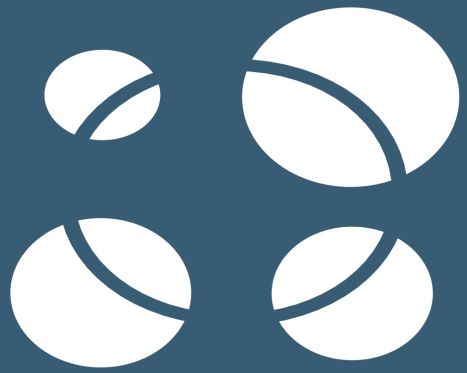
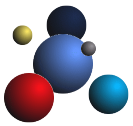
Marzo, 2007

GridWay@CNB



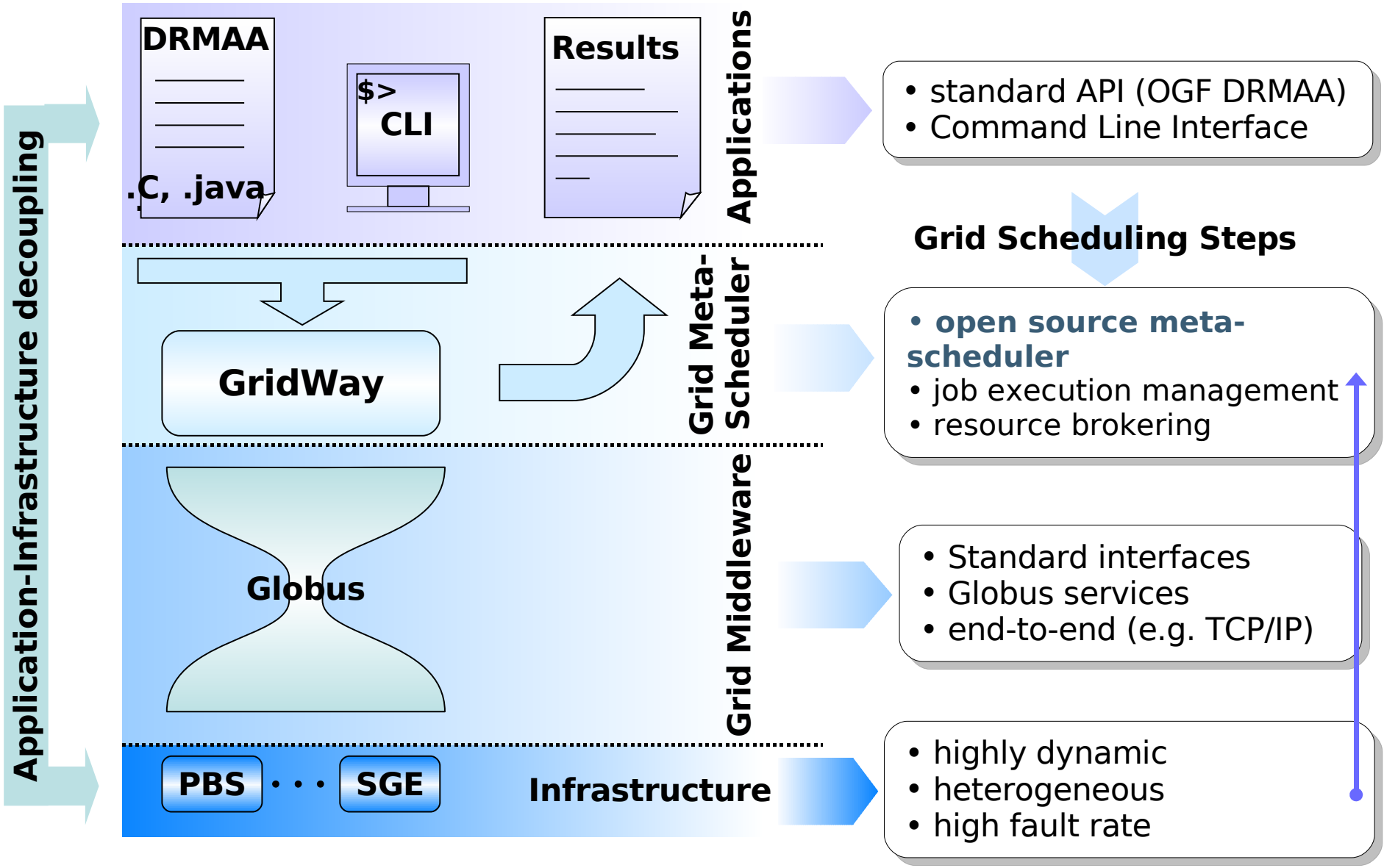
José Luis Vázquez-Poletti
Equipo GridWay
www.GridWay.org

