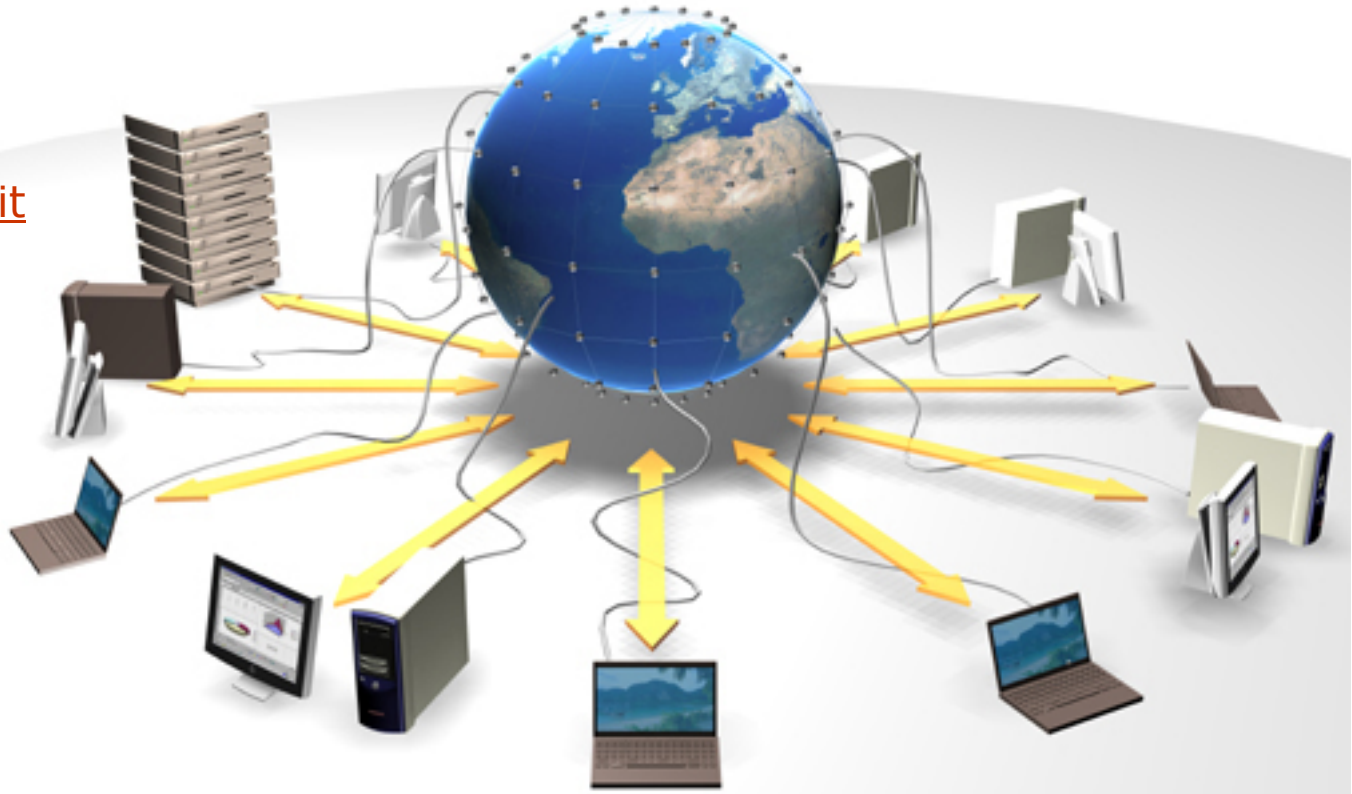
# The EDG Workload Management System: release 2

