GridICE

The eyes of the grid

A monitoring tool for a
Grid Operation Center by

DataTAG WP4

Sergio Fantinel, INFN LNL/PD

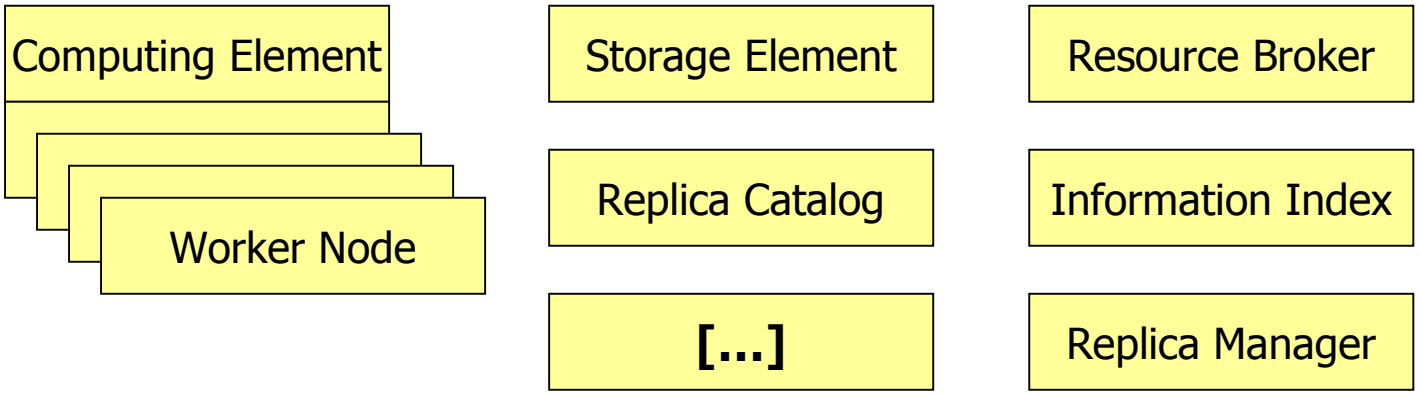
GridICE Actual Implemantation Outline

- Monitoring scenario
- Collection of info: EDG WP4 Fmon Framework & GLUE/GLUE+ Schema
- Discovery service: resources, components
- Server side services layout
- Graphs/data presentation service
- Next steps

Monitoring scenario

•Different layers of info generation

•Different points of view



LOW LEVEL measurements

- CPU load
- memory usage
- disk usage (per partition)
- network activity
- number of processes
- number of users (UI)
- ...

SERVICE checks

- gatekeeper
- gsiftp
- gris
- gdmp
- RB/LB
- ...

"GRID/VO" measurements

- number of total CPUs
- number of free CPUs
- number of running jobs
- number of waiting jobs
- SE free disk space
- ...

PROBLEMS:

- How to publish to the "world" the information of a site?
  - GLUE schema choice -> limitations -> GLUE +
- How to collect the information inside the site?
  - FMON choice - integration and enhancement

GLUE Schema

- Conceptual model of grid resources to be used as a base schema of the GIS (Grid Information Service) for **discovery** and **monitoring** purposes
  - model of computing resources (CE)
  - model of storage resources (SE)
  - model of relationships among them (close CE/SE)

- Implementation status (v. 1.1) (for Globus MDS)
  - LDAP schema (DataTAG WP4.1)
  - information providers (CE/SE)

(previous lecture made by S.Andreozzi on 21-07-2003)

Title : Glue Schema 1.0 (UML diagram)
Filename : Glue_1_0_Ext.vsd
Author : Sergio Andreozzi
Date : 09/01/2003
Namespace: Glue/EDT
Status : DRAFT

