DeployWare: A Framework for Automatic Deployment of Software Systems on Grids

A. Flissi, J. Dubus, N. Dolet and P. Merle
GOAL/ADAM Team - INRIA & LIFL-CNRS UMR 8022

Third EGEE User Forum, Clermont-Ferrand, FRANCE

12 FEB. 2008

Outline

Deployment of distributed software systems on Grids

Definition

- Deployment = set of tasks to orchestrate in a defined order
  - Installation/uninstallation of software, middleware, libraries, binaries on remote nodes,
  - Configuration of software and environment,
  - Starting/stopping of application servers, processes, daemons...
  - Instanciation of applications, programs...
  - etc.

Deployment of distributed software systems on Grids

Complex activity

- Clusters and grids: thousands of nodes
- Automatization is necessary
- Users/deployers not necessarily computer science researchers or engineers!

Deployment of distributed software systems on Grids

Complex activity

- Clusters and grids: thousands of nodes
- Automatization is necessary
- Users/deployers not necessarily computer science researchers or engineers!

Many challenges raised

- Complexity:
  - software dependencies, orchestration & administration
- Hardware and software heterogeneity
- Scalability

Complexity

Software dependencies

- Deployment of `software B` depends on `software A`
- Starting of `software B` depends on *installation* and *configuration* of `software A`

Orchestration of deployment tasks

- `software A` deployed before `software B`,
- {`software A`, {`software B`}} sequentially deployed, {`software B`} in parallel
- `software B` to start after the *installation* of `software A`

Complexity

Administration of large scale infrastructures

- Monitoring
- Management of the deployed system?

Hardware and Software Heterogeneity

Heterogeneity of the software to deploy

- **Different paradigms**:
  - parallel programming,
  - component-based or aspect-based software engineering,
  - object-oriented, etc.

Hardware and Software Heterogeneity

Heterogeneity of the software to deploy

- **Different paradigms**:
  - parallel programming,
  - component-based or aspect-based software engineering,
  - object-oriented, etc.
- **Plethora of runtime platforms / middleware**:
  - SOA, JEE, CCM, Fractal systems,
  - middleware for grids : GridCCM, Globus, ProActive...
  - MPI distributions: LAM-MPI, MPICH2, etc.

Hardware and Software Heterogeneity

Heterogeneity of the software to deploy

- **Different paradigms**:
  - parallel programming,
  - component-based or aspect-based software engineering,
  - object-oriented, etc.
- **Plethora of runtime platforms / middleware**:
  - SOA, JEE, CCM, Fractal systems,
  - middleware for grids : GridCCM, Globus, ProActive...
  - MPI distributions: LAM-MPI, MPICH2, etc.
- **Granularity** of software :
  - middleware, application servers,
  - software components or objects,
  - libraries or binaries,
  - virtual operating systems!

Hardware and Software Heterogeneity

Heterogeneity of the physical infrastructure/grid

- hardware,
- network,
- operating systems/architecture,
- *shells* : Bourne-Shell, C-Shell, Windows MS-DOS, etc.,
- *low-level* deployment mechanisms :
  - remote access protocols: SSH, Telnet, rlogin,
  - filetransfer protocols: FTP, HTTP, SCP

Scalability

Very very large scale

- Grid infrastructures composed of thousands of nodes
- Interconnexion between grids (*e.g.* French grid Grid'5000 & Japanese intitiative NAREGI)

Outline

Deploy...what?

