

# The EU DataGrid Workload Management System: towards the second major release



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

[massimo.sgaravatto@pd.infn.it](mailto:massimo.sgaravatto@pd.infn.it)





# Talk Outline

- ◆ DataGrid and Datagrid WP1
- ◆ The DataGrid Workload Management System
- ◆ The new revised WP1 WMS architecture
  - What's has been revised and new functionalities
- ◆ Future work & conclusions

## Authors

G. Avellino, S. Beco, B. Cantalupo, F. Pacini, A. Terracina, A. Maraschini (*DATAMAT S.p.A*)

D. Colling (*Imperial College*)

S. Monforte, M. Pappalardo (*INFN, Sezione di Catania*)

L. Salconi (*INFN, Sezione di Pisa*)

F. Giacomini, E. Ronchieri (*INFN, CNAF*)

D. Kouril, A. Krenek, L. Matyska, M. Mulac, J. Pospisil, M. Ruda, Z. Salvat, J. Sitek, M. Vocu (*CESNET*)

M. Mezzadri, F. Prelz (*INFN, Sezione di Milano*)

Gianelle, R. Peluso, **M. Sgaravatto** (*INFN, Sezione di Padova*)

S. Barale, A. Guarise, A. Werbrouck (*INFN, Sezione di Torino*)



# DataGrid and DataGrid WP1

## ◆ European DataGrid Project

- Goal: Grid software projects meet real-life scientific applications (High Energy Physics, Earth Observation, Biology) and their deadlines, with mutual benefit
- Middleware development and integration of existing middleware
- Bring the issues of data identification, location, transfer and access into the picture
- Large scale testbed

## ◆ WP1 (Grid Workload Management)

- Mandate: *"To define and implement a suitable architecture for distributed scheduling and resource management on a GRID environment"*
- This includes the following areas of activity:
  - Design and development of a useful (as seen from the DataGrid applications perspective) grid scheduler, or Resource Broker
  - Design and development of a suitable job description and monitoring infrastructure
  - Design and implementation of a suitable job accounting structure



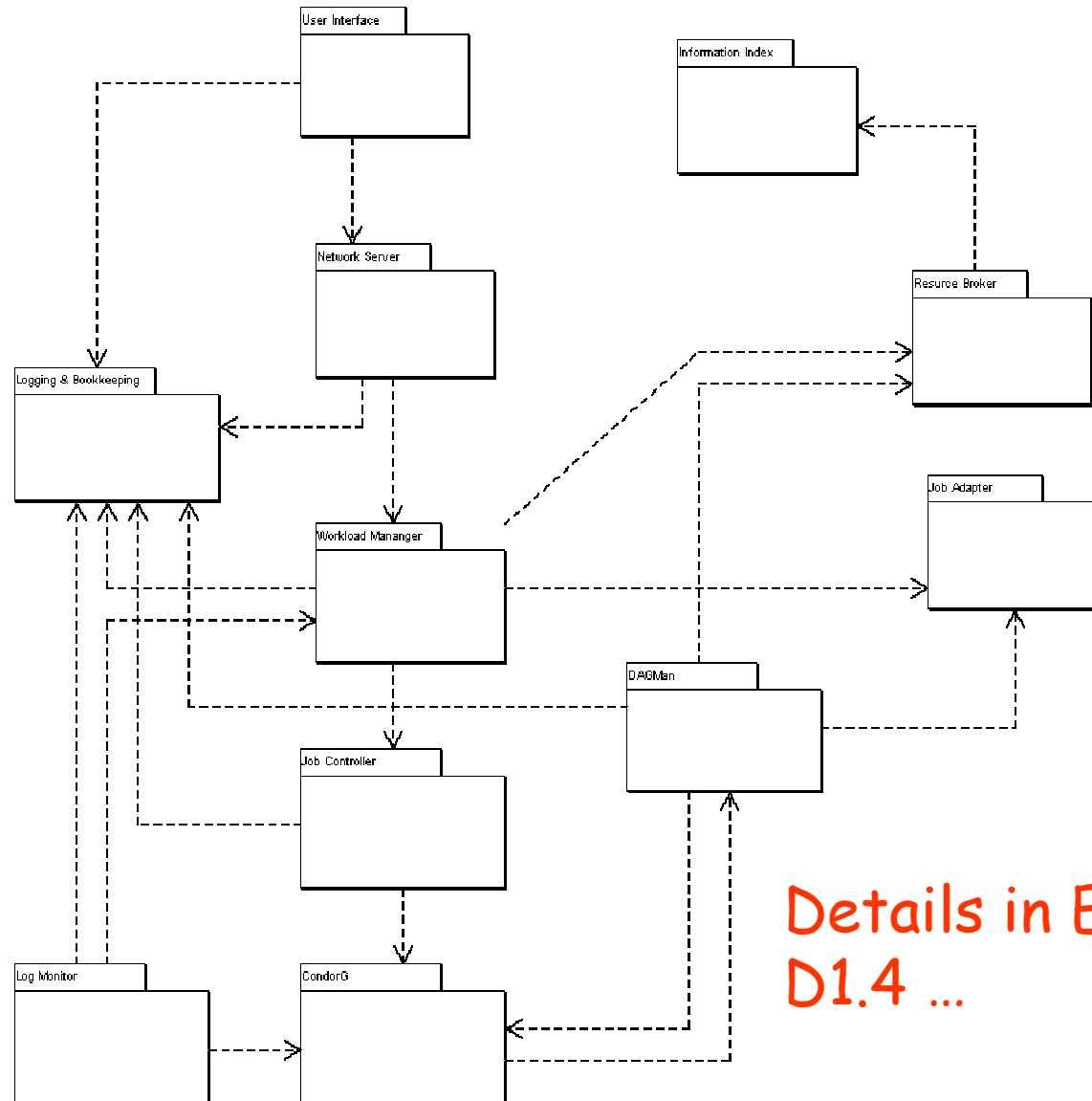
# WP1 Workload Management System

- ◆ Working Workload Management System prototype implemented by WP1 in the first phase of the EDG project (presented at CHEP2001)
  - Ability to define (via a Job Description Language, JDL) a job, submit it to the DataGrid testbed from any user machine, and control it
  - WP1's Resource Broker chooses an appropriate computing resource for the job, based on the constraints specified in the JDL
    - Where the submitting user has proper authorization
    - That matches the characteristics specified in the job JDL (Architecture, computing power, application environment, etc.)
    - Where the specified input data (and possibly the chosen output Storage Element) are determined to be "close enough" by the appropriate resource administrators
- ◆ Application users have now been experiencing for about one year and a half with this first release of the workload management system
  - Stress tests and semi-production activities (e.g. CMS stress tests, Atlas efforts)
  - Significant achievements exploited by the experiments but also various problems were spotted
    - Impacting in particular the reliability and scalability of the system

