- Isabel Campos y Enol Fernández son los autores de la mayor parte de este material así como de las herramientas que lo hacen posible.

Contenidos

- **Entorno de Prácticas**
- Definición de Trabajos en IAGs
  - Introducción
  - Lenguaje de Descripción de Trabajos (JDL)
- Lanzamiento y Monitorización de Trabajos
  - Ciclo de Vida
  - Command Line Interfaces /CLIs
- Practicas
  - P1. Simple Job
  - P2. Simple Job in Parallel Resources
  - P3. Simple Parallel Job
  - P4. Simple MPI Job
  - P5. Simple MPI Job & Pre-Hooks (Remote Compilation)
  - P6. Simple MPI Job & Post-Hooks

Entorno de Trabajo

- **forward.i3m.upv.es**
- **Users**
  - User: mpg
  - Passwd: XXX

Instituto de Física de Cantabria

Contenidos

- Entorno de Prácticas
- **Definición de Trabajos en IAGs**
  - Introducción
  - Lenguaje de Descripción de Trabajos (JDL)
- Lanzamiento y Monitorización de Trabajos
  - Ciclo de Vida
  - Command Line Interfaces /CLIs
- Practicas
  - P1. Simple Job
  - P2. Simple Job in Parallel Resources
  - P3. Simple Parallel Job
  - P4. Simple MPI Job
  - P5. Simple MPI Job & Pre-Hooks (Remote Compilation)
  - P6. Simple MPI Job & Post-Hooks

Definición de Trabajos en IAGs
Introducción

- Definición de Trabajos.
- CLIs para la gestión del ciclo de vida de los trabajos.

Definición de Trabajos
Introducción

- Para poder Lanzar un Trabajo hay que Definirlo previamente, para ello hay que definir:
  - Características del Trabajo.
  - Requerimientos y Preferencias del Trabajo Con Respecto a los Recursos de Proceso, Incluyendo las Dependencias de Software.
  - Requerimientos de los Trabajos en lo que se Refiere a Datos.
- Toda esta Información se Especifica en un Fichero Mediante el Lenguaje de Especificación de Trabajos (Job Description Language ó JDL)
  - Especifica una Serie de Atributos que son Utilizados por el WMP en la Selección de los Recursos.

Definición de Trabajos
Lenguaje de Descripción de Trabajos (JDL)

- Un Atributo es un Par (Clave, Valor), Donde Valor Puede ser un Booleano, un Entero o una Lista de Cadenas.
  - `<atributo> = <valor>`
- En el Caso de una Cadena se Especifica entre Comillas Dobles.
  - Las Comillas Dobles Pueden Incluirse como Carácter Precedidas por la Contrabarra.
    - `Arguments = " \"Hola\" 10";`
  - Los Caracteres Especiales como &, |, <, > se Permiten Únicamente
    - Si Están Dentro de una Cadena Entre Comillas y
    - Si Están Precedidas de una Triple Contrabarra
      - `Arguments = "-f file1\\\&file2";`
- Los Comentarios se Preceden de una Almohadilla o Mediante la Sintaxis C++.
- Cuidado con los Tabuladores y Espacios en Blanco.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) - Atributos

- Los Atributos Soportados se Agrupan en Dos Categorías
  - Atributos del Trabajo
    - Definen el Trabajo en Sí Mismo.
  - Atributos de Expresión de Recursos
    - Son Utilizados por el Gestor de Recursos para Realizar la Selección de los Recursos Más Adecuados Para el Trabajo.
    - También Definen los Recursos de Almacenamiento.
  - Atributos de Expresión de Requerimientos y Rangos de Selección de Recursos.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) - Atributos

