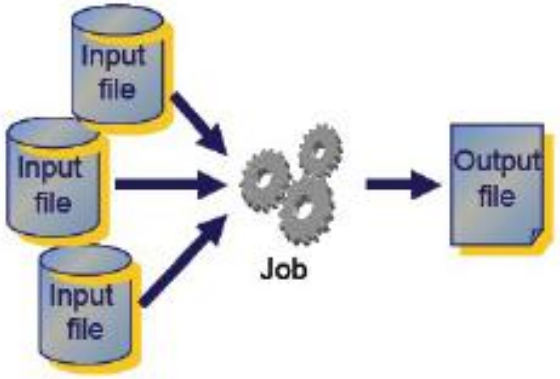
Laboratory: Hands-on using EGEE Grid and gLite middleware

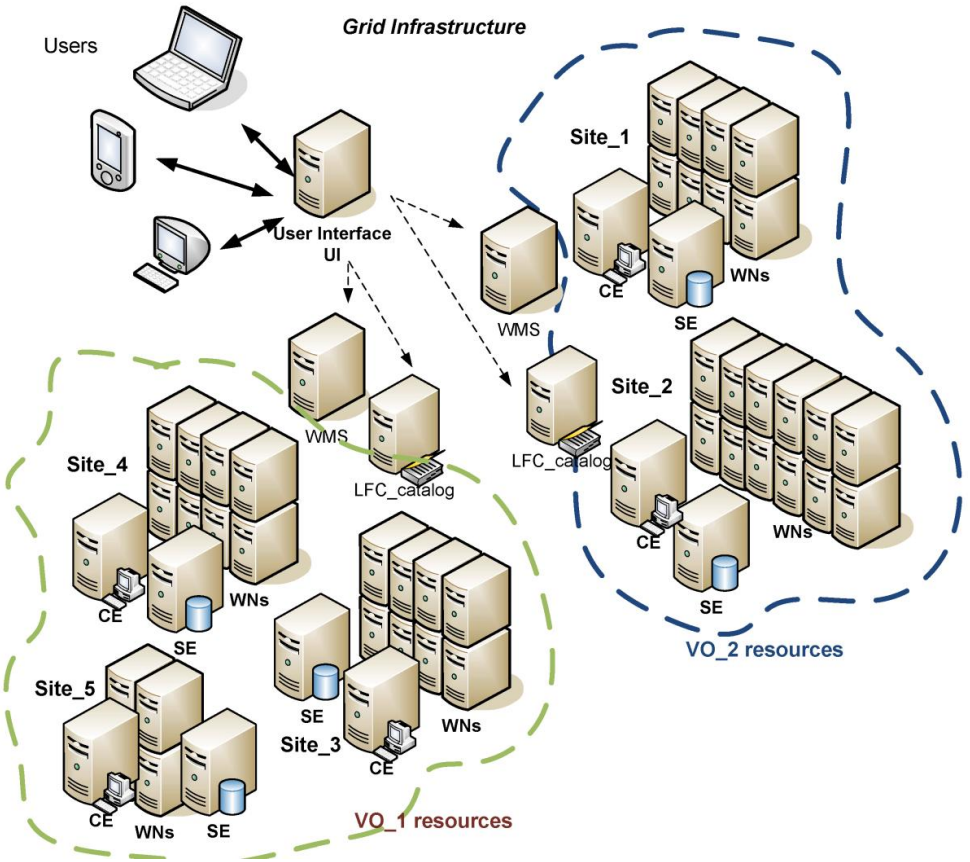
Athanasia Asiki
aassiki@cslab.ece.ntua.gr
Computing Systems Laboratory,
National Technical University of Athens

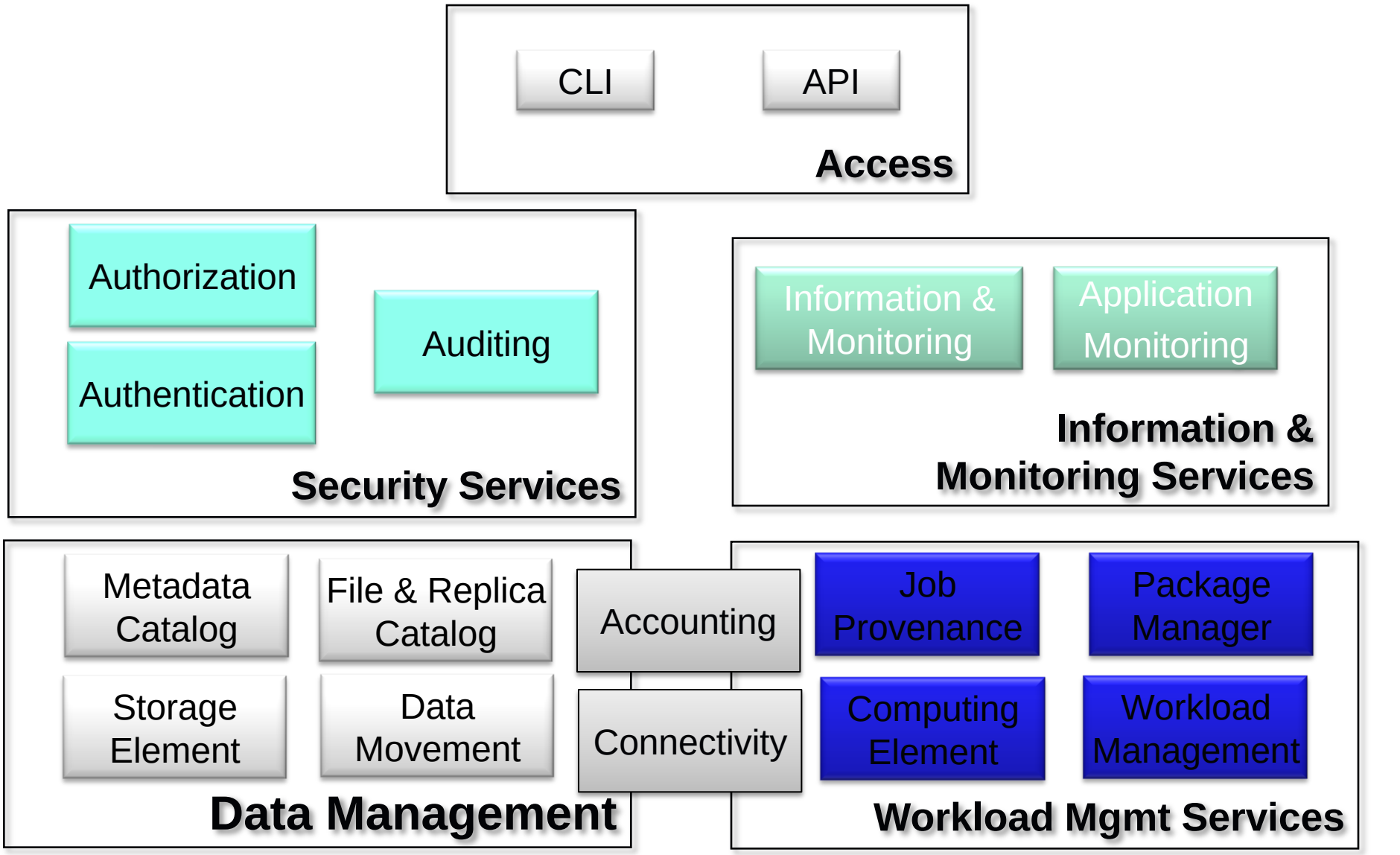
Application's structure



- **The execution of a *typical* Grid application follows this scenario:**
  - The user submits its application's job to the "Grid"
  - The job is being executed
  - The job's execution may include the processing of one or more **Input Files** stored in a Storage node
  - The job may produce one or more **Output Files**
  - The **Output Files** can be stored somewhere in the Grid system (perhaps in the Storage Element or in the User Interface)
  - The User can access the **Output Files** using the corresponding Grid mechanisms

