GoDIET: a deployment tool for distributed middleware on Grid'5000

Eddy Caron, Pushpinder Kaur Chouhan, Holly Dail

19 June 2006
ExpGrid Workshop

Outline

Outline

What is Deployment

A deployment is the mapping of a common platform and middleware across many resources.

- Software deployment maps and distributes a collection of software components on a set of resources. Software deployment includes activities such as releasing, configuring, installing, updating, adapting, de-installing, and even de-releasing a software system.
- System deployment involves two steps, physical and logical. In physical deployment all hardware is assembled (network, CPU, power supply etc), whereas logical deployment is organizing and naming whole cluster nodes as master, slave, etc.

Deployment tools

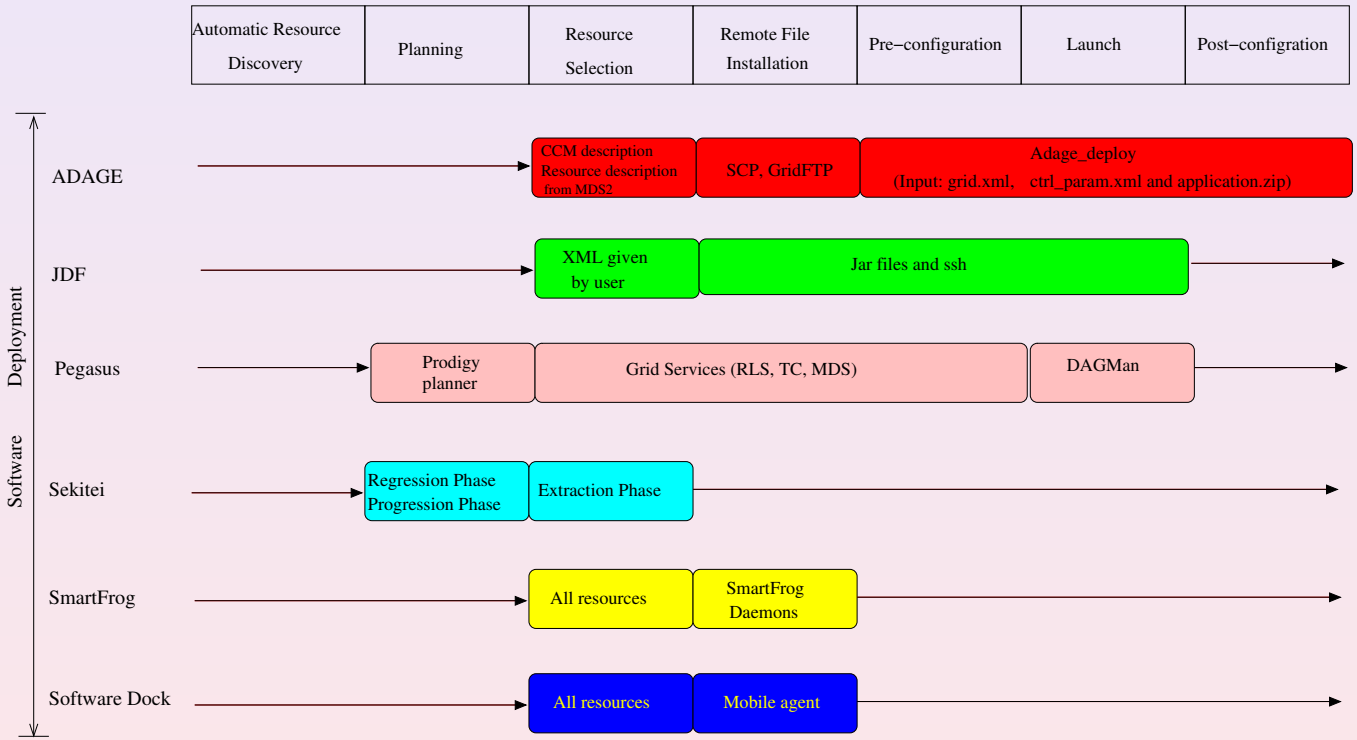
A tool that deploys middleware or an application on a cluster or grid according to a deployment plan is a deployment tool.
Deployment plans define:

- which resources should be used
- how resources should be connected
- path of binaries on resource

