Enabling Grids for E-science in Europe

Middleware components in EGEE

Mike Mineter
NeSC Training team
mjm@nesc.ac.uk

http://egee-intranet.web.cern.ch

EGEE is a project funded by the European Union under contract IST-2003-508833

Acknowledgements

This presentation includes slides and information from many sources:

- Roberto Barbera (Slides on middleware are based on presentations given in Edinburgh, April 2004)
- Other colleagues in EGEE (project overview slides)
- The European DataGrid training team
- Authors of the LCG-2 User Guide v. 2.0 : Antonio Delgado Peris, Patricia Méndez Lorenzo, Flavia Donno, Andrea Sciabà, Simone Campana, Roberto Santinelli https://edms.cern.ch/file/454439//LCG-2-UserGuide.html

Outline

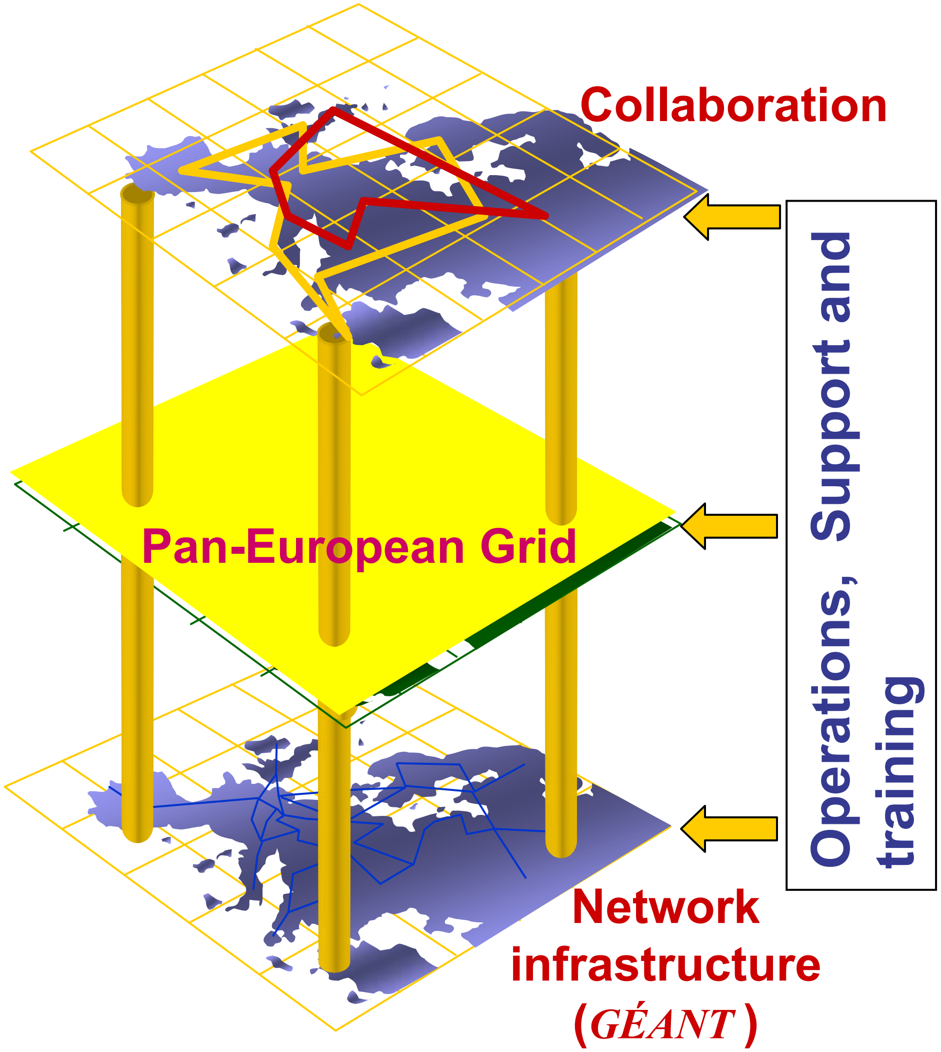
- Overview
- Major components
- Data management
- Lifecycle of a job
- Summary

Towards a European e-Infrastructure

- To underpin European science and technology in the service of society

- To link with and build on
  - National, regional and international initiatives
  - Emerging technologies (e.g. fibre optic networks)

- To foster international cooperation
  - both in the creation and the use of the e-infrastructure

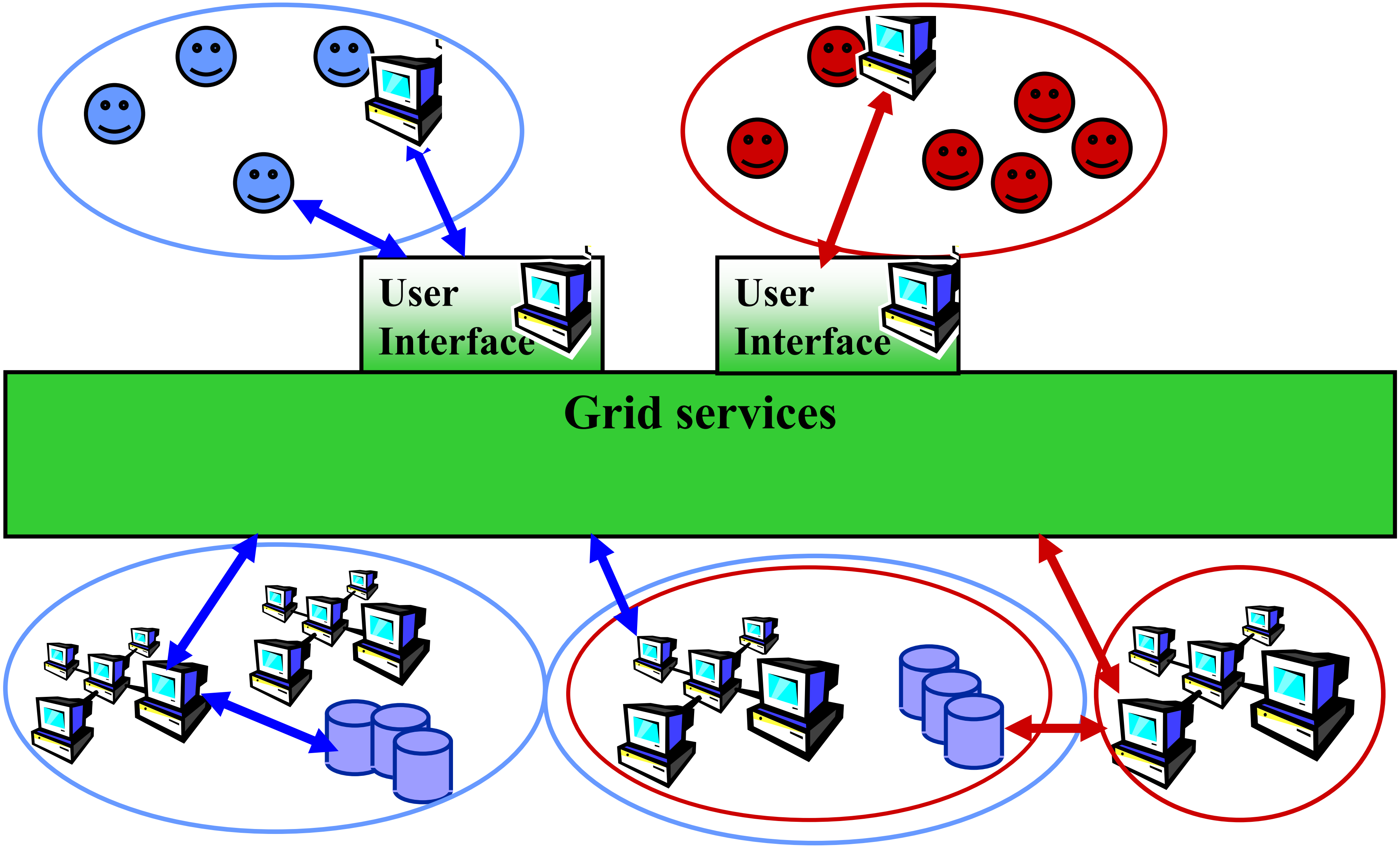

Collaboration
Pan-European Grid
Network infrastructure (GÉANT)
Operations, Support and training

EGEE: major themes

- e-Infrastructure
  - Integrating networks, grids and emerging technologies
  - Based on standards
  - Underpinning research, industry, … the “knowledge economy”
- International, collaborative effort
- Moving to a Service Orientated Architecture
- EGEE focus: Production grids for multiple VOs
  - Demands massive effort in organisation and administration:
    - Operations
    - Support
    - Training

User-view of EGEE: a multi-VO Grid

Enabling Grids for E-science in Europe



1997- Present: Globus

- A software toolkit addressing certain technical problems in the development of Grid enabled tools, services, and applications
  - Offers a modular “bag of technologies”
  - Made available under liberal open source license
- *Not* turnkey solutions, but *building blocks* and *tools* for application developers and system integrators

Globus: Key components

- Grid Security Infrastructure (GSI)
  - X.509 authentication with delegates and single sign-on
- Grid Resource Allocation Mgmt (GRAM)
  - Remote allocation, monitoring of job, control of compute resources
- GridFTP protocol (FTP extensions)
  - High-performance data access & transport
- Grid Resource Information Service (GRIS) + Monitoring and Discovery Service (MDS)
  - Access to structure & state information
- Others…

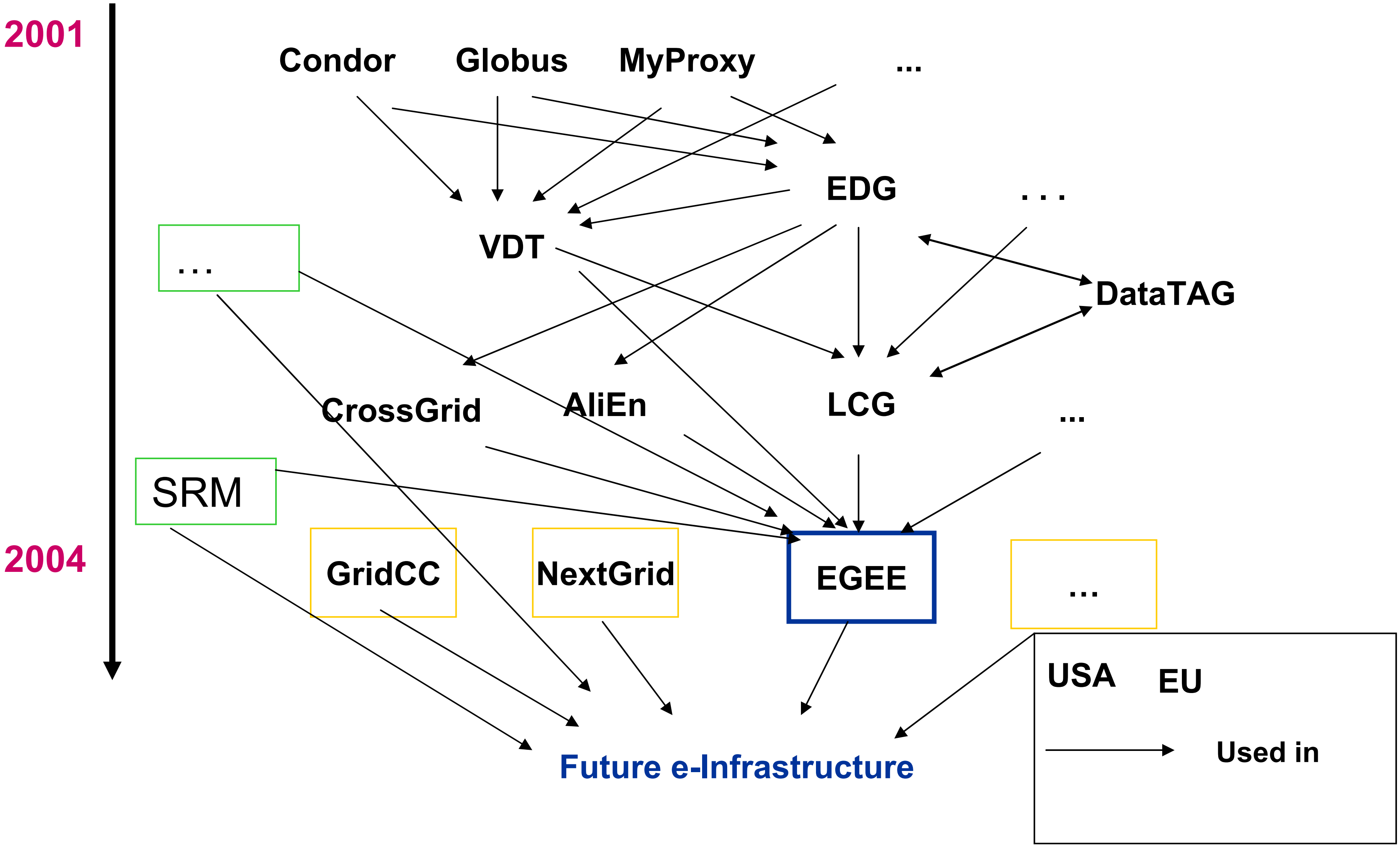
VDT

- “The Virtual Data Toolkit (VDT) is an ensemble of grid middleware that can be easily installed and configured. In our experience, installing grid software is challenging and time consuming. The goal of the VDT is to make it as easy as possible for users to deploy, maintain and use grid middleware.” http://www.cs.wisc.edu/vdt/

Virtual Data Toolkit

- http://www.cs.wisc.edu/vdt/
- Condor Group
  - Condor/Condor-G
  - DAGMan
  - Fault Tolerant Shell
  - ClassAds
- Globus Alliance
  - Job submission (GRAM)
  - Information service (MDS)
  - Data transfer (GridFTP)
  - Replica Location (RLS)
- EDG & LCG
  - Make Gridmap
  - Certificate Revocation List Updater
  - GLUE Schema
- ISI & UC
  - Chimera & Pegasus
- NCSA
  - MyProxy
  - GSI OpenSSH
  - UberFTP
- LBL
  - PyGlobus
  - Netlogger
- Caltech
  - MonaLisa
- VDT
  - VDT System Profiler
  - Configuration software
- Others
  - KX509 (U. Mich.)

Part of the Grid "ecosystem"



2001

Condor

Globus

MyProxy

...

EDG

. . .

VDT

. . .

DataTAG

CrossGrid

AliEn

LCG

...

SRM

2004

GridCC

NextGrid

EGEE

...

