Identificazione documento

Titolo	Tipo	Nome file
Guida GRID di primo livello	Documentazione	SIS_guida_grid_unina_v7

Approvazioni

	Nome	Data	Firma
Redatto da	Boccia, Monorchio	30/04/2009	
Revisionato da			
Approvato da	Barone	04/05/2009	

Variazioni

Versione	Data	Autore	Paragrafi modificati
1	30/04/2009	Boccia, Monorchio	
2	22/06/2009	Boccia, Monorchio	
3	14/06/2010	Pollio	1, 1.1, 1.2, 2.2
4	01/12/2010	Boccia	2, 2.1, 2.3, 3.1, 3.2, 3.3
5	25/01/2011	Boccia, Carracciuolo, Bottalico	Revisione completa della documentazione dovuta al passaggio da gLite 3.1 a gLite 3.2
6	08/02/2013	Boccia, Carracciuolo, Bottalico	Prima revisione della documentazione dovuta al passaggio da gLite 3.2 ad EMI2
7	09/02/14	Boccia, Bottalico, Campagna, Carracciuolo	Revisione completa della documentazione
8	08/06/2021	Carracciuolo	3.4
9	15/06/2023	Carracciuolo	3.2
10	09/04/2024	Carracciuolo	3.2

Indice dei contenuti

1 NOTE PRELIMINARI PER USUFRUIRE DELLE RISORSE DI UNINA DATACENTER.....1

2 PRIMO ACCESSO ALLE RISORSE.....1

3 UTILIZZO DELLE RISORSE.....5

4 RIFERIMENTI.....17

1 Note preliminari per usufruire delle risorse di UNINA DATACENTER

Per poter **accedere** alle risorse del DATACENTER è **indispensabile** essere in **possesso** di un **certificato personale**. Per tutte le informazioni relative alla richiesta di certificato personale e l'iscrizione alla Virtual Organization *unina.it*, fare riferimento alla guida "Come_accedere_alla_Grid_SCoPE" disponibile sul portale SCoPE. Una volta diventati utenti del DATACENTER è necessario prendere visione ed attenersi alle regole di buon utilizzo così come descritte nel documento [4].

2 Primo accesso alle risorse

L'accesso ai servizi della griglia avviene attraverso un unico punto di accesso configurato con il ruolo GRID di User Interface. È possibile sia avere una propria User Interface, sia utilizzare la User Interface messa a disposizione degli utenti.

RICHIESTA ACCOUNT

In quest'ultimo caso è necessario richiedere un account al CSI inviando una email all'indirizzo: scopeadmin@unina.it che abbia come subject: "Richiesta account sulla UI" e che contenga le seguenti informazioni:

- Nome e Cognome del richiedente
- Dipartimento di appartenenza e ruolo se strutturati oppure responsabile di attività se studenti/dottorandi/borsisti o collaboratori

A tale email il servizio utenti risponderà inviando la seguente email:

> Salve,
> abbiamo provveduto a creare l'account XXXXXXXXXX sulla User Interface ui.scope.unina.it
>
> Lo userName (o login) coincide con il nome dell'account, mentre la password è stata impostata agli ultimi 8 caratteri del suo codice fiscale (maiuscolo): le consigliamo, al primo accesso di cambiarla.
>
> E` disponibile online, all'indirizzo www.scope.unina.it nella sezione "Servizio Utenti", la guida GRID di primo livello con le procedure di base per l'utilizzo della nuova infrastruttura GRID.
>
> Per qualsiasi informazione e/o problema, può rivolgersi al Contact Center di Ateneo.
>
> Cordiali saluti

Per effettuare il cambio della password consigliato dai gestori si veda paragrafo 2.1.

RINNOVO ACCOUNT

La durata degli account sulle User Interface è di un anno. Per richiederne il prolungamento è sufficiente inviare una email a scopeadmin@unina.it che abbia come subject: "Richiesta di prolungamento durata account sulla UI" e che contenga le informazioni relative all'account.

Per utilizzare le risorse è necessario disporre del proprio certificato personale sull'UI di accesso. Di seguito vengono descritte in maniera specifica le modalità di accesso ai servizi della griglia.

2.1 Collegamento alla User Interface (UI)

In questa sezione verrà descritta la procedura per collegarsi alla UI (ui.scope.unina.it) configurata per l'accesso alla GRID.

L'ambiente operativo delle UI è Linux.

ACCESSO DA UN PC LINUX

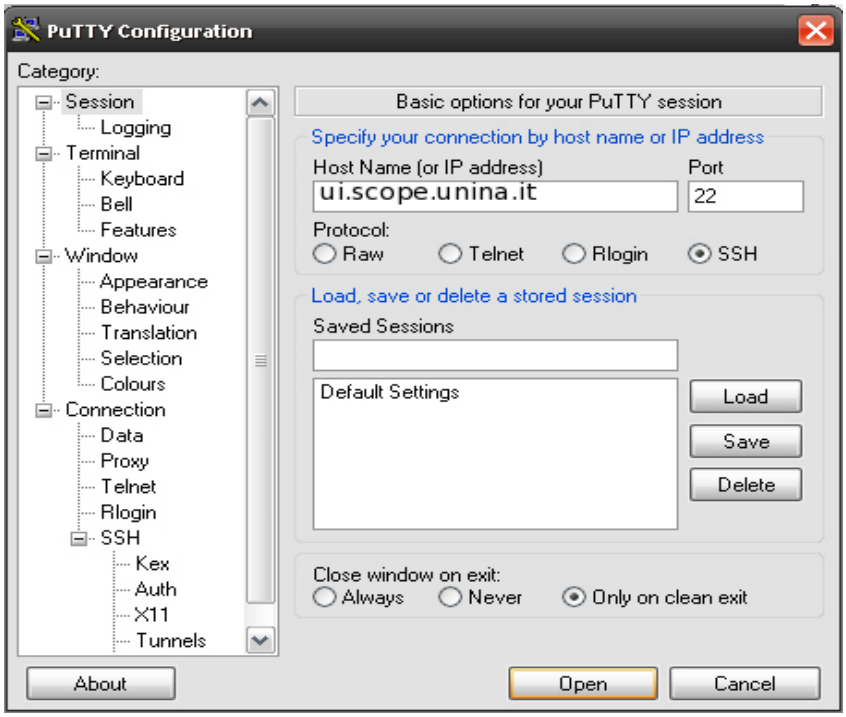
Per collegarsi alla UI si può utilizzare il comando ssh da un terminale.