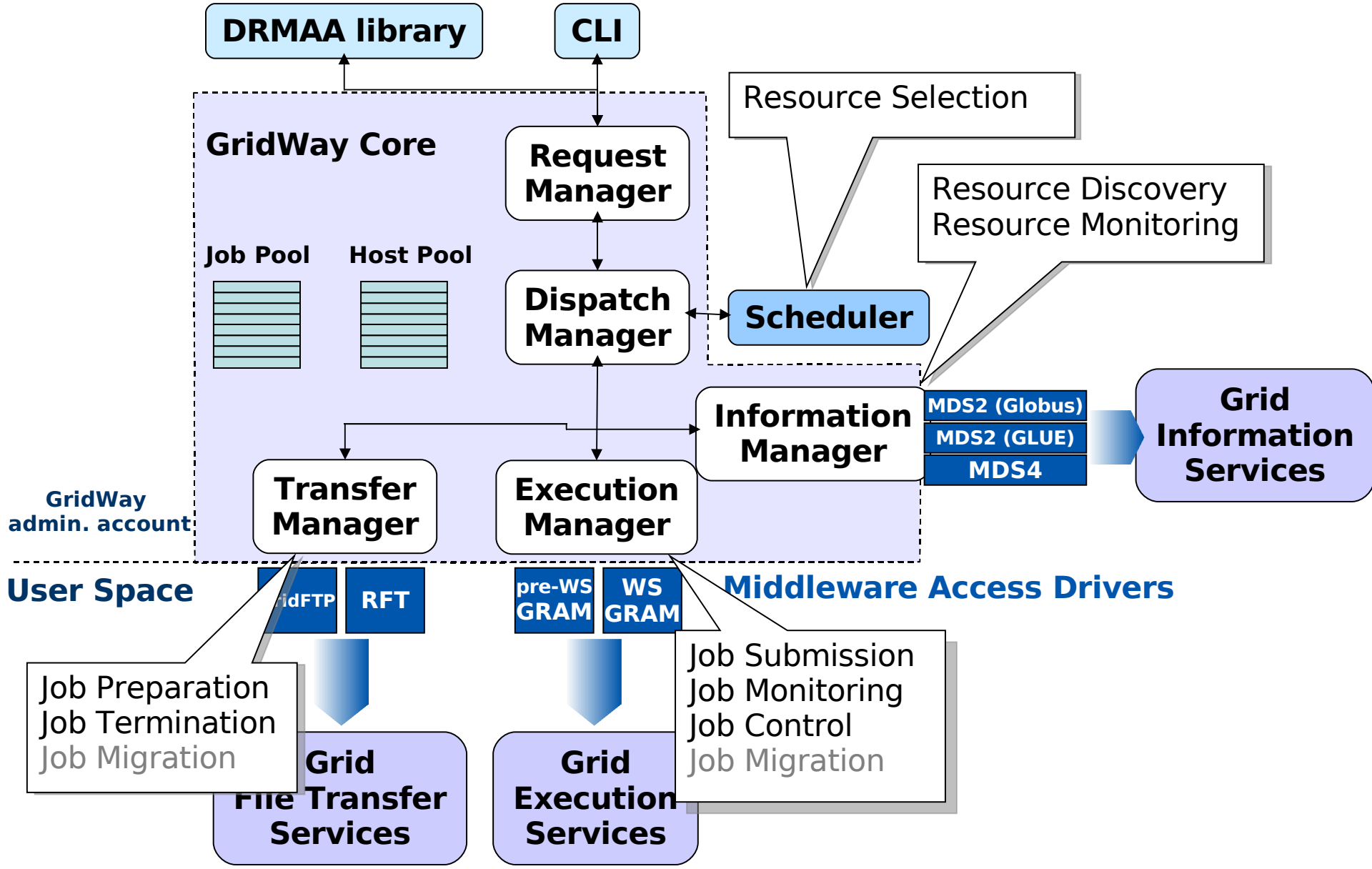


Distributed Systems Architecture Group
Departamento de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Contenidos

1. Una Visión Global
2. GridWay Ahí Fuera
3. Interactuar con GridWay (1ª Parte)
4. Interactuar con GridWay (2ª Parte)
5. Una Pequeña Demo


DRMAA
.C, .java
$>
CLI
Results
Applications
Grid Meta-Scheduler
GridWay
Grid Middleware
Globus
PBS
SGE
Infrastructure
Application-Infrastructure decoupling
• standard API (OGF DRMAA)
• Command Line Interface
Grid Scheduling Steps
• open source meta-scheduler
• job execution management
• resource brokering
• Standard interfaces
• Globus services
• end-to-end (e.g. TCP/IP)
• highly dynamic
• heterogeneous
• high fault rate


DRMAA library
CLI
GridWay Core
Request Manager
Job Pool
Host Pool
Dispatch Manager
Scheduler
Resource Selection
Resource Discovery
Resource Monitoring
Information Manager
MDS2 (Globus)
MDS2 (GLUE)
MDS4
Grid Information Services
GridWay admin. account
Transfer Manager
Execution Manager
User Space
idFTP
RFT
pre-WS GRAM
WS GRAM
Middleware Access Drivers
Job Preparation
Job Termination
Job Migration
Job Submission
Job Monitoring
Job Control
Job Migration
Grid File Transfer Services
Grid Execution Services