**Massimo Sgaravatto**
**INFN Padova - DataGrid WP1**

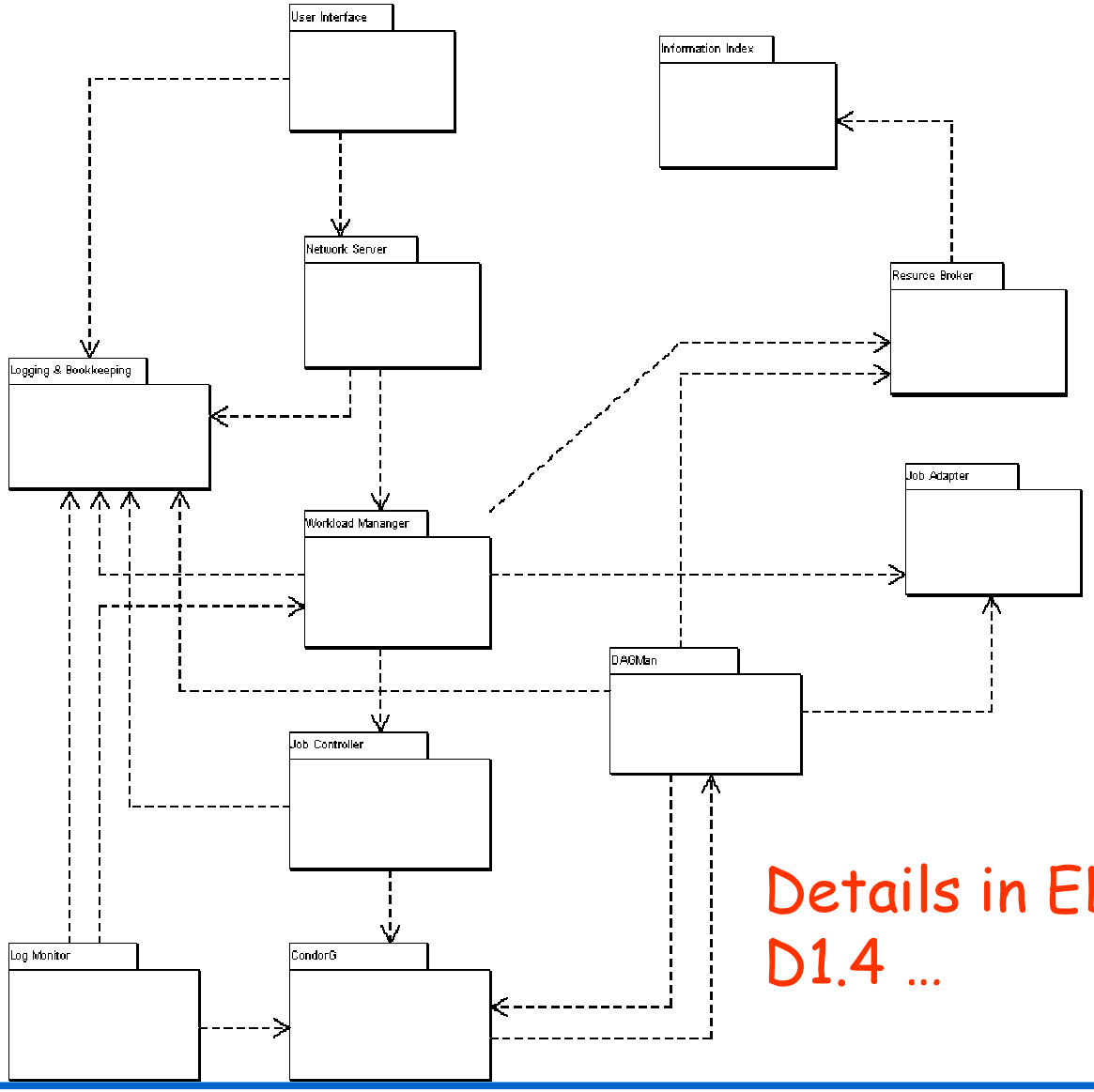
massimo.sgaravatto@pd.infn.it



# Review of WP1 WMS architecture

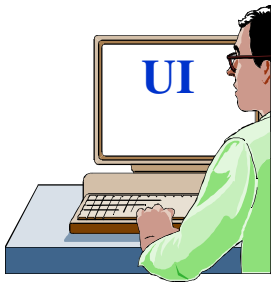
- WP1 WMS architecture reviewed
  - To apply the "lessons" learned and addressing the shortcomings emerged with the first release of the software, in particular
    - To increase the reliability problems
    - To address the scalability problems
  - To support new functionalities
  - To favor interoperability with other Grid frameworks, by allowing exploiting WP1 modules (e.g. RB) also "outside" the EDG WMS