- JobType (Tipo de Trabajo)
  - Normal (Trabajo Secuencial), Interactive, MPICH, Checkpointable.
  - O Cualquier Combinación Compatible de los Mismos.
- Executable (Obligatorio)
  - Nombre del Programa Ejecutable.
- Arguments (Opcional)
  - Argumentos de Línea de Comandos.
- StdInput, StdOutput, StdError (Opcional)
  - Entrada / Salida / Error Estándar del Trabajo.
- Environment (Opcional)
  - Lista de las Variables de Entornos.
- InputSandbox (Opcional)
  - Lista de Ficheros en el Disco Local del UI Necesitados para la Ejecución del Trabajo que Se Copiarán Automáticamente en el Recurso Remoto.
- OutputSandbox (Opcional)
  - Lista de Ficheros Generados por el Trabajo que serán Recuperados Tras la Ejecución del Trabajo.
- VirtualOrganisation (Opcional)
  - Especificación de la VO.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Atributo Rank

- El Atributo “Rank” Permite Expresar la Preferencia en la Selección de los Recursos que son Compatibles con los Requerimientos.
- Se Expresa como un Valor en Coma Flotante.
- Se Selecciona el CE con el Valor Más Alto del Rango.
- Si no se Especifica Ningún Rango, se Utiliza el Indicado por Defecto en el User Interface.
- Ejemplo:
  - rank = (-other.GlueCEStateEstimatedResponseTime);

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Atributo Rank

- Ejemplos de uso:
  - `-other.GlueCEStateEstimatedResponseTime`
    - El Tiempo Mínimo Estimado Total.
    - Típicamente el Valor por Defecto.
  - `other.GlueCEStateFreeCPUs`
    - the highest number of free CPUs
    - Confusa. El Número de CPUs se Publica por Cola no por VO, por lo que una VO Puede no Tener Acceso.
  - `(other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs : -other.GlueCEStateWaitingJobs)`
    - Se Prefieren los Recursos que no Tienen Trabajos en Espera, y Ordenados a su vez por el Número de CPUs Libres. En Caso de que Hayan Trabajos en Espera, se Ordena en Función del Número de Trabajos.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Atributo Requirements

- Requerimientos del Trabajo Con Respecto a los Recursos.
- Especificados a Partir de los Atributos de los Recursos de Acuerdo con el “GLUE-Schema” Publicados por el Sistema de Información del Grid.
- Su Valor es una Expresión Booleana.
- Sólo se Puede Especificar una Línea de Requierements.
- Si no se Especifica, se Utiliza la Configuración por Defecto Indicada en el UI.
  - Por Defecto: other.GlueCEStateStatus == "Production" (el Recurso Tiene que Permitir la Ejecución de Trabajos).

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Glue Schema (I)

- Estado (objectclass GlueCEState)
  - GlueCEStateRunningJobs:
    - Número de Trabajos en Ejecución.
  - GlueCEStateWaitingJobs:
    - Número de Trabajos en Espera.
  - GlueCEStateTotalJobs:
    - Número Total de Trabajos (Ejecución + Espera).
  - GlueCEStateFreeCPUs:
    - Número de CPUs Disponibles.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Glue Schema (II)

- Estado (objectclass GlueCEState)
  - GlueCEStateStatus:
    - Estado de la Cola
      - queueing (Se Aceptan los Trabajos pero no se Ejecutan).
      - production (Se Aceptan los Trabajos y se Ejecutan).
      - closed (No se Aceptan Trabajos).
      - draining (No se Aceptan Trabajos pero Aún Quedan Trabajos en Cola)
  - GlueCEStateWorstResponseTime:
    - Cota Superior Tiempo Estimado entre el Lanzamiento del Trabajo y su Puesta en Marcha.
  - GlueCEStateEstimatedResponseTime:
    - Tiempo Estimado.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Glue Schema (III)

- Política (objectclass GlueCEPolicy)
  - GlueCEPolicyMaxWallClockTime:
    - Tiempo Máximo que Pueden Consumir los Trabajos (seg.).
  - GlueCEPolicyMaxCPUTime:
    - Tiempo Efectivo (de CPU) que Pueden Consumir los Trabajos (seg.) .
  - GlueCEPolicyMaxTotalJobs:
    - Máximo Número de Trabajos Permitidos en la Cola.
  - GlueCEPolicyMaxRunningJobs:
    - Máximo Número de Trabajos En Ejecución.

