccm_install & comanche_start & ns_start"

$ ssh <another_node> "ccm_install & comanche_start & dci_start"

$ ssh <yet_another_node> "ccm_install &
                           ns_set http://<first_node>:8080/NameService.IOR &
                           dci_set http://<first_node>:8080/DCI.IOR &
                           factory_start MyFactory"

$ for node in nodelist do
    ssh $node "ccm_install &
              ssh ns_set http://<first_node>:8080/NameService.IOR &
              ssh dci_set http://<first_node>:8080/DCI.IOR &
              ssh node_start <node_name>"
```

5) *Instantiation*

(viii) Finally start the "high-level" deployment process of the application:

- download and install component binaries on nodes, configure, instantiate, and connecting them thru their ports

```
$ ssh <a_node> ccm_deploy -F MyFactory
                                     ~/archives/HelloWorld.aar
```

6) **Stop...**

(ix) Stopping all servers & processes

```
$ for node in nodelist do
    ssh $node ...
```

7) **Uninstall...**

(x) Uninstall software

```
$ for node in nodelist do
    ssh $node rm -rf...
```

8) **Releasing resources...**

(xi) Kill job and release nodes

```
$ ssh oar oargriddel <IdJob>
```

- **Deployment process on large systems is a complex activity**
 - **Heterogeneous tasks**
 - such as resources access, nodes reservation, installation of software, configuration, remote processes starting, etc.
 - **User/deployer must be aware of targeted platforms when scripting/programming their deployment**
 - Which are my nodes list this time and how my program get it?
 - node-X is accessible with SSH using RSA keys authentication system whereas node-Y is accessible using Telnet protocol with login/password information
 - Node-X's shell is SH / node-Y's one is CSH
 - Id. for transfer protocols which can be various
 - **Tasks or software dependencies problems**
 - reservation must be obviously made before software installation
 - JRE or ORB have to be installed before OpenCCM
 - Java processes for application servers cannot be launched before some specific services have been launched and in a started state, etc.
 - **Deployment scripts or programs are very difficult to re-use in another context or simply for the next experience without manually adapt them**
 - **Automatizing or parallelizing the deployment process is very difficult to reach**
 - **Users/deployers of grid applications are not always computer science scientists or engineers!**

- Introduction, problem, scenario
- **Our Proposition: A Generic Deployment Framework to automatize deployment on grids**
- Illustration: Deployment of OpenCCM middleware and CORBA Components-based applications on Grid5000
- Evaluation & Performances
- Conclusion & Perspectives

Generic Deployment Framework

- **Motivations, Objectives:**
 - Allow users to **describe**, in a simple natural language, **configurations to deploy** instead of programming or scripting how the deployment process will be realized/implemented
 - **Abstract dependencies or synchronization problems** between elementary tasks and/or software
- => Ultimate goal = complete automatization of complex deployment processes
- **Our proposition = FDF (Fractal Deployment Framework)**
 - A generic tool for **automatization of deployment on grids**
 - Implemented using the Fractal software component model (Java)
 - Independent of both **technologies** & **granularity** of software to deploy
 - Independent of **targeted platforms/infrastructures**
 - e.g. Grids (e.g. Grid5000), clusters, LAN...

- Approach

Some key principles:

- Everything is reified as "components":
 - Remote access protocols, e.g. SSH, Telnet, rlogin
 - File transfer protocols, e.g. FTP, HTTP, scp, rsync
 - Shells, e.g. SH, CSH, MS Windows Command Shell
 - Shell variables and commands...
- Targeted platforms and physical hosts, e.g. Grid5000 nodes
- Software, application servers, middleware services...

} deployment components

} physical infrastructure

} software personalities

Not reimplementing the wheel but rely on existing mechanisms and tools!

- Compose these components to obtain a composite that symbolizes the software to deploy
 - Bindings between components represent dependencies

Executing this composite means execute the deployment process

- Methodology

- Use and assemble the generic deployment framework (common library of reusable deployment components) and personalities to build the appropriate tool that:
 - Automatically orchestrate the deployment process of any software, regardless of:
 - The technology:
 - » J2EE, CCM, Web Services, OSGi, Object-Oriented applications, etc.
 - The granularity:
 - » Software components, objects, scripts, programs, middleware services, application servers, libraries, binaries, processes, etc.

- (1) Deployment components

- FDF provides a set of generic, and extensible, deployment components used by developers & end-users

- These components implement basic generic server interfaces

- » *Protocol, User, Port, Hostname, Shell, Transfer*, etc.

- Many implementations available

- » *Protocol* interface implemented by *SSH, Telnet* components

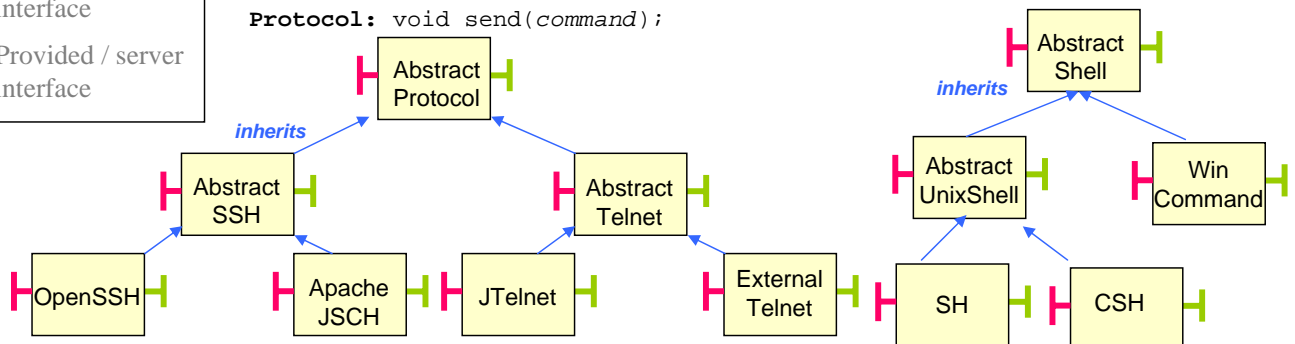
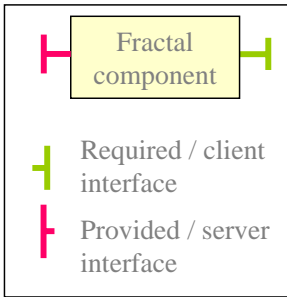
- » *Shell* interface implemented by *SH, CSH* or *WinCommand* components