A typical structure of a Grid platform



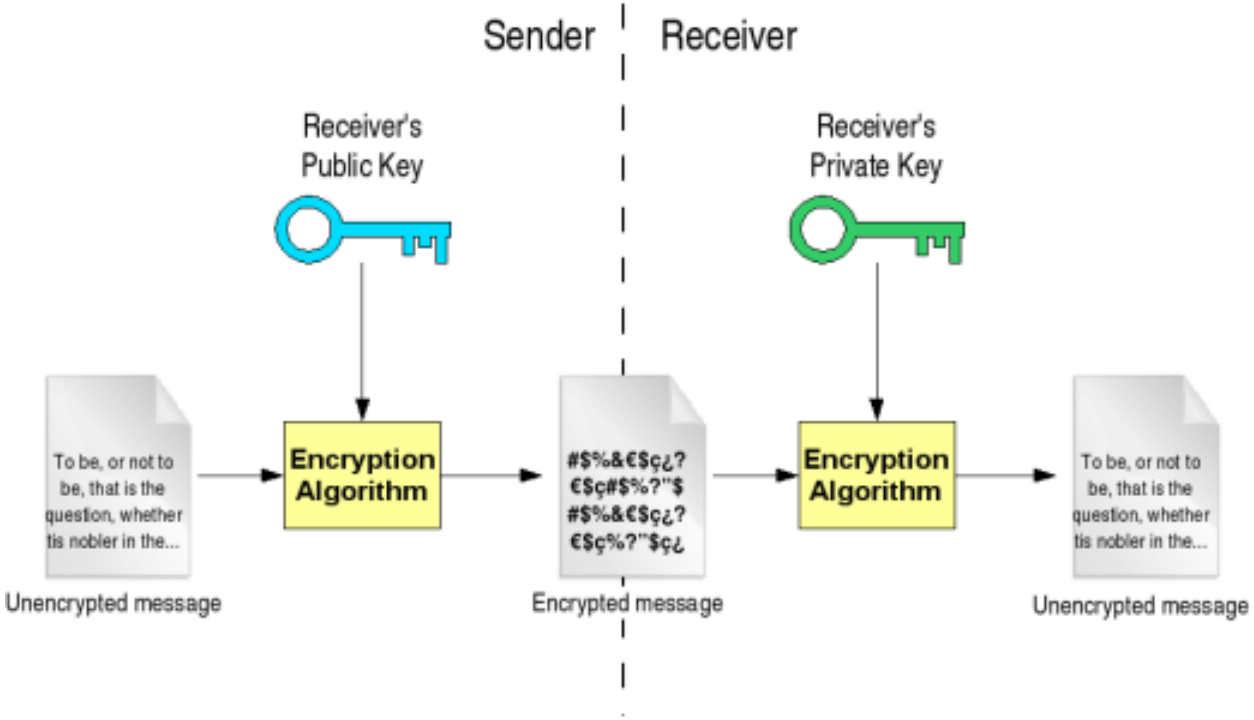

CLI
API
Access
Authorization
Authentication
Auditing
Security Services
Information & Monitoring
Application Monitoring
Information & Monitoring Services
Metadata Catalog
File & Replica Catalog
Storage Element
Data Movement
Accounting
Connectivity
Data Management
Job Provenance
Package Manager
Computing Element
Workload Management
Workload Mgmt Services

Basic gLite components

- **Security**
  - Virtual Organization Server (VOMS)
  - MyProxy server (Proxy)
- **Information System (IS)**
- **Job handling**
  - Workload Management System (WMS)
  - Logging & Bookkeeping (LB)
- **Data Management**
  - File Catalog
  - File Transfer Service
  - File Placement Service

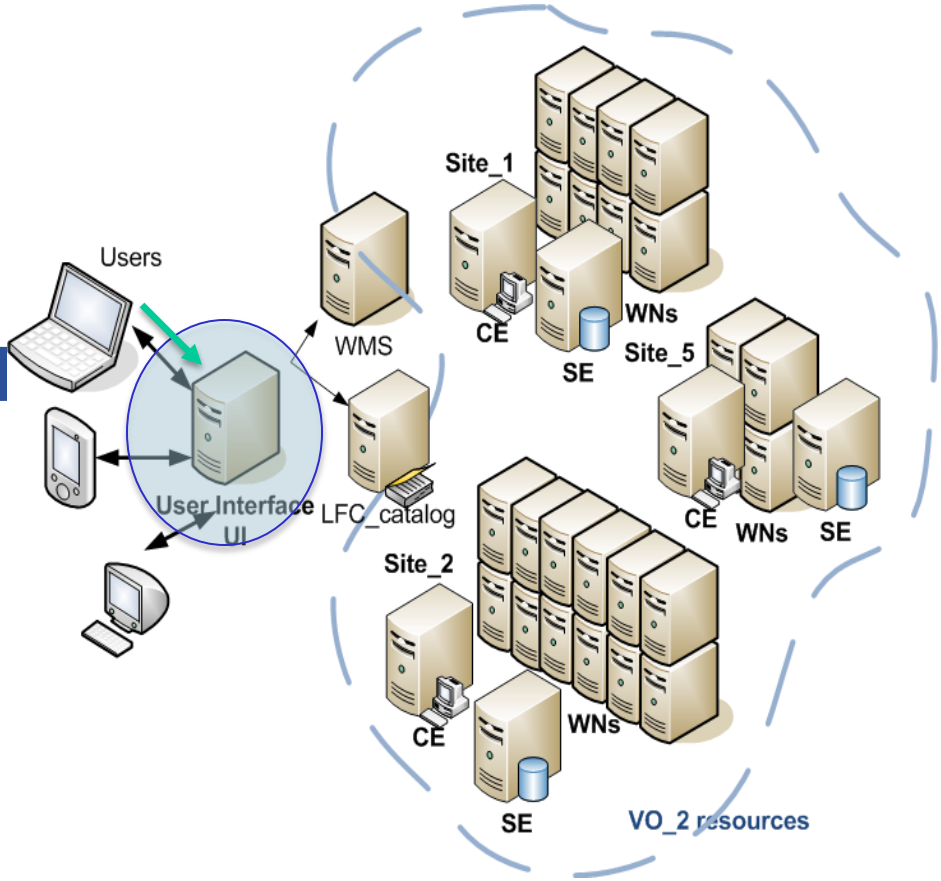
Security

- **Security aspects**
  - Authentication
  - Authorization
  - Delegation
- **Authentication**
  - Private Key: Strictly personal
  - Public Key: Known to everyone



User Interface (1)

- Allows users to access Grid functionalities
- A machine where users have a personal account and where the user certificate is installed
- Gateway to Grid Services



User Interface (2)

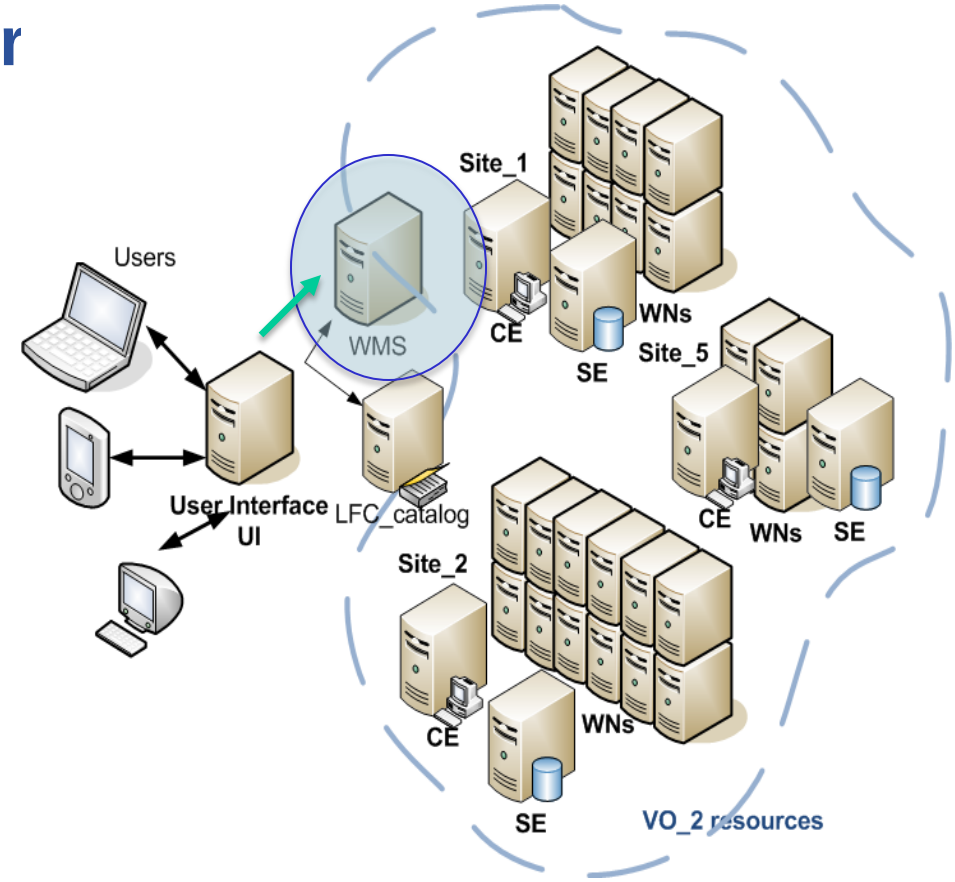
- *It provides a Command Line Interface to perform some basic Grid operations such as:*