Deployment phases

- Resource discovery phase
- Deployment planning phase
- Resource selection phase
- Remote files installation phase
- Pre-configuration phase
- Launch phase
- Post-configuration phase

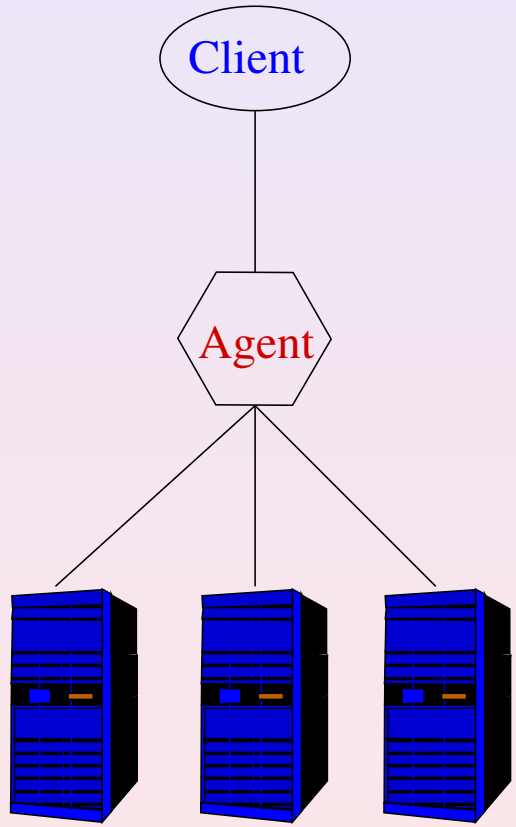
Some deployment tools



Outline

Distributed Interactive Engineering Toolbox

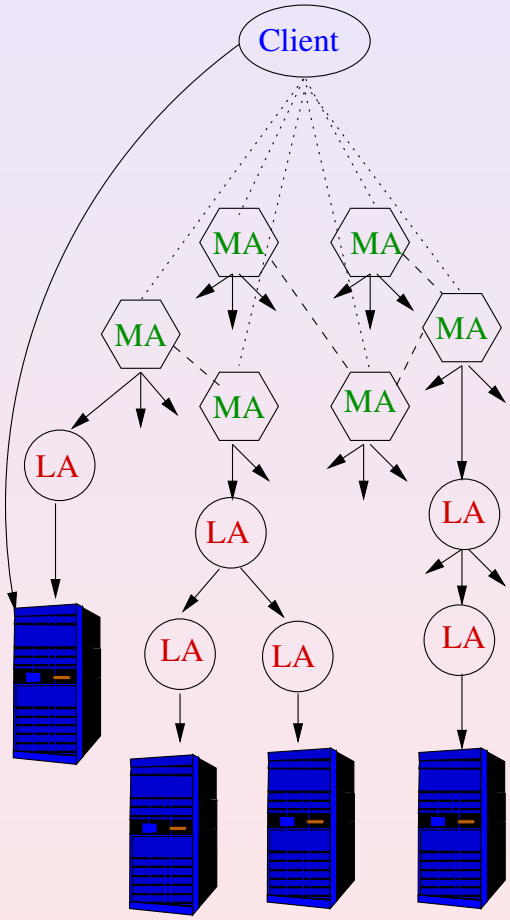
- Hierarchical architecture for improved scalability
- Distributed information in the tree
- Plug-in schedulers



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

Distributed Interactive Engineering Toolbox

- Hierarchical architecture for improved scalability
- Distributed information in the tree
- Plug-in schedulers



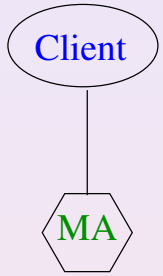
http://graal.ens-lyon.fr/DIET/

DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client

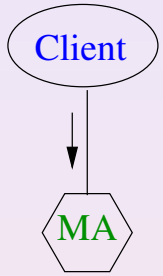
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