# WP1 WMS reviewed architecture




Details in EDG deliverable D1.4 …

# Job submission example




RB node

Network Server

Workload Manager

Job Contr. - CondorG

RLS

Inform. Service

CE characts & status

SE characts & status

Computing Element

Storage Element

# Job subm

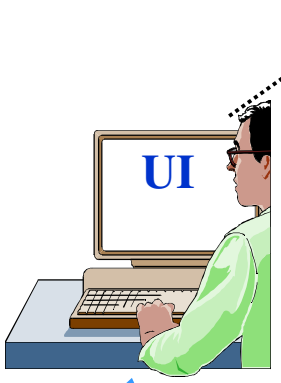
edg-job-submit myjob.jdl

Myjob.jdl

```
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. GlueHostOperatingSystemRelease == "Red Hat 6.2" &&
other.GlueCEPolicyMaxWallClockTime > 10000;
Rank  = other.GlueCEStateFreeCPUs;
```

Job Status

submitted




UI: allows users to access the functionalities of the WMS

Workload Manager

Job Contr. - CondorG

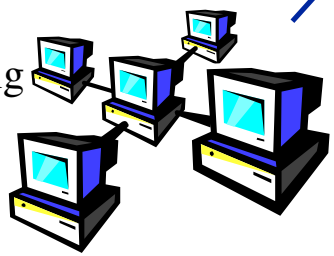
Inform. Service

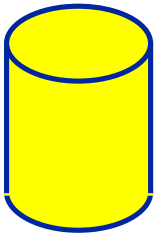
Job Description Language (JDL) to specify job characteristics and requirements

CE characts & status

SE characts & status

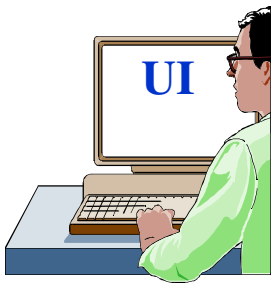
Computing Element





Storage Element

# Job submission




Job

Input
Sandbox
files

RB
storage

RB node

Network
Server

Workload
Manager

Job Contr.
-
CondorG

NS: network daemon
responsible for accepting
incoming requests

RLS

Inform.
Service

CE characts
& status

SE characts
& status

Computing
Element

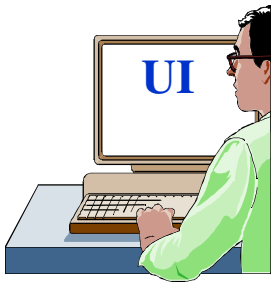
Storage
Element

Job Status

submitted

waiting

# Job submission




RB node

Network Server

Job

Workload Manager

RB storage

Job Contr. - CondorG

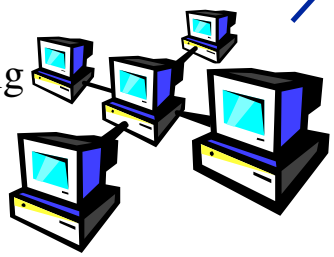
WM: responsible to take the appropriate actions to satisfy the request

RLS

Inform. Service

CE characts & status

SE characts & status



Computing Element

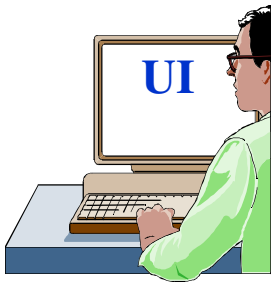
Storage Element

Job Status

submitted

waiting

# Job submission




RB node

Network Server

Match-Maker/ Broker

RLS

Job Status

submitted

waiting

RB storage

Workload Manager

Where must this job be executed ?