```
void execute(cmd_to_exec);
```

```
Shell: void setVariable(name,value);
```

```
void unsetVariable(name,value);
```

```
Protocol: void send(command);
```



© A. Flissi

17

- (1) Deployment components

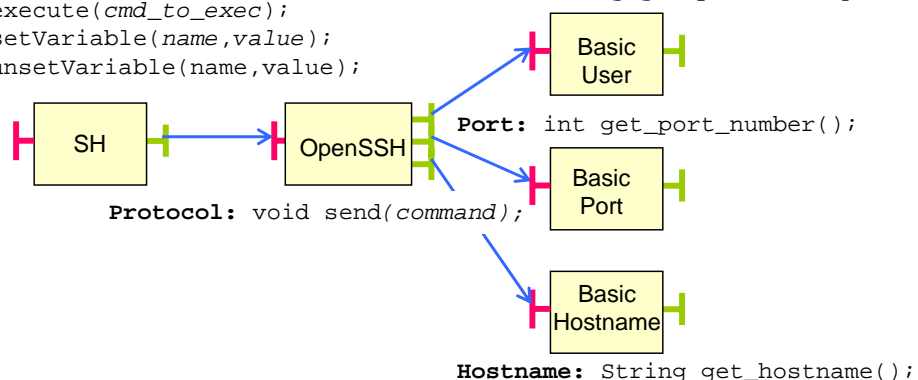
- Bindings between components symbolize dependencies

- *Protocol* component reifies a "real" protocol, is bound to *User, Port* and *Hostname* components to respectively get user access information, port number and hostname

- *Shell* component reifies a "real" shell, provides a method to execute shell commands, needs a remote access protocol to send the command

```
void execute(cmd_to_exec);
Shell: void setVariable(name,value);
void unsetVariable(name,value);
```