Definición de Trabajos
Lenguaje de Descripción de Trabajos (JDL) – Glue Schema (IV)

- Arquitectura (objectclass GlueHostArchitecture)
  - GlueHostArchitecturePlatformType:
    - Descripción de la Plataforma.
  - GlueHostArchitectureSMPSize:
    - Número de CPUs.
- Procesador (objectclass GlueHostProcessor)
  - GlueHostProcessorVendor:
    - Fabricante de la CPU.
  - GlueHostProcessorModel:
    - Modelo de la CPU.
  - GlueHostProcessorVersion:
    - Versión de la CPU.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Glue Schema (V)

- Software de Aplicación (objectclass GlueHostApplicationSoftware)
  - GlueHostApplicationSoftwareRunTimeEnvironment:
    - Listado del Software Instalado.
- Memoria Principal (objectclass GlueHostMainMemory)
  - GlueHostMainMemoryRAMSize:
    - RAM Física.
  - GlueHostMainMemoryVirtualSize:
    - Tamaño de la Memoria Virtual Configurada.
- Benchmark (objectclass GlueHostBenchmark)
  - GlueHostBenchmarkSI00:
    - Valor del SpecInt2000 Benchmark.
  - GlueHostBenchmarkSF00:
    - Valor del SpecFloat2000 Benchmark.
- Adaptador de Red (objectclass GlueHostNetworkAdapter)
  - GlueHostNetworkAdapterOutboundIP:
    - Posibilidad de Realizar Conexiones al Exterior.
  - GlueHostNetworkAdapterInboundIP:
    - Posibilidad de Recibir Conexiones desde el Exterior.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Ej. Requeriments

- Ejemplos de Requerimientos Interesantes
  - `other.GlueCEInfoLRMSType == "PBS" && other.GlueCEInfoTotalCPUs > 1`
    - El Trabajo Necesita que el Recurso Disponga del Gestor PBS y que los Working Nodes Tengan al Menos 2 CPUs.
  - `RegExp("upv.es", other.GlueCEUniqueId)`
    - El Trabajo debe Ejecutarse en el Dominio de la upv.
  - `other.GlueHostNetworkAdapterOutboundIP == true) && (other.GlueCEPolicyMaxWallClockTime > 86000) && Member("GEANT4", other.GlueHostApplicationSoftwareRunTimeEnvironment)`
    - El Recurso Deberá Tener Instalado el Paquete "Geant4" y Debe Permitir que se Ejecuten Trabajos que Puedan Tardar más de 86000 Segundos.

Definición de Trabajos

Lenguaje de Descripción de Trabajos (JDL) – Ejemplo Trabajo

```
[
JobType = “Normal”;
Executable = "$(CMS)/exe/sum.exe";
InputSandbox =
{"/home/user/WP1testC","/home/file*”,
"/home/user/DATA/*"};
OutputSandbox = {“sim.err”, “test.out”,
“sim.log"};
Requirements  = (other.
GlueHostOperatingSystemName == “linux") &&
(other.GlueCEPolicyMaxWallClockTime > 10000);
Rank = other.GlueCEStateFreeCPUs;
]
```

Definición de Trabajos en IAGs
Introducción

- Definición de Trabajos.
- CLIs para la gestión del ciclo de vida de los trabajos.

CLIs para la gestión del ciclo de vida de los trabajos.

1 - Especificar Trabajo Grid → Ficheros JDL

2 - Crear las Credenciales (proxy voms) → voms-proxy-init –voms / myproxy-get-delegation

3 - Listar Recursos Compatibles con el Trabajo Grid JDL → glite-wms-job-list-match

4 - Enviar el Trabajo al WMS → glite-wms-job-submit -a <jdl_file>

5 - Monitorizar el estado del trabajo → glite-wms-job-status

6 - Recuperar la salida → glite-wms-job-output

7 - Destruir Proxies → voms-proxy-destroy

CLIs para la gestión del ciclo de vida de los trabajos.

- glite-wms-job-list-match
  - Lista los Recursos Compatibles con una Determinada Especificación.
  - La Opción –rank Muestra la Clasificación por Preferencia de los Recursos.
  - No Realiza el Lanzamiento del Trabajo.
- glite-wms-job-cancel
  - Cancela el Trabajo Especificado.
- glite-wms-job-status
  - Muestra el Estado de Un Trabajo.