Inform. Service

Job Contr. - CondorG

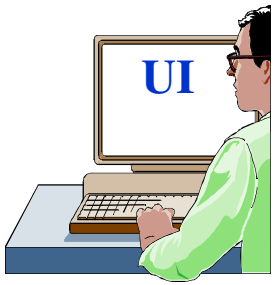
CE characts & status

SE characts & status

Computing Element

Storage Element

# Job submission




RB node

Matchmaker: responsible to find the "best" CE where to submit a job

Match-Maker/ Broker

RB storage

Workload Manager

Job Contr. - CondorG

RLS

Inform. Service

CE characts & status

SE characts & status

Computing Element

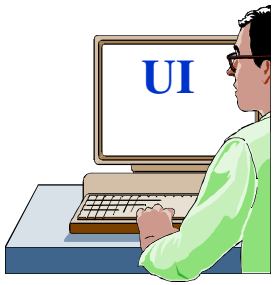
Storage Element

Job Status

submitted

waiting

# Job submission




RB node

Network Server

Workload Manager

Job Contr. - CondorG

RB storage

Match-Maker/ Broker

Where are (which SEs) the needed data ?

RLS

Inform. Service

What is the status of the Grid ?

CE characts & status

SE characts & status

Computing Element

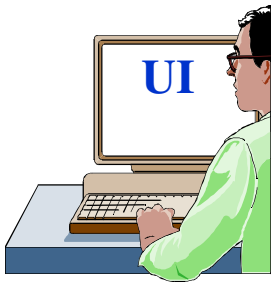
Storage Element

Job Status

submitted

waiting

# Job submission




RB node

Network Server

Match-Maker/ Broker

Workload Manager

CE choice

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

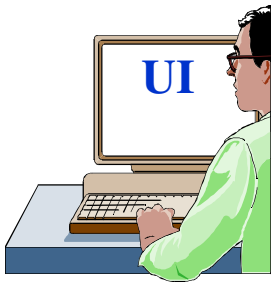
CE characts & status

SE characts & status

Computing Element

Storage Element

# Job submission




RB node

Network Server

Workload Manager

Job Contr. - CondorG

Job Adapter

RB storage

RLS

Inform. Service

Job Status

submitted

waiting

JA: responsible for the final "touches" to the job before performing submission (e.g. creation of wrapper script, etc.)

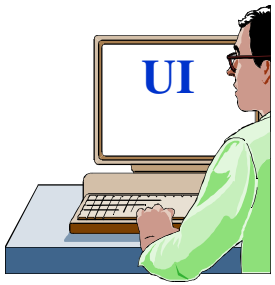
haracts
us

SE characts & status

Computing Element

Storage Element

# Job submission




RB node

Network Server

RB storage

Workload Manager

Job

Job Contr. - CondorG

JC: responsible for the actual job management operations (done via CondorG)