```
String get_login();
User: String get_password();
String get_private_key();
```

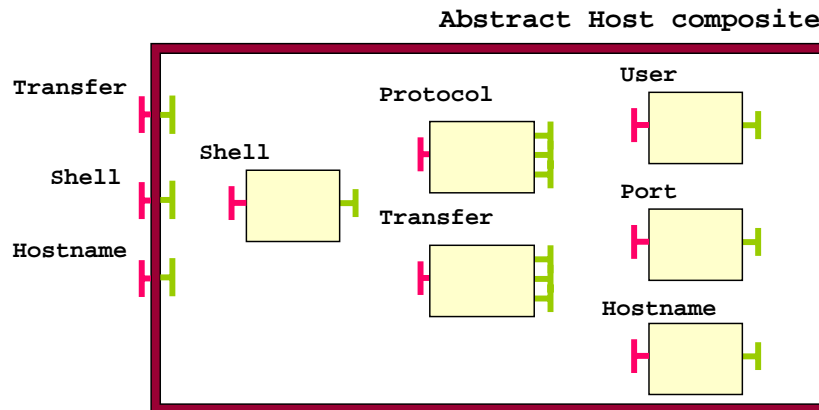


© A. Flissi

18

Generic Deployment Framework

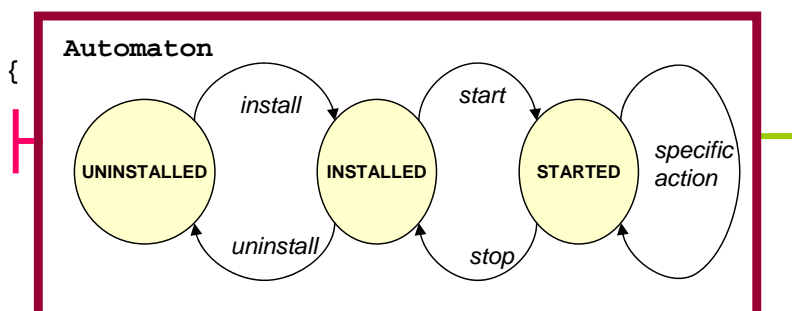
- (2) Generic abstract *host* composite
 - Reify physical hosts such as nodes of grids.
 - Composition of some generic deployment components
 - Main server interfaces exported, such as *Shell*, *Protocol*...



Generic Deployment Framework

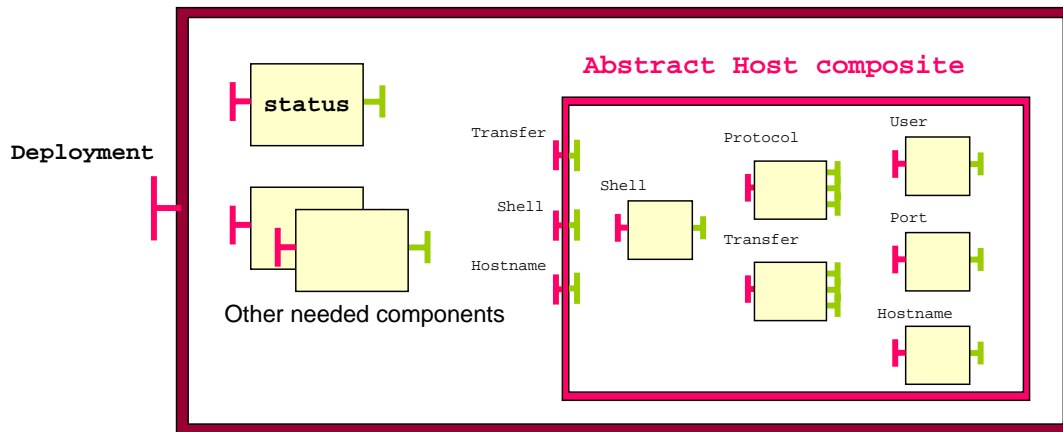
- (3) Software components (composite) → personalities
 - Implement generic *Deployment* interface that provides main elementary deployment *actions* (install, start, stop, uninstall)
 - Personalities are written once by developers for each *software* to deploy
 - Using *Fractal ADL* (XML) files that "textually" describe the composite representing the software and the associated deployment actions
 - Generic deployment components and host composite are used
 - Integrated automaton for orchestration