ACCESSO DA UN PC WINDOWS

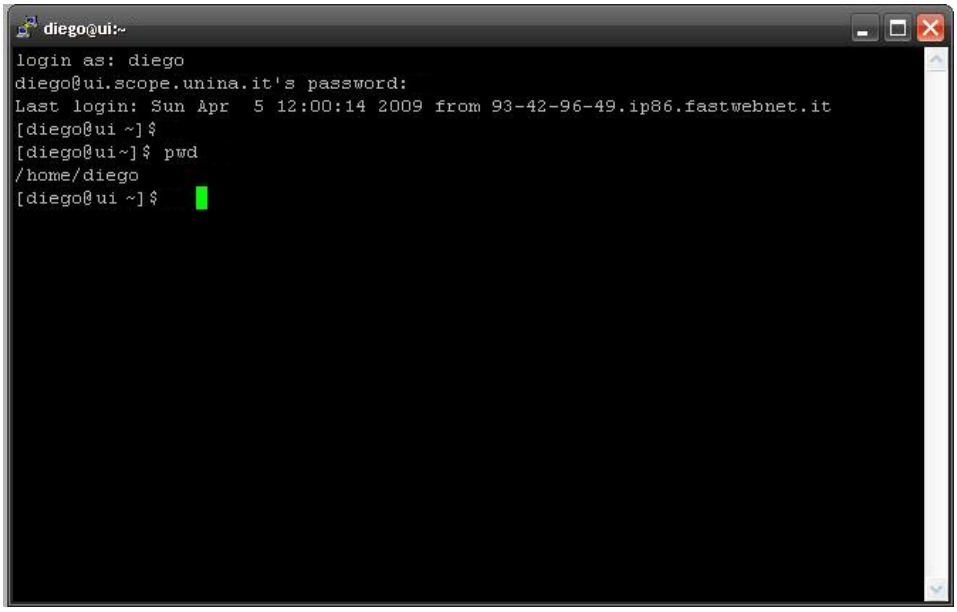
Per collegarsi alla UI è necessario utilizzare un emulatore di terminale, ad esempio "Putty", disponibile alla pagina web http://web.na.infn.it/fileadmin/compreso/ssh/ssh-pc.html

Le informazioni necessarie da inserire sono:

- Host Name: ui.scope.unina.it
- Port: 22
- Connection type: SSH



Il tasto "OPEN" apre il terminale come visualizzato nella successiva figura in cui vengono richiesti i propri user name e password di accesso alla UI.



A questo punto è possibile accedere alle proprie directory ed ai propri file sulla UI.

Per essere riconosciuti in GRID ed utilizzare le risorse di calcolo e storage dell'infrastruttura sarà necessario installare nella propria home directory il certificato personale ottenuto dalla CA.

CAMBIO DELLA PASSWORD

Al primo accesso è richiesto all'utente il cambio della password del proprio account rispetto a quella impostata dagli amministratori. Per farlo è sufficiente seguire le indicazioni.

2.2 Esportazione del certificato personale dal browser (Mozilla e Internet Explorer

Per le istruzioni su come eseguire l'esportazione del certificato personale dal browser, fare riferimento alla guida "Come_accedere_alla_Grid_SCoPE" [1] disponibile sul portale SCoPE.

2.3 Copia del certificato personale sulla User Interface (UI)

Una volta che il certificato personale sia stato esportato dal browser, occorre copiare il certificato in formato .p12 o .pfx nella propria home directory sulla UI.

COPIA DA UN PC LINUX

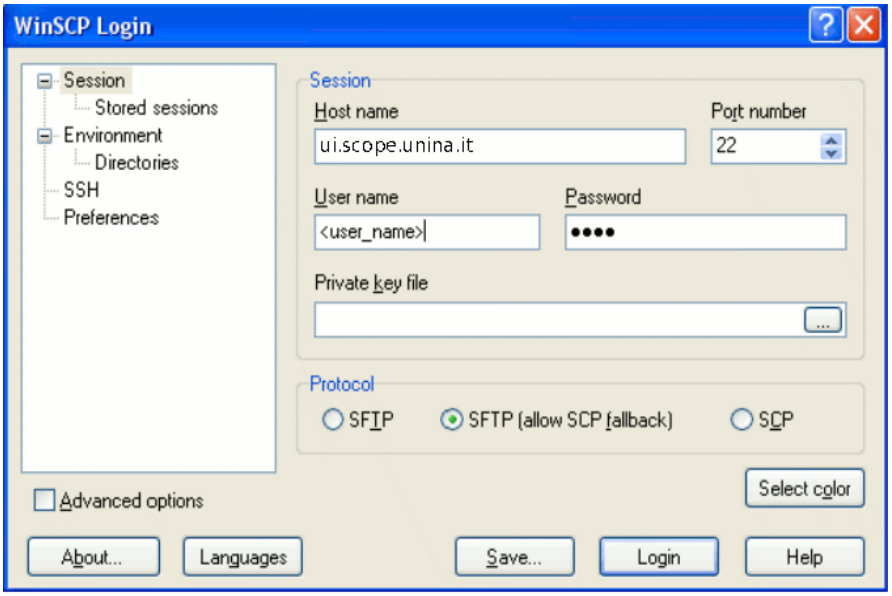
Se il certificato si trova su una macchina Linux, la copia va effettuata da terminale attraverso il comando scp, utilizzando la seguente sintassi:

```
$ scp  certificato_personale.p12  userName@ui.scope.unina.it:./
```