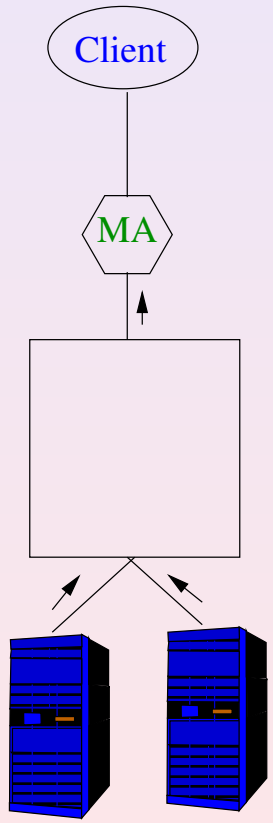
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



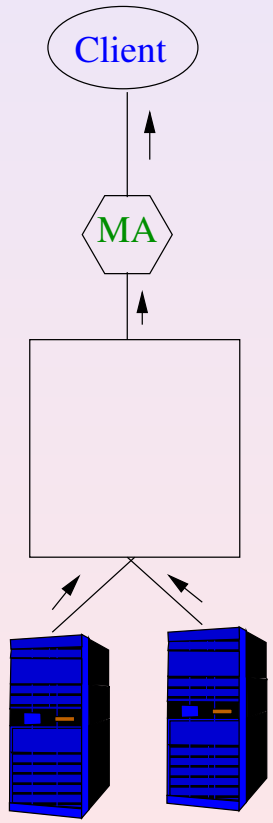
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



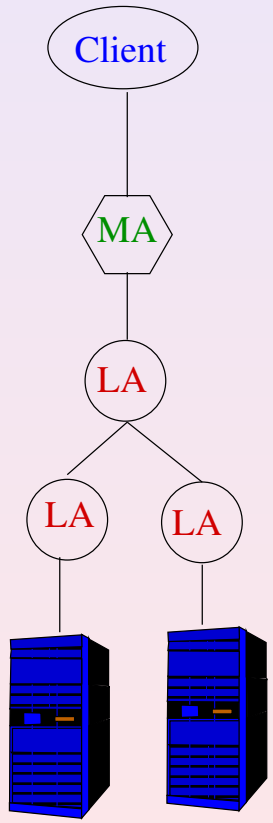
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



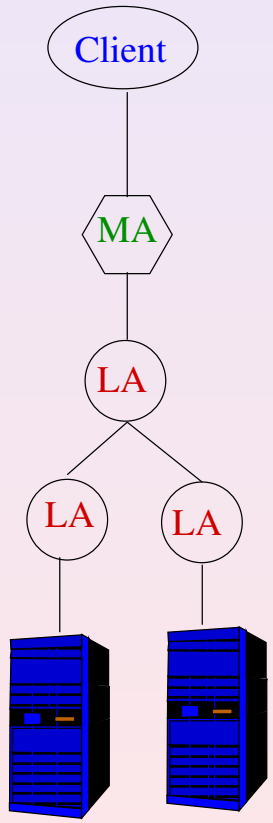
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



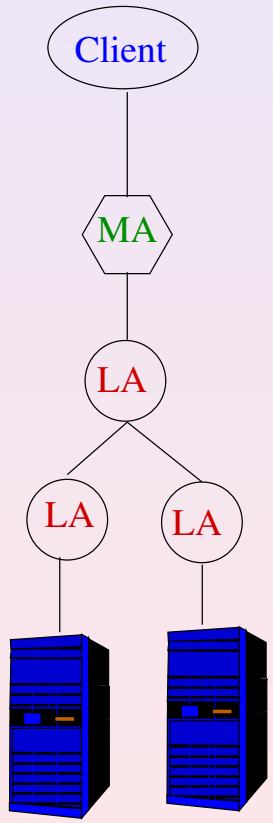
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



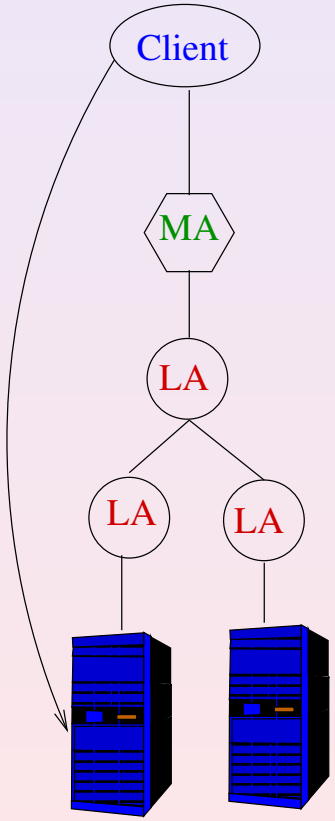
DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