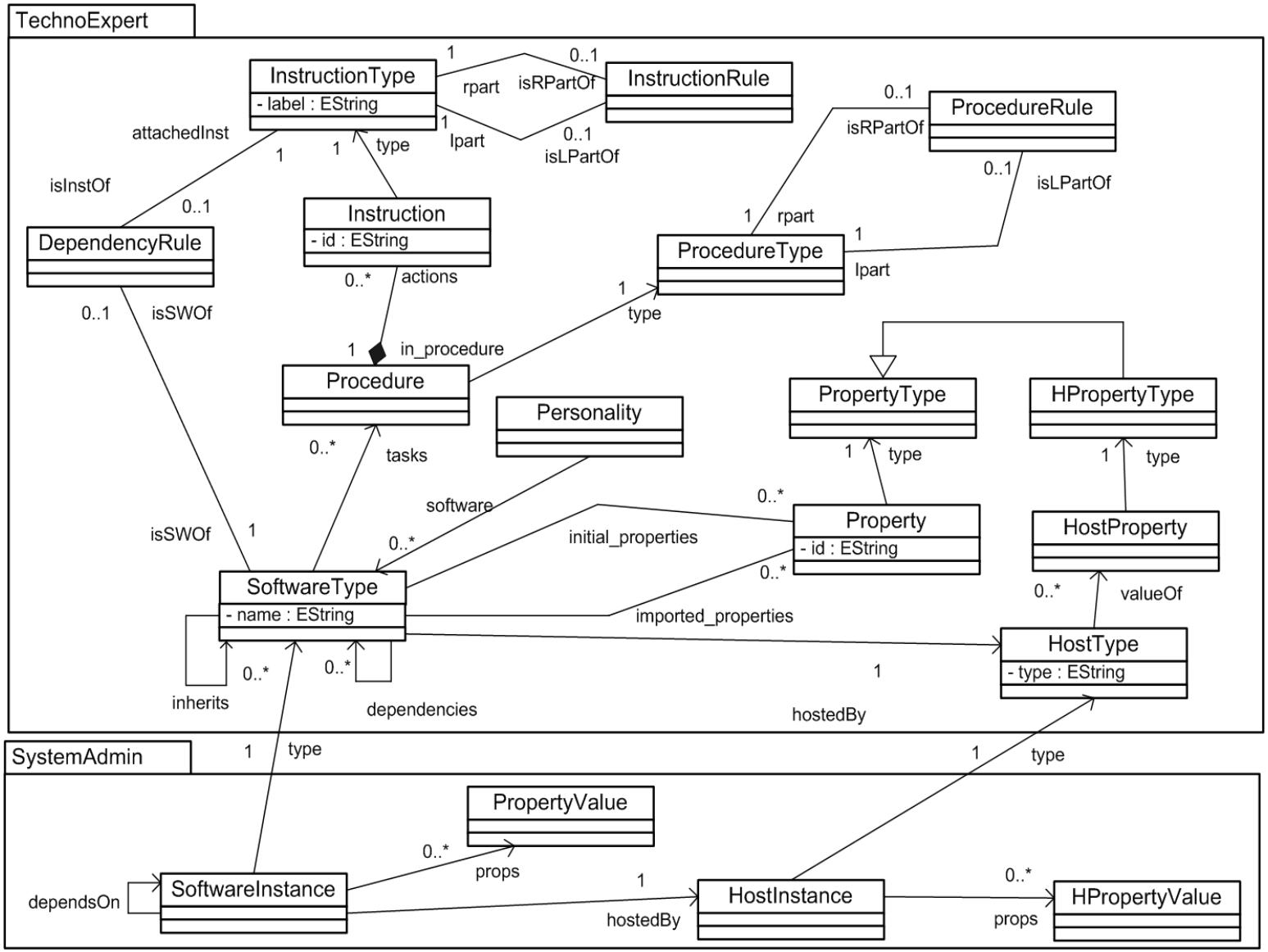
What is DeployWare?

- A framework to describe, deploy and manage heterogeneous and distributed software systems, based on:
  - A metamodel that captures abstract concepts of the deployment
  - A virtual machine (`FDF`), a library of deployment low-level components
  - A graphical user interface (`DeployWare eXplorer`)

The DeployWare Metamodel

Goal

- Capture abstract concepts of the deployment
  - Personality, software entity, dependency, procedure, instruction, (physical) node...
- Regardless paradigm/technology/granularity
- A concrete syntax to describe complex systems to deploy


TechnoExpert
InstructionType
- label : EString
InstructionRule
rpart
isRPartOf
lpart
isLPartOf
attachedInst
type
isInstOf
DependencyRule
Instruction
- id : EString
actions
ProcedureRule
isRPartOf
isLPartOf
ProcedureType
rpart
lpart
isSWOf
in_procedure
Procedure
tasks
Personality
PropertyType
HPropertyType
type
Property
- id : EString
HostProperty
initial_properties
software
SoftwareType
- name : EString
imported_properties
valueOf
HostType
- type : EString
inherits
dependencies
hostedBy
SystemAdmin
PropertyValue
props
SoftwareInstance
dependsOn
HostInstance
hostedBy
HPropertyValue
props

