DRM4G: an open source framework for distributed computing

Carlos Blanco, A. S. Cofiño

Santander Meteorology Group
Department of Applied Mathematics and Computer Sciences
Universidad de Cantabria, Santander, Spain

Aknowledgments: This work is funded by the Spanish PLAN NACIONAL de I+D+i 2008-2011 (Ref.# CGL2011-28864, WRF4G) and ERDF
Outline

• Statement of the problem
• Requirements
• DRM4G
• How does it work?
• Application example: WRF4G
• Conclusion
Statement of the problem
Statement of the problem

PC
Blizzard

SSH

Workstation Oceano
UC

Supercomputer Altamira
RES Infrastructure

marcream01.in2p3.fr
ce130.cern.ch

Grid
EGI (VO esr)

carlos

SSH

SSH+PROXY

PC
Blizzard

gridgate.cs.tcd.ie
Statement of the problem

10 JOBS
Workstation Oceano
UC

10 JOBS
Supercomputer Altamira
RES Infrastructure

30 JOBS

10 JOBS
Grid
EGI (VO esr)

CARLOS
Blizzard
SSH
SSH+PROXY

marcreem01.in2p3.fr
ce130.cern.ch
gridgate.cs.tcd.ie
Statement of the problem

- **Workstation Oceano UC**
- **Supercomputer Altamira RES Infrastructure**
- **CARLOS Blizzard**
- **SSH**
- **SSH+PROXY**
- **20,000 JOBS**
- **marcream01.in2p3.fr**
- **ce130.cern.ch**
- **gridgate.cs.tcd.ie**
- **Grid EGI (VO esr)**
- **? JOBS**
Statement of the problem

20,000 JOBS

Workstations Oceano, Sea, ...
UC

Supercomputer Altamira, MN3, ...
RES Infrastructure

Grid
EGI (VO esr),
GISELA (VO prod.vo.eu-eela)
Outline

- Statement of the problem
- Requirements
- DRM4G
- How does it work?
- Application example: WRF4G
- Conclusion
Requirements

- Uniform access to available resources
- Simple interface
- Robust and scalable
- Easy configuration
- Easy installation (batteries included)
• Statement of the problem
• Requirements
• DRM4G
• How does it work?
• Application example: WRF4G
• Conclusion
Key features

- Based on **GridWay** meta-scheduler
- Written in **python** and **C**
- **CLI** interface (one command) and **python API**
- Tools to manage **identities** (private/public keys and grid certificates)
- **Scalable** ~100,000 jobs
- **Ready-to-run** (Linux)
• **CLI** offers users a command to submit, cancel, and monitor jobs and configure resources.
• **CLI** offers users a command to submit, cancel, and monitor jobs and configure resources.

• **GridWay core** is in charge of job execution and resource brokering.
• **CLI** offers users a command to submit, cancel, and monitor jobs and configure resources.

• **GridWay core** is in charge of job execution and resource brokering.

• **Sched** is responsible for scheduling jobs.
• **CLI** offers users a command to submit, cancel, and monitor jobs and configure resources.

• **GridWay core** is in charge of job execution and resource brokering.

• **Sched** is responsible for scheduling jobs.

• Middleware Access Driver (MAD):
  - Resource Managers:
    - FORK, SGE, PBS SLURM, CREAM, GLOBUS, ...
  - Communicators:
    - LOCAL, SSH, GSISSH
Outline

• Statement of the problem
• Requirements
• DRM4G
• How does it work?
• Application example: WRF4G
• Conclusion
Quick start

$ bash -c "$(wget -O- https://meteo.unican.es/work/DRM4G/install.sh)"

DRM4G installation script

This script will install DRM4G 2.2.0

--> Downloading drm4g-2.2.0-x86_64.tar.gz ...

--> Unpacking drm4g-2.2.0-x86_64.tar.gz in directory /home/user ...

Installation of DRM4G is done!

In order to work with DRM4G you have to enable its environment with the command:

. /home/user/drm4g/bin/drm4g_init.sh

You need to run the above command on every new shell you open before using DRM4G, but just once per session.
Quick start

```bash
$ bash -c "$(wget -O- https://meteo.unican.es/work/DRM4G/install.sh)"
```

DRM4G installation script

This script will install DRM4G 2.2.0

--> Downloading drm4g-2.2.0-x86_64.tar.gz ...

--> Unpacking drm4g-2.2.0-x86_64.tar.gz in directory /home/user ...

Installation of DRM4G is done!