La stessa operazione può essere effettuata utilizzando un client grafico scp (ad es. gftp).

COPIA DA UN PC WINDOWS

Se il certificato si trova su una macchina Windows si può utilizzare il software WinSCP, disponibile alla pagina web http://web.na.infn.it/fileadmin/compreso/ssh/ssh-pc.html. Tale software consente di collegarsi alla UI e di trasferire il certificato alla home directory.



Una volta copiato il proprio certificato personale sulla UI, è necessario, a partire da esso generare due file (chiave privata e chiave pubblica in formato PEM) mediante l'utilizzo dei comandi seguenti:

```
$ openssl pkcs12 -nocerts  -in certificato_personale.p12  -out userkey.pem
$ openssl pkcs12 -clcerts  -nokeys -in certificato_personale.p12  -out usercert.pem
```

dove:

certificato_personale.p12 è l'input file in formato PKCS12.

userkey.pem è l'output file in formato PEM della chiave privata.

usercert.pem è l'output file in formato PEM del certificato.

Il primo dei due comandi suddetti richiede l'inserimento, da parte dell'utente, di due password: la prima è quella che è stata utilizzata dall'utente per esportare il certificato.p12 dal browser, mentre la seconda (che, per semplicità può coincidere con la prima) deve essere scelta dall'utente e inserita una prima e una seconda volta (come conferma). Tale password (definita anche in “grid passphrase”) sarà utilizzata per essere riconosciuti in grid.

Il secondo dei due comandi suddetti richiede l'inserimento, da parte dell'utente, di una sola password: quella che è stata utilizzata dall'utente per esportare il certificato.p12 dal browser.

Per quanto riguarda i file che contengono la chiave privata ed il certificato pubblico dell’utente, si devono utilizzare i nomi standard “usercert.pem” e “userkey.pem”. Tali file vanno poi spostati nella directory .globus mediante i comandi:

```
$ mkdir .globus
$ cp userkey.pem usercert.pem .globus/
```

A questo punto occorre cambiare i permessi a queste chiavi:

```
$ cd .globus/
$ chmod 644 usercert.pem
$ chmod 400 userkey.pem
```

3 Utilizzo delle risorse

3.1 Fase di autenticazione: generazione proxy delle credenziali

Comandi voms-proxy-init e voms-proxy-info

La sottomissione di un job su Grid necessita della creazione di un certificato proxy utente utilizzando il seguente comando, che fa anche richiesta della password (o “grid passphrase”) con la quale è stata generata la chiave privata (si veda paragrafo 2.3):

```
$ voms-proxy-init --voms unina.it

Enter GRID pass phrase: <digitare password>
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome Cognome
Creating temporary proxy ..................................................... Done
Contacting  voms01.scope.unina.it:15003 [/C=IT/O=INFN/OU=Host/L=Federico II/CN=voms01.scope.unina.it]
"unina.it" Done
Creating proxy ..................................... Done
Your proxy is valid until Sat Feb  9 04:55:33 2013
```

E’ possibile visualizzare le informazioni del proprio certificato proxy con il comando:

```
$ voms-proxy-info  -all

subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN= Nome Cognome /CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN= Nome Cognome
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN= Nome Cognome
type      : proxy
strength  : 1024 bits
path      : /tmp/x509up_u501
timeleft  : 11:59:32
key usage : Digital Signature, Key Encipherment, Data Encipherment
=== VO unina.it extension information ===
VO        : unina.it
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN= Nome Cognome
issuer    : /C=IT/O=INFN/OU=Host/L=Federico II/CN=voms01.scope.unina.it
attribute : /unina.it/Role=NULL/Capability=NULL
timeleft  : 11:59:31
uri       : voms01.scope.unina.it:15003
```

3.2 Scelta delle risorse da utilizzare per la computazione

L'infrastruttura Grid fornisce all’utente la scelta delle risorse di calcolo, rappresentate da un CE che gestisce un insieme di Worker Nodes, sui quali desidera sottomettere un determinato job e di storage SE, che gestisce una certa quantità di spazio disco.

La scelta delle risorse di calcolo per la sottomissione dei job va effettuata tenendo conto delle caratteristiche delle applicazioni che devono girare come la durata media dei job ed il tipo di calcolo (seriale, parallelo o multicore). Le risorse di calcolo sono assegnate, in modo equo e bilanciato, agli utenti grazie ad un sistema di code.