The DeployWare Metamodel

Approach

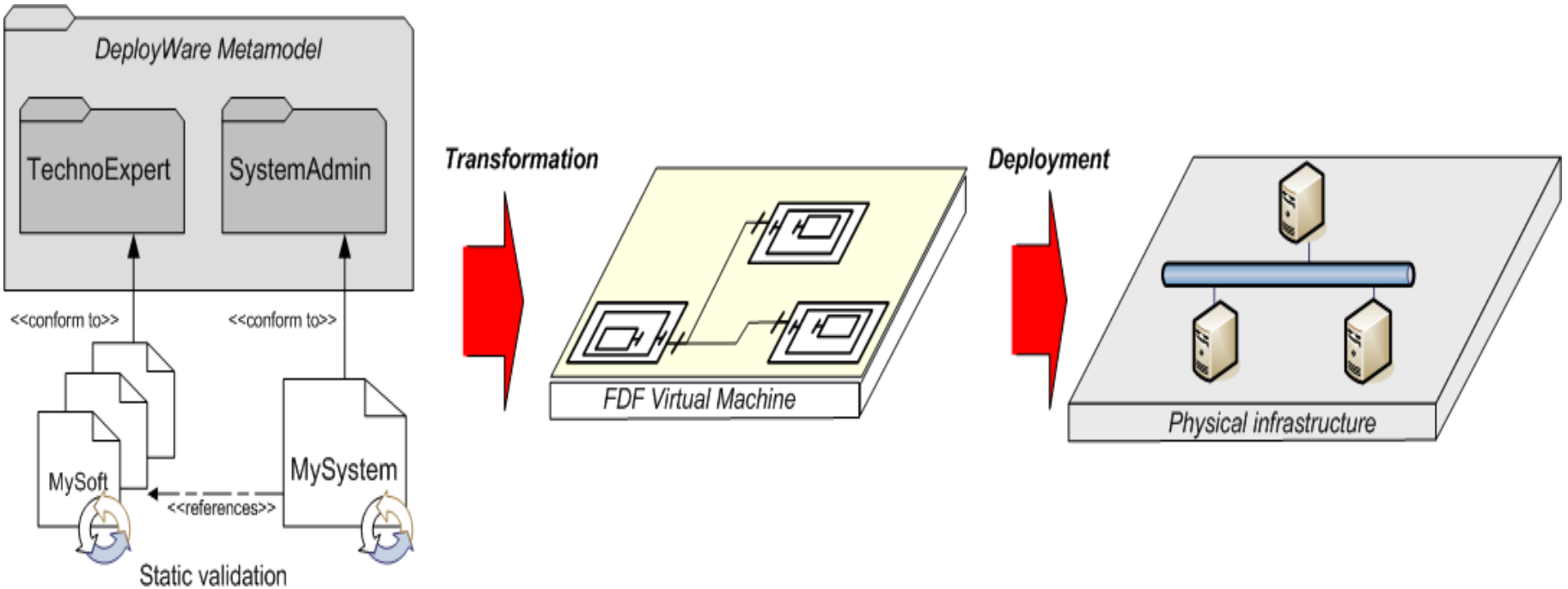
- Transformation of DeployWare models to concrete descriptions (*FDF* syntax)
  - Model-driven approach (*i.e.* 'à la' OMG MDA)
- DeployWare descriptions are based on an ADL (subset of Fractal ADL)
  - Generation of files for each personality
  - Final end-user description contains references to ID of these personalities

The Virtual Machine

FDF (*Fractal Deployment Framework*)

- **Goal**:
  Automatic execution of deployment processes
- Deployments are described through DeployWare models / FDF language
- FDF Virtual Machine
  - An implementation of the execution platform of DeployWare (*i.e.* a PSM!)
  - Component-based framework