- List all the resources suitable to execute a given job
- Submit jobs for execution
- Show the status of submitted jobs
- Cancel one or more jobs
- Retrieve the logging and bookkeeping information of jobs
- Retrieve the output of finished jobs
- Copy, replicate and delete files from Grid

Workload Management System

- The resource broker is responsible for the acceptance of submitted jobs and for sending those jobs to the appropriate Computing Element

- Retrieves information from Information Catalogues so as to find the proper available resources depending on the job requirements



Computing Element

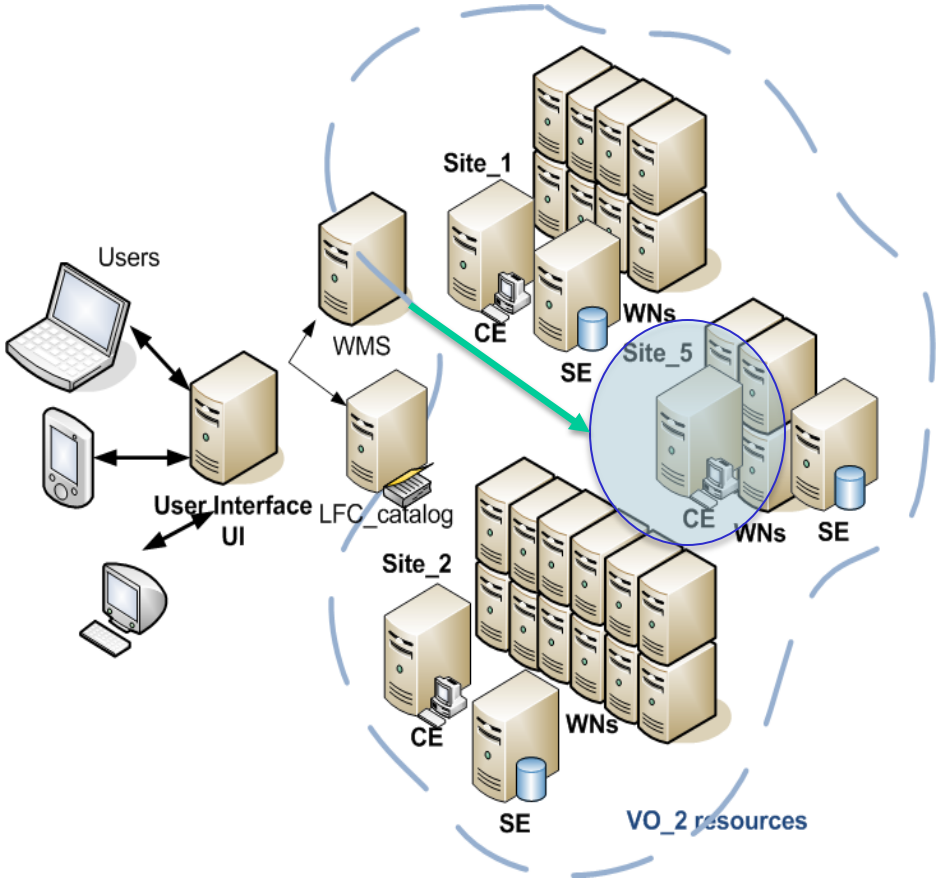
- **Grid interface”**

- **It is built on a farm of a computing nodes called Worker Nodes (WNs)**

- **Executes the basic queues functions**

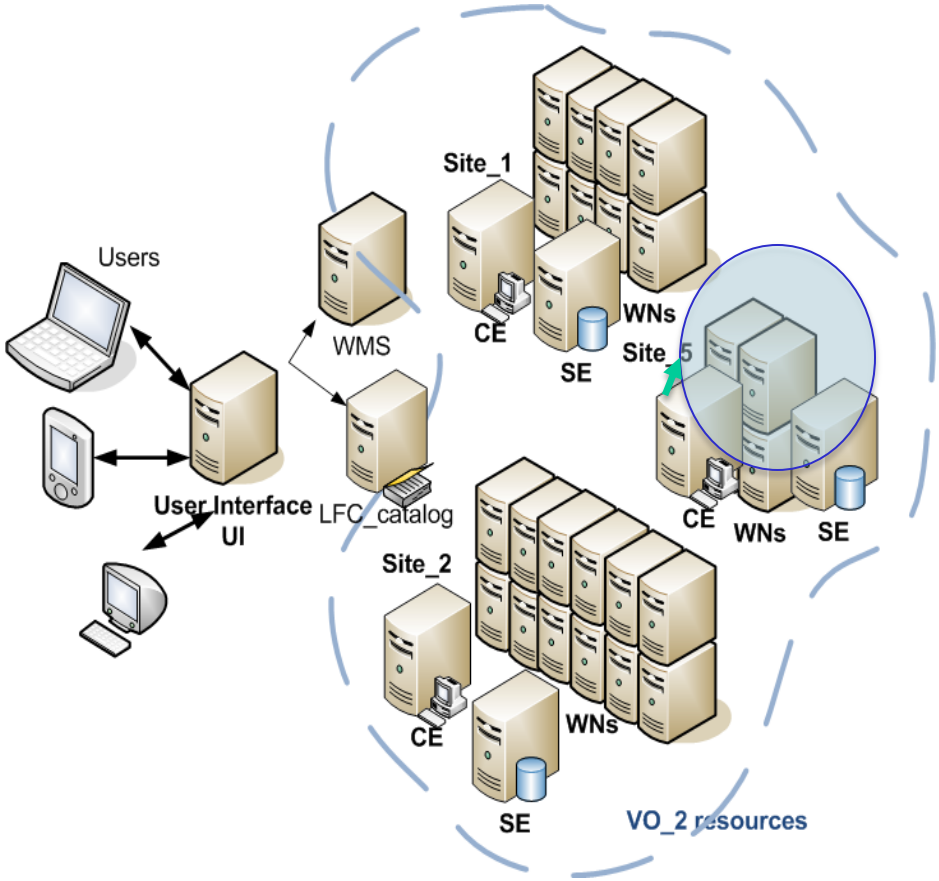
- **In the Computing Element, a process is being executed that accepts jobs and dispatch them for execution to the Worker nodes (WNs)**