Workload Management

- Funciones de planificación avanzadas y soporte de nuevas políticas de planificación
- Detección y recuperación de fallos
- Servicio de Accounting
- Trabajos sencillos, matrices de trabajos y DAGs

Interfaz de Usuario

- Soporte completo para el estándar OGF DRMAA (C y JAVA)
- Interfaz de comandos similar al encontrado en los sistemas DRM

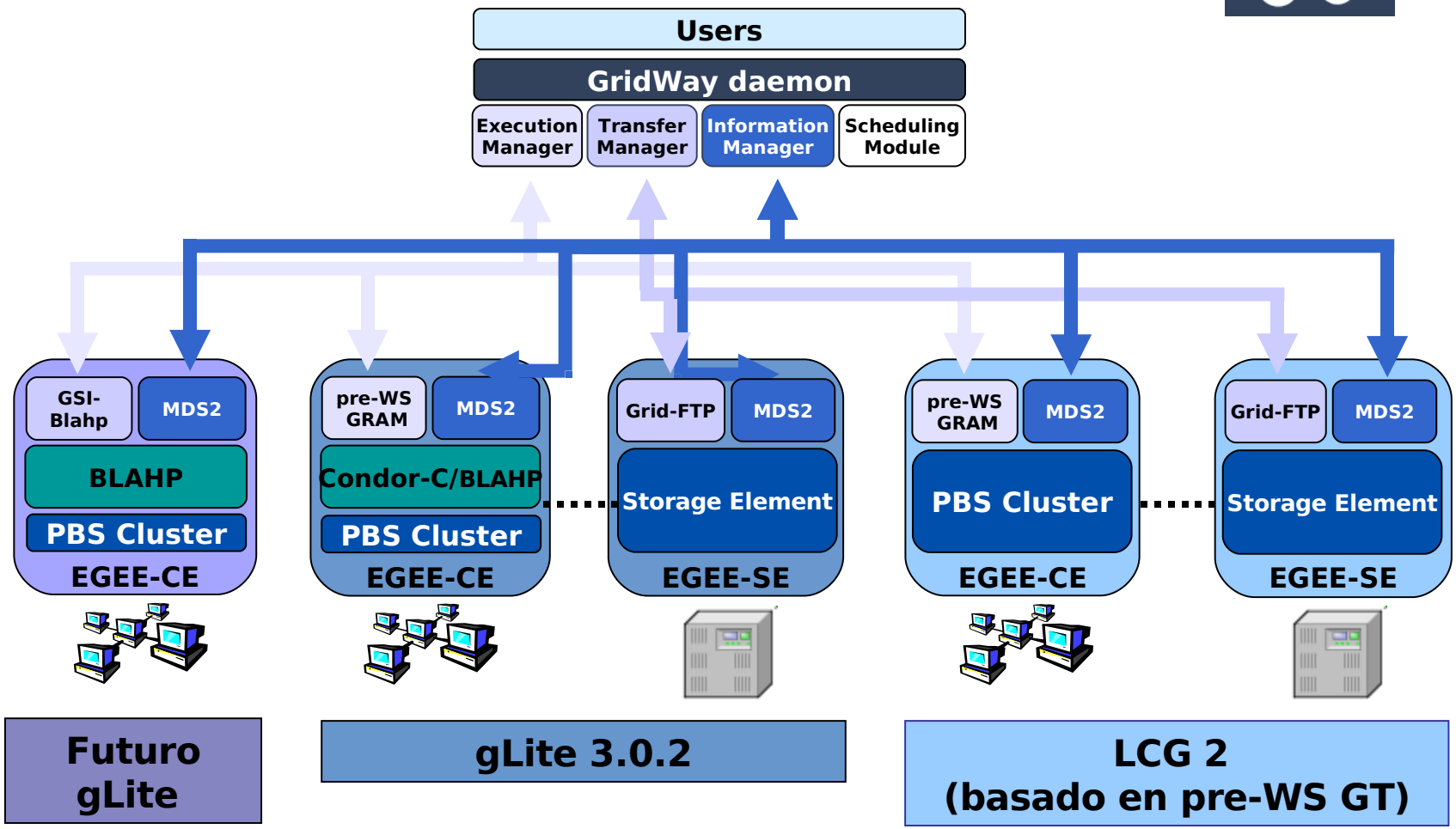
Integración

- No es necesario desplegar nuevos servicios
- Interfaz con nuevos servicios grid
- Interoperatibilidad con otras infraestructuras