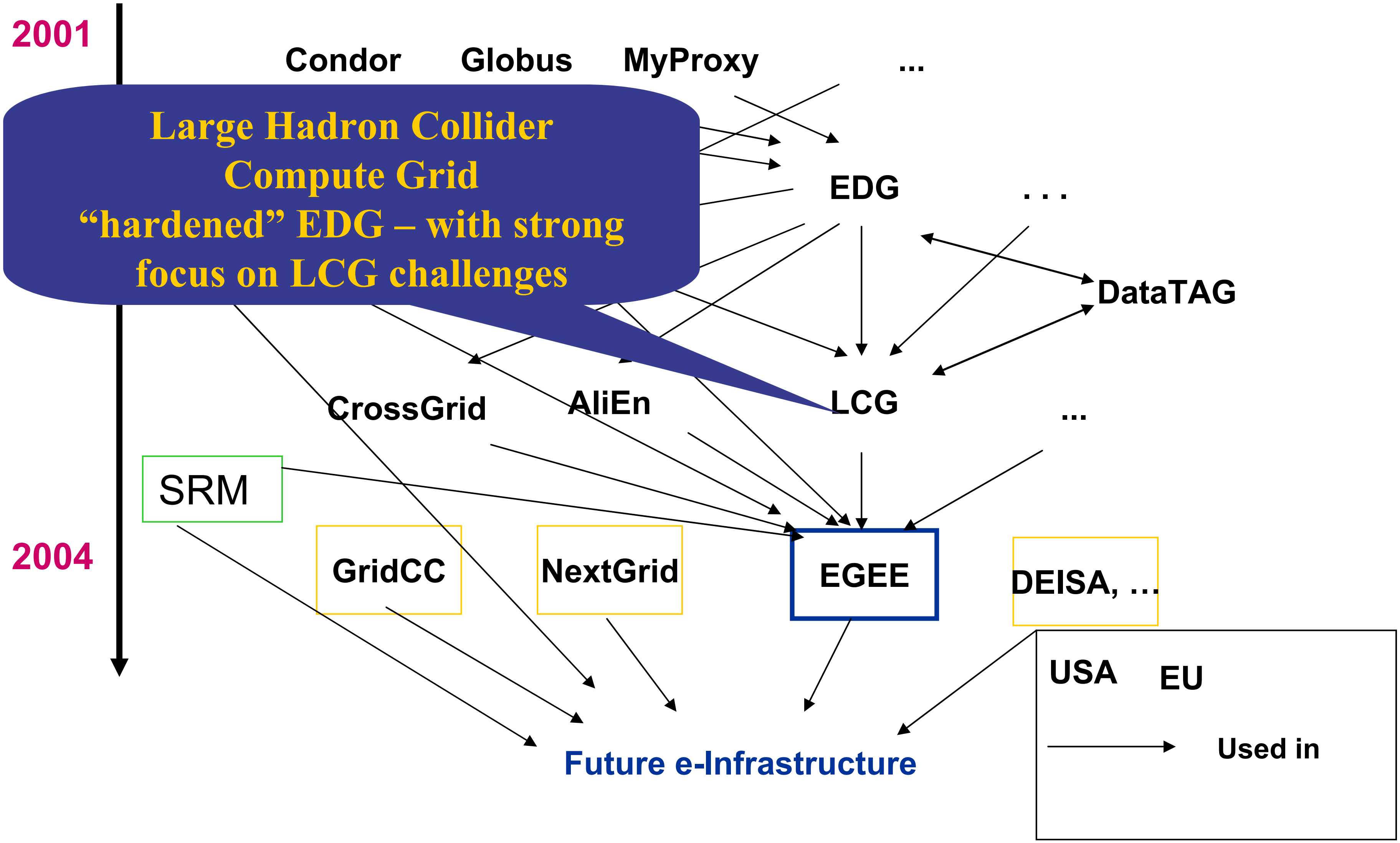
USA

EU

Used in

Future e-Infrastructure

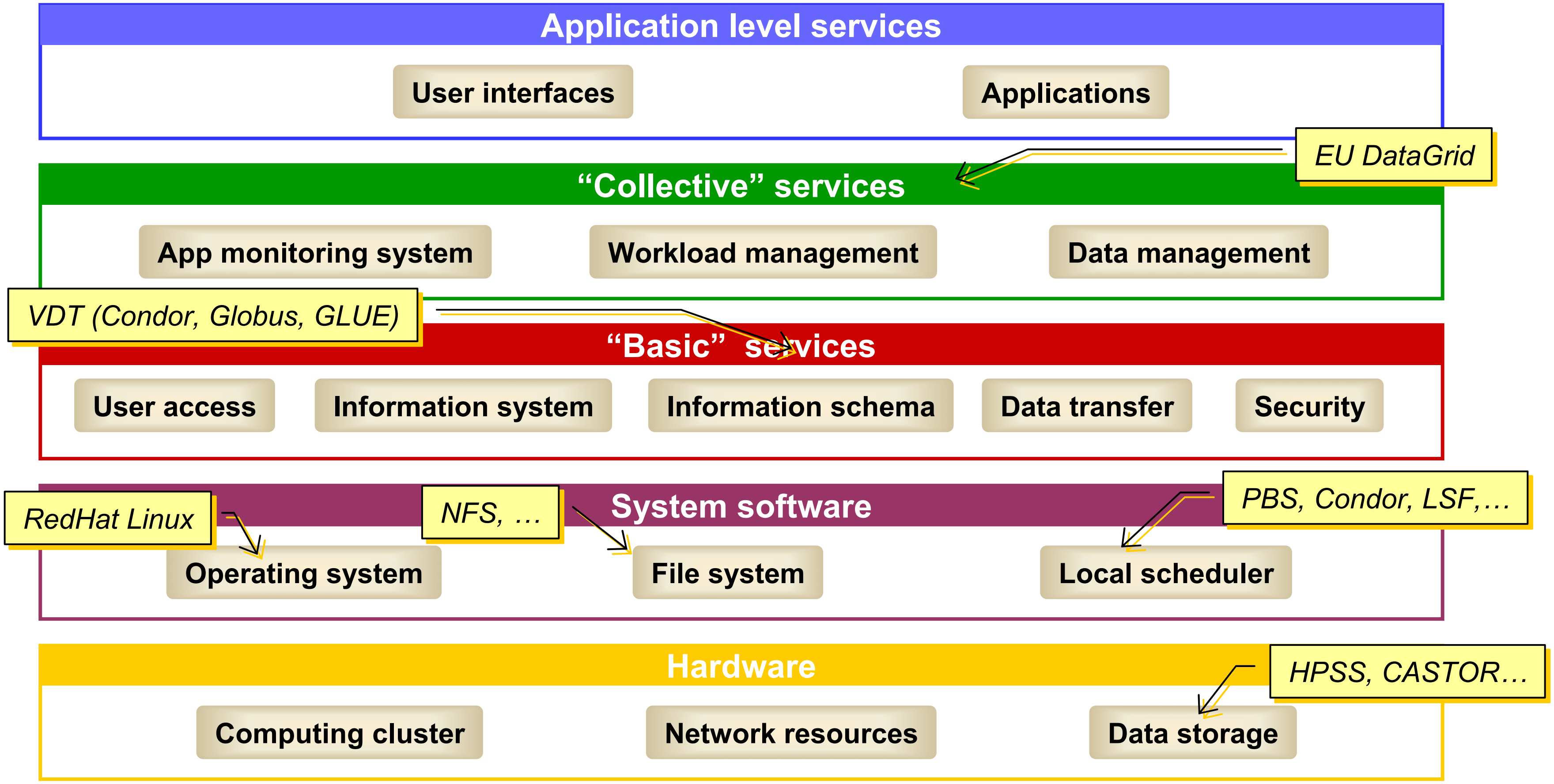
Part of the Grid "ecosystem"



Large Hadron Collider Compute Grid "hardened" EDG – with strong focus on LCG challenges

Current production mware: LCG-2

Enabling Grids for E-science in Europe



Application level services

User interfaces

Applications

EU DataGrid

“Collective” services

App monitoring system

Workload management

Data management

VDT (Condor, Globus, GLUE)

“Basic” services

User access

Information system

Information schema

Data transfer

Security

RedHat Linux

NFS, …

System software

PBS, Condor, LSF,…

Operating system

File system

Local scheduler

Hardware

HPSS, CASTOR…

Computing cluster

Network resources

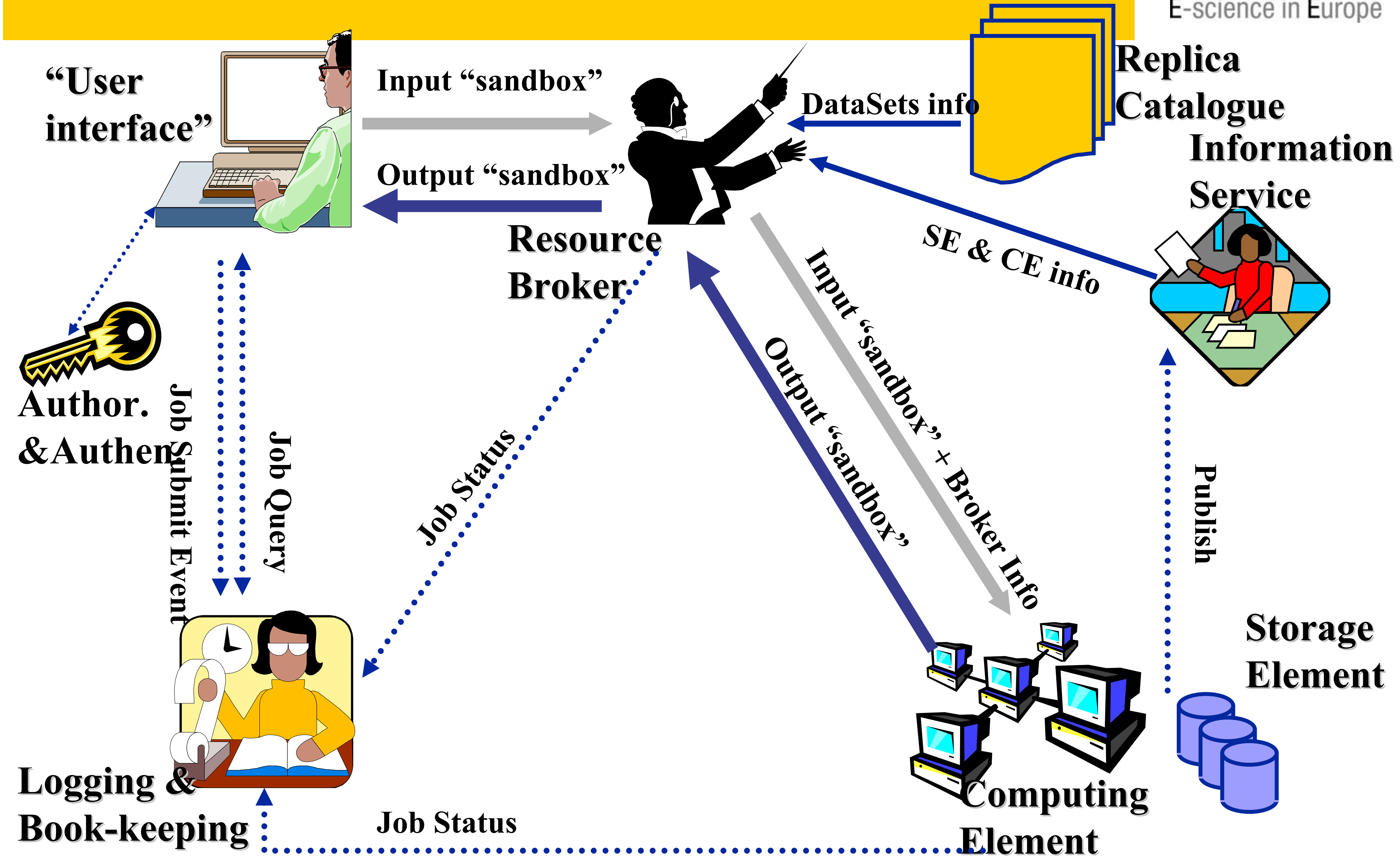
Data storage

Outline

Enabling Grids for E-science in Europe

- Overview
- **Major components**
- Data management
- Lifecycle of a job
- Summary

Major components



“User interface”

Input “sandbox”

Output “sandbox”

Resource Broker

DataSets info

Replica Catalogue

Information Service

SE & CE info

Author. &Authen.

Job Submit Event

Job Query

Job Status

Input “sandbox” + Broker Info

Output “sandbox”

Publish

Storage Element

Computing Element

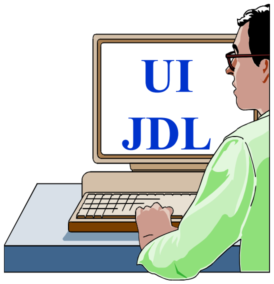
Logging & Book-keeping

Job Status

User Interface node

- The user's interface to the Grid
- Command-line interface to
  - Proxy server
  - Job operations
    - To submit a job
    - Monitor its status
    - Retrieve output
  - Data operations
    - Upload file to SE
    - Access file
    - …
  - Other grid services

- Also C++ and Java APIs



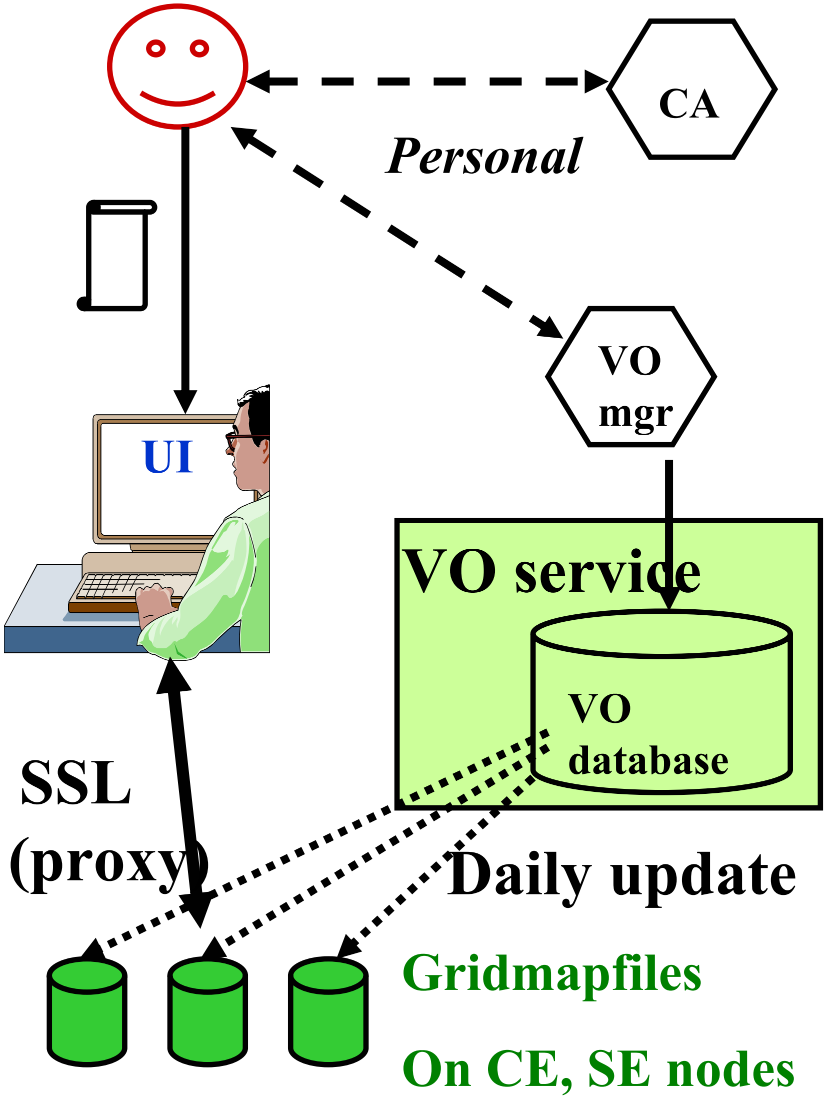
- To run a job user creates a JDL (Job Description Language) file

Authentication, Authorisation

- Authentication
  - User obtains certificate from CA
  - Connects to UI by ssh
  - Downloads certificate
  - Invokes Proxy server
  - **Single logon** – to UI - then **Secure Socket Layer with proxy identifies user to other nodes**

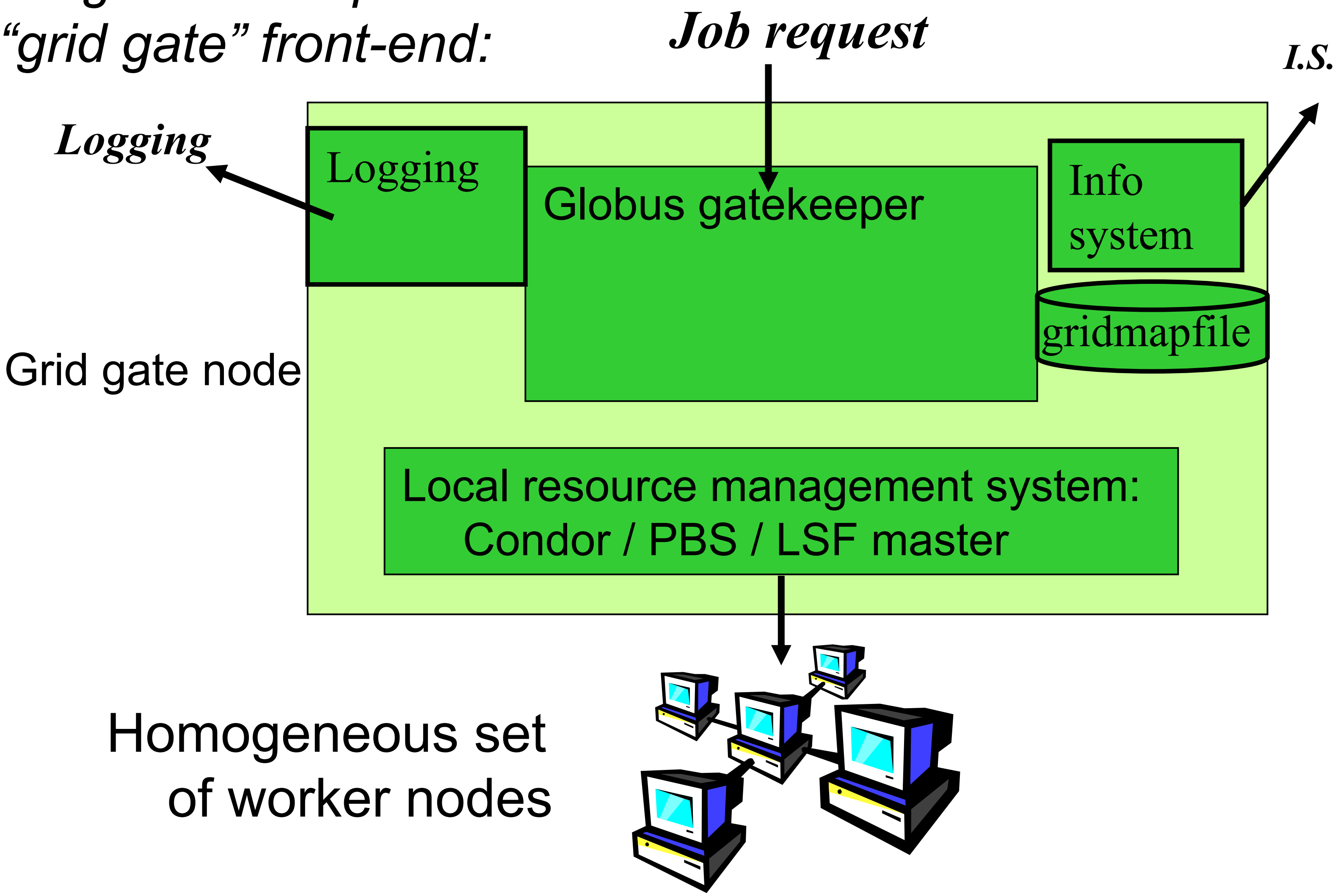
- Authorisation - currently
  - User joins Virtual Organisation
  - VO negotiates access to Grid nodes and resources (CE, SE)
  - Authorisation tested by CE, SE:

gridmapfile maps user to local account



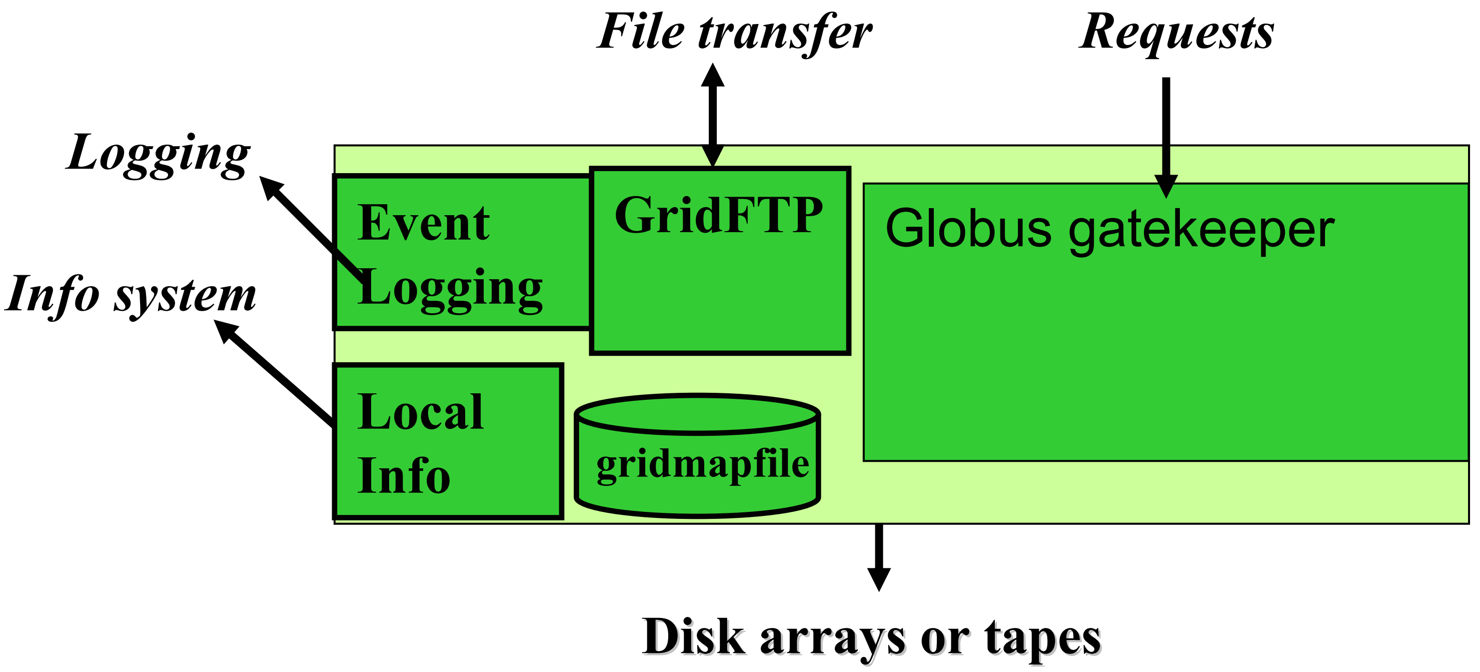
“Compute element”