- **The state of an executing job is being watched by the Computing Element**



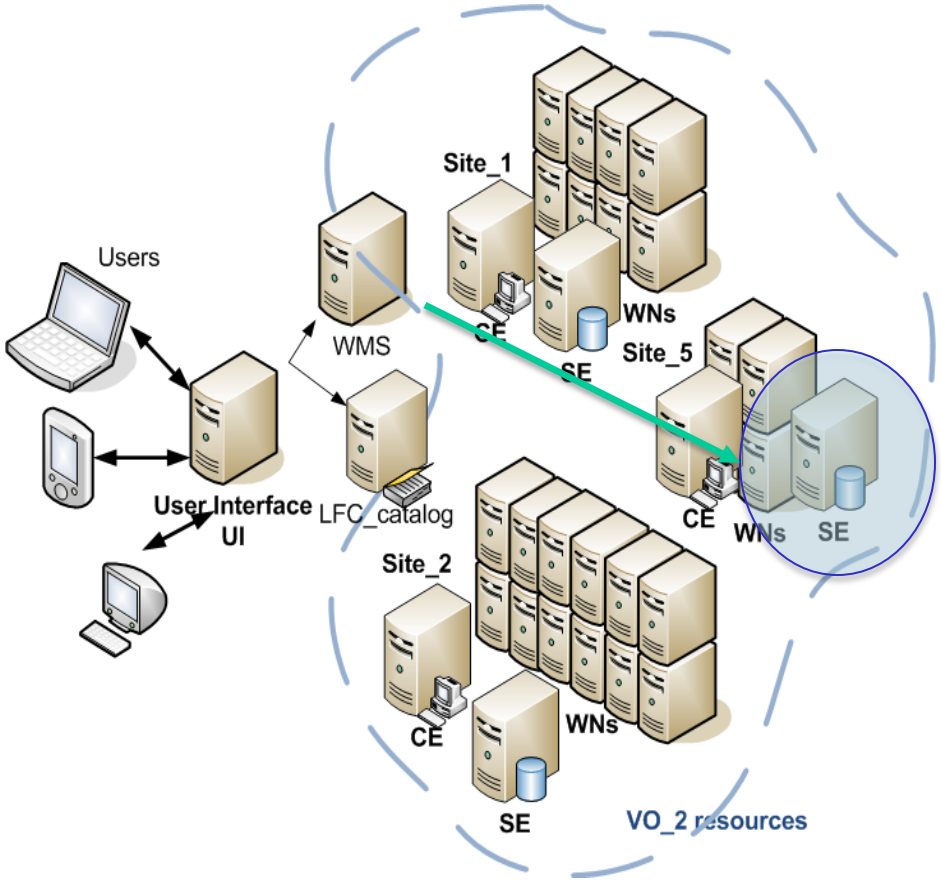
Worker Node

- **The submitted jobs are being executed in the Worker nodes**
- **Need only inbound connectivity**
- **Only basic services of middleware are required to be provided by the Worker nodes such as**
  - Application libraries
  - Application Programming Interfaces (API)
  - Commands for performing actions on Grid resources and Grid data



Storage Element

- It provides uniform access to storage resources
  (it may control simple disk servers, large disk arrays or Mass Storage Systems (MSS)
- Each site may provide one or more SEs



Getting started

- **Obtaining a certificate**
- **Registering with LCG / EGEE**
- **Choosing a VO**
- **Accounts for the training events:**
  - ssh ui01.isabella.grnet.gr (Putty)
    - login as: **egee05– egee50**

Installing the certificate

- ✓ [egee@ui01 egee]$ **mkdir .globus**
  - Create directory .globus under the user home directory
- ✓ [eg…em
  - …em
- ✓ [ege…em
  - …
- ✓ [eg…n
  - …
- ✓ [egee01@ui01 egee01]$ **chmod 400 ~/.globus/userkey.pem**
  - The key must be readable only by the user

- Prepare certificates for the training event only:

[egee01@ui01 egee01]$ **./preparecerts.sh**

[egee01@ui01 egee01]$ **ls -l ~/.globus**

```
total 12
-r--r--r--  1 egee01  training  5535 Sep 14 16:55 usercert.pem
-r--------  1 egee01  training   963 Sep 14 16:55 userkey.pem
```

[egee01@ui01 egee01]$ **grid-cert-info**

Authentication with digital certificates

- **Each entity (user, resource) must obtain a certificate**
- **The certificate includes information, such as the expiration date, the Certification Authority that signed it, the owner's public key and a DN**
- **The DN defines uniquely the owner and has the following fields:**

C = Owner's country
O = Owner's organization
OU = Owner's group
CN = Owner's name
Public Key
Signature

Certificate Info

- **Retrieving information about the user certificate**

✓ **[egee@ui01 egee]$ grid-cert-info**

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3193 (0xc79)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=GR, O=HellasGrid Demos, OU=Certification Authoritites, CN=Hell
      asGrid Demo CA 2006
        Validity
            Not Before: Feb 17 08:30:02 2010 GMT
            Not After : Mar 22 08:30:02 2010GMT
        Subject: C=GR, O=HellasGrid Demos, OU=People, L=Training, CN=User
      3193
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
[…]
```

Proxy certificates

- A new temporal certificate created taking into account the issued certificate by the corresponding CA
  ⇨ a new key pair is created to be used during the period that the proxy is valid

- The new private key is not secured by a password

- The use of a proxy is recommended because:

  ✓ the *proxy* has a short lifetime
  ✓ uses a different private key from the issued certificate

Proxy certificate commands

- **Creating a proxy**

✓ **[egee@ui01 egee]$ voms-proxy-init --voms=hgdemo**

Enter GRID pass phrase: keyforcert

- **Retrieving information about the user proxy**

- **[egee@ui01 egee]$ grid-proxy-info**

Proxy certificate

- **Creating a proxy**

✓ **[egee@ui01 egee]$ voms-proxy-init --voms=hgdemo**

Enter GRID pass phrase:

Your identity: /C=GR/O=HellasGrid Demos/OU=People/L=Lamia_Training/CN=User 3193

Creating temporary proxy ..... Done

Contacting voms.grid.auth.gr:15030 [/C=GR/O=HellasGrid/OU=auth.gr/CN=voms.grid.auth.gr] "hgdemo" Done

Creating proxy ..... Done

Your proxy is valid until Wed May 20 06:09:15 2009

- **Destroying a proxy**

✓ **[egee@ui01 egee]$ voms-proxy-destroy**

Proxy Info

- **Retrieving information about the user proxy**

✓ **[egee@ui01 egee]$ grid-proxy-info**