# Review of WP1 architecture

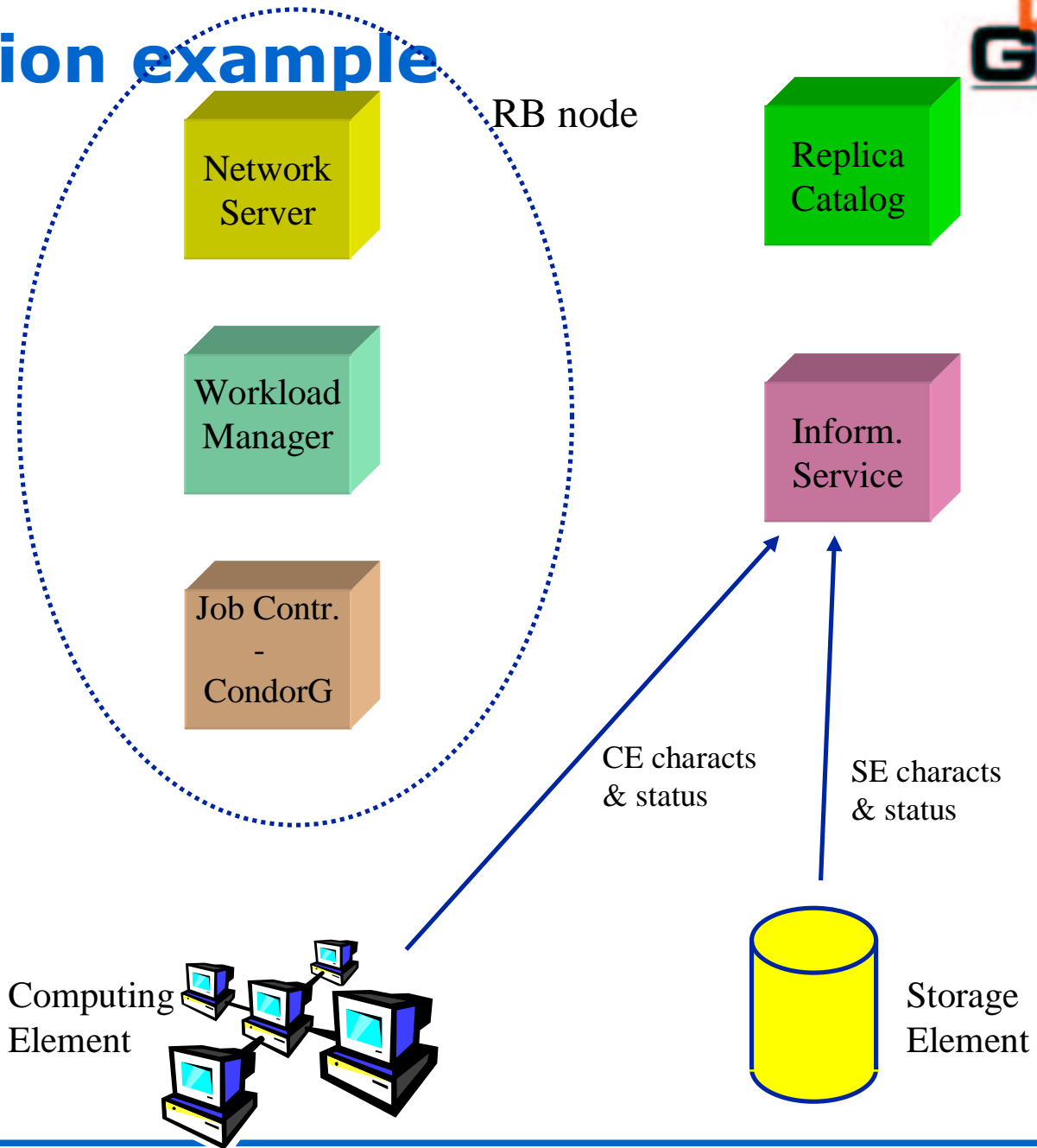
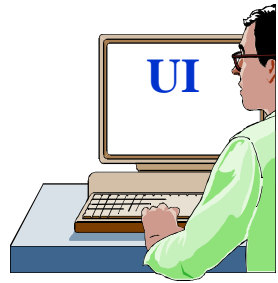
- ◆ WP1 WMS architecture reviewed
  - To apply the “lessons” learned and addressing the shortcomings emerged with the first release of the software
    - Reduce of persistent job info repositories
    - Avoid long-lived processes
    - Delegate some functionalities to pluggable modules
    - Make more reliable (e.g.via file system) communication among components
    - ... (see also other EDG WP1 talk given by Francesco Prelz)
  - To increase the reliability of the system
  - To support new functionalities
  - To favor interoperability with other Grid frameworks, by allowing exploiting WP1 modules (e.g. RB) also “outside” the WP1 WMS
  - New WMS (v. 2.0) presented at the 2<sup>nd</sup> EDG review and scheduled for integration at April 2003

# WP1 reviewed architecture

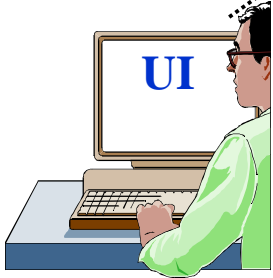


Details in EDG deliverable D1.4 ...