```
interface Deployment {
  install();
  start();
  stop();
  uninstall();
}
```

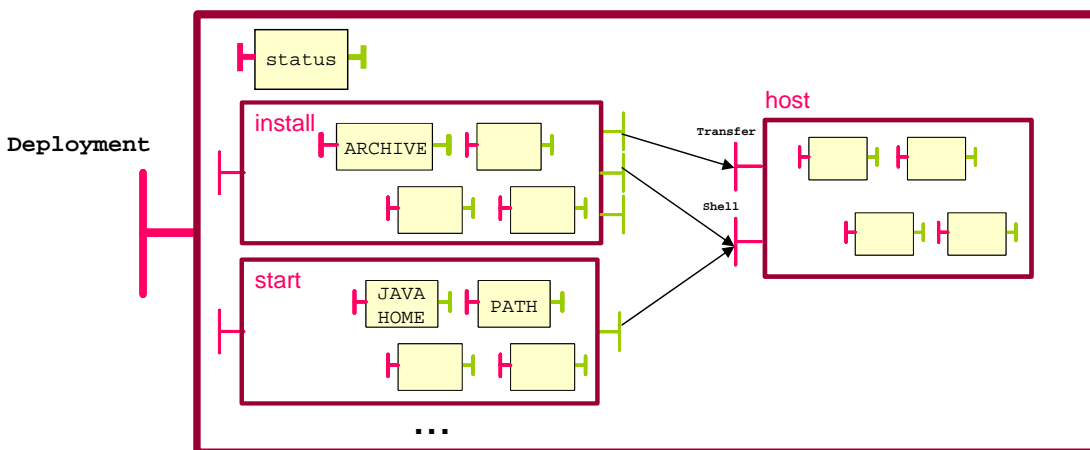


- (3) Software components (composites) → personalities
 - Contain an abstract *Host* composite + *status* component (INSTALLED, STARTED...)
 - Abstract components in *Host* will be replaced with concrete component by users in end-user configuration, i.e.
 - » *AbstractProtocol* --> *OpenSSH* | *JTelnet*... implementing Protocol interface
 - » *AbstractTransfer* --> *Apache_FTP* | *SCP*... implementing Transfer interface
 - » *hostname, user, port* component' s attributes are set

Software component



- Ex. JRE composite = Java software personality
 - => Specified in Fractal ADL XML files (JRE.fractal)



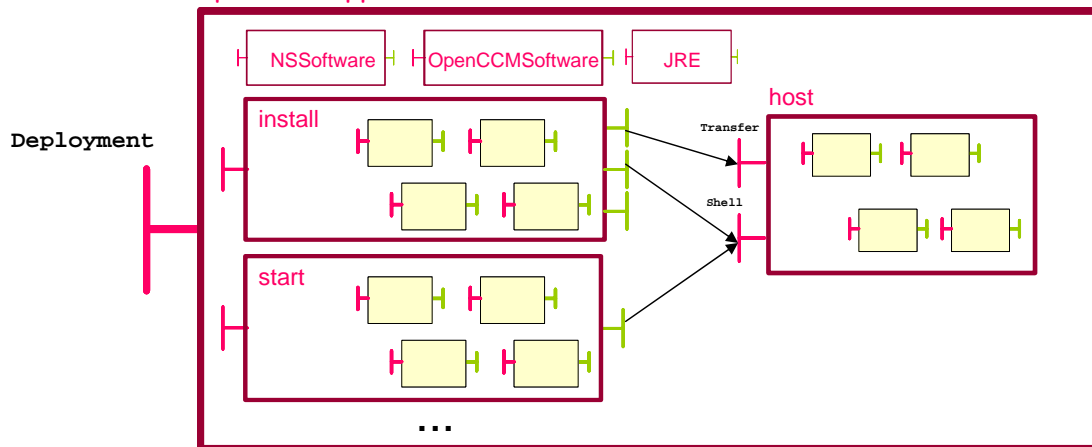
- Introduction, problem, scenario
- Our Proposition: A Generic Deployment Framework to automatize deployment on grids
- **Illustration: Deployment of OpenCCM middleware and CORBA Components-based applications on Grid5000**
- Evaluation & Performances
- Conclusion & Perspectives

- **OpenCCM middleware**
 - Open-source platform for CORBA Components-based, distributed, and heterogeneous applications
 - 1st implementation of the *OMG CCM* specification
- **Deployment of OpenCCM platform and CCM applications on large infrastructure as Grid5000:**
 - Deployment of some middleware services (e.g. a CORBA CosNaming service), few micro-http servers to give access to services/objects CORBA references, OpenCCM specific processes/daemons (DCI)
 - OpenCCM application servers that will host business components of the application
 - Applications
- **FDf personalities**
 - **OpenCCM software personality:**
 - Write Fractal ADL files to describe software, i.e. components representing OpenCCM entities (software, services, app. servers, deploy tool, etc.)
 - **Grid5000 personality:**
 - Components reifying Grid5000 tools such as OARGrid (reservation of nodes in many clusters), implied in global deployment process

- OpenCCM personality's components:

- OpenCCM.**OpeCCMSoftware** -> reify the OpenCCM software, contains other software such as JRE, ORB as dependencies
- OpenCCM.**NSSoftware** -> reify CORBA CosNaming service
- OpenCCM.**AppServerSoftware** -> reify the OpenCCM application server, contains the OpenCCMSoftware and NSSoftware components as dependencies

- Ex. **OpenCCM Application Server Software**



© A. Flissi

25