CLIs para la gestión del ciclo de vida de los trabajos.

- glite-wms-job-submit <delegation-opts> [options] <jdl_file>
  - -r Indica el Recurso que se Quiere Utilizar
  - -c El fichero de Configuración a Utilizar en Lugar del Por Defecto
  - -a Delegación Automática, quiere decir, Autorizar al WMP a Actuar a Nombre del Usuario Especificado en las Credenciales de Usuario.
  - --vo Organización Virtual
  - -o Volcado del Identificador del Trabajo en un fichero.

CLIs para la gestión del ciclo de vida de los trabajos.

- glite-wms-job-output
  - Retorna la Salida de un Trabajo (Ficheros del OutputSandbox) al Usuario.
- glite-wms-job-logging-info
  - Muestra la Información Registrada por un Trabajo Lanzado Sobre Todos los Eventos
  - Diferentes Niveles de Detalle (opción –v) :
    - Nivel 1 Suficiente para la Depuración.
    - Nivel 2 Quizás Demasiada Información.

Uso de MPI en Grids:

Ejercicios Prácticos

P1. Simple Job

P1. Simple Job

- Sitúate en la carpeta siguiente:

```
[ccgc@XXX~]$ cd /home/mpg/Evidencias/Seminario_MPI_Grid/P1
```

- Todos los ficheros que generes en esta práctica debes de guardarlo en esta carpeta para que consten como evidencia de su realización.

P1. Simple Job

hello.jdl

```
JobType          = "Normal";
Executable       = "hello.sh";
Arguments        = "hola";
StdOutput        = "std.out";
StdError         = "std.err";
InputSandbox     = {"hello.sh"};
OutputSandbox    = {"std.err","std.out"};
```

hello.sh

```
#!/bin/bash
echo "hello world from `hostname`"
echo "$1"
```

P1. Simple Job

- Crea el trabajo Grid fichero hello.JDL
- Incluye un requerimiento para que se ejecute solo en CE que se encuentren la “ceta-ciemat.es”.
- Consulta los recursos disponibles para el lanzamiento de este JDL y vuelca el resultado en un fichero de texto llamado “recursos.txt”.
- Lanza el trabajo en la vo “biomed” y monitorizalo hasta que este finalice.
- Descarga los ficheros resultado una vez finalizado en la carpeta “P1”.

P1. Simple Job

- En la carpeta "home/mpg/Evidencias/Seminario_MPI_Grid/P1" deben aparecer los siguientes archivos:
  - hello.jdl
  - hello.sh
  - Recursos.txt
  - Carpeta mpg_XXXXXX con los ficheros std.out y std.err

P2. Simple Job in Parallel Resources

P2. Simple Job in Parallel Resources

- Sitúate en la carpeta siguiente:

```
[ccgc@XXX~]$ cd /home/mpg/Evidencias/Seminario_MPI_Grid/P2
```

- Todos los ficheros que generes en esta práctica debes de guardarlo en esta carpeta para que consten como evidencia de su realización.

P2. Simple Job in Parallel Resources
Finding where to run (I)

Sites publish in their tags:

- MPI-START (if they have mpi-start installed)
- Then for each MPI flavour supported:
  - <MPI flavour>
  - <MPI flavour>-<MPI version>
  - <MPI flavour>-<MPI version>-<Compiler>
- And interconnect info (optional and mostly absent):
  - MPI-<interconnect>
  - Interconnects: Ethernet, Infiniband, SCI, Myrinet

P2. Simple Job in Parallel Resources
Finding where to run (II)

```
$ lcg-info –vo biomed --list-ce --query 'Tag=MPI-START’
- ce-01.roma3.infn.it:8443/cream-pbs-fastgrid
- ce.ceta-ciemat.es:8443/cream-sge-biomed
- …
$ lcg-info –vo biomed --list-ce --query 'Tag=MPICH2’
- ce-02.roma3.infn.it:8443/cream-pbs-grid
- ce-grid.obspm.fr:2119/jobmanager-pbs-biomed
- …
$ lcg-info --vo biomed --list-ce --query 'Tag=OPENMPI-1.4.4'
- CE: cream.sg.grena.ge:8443/cream-pbs-biomed
- CE: gridce01.ifca.es:8443/cream-sge-biomed
- CE: gridce02.ifca.es:8443/cream-sge-biomed
- …
$ lcg-info --vo biomed --list-ce --query 'Tag=OPENMPI-1.5.4'
- CE: ce05.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long
- CE: ce05.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_2G_long
- CE: ce06.esc.qmul.ac.uk:8443/cream-sge-sl6_lcg_1G_long
- …
```