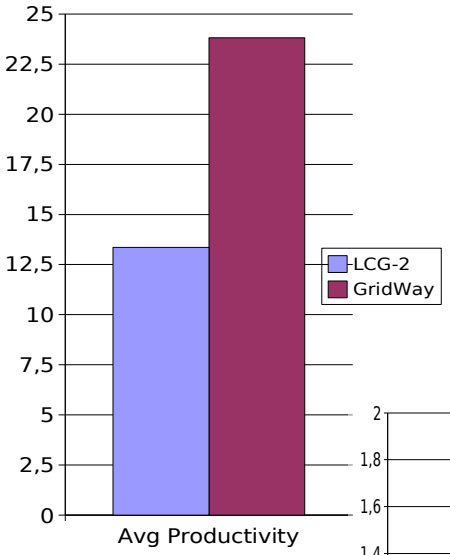
GridWay complementa a gLite

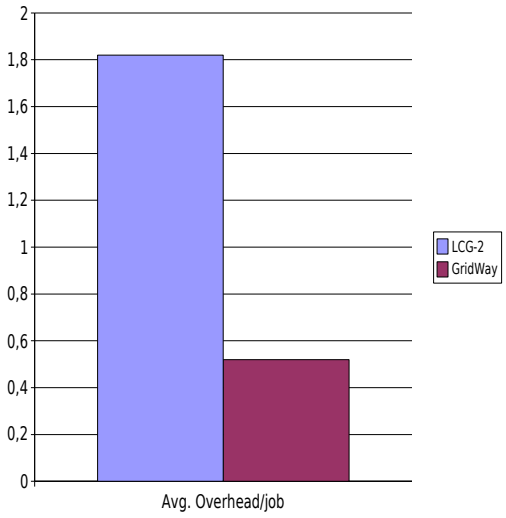
CARACTERÍSTICAS	BENEFICIOS
Soporte para el estándar DRMAA (C y JAVA)	Compatibilidad de aplicaciones sistemas DRM que implementa el estándar como SGE, Torque,...
Interfaz de comandos DRM (permite a los usuarios enviar, matar, migrar, monitorizar y sincronizar trabajos)	Interfaz de comandos similar al encontrado en los gestores de recursos locales
Middleware ligero	Mayor eficiencia para ciertos perfiles de aplicación
Accounting a nivel de site y políticas de planificación	Análisis del uso, determinando costumbres de uso y monitorizando el comportamiento del usuario
Mínimos requisitos de instalación	Despliegue y mantenimiento sencillo
Interoperatibilidad	Acceso simultaneo a diferentes infraestructuras

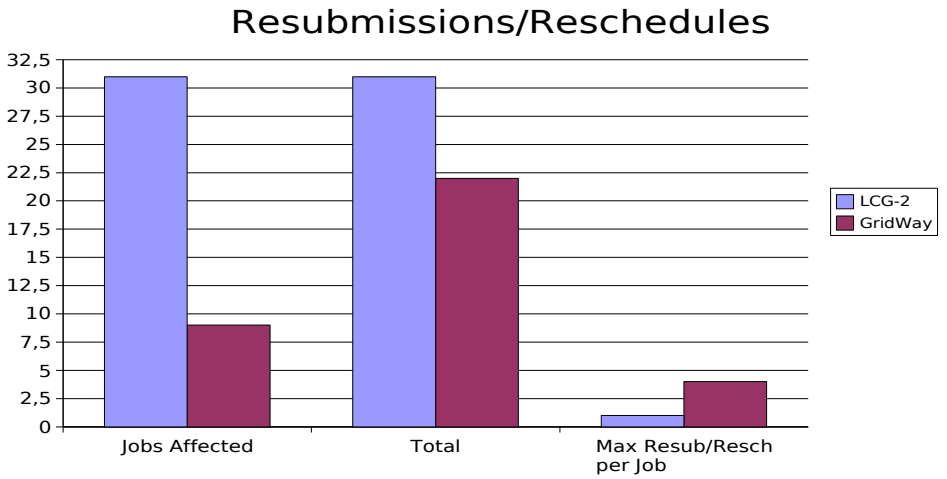
Infraestructura Grid

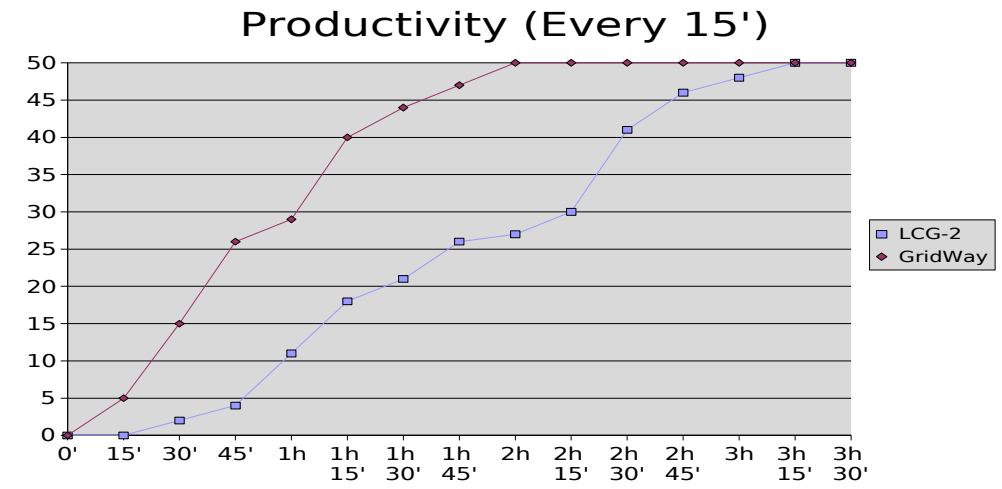


Comparativa GridWay / LCG-2 RB
EGEE User Forum (CERN, 1-3 Marzo 2006)