L’elenco delle code disponibili è reso disponibile all'utente mediante il comando:

```
$ lcg-infosites --vo unina.it ce
```

Elenco delle code disponibili per gli utenti unina.it:

Nome della coda	Limite Tempo di esecuzione (in ore)	Numero massimo di job accodabili per utente	Descrizione delle risorse
recasna-ce02.unina.it:8443/cream-pbs-long_pe6420	24	1000	12 Nodi biprocessore 14-core Intel Xeon, 376 GB RAM, connettività Eth
recasna-ce02.unina.it:8443/cream-pbs-long_pe7425	24	1000	35 Nodi biprocessore 16-core AMD EPYC, 251 GB RAM, connettività Eth
recasna-ce02.unina.it:8443/cream-pbs-short_pe6420	2	1000	
recasna-ce02.unina.it:8443/cream-pbs-short_pe7425	2	1000	
recasna-ce02.unina.it:8443/cream-pbs-unina_tv100	24	50	5 Nodi biprocessore 16-core AMD EPYC, 377 GB RAM, connettività Eth, scheda NVIDIA Tesla V100-PCIE-32GB

Nei prossimi paragrafi verrà descritta la procedura che un utente deve adottare per specificare le risorse che ha deciso di utilizzare per un determinato job.

L'elenco delle risorse di storage, invece, è visualizzabile mediante il comando:

```
$ lcg-infosites --vo unina.it se
```

3.3 Sottomissione di job sequenziali su Grid

Per eseguire job su Grid è necessario descrivere il job mediante una precisa sintassi, detta appunto JDL (Job Description Language). Ciò che è necessario fare è fornire all'infrastruttura Grid tutte le informazioni necessarie a specificare l'eventuale scelta delle risorse di calcolo, la quantità di risorse da utilizzare, l'input, l'output della propria esecuzione ed altri parametri. Di seguito è riportato un semplice esempio di quanto detto. Una trattazione completa della sintassi JDL è riportata in [2].

Il classico “Hello World”

Un esempio di job da sottomettere può essere il semplice “Hello World”, che usufruisce del comando /bin/echo. Per sottomettere un job su grid è necessario creare un file di configurazione in formato jdl (Job Description Language) che contiene tutte le informazioni utili al fine di individuare le risorse adatte all’esecuzione del job. Nel semplice esempio “Hello World” il file jdl sarà del tipo:

```
Arguments = "Hello World";
Executable = "/bin/echo";
StdOutput = "message.txt";
StdError = "stderror";
Requirements = RegExp("emi2-ce0[1-2].scope.unina.it:8443/cream-pbs-unina_short",
other.GlueCEUniqueID);
OutputSandbox = {"message.txt","stderror"};
```

“Executable” indica l’eseguibile presente su macchina che dovrà effettuare il calcolo, “Arguments” indica gli argomenti dell’eseguibile, “StdOutput” e “StdError” indicano i file dove verranno rediretti rispettivamente lo standard output e lo standard error. “OutputSandbox” indica il nome del file che viene restituito da Grid in output. Il tag “Requirements” serve invece a selezionare le risorse di calcolo sulla base di alcune indicazioni. Quella specificata nell'esempio, richiede l'utilizzo della coda unina_short di un CE a scelta tra emi2-ce01 e emi2-ce02, lasciando al middleware il compito di selezionare, tra i due CE, quello al momento più scarico.

Per verificare la sintassi di un JDL file personale, è sempre opportuno, la prima volta eseguire il comando:

```
$ glite-wms-job-list-match -a Hello.jdl
```

```
Connecting to the service https://wms01.scope.unina.it:7443/glite_wms_wmproxy_server


==========================================================================
=     COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been found:
            *CEId*
-           emi2-ce02.scope.unina.it:8443/cream-pbs-unina_short
-           emi2-ce01.scope.unina.it:8443/cream-pbs-unina_short
==========================================================================
```

E’ possibile sottomettere il job utilizzando il comando:

```
$ glite-wms-job-submit -a Hello.jdl
```

L’opzione –a garantisce l'operazione di delega del proxy al wms (Workload Management System) ed è necessaria affinché il job venga correttamente sottomesso

Il comando glite-wms-job-submit restituirà il seguente output che contiene l'ID del job come nell’esempio:

```
Connecting to the service https://wms01.scope.unina.it:7443/glite_wms_wmproxy_server
====================== glite-wms-job-submit Success ======================
The job has been successfully submitted to the WMProxy
Your job identifier is:
            https://wms01.scope.unina.it:9000/q39KjuorIWv185AhHFzCTg
==========================================================================
```

Il formato del jobID è:

```
https://<LB_hostname>[:<port>]/<unique_string>
```

dove <unique_string> è una stringa univocamente assegnata al job e <LB_hostname> è l’host-name del server di Logging & Bookkeeping (LB) utilizzato per il JOB.

Per visionare lo stato del job occorre utilizzare il numero ID ad esso associato con il comando:

```
$ glite-wms-job-status https://wms01.scope.unina.it:9000/q39KjuorIWv185AhHFzCTg
```

L’output che segue corrisponde allo stato “Running” del job

```
**************************************************************
BOOKKEEPING INFORMATION:
Status info for the Job : https://wms01.scope.unina.it:9000/q39KjuorIWv185AhHFzCTg
```

```
Current Status:     Running
Status Reason:      Job successfully submitted to Globus
Destination:        emi2-ce02.scope.unina.it:8443/cream-pbs-unina_short
Submitted:          Tue Jan 25 16:17:57 2011 CET
**************************************************************
```

Se il jobs viene completato con successo, mediante il comando:

```
$ glite-wms-job-status https://wms01.scope.unina.it:9000/q39KjuorIWv185AhHFzCTg
```

si ottiene il seguente output:

```
**************************************************************
BOOKKEEPING INFORMATION:
Status info for the Job : https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
Current Status:     Done (Success)
Logged Reason(s):
    -
    - Job terminated successfully
Exit code:          0
Status Reason:      Job terminated successfully
Destination:        emi2-ce02.scope.unina.it:8443/cream-pbs-unina_short
Submitted:          Tue Apr  7 14:52:24 2009 CEST
**************************************************************
```