# Job submission example



# Job subm



```
edg-job-submit myjob.jdl
```

```
Myjob.jdl
```

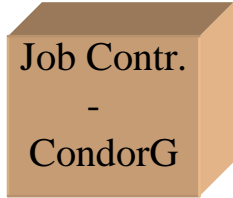
```
JobType = "Normal";  
Executable = "${CMS}/exe/sum.exe";  
InputData = "LF:testbed0-00019";  
ReplicaCatalog = "ldap://sunlab2g.cnaf.infn.it:2010/rc=WP2 INFN Test Replica  
Catalog,dc=sunlab2g,dc=cnaf,dc=infn,dc=it";  
DataAccessProtocol = "gridftp";  
InputSandbox = {"/home/user/WP1testC", "/home/file*", "/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

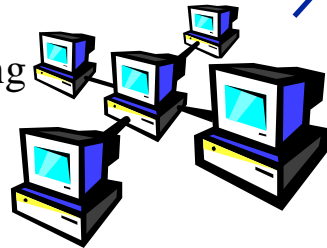
submitted

UI: allows users to access the functionalities of the WMS

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

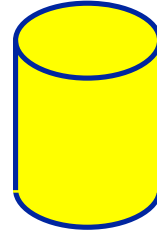


Computing Element



CE characts & status

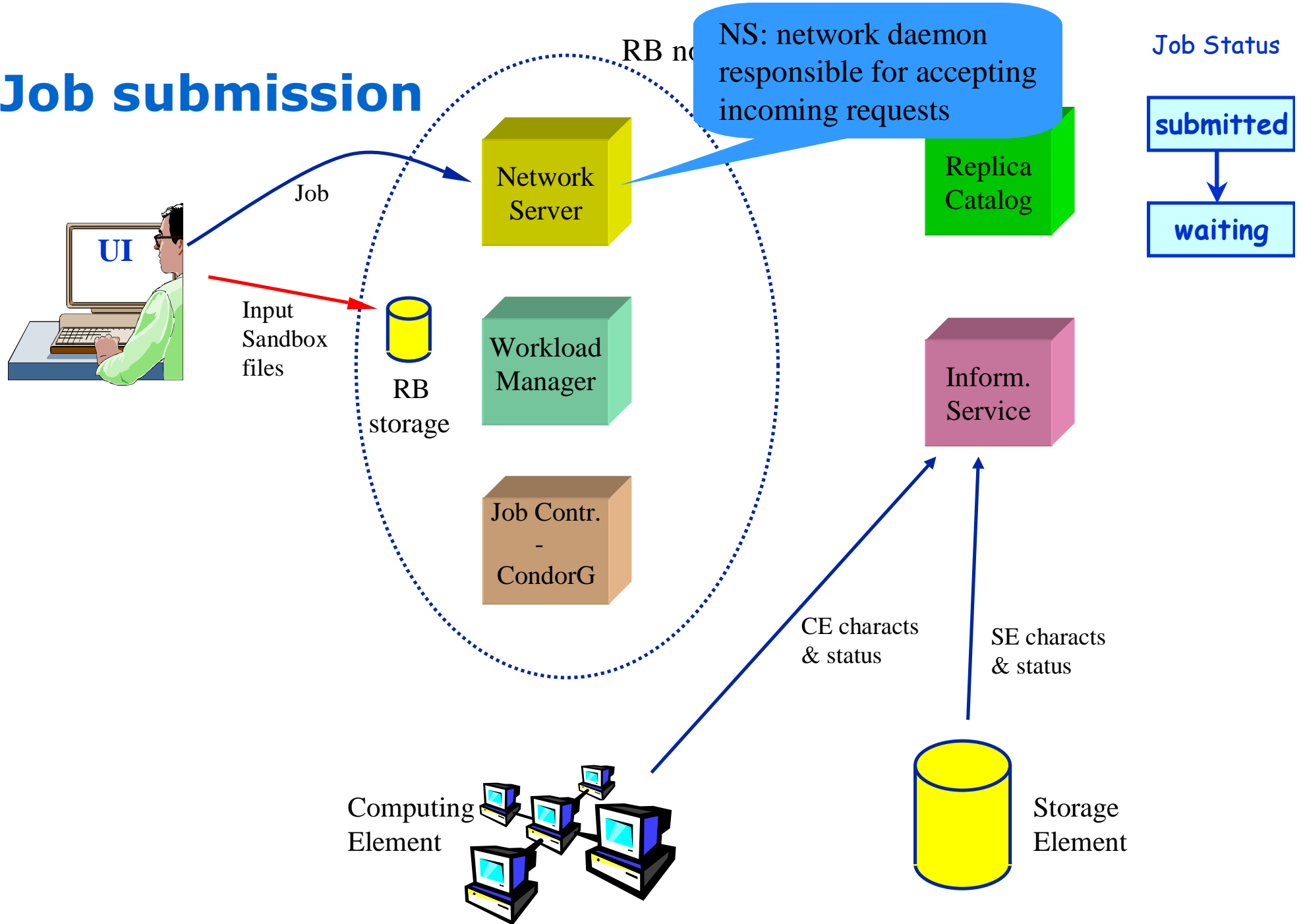
SE characts & status



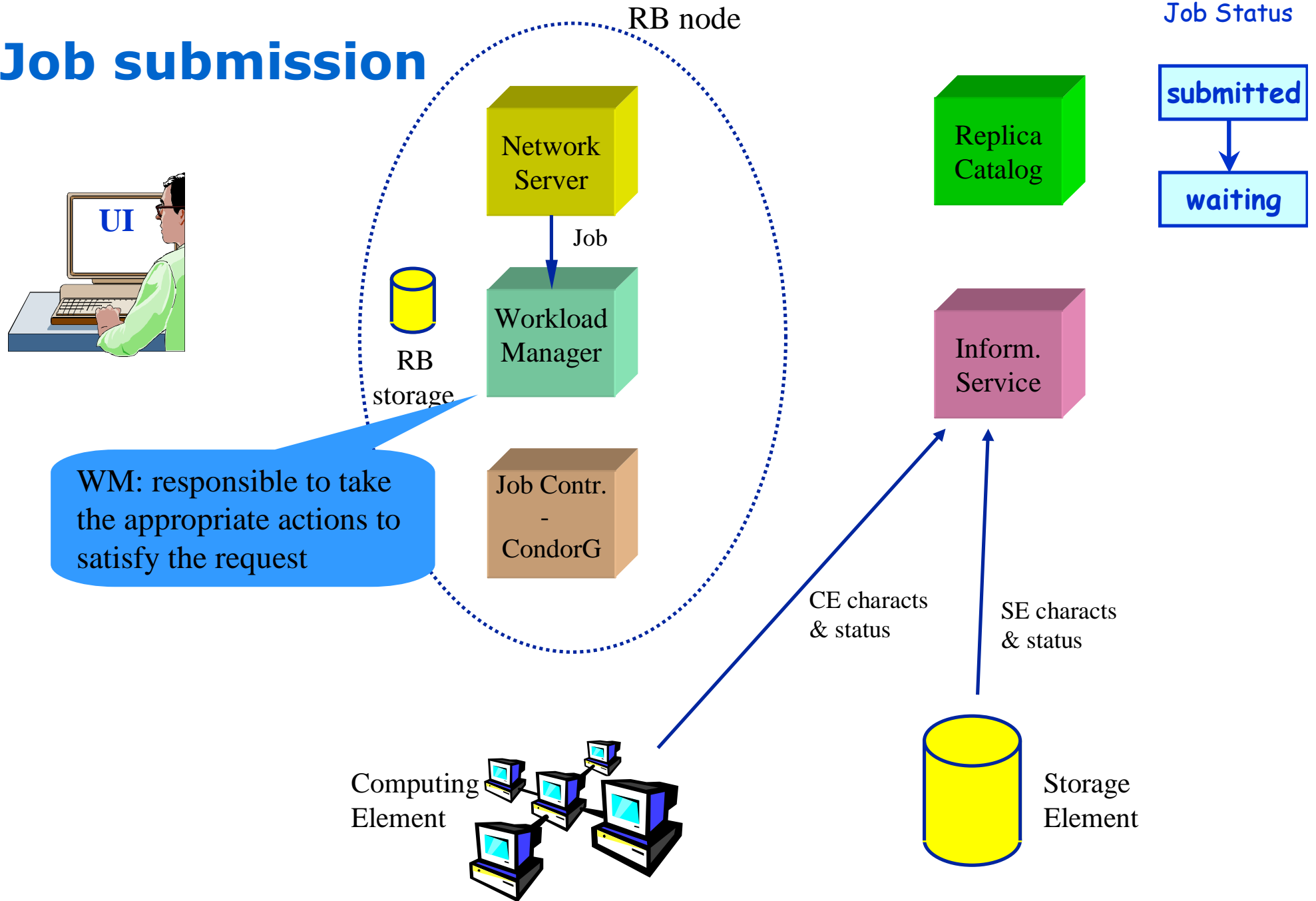