RLS

Inform. Service

CE characts & status

SE characts & status

Computing Element

Storage Element

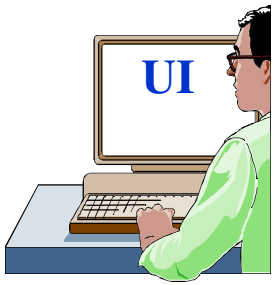
Job Status

submitted

waiting

ready

# Job submission




RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

Input Sandbox files

Job

Computing Element

RLS

Inform. Service

CE characts & status

SE characts & status

Storage Element

Job Status

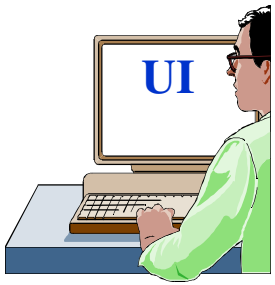
submitted

waiting

ready

scheduled

# Job submission




RB node

Network Server

Workload Manager

Job Contr. - CondorG

RB storage

Input Sandbox

RLS

Inform. Service

Job Status

submitted

waiting

ready

scheduled

running

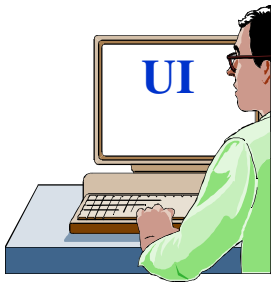
Computing Element

Job

"Grid enabled" data transfers/ accesses

Storage Element

# Job submission




RB node

Network Server

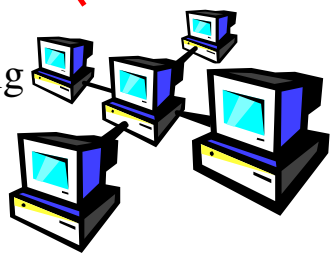
Workload Manager

Job Contr. - CondorG

RB storage

Output Sandbox files

Computing Element



RLS

Inform. Service

Storage Element

Job Status

submitted

waiting

ready

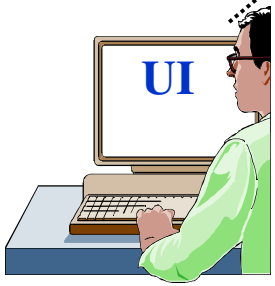
scheduled

running

done

# Job submission

edg-job-get-output <dg-job-id>




RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

Output Sandbox

RLS

Inform. Service

Job Status

submitted

waiting

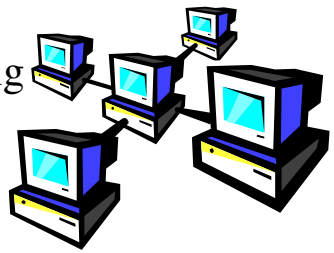
ready

scheduled

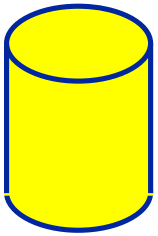
running

done

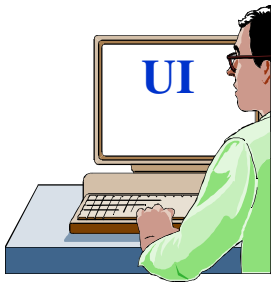
Computing Element



Storage Element



# Job submission




Output Sandbox files

RB node

RB storage

Network Server

Workload Manager

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

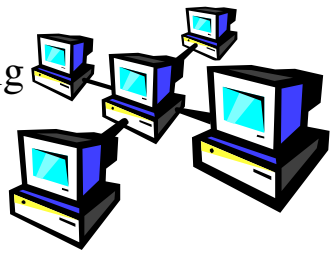
ready

scheduled

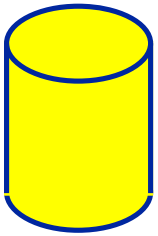
running

done

cleared

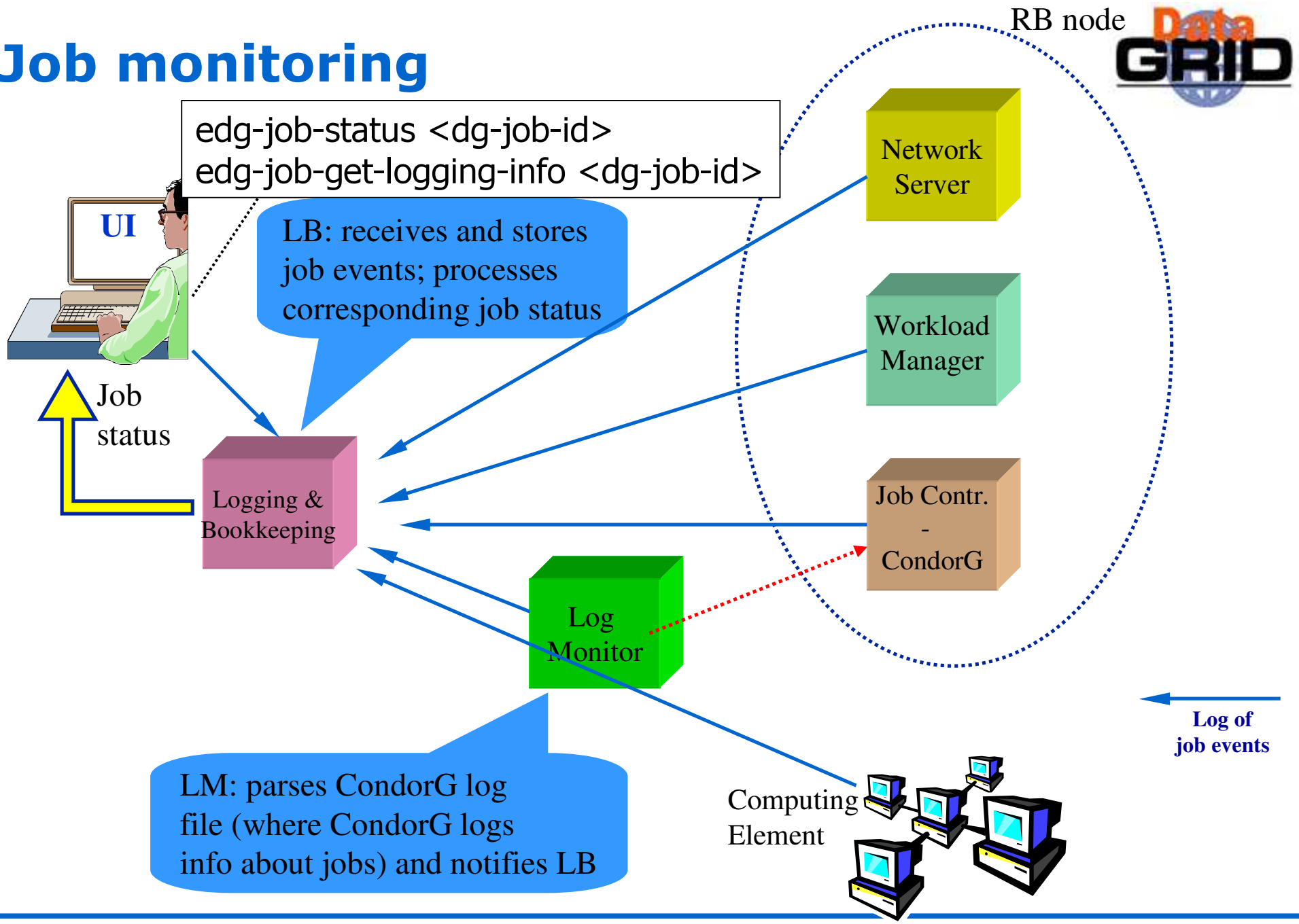


Computing Element



Storage Element

# Job monitoring



# Addressing reliability and scalability

- No more a monolithic long-lived process
  - Some functionalities (e.g. matchmaking) delegated to pluggable modules