L'opzione -o <file path> di glite-wms-job-submit permette agli utenti di specificare un file nel quale aggiungere il jobID del job sottomesso. Questo file può essere dato come argomento ad altri comandi di job management per effettuare operazioni su più di un job con un singolo comando ed è un modo conveniente per tener traccia dei jobs sottomessi. Supponiamo ad esempio di aver sottomesso due jobs, raccogliendo i jobID nel file "prova_JOBLIST" con i comandi:

```
$ glite-wms-job-submit -a -o prova_JOBLIST Ciao.jdl
$ glite-wms-job-submit -a -o prova_JOBLIST Hola.jdl
```

il contenuto del file "prova_JOBLIST" è il seguente:

```
$ more prova_JOBLIST

###Submitted Job Ids###
https://wms01.scope.unina.it:9000/0y9-abow_HgUvCQ_l-sfFQ
https://wms01.scope.unina.it:9000/VCAZHjCUfTbObmFQWl0-oQ
```

Possiamo passare in input il file "prova_JOBLIST" al comando glite-wms-job-status attraverso l'opzione –i:

```
$ glite-wms-job-status -i prova_JOBLIST
```

ottenendo l'output:

```
--------------------------------------------------------------
1 : https://wms01.scope.unina.it:9000/0y9-abow_HgUvCQ_l-sfFQ
2 : https://wms01.scope.unina.it:9000/VCAZHjCUfTbObmFQWl0-oQ
a : all
q : quit
--------------------------------------------------------------

Choose one or more jobId(s) in the list - [1-2]all:
```

Possiamo dunque selezionare i jobs della lista per i quali vogliamo monitorare lo stato.

Nel messaggio relativo al BOOKKEEPING INFORMATION, ottenuto con il comando, l'argomento "Destination" indica che il job viene indirizzato verso il Computing Element (CE) emi2-ce02.scope.unina.it. Nel caso in cui, come in precedenza, il file jdl non contenga alcun requisito riguardo la scelta del CE da utilizzare per la computazione, il job girerà sulla macchina con capacità computazionale migliore scelta da Grid in maniera trasparente all'utente.

Nel caso in cui invece l'utente desideri eseguire il job su uno specifico CE ed in particolare su una specifica coda di job, in base ai requisiti della propria applicazione, leggere il prossimo paragrafo.

Per ottenere i file di output di un job si può utilizzare il comando glite-wms-job-output passandogli come argomento il jobID opportuno:

```
$ glite-wms-job-output https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
```

oppure

```
$ glite-wms-job-output -i IDs
```

```
Connecting to the service https://wms01.scope.unina.it:7443/glite_wms_wmproxy_server
================================================================================
JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
https://wms01.scope.unina.it:9000/q39KjuorIWv185AhHFzCTg
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/vania_q39KjuorIWv185AhHFzCTg
================================================================================
```

A questo punto l'output sarà copiato nella directory /tmp/jobOutput/vania_q39KjuorIWv185AhHFzCTg

Il contenuto del file di output "message.txt" sarà la stringa "Hello World":

```
$ ls /tmp/jobOutput/vania_q39KjuorIWv185AhHFzCTg
```

```
message.txt  sterror
```

```
$ more /tmp/jobOutput/vania_q39KjuorIWv185AhHFzCTg
/message.txt
```

```
Hello World
```

Un esempio più complesso (esportazione dell'eseguibile e degli input file)

Nell'esempio "Hello World" il job che viene eseguito fa uso solo del comando di sistema /bin/echo che risiede sui worker nodes sui quali il job verrà eseguito (oltre che sulla User Interface da cui viene sottomesso).

Supponiamo ora di aver compilato sulla UI un programma in linguaggio C++ e di avere un eseguibile chiamato simulazione.exe in /home/<user_name>/SIMULAZIONE. Affinché il job possa essere eseguito è necessario che il file eseguibile ed eventualmente tutti i file di input, vengano copiati sui Worker Nodes.

Per fare ciò, il middleware mette a disposizione il campo "InputSandbox" nel file JDL.

Il campo "InputSandbox" specifica tutti i file eseguibili (ed eventualmente i file di input dell'eseguibile) da inviare (assieme al file JDL) in input al Worker Node (WN) che eseguirà il job.

Se l'esecuzione del job necessita solo del file eseguibile, occorre utilizzare tale sintassi:

```
Executable = "Simulazione.exe";
StdOutput = "mc.out";
StdError = "mc.err";
Arguments = "aaa";
InputSandbox = "/home/<user_name>/SIMULAZIONE/Simulazione.exe";
OutputSandbox = {"mc.out","mc.err"};
```

Se l'esecuzione del job necessita anche di file di dati in input, occorre utilizzare tale sintassi:

```
InputSandbox = {"/home/<user_name>/SIMULAZIONE/Simulazione.exe";
"/home/<user_name>/SIMULAZIONE/input1";"…"}
```

Nel caso in cui il job necessiti di più argomenti utilizzare tale sintassi nel file jdl:

```
Arguments = "arg1 arg2 …";
```

L'argomento "InputSandbox" è necessario nel caso in cui la macchina che deve eseguire un determinato job non contiene l'eseguibile o non contiene i files di input necessari.

NOTA BENE: Le dimensioni massime consentite per spostare file mediante gli attributi InputSandbox e OutputSandbox sono rispettivamente di 10MB e 100MB. Per spostare file di dimensioni maggiori si devono utilizzare gli opportuni servizi (e relativi comandi) per la gestione dei dati (SE ed LFC, cfr. Sezione 3.4) .