```
subject  : /C=GR/O=HellasGrid Demos/OU=People/L=Training/CN=User 3193/CN=proxy
issuer   : /C=GR/O=HellasGrid Demos/OU=People/L=Training/CN=User 3193
identity : /C=GR/O=HellasGrid Demos/OU=People/L=Training/CN=User 3193
type     : full legacy globus proxy
strength : 1024 bits
path     : /tmp/x509up_u1049
timeleft : 11:58:53
```

MyProxy

- **Remote service that stores user credentials**

- **Proxy certificate that lasts**
  - more than the maximum allowed hours (by default 12 hours)
  - less than the remaining lifetime of user's certificate
  - Job has to end before the WallClockTime limit, which duration depends on the CE

- **gLite – Using MyProxy service**
  - myproxy-init -d -n –s MY_PROXY_SERVER_LOCATION
    - -n: automatic renewal without passphrase
    - -d: associate the user's DN with the proxy
  - voms-proxy-init --voms VO-NAME

VOMS service (1)

- **Virtual Organization Membership Service (VOMS)**
  - Central database for VO membership information

- **Allows a proxy to have extensions containing information:**
  - About the VO
  - The groups the user belongs to in the VO
  - Any roles the user is entitled to have

- **Creation of group and roles**
  - VO administrators differentiate users' privileges and rights

VOMS service (2)

- ***Group*: subset of the VO containing members who share some responsibilities or privileges in the project**
  - Hierarchically organized
  - A user can be a member of any number of groups
  - VOMS proxy contains the list of all groups the user belongs to
  - Group ⇨ privileges the user **ALWAYS** has

- ***Role*: Attribute which typically allows a user to acquire special privileges to perform specific tasks**
  - Role ⇨ privileges the user needs to have only from time to time

Information service (1)

- **Globus Moinitoring and Discovery service**
  - resource discovery and publishing of he resource status
  - Use of OpenLDAP (open source implementation of the *Lightweight Directory Access Protocol (LDAP))*
- **Relational Grid Monitoring Architecture (R-GMA)**
  - Producers – consumers – registry
- **MDS hierarchical architecture:**
  - CEs and SEs publish information on *resource-level BDII (Berkeley Database Information Index)*
  - Each site publishes data from all the resource-level BDIIs through the site-level BDII
  - A top-level BDII is used to read from a group of sites, depicting a view of the overall Grid resources (on top of the hierarchy)

A Command-line tool for querying MDS

- **lcg-infosites ⇨ obtain VO-specific information on existing Grid resources**

  lcg-infosites --vo <vo> <option> -v <verbosity> -f <site> --is <bdii>

  where:

  --vo <vo>: the name of the VO to which the information to print is related (mandatory)

  <option>: specifies what information has to be printed. It can take the following values:

  ce: the number of CPUs, running jobs, waiting jobs and CE names (global, no VO-specific information)

  se: the names of the SEs supporting the VO, the type of storage system and the used and available space;

  -v 1: only the CE / SE names

  -v 2: the cluster names, the amount of RAM, the operating system name and version and the processor model

  all: the information given by ce and se

  closeSE: the names of the CEs supporting the VO and their close SEs

  tag: the software tags published by each CE supporting the VO

  lfc: the hostname of the LFC catalogues available to the VO

  lfcLocal: the hostname of the local LFC catalogues available to the VO

  rb: the hostname and port of the RBs available to the VO

  dli: the Data Location Index servers available to the VO

  dliLocal: the local Data Location Index servers available to the VO

  sitenames: the names of all WLCG/EGEE sites;

Examples

- **Obtaining information**

- ✓ [egee@ui01 egee]$ **lcg-infosites --vo hgdemo ce**
- ✓ [egee@ui01 egee]$ **lcg-infosites --vo hgdemo se**
- ✓ [egee@ui01 egee]$ **lcg-infosites --vo hgdemo lfc**
- ✓ [egee@ui01 egee]$ **lcg-infosites --vo see tag**
- ✓ [egee@ui01 egee]$ **lcg-infosites --vo hgdemo sitenames**

Examples (1)

- **Obtaining information about computing resources**
  - ✓ **[egee@ui01 egee]$ lcg-infosites --vo hgdemo ce**

#CPU	Free	Total Jobs	Running	Waiting	ComputingElement
180	84	0	0	0	ce01.marie.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
12	11	0	0	0	ce01.grid.auth.gr:2119/jobmanager-pbs-hgdemo
63	1	1	0	1	ce01.isabella.grnet.gr:2119/jobmanager-pbs-hgdemo
224	11	0	0	0	ce01.athena.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
120	120	0	0	0	ce01.grid.uoi.gr:2119/jobmanager-pbs-hgdemo
93	72	0	0	0	node001.grid.auth.gr:2119/jobmanager-pbs-hgdemo
220	124	0	0	0	ce01.ariagni.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
180	83	0	0	0	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-hgdemo
114	13	0	0	0	ce01.kallisto.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
360	86	0	0	0	ce02.athena.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
30	22	0	0	0	ce02.marie.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
118	2	0	0	0	ce01.afroditi.hellasgrid.gr:2119/jobmanager-pbs-hgdemo

Examples (2)

- **Obtaining information about storage resources**

✓ **[egee@ui01 egee]$ lcg-infosites --vo hgdemo se**

Avail Space(Kb)	Used Space(Kb)	Type	SEs
1293436933	1911206947	n.a	se01.afroditi.hellasgrid.gr
2320000000	3939	n.a	se01.kallisto.hellasgrid.gr
942180000	523256	n.a	se01.isabella.grnet.gr
1024484	8588910108	n.a	se01.athena.hellasgrid.gr
25815377	1932174858	n.a	se01.grid.auth.gr
1930000000	34	n.a	se01.ariagni.hellasgrid.gr
892120000	n.a	n.a	se01.grid.uoi.gr
334180000	n.a	n.a	se02.marie.hellasgrid.gr
1470000000	n.a	n.a	se01.marie.hellasgrid.gr

Examples (3)

- **Listing the hostname of the LFC catalogues**
  - ✓ **[egee@ui01 egee]$ lcg-infosites --vo hgdemo lfc**
    lfc.isabella.grnet.gr

- **Listing the software tags published by each CE supporting the VO**
  - ✓ **[egee@ui01 egee]$ lcg-infosites --vo see tag**

[…] VO-ops.vo.egee-see.org-SAM
VO-see-Espresso
VO-see-HYDRA-CLIENT
VO-see-octave-2.1.73
VO-see-gsl-1.9
VO-see-meep
VO-see-RNAHybrid-2.1
VO-see-sunjdk1.6.0_04
VO-see-FLUKA-2008.3.7 […]

- **Listing all WLCG/EGEE sitenames**
  - ✓ **[egee@ui01 egee]$ lcg-infosites --vo hgdemo sitenames**

lcg-info

- Listing the supported attributes
  - ✓ [egee@ui01 egee]$ **lcg-info --list-attrs**

```
            Attribute name      Glue object class     Glue attribute name
WorstRespTime               GlueCE              GlueCEStateWorstResponseTime
CEAppDir                    GlueCE              GlueCEInfoApplicationDir
TotalCPUs                   GlueCE              GlueCEInfoTotalCPUs
MaxRunningJobs              GlueCE              GlueCEPolicyMaxRunningJobs
CE                                    GlueCE              GlueCEUniqueID
WaitingJobs                 GlueCE              GlueCEStateWaitingJobs
```

- Querying the sites supporting the specific VO and have a specific OS about the processor, the running jobs and the free CPUs
  - ✓ [egee@ui01 egee]$ **lcg-info --vo hgdemo --list-ce --query 'OS=*Scientific*' --attrs 'Processor,RunningJobs,FreeCPUs'**

```
[ …] - CE: ce01.afroditi.hellasgrid.gr:2119/jobmanager-pbs-see
 - Processor        Xeon
 - RunningJobs       4
 - FreeCPUs         0

- CE: ce01.ariagni.hellasgrid.gr:2119/jobmanager-pbs-see
 - Processor        xeon
 - RunningJobs       11
 - FreeCPUs         115

- CE: ce01.athena.hellasgrid.gr:2119/jobmanager-pbs-see
 - Processor        Xeon
 - RunningJobs       2
 - FreeCPUs         180

[…]
```

Job Description Language

- A high-level language based on the ***Classified Advertisement (ClassAd) language***
- JDL describes jobs and aggregates of jobs with arbitrary dependency relations
- JDL specifies the desired job characteristics and constraints, which are taken into account by the WMS to select the best resource to execute the job
- A JDL file consists of lines having the format:

  attribute = expression;

  - Expressions can span several lines, but only the last one must be terminated by a semicolon
  - Literals are enclosed in double quotes
  - " in strings must be escaped with a backslash ("\"Hallo")
  - The character " ' " cannot be used in the JDL
  - Comments of each line begin with # or //
  - Multi-line comments must be enclosed between "/*" and "*/"
  - **No blank characters or tabs should follow the semicolon at the end of a line**

Attributes (1)

Executable	✓ The value of this attribute is the executable filename or the command to be run by the job ✓ If the command is already present on the WN, it must be expressed as a absolute path
StdOutput	✓ The name of the files containing the standard output
StdError	✓ The name of the files containing the standard error
StdInput	✓ The names of the files used as Input files
InputSandbox	✓ The files to be transferred from the UI needed by the job
OutputSandbox	✓ The files to be transferred back to the UI after the job is finished
Virtual Organisation	✓ Explicitly specify the VO of the user
Requirements	✓ Expresses constraints on the resources where the job should run ✓ Its value is a Boolean expression that must evaluate to true for a job to run on that specific CE (example: Requirements = other.GlueCEInfoLRMSType == "PBS" && other.GlueCEInfoTotalCPUs > 1;)

Attributes (2)

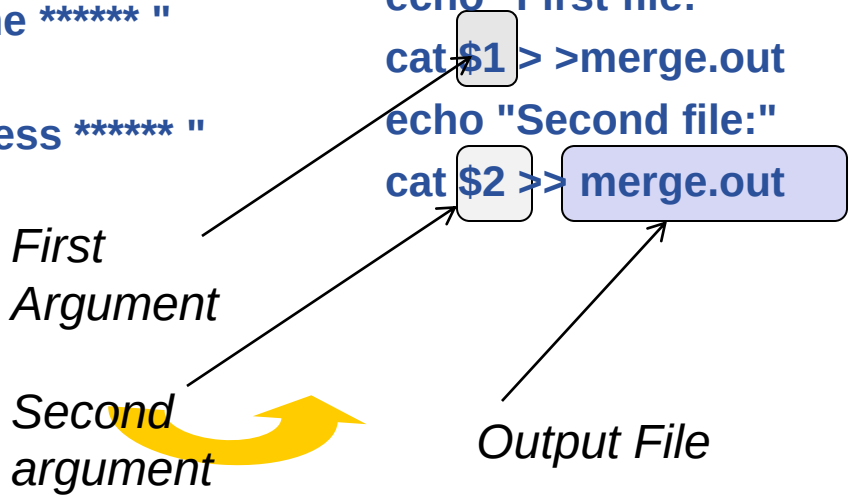
RetryCount MaxRetryCount	✓ Times that the WMS automatically resubmitts jobs which failed for some reason (deep resubmission ⇨ when the job failed after started running in a WN)
ShallowRetryCount MaxShalowRetryCount	✓ Times that the WMS automatically resubmitts jobs which failed for some reason (shallow resubmission – gLite)
MyProxyServer	✓ The Proxy server to be used for certificate renewal
Rank	✓ The CE with the highest rank is selected by the WMS to execute a job ✓ by default *Rank* = *other.GlueCEStateEstimatedResponseTime* (but other.GlueCEStateFreeCPUs other.GlueCEStateWaitingJobs)

A simple job

✓ [egee@ui01 egee]$ **less testJob1.sh**