Storage Element



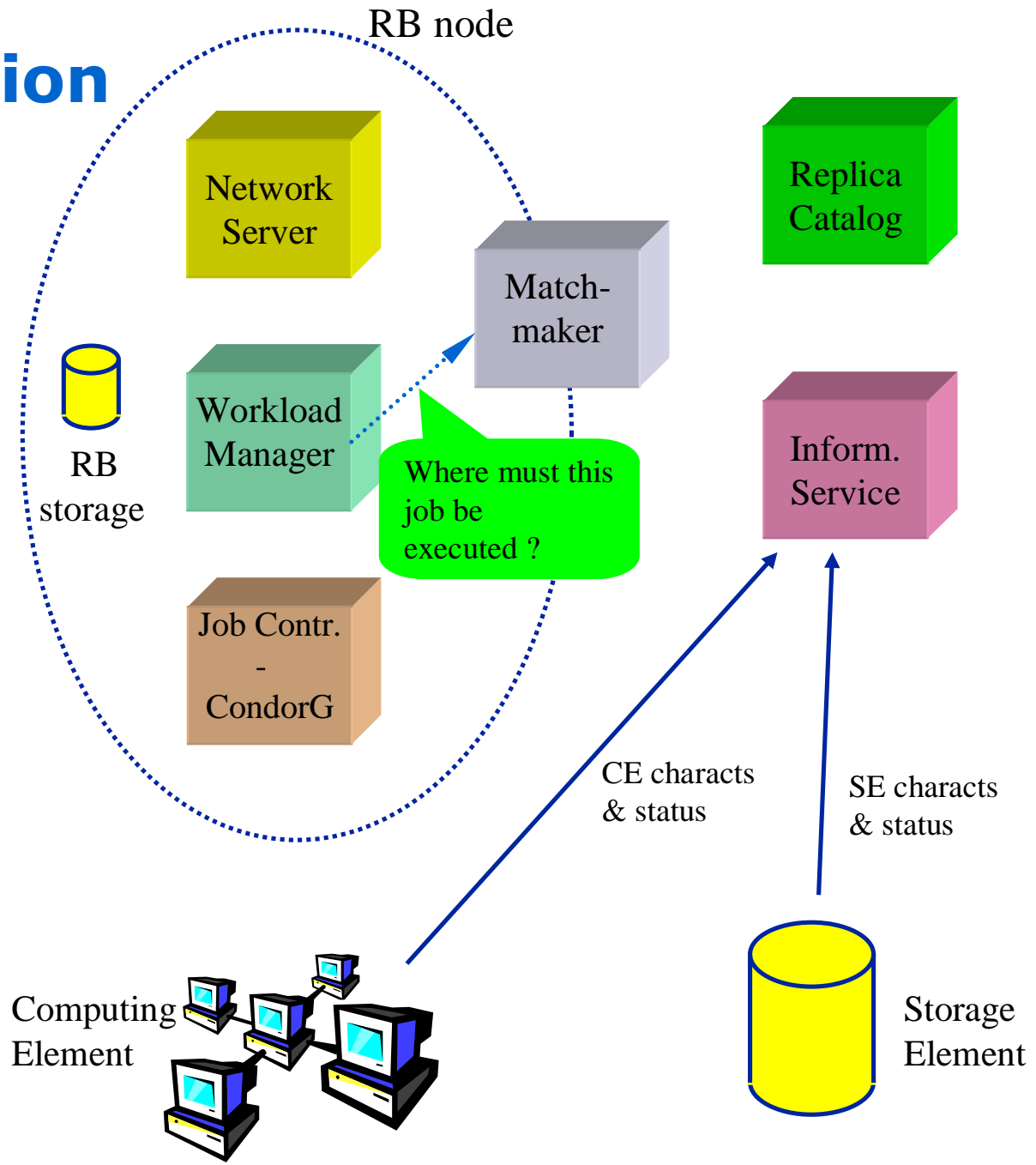
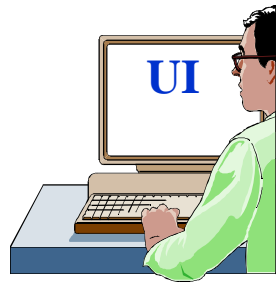
# Job submission



# Job submission



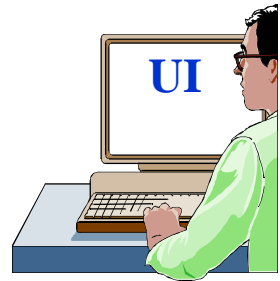
# Job submission



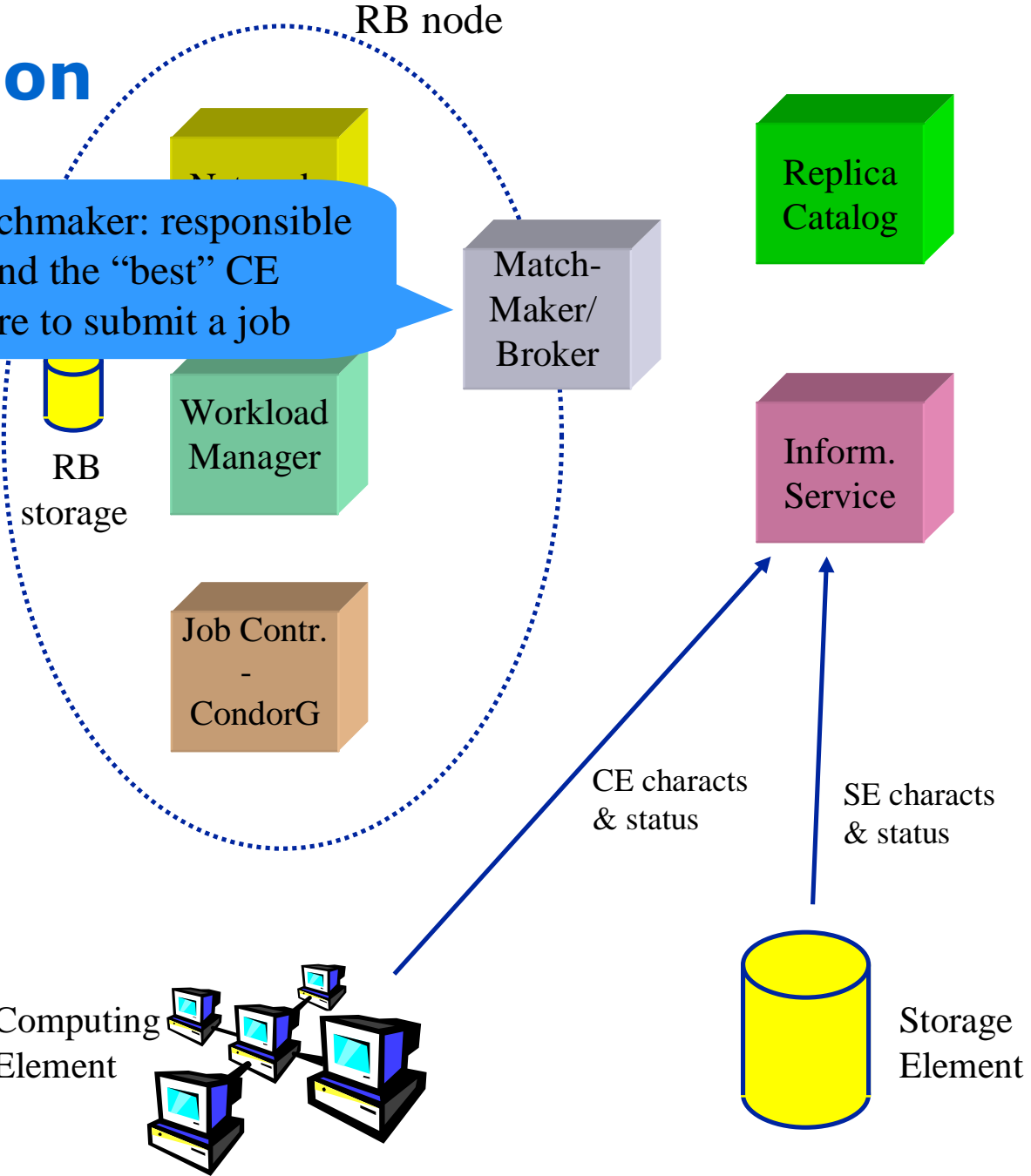
## Job Status



# Job submission



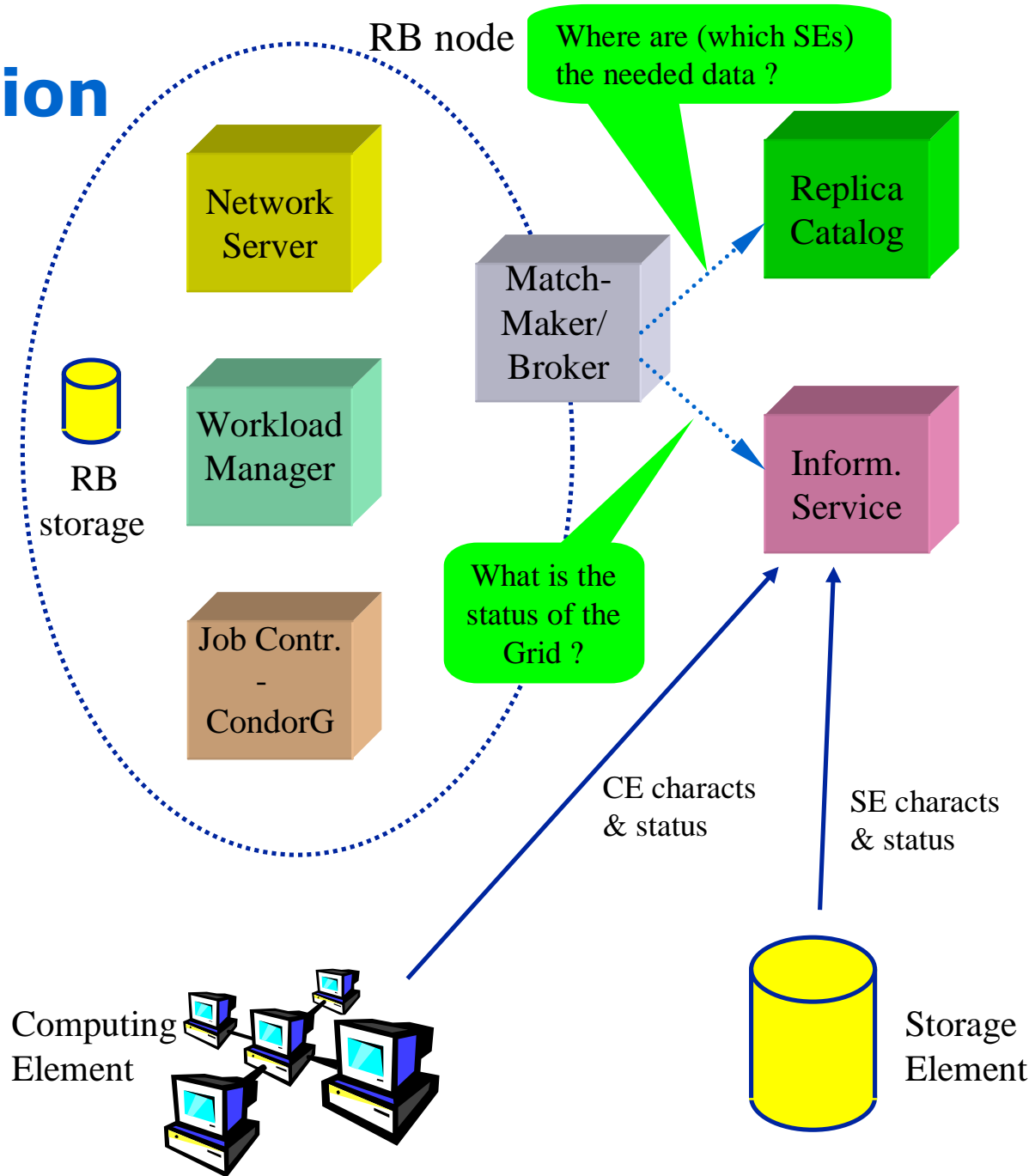
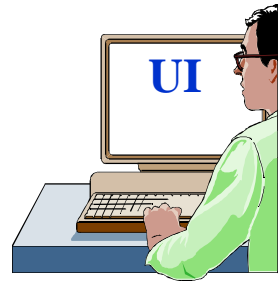
Matchmaker: responsible to find the "best" CE where to submit a job



## Job Status



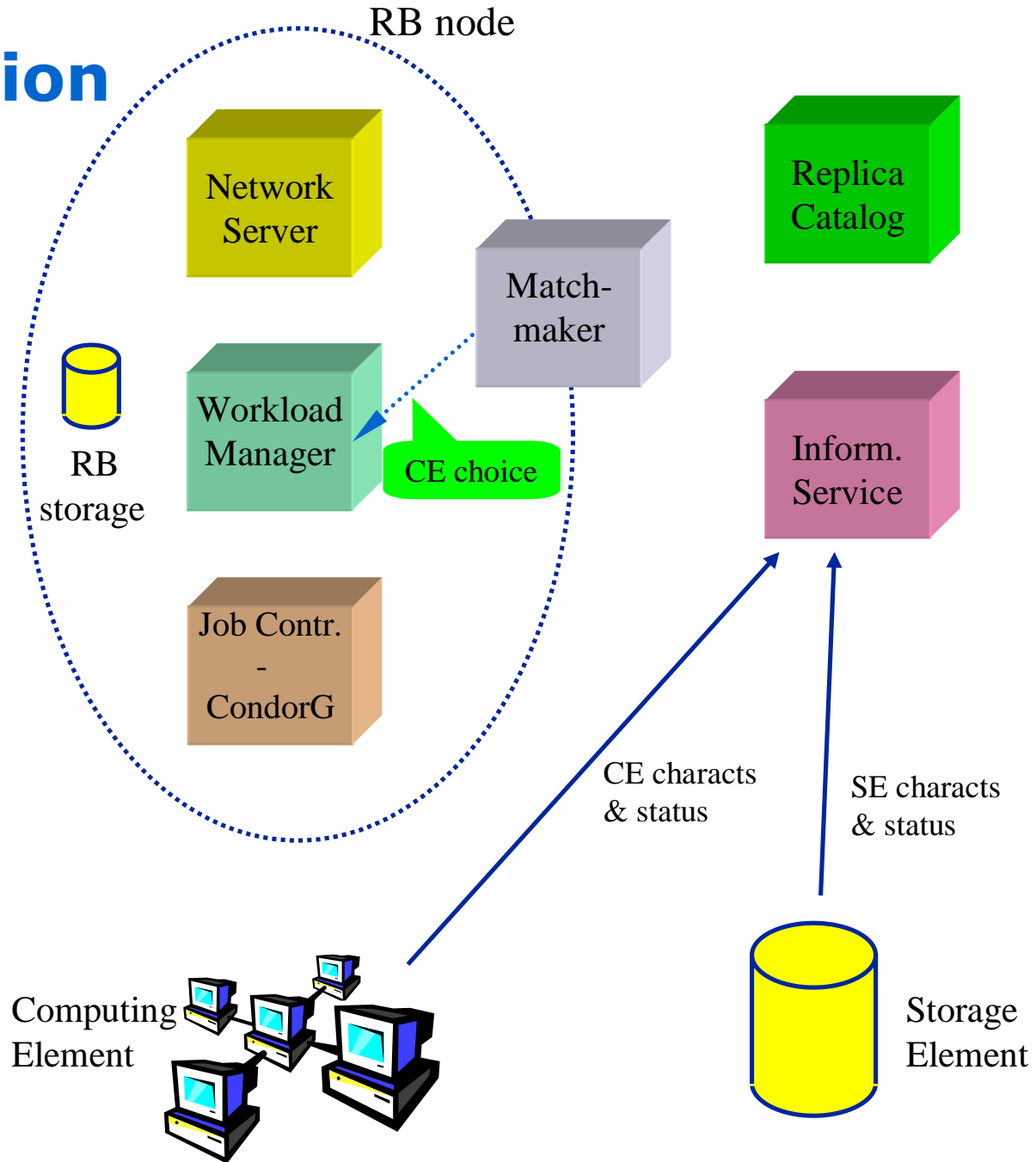
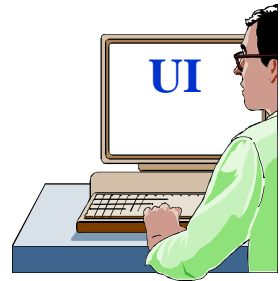
# Job submission



Job Status



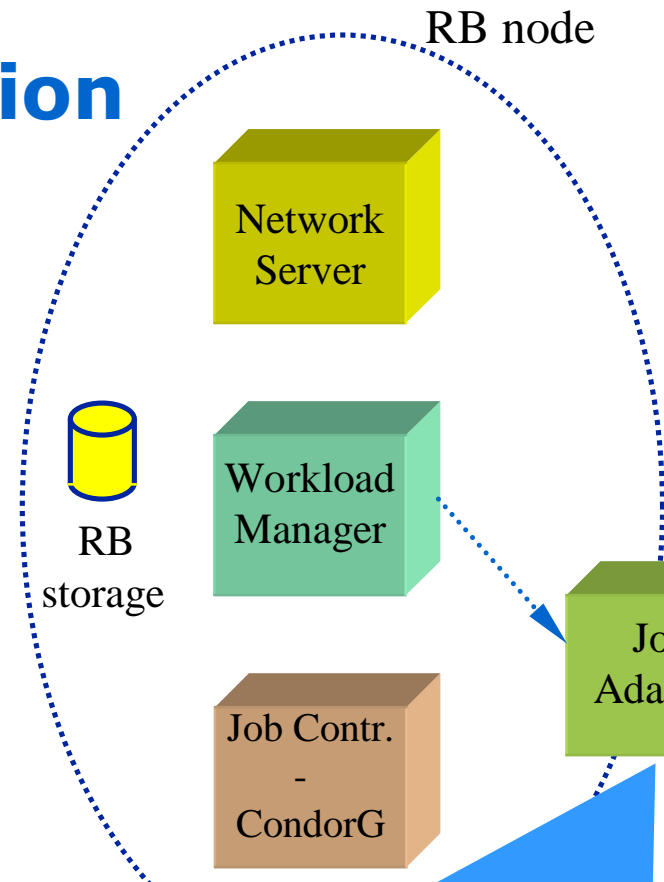
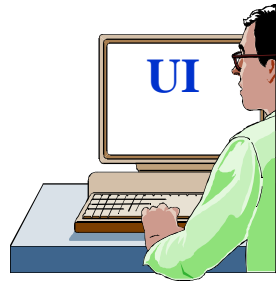
# Job submission



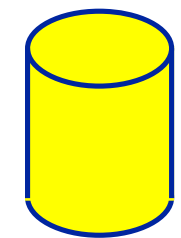
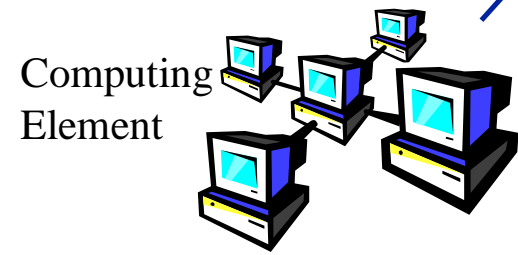
## Job Status



# Job submission



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