  - Less exposed to memory leaks (coming not only from EDG software)
- No more multiple job info repositories
  - No more job status inconsistencies which caused problems
  - Also improvement from a user perspective (e.g job status not OutputReady, but it was possible to retrieve output sandbox)
- Reliable communications among components
  - Done via the file system (filequeues)
  - For example jobs are not lost if the target entity is temporary down: when it restarts it gets and "process" the jobs

# Addressing reliability and scalability

- No more STL string problems on SMP machines
  - Which caused problems to the RB and required the cleaning of internal databases and loss of all active jobs
  - WP1 sw built with gcc 3.2
- Integration of newer (v. 6.5.1) CondorG (deployed via VDT)
  - Better job submission system
  - No more 'max 512 concurrent jobs' limit
  - Actually still some issues to address (already in touch with Condor developers)

# Other improvements wrt release 1.x

- Security of sandbox files
  - Not possible anymore to access sandbox files of other users
  - Done via a patched gridftp server and via a proper grid-mapfile in the "RB node"
- Addressed disk space mgmt problem on the "RB node"
  - Could get completely filled with sandbox files
  - Problem addressed:
    - WMS admin can specify a maximum size for the input sandbox (if greater: job refused)
    - WMS can also specify a percentage of disk: if free disk less than this value, new jobs refused
    - Also possible to rely on disk quota (WMS admin can set disk quota for the various users)
    - Also possible to rely on "dynamic" disk quota (WMS admin can not set disk quota for the various users: when a job is submitted, its quota is automatically increased of a certain amount of space, which is released when job completes)
  - Also possible to run from time to time (e.g. via a cron job) a purger, which cleans "old" sandbox directories (configurable policies)