A CE is a grid batch queue with a “grid gate” front-end:


Job request
I.S.
Logging
Logging
Globus gatekeeper
Info system
gridmapfile
Grid gate node
Local resource management system:
Condor / PBS / LSF master
Homogeneous set of worker nodes

Storage elements and files

- Storage elements hold files: write once, read many



Workload Management System (WMS)

- Distributed scheduling
  - multiple UI’s where you submit your job
  - multiple RB’s from where the job is sent to a CE
  - multiple CE’s where the job can be put in a queuing system
- Distributed resource management
  - multiple information systems that monitor the state of the grid
  - Information from SE, CE, sites

Resource Broker nodes

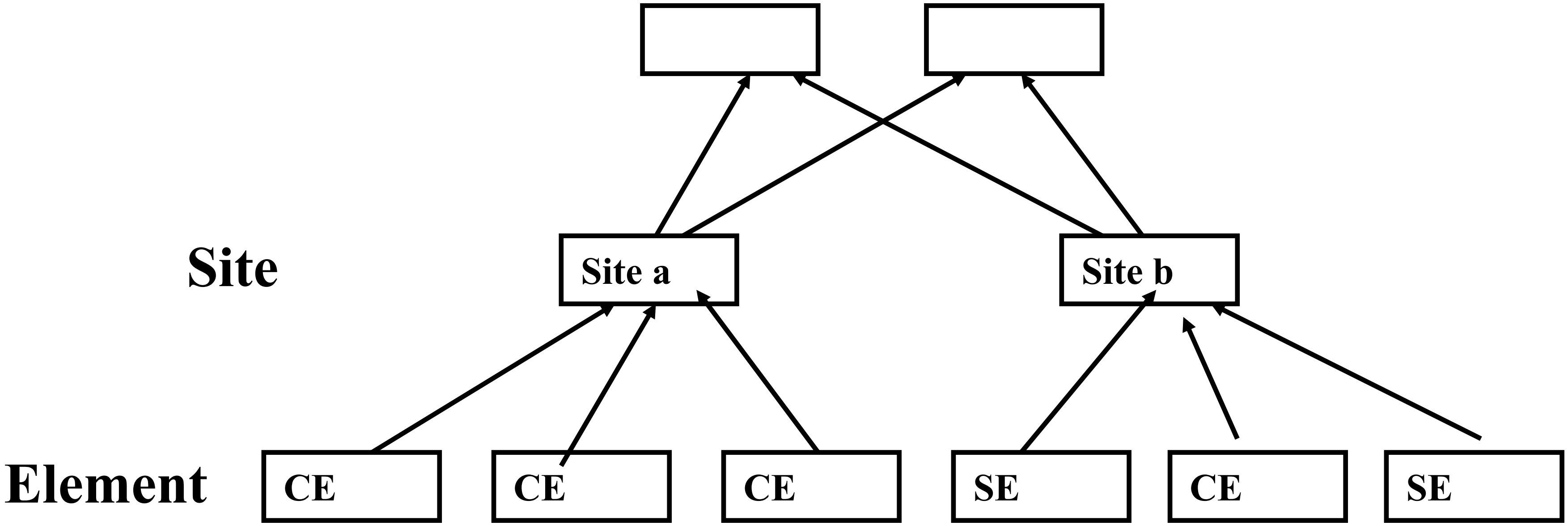
- Run the Workload Management System
  - To accept job submissions
  - Dispatch jobs to appropriate Compute Element (CE)
  - Allow users
    - To get information about their status
    - To retrieve their output
- A configuration file on each UI node determines which RB node(s) will be used
- When a user submits a job, JDL options are to:
  - Specify CE
  - Allow RB to choose CE (using optional tags to define requirements)
  - Specify SE (then RB finds “nearest” appropriate CE, after interrogating Replica Location Service)

Logging and Book-keeping

- Who did what when??
- What’s happening to my job?
- Usually runs on Resource Broker node
- See LCG-2 user guide for a bit more on this

Information System

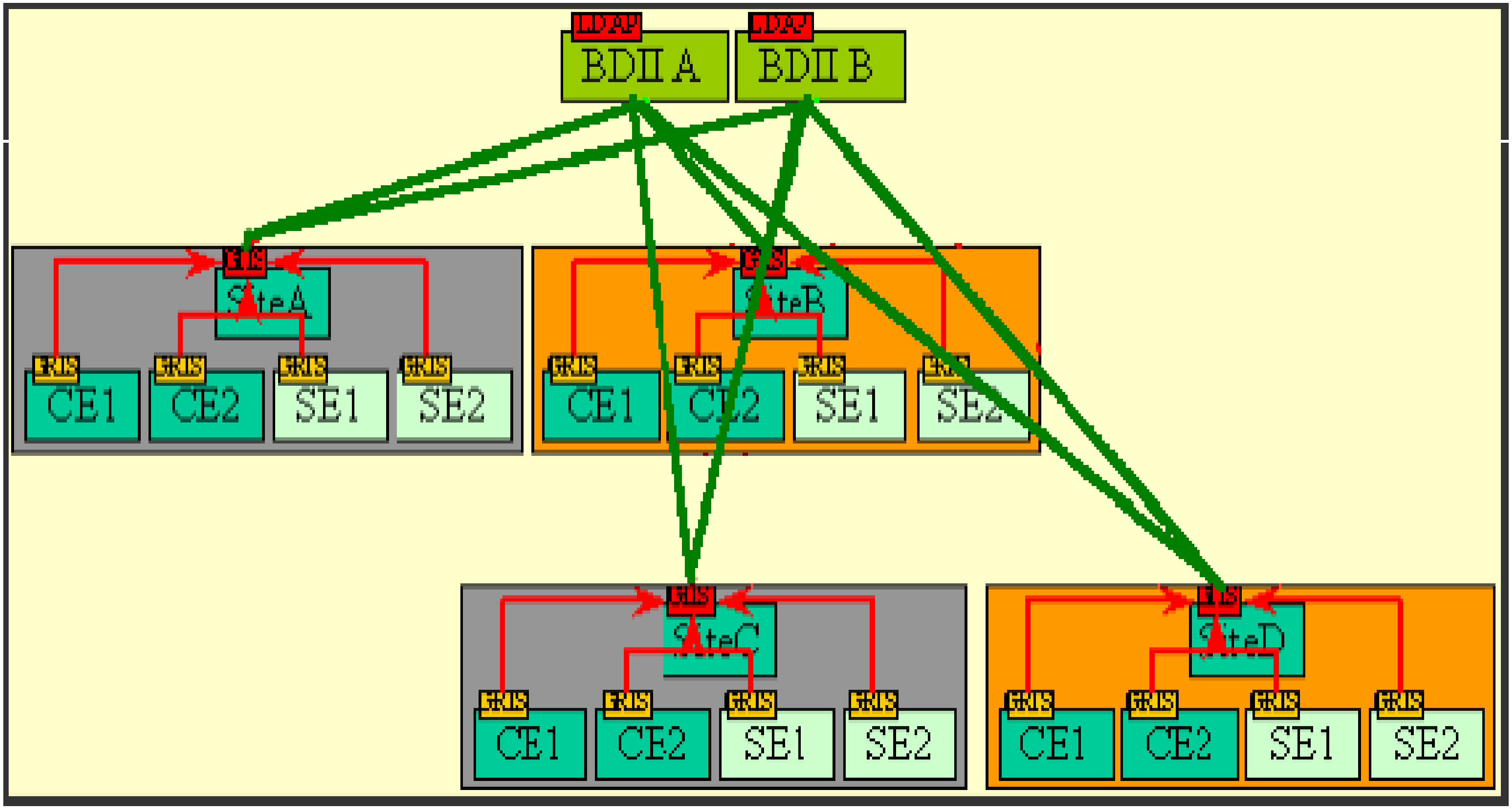
- Receives periodic (~5 minutes) updates from CE, SE
- Used by RB node to determine resources to be used by a job
- “Leaf/node” system: currently BDII is used

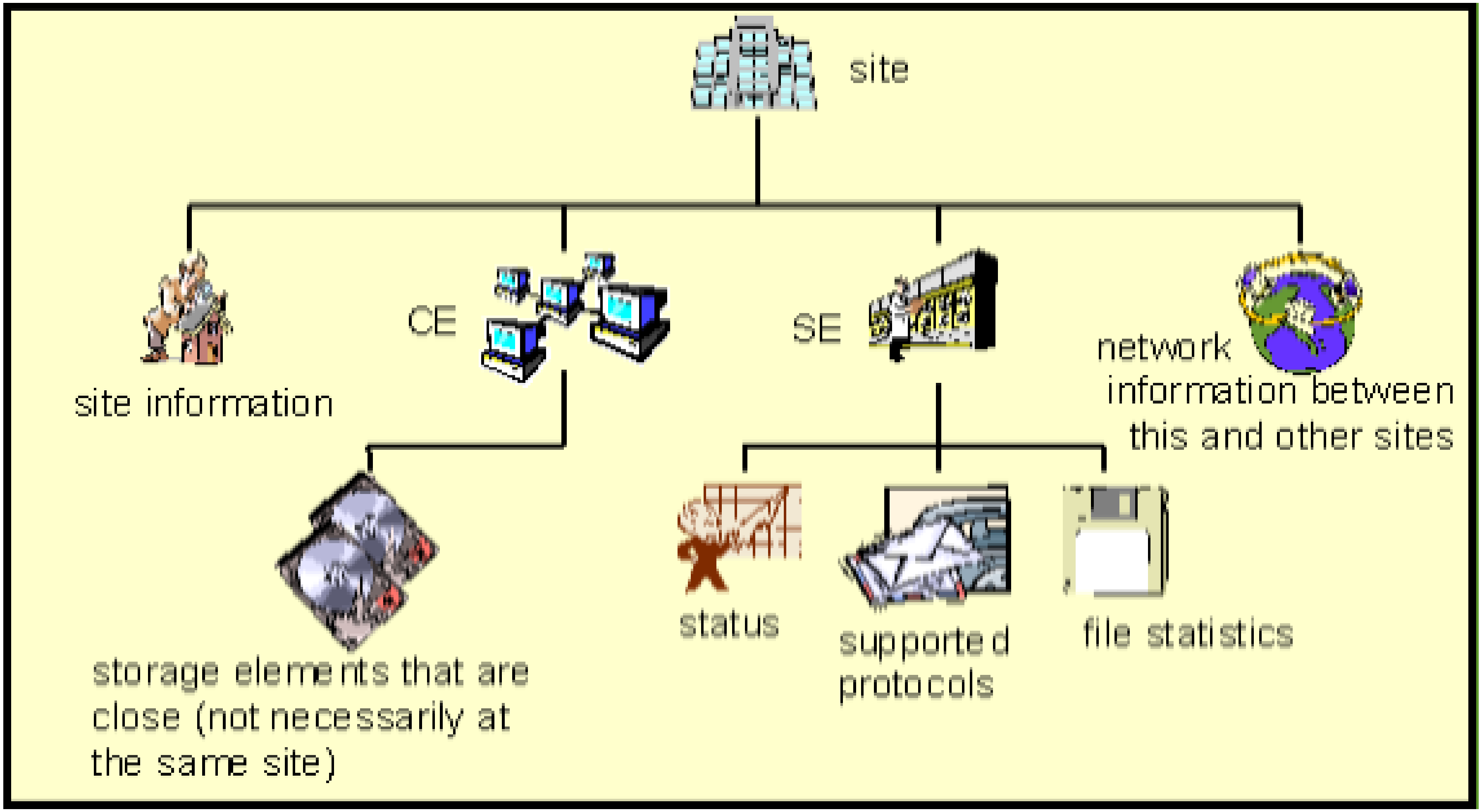


Information System

- Based on the Globus “Monitoring and Discovery Service”
- Receives periodic (~5 minutes) updates from CE, SE
- Used by RB node to determine resources to be used by a job
- Uses “GLUE schema”

Information System




site
site information
CE
SE
network information between this and other sites
storage elements that are close (not necessarily at the same site)
status
supported protocols
file statistics

Outline

- Overview
- Major components
- **Data management**
- Lifecycle of a job
- Summary

Data management

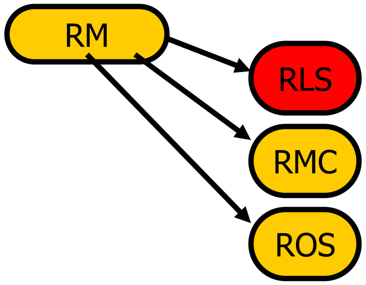
- User data: generally file-oriented (some RDBMS exceptions exist)

- Small files: On UI; passed to/from CE via sandbox

- Large files: require SE
  - Replica files on different SEs
    - Fault tolerance
    - Performance:
      - run job on CE “close” to data
      - share load on SE
  - Replica Catalogue - what replicas exist for a file?
  - Replica Location Service - where are they?

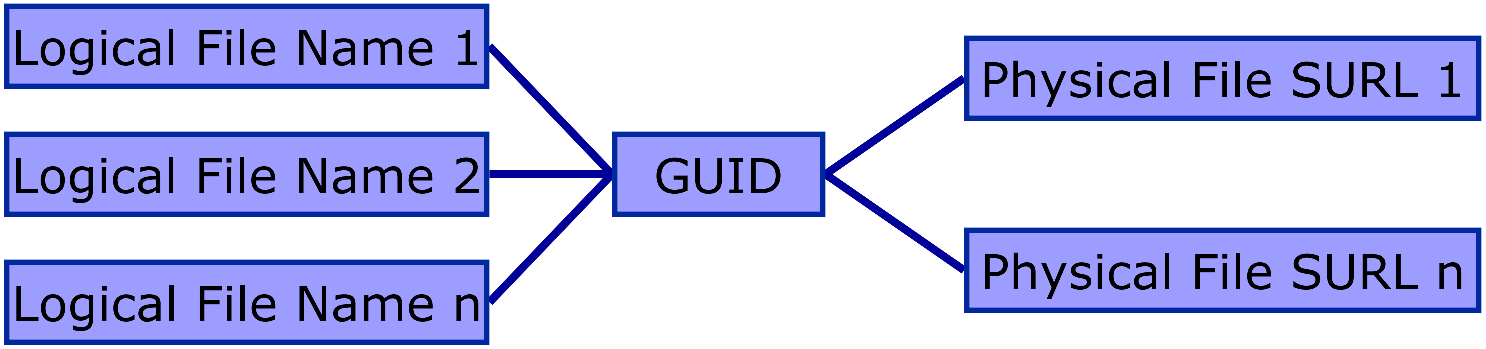
Replica Location Service (RLS)

- The Replica Location Service is a system that maintains and provides access to information about the physical location of copies of data files.
- It is a distributed service that stores mappings between globally unique identifiers of datafiles and the physical identifiers of all existing replicas of these datafiles.
- Design was a collaboration between Globus and EDG