```

Notepad++ - E:\af\lgada06\example.fractal
Fichier Edition Recherche Affichage Format Document Langage Paramètre Macro Exécution Plugins ?
example.fractal
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE definition PUBLIC "-//objectweb.org//DTD Fractal ADL 2.0//EN"
3 "classpath://org/objectweb/fractal/adl/xml/standard.dtd">
4 <!--
35 <!-- ADL for the OpenCCM Java Component Server software. -->
36 <definition
37 name="org.objectweb.fdf.components.openccm.JCSSoftware"
38 extends="org.objectweb.fdf.components.software.Software(jcs,jcs),..."
39 arguments="cs_name,jcs_args="
40 >
41 <!-- Sub components used to configure the JCS software. -->
42 <component name="openccm" />
43 <component name="ns" />
44 <component name="ior_ns" definition="OpenCCM.IOR(IOR_NS NOT SET!!)" />
45 <!-- Implementation of the JCS software. -->
46 <component name="jcs">
47 <component name="internal-deployment">
48 <component name="ns" definition="./ns" />
49 <component name="openccm" definition="./openccm" />
50 <component name="install">
51 ...
52 </component>
53 <component name="start">
54 <component name="echo-begin" definition="org.objectweb.fdf.util.printer.lib.runnable.Echo(BEGIN START)" />
55 <component name="set_ns_ior" definition="org...shell.lib.runnable.Execute(ccm_set_ior NameService @[ior_ns]@)" />