DIET Components

- Client : An application that uses DIET to solve problems
- Master Agent (MA) :
  - Receives computation requests from clients
  - Collects computational abilities from servers and chooses the best one
  - Returns the reference of the chosen server to the client
- Local Agents (LA) :
  - Act as transmitter
  - Share the workload of scheduling
- Servers (SeD) : Perform actual computation for client



DIET Deployment

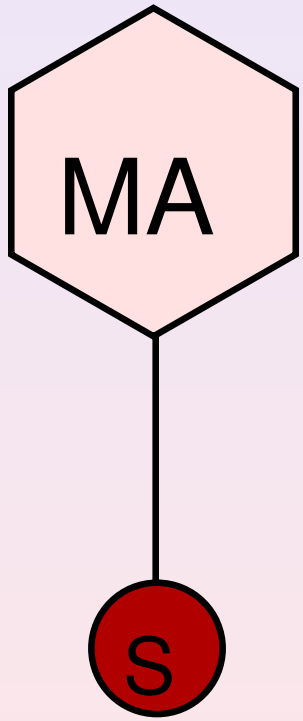
- Naming service is launched
-
-

DIET Deployment

- Naming service is launched
- MA is launched

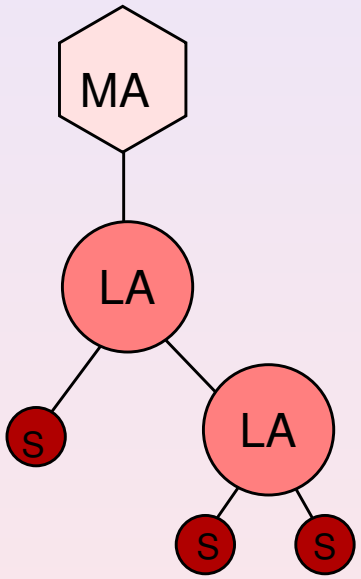
DIET Deployment

- Naming service is launched
- MA is launched
- Other DIET elements according to their place in the hierarchy