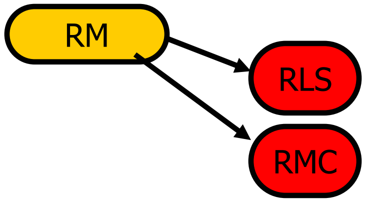
Naming Conventions

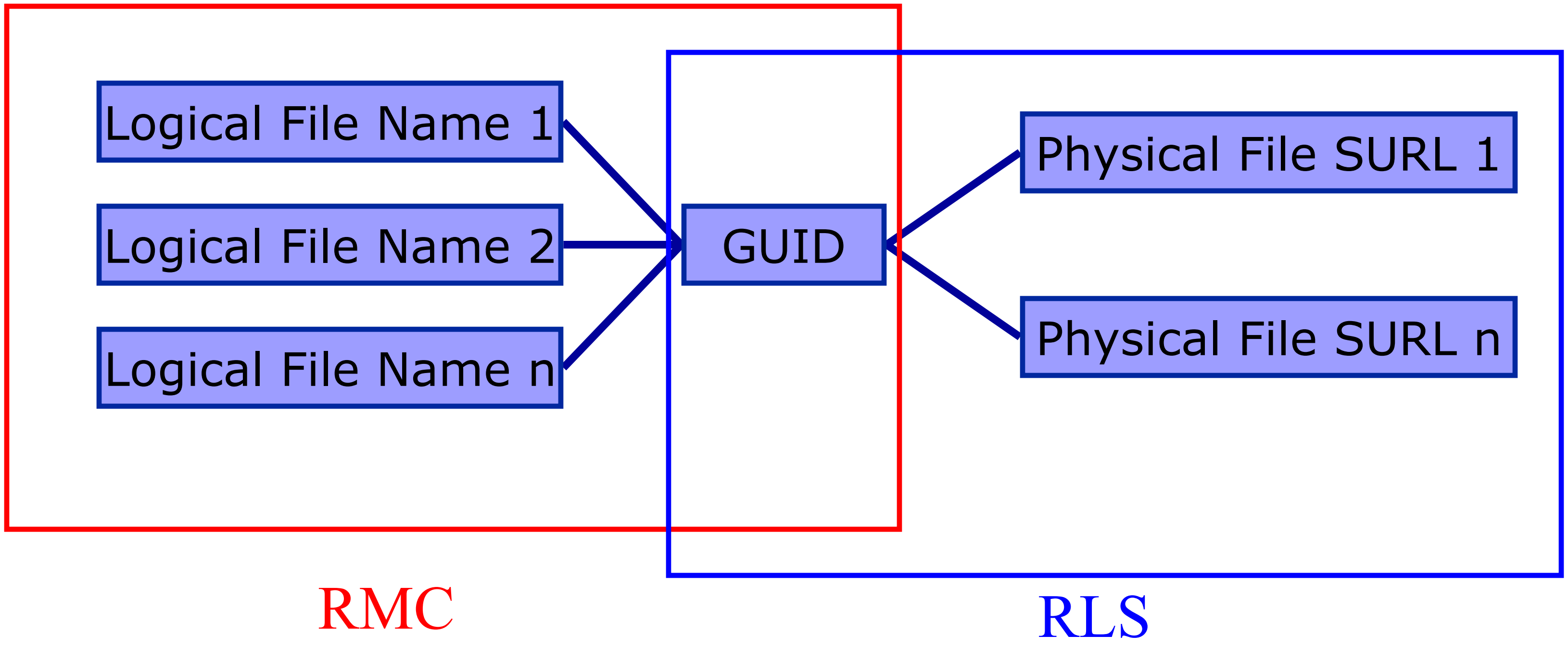
- Logical File Name (LFN)
  - An alias created by a user to refer to some item of data e.g. “lfn:cms/20030203/run2/track1”
- Site URL (SURL) (or Physical File Name (PFN))
  - The location of an actual piece of data on a storage system e.g. “srm://pcrd24.cern.ch/flatfiles/cms/output10_1”
- Globally Unique Identifier (GUID)
  - A non-human readable unique identifier for an item of data e.g. “guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6”



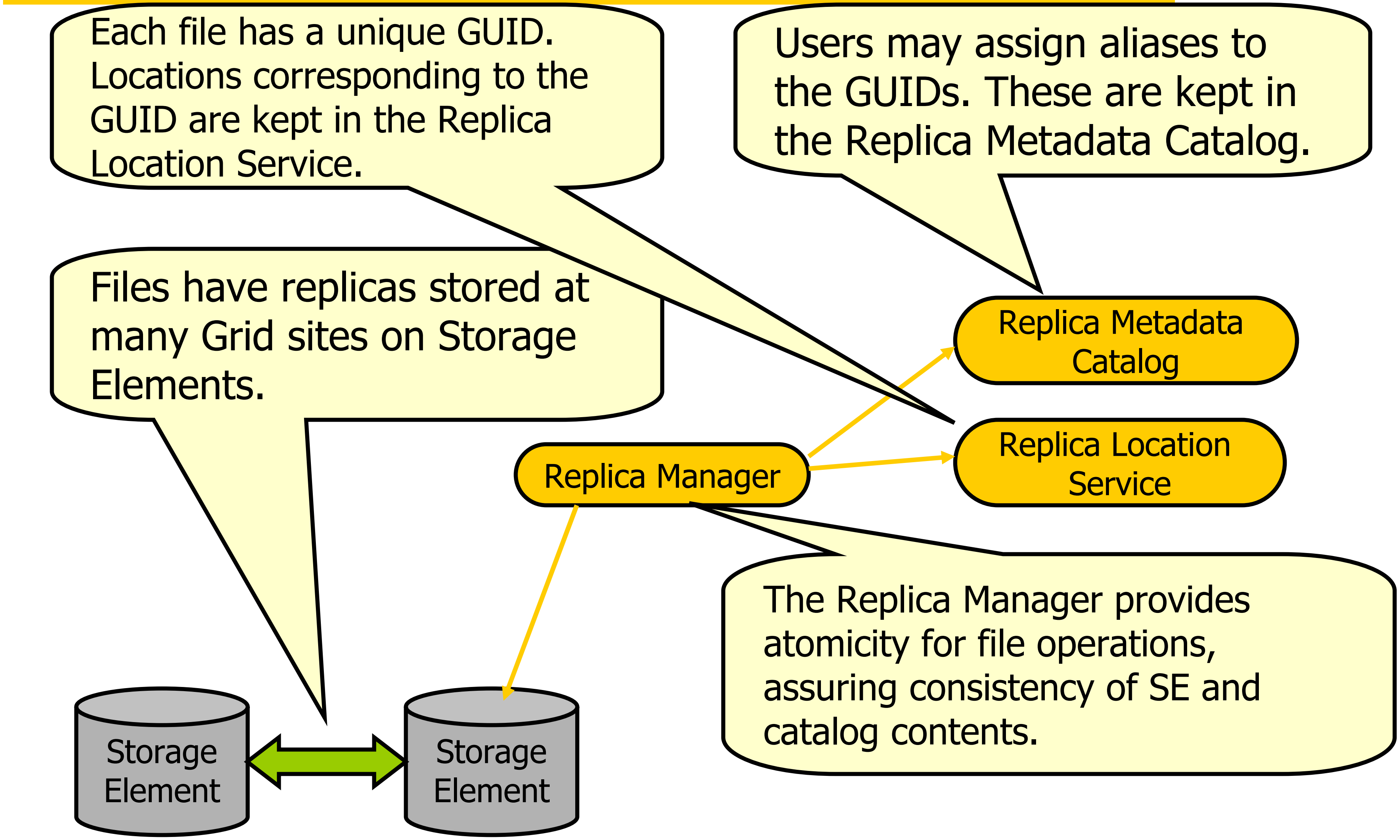
Replica Metadata Catalog (RMC)
Replica Location Service (RLS)

- RMC:
  - Stores LFN-GUID mappings
- RLS:
  - Stores GUID-SURL mappings





Data Replication Services: Basic Functionality



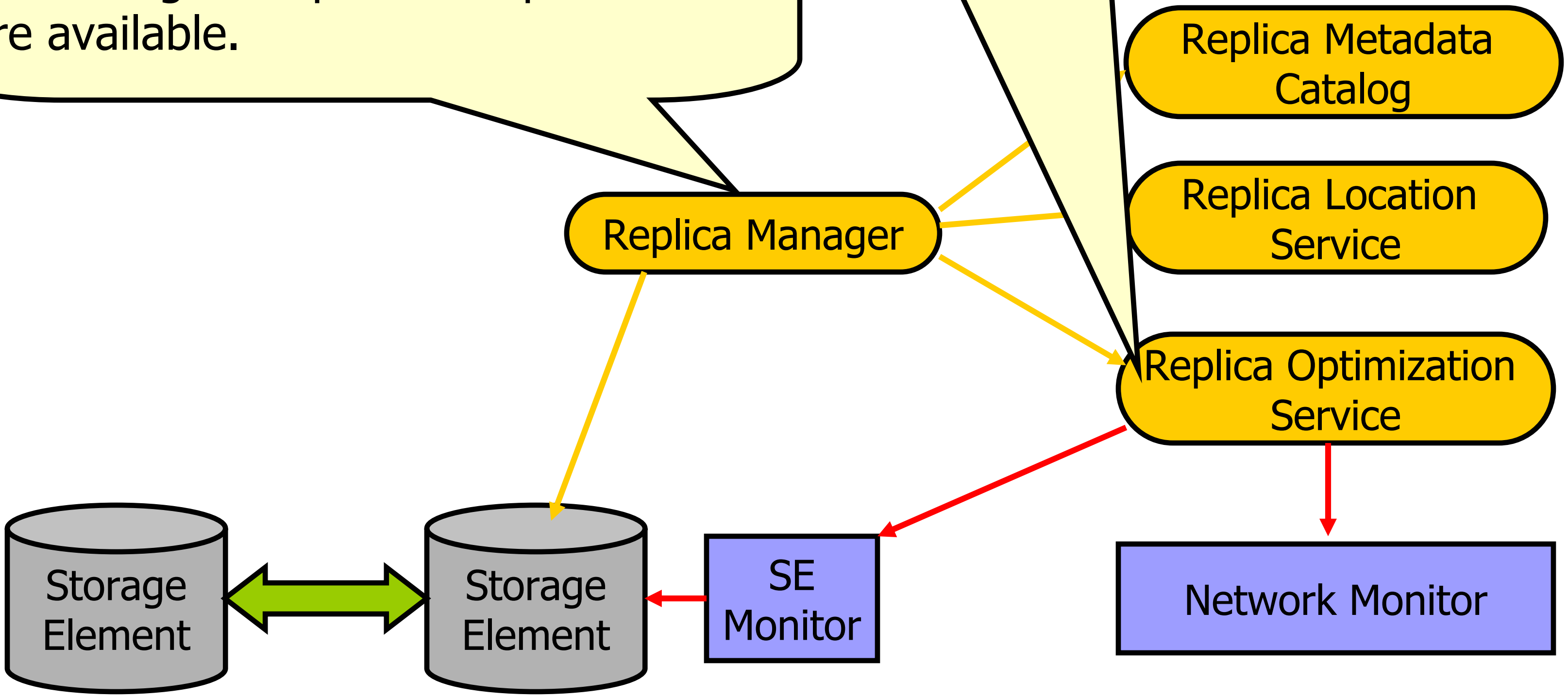
Each file has a unique GUID. Locations corresponding to the GUID are kept in the Replica Location Service.

Users may assign aliases to the GUIDs. These are kept in the Replica Metadata Catalog.

Files have replicas stored at many Grid sites on Storage Elements.

The Replica Manager provides atomicity for file operations, assuring consistency of SE and catalog contents.

Higher Level Replication Services

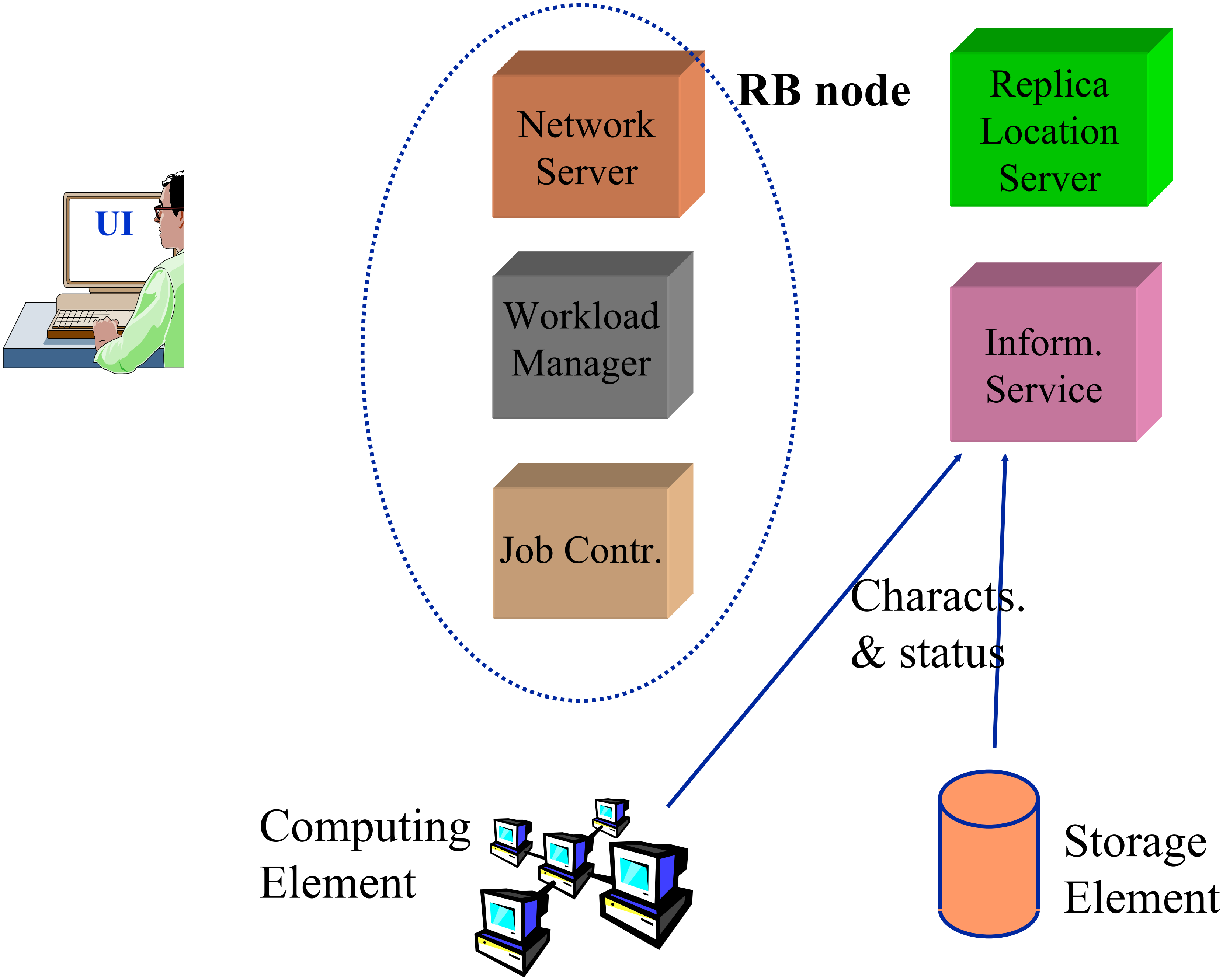


The Replica Manager may call on the Replica Optimization service to find the best replica among many based on network and SE monitoring.

Hooks for user-defined pre- and post-processing for replication operations are available.

Outline

- Overview
- Major components
- Data management
- **Lifecycle of a job**
- Summary



UI

Network Server

Workload Manager

Job Contr.

RB node

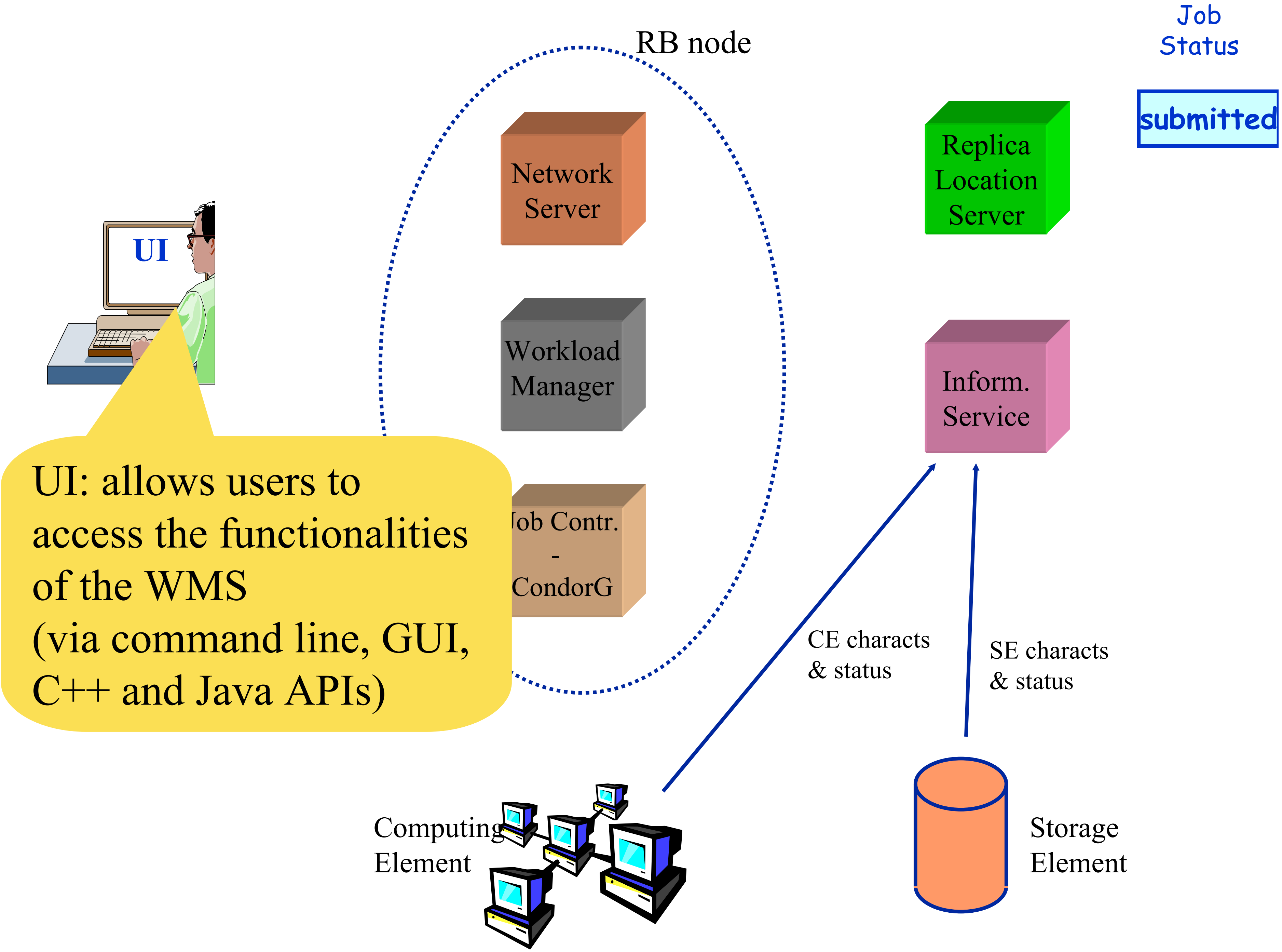
Replica Location Server

Inform. Service

Characts. & status

Computing Element

Storage Element



UI: allows users to access the functionalities of the WMS
(via command line, GUI, C++ and Java APIs)