```
#!/bin/bash
echo "****** Running...  date ****** "
date
echo "****** Running... hostname ******"
hostname
echo "****** Running... pwd ****** "
pwd
echo "****** Running... ls ****** "
ls -l
echo "****** Running... uptime ****** "
uptime
echo "****** Learn your process ****** "
ps aux | grep home
```

```
echo "****** Running... ls ****** "
ls -l
echo "****** Printing Input files ****** "
echo "First file:"
cat $1 > >merge.out
echo "Second file:"
cat $2 >> merge.out
```



A simple JDL file

- [egee@ui01 egee]$ **less testJob1.jdl**

```
Executable = "testJob1.sh";
Arguments = "fileA fileB";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {“./testJob.sh", “./fileA", “./fileB"};
OutputSandbox = {"std.out", "std.err",  merge.out"};
```

Job submission overall …

- [egee@ui01 egee]$ **cd ~/training/simpleJob**
- [egee@ui01 egee]$ **glite-wms-job-list-match -a testJob1.jdl**
- [egee@ui01 egee]$ **glite-wms-job-submit -o jobId -a testJob1.jdl**
- [egee@ui01 egee]$ **glite-wms-job-status -i jobId**

Job matching to CEs

- **Listing computing elements that match a job description**
  - ✓ **[egee@ui01 egee]$ glite-wms-job-list-match -a testJob1.jdl**

```
Connecting to the service https://wms02.egee-see.org:7443/glite_wms_wmproxy_server

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

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

      *CEId*
- ce01.afroditi.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
- ce01.ariagni.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
- ce01.athena.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
- ce01.grid.auth.gr:2119/jobmanager-pbs-hgdemo
- ce01.kallisto.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
- ce01.marie.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
- ce02.athena.hellasgrid.gr:2119/jobmanager-pbs-hgdemo
- node001.grid.auth.gr:2119/jobmanager-pbs-hgdemo
- ce01.isabella.grnet.gr:2119/jobmanager-pbs-hgdemo

==========================================================================
```

Job submission

- **Single Job submission**
  - ✓ **[egee@ui01 egee]$ glite-wms-job-submit -o jobId -a testJob1.jdl**

Connecting to the service https://wms01.egee-see.org:7443/glite_wms_wmproxy_server

========== glite-wms-job-submit Success ==========

The job has been successfully submitted to the WMProxy
Your job identifier is:

https://lb01.egee-see.org:9000/Un97vYtCozCRTARWWJB5RA

Job Id

The job identifier has been saved in the following file:
/home/training/egee02/jobId

File where the Job Id is stored

==========

  - ✓ **glite-wms-job-submit -o jobId -r <CE Id> o -a testJob1.jdl**
    - -r : sends the job directly to the specified CE

Job status' retrieval

- ## Retrieving the status of a job

  ✓ [egee@ui01 egee]$ **glite-wms-job-status -i jobId**

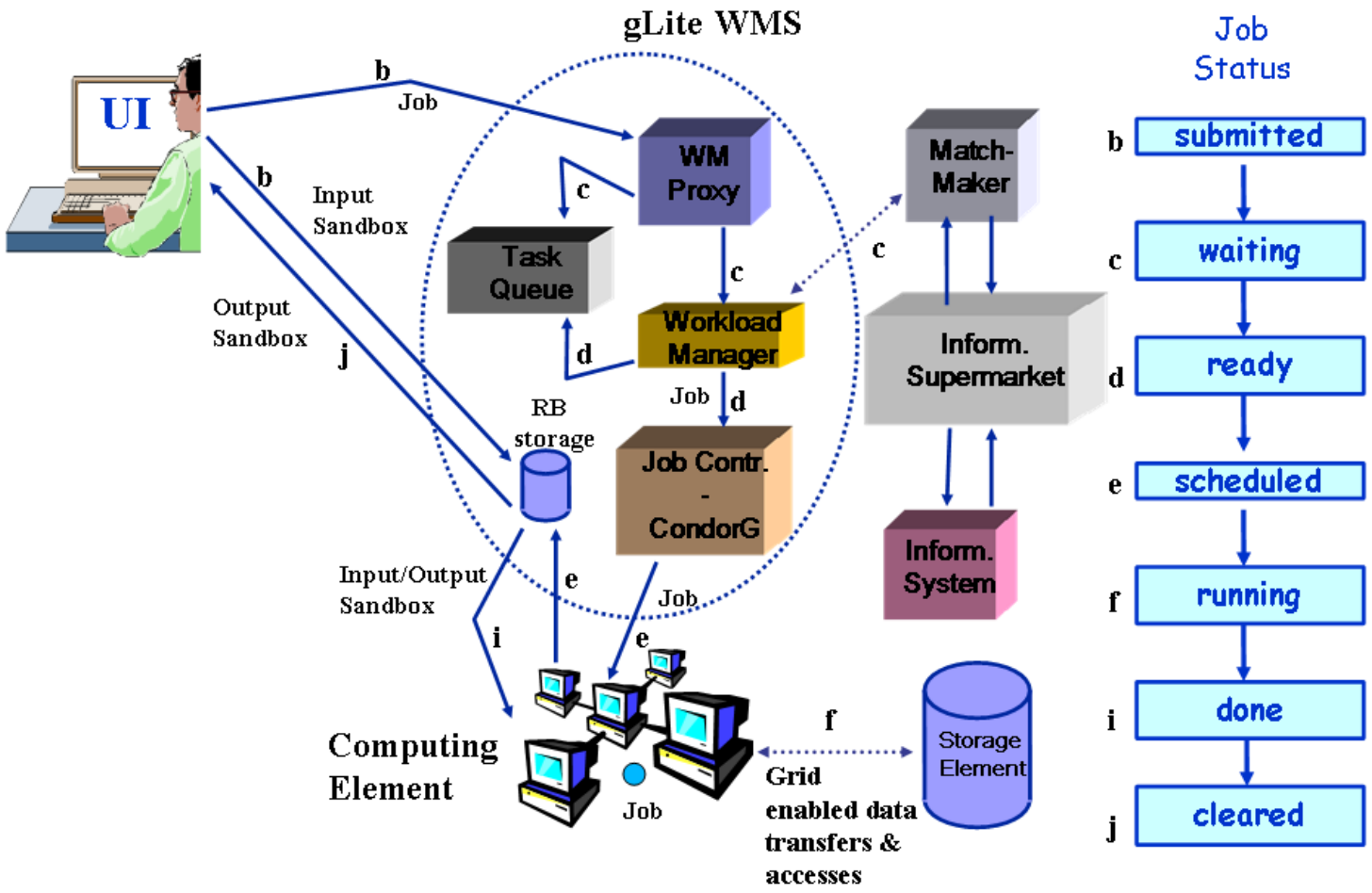
```
**********************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://wms.grid.hgdemo.gr:9000/-HsYciupi_keZWh2GNA7YQ
Current Status:     Running
Status Reason:      unavailable
Destination:        ce01.grid.hgdemo.gr:2119/jobmanager-pbs-hgdemo
Submitted:          Mon May  4 01:40:59 2009 EEST
**********************************************************
```

  ✓ [egee@ui01 egee]$ **watch "glite-job-status -i jobId"**

  (To exit ctrl + C)

Job flow (1)


gLite WMS
UI
b
Job
b
Input
Sandbox
Output
Sandbox
j
WM
Proxy
c
Task
Queue
c
Workload
Manager
d
Job
d
RB
storage
Job Contr.
-
CondorG
Input/Output
Sandbox
e
Job
i
e
Computing
Element
Job
f
Grid
enabled data
transfers &
accesses
Storage
Element
Match-
Maker
c
Inform.
Supermarket
Inform.
System
Job
Status
b
submitted
c
waiting
d
ready
e
scheduled
f
running
i
done
j
cleared

Job flow (2)

- **Job submission**
  - Creation of proxy
  - Submission of the job to WMS
  - Files needed in the WN (**Input Sandbox**) are copied from the UI to the WMS
  - **Job status ⇨ SUBMITTED**

- **Finding the proper CE by the WMS**
  - the **Information Supermarket (ISM)** (an internal cache of information read from the BDII) is queried for the status of computational and storage resources
  - the File Catalogue is queried for the location of any required input files
  - **Job status ⇨ WAITING**

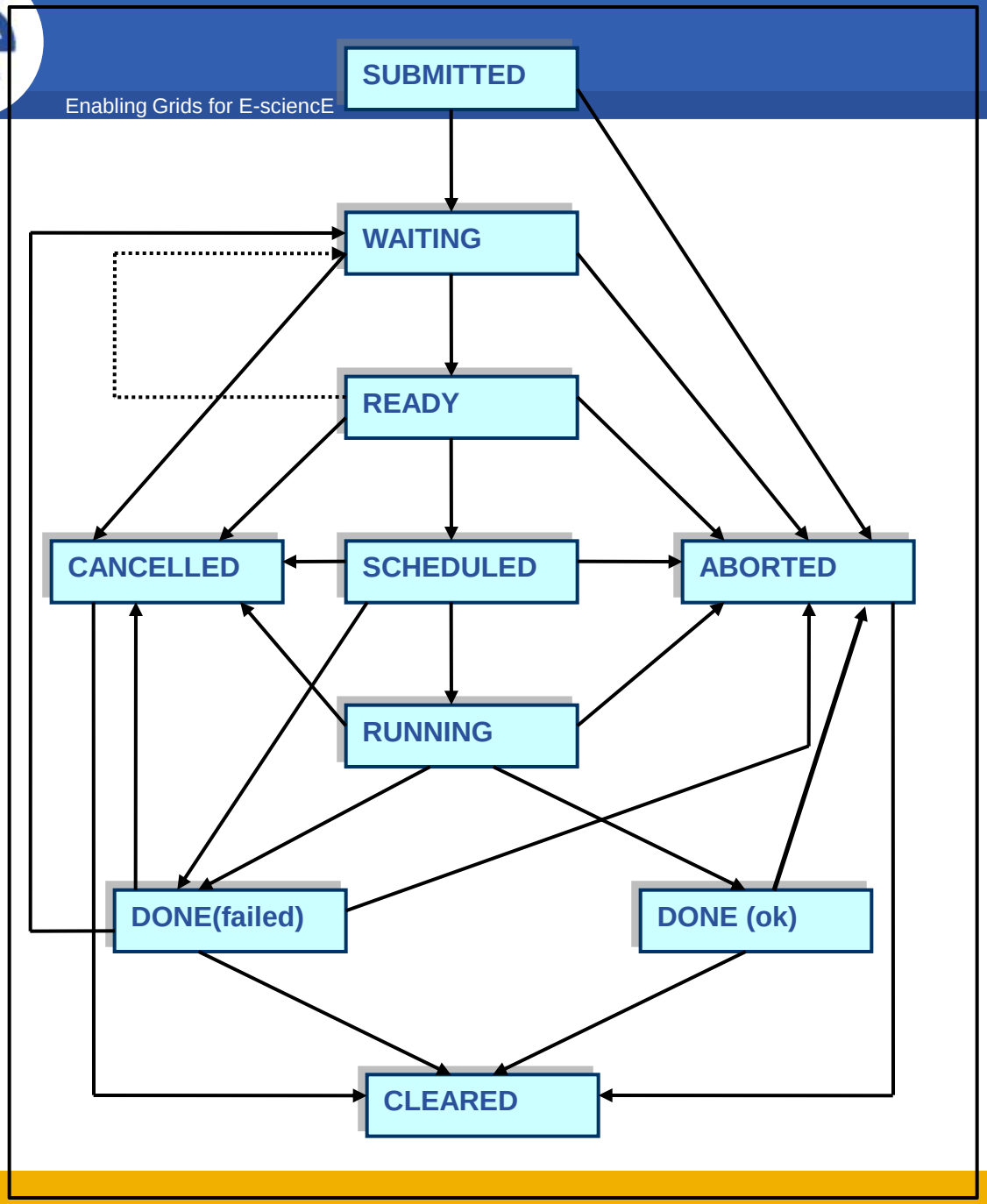
- **Job submission from the WMS to the selected CE**
  - A wrapper script along with other parameters is passed from the WMS to the CE.
  - **Job status ⇨ READY**

Job flow (2)

- **Job arrival to the CE**
  - Job is sent for execution to the local LRMS
  - Event is logged in the LB
  - **Job status ⇨SCHEDULED**
- **Job submission to the WN**
  - LRMS sends the job for execution to the WN
  - Input Sandbox files are copied from the WMS to the WN
  - Grid files can be directly accessed from a SE with Data Management tools during execution
  - Output files can be uploaded to the Grid (copy it to the SE and register it to the file catalogue)
  - **Job status ⇨ RUNNING**

Job flow (3)

- **Job finished without errors**
  - **Output Sandbox** (small output files specified by the user) are transferred back to the WMS
  - Event logged in the LB
  - **Job status ⇨ DONE**
- **Output retrieval**
  - Retrieval of output files in the UI
  - **Job status ⇨ Cleared**


SUBMITTED
WAITING
READY
CANCELLED
SCHEDULED
ABORTED
RUNNING
DONE(failed)
DONE (ok)
CLEARED

Job cancelling

- ## Cancelling a job

  ✓ [egee@ui01 egee]$ **glite-wms-job-cancel -i jobId**

  Are you sure you want to remove specified job(s) [y/n]y : y

  Connecting to the service https://wms.grid.hgdemo.gr:7443/glite_wms_wmproxy_server

  ========== glite-wms-job-cancel Success ==========

  The cancellation request has been successfully submitted for the following job(s):

  - https://wms.grid.hgdemo.gr:9000/p9iiejqpI9dXy4zkHibbbQ

  ==========

Output retrieval

- **If the job's status is DONE, then its output can be copied to the UI with the commands:**
  - ✓ **[egee@ui01 egee]$ glite-wms-job-output -i jobId**