The Virtual Machine



Example: The LAM-MPI implem. (software expert)

```
1  MPI.LAM {
2    properties {
3        lamhosts-file = MPI.HOME(UNDEFINED);
4        archive = MPI.ARCHIVE(UNDEFINED);
5        home = MPI.HOME(UNDEFINED);
6    }
7    deployment {
8      install {
9        upload-archive = TRANSFER.Upload(#[archive],#[home]);
10       cd-destination = SHELL.ChangeDirectory(#[home]);
11       unarchive = SHELL.Unarchive(#[archive]);
12       config-lam = SHELL.Execute(./configure -without-fc --prefix=#[home]);
13       make = SHELL.Execute(make);
14       make-install = SHELL.Execute(make install);
15     }
16     configure {
17       file-hosts = TRANSFER.UploadGeneratedFile(#[lamhosts-file],#[home]);
18       set-bin-path = SHELL.AddPath(#[home]/bin);
19       config-ssi = SHELL.SetVariable(boot_rsh_ignore_stderr,1);
20     }
21     start {
22       start-rte = SHELL.Execute(#[home]/bin/lamboot -v [lamhosts-file]);
23     }
24     stop {
25       stop-lam-rte = SHELL.Execute(#[home]/bin/lamhalt);
26     }
27   }
```

An instance of MPI (end-user)

```
1  mpi_example {
2      lam-mpi = FDF.SEQUENTIAL-COLLECTION(LAM-MPI Deployment) {
3          mpi-rte = MPI.LAM {
4              archive = MPI.ARCHIVE(~/archives/fdf/lam-7.1.4.tar.gz);
5              home = MPI.HOME(/tmp/lam-7.1.4);
6              host = Hosts/node_0;
7              properties {
8                  lamhosts-file = MPI.HOME(~/config/lamhosts);
9              }
10         }
11         mpirun-spmd = MPI.RUN {
12             lam = lam-mpi/mpi-rte;
13             host = Hosts/node_0;
14             properties {
15                 np = software.Parameter(NP,64);
16                 prog = MPI.HOME(/tmp/lam-7.1.4/spmd);
17             }
18         }
19     }
20 }
```

About Scalability...