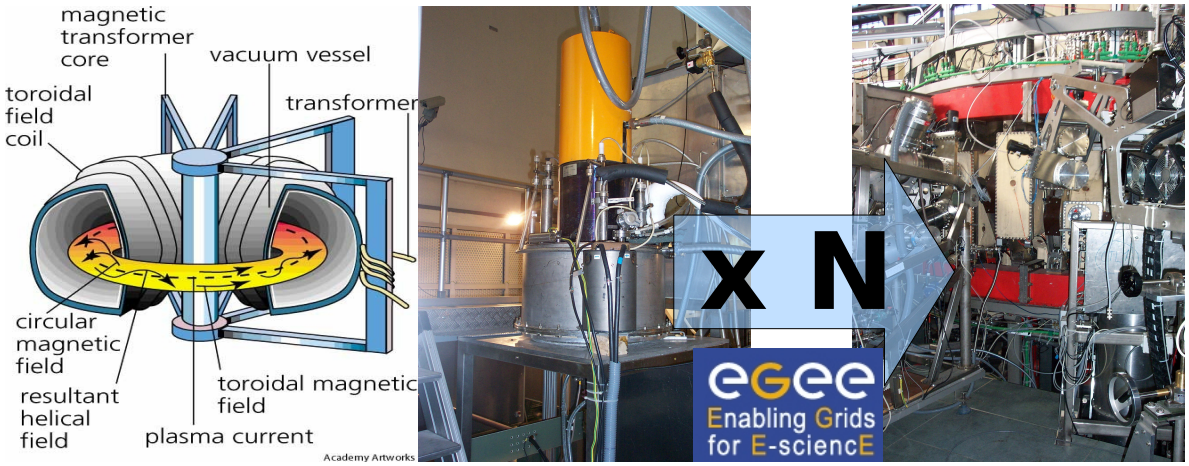
Algunos Proyectos e Infraestructuras

- IRISGrid
- Politecnico di Torino
- CABGrid (Centro de Astrobiología)
- C2VO (Universidad de Castilla La Mancha)
- Grid en ESAC (Agencia Espacial Europea)
- CRO-GRID (Croacia)
- Sun Microsystems Solution Center World Grid
- Infraestructura EGEE
- Proyecto BeinGRID
- GridX1 (Grid canadiense para aplicaciones HEP)
- Universidade do Porto
- Madras Institute of Technology
- National Center for High-Performance Computing

Algunos Ámbitos de Portado de Aplicaciones

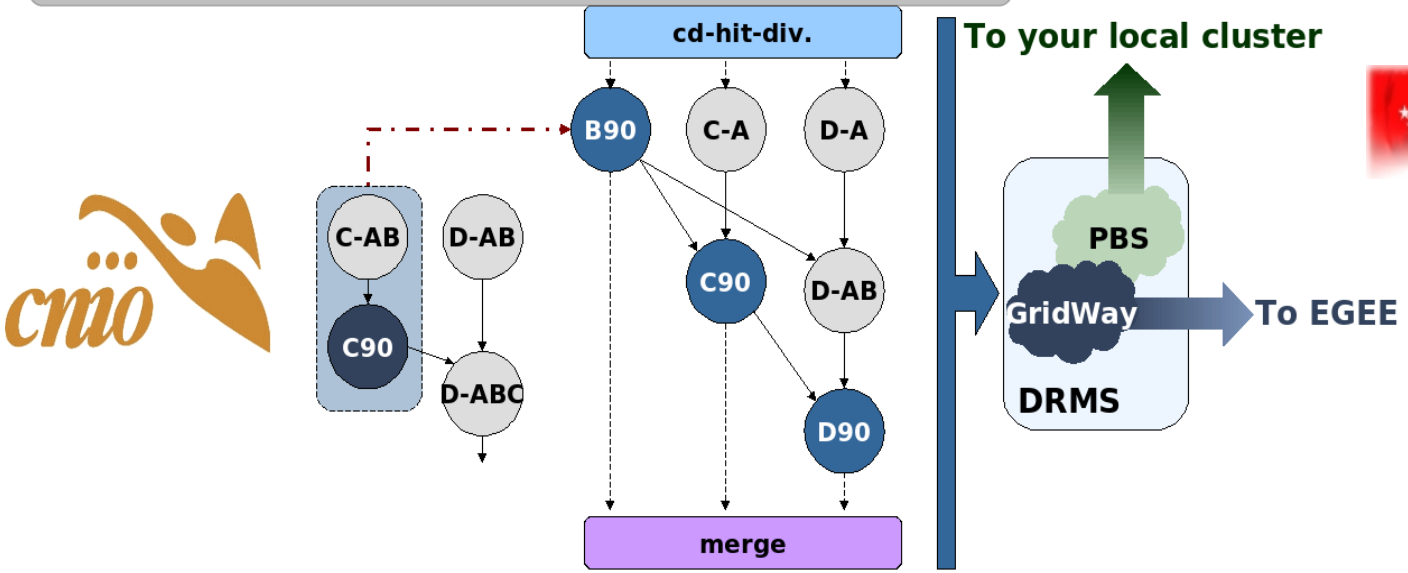
- Ciencias de la Vida
- Aeroespacio
- Física de Fusión
- Química Computacional