# Other improvements wrt release 1.x

- Improvements in LB
  - No more one LB server per "RB", but possible to have more LB servers
    - Could be useful in case of LB overloaded
  - Extended querying capabilities
    - E.g. Give me all jobs marked as 'XYZ' (user tag) and running on CE1 or C'E2
    - Necessary to create custom indices
      - LB server refuses to process a query which would not utilize any index to prevent overloading the underlying database engine
  - R-GMA interfaces
    - LB server capable of feeding the R-GMA infrastructure with notifications on job state changes
- Proxy renewal
  - Fixed (very silly) problem of release 1 (renewed proxy was too short)
  - Reliable proxy renewal service (doesn't forget about registered proxies in case of service restart)

# Other improvements wrt release 1.x

- Other typical rel. 1 problems fixed
  - CE chosen in a random way among the CEs which meet all the requirements and have the same best rank
  - No more JSSparser stuck → status of jobs don't updated
    - The problem was in PSQL, not used anymore
- Other fixes and improvements
- Much better stability sought in our internal tests
- But difficult for us to perform stress tests in our small WP1 testbed
  - Prepared to address problems we will find when performing the real integrated stress tests on the real big testbed

# Other improvements/changes

- Integration with WP2 software
  - Interaction with WP2 RLS (instead of the "old" RC to have the logical → physical file name mappings)
  - Possible to rely on the WP2 getAccessCost as JDL rank
    - I.e. getAccessCost finds the best CE among the ones meeting all the requirements (taking into account data location)
- Integration with R-GMA (wrt Information Services) completely transparent for WP1 sw (via GIN, GOUT mechanisms)
- New Glue schema
  - → JDL expressions must rely on this new IS schema

# New functionalities introduced

- User APIs
  - Including a Java GUI
- "Trivial" job checkpointing service
  - User can save from time to time the state of the job (defined by the application)
  - A job can be restarted from an intermediate (i.e. "previously" saved) job state
  - Presented at the EDG review
- Gangmatching
  - Allow to take into account both CE and SE information in the matchmaking
    - For example to require a job to run on a CE close to a SE with "enough space"
- Output data upload and registration
  - Possible to trigger (via proper JDL attribute) automatic data upload and registration at job completion