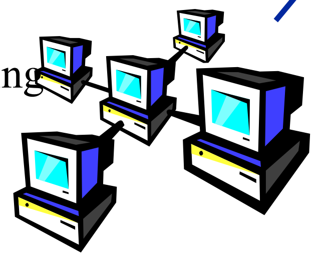
Job Statu s

submitted

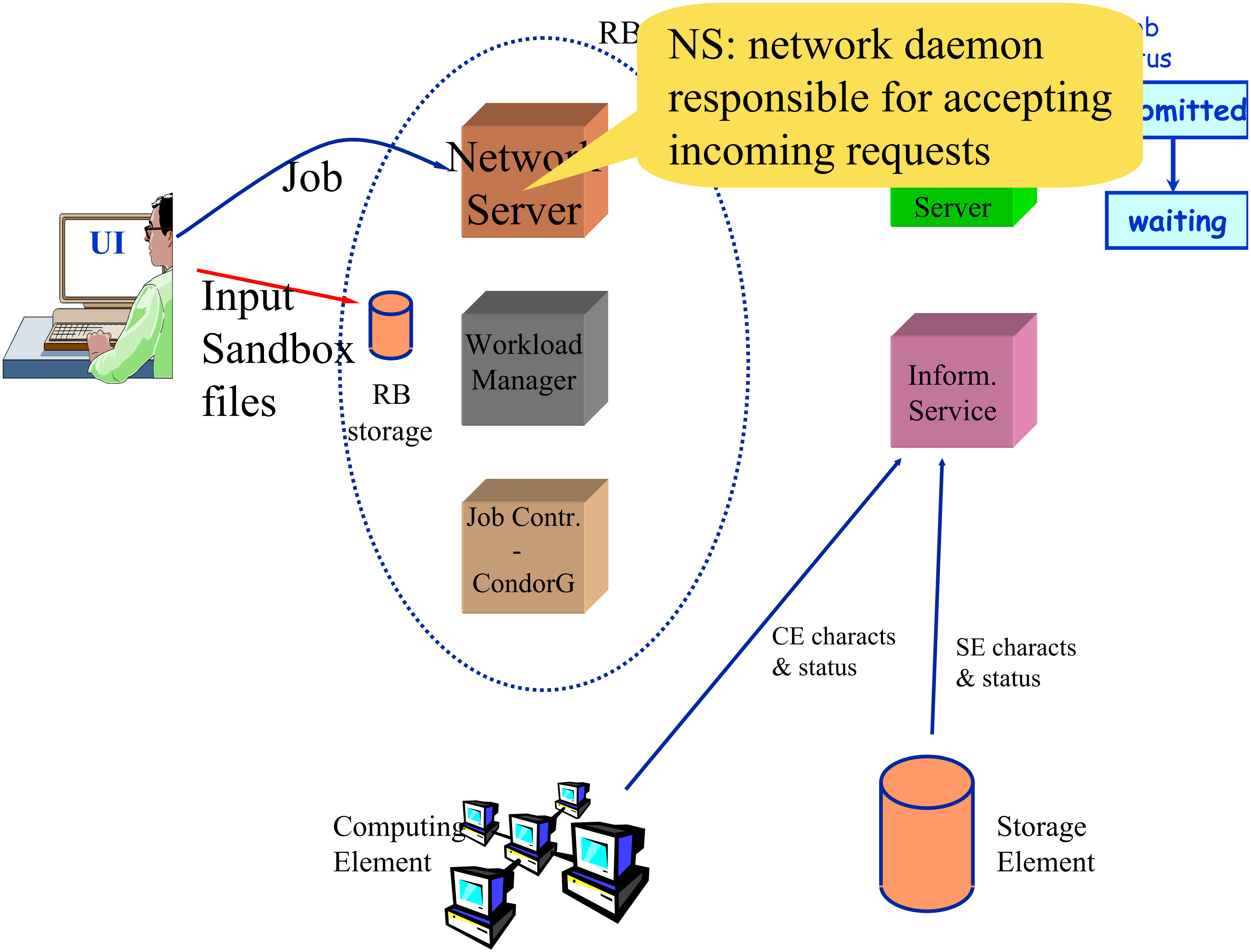
```
edg-job-submit myjob.jdl
    Myjob.jdl
        JobType = “Normal”;
        Executable      = "$(CMS)/exe/sum.exe";
        InputSandbox = {"/home/user/WP1testC","/home/file*”,
        "/home/user/DATA/*"};
        OutputSandbox = {“sim.err”, “test.out”, “sim.log"};
        Requirements    = other. GlueHostOperatingSystemName ==
        “linux" &&
        other. GlueHostOperatingSystemRelease == "Red Hat 7.3"
        && other.GlueCEPolicyMaxCPUTime > 10000;
        Rank  = other.GlueCEStateFreeCPUs;
```

haracts

SE characts & status

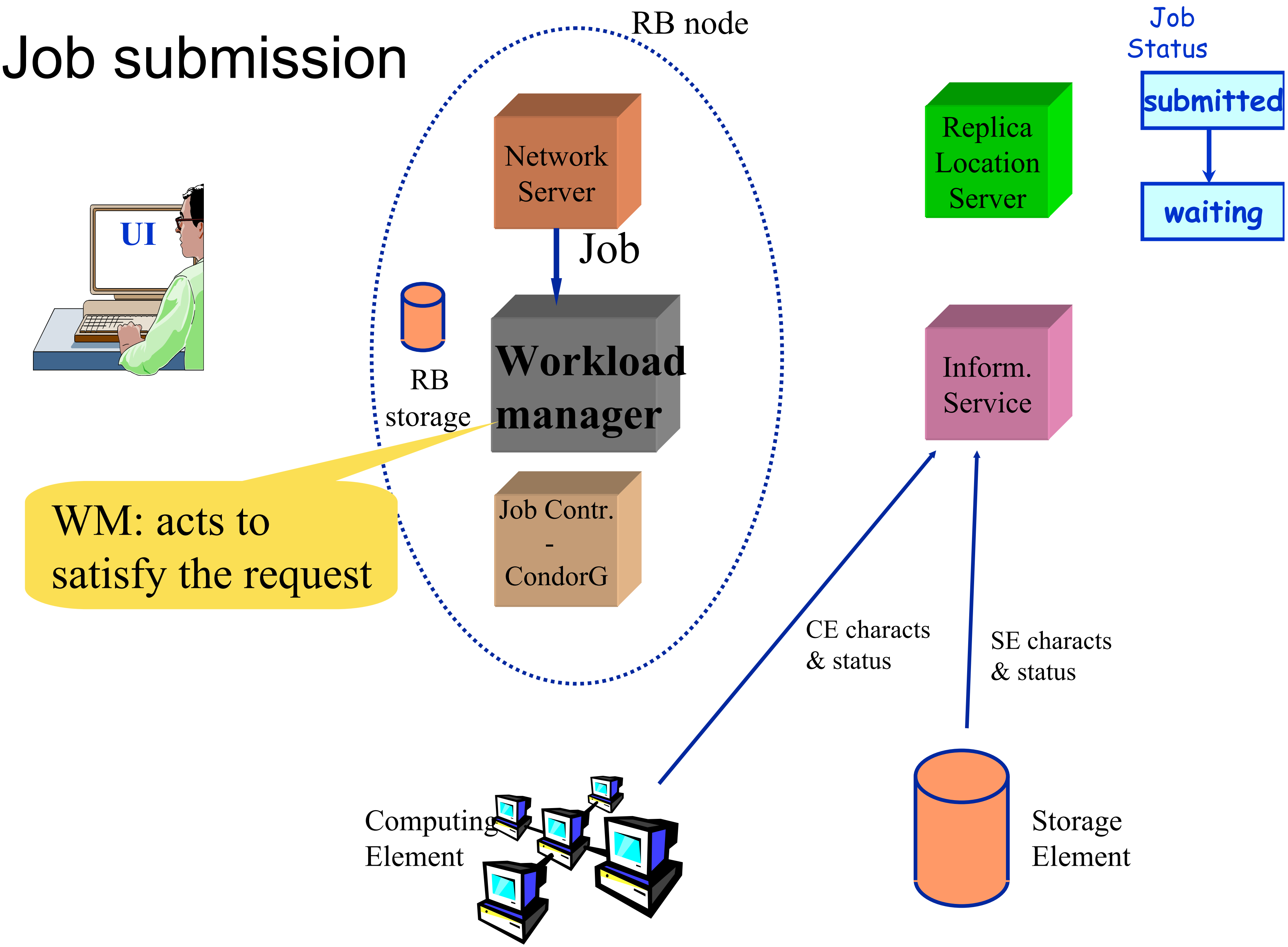


Job Description Language (JDL) to specify job characteristics and requirements



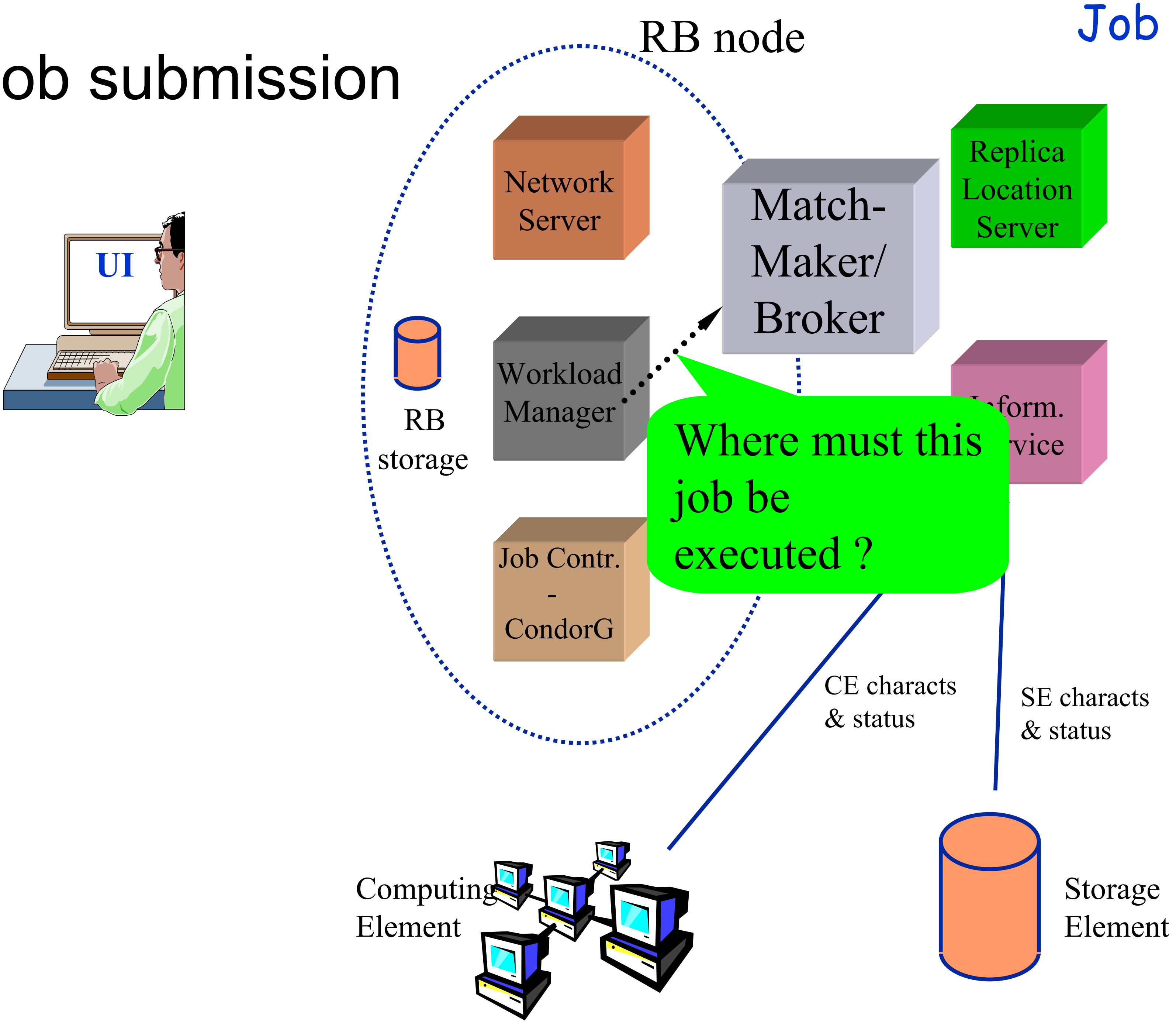
NS: network daemon responsible for accepting incoming requests

Job submission



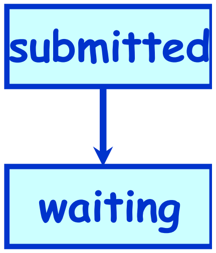
WM: acts to satisfy the request

Job submission

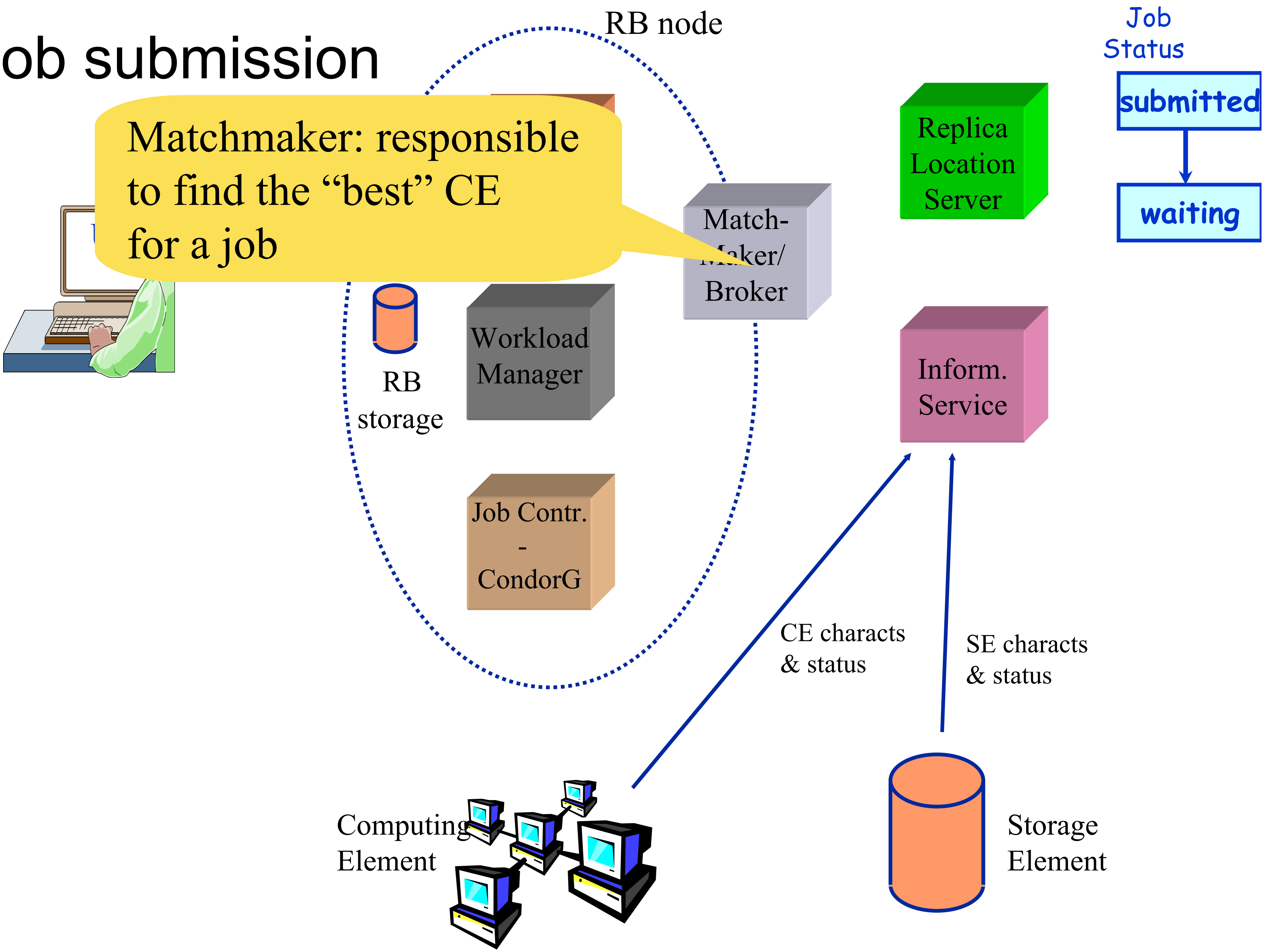


Where must this job be executed ?