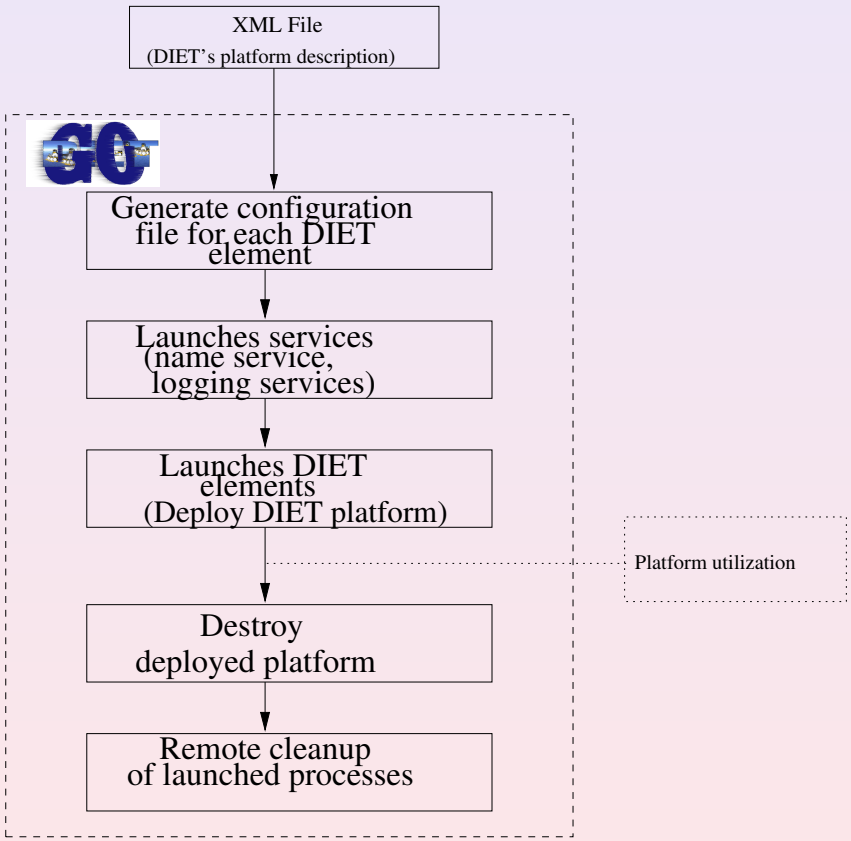
DIET Deployment

- Naming service is launched
- MA is launched
- Other DIET elements according to their place in the hierarchy



Outline

Working of GoDIET



GoDIET error detection

Identify errors in the deployment process and launches those elements that do not depend on the failed element.

Status of launched elements:

- confused - GoDIET can not be sure of the status of the element.
- none - unlaunched elements.
- failed - can not launch the element.
- running - correctly launched elements.

File transfer and task execution

`scp` and `ssh` to provide secure file transfer and task execution.

- GoDIET can configure environment variables
- Specify the binary to launch with appropriate command line parameters
- Specify different files for the stdout and stderr of the launched process
- GoDIET can retrieve the PID of the launched process
- This PID can then be used later for shutting down the DIET deployment

An example of GoDIET's XML file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE diet_configuration SYSTEM "GoDIET.dtd"
      >
<diet_configuration>
  <goDiet debug="1" saveStdOut="no"
          saveStdErr="no"
          useUniqueDirs="no" />
  <resources>
    <scratch dir="/homePath/scratch_godiet" />
    <storage> .. </storage>
          .
          .
    <cluster>  .. </cluster>
          .
          .
    <compute>  .. </compute>
  </resources>
  <diet_services> .. </diet_services>
  <diet_hierarchy> .. </diet_hierarchy>
</diet_configuration>
```

An example of GoDIET's XML file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE diet_configuration SYSTEM "GoDIET.dtd"
      >
<diet_configuration>
  <goDiet debug="1" saveStdOut="no"
          saveStdErr="no"
          useUniqueDirs="no" />
  <resources>
    <scratch dir="/homePath/scratch_godiet" />
    <storage> .. </storage>
          .
          .
    <cluster>  .. </cluster>
          .
          .
    <compute>  .. </compute>
  </resources>
  <diet_services> .. </diet_services>
  <diet_hierarchy> .. </diet_hierarchy>
</diet_configuration>
```

```
<storage label="g5kBordoDisk">
  <scratch dir="/homePath/user/
         scratch_runtime" />
  <scp server="frontale.bordeaux
         .grid5000.fr" login="
         pkchouhan" />
</storage>
.
.
<storage label="g5kOrsayDisk">
  <scratch dir="/homePath/user/
         scratch_runtime" />
  <scp server="frontale.orsay.
         grid5000.fr" login="
         pkchouhan" />
</storage>
```

An example of GoDIET's XML file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE diet_configuration SYSTEM "GoDIET.dtd"
    >
<diet_configuration>
  <goDiet debug="1" saveStdOut="no"
          saveStdErr="no"
          useUniqueDirs="no"/>
  <resources>
    <scratch dir="/homePath/scratch_godiet"/>
    <storage> .. </storage>
          .
          .
    <cluster> .. </cluster>
          .
          .
    <compute> .. </compute>
  </resources>
  <diet_services> .. </diet_services>
  <diet_hierarchy> .. </diet_hierarchy>
</diet_configuration>
```