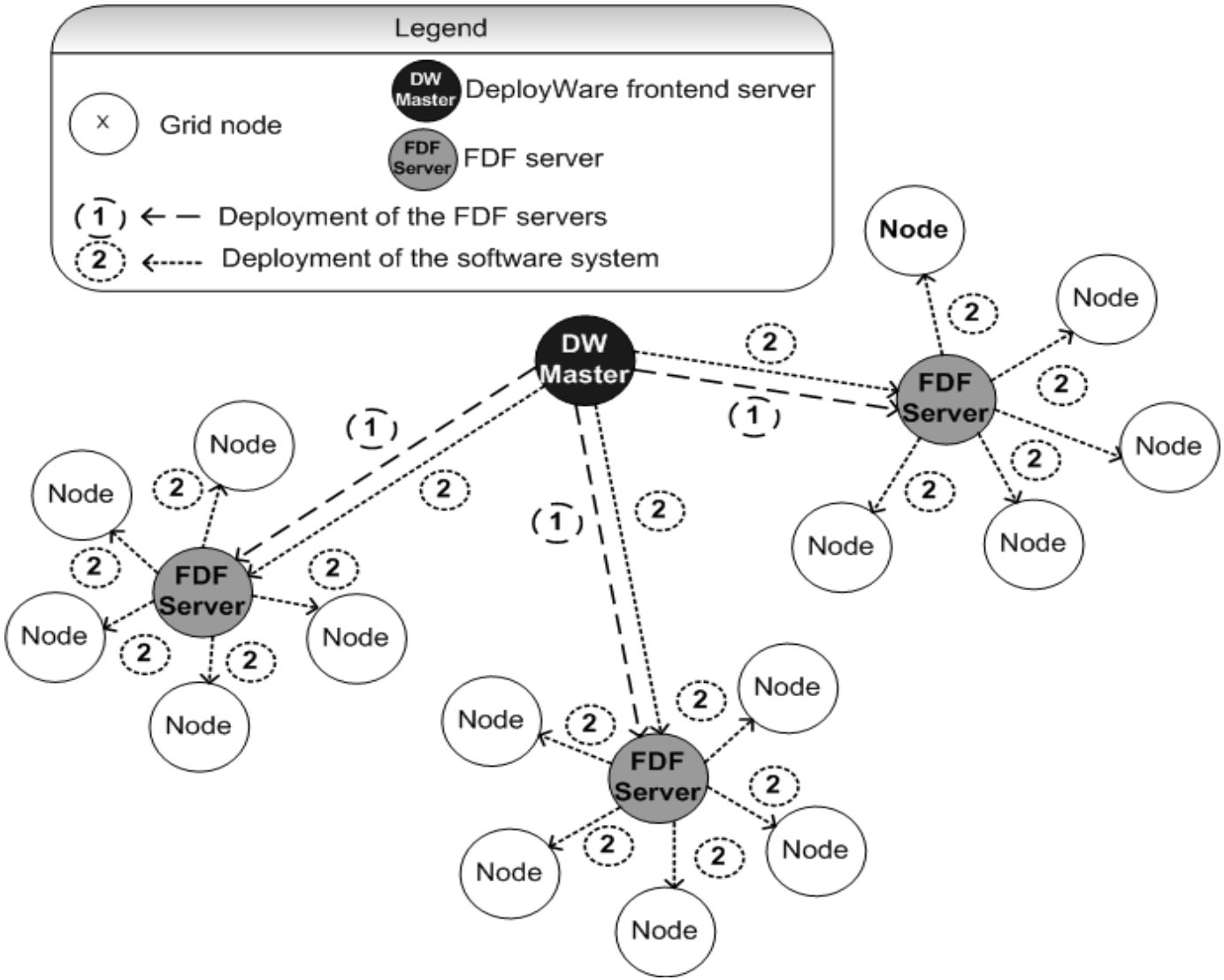
FDF Virtual Machine

- Deployed on one node
- Deploys on the other nodes

But...

- How to address very very large scale?
  - e.g. 5 000, 10 000 ou 50 000 nodes...
- Limits of physical resources (of one node)
  - Sockets, memory, CPU usage, number of threads, etc.