```
Connecting to the service https://wms.grid.hgdemo.gr:7443/glite_wms_wmproxy_server

================================================================
                JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
https://wms.grid.hgdemo.gr:9000/j6SI9Y6yj1U9J9GfnEzqxw
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/egee01_j6SI9Y6yj1U9J9GfnEzqxw
================================================================
```

Basic commands

- Creating a proxy certificate
  - voms-proxy-init --voms=hgdemo
- Listing Computing Elements that match a job description
  - glite-wms-job-list-match -a testJob1.jdl
- Submitting a job
  - glite-wms-job-submit -o jobId -a testJob1.jdl
- Retrieving the status of a job
  - glite-job-status -i jobId
- Retrieving the output of a job
  - glite-wms-job-output -i jobId

Job “Hallo.c”

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#define BUF_SIZE 1000
int main( int argc, char *argv[]) {
   char *hostname;
if(argc<3){
      fprintf(stdout,"ERROR:START AND END OF THE FOR
       LOOP SHOULD BE PROVIDED!\
n");
      exit -1;
   }
   else if(argc>3){
      fprintf(stdout,"ERROR: TOO MUCH ARGUMENTS...\n");
      exit -1;
   }
   // Start and End value for the executed for loop
   int start = atoi(argv[1]);
   int stop = atoi(argv[2]);
   int i =0;
  fprintf(stdout,"Hallo World from C!!!!\n");
  fprintf(stdout,"This program prints \"Hallo World\" for %d - %d times in the
output file test.out\n", stop, start);
  fflush(stdout);
  FILE *out_file;
  out_file = fopen("test.out","w");

  if (out_file == NULL) {
     fprintf(stderr,"Can not open output file\n");
     exit (-1);
  }
hostname = (char *) malloc(BUF_SIZE);
  gethostname(hostname,255);
  printf("The hostname of the node that executes the program is
      %s\n",hostname);
  for (i=start;i<=stop; i++){
     fprintf(out_file,"Hallo World from WN %s for time %d
      !\n",hostname,i);
     if(i%100==0){
        fprintf(stdout,"Completed printing for time %d\n",i);
        fflush(stdout);
     }
  }
  free(hostname);
  fclose(out_file);
  return 0;
}
```

First Argument: Start of counting

Second Argument: End of counting

Tips for Exercise 1

- **Modify a copy of the testJob1.jdl**
  - cd ~/training/CHalloWorldSimple/
  - vi Hallo.jdl
- **Compile:**
  - gcc -o Hallo Hallo.c
- **Identify the correct executable for the new job**
  - Which is the correct executable???
  - Are there any arguments????
  - Are there any output files????