```
<cluster label="g5kBordo" disk="
      g5kBordeauxDisk" login="
      pkchouhan">
  <env path="/homePath/user/demo
       /bin" LD_LIBRARY_PATH="
       /homePath/user/demo/lib
       "/>
  <node label="node-1.bordeaux.
       grid5000.fr">
    <ssh server="node-1.bordeaux
         .grid5000.fr"/>
  </node>
          .
          .
  <node label="node-47.bordeaux.
       grid5000.fr">
      <ssh server="node-47.
           bordeaux.grid5000.
           fr"/>
  </node>
</cluster>
      .
      .
<cluster label="g5kOrsay" ..>
      .
      .
</cluster>
```

An example of GoDIET's XML file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE diet_configuration SYSTEM "GoDIET.dtd"
      >
<diet_configuration>
  <goDiet debug="1" saveStdOut="no"
          saveStdErr="no"
          useUniqueDirs="no" />
  <resources>
    <scratch dir="/homePath/scratch_godiet" />
    <storage> .. </storage>
          .
          .
    <cluster>  .. </cluster>
          .
          .
    <compute>  .. </compute>
  </resources>
  <diet_services> .. </diet_services>
  <diet_hierarchy> .. </diet_hierarchy>
</diet_configuration>
```

```
<compute label="host1" disk="
       disk1">
   <ssh server="host1.site1.fr"
        login="<your_login>" />
   <env path="<bindir1>:<bindir2
        >:..."
       LD_LIBRARY_PATH="<
            libdir1>:<libdir2
           >:..." />
   <end_point contact="
        192.5.59.198" />
</compute>
     .
     .
<compute label="host2" disk="
       disk1">
    <ssh server="host2.site1.fr"
         />
    <env path="<bindir1>:<
         bindir2>:..."
        LD_LIBRARY_PATH="<
             libdir1>:<
             libdir2>:..." />
</compute>
```

An example of GoDIET's XML file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE diet_configuration SYSTEM "GoDIET.dtd"
      >
<diet_configuration>
  <goDiet debug="1" saveStdOut="no"
          saveStdErr="no"
          useUniqueDirs="no" />
  <resources>
    <scratch dir="/homePath/scratch_godiet" />
    <storage> .. </storage>
          .
          .
    <cluster>  .. </cluster>
          .
          .
    <compute>  .. </compute>
  </resources>
  <diet_services> .. </diet_services>
  <diet_hierarchy> .. </diet_hierarchy>
</diet_configuration>
```

```
<diet_services>
  <omni_names contact="gdx0005.
        orsay.grid5000.fr" port="
        2809">
      <config server="gdx0005.
            orsay.grid5000.fr"
            remote_binary="
            omniNames" />
  </omni_names>
  <log_central>
      <config server="gdx0007.
            orsay.grid5000.fr"
            remote_binary="
            LogCentral" />
  </log_central>
  <log_tool>
      <config server="gdx0007.
            orsay.grid5000.fr"
            remote_binary="
            DIETLogTool" />
  </log_tool>
</diet_services>
```

An example of GoDIET's XML file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE diet_configuration SYSTEM "GoDIET.dtd"
      >
<diet_configuration>
  <goDiet debug="1" saveStdOut="no"
          saveStdErr="no"
          useUniqueDirs="no" />
  <resources>
    <scratch dir="/homePath/scratch_godiet" />
    <storage> .. </storage>
          .
          .
    <cluster>  .. </cluster>
          .
          .
    <compute>  .. </compute>
  </resources>
  <diet_services> .. </diet_services>
  <diet_hierarchy> .. </diet_hierarchy>
</diet_configuration>
```

```
<diet_hierarchy>
  <master_agent label="MA1">
    <config server="node-5.
            toulouse.grid5000.fr"
            remote_binary="
            dietAgent" />
    <local_agent label="LA1">
      <config server="node-75.
              sophia.grid5000.fr"
              remote_binary="
              dietAgent" />
      <SeD label="SeD1">
          <config server="node-65.
                  sophia.grid5000.
                  fr" remote_binary
                  =" BLASserver" />
      </SeD>
  </master_agent>
</diet_hierarchy>
```