Sottomissione di job con specifica dei requisiti per le risorse da utilizzare.

Gli utenti possono specificare dei requisiti per le risorse che intendono utilizzare per l'esecuzione di un job. Ad esempio un utente che vuole eseguire un job di breve durata troverà conveniente utilizzare le code "short" che, come visto nel paragrafo 3.2, impongono un limite massimo alla durata dei job sottomessi.

L'opportunità di scegliere una specifica coda opportunità viene garantita inserendo nel file jdl il requisito:

```
Requirements=RegExp("emi2-ce0[1-2].scope.unina.it:8443/cream-pbs-unina_short", other.GlueCEUniqueID);
```

dove:

Requirements: indica la presenza di un requisito nel file jdl.

other.GlueCEUniqueID: indica che il requisito si riferisce alla scelta di uno specifico CE.

3.3 Strumenti per il monitoraggio dell'esecuzione dei job lunghi

Per meglio seguire la fase di esecuzione di job lunghi è consentito abilitare il meccanismo di *Perusal* previsto dal middleware EMI per ottenere versioni temporanee dei file inclusi nella OutputSandbox. Se, ad esempio, insieme ai risultati della computazione, il job produce già un file di log durante la sua esecuzione e se tale file è richiesto esplicitamente nell'OutputSandbox, per abilitare tale meccanismo sul tale file di log è necessario:

- Aggiungere le linee seguenti nel file JDL:

```
PerusalFileEnable = true;
PerusalTimeInterval = 1800;
```

- Dopo aver sottommesso il job, eseguire il comando:

```
glite-wms-job-perusal --set -f <nome del file di log> <jobID>
```

- Una volta abilitato il meccanismo di Perusal e sottomesso il job, per recuperare la versione temporanea del file di log è necessario eseguire il comando:

```
glite-wms-job-perusal --get -f <nome del file di log> <jobID>
```

Per maggiori dettagli si faccia riferimento alla guida utente ufficiale EMI [3].

3.4 Gestione dei dati

Gli esempi finora illustrati prevedono l'uso dei campi InputSandbox e OutputSandbox che forniscono il trasferimento di files di piccole dimensioni necessari per l'esecuzione del job e per il controllo dei suoi risultati. I files di output specificati in OutputSandbox vengono trasferiti al termine del job sul WMS e possono essere scaricati localmente sulla UI attraverso il comando

```
glite-wms-job-output
```

come descritto in precedenza. È importante sottolineare che il WMS non fornisce un servizio di storage vero e proprio e dunque impone un limite massimo di 100 MB sulle dimensioni dei file di output. Inoltre i file scritti sul WMS vengono periodicamente cancellati dal sistema (con cadenza settimanale) e dunque l'utente deve avere cura di prelevare tempestivamente l'output a cui è interessato. Qualora si decida di sottomettere un numero molto elevato di job, tale che le dimensioni complessive dei file di output da scrivere raggiungano le decine di GB, è buona norma destinare l'output su uno Storage Element e contestualmente registrarlo con nome logico (per il recupero successivo) sul servizio di LFC (Logical File Name). La procedura per scrivere dati sull'SE e registrarli sull'LFC è illustrata nel seguente esempio.

I membri della VO unina.it hanno i permessi di scrittura e di lettura nell'area "virtuale" dell'LFC: /grid/unina.it. Ciò significa che i Logical File Names (LFN) dei file che verranno scritti sugli Storage Elements saranno caratterizzati dal path:

```
/grid/unina.it/<sub_path>
```

La gestione dei file può avvenire attraverso i comandi lfc-* ed lcg-*.

Per creare una directory in cui scrivere un file è necessario utilizzare il comando:

```
$ lfc-mkdir  /grid/unina.it/Testdir
```

Supponiamo di voler copiare il file Ciao.txt dalla UI ad uno Storage Element e di volerlo registrare sul Logical File Catalogue. Quest'operazione può essere effettuata attraverso il comando lcg-cr:

```
$ lcg-cr Ciao.txt --vo unina.it -l lfn://grid/unina.it/Testdir/Ciao.txt
```

Dove l'opzione il Logical File Name "lfn://grid/unina.it/ Testdir /Ciao.txt" viene assegnato attraverso l'opzione –l mentre l'opzione "--vo unina.it" specifica la VO di appartenenza. Siccome non è stato specificato uno SE sul quale scrivere il file, esso verrà assegnato dal sistema. L'output che riceveremo è il seguente:

```
[BDII] topbdii.scope.unina.it:2170: Warning, no GlueVOInfo information found about tag '(null)' and SE
'se01.scope.unina.it'
guid:160c18c3-5105-4919-b700-a2a9b79e8297
```

da cui si evince che il file è stato scritto sullo SE se01.scope.unina.it e che gli è stato assegnato il guid (Grid Unique Identifier) 160c18c3-5105-4919-b700-a2a9b79e8297.

Per specificare lo SE su cui scrivere il file si può utilizzare l'opzione –d :

```
$ lcg-cr Ciao.txt --vo unina.it -l lfn://grid/unina.it/Testdir/Ciao_2.txt -d se01.scope.unina.it
[BDII] topbdii.scope.unina.it:2170: Warning, no GlueVOInfo information found about tag '(null)' and SE
'se01.scope.unina.it'
guid:24971448-60cc-4ace-b916-cfd0eac5515f
```

La lista dei files presenti all'interno di un certo path si ottiene attraverso il comando lcg-ls:

```
$ lcg-ls -l lfn:/grid/unina.it/Testdir
-rw-rw-r--   1   105   101      10 Ciao.txt
-rw-rw-r--   1   105   101      10 Ciao_2.txt
```

Per poter leggere copiare un file dallo Storage Element alla User Interface si può utilizzare il comando lcg-cp:

```
$ lcg-cp  lfn:/grid/unina.it/Testdir/filename1 filename2
```

Il file è ora disponibile localmente sulla UI e può essere letto dall'utente.