Exercise 2…

- **BuildandRun.sh**

```
#!/bin/sh
if [ X$1 == X ]; then
        echo "Error: The file to compile is not given"
        exit 1
fi
if [ X$4 == X ]; then
        echo "Only three Inputs"
        gcc $1.c -o $1
        ./$1 $2 $3
        exit 0
fi
```

Required arguments

Tips for Exercise 2

- **Use a bash script to run the compiler and then run the executable (BuildandRun.sh)**

- **Create the correct JDL file**

```
cd  ~/training/CHalloWorld/
vi Hallo.jdl
```

- **Which is the initial executable????**
- **Which is the correct argument????**
- **Which files are stored locally and should be transferred to the remote machine??**
- **Is there any EXTRA output file????**

Basic commands

- **Creating a proxy certificate**
  - **voms-proxy-init --voms=hgdemo**
- **Listing Computing Elements that match a job description**
  - **glite-wms-job-list-match -a Hallo.jdl**
- **Submitting a job**
  - **glite-wms-job-submit -o jobId -a Hallo.jdl**
- **Retrieving the status of a job**
  - **glite-job-status -i jobId**
- **Retrieving the output of a job to a specific directory**
  - **glite-wms-job-output -i jobId --dir** ~/job

And some more commands…

- **Retrieving the status of a job**
  - glite-wms-job-status --all
  - glite-wms-job-status --all -s CLEARED
- **Retrieving job output to a specific directory**
  - glite-wms-job-output -i jobId **--dir** ~/job1
- **Retrieving logging information about submitted jobs**
  - glite-wms-job-logging-info -i jobId

Job Collections

- **Job Collection: Submit a set of independent jobs**

Preparation

- **Create a jdl directory**
  - cd ~/training/collection/
  - ls jdl/

Job Collections

- **Submit job collection**
  - **glite-wms-job-submit -o collec -a --collection jdl**

Connecting to the service https://wms.grid.hgdemo.gr:7443/glite_wms_wmproxy_server

========== glite-wms-job-submit Success ==========
The job has been successfully submitted to the WMProxy
Your job identifier is:

https://wms.grid.hgdemo.gr:9000/v98yKZixxr97I1GNTN23XQ

The job identifier has been saved in the following file:
/storage/hgdemolocal/egee01/collec
==========

- **Retrieve status**

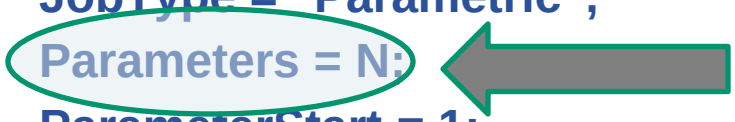
  glite-wms-job-status -i collec

Parametric job

- One or parametric attributes in the JDL
- Submission of a *Parametric job results in the submission of a set of jobs having the* same descriptions apart from the values of the parametric attributes
- Both the parametric job and all jobs resulting from the submission of it are assigned by the WMS with an identifier so that it is possible to monitor and control each of them separately and as a single entities

JDL file for parametric job

```
[
    Type = "job";
    JobType = "Parametric";
    Parameters = N;
    ParameterStart = 1;
    ParameterStep =10;
    RetryCount = 0;
    ShallowRetryCount = 3;
    Executable = "BuildandRun.sh";
    InputSandbox = {"BuildandRun.sh","Hallo.c"};
    Arguments =  "Hallo 1 _PARAM_";
    StdOutput = "std.out";
    StdError = "std.err";
    OutputSandbox = {"std.out", "std.err"};
]
```



Job execution

- **Submit job**

  glite-wms-job-submit -o paramId -a parametric.jdl

- **Watch the job status**

  watch "glite-wms-job-status -i paramId "

- **Retrieve the job output**

  glite-wms-job-output -i paramId

Using VO-specific packages

- **Example: Using sunjdk1.6.0_04**
- **JDL file:**

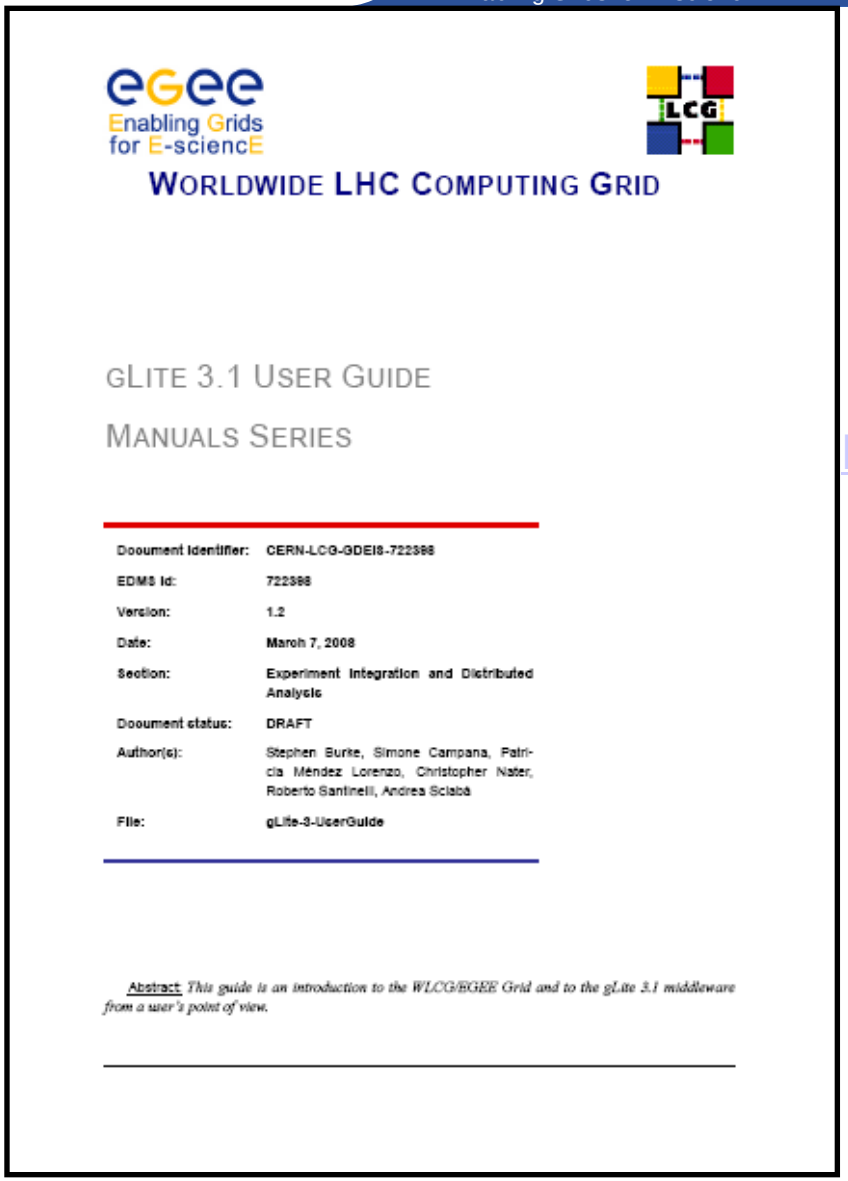
```
Executable = "testJob1.sh";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"testJob1.sh", "Hallo.jar"};
OutputSandbox = {"std.out", "std.err"};
Requirements = Member("VO-see-sunjdk1.6.0_04",
    other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

- **Script .sh**
  - export JAVA_HOME=$VO_SEE_SW_DIR/jdk1.6.0_04
  - export PATH=$JAVA_HOME/bin:$PATH
  - $JAVA_HOME/bin/java -classpath ./:Hallo.jar test.HalloWorld

Q&A

Thank you !

Documentation



Enabling Grids for E-sciencE

WORLDWIDE LHC COMPUTING GRID

gLITE 3.1 USER GUIDE

MANUALS SERIES

Document Identifier:	CERN-LCG-GDEIS-722398
EDMS Id:	722398
Version:	1.2
Date:	March 7, 2008
Section:	Experiment Integration and Distributed Analysis
Document status:	DRAFT
Author(s):	Stephen Burke, Simone Campana, Patricia Méndez Lorenzo, Christopher Nater, Roberto Santinelli, Andrea Sciabà
File:	gLite-3-UserGuide

Abstract: *This guide is an introduction to the WLCG/EGEE Grid and to the gLite 3.1 middleware from a user's point of view.*

http://glite.web.cern.ch/glite/documentation/