P2. Simple Job in Parallel Resources
Finding where to run (III)

```
$ lcg-info --vo biomed --list-ce --query 'Tag=MPI-INFINIBAND'
- CE: ce-01.roma3.infn.it:8443/cream-pbs-fastgrid
- CE: ce-01.roma3.infn.it:8443/cream-pbs-grid
- CE: creamce.reef.man.poznan.pl:8443/cream-pbs-biomed

$ lcg-info --vo biomed --list-ce --query 'Tag=OPENMPI,Tag=MPI-
START,Tag=MPI-ETHERNET'
- CE: cream.sg.grena.ge:8443/cream-pbs-biomed
- CE: gridsrv2-4.dir.garr.it:8443/cream-pbs-grid
```

P2. Simple Job in Parallel Resources

- Create this files without requirements.

hello.jdl

```
JobType             = "Normal";
Executable          = "hello.sh";
Arguments           = "hola";
StdOutput           = "std.out";
StdError            = "std.err";
InputSandbox        = {"hello.sh"};
OutputSandbox       = {"std.err","std.out"};
```

hello.sh

```
#!/bin/bash
echo "hello world from `hostname`"
echo "$1"
```

P2. Simple Job in Parallel Resources
Finding where to run (IV)

```
$ glite-wms-job-list-match --vo biomed -a hello.jdl
```

```
Connecting to the service
https://wms1.grid.sara.nl:7443/glite_wms_wmproxy_server

==========================================================
=========

                      COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been
found:

        *CEId*
 - cccreamceli09.in2p3.fr:8443/cream-sge-short
 - cccreamceli10.in2p3.fr:8443/cream-sge-short
 - ce3.ui.savba.sk:8443/cream-pbs-biomed
 - clrccece03.in2p3.fr:8443/cream-pbs-biomed
 - cream-ce02.cat.cbpf.br:8443/cream-pbs-biomed
…
```

P2. Simple Job in Parallel Resources
Finding where to run (V)

- Use the JDL requirements to match to sites supporting your needs:

```
Requirements    =
Member("MPI-START",
other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

- Create a new File named hello_req.jdl:

P2. Simple Job in Parallel Resources

- Crea el trabajo Grid fichero hello_req.JDL
- Consulta los recursos disponibles para el JDL y vuelca el resultado en un fichero de texto llamado “recursos.txt”.
- Lanza el trabajo en la vo “biomed” y monitorizalo hasta que este finalice.
- Descarga los ficheros resultado una vez finalizado en la carpeta “P2”.

P2. Simple Job in Parallel Resources

- En la carpeta "home/mpg/Evidencias/Seminario_MPI_Grid/P2" deben aparecer los siguientes archivos:
  - hello_req.jdl
  - hello.sh
  - recursos.txt
  - Carpeta mpg_XXXXXX con los ficheros std.out y std.err

P3. Simple Parallel Job

P3. Simple Parallel Jobs

- Sitúate en la carpeta siguiente:

```
[ccgc@XXX~]$ cd /home/mpg/Evidencias/Seminario_MPI_Grid/P3
```

- Todos los ficheros que generes en esta práctica debes de guardarlo en esta carpeta para que consten como evidencia de su realización.

P3: Simple Parallel Jobs (parallel.jdl)

Basic example

parallel.jdl

```
JobType          = "Normal";
nodeNumber       = 4;
Executable       = "starter.sh";
Arguments        = "hello.sh OPENMPI";
InputSandbox     = {"starter.sh", "hello.sh"};
StdOutput        = "std.out";
StdError         = "std.err";
OutputSandbox    = {"std.out","std.err"};
Requirements     =
Member("MPI-START",
other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