Di seguito descriveremo come usare il sistema di storage per ovviare ai limiti imposti sulle dimensioni delle InputSandbox ed OutputSandbox. Supponiamo di voler analizzare un file di grosse dimensioni il cui nome è input.txt e di volere memorizzare l'output di tale analisi nel file con nome output.txt, si può procedere effettuando le seguenti azioni:

1. Memorizzare il file input.txt sul sistema di storage mediante il comando lcg-cr

   ```
   lcg-cr input.txt -l lfn://grid/unina.it/Testdir/input.txt
   ```

2. scrivere un file di scrip bash test.sh con almeno le linee seguenti

   ```
   lcg-cp lfn://grid/unina.it/Testdir/input.txt file:$PWD/input.txt

   <comando del programma di analisi>

   lcg-cr output.txt -l lfn://grid/unina.it/Testdir/output.txt
   ```

3. modificare il tag Executable del il file JDL prova.jdl come di seguito

   Executable = "test.sh";

   eliminando gli eventuali riferimenti ai file input.txt e output.txt da InputSandbox ed OutputSandbox
4. Sottomettere il job relativo descritto dal nuovo file prova.jdl
5. Attendere la terminazione del job prima di trasferire il file output.txt dallo storage sulla UI medinate il comando lcg-cp

   ```
   lcg-cp lfn://grid/unina.it/Testdir/output.txt file:$PWD/output.txt
   ```

3.5 Sottomissione di job per il calcolo parallelo (job MPI)

In questa sezione sarà illustrata la procedura per la sottomissione di job paralleli sull'infrastruttura di calcolo di Scope. Rispetto alla sottomissione di job non paralleli, sarà necessario introdurre una modifica al file JDL ed utilizzare un wrapper che provveda a copiare via ssh l'eseguibile MPI sui vari nodi allocati per l'esecuzione.

Proponiamo di seguito un esempio di job parallelo che fa uso dell'implementazione OPENMPI dello standard 2 di MPI.

Creiamo una directory di test:

```
$ mkdir MPI_TEST
$ cd MPI_TEST
```

Copiamo nella directory appena creata i seguenti file:

hello.c (esempio di codice sorgente che produce la stampa dell'hostname da parte di tutti i processi coinvolti nel calcolo)

```
# /* C Example */
#include <stdio.h>
#include <mpi.h>
int main (argc, argv)
     int argc;
     char *argv[];
{
     int rank, size;
     char hostname[256];

     MPI_Init (&argc, &argv);

     /* starts MPI */
     MPI_Comm_rank (MPI_COMM_WORLD, &rank);
     /* get current process id */
     MPI_Comm_size (MPI_COMM_WORLD, &size);
     /* get number of processes */
     gethostname(hostname,255);
     /* get the host names */

    printf( "Hello world from process %d of %d on node %s\n", rank, size, hostname );
    MPI_Finalize();
    return 0;
}
```

mpi-hooks.sh

```
#!/bin/sh

# This function will be called before the MPI executable is started.
# You can, for example, compile the executable itself.
#
pre_run_hook () {

  echo "Executing pre 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"
  echo $cmd
  $cmd
  if [ ! $? -eq 0 ]; then
     echo "Error compiling program. Exiting..."
     exit 1
  fi
# Everything's OK.
  echo "Successfully compiled ${I2G_MPI_APPLICATION}"
  echo "Finished the pre hook."
  return 0
```

```
}

# This function will be called before the MPI executable is finished.
# A typical case for this is to upload the results to a storage element.
post_run_hook () {
  echo "Executing post hook."
  echo "Finished the post hook."

  return 0
}
```

mpi-start-wrapper.sh

```
#!/bin/bash

# Pull in the arguments.
MY_EXECUTABLE=`pwd`/$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

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
export I2G_MPI_PRE_RUN_HOOK=mpi-hooks.sh
export I2G_MPI_POST_RUN_HOOK=mpi-hooks.sh

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

chmod 755 $MY_EXECUTABLE

# Invoke mpi-start.
$I2G_MPI_START
```

wrapper-OPENMPI.jdl (file JDL per la sottomissione)

```
Executable = "mpi-start-wrapper.sh";
Arguments = "hello OPENMPI";
CpuNumber = 64;
StdOutput = "mpi.out";
```

```
StdError = "mpi.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh","hello.c"};
OutputSandbox = {"mpi.err","mpi.out"};
Requirements = Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment) &&
               Member("OPENMPI", other.GlueHostApplicationSoftwareRunTimeEnvironment)   &&
               RegExp("emi2-ce0[1-2].scope.unina.it:8443/ cream-pbs-unina_short ",other.GlueCEUniqueID);
RetryCount = 0;
```

Si notino nel file JDL (rispetto alla guida precedente)

- **La scomparsa del tag "JOBTYPE=MPICH"**
- **La scelta, effettuata mediante i Requirements, della risorsa che supporta MPI**

```
$ glite-wms-job-submit -a -o jobID.txt wrapper-OPENMPI.jdl
```

Con le impostazioni adottate in wrapper-OPENMPI.jdl il codice sorgente viene copiato e compilato sui worker nodes.