56 <component name="set_proxy_ior" definition="org...shell.lib.runnable.Execute(ccm_set_ior Proxy @[ior_proxy]@)" />
57 <component name="set_display" definition="org...shell.lib.runnable.SetVariable(DISPLAY,@#[display]@)" />
58 <component name="jcs_start" definition="org...shell.lib.runnable.Execute(jcs_start ${jcs_args} ${cs_name})" />
59 <component name="echo-end" definition="org.objectweb.fdf.util.printer.lib.runnable.Echo(END START)" />
60 </component>
61 <component name="stop">
62 ...
63 </component>
64 <component name="uninstall">
65 ...
66 </component>
67 </component>
68 </definition>
69 </component>
70 </component>
71 </definition>

```

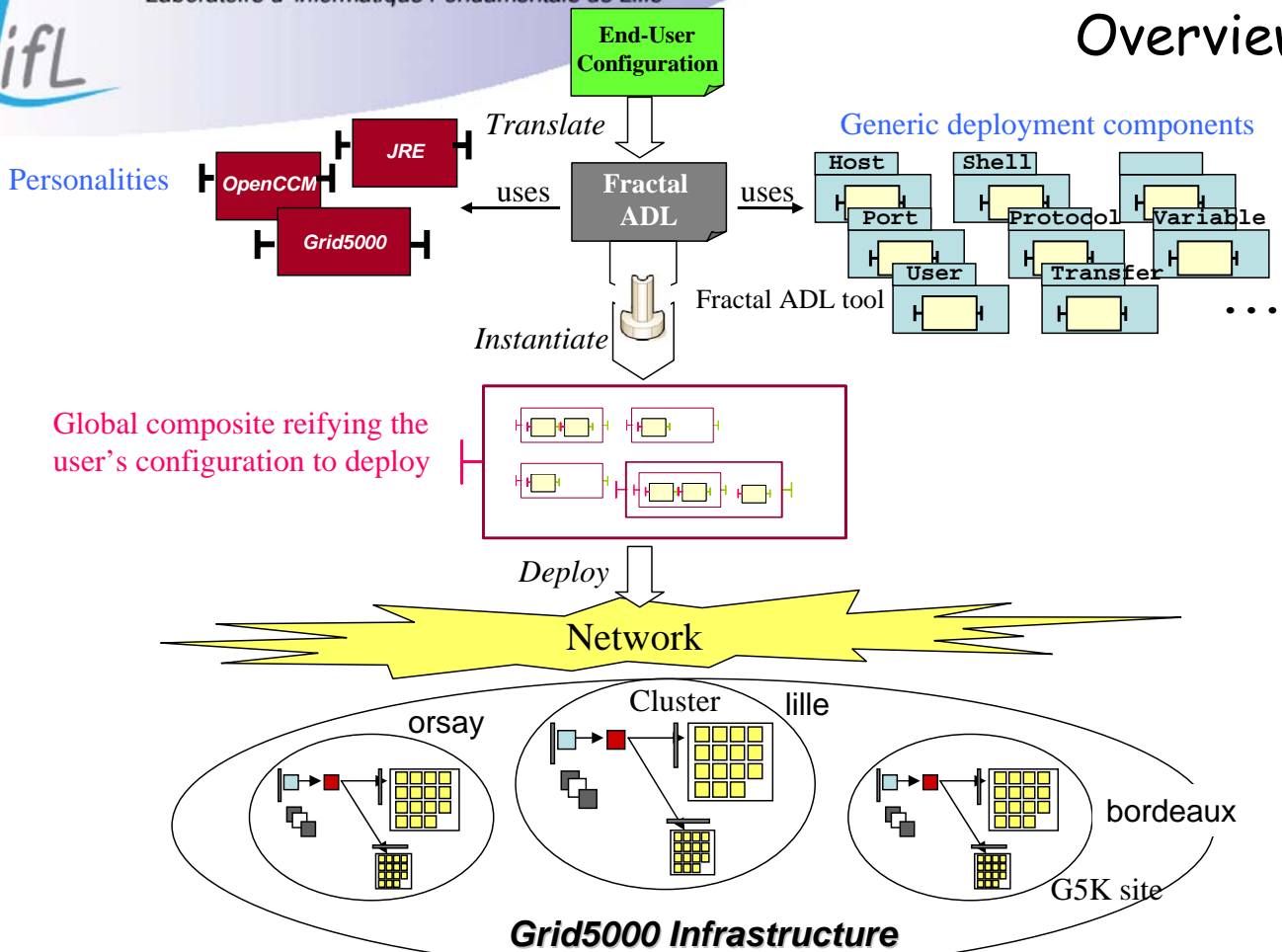
- Grid5000 personality's components:
 - Wrapping G5K tools as OARGrid => OARGrid.fractal ADL
- Deployment of OpenCCM on 1000 nodes of many clusters of Grid5000 using FDF+OpenCCM+Grid5000 personalities:
 - End-user file describing the configuration to deploy: [Example.fdf](#)
 - FDF Parser translates this configuration and Fractal ADL file representing the user's final configuration is generated: [Example.fractal](#)
 - Fractal ADL tool interprets it and execute the deployment process!
- End-user configuration is divided in 2 parts:
 - **Hosts declarations** ~ describing the physical nodes
 - **Services declarations** ~ describing the software to deploy or other elementary deployment tasks