MAssive RAy TRAcing in Fusion Plasmas



- **Executable: *Truba***
  - 1,8 MB – 9' – 50 Executions
- **Input files =~ 70 KB**
- **Output files =~ 549 KB**

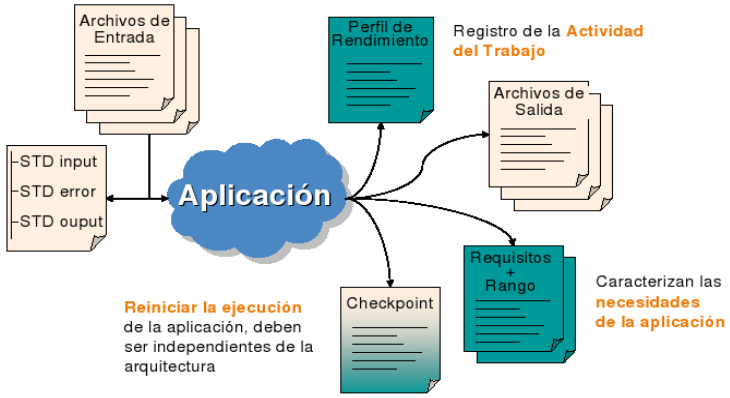
CD-HIT

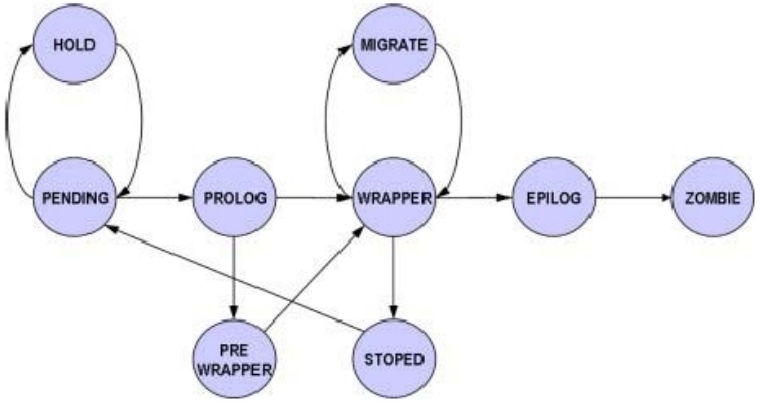


Comandos Principales

- **gwps:** Muestra información y estado de cada trabajo
- **gwhistory:** Muestra la historia de ejecución
- **gwkill:** Envío de señales a un trabajo (kill, stop, resume, reschedule)
- **gwsubmit:** Envío de un trabajo o array
- **gwwait:** Espera finalización de trabajo (any, all, set)
- **gwuser:** Monitorización de usuarios
- **gwhost:** Monitorización de hosts
- **gwacct:** Contabilidad

Modelo de Aplicación y Ciclo de Vida





Job Template (Ejemplo)

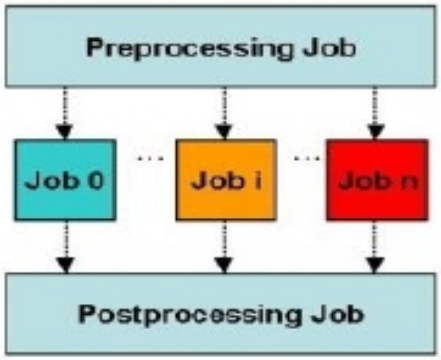
```
# Execution variables
EXECUTABLE  = job
ARGUMENTS   = ${TASK_ID} ${TOTAL_TASKS} 100000
ENVIRONMENT = LD_LIBRARY_PATH=/usr/local/lib

# Resource selection parameters
REQUIREMENTS = HOSTNAME= "*.dacya.ucm.es"
RANK         = CPU_MHZ

# I/O files
INPUT_FILES  = my_inputfile
OUTPUT_FILES = my_outputfile

# Standard streams
STDOUT_FILE = stdout_file.${TASK_ID}
STDERR_FILE = stderr_file.${TASK_ID}
… otras variables relacionadas con checkpointing,
tolerancia de fallos, rendimiento,…
```