P3: Simple Parallel Jobs

starter.sh

```
#!/bin/bash

MY_EXECUTABLE=$1
MPI_FLAVOR=$2

# Convert flavor to lowercase for passing to mpi-start.
MPI_FLAVOR_LOWER=`echo $MPI_FLAVOR | tr '[:upper:]' '[:lower:]'`

# Pull out the correct paths for the requested flavor.
eval MPI_PATH=`printenv MPI_${MPI_FLAVOR}_PATH`

# Ensure the prefix is correctly set.  Don't rely on the defaults.
eval I2G_${MPI_FLAVOR}_PREFIX=$MPI_PATH
export I2G_${MPI_FLAVOR}_PREFIX

# Touch the executable.  It exist must for the shared file system check.
# If it does not, then mpi-start may try to distribute the executable
# when it shouldn't.
touch $MY_EXECUTABLE
chmod +x $MY_EXECUTABLE

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
#export I2G_MPI_APPLICATION_ARGS="./ 0 0 1"
# optional hooks
#export I2G_MPI_PRE_RUN_HOOK=hooks.sh
#export I2G_MPI_POST_RUN_HOOK=hooks.sh

# If these are set then you will get more debugging information.
#export I2G_MPI_START_VERBOSE=1
#export I2G_MPI_START_TRACE=1
#export I2G_MPI_START_DEBUG=1

# Invoke mpi-start.
$I2G_MPI_START
```

P3: Simple Parallel Jobs
hello.sh

hello.jdl

```
#!/bin/bash

echo "hello world from `hostname`"
```

… things may go wrong

- Errors happen frequently, we need a way to figure out what went wrong
- Use the MPI-START debugging variables:
  - Basic information:
    - I2G_MPI_START_VERBOSE=1
  - Debug information:
    - I2G_MPI_START_DEBUG=1
  - Full trace of the execution (normally too much, but useful for mpi-start developers)
    - I2G_MPI_START_TRACE=1
- Set them in the starter.sh or in the JDL:

```
Environment = {“I2G_MPI_START_VERBOSE=1”,…}
```

Debugging output: parallel-debug.jdl

```
************************************************************************
UID     =  ngifor080
HOST    =  gcsic177wn
DATE    =  Tue Jul 6 12:07:15 CEST 2010
VERSION =  0.0.66
************************************************************************
mpi-start [DEBUG  ]: dump configuration
mpi-start [DEBUG  ]: => I2G_MPI_APPLICATION=hello.sh
mpi-start [DEBUG  ]: => I2G_MPI_APPLICATION_ARGS=
mpi-start [DEBUG  ]: => I2G_MPI_TYPE=openmpi
[…]
mpi-start [INFO   ]: search for scheduler
mpi-start [DEBUG  ]: source /opt/i2g/bin/../etc/mpi-start/lsf.scheduler
mpi-start [DEBUG  ]: checking for scheduler support : sge
mpi-start [DEBUG  ]:  checking for $PE_HOSTFILE
mpi-start [INFO   ]: activate support for sge
mpi-start [DEBUG  ]:  convert PE_HOSTFILE into standard format
mpi-start [DEBUG  ]: dump machinefile:
mpi-start [DEBUG  ]: => gcsic177wn.ifca.es
mpi-start [DEBUG  ]: => gcsic177wn.ifca.es
mpi-start [DEBUG  ]: => gcsic177wn.ifca.es
mpi-start [DEBUG  ]: => gcsic177wn.ifca.es
mpi-start [DEBUG  ]: starting with 4 processes.
```

P3: Simple Parallel Jobs

- Crea el trabajo Grid fichero parallel.JDL y crea los ficheros starter.sh y hello.sh
- Consulta los recursos disponibles para el JDL y vuelca el resultado en un fichero de texto llamado “recursos.txt”.
- Lanza el trabajo en la vo “biomed” y monitorizalo hasta que este finalice.
- Descarga los ficheros resultado una vez finalizado en la carpeta “P3”.