Outline

Experimental Platform

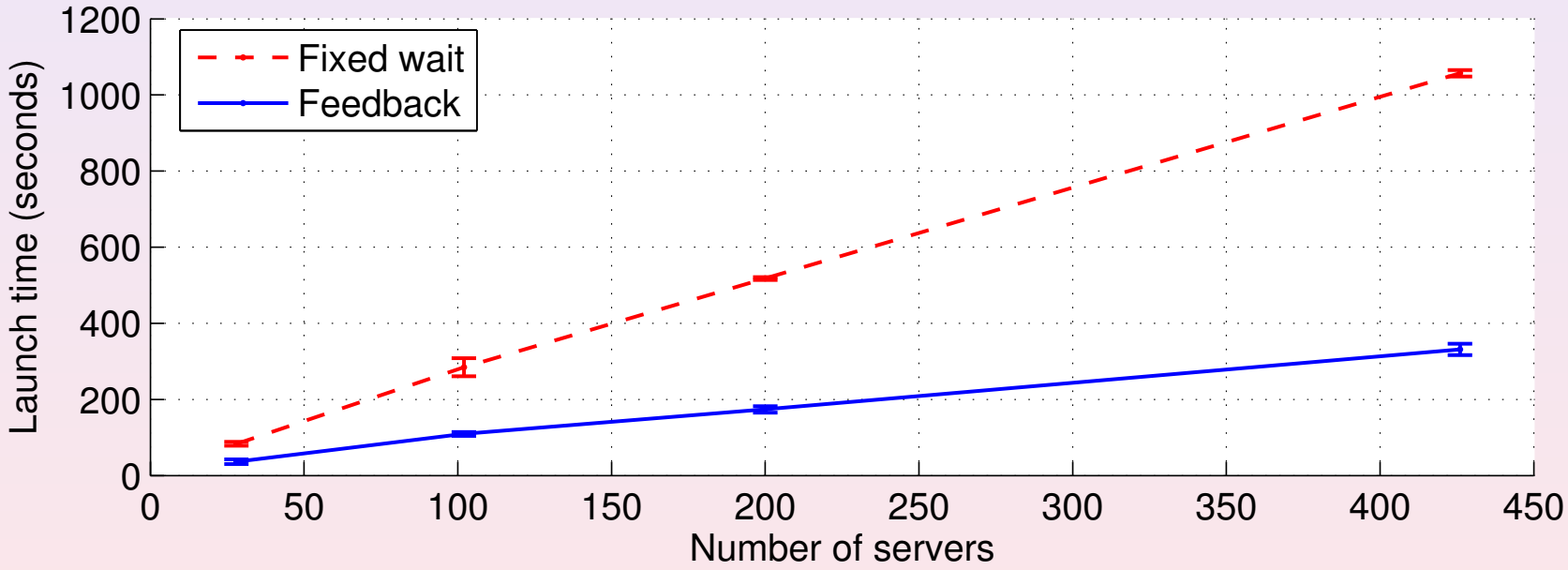
Site Cluster	Nodes	Memory	Processor Type
Bordeaux	48	2 GB	dual AMD Opteron 248 2.2GHz
Lille	53	4 GB	dual AMD Opteron 248 2.2GHz
Lyon	56	2 GB	dual AMD Opteron 2.0
Orsay	216	2 GB	dual AMD Opteron 246 2.0GHz
Rennes Paraci	64	2 GB	dual Intel Xeon 2.4 GHz
Sophia	138	2 GB	dual AMD Opteron 246 2.0GHz
Toulouse	57	2 GB	dual AMD Opteron 248 2.2GHz
Total nodes	632		

Table: Description of the Grid'5000 clusters used in our experiments.

Two launch approaches

- Fixed wait - sleeping for a fixed period before launching dependent components.
- Feedback - This approach uses real-time feedback from LogService to guide the launch process.