Distributed Deployment with DeployWare

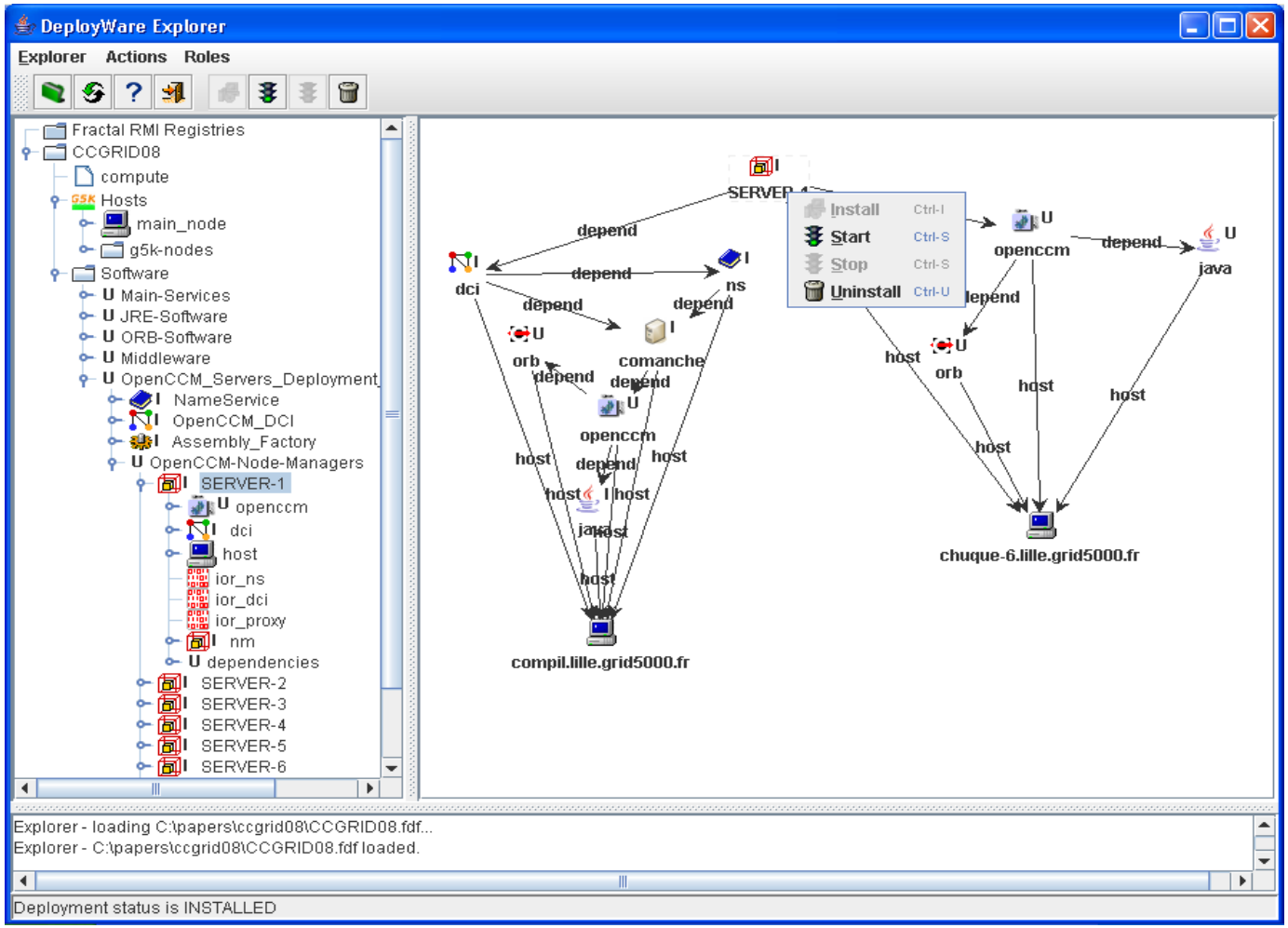


About Administration...

DeployWare eXplorer

- GUI to load and execute DeployWare descriptions
- Hierarchical view of the whole system, *i.e.* the nodes & software
- **Actions** on it
- **Visualize** dependencies between software

DeployWare eXplorer



Outline

Conclusion

DeployWare

- Automatic deployment of distributed, heterogeneous software systems on very large scale infrastructures
- DeployWare provides a metamodel that capture abstract concepts of the deployment
  - Describe any deployment process of a system
  - Regardless paradigm/technology/runtime platform and granularity of software
- FDF Virtual Machine to execute the deployment

Conclusion

Supported personalities

- CORBA-based systems, OpenCCM middleware and CCM applications,
- Tools for the grid (OAR/OARGrid),
- SOA systems (SCA, Apache Tuscany, BPEL processes, Orchestra & ActiveBPEL, PEtALS JBI),
- JEE systems (Apache Geronimo, JBoss, JOnAS, SUN GlassFish), autonomous systems (JASMINe/Jade)
- Fractal (Julia, Fractal ADL and RMI),
- Grid middleware (LAM-MPI)

Conclusion

Supported personalities

- Apache Tomcat, HTTPd daemon,
- Java Runtime Environment (JRE), JamVM,
- Apache Ant,
- OpenLDAP, SGBD MySQL,
- QEMU,
- Nexuiz (*doom-like* multi-players network game).

Perspectives

- Grid-specific middleware and applications
  - Globus Toolkit, ProActive, MPI...

Perspectives

- Grid-specific middleware and applications
  - Globus Toolkit, ProActive, MPI...
- **Looking for concrete experiences of the production grid!**
  - EGEE application?

This is the end... Thank you!

- Questions?