P3: Simple Parallel Jobs

- En la carpeta "home/mpg/Evidencias/Seminario_MPI_Grid/P3" deben aparecer los siguientes archivos:
  - parallel.jdl
  - hello.sh
  - recursos.txt
  - Carpeta mpg_XXXXXX con los ficheros std.out y std.err

P4. Simple MPI Job

P4. Simple MPI Jobs

- Sitúate en la carpeta siguiente:

```
[ccgc@XXX~]$ cd /home/mpg/Evidencias/Seminario_MPI_Grid/P4
```

- Todos los ficheros que generes en esta práctica debes de guardarlo en esta carpeta para que consten como evidencia de su realización.

P4. simple MPI Job

1. Compile your application (the source is located at poliformaT)

```
$ mpicc cpi.c –o cpi
```

2. Change your JDL (from P3) and assign the name cpi.jdl:
3. Modify the JDL File

```
InputSandbox = {"starter.sh", "cpi"};
Arguments    = "cpi OPENMPI”;
```

4. Submit

Output

```
$ glite-wms-job-output --dir ./ https://wms01.ncg.ingrid.pt 9000/3hXhJD3eRRgiSLPqW3vSKg

Connecting to the service https://wms01.ncg.ingrid.pt:7443/glite_wms_wmproxy_server


================================================================================

                            JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://wms01.ncg.ingrid.pt:9000/3hXhJD3eRRgiSLPqW3vSKg
have been successfully retrieved and stored in the directory:
/home/enol/tests/eciencia/cpi/enol_3hXhJD3eRRgiSLPqW3vSKg

================================================================================

$ cat /home/enol/tests/eciencia/cpi/enol_3hXhJD3eRRgiSLPqW3vSKg/*
Process 3 on gcsic093wn: n=1
Process 2 on gcsic091wn: n=1
Process 1 on gcsic091wn: n=1
Process 0 on gcsic026wn: n=1
Using 16384 intervals
pi is approximately 3.1415926539002341, Error is 0.0000000003104410
wall clock time = 0.003501
```

P4. simple MPI Job

- En la carpeta “home/mpg/Evidencias/Seminario_MPI_Grid/P4” deben aparecer los siguientes archivos:
  - cpi.c
  - cpi
  - cpi.jdl
  - Starter.sh
  - Carpeta mpg_XXXXXX con los ficheros std.out y std.err

P5. Simple MPI Job & Pre-Hooks (Remote Compilation)

P5. Simple MPI Job & Pre-Hooks

- Sitúate en la carpeta siguiente:

```
[ccgc@XXX~]$ cd /home/mpg/Evidencias/Seminario_MPI_Grid/P5
```

- Todos los ficheros que generes en esta práctica debes de guardarlo en esta carpeta para que consten como evidencia de su realización.

P5. Simple MPI Job & Pre-Hooks

- What if your binary does not match the site environment?
- Remote compilation is needed
- Mpi-start hooks can help here!

P5. Simple MPI Job & Pre-Hooks
Compilation using hooks

pre-hook.sh

```
#!/bin/sh

pre_run_hook () {

  # Compile the program.
  echo "Compiling ${I2G_MPI_APPLICATION}"

  # Actually compile the program.
  cmd="mpicc ${MPI_MPICC_OPTS} -o ${I2G_MPI_APPLICATION} ${I2G_MPI_APPLICATION}.c"
  $cmd
  if [ ! $? -eq 0 ]; then
    echo "Error compiling program.  Exiting..."
    return 1
  fi

  # Everything's OK.
  echo "Successfully compiled ${I2G_MPI_APPLICATION}"

  return 0
}
```

P5. Simple MPI Job & Pre-Hooks

Compilation using hooks

Parallel_hooks.jdl

```
JobType          = "Normal";
nodeNumber       = 4;
Executable       = "starter.sh";
Arguments        = ”cpi OPENMPI";
InputSandbox     = {"starter.sh", “cpi.c”, “pre-hook.sh”};
StdOutput        = "std.out";
StdError         = "std.err";
OutputSandbox    = {"std.out","std.err"};
Requirements     =
Member("MPI-START”,
other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI”,
other.GlueHostApplicationSoftwareRunTimeEnvironment);
Environment      = {“I2G_MPI_PRE_RUN_HOOK=pre-hook.sh”};
```

P5. Simple MPI Job & Pre-Hooks

Compilation using hooks

- Crea el trabajo Grid fichero parallel_hooks.jdl, starter.sh y pre-hook.sh
- Lanza el trabajo en la vo “biomed” y monitoriza el trabajo hasta que este finalice.
- Descarga los ficheros resultado una vez finalizado en la carpeta “P5”.