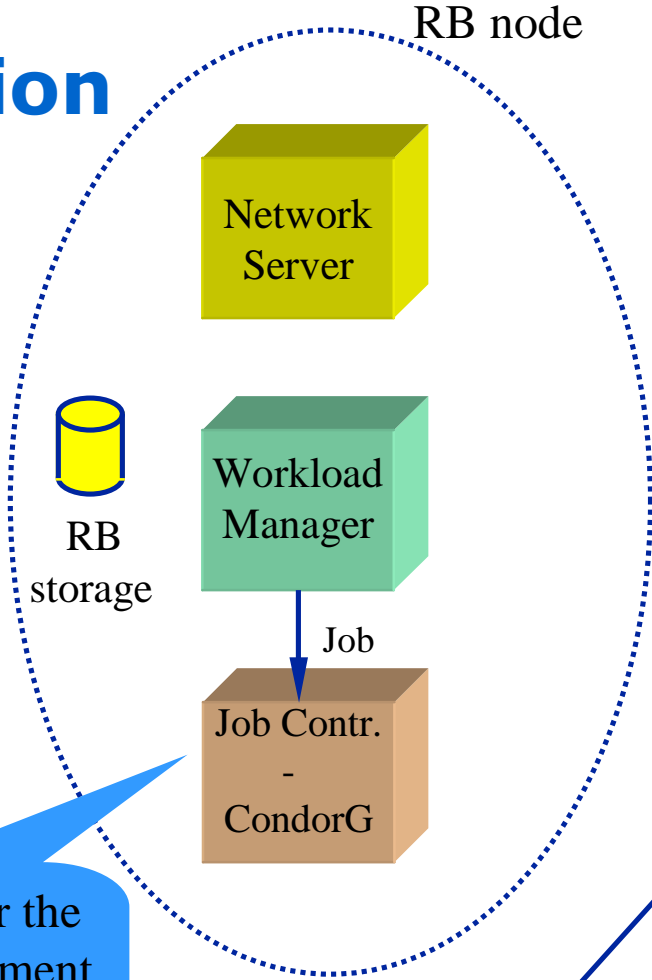
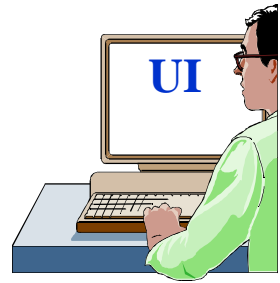
Job Status



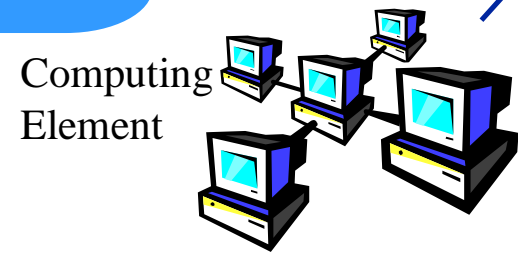
SE characts & status

characts

# Job submission

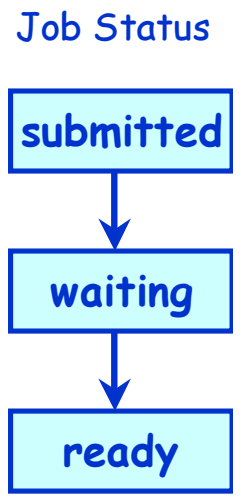
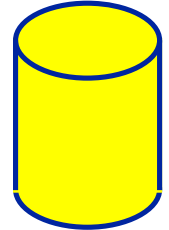


JC: responsible for the actual job management operations (done via CondorG)



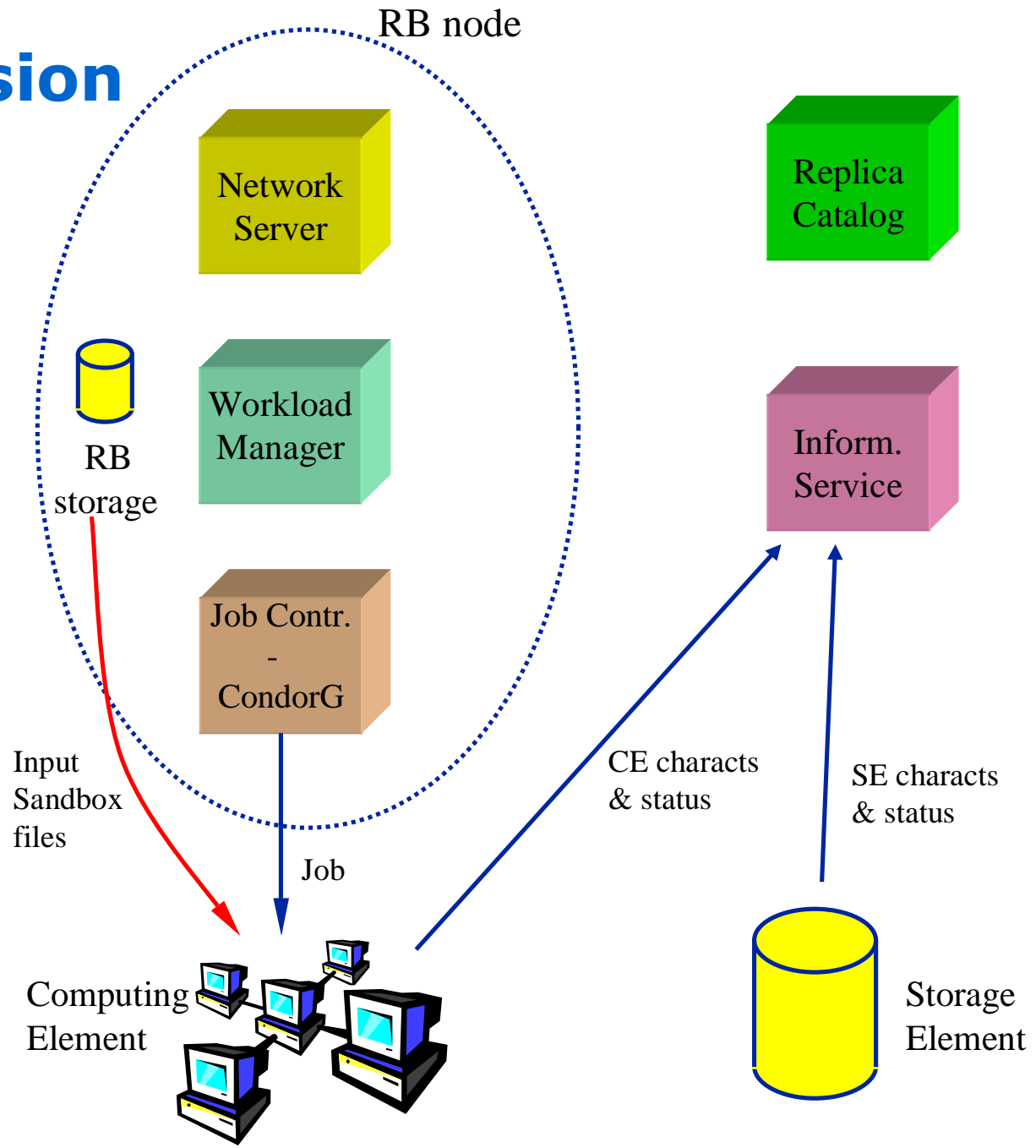
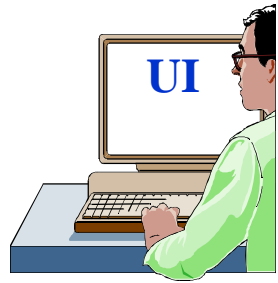
CE characts & status

SE characts & status





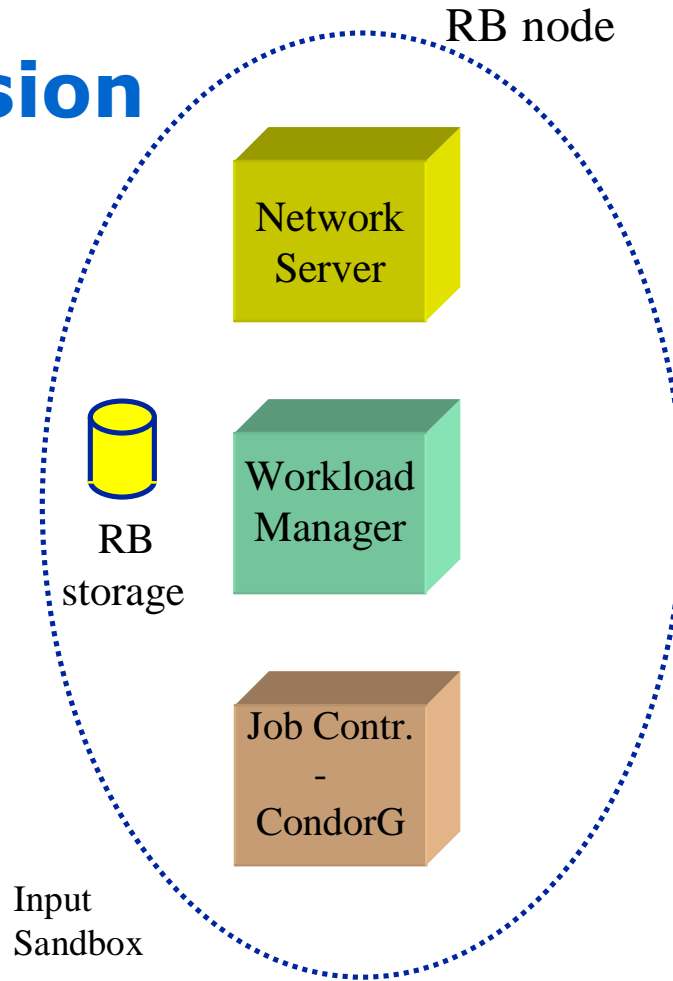
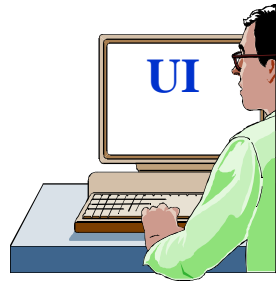
# Job submission



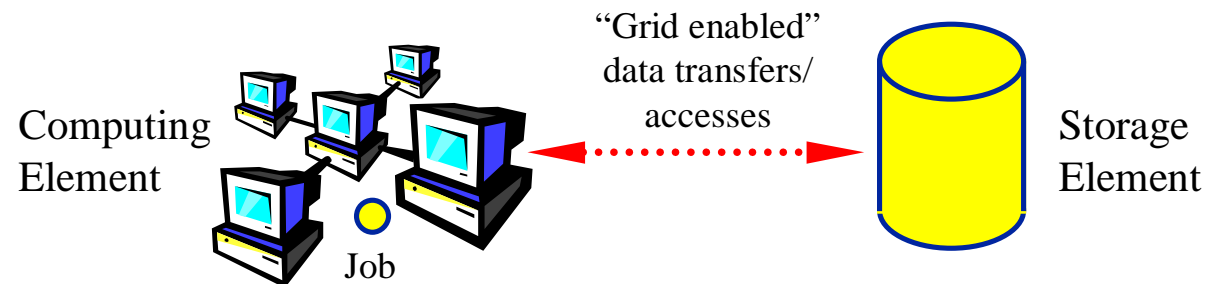
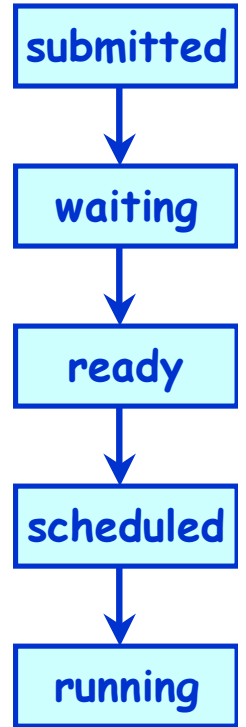
## Job Status



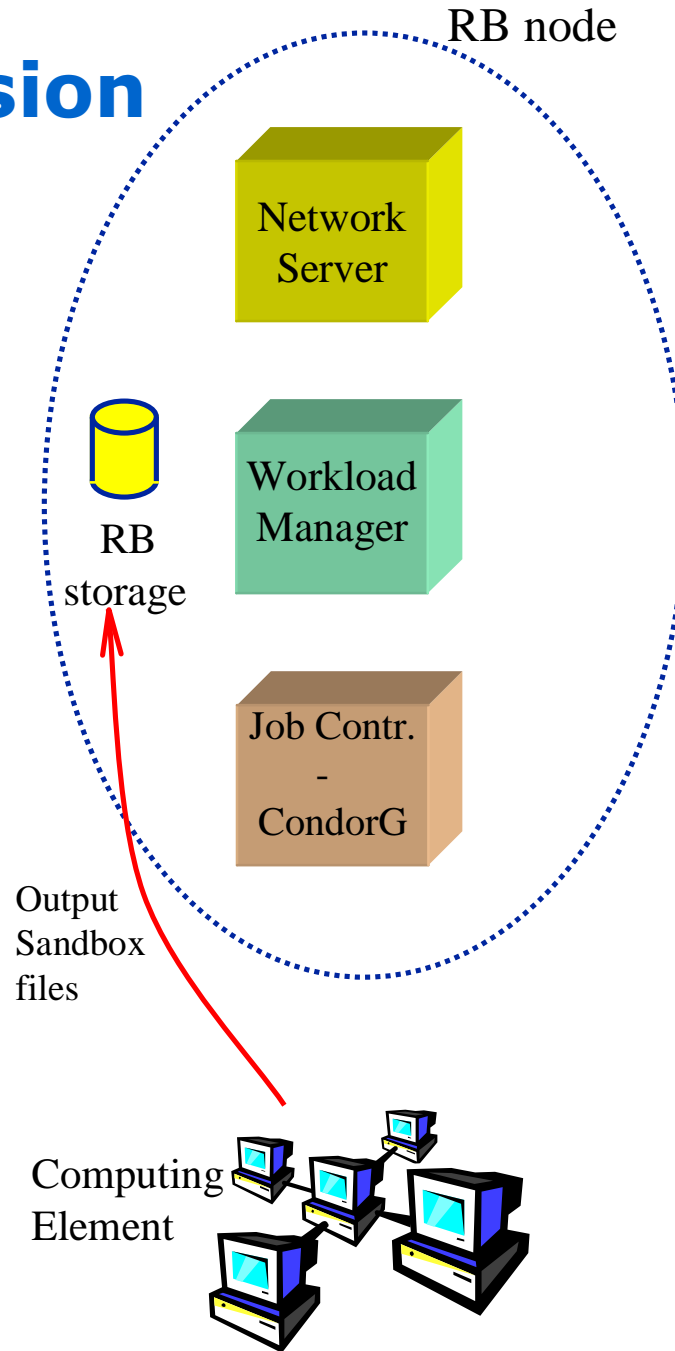
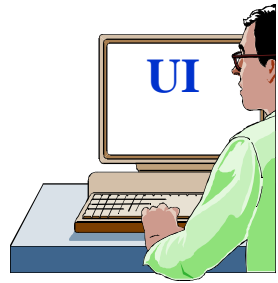
# Job submission



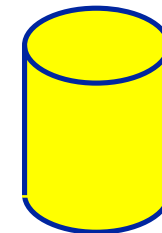
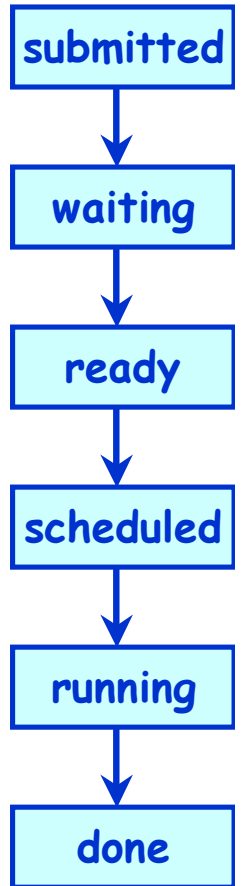