In order to work with DRM4G you have to enable its environment with the command:

```
. /home/user/drm4g/bin/drm4g_init.sh
```

You need to run the above command on every new shell you open before using DRM4G, but just once per session.
Quick start

```bash
$ bash -c "$(wget -O- https://meteo.unican.es/work/DRM4G/install.sh)"
```

DRM4G installation script

This script will install DRM4G 2.2.0

--> Downloading drm4g-2.2.0-x86_64.tar.gz ...

--> Unpacking drm4g-2.2.0-x86_64.tar.gz in directory /home/user ...

Installation of DRM4G is done!

In order to work with DRM4G you have to enable its environment with the command:

```bash
./home/user/drm4g/bin/drm4g_init.sh
```

You need to run the above command on every new shell you open before using DRM4G, but just once per session.
Starting

$ . /home/user/drm4g/bin/drm4g_init.sh

$ drm4g start
Creating a DRM4G local configuration in '/home/user/.drm4g'
Creating '/home/user/.drm4g/var' directory
Coping from '/home/user/drm4g/etc' to '/home/user/.drm4g/etc'
Starting DRM4G ....
OK

$ drm4g status
DRM4G is running

$ drm4g resource edit
Resource Configuration I

resources.conf

[DEFAULT]
enable = True
communicator = local
frontend = localhost
lrms = fork

[blizzard]
max_jobs_running = 1

[altamira]
communicator = ssh
username = user
frontend = altamiral1.ifca.es
private_key = ~/.ssh/id_rsa
lrms = slurm_res
max_jobs_running = 40
max_jobs_in_queue = 50
## Resource Configuration II

### resources.conf

<table>
<thead>
<tr>
<th>Group</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[oceano]</td>
<td>communicator</td>
<td>ssh</td>
</tr>
<tr>
<td></td>
<td>username</td>
<td>user</td>
</tr>
<tr>
<td></td>
<td>frontend</td>
<td>oceano.unican.es</td>
</tr>
<tr>
<td></td>
<td>private_key</td>
<td>~/.ssh/id_rsa</td>
</tr>
<tr>
<td></td>
<td>max_jobs_running</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[egi_esr]</td>
<td>communicator</td>
<td>ssh</td>
</tr>
<tr>
<td></td>
<td>username</td>
<td>user</td>
</tr>
<tr>
<td></td>
<td>frontend</td>
<td>ui.macc.unican.es</td>
</tr>
<tr>
<td></td>
<td>private_key</td>
<td>~/.ssh/id_rsa</td>
</tr>
<tr>
<td></td>
<td>lrms</td>
<td>cream</td>
</tr>
<tr>
<td></td>
<td>vo</td>
<td>esr</td>
</tr>
</tbody>
</table>
$ drm4g resource egi_esr id --conf \ 
  --grid-cred=~/cerd.p12 \ 
  --public-key=~/.id_rsa.pub \ 
  --lifetime=168

--> Configuring private and public keys ...
Enter passphrase for key '/home/user/.ssh/id_rsa':
Identity added: /home/user/.ssh/id_rsa (/home/user/.ssh/id_rsa)
Adding 'id_rsa.pub' to 'authorized_keys' on 'ui.macc.unican.es' ...
Lifetime set to 604800 seconds

--> Configuring grid certificate ...
Converting 'cerd.p12' key to pem format ...
Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Converting 'cerd.p12' certificate to pem format ...
Enter Import Password:
Copying 'userkey.pem' to 'ui.macc.unican.es' ...
Copying 'usercert.pem' to 'ui.macc.unican.es' ...
$ drm4g resource egi_esr id --conf \
--grid-cerd=~/cerd.p12 \ 
--public-key=~/id_rsa.pub \ 
--lifetime=168

--> Configuring private and public keys ...
Enter passphrase for key '/home/user/.ssh/id_rsa':
Identity added: /home/user/.ssh/id_rsa (/home/user/.ssh/id_rsa)
Adding 'id_rsa.pub' to 'authorized_keys' on 'ui.macc.unican.es' ...
Lifetime set to 604800 seconds

--> Configuring grid certificate ...
Converting 'cerd.p12' key to pem format ...
Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Converting 'cerd.p12' certificate to pem format ...
Enter Import Password:
Copying 'userkey.pem' to 'ui.macc.unican.es' ...
Copying 'usercert.pem' to 'ui.macc.unican.es' ...
$ drm4g resource egi_esr id --conf \
--grid-cerd=~/cerd.p12 \
--public-key=~/id_rsa.pub \
--lifetime=168