# New functionalities introduced

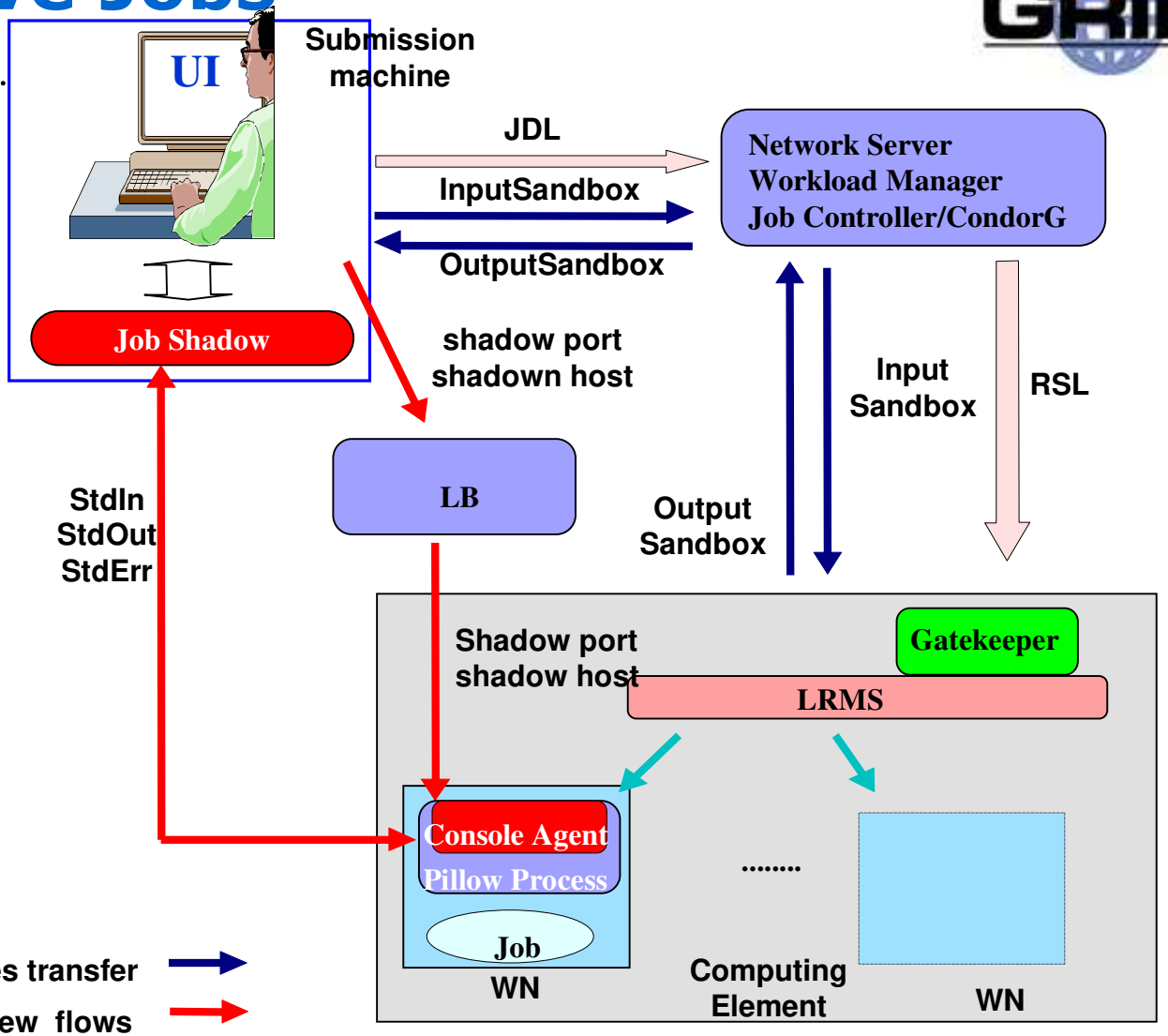
- Support for parallel MPI jobs
  - Parallel jobs within single CEs
- Grid accounting services
  - Just for economic accounting for Grid User and Resources
    - The users pay for resource usage while the resources earn virtual credits executing user jobs
  - No integration with "brokering" module
- Support for interactive jobs
  - Jobs running on some CE worker node where a channel to the submitting (UI) node is available for the standard streams (by integrating the Condor Bypass software)

# Interactive Jobs

edg-job-submit jobint.jdl

jobint.jdl

```
[JobType = "interactive";
ListenerPort = 2654;
Executable      = "int-prg.exe";
StdOutput = Outfile;
InputSandbox = "/home/user/int-prg.exe",
OutputSandbox = "Outfile",
Requirements =
other. GlueHostOperatingSystemName ==
"linux" &&
Other.GlueHostOperatingSystemRelease ==
"RH 6.2";]
```



# Future functionalities

- Hooks in place for other future functionalities
  - Dependencies of jobs
    - Integration of Condor DAGMan
    - "Lazy" scheduling: job (node) bound to a resource (by RB) just before that job can be submitted (i.e. when it is free of dependencies)
  - Support for job partitioning
    - Use of job checkpointing and DAGMan mechanisms
      - Original job partitioned in sub-jobs which can be executed in parallel
      - At the end each sub-job must save a final state, then retrieved by a job aggregator, responsible to collect the results of the sub-jobs and produce the overall output
  - Integration of Grid Accounting with "matchmaking" module
    - Based upon a computational economy model
      - Users pay in order to execute their jobs on the resources and the owner of the resources earn credits by executing the user jobs
    - To have a nearly stable equilibrium able to satisfy the needs of both resource `producers' and `consumers'
  - Advance reservation and co-allocation
    - Globus GARA based approach
- Development of these new functionalities already started (most of this software already in a good shape)

# Conclusions

- Revised WMS architecture
  - To address emerged shortcomings, e.g.
    - Reduce of persistent job info repositories
    - Avoid long-lived processes
    - Delegate some functionalities to pluggable modules
    - Make more reliable communication among components
  - To support new functionalities
    - APIs, Interactive jobs, Job checkpointing, Gangmatching, …
  - Hooks to support other functionalities planned to be integrated later
    - DAGman, Job partitioning, Resource reservation and co-allocation, …