Two launch approaches

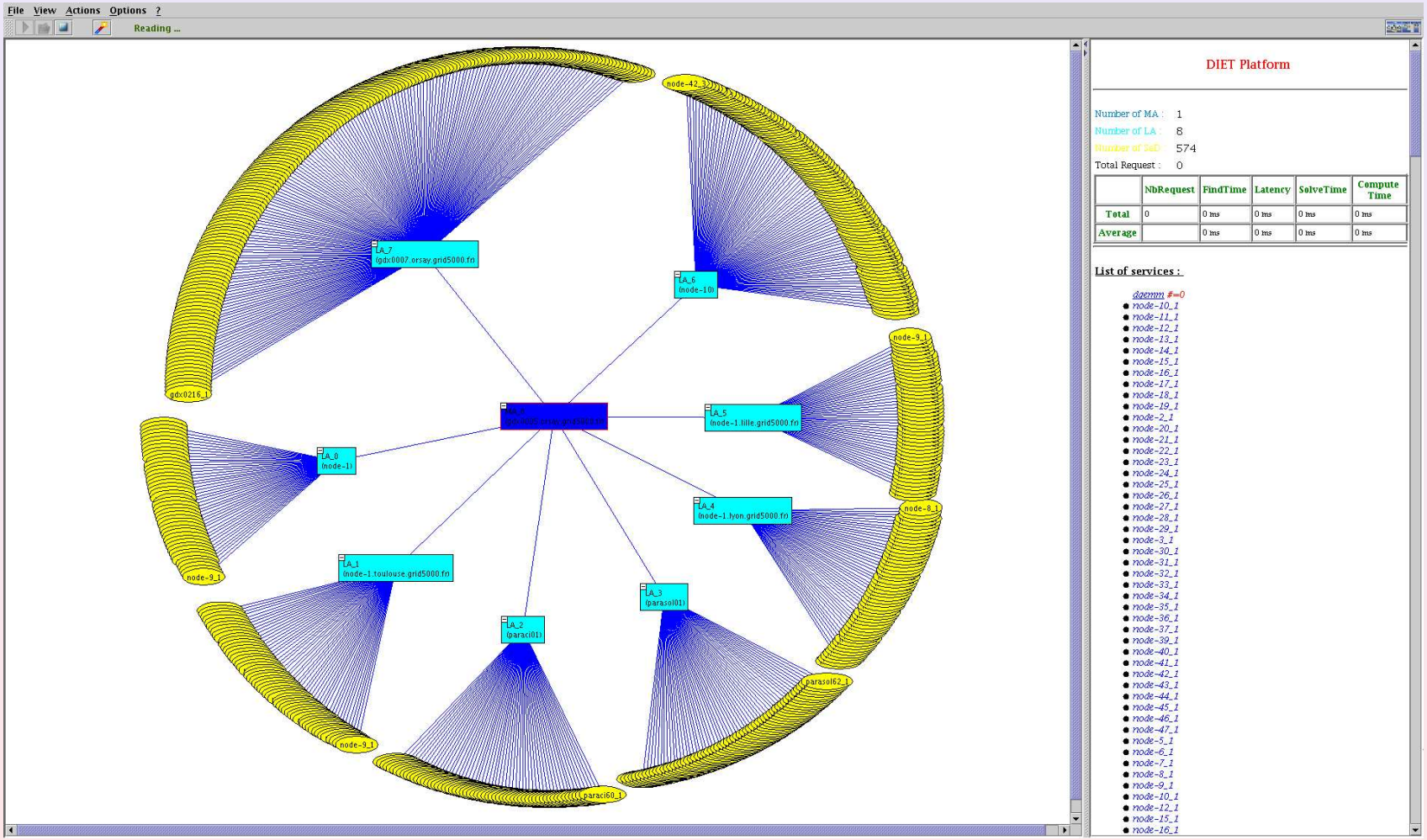


Launch problem identification

Approach	Sites	Rep. 1	Rep. 2	Rep. 3
Fixed wait	1	0/0	0/0	0/0
	3	0/0	0/0	1/40
	5	1/50	0/49	0/39
	7	2/91	1/129	-
Feedback	1	0/0	0/0	0/0
	3	1/1	0/0	0/0
	5	0/0	0/0	0/0
	7	2/30	1/29	-

Table: Problem SeDs as identified by GoDIET / by clients.

DIET deployment on Grid'5000



Grid'5000 DIET Deployment with 1 MA, 8 LA, 574 SeD

Cluster	SeDs	Launch time (secs)
Bordeaux	44	70
Lille	44	118
Lyon	49	52
Orsay	176	88
Paraci	59	82
Parasol	59	81
Sophia	89	68
Toulouse	54	33

Table: Time taken to launch 574 SeDs. Time to deploy platform is 592seconds.

Outline

Conclusion

- GoDIET, a tool for hierarchical deployment
- DIET, hierarchical Problem Solving Environments
- Experiments on Grid'5000 to prove the efficiency of GoDIET

Future work

- Visual tool, graphical model of desired deployment
- An interactive user tool for automated deployment planning and execution
- Integrate this tool and deployment planning models