--> Configuring private and public keys ...
Enter passphrase for key '/home/user/.ssh/id_rsa':
Identity added: /home/user/.ssh/id_rsa (/home/user/.ssh/id_rsa)
Adding 'id_rsa.pub' to 'authorized_keys' on 'ui.macc.unican.es' ...
Lifetime set to 604800 seconds

--> Configuring grid certificate ...
Converting 'cerd.p12' key to pem format ...
Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Converting 'cerd.p12' certificate to pem format ...
Enter Import Password:
Copying 'userkey.pem' to 'ui.macc.unican.es' ...
Copying 'usercert.pem' to 'ui.macc.unican.es' ...
$ cat wrf4g.jt

EXECUTABLE   = /bin/bash ./wrfg4.sh
INPUT_FILES  = WRF4G-1.5.1-x86_64.tar.gz, wrfg4.sh
REQUIREMENTS = ARCH = 'x86_64'

$ drm4g job submit wrf4g.jt
Job id:  0

$ drm4g job list 0

JID  EM   START    END    EXIT NAME                 HOST
0 actv 18:05:33 --:--:-- --  wrf4g.jt  esr::sbgce2.in2p3.fr

$ drm4g job cancel 0
$ cat wrf4g.jt

EXECUTABLE = /bin/bash ./wrfg4.sh
INPUT_FILES = WRF4G-1.5.1-x86_64.tar.gz, wrfg4.sh
REQUIREMENTS = ARCH = 'x86_64'

$ drm4g job submit wrf4g.jt

Job id: 0

$ drm4g job list 0

JID EM START END EXIT NAME HOST
0 actv 18:05:33 --:--:-- -- wrf4g.jt esr::sbgce2.in2p3.fr

$ drm4g job cancel 0
$ cat wrf4g.jt

EXECUTABLE   = /bin/bash ./wrfg4.sh
INPUT_FILES  = WRF4G-1.5.1-x86_64.tar.gz, wrfg4.sh
REQUIREMENTS = ARCH = 'x86_64'

$ drm4g job submit wrf4g.jt

Job id: 0

$ drm4g job list 0

JID EM START END EXIT NAME HOST
0   actv 18:05:33 --:--:-- --  wrf4g.jt  esr::sbgce2.in2p3.fr

$ drm4g job cancel 0
$ cat wrf4g.jt

EXECUTABLE = /bin/bash ./wrfg4.sh
INPUT_FILES = WRF4G-1.5.1-x86_64.tar.gz, wrfg4.sh
REQUIREMENTS = ARCH = 'x86_64'

$ drm4g job submit wrf4g.jt

Job id: 0

$ drm4g job list 0

<table>
<thead>
<tr>
<th>JID</th>
<th>EM</th>
<th>START</th>
<th>END</th>
<th>EXIT</th>
<th>NAME</th>
<th>HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>actv</td>
<td>18:05:33</td>
<td>--:--:--</td>
<td>--</td>
<td>wrf4g.jt</td>
<td>esr::sbgce2.in2p3.fr</td>
</tr>
</tbody>
</table>

$ drm4g job cancel 0
Outline

• Statement of the problem
• Requirements
• DRM4G
• How does it work?
• Application example: WRF4G
• Conclusion
WRF4G is a framework developed by the Santander Meteorology Group, provides:

- Flexible WRF experiment design, execution and monitoring, and ...

- … the ability to run these experiments on different computing resources at the same time in a transparent way.
**SeaWind experiment set**: past re-forecasts (1989-2009) of an improved wind field over Europe for off-shore wind farms.

- **Features of the experiment**
  - 21 years of daily re-forecasts (36h each)
  - 7,665 independent simulations

- **Computing cost of the experiment**
  - Working Node Architecture
    - CPU: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz 8 Cores
    - RAM Memory: 16 GB
  - Result
    - Walltime (MPI job) = 21 x 365 x 70' ~ 1 year
    - Output = 21 x 365 x 17 GB ~ 130 TB
Outline

• Statement of the problem
• Requirements
• DRM4G
• How does it work?
• Application example: WRF4G
• Conclusion
Conclusions

• Easy
  – To install
  – To configure
• Useful to other frameworks
  – Providing a single access point
  – Python API
• Scalability (~100,000 jobs)
  – Stable
  – Reliable
• Future work → cloud
Thank you!

Contact: blancojc@unican.es

More info: https://www.meteo.unican.es

Aknowledgments: This work is funded by the Spanish PLAN NACIONAL de I+D+i 2008-2011 (Ref. # CGL2011-28864, WRF4G) and ERDF