DRMAA API

- **Distributed Resource Management Application API**
  - http://www.drmaa.org/
- Estándar del Open Grid Forum
- Interfaz homogeneo a diferentes sitemas de gestión de recursos distribuidos (DRMS):
  - SGE
  - Condor
  - PBS (en camino)
  - **GridWay**
    - **C**
    - JAVA

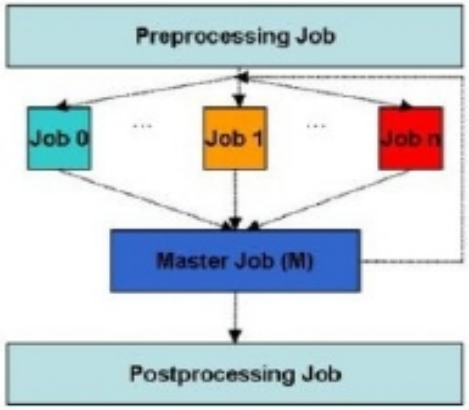
Perfiles de Aplicación con DRMAA

- Embarrasingly Distributed




```
rc = drmaa_init (contact, err);
// Execute initial job and wait for it
rc = drmaa_run_job(job_id, jt, err);
rc = drmaa_wait(job_id, &stat, timeout, rusage, err);
// Execute n jobs simultaneously and wait
      rc      =      drmaa_run_bulk_jobs(job_ids,jt,1,
JOB_NUM,1,err);
rc = drmaa_synchronize(job_ids, timeout, 1, err);
// Execute final job and wait for it
rc = drmaa_run_job(job_id, jt, err);
rc = drmaa_wait(job_id,&stat, timeout, rusage, err);
rc = drmaa_exit(err_diag);
```

- Master-Worker




```
rc = drmaa_init(contact, err_diag);
// Execute initial job and wait for it
rc = drmaa_run_job(job_id, jt, err_diag);
rc = drmaa_wait(job_id, &stat, timeout, rusage, err_diag);
while (exitstatus != 0)
{
 // Execute n Workers concurrently and wait
   rc = drmaa_run_bulk_jobs(job_ids, jt, 1, JOB_NUM, 1,
err_diag);
 rc = drmaa_synchronize(job_ids, timeout, 1, err_diag);
 // Execute the Master, wait and get exit code
 rc = drmaa_run_job(job_id, jt, err_diag);
    rc = drmaa_wait(job_id, &stat, timeout, rusage,
err_diag);
 rc = drmaa_wexitstatus(&exitstatus, stat, err_diag);
}
rc = drmaa_exit(err_diag);
```

Estructura de Programa y Compilación

Incluir la librería de DRMAA

```
#include “drmaa.h”
```

Verificar la siguiente variable de entorno (.bashrc)

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$GW_LOCATION/lib/
```

Incluir opciones de compilación para DRMAA

```
-L $GW_LOCATION/lib
-I $GW_LOCATION/include
-ldrmaa
```

```
gcc ejemplo.c -L $GW_LOCATION/lib
      -I $GW_LOCATION/include -ldrmaa -o ejemplo
```

Sesiones DRMAA

Iniciar Sesión

```
int drmaa_init (const char *contact, char *error_diagnosis, size_t error_diag_len)
```

Finalizar Sesión

```
int drmaa_exit (char *error_diagnosis, size_t error_diag_len)
```

Creación del Job Template

Asignación del Job Template

```
int drmaa_allocate_job_template (drmaa_job_template_t **jt, char *error_diagnosis, size_t error_diag_len)
```

Introducción de un Atributo Escalar

```
int drmaa_set_attribute (drmaa_job_template_t *jt, const char *name,
                         const char *value, char *error_diagnosis, size_t error_diag_len)
```

Introducción de un Atributo Vector (ej: cadena con argumentos del ejecutable)

```
int drmaa_set_vector_attribute (drmaa_job_template_t *jt, const char *name,
                                const char *value[], char *error_diagnosis, size_t error_diag_len)
```

Envío del Trabajo

Envío del Trabajo

```
int drmaa_run_job (char *job_id, size_t job_id_len,
                   drmaa_job_template_t *jt, char *error_diagnosis, size_t error_diag_len)
```

Esperar el Trabajo

```
int drmaa_wait (const char *job_id, char *job_id_out, size_t job_id_out_len, int *stat, signed long timeout,
                drmaa_attr_values_t **rusage, char *error_diagnosis, size_t error_diag_len)
```

Obtención del Código de Salida

```
int drmaa_wexitstatus (int *exit_status, int stat, char *error_diagnosis, size_t error_diag_len)
```

Obtención de estadísticas de uso remoto

```
int drmaa_get_next_attr_name (drmaa_attr_names_t *values, char *value, size_t value_len)
```

Borrar Job Template

```
int drmaa_delete_job_template (drmaa_job_template_t *jt, char *error_diagnosis, size_t error_diag_len)
```

Estado y Control del Trabajo

Obtención del Estado de un Trabajo