Al termine del job, l'output può essere recuperato nel modo usuale:

```
$ glite-wms-job-output -i jobID.txt
```

Nell'esempio proposto "JobType" è l'attributo che serve ad indicare al middleware che il job è di tipo parallelo. Le informazioni riguardo all'implementazione di MPI scelta dall'utente per l'esecuzione vengono inserite nell'attributo Arguments. In Arguments oltre all'implementazione di MPI da utilizzare per l'esecuzione, va specificato anche il nome dell'eseguibile e il numero di processori da utilizzare. Atro attributo importante è CpuNumber che rappresenta il numero di processori della risorsa che si richiede di allocare (in generale deve essere >= del numero di processori specificato in Arguments).

3.6 Rinnovo automatico delle credenziali (myproxy service)

Il proxy delle credenziali (proxy locale alla UI) generato con il comando voms-proxy-init ha una durata di default di 12 ore. I job sottomessi dunque possono stare in stato di "Running" solo per tale periodo, al termine del quale vengono distrutti nonostante non abbiano terminato i propri calcoli.

Tale fenomeno rappresenterebbe una limitazione per i job di lunga durata che vanno oltre l'intervallo di tempo in cui il proxy locale rimane valido. Il problema è risolto mediante l'utilizzo del servizio MyProxy, che consente il rinnovo automatico del proxy utente.

La procedura di sottomissione dei job è simile a quella già descritta nei paragrafi precedenti, con l'aggiunta dei comandi che gestiscono il myproxy delle credenziali.

Il primo passo prevede dunque, come già visto, la creazione del proxy delle credenziali locale alla UI mediante il comando:

```
$ voms-proxy-init --voms unina.it
```

```
Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome Cognome
Creating temporary proxy ..................................... Done
```

```
Contacting  voms01.scope.unina.it:15003 [/C=IT/O=INFN/OU=Host/L=Federico II/CN=voms01.scope.unina.it]
"unina.it" Done
Creating proxy ..... Done
Your proxy is valid until Thu Apr  9 02:41:54 2009
```

A questo punto è possibile generare il myproxy delle credenziali mediante il comando:

```
$ myproxy-init -d -n --voms unina.it
```

Il comando myproxy-init memorizza le credenziali sul myproxy server rendendole disponibili di default per 168 ore durante le quali le credenziali utente verranno rinnovate automaticamente allo scadere di ciascun proxy utilizzato dal WMS.

L'opzione –d istruisce il server ad associare il DN (Distinguished Name) dell'utente al proxy e l'opzione –n evita l'uso di una password per accedere al proxy così che il WMS può effettuare il rinnovo automaticamente.

L'output restituito è il seguente:

```
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome Cognome
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Proxy Verify OK
Your proxy is valid until: Wed Apr 15 14:48:07 2009
A proxy valid for 168 hours (7.0 days) for user /C=IT/O=INFN/OU=Personal Certificate/L=Federico
II/CN=Nome Cognome now exists on myproxy01.scope.unina.it.
```

Le informazioni del proprio certificato myproxy vengono visualizzate con il comando:

```
$ myproxy-info --d
```

```
username: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome Cognome
owner: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome Cognome
  timeleft: 167:51:50  (7.0 days)
```

Nel caso in cui si voglia estendere la durata del myproxy usare l'opzione –c hours in questo modo:

```
$ myproxy-init -d -n -c 200
```
(estensione a 200 ore)

```
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome Cognome
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Proxy Verify OK
Your proxy is valid until: Fri Apr 17 00:22:40 2009
A proxy valid for 200 hours (8.3 days) for user /C=IT/O=INFN/OU=Personal Certificate/L=Federico
II/CN=Nome Cognome now exists on myproxy01.scope.unina.it.
```

Nel caso in cui si voglia distruggere il myproxy delle credenziali eseguire il comando:

```
$ myproxy-destroy --d
```

```
Default MyProxy credential for user /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Nome
Cognome was successfully removed.
```

Una volta generato il myproxy, è possibile sottomettere job, monitorarli e importarne l'output attraverso i comandi glite-wms-job-submit, glite-wms-job-status e glite-wms-job-output come descritto nei paragrafi precedenti. Un fattore importante da ricordare è che non è possibile eseguire questi tre comandi quando il proxy delle credenziali locale alla UI (generato con voms-proxy-init) è scaduto. In questo caso dunque, l'utente che desideri eseguire uno di questi tre comandi deve rigenerare il proxy delle credenziali locali alla UI con il solito comando voms-proxy-init, senza occuparsi di ricreare il myproxy delle credenziali, a meno che quest'ultimo non sia scaduto.

Infine nell'intervallo di tempo in cui il myproxy delle credenziali sia valido, l'utente ha la possibilità di eseguire qualsiasi operazione sul proxy delle credenziali (voms-proxy-destroy, voms-proxy-init), non determinando la distruzione dei job in running, poiché tali comandi non hanno effetto sui proxy "remoti" (sul WMS e sul MYPROXY).

4 Riferimenti

[1] "Come_accedere_alla_Grid_SCoPE":

http://www.scope.unina.it/servizioutenti/docs/Come_accedere_alla_Grid_SCoPE_V2.pdf

[2] "Job Description Language Attributes Specification":

https://edms.cern.ch/file/590869/1/WMS-JDL.pdf

[3] "EMI 2 User Guide" Guida utente ufficiale- reperibile all'URL:

https://twiki.cern.ch/twiki/pub/EMI/EMIui/EMI_UI_v2_0_0_1.pdf

[4] "Servizi erogati dal Data Center e livelli di servizio del supporto utenti" - reperibile all'URL:

http://www.scope.unina.it/servizioutenti/docs/SIS_sla_v3.pdf