```
<!-- The declaration of user configuration to deploy -->
MyDeployment {
<!-- The declaration of nodes -->
Hosts = {
    oar_node = INTERNET.HOST {
        user = INTERNET.USER(flissi, ~/.ssh/id_rsa)
        hostname = INTERNET.HOSTNAME(oar.lille.grid5000.fr)
        protocol = PROTOCOL.OpenSSH
        shell = SHELL.SH
    }
    g5k-nodes = {
        apply For(i,0,1000) {
            node-%{i} = GRID5000.DYNAMICNODE(~/.nodeslist) {
                user = INTERNET.USER(flissi, ~/.ssh/id_rsa)
                transfer = TRANSFER.SCP
            }
        }
    }
} <!-- end declaration of nodes -->
```

```

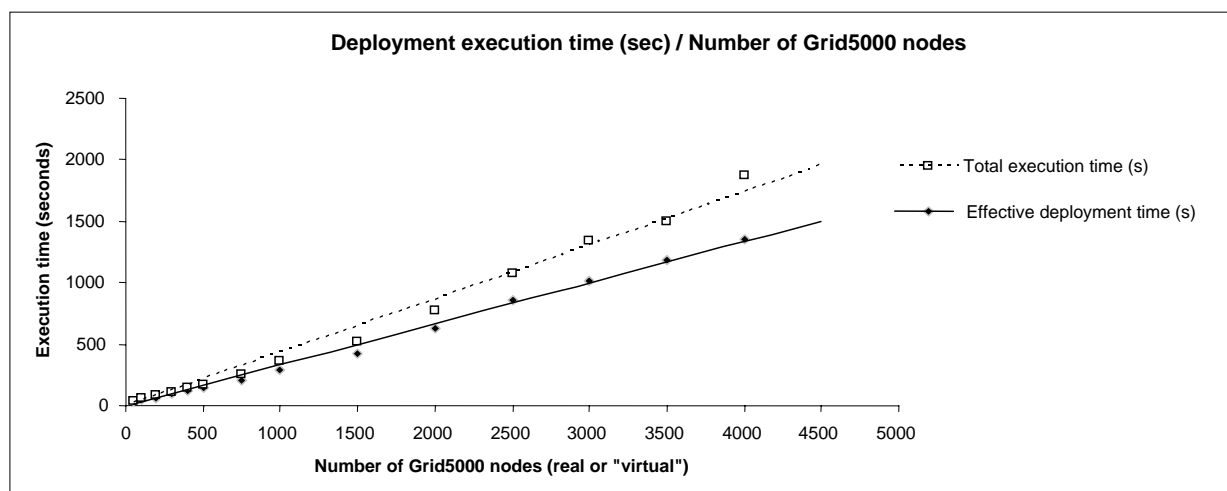
<!-- The tasks, services and software to deploy on declared nodes-->
OARGrid Nodes Reservation = GRID5000.OARGRID (lille:nodes=50/.../bordeaux:nodes=50 -w 2:30:00,~/nodeslist)
{
    host = Hosts/oar_node
}
OpenCCM-Application-Servers = FDF.SEQUENTIAL-COLLECTION {
    CORBA-CosNaming-Service = OpenCCM.NS {
        host = Hosts/g5k-nodes/node-0
        ior_ns = OpenCCM.IOR(http://...:8080/NameService.IOR)
    }
    OpenCCM-Java-Component-Servers-Cluster = FDF.PARALLEL-COLLECTION {
        apply For(i,0,1000) {
            node-%{i}" = OpenCCM.JCS(CS_%{i}_`hostname`) {
                java = JAVA.JRE {
                    archive = JAVA.ARCHIVE(/tmp/JRE-1.5.tgz)
                    home = JAVA.HOME(/opt/jdk1.5.0_05)
                }
                openccm = OpenCCM.SERVER(/opt/OpenCCM-JacORB-2.1,/opt/ORB/JacORB-2.1)
                ns = OpenCCM-Application-Servers/CORBA-CosNaming-Service
                host = Hosts/g5k-nodes/node-%{i}
            }
        }
    }
    CCM Applications Deployment = {
        chat demo = OpenCCM.DEPLOYER(chat,/tmp/chat.aar,DefaultFactory) {
            host = Hosts/g5k-nodes/node-0
            ior_ns = OpenCCM.IOR(http://...:8080/NameService.IOR)
        }
    }
}
} <!-- end declaration of services -->
} <!-- end of configuration declaration -->

```



- Introduction, problem, scenario
- Our Proposition: A *Generic Deployment Framework* to automatize deployment on grids
- Illustration: Deployment of *OpenCCM* middleware and *CORBA* Components-based applications on *Grid5000*