Job Status



Job submission



Matchmaker: responsible to find the “best” CE for a job

RB node

RB storage

Workload Manager

Match- Maker/ Broker

Job Contr. - CondorG

Replica Location Server

Inform. Service

CE characts & status

SE characts & status

Computing Element

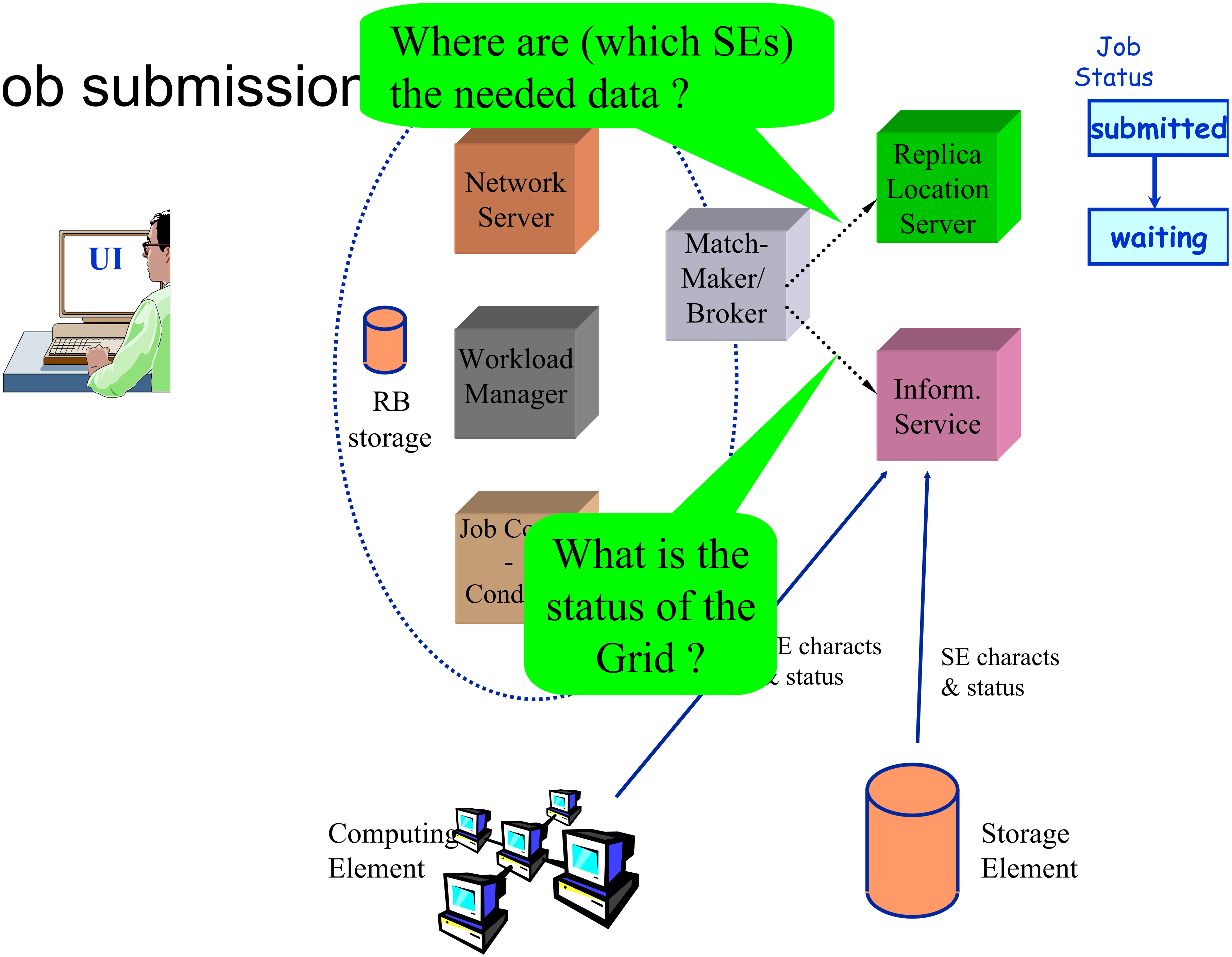
Storage Element

Job Status

submitted

waiting

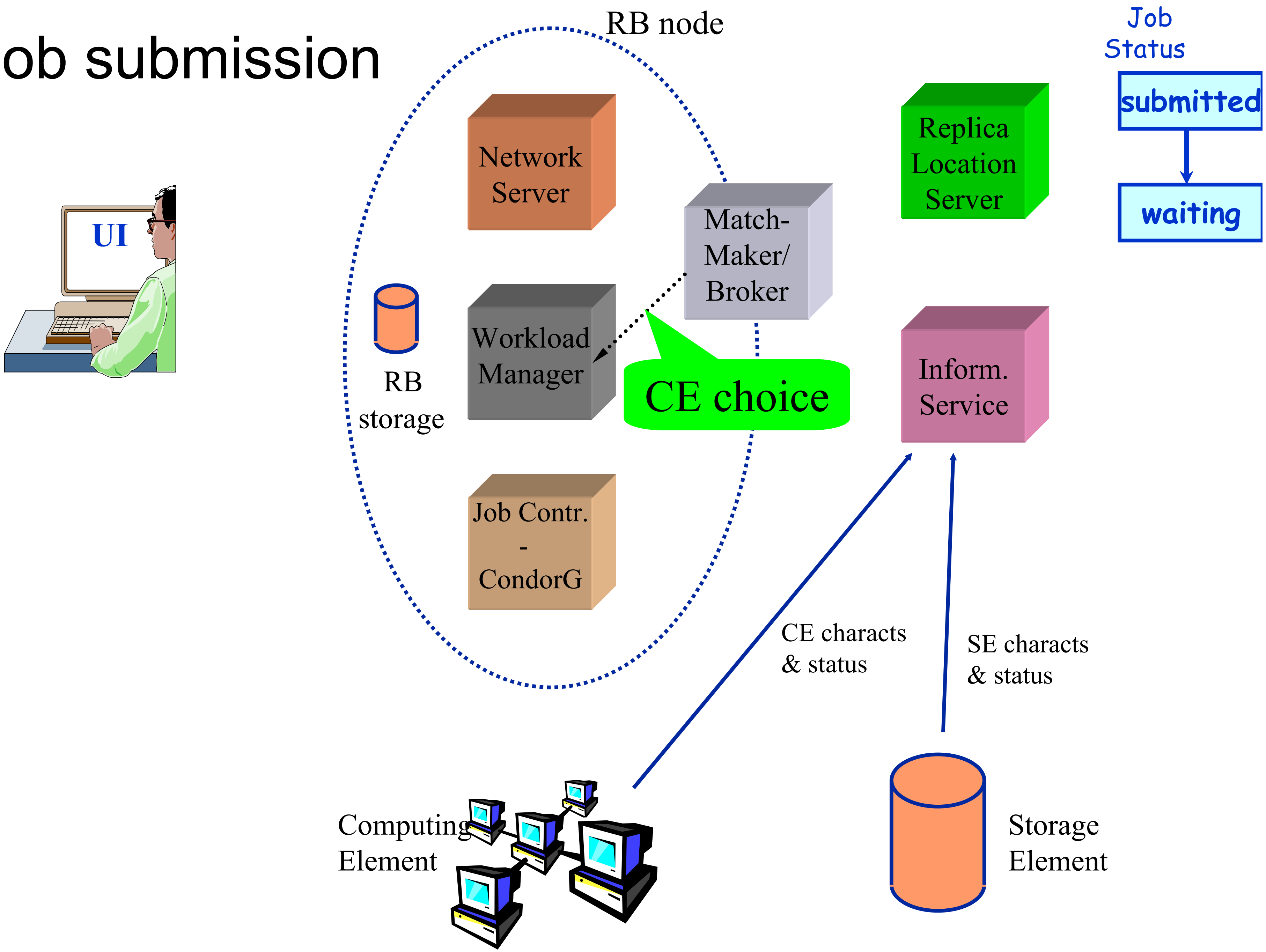
Job submission



Where are (which SEs) the needed data ?

What is the status of the Grid ?

Job submission



UI

RB node

Network Server

Workload Manager

Job Contr. - CondorG

RB storage

Match-Maker/ Broker

CE choice

Replica Location Server

Inform. Service

CE characts & status

SE characts & status

Computing Element

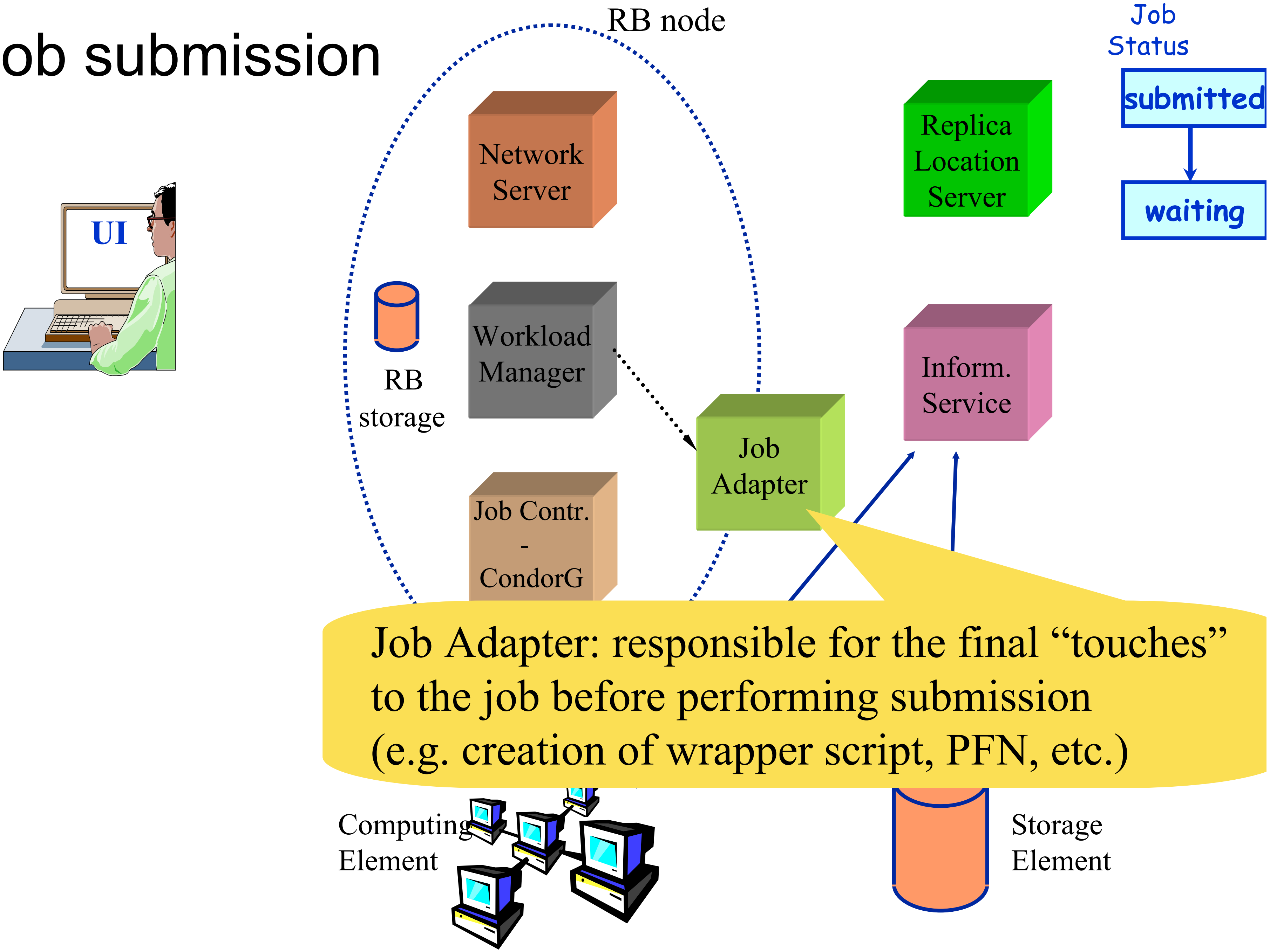
Storage Element

Job Status

submitted

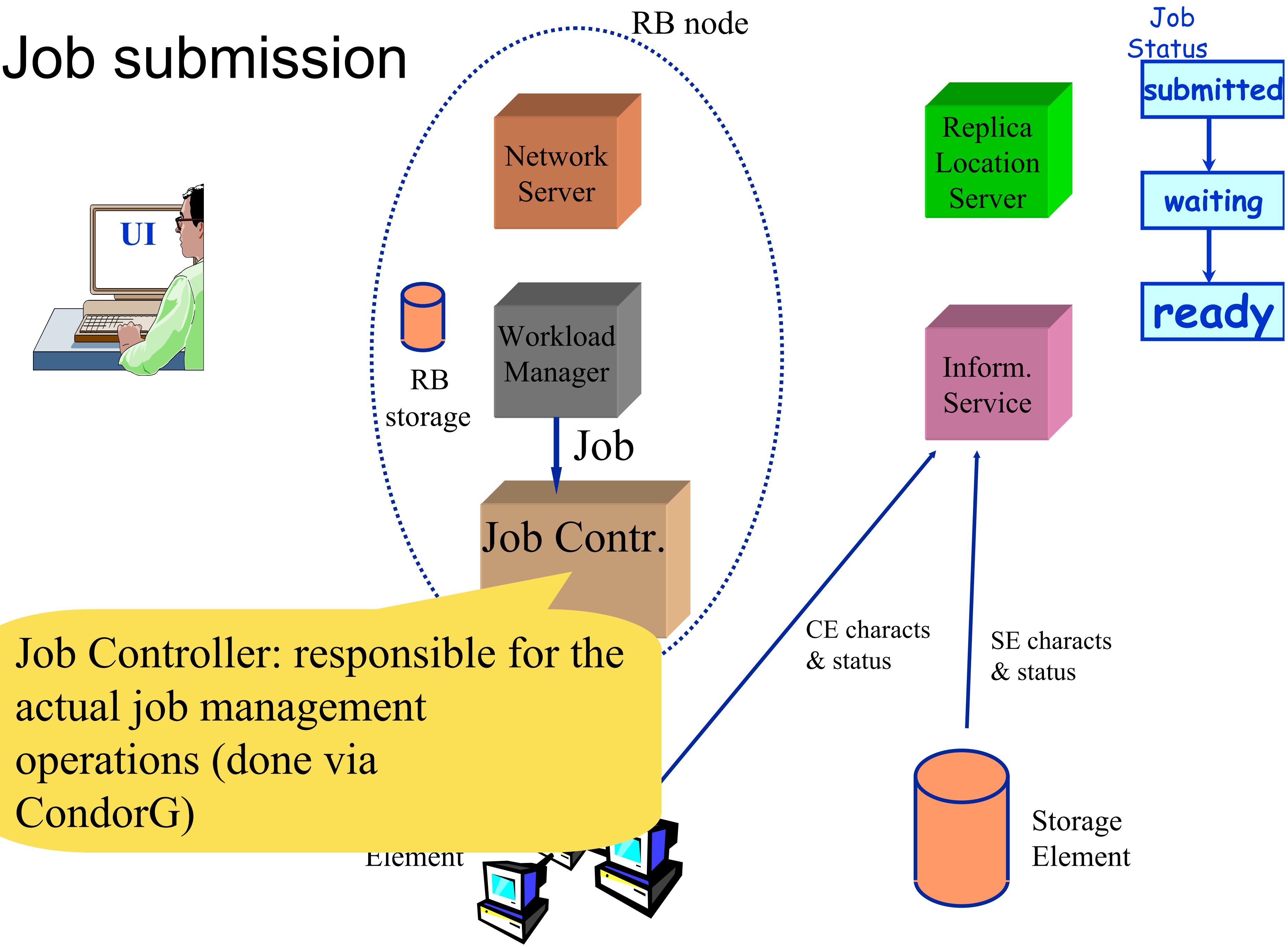
waiting

Job submission



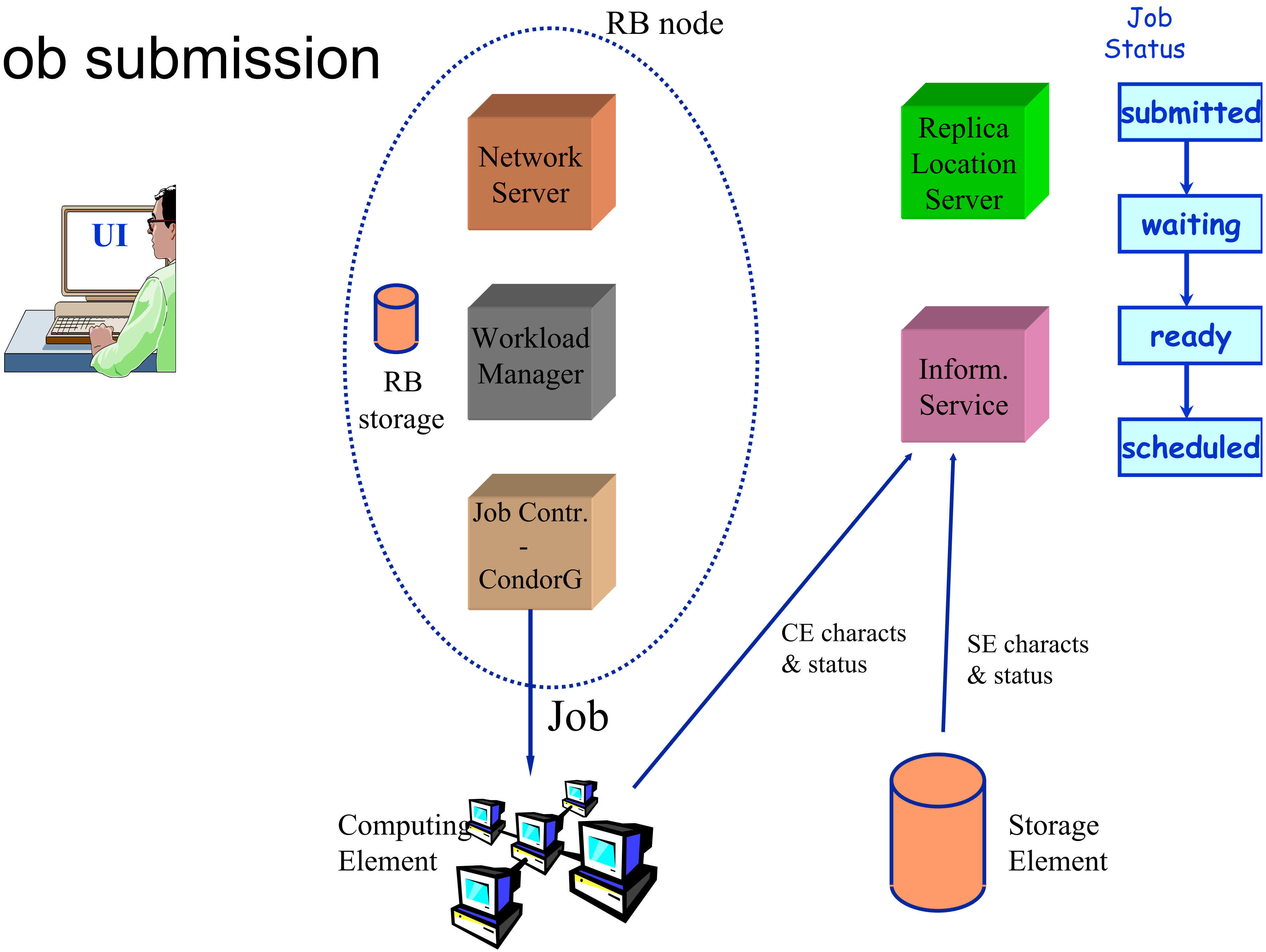
Job Adapter: responsible for the final “touches” to the job before performing submission (e.g. creation of wrapper script, PFN, etc.)