## Job Status



# Job submission



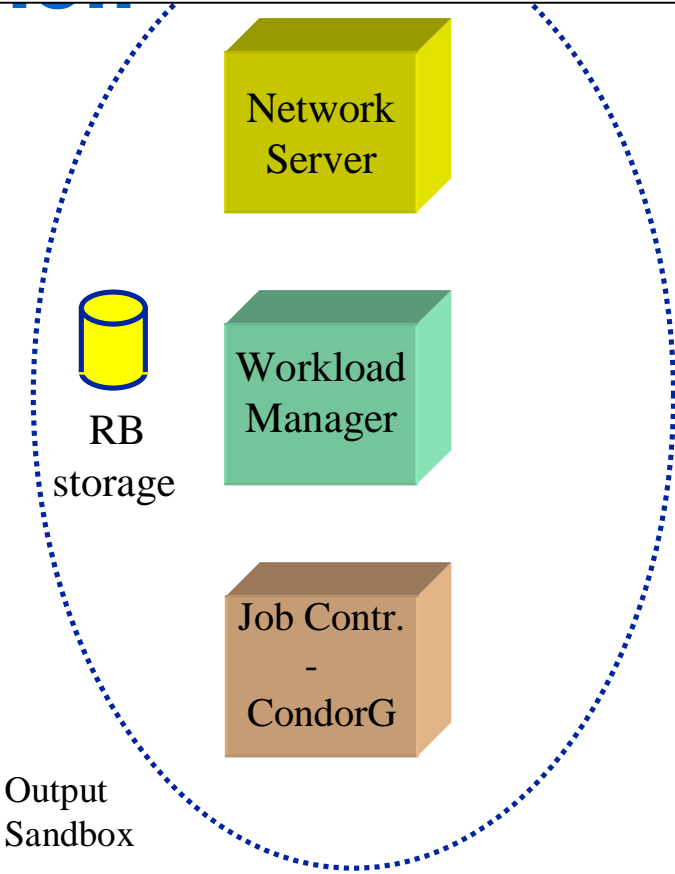
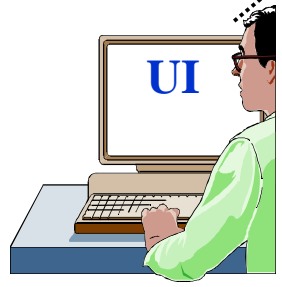
## Job Status



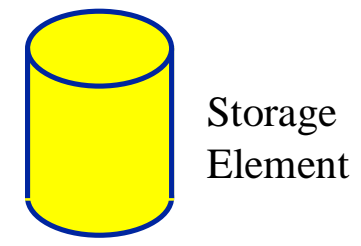
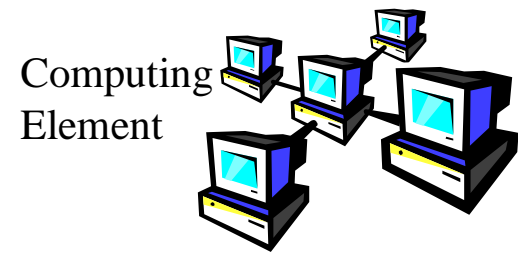
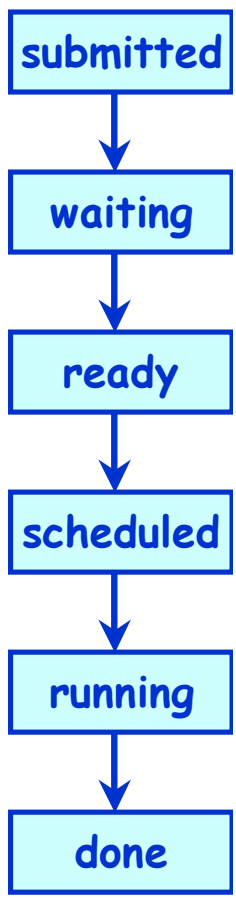
Storage Element

# Job submission

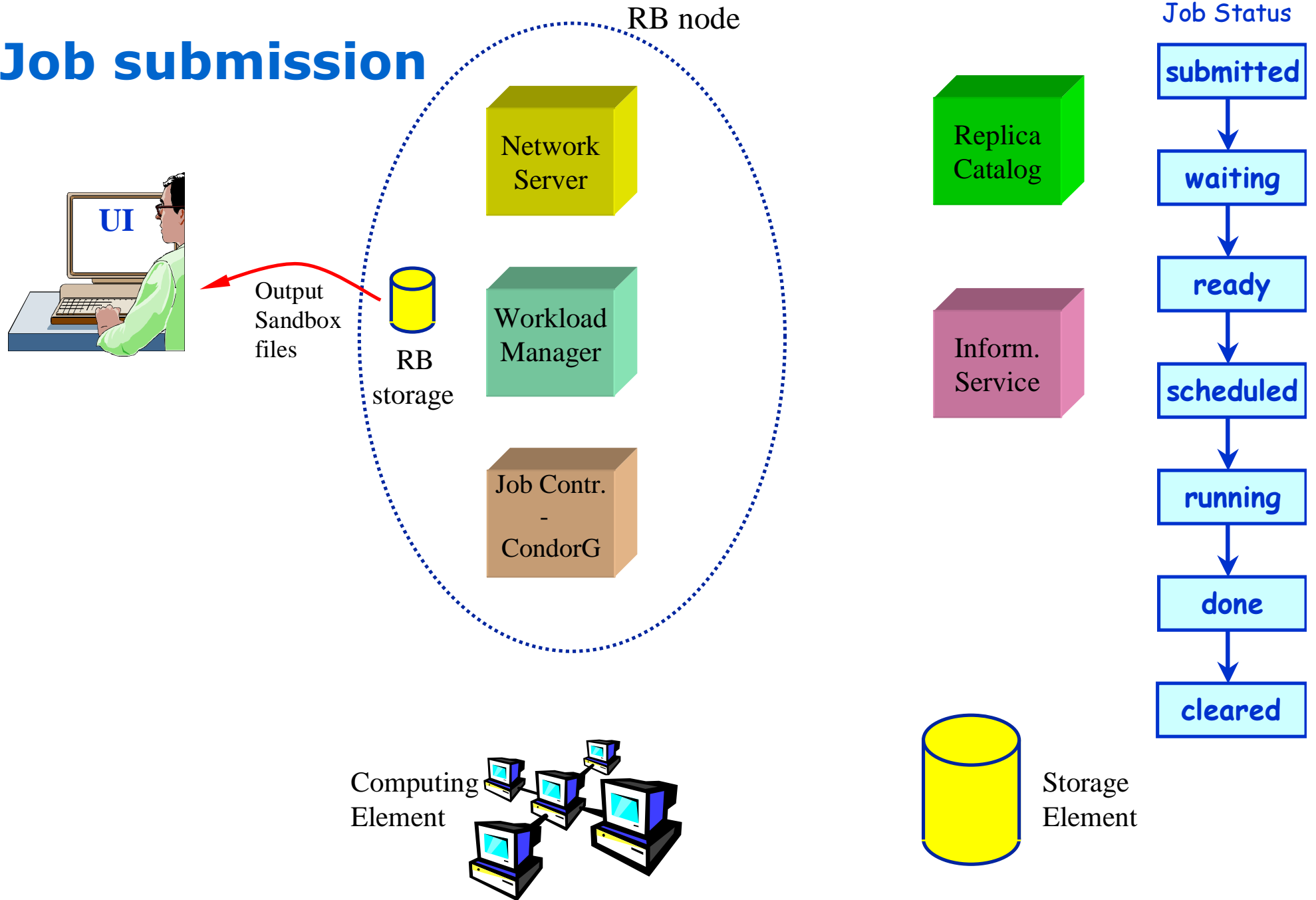
```
edg-job-get-output <dg-job-id>
```



## Job Status



# Job submission

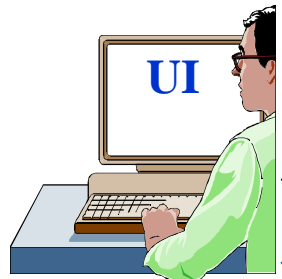


# Job monitoring



RB node

edg-job-status <dg-job-id>



LB: receives and stores job events; processes corresponding job status

Job status

Logging & Bookkeeping

Network Server

Workload Manager

Job Contr. - CondorG

Log Monitor

LM: parses CondorG log file (where CondorG logs info about jobs) and notifies LB



Log of job events



# New functionalities introduced

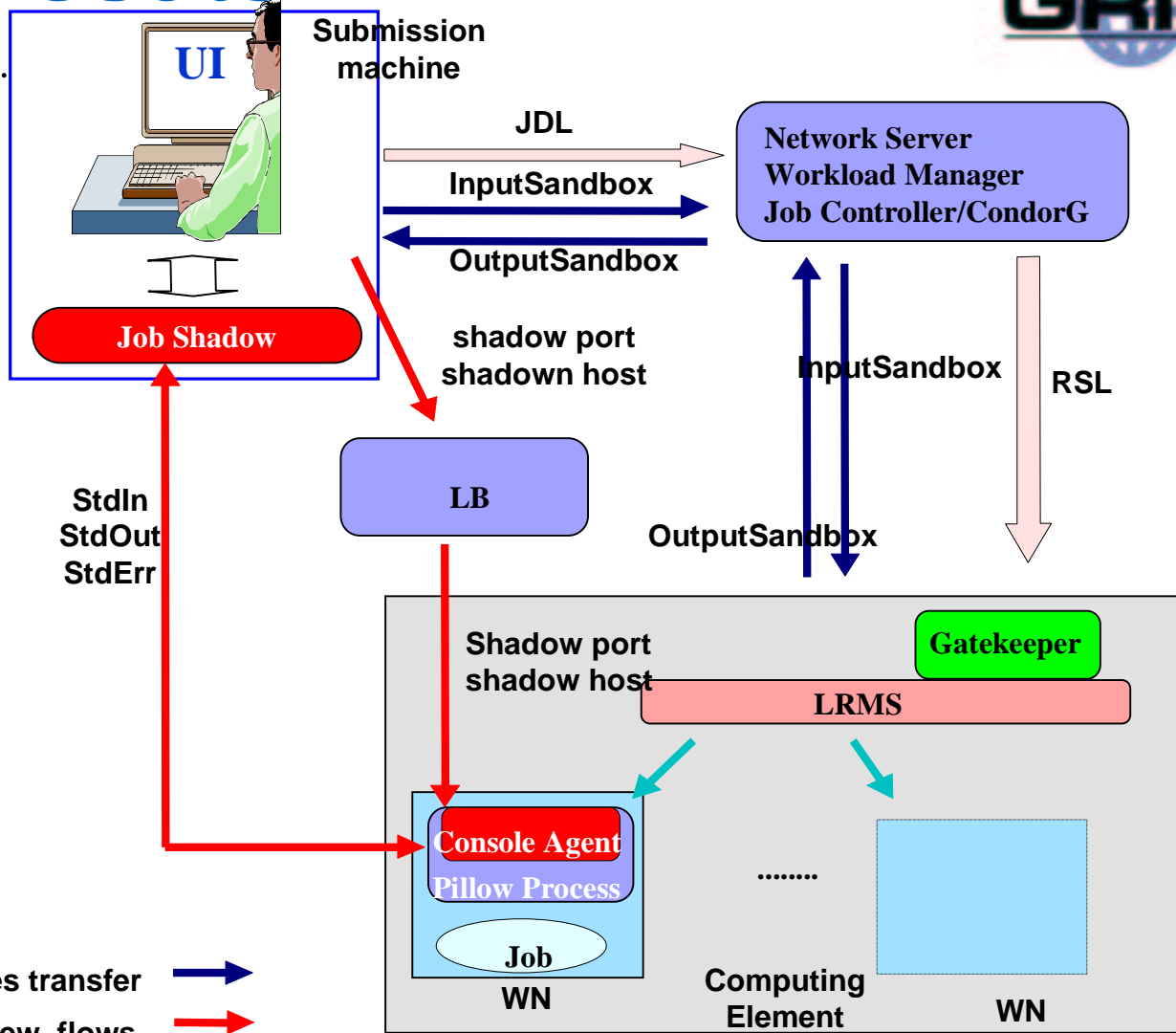