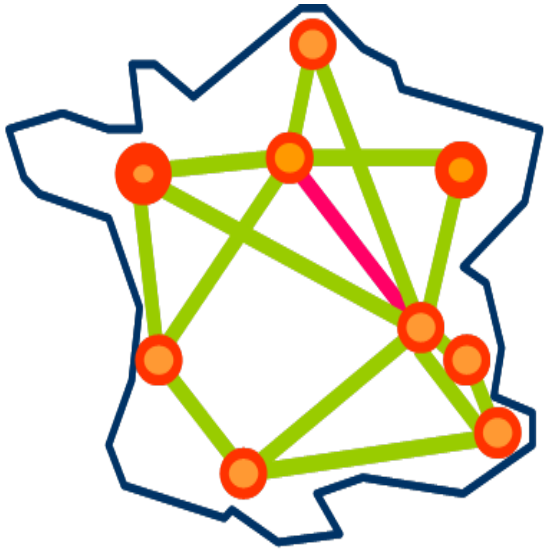
Download DeployWare/FDF

- http://fdf.gforge.inria.fr/

Experiences

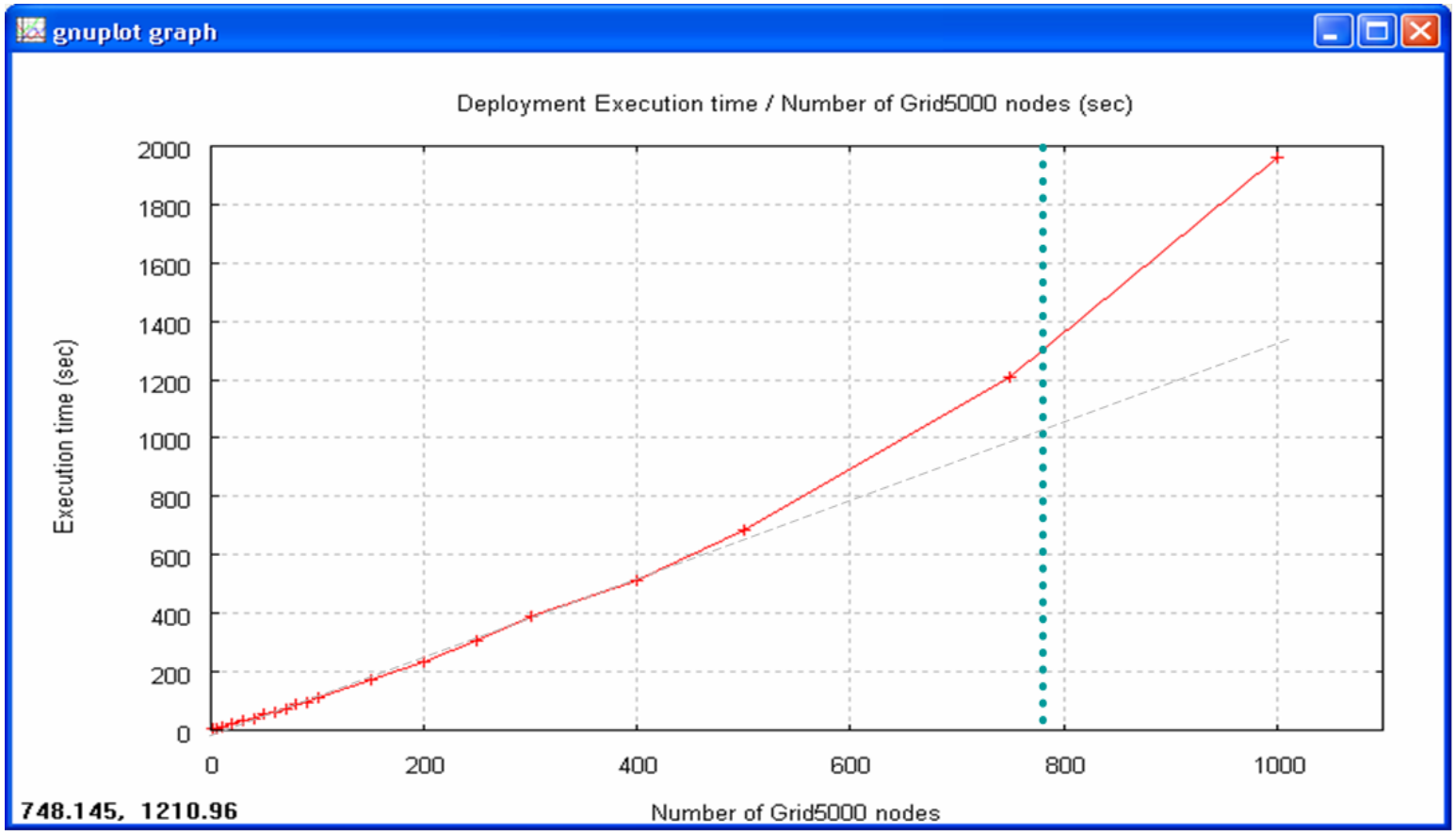
Deployment on the Grid'5000 platform

- 5000 processors distributed on 9 sites (many clusters/site)
- Deployment of the OpenCCM middleware: 4500 application servers on one thousand of nodes the grid



Performances

- (deployment process execution time) / (number of grid nodes)



Performances

- (deployment process execution time) / (number of grid nodes) / (number of FDF servers)