```
int drmaa_job_ps (const char *job_id, int *remote_ps, char *error_diagnosis, size_t error_diag_len)
```

Esperar finalización de un Trabajo

```
int drmaa_synchronize (const char *job_ids[], signed long timeout,
                       int dispose, char *error_diagnosis, size_t error_diag_len)
```

Emitir Órdenes de Control al Trabajo

```
int drmaa_control (const char *jobid, int action, char *error_diagnosis, size_t error_diag_len)
```

Matriz (Array) de Trabajos

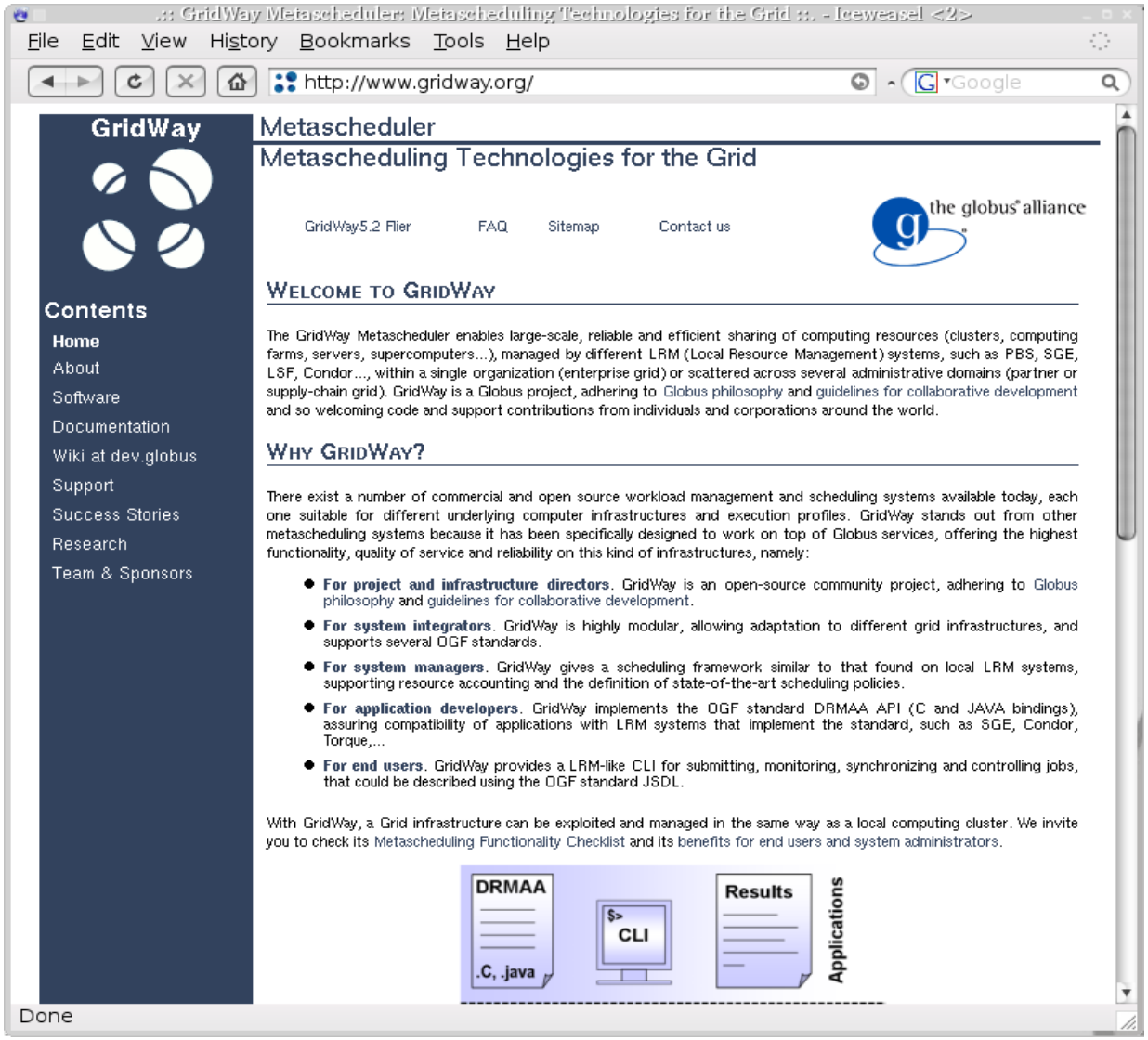
Lanzar Matriz de Trabajos

```
int drmaa_run_bulk_jobs (drmaa_job_ids_t **jobids, drmaa_job_template_t *jt, int start,
                         int end, int incr, char *error_diagnosis, size_t error_diag_len)
```

Obtener el siguiente Identificador de Trabajo

```
int drmaa_get_next_job_id (drmaa_job_ids_t *values, char *value, size_t value_len)
```

http://www.GridWay.org/



¡Muchas Gracias!