- ◆ User APIs
  - Including a Java GUI
- ◆ “Trivial” job checkpointing service
  - User can save from time to time the state of the job (defined by the application)
  - A job can be restarted from an intermediate (i.e. “previously” saved) job state
- ◆ Glue schema compliance
- ◆ Gangmatching
  - Allow 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”
- ◆ Integration of EDG WP2 Query Optimisation Service
  - Help for RB to find the best CE based on data location
- ◆ Support for parallel MPI jobs
- ◆ 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";]
    
```



Files transfer →  
 New flows →  
 Usual Job Submission flows →



# 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
- ◆ Grid Accounting
  - 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'
  - To credit of job resource usage to the resource owner(s) after execution
- ◆ Advance reservation and co-allocation
  - Globus GARA based approach
- ◆ Development already started (most of this software already in a good shape) but integration foreseen after release 2.0

# Conclusions

- ◆ In the first phase of the EDG project, WP1 implemented a working Workload Management System prototype
- ◆ Applications have been experiencing with this WMS for one year and a half
- ◆ Revised WMS architecture (WMS v. 2.0 planned for integration in Apr 2003)
  - 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, Grid accounting, Resource reservation and co-allocation
- ◆ Other info
  - <http://www.infn.it/workload-grid> (Home page for EDG WP1)
  - <http://www.eu-datagrid.org> (Home page for EDG project)

Thanks to the EU and our national funding agencies for their support of this work