Job submission

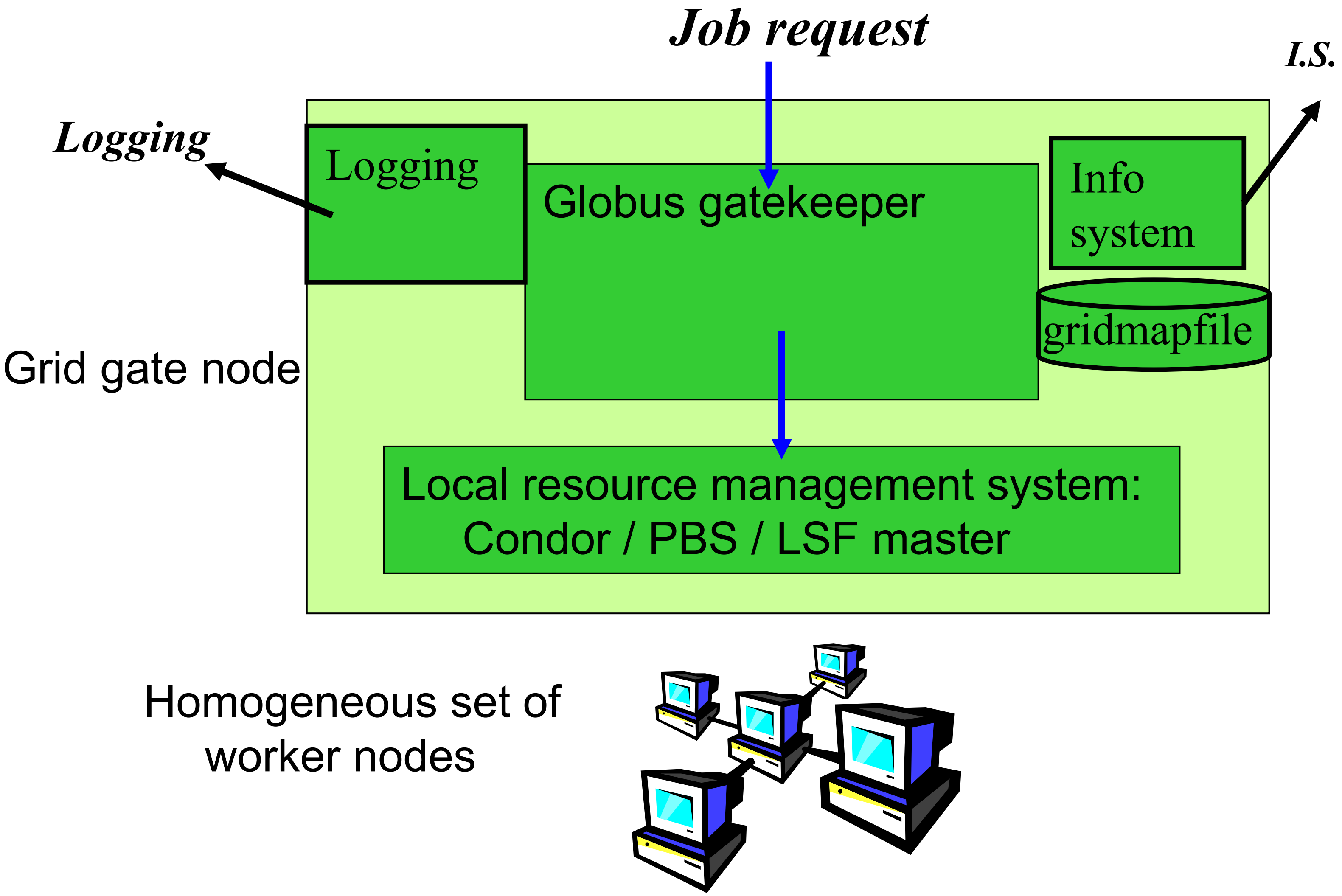


Job Controller: responsible for the actual job management operations (done via CondorG)

Job submission



“Compute element” – reminder!



Job request

I.S.