P5. Simple MPI Job & Pre-Hooks
Evidencias

- En la carpeta "home/mpg/Evidencias/Seminario_MPI_Grid/P5" deben aparecer los siguientes archivos:
  - parallel_hooks.jdl
  - pre-hook.sh
  - starter.sh
  - Carpeta mpg_XXXXXX con los ficheros std.out y std.err

P6. Simple MPI Job & Post-Hooks

P6. Simple MPI Job & Post-Hooks

- Sitúate en la carpeta siguiente:

```
[ccgc@XXX~]$ cd /home/mpg/Evidencias/Seminario_MPI_Grid/P6
```

- Todos los ficheros que generes en esta práctica debes de guardarlo en esta carpeta para que consten como evidencia de su realización.

Getting output using hooks

post_hook.sh

```
#!/bin/sh

export OUTPUT_PATTERN="host_*.txt"
export OUTPUT_ARCHIVE=myout.tgz

# the first paramater is the name of a host in the
copy_from_remote_node() {
    if [[ $1 == `hostname` || $1 == 'hostname -f' || $1 == "localhost" ]]; then
        echo "skip local host"
        return 1
    fi
    CMD="scp -r $1:\"$PWD/$OUTPUT_PATTERN\" ."
    $CMD
}

post_run_hook () {
    echo "post_run_hook called"

    if [ "x$MPI_START_SHARED_FS" == "x0" -a "x$MPI_SHARED_HOME" != "xyes" ] ; then
        echo "gather output from remote hosts"
        mpi_start_foreach_host copy_from_remote_node
    fi

    echo "pack the data"
    tar cvzf $OUTPUT_ARCHIVE $OUTPUT_PATTERN
    return 0
}
```

Getting output using hooks (parallel hooks2.jdl)

Parallel_hooks2.jdl

```
JobType          = "Normal";
nodeNumber       = 4;
Executable       = "starter.sh";
Arguments        = ”cpi2 OPENMPI";
InputSandbox     = {"starter.sh", “cpi2.c”, “pre-hook.sh”,”post-hook.sh”};
StdOutput        = "std.out";
StdError         = "std.err";
OutputSandbox    = {"std.out","std.err“, “myout.tgz”};
Requirements     =
Member("MPI-START”,      other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI”, other.GlueHostApplicationSoftwareRunTimeEnvironment);
Environment        = {“I2G_MPI_PRE_RUN_HOOK=pre-hook.sh”,
                     “I2G_MPI_POST_RUN_HOOK=post-hook.sh”};
```

P6. Simple MPI Job & Post-Hooks

- Crea el trabajo Grid fichero parallel_hooks2.jdl, starter.sh, pre-hook.sh y post-hook.sh
- Lanza el trabajo en la vo “biomed” y monitoriza el trabajo hasta que este finalice.
- Descarga los ficheros resultado una vez finalizado en la carpeta “P6”.

Getting output using hooks

```
[…]
mpi-start [DEBUG  ]: Try to run post hooks at hooks.sh
mpi-start [DEBUG  ]: call post-run hook
-<START POST-RUN HOOK>----------------------------------------
post_run_hook called
pack the data
host_0.txt
host_1.txt
host_2.txt
host_3.txt
-<STOP POST-RUN HOOK>-----------------------------------------
```

P6. Simple MPI Job & Post-Hooks

- En la carpeta "home/mpg/Evidencias/Seminario_MPI_Grid/P6" deben aparecer los siguientes archivos:
  - parallel_hooks.jdl
  - pre-hook.sh
  - post-hook.sh
  - starter.sh
  - Carpeta mpg_XXXXXX con los ficheros std.out y std.err