- **Evaluation & Performances**
- Conclusion & Perspectives

- Execution time of the deployment process of *OpenCCM* middleware on *Grid5000* / number of nodes

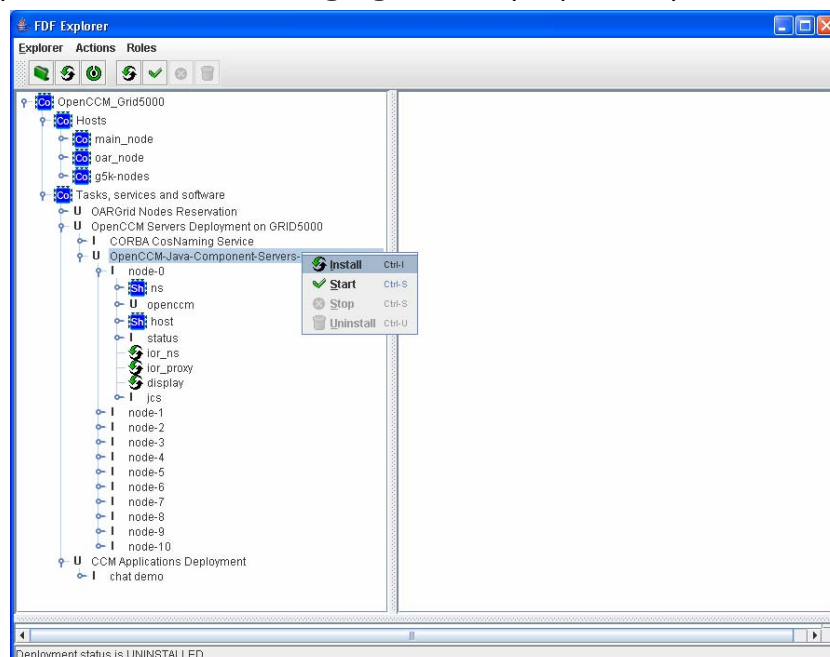


- Results:
 - Large scale deployment
 - Automatic deployment of OpenCCM application servers on 1000 physical nodes of Grid5000 (many G5K sites)
 - Execution time of the whole deployment process (deployment time)
 - Deployment time < 300 seconds for 1000 nodes!
 - Measures from 10 to 1000 real nodes and 4500 "virtual" nodes (many application servers on a same physical node)
 - Linear graph until 4500 servers

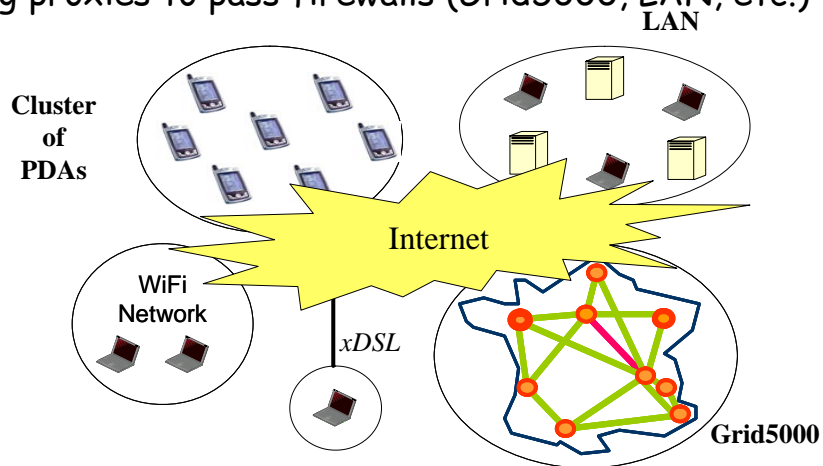
- Introduction, problem, scenario
- Our Proposition: A Generic Deployment Framework to automatize deployment on grids
- Illustration: Deployment of OpenCCM middleware and CORBA Components-based applications on Grid5000
- Evaluation & Performances
- **Conclusion & Perspectives**

- FDF: a generic deployment framework that simplifies large scale deployment of software and distributed applications
 - Independent of technologies, granularity and targeted platforms
 - Automatization of heterogeneous deployment tasks
 - Abstraction of software dependencies and execution order of deployment tasks
 - Extensible (personalities, deployment components implementation)
 - End-users simple description language
 - Execution policies of deployment process choice by user (parallel, sequential...)
- Applying the component-based approach (Fractal model)
 - Reification of the deployment process
- Successfully tested with other technologies/software
 - Deployment of J2EE (JOnAS) and PeTALS (JBI components) servers

- Monitoring / administration
 - Fractal Explorer console managing the deployment process



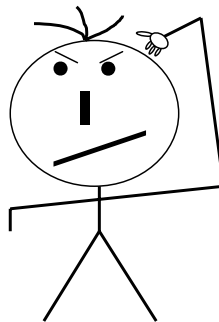
- Distribute FDF for very very large scale deployment
 - Using FDF to deploy FDF on multiple nodes
- Local resources control during deployment process
 - Threads, sockets...
- Deployment on heterogeneous platforms
 - Using proxies to pass firewalls (Grid5000, LAN, etc.)



© A. Flissi

37

???



© A. Flissi

38