Logging

Logging

Globus gatekeeper

Info system

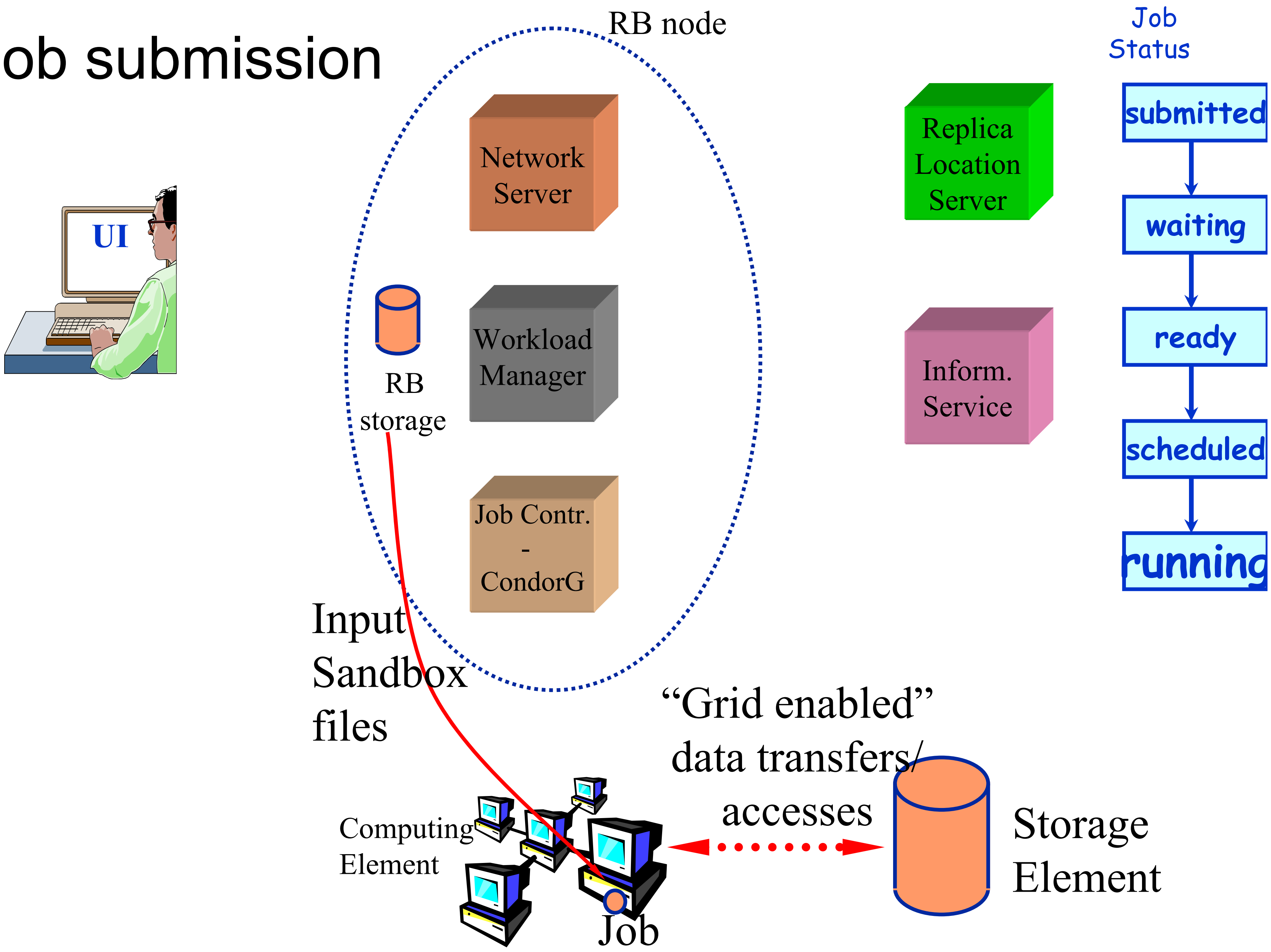
gridmapfile

Grid gate node

Local resource management system: Condor / PBS / LSF master

Homogeneous set of worker nodes

Job submission



UI

RB node

Network Server

Workload Manager

Job Contr. - CondorG

RB storage

Replica Location Server

Inform. Service

Job Status

submitted

waiting

ready

scheduled

running

Input Sandbox files

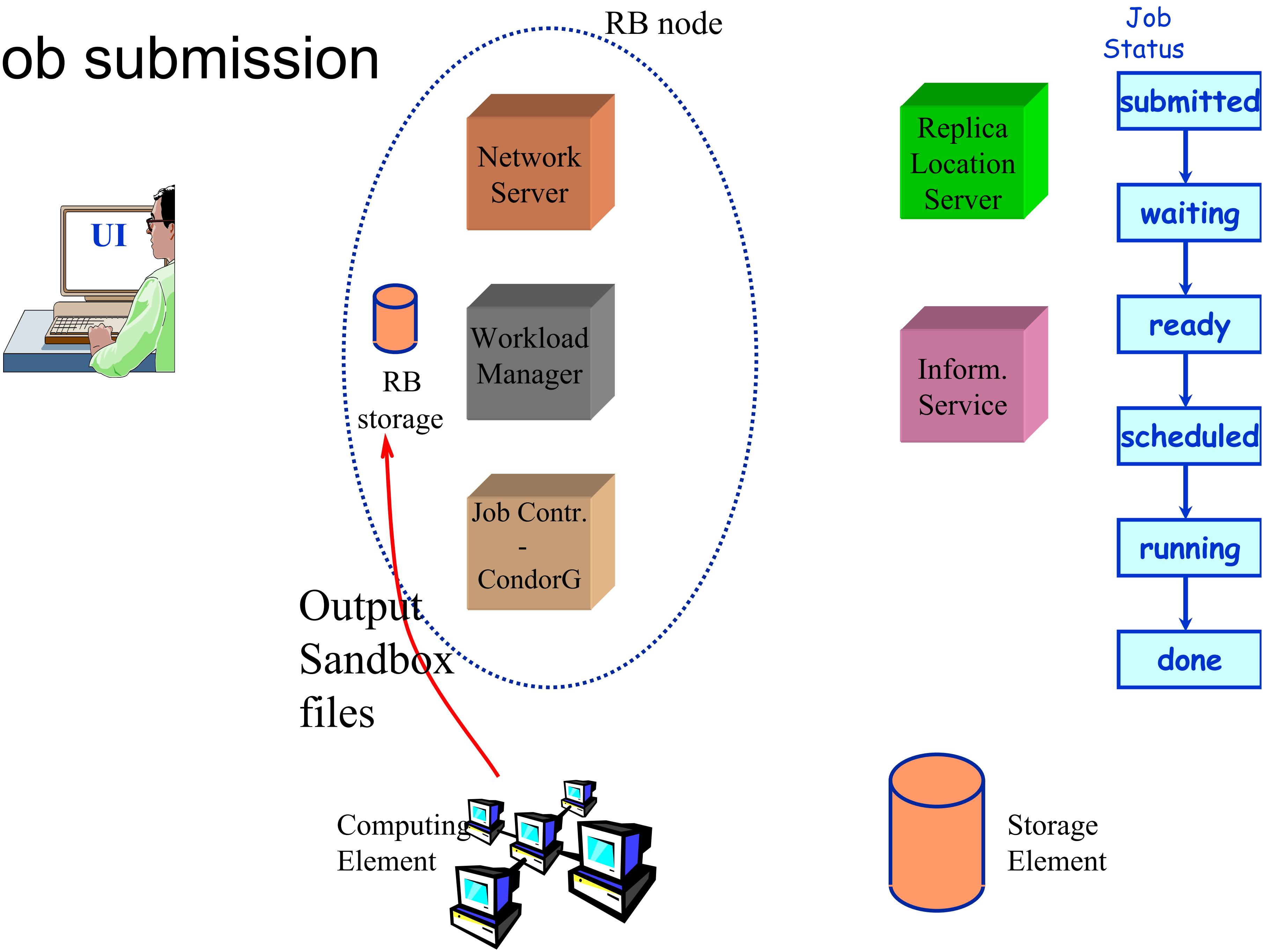
"Grid enabled" data transfers/ accesses

Computing Element

Job

Storage Element

Job submission



RB node

UI

Network Server

Workload Manager

Job Contr. - CondorG

RB storage

Output Sandbox files

Replica Location Server

Inform. Service

Computing Element

Storage Element

Job Status

submitted

waiting

ready

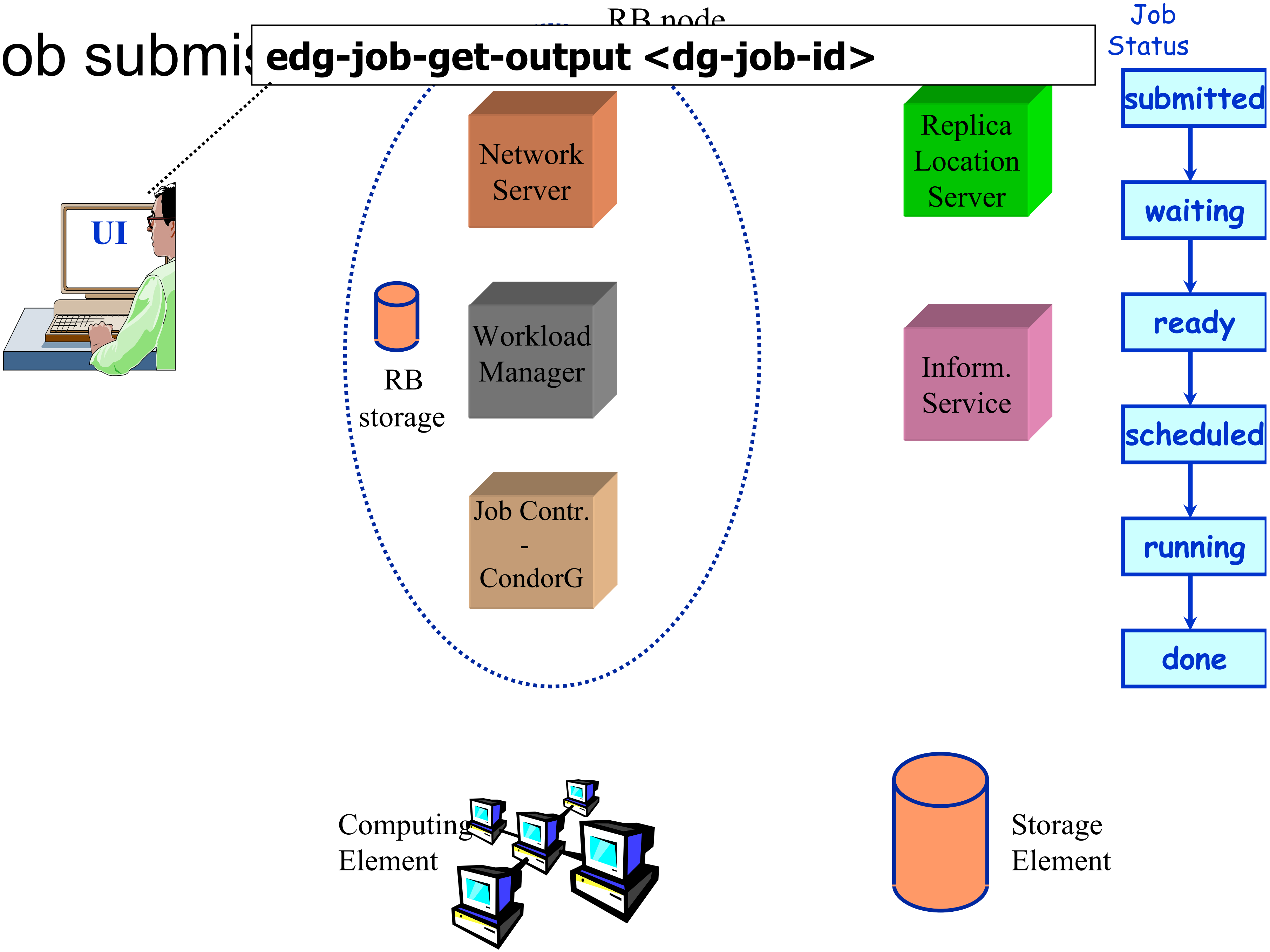
scheduled

running

done

Job submis

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



RB node

UI

Network Server

RB storage

Workload Manager

Job Contr. - CondorG

Replica Location Server

Inform. Service

Job Status

submitted

waiting

ready

scheduled

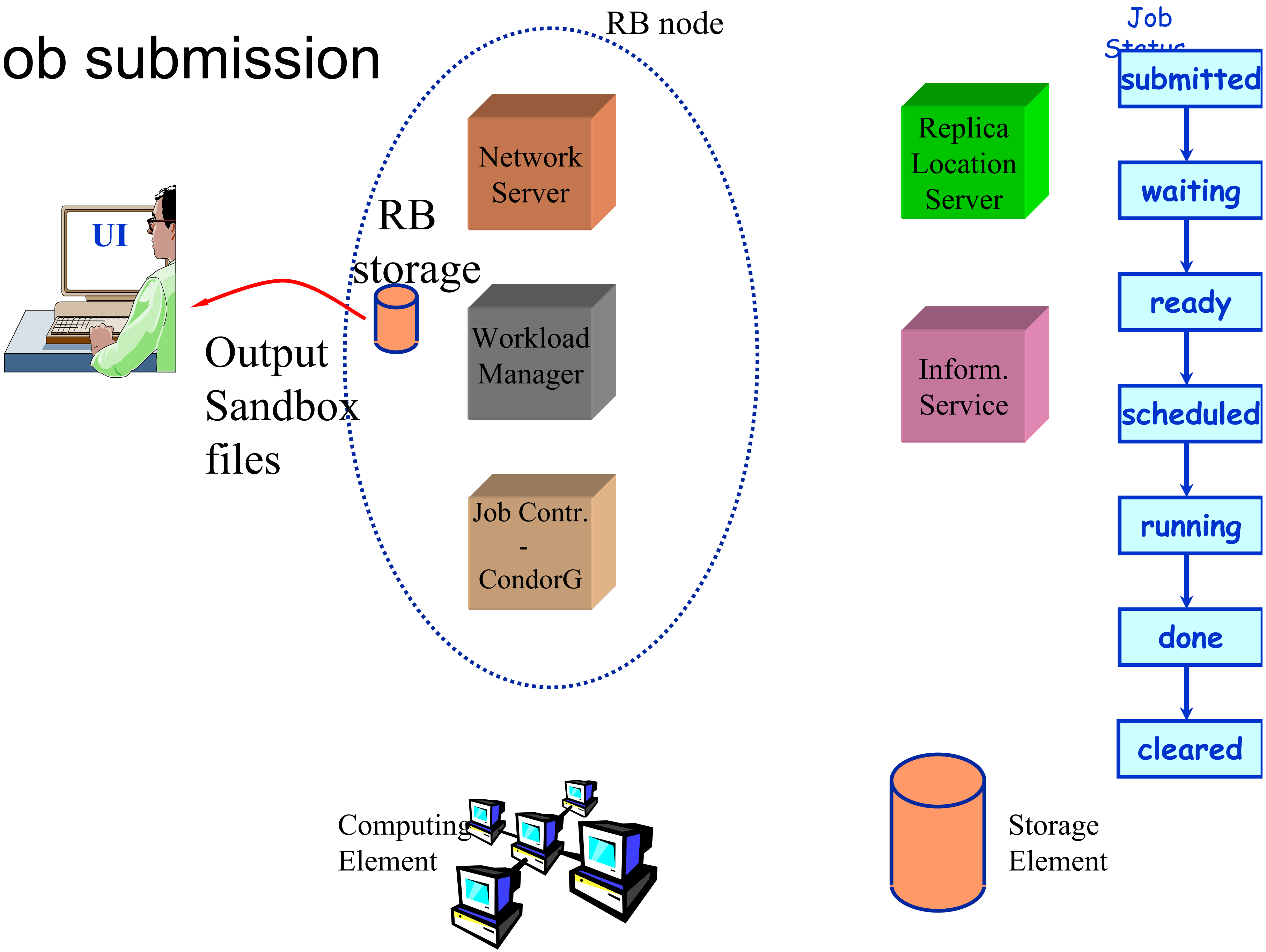
running

done

Computing Element

Storage Element

Job submission



RB node

UI

RB storage

Output Sandbox files

Network Server

Workload Manager

Job Contr. - CondorG

Replica Location Server

Inform. Service

Job Status

submitted

waiting

ready

scheduled

running

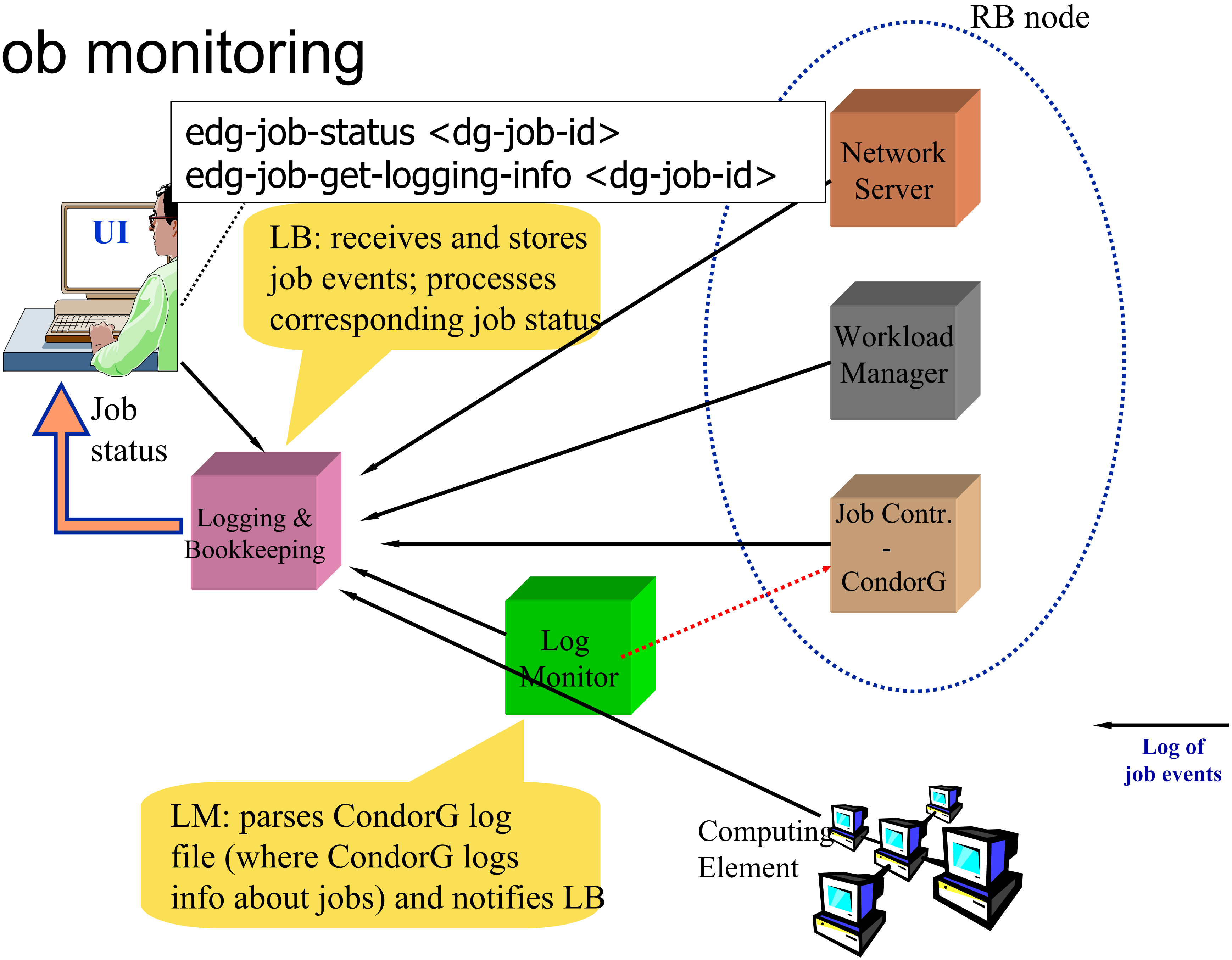
done

cleared

Computing Element

Storage Element

Job monitoring



```
edg-job-status <dg-job-id>
edg-job-get-logging-info <dg-job-id>
```

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

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

Interfaces

- Command-line: ssh onto user interface machine
- Portals – e.g. GENIUS – access from browser
- API's: functions invoked from programs
  - Job submission
  - Details: see talks in Madrid via EGEE training website!

About jobs…

- Where is the exe?
  - On the UI, downloaded in the sandbox
  - OR: On the Worker Nodes, downloaded for a VO
- Can MPI be run?
  - On some compute elements
  - NOT across compute elements
  - EGEE – DEISA links for HPC are intended !
- Can they be interactive?
  - Its been seen…BUT it is not supported…

Command-line: Job submission

- **edg-job-submit [–r *<res_id>]* [-c *<config file>]* [-vo *<VO>]* [-o *<output file>]* *<job.jdl>***
  - -r the job is submitted directly to the computing element identified by <*res_id*>
  - -c the configuration file <*config file*> is pointed by the UI instead of the standard configuration file
  - -vo the Virtual Organization (if user is not happy with the one specified in the UI configuration file)
  - -o the generated edg_jobId is written in the <*output file*>
    - Useful for other commands, e.g.:
    - **edg-job-status –i** <*input file*> (or edg_jobId)

      -i the status information about edg_jobId contained in the <*input file*> are displayed

Job Definition Attributes

- **Executable** (mandatory)
  - The command name
- **Arguments** (optional)
  - Job command line arguments
- **StdInput, StdOutput, StdErr** (optional)
  - Standard input/output/error of the job
- **Environment** (optional)
  - List of environment settings
- **InputSandbox** (optional)
  - List of files on the UI local disk needed by the job for running
  - The listed files are staged from the UI to the remote CE
- **OutputSandbox** (optional)
  - List of files, generated by the job, which have to be retrieved

Resource Attributes

- **Requirements**
  - Job requirements on computing resources
  - Specified using attributes of resources published in the Information System
  - If not specified, default value defined in UI configuration file is considered
    - Default: other.GlueCEStateStatus == "Production" (the resource has to be in the Production grid)
- **Rank**
  - Expresses preference (how to rank resources that have already met the Requirements expression)
  - Specified using attributes of resources published in the Information Service
  - If not specified, default value defined in the UI configuration file is considered
    - Default: - other.GlueCEStateFreeCPUs (the highest number of free CPUs)

“Data” Attributes

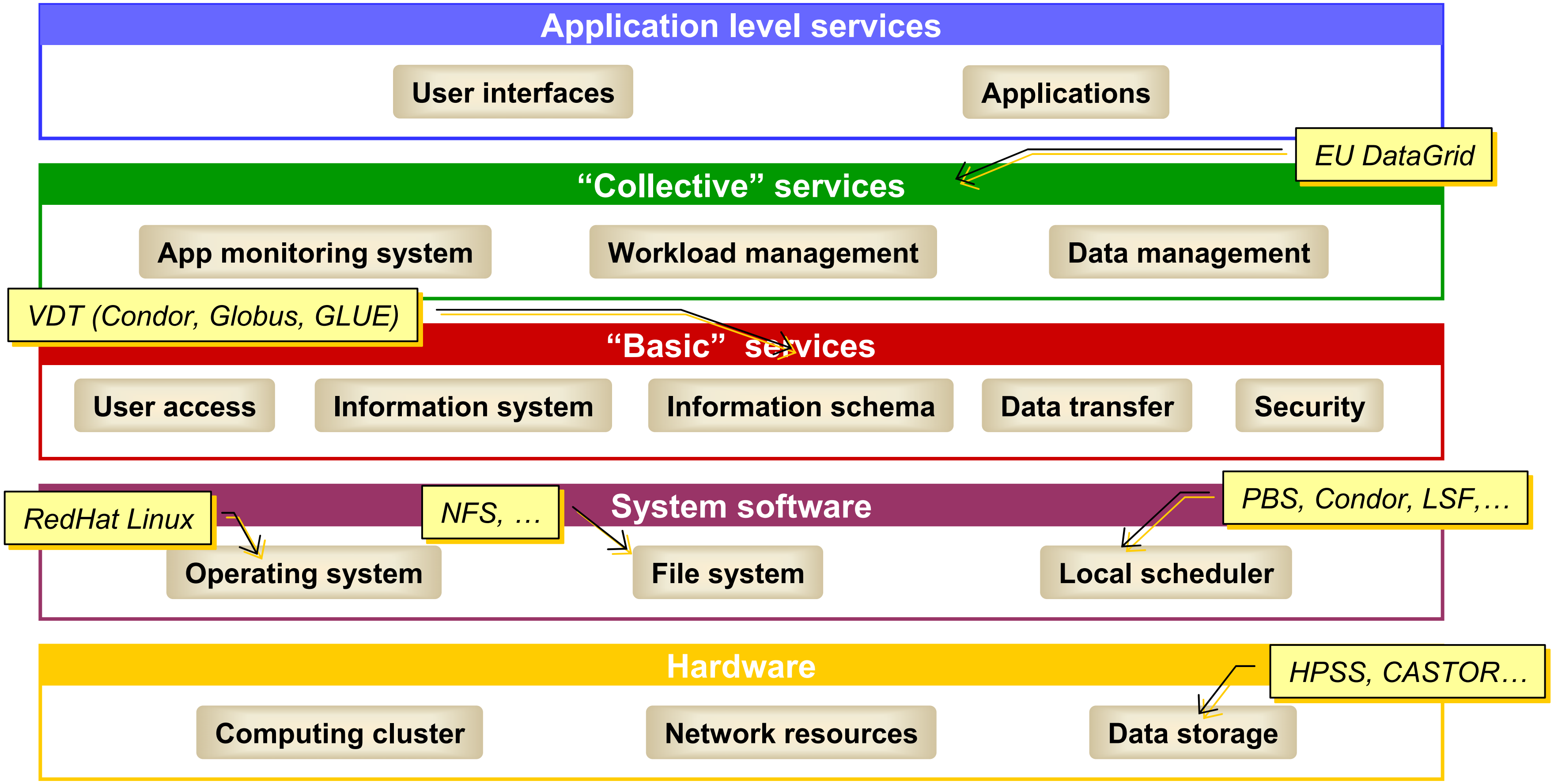
- **InputData** (optional)
  - Refers to data used as input by the job: these data are published in the Replica Catalog and stored in the SEs)
  - PFNs and/or LFNs
- **DataAccessProtocol** (mandatory if InputData specified)
  - The protocol or the list of protocols which the application is able to speak with for accessing ***InputData*** on a given SE
- **OutputSE** (optional)
  - The hostname of the output SE
  - RB uses it to choose a CE that is compatible with the job and is close to SE
- **OutputData** (optional)
  - Output Data that will be registered at the end of the job

Example JDL File

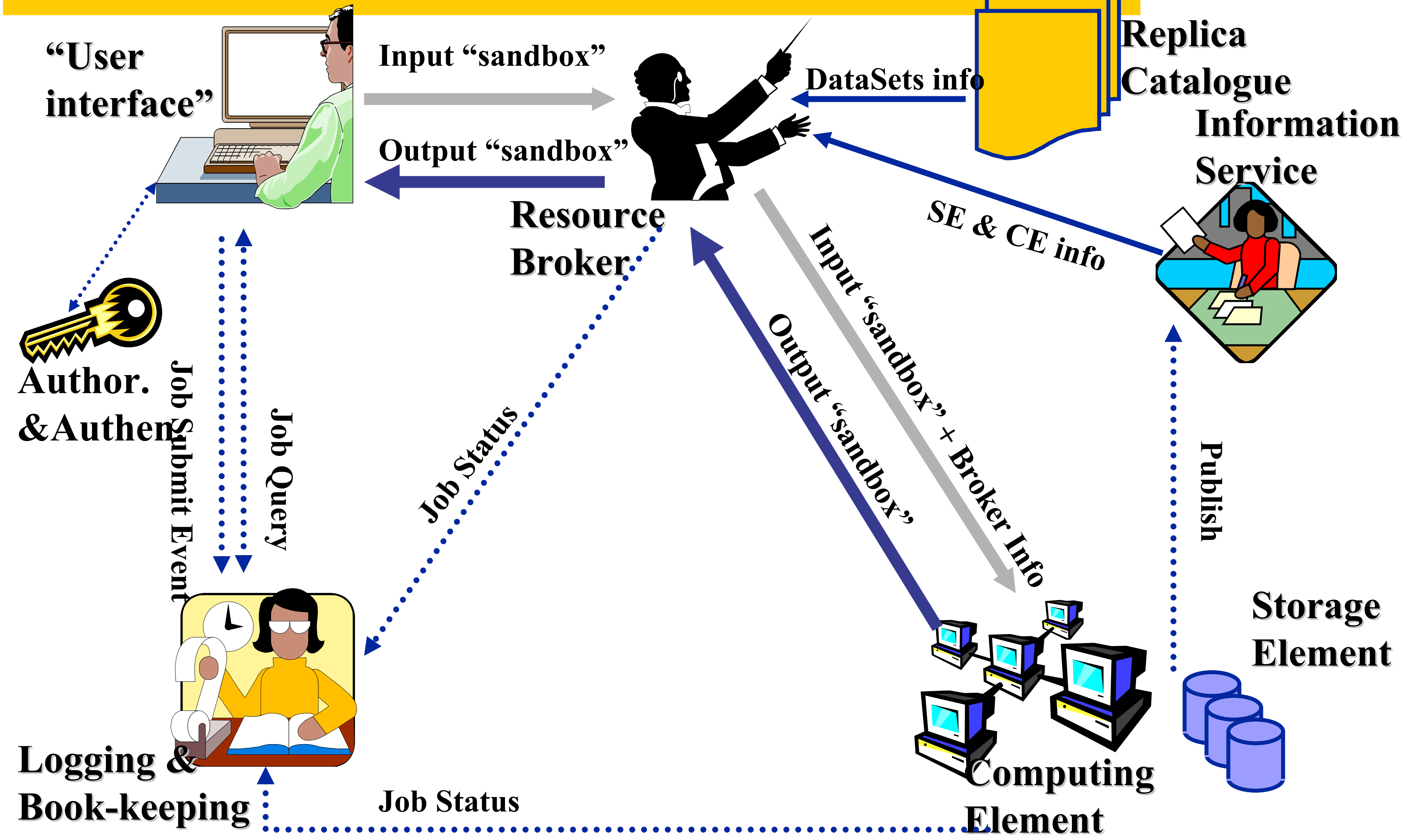
```
Executable = “gridTest”;
StdError = “stderr.log”;
StdOutput = “stdout.log”;
InputSandbox = {“/home/joda/test/gridTest”};
OutputSandbox = {“stderr.log”, “stdout.log”};
InputData = “lfn:testbed0-00019”;
DataAccessProtocol = “gridftp”;
Requirements = other.Architecture==“INTEL” && \
              other.OpSys==“LINUX” && other.FreeCpus >=4;
Rank = “other.GlueHostBenchmarkSF00”;
```

Current production mware: LCG-2

Enabling Grids for E-science in Europe



Summary: EGEE components



"User interface"
Input "sandbox"
Output "sandbox"
Resource Broker
DataSets info
Replica Catalogue
Information Service
SE & CE info
Author. &Authen.
Job Submit Event
Job Query
Job Status
Input "sandbox" + Broker Info
Output "sandbox"
Publish
Storage Element
Logging & Book-keeping
Job Status
Computing Element