Host element -
DataTAG extensions
for the monitoring service

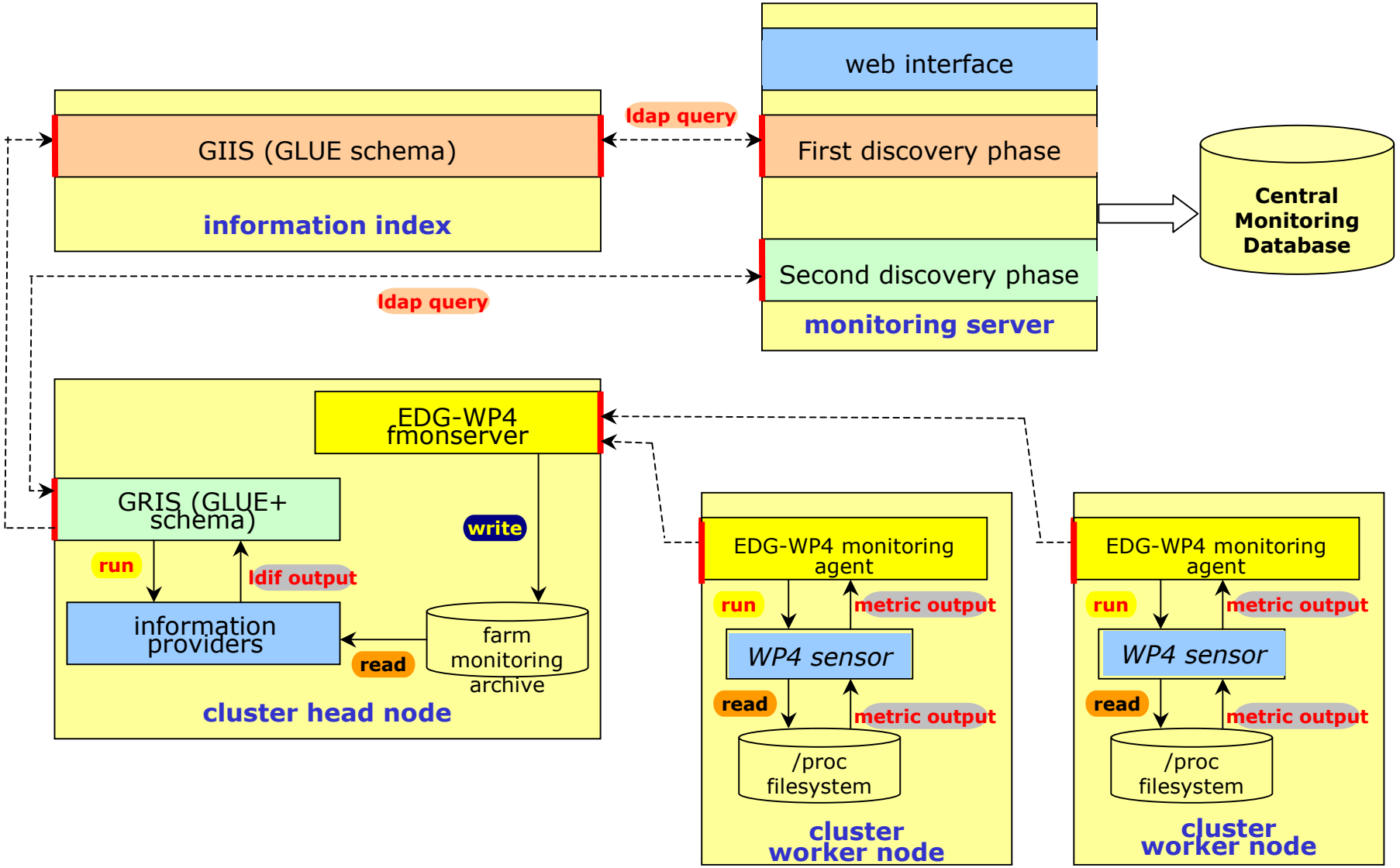
EDG WP4 FMon Framework

- It provides a client (**Monitoring Sensor Agent - MSA**) running sensors (Monitoring Sensors - MS) on each node to monitor, and a central server (**Fabric Monitoring Server - fmonServer**) to collect data.

- The server receives samples as they are measured by MSA, and stores them in a **flat file / Oracle** database

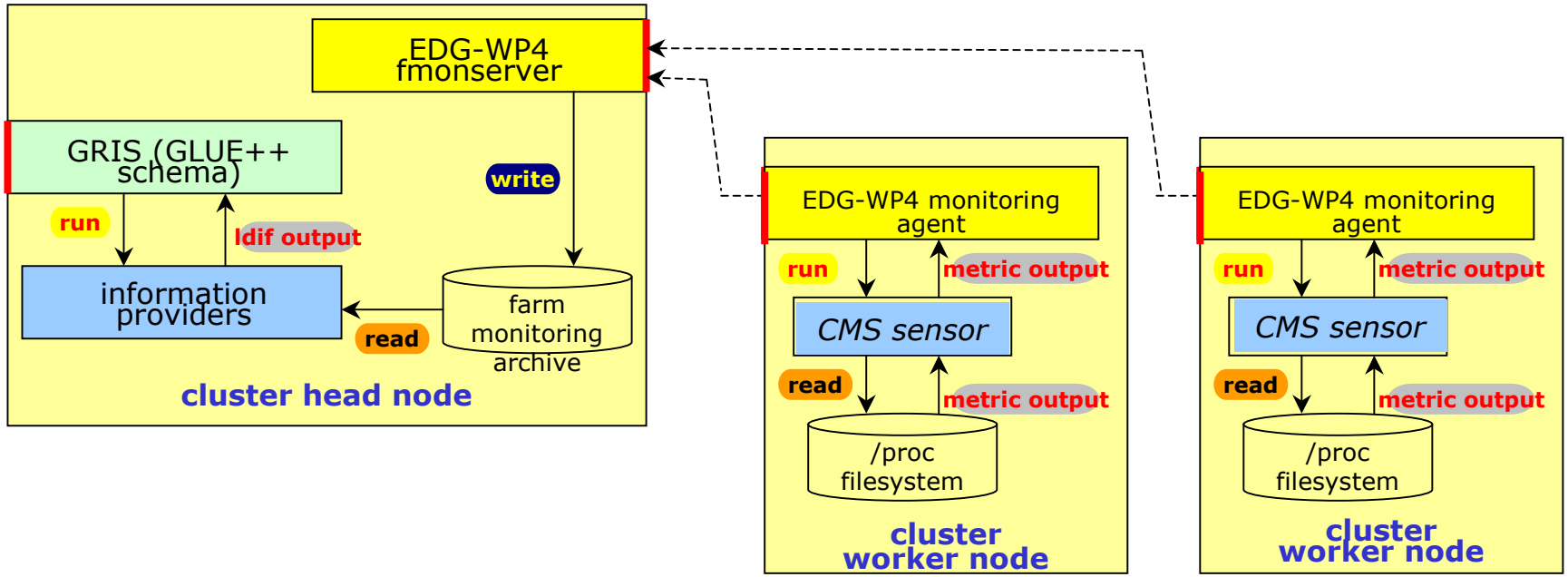
- The client is provided with a sensor (sensorLinuxProc) which uses /proc file system to measure various basic quantities on Linux (CPU load, network, etc).

An example scenario: cluster



Experiment Specific Measures Integration

- Possible and easy integration of VO/Experiment measures publication
- It must be modified the GLUE schema and write the experiment sensors (ex. CMS KIN/SIM event production)



PROBLEM:

- How to track new available or old dead resources?
- Different layers (GRID/Site) of resources
- Examples:
  - Computing service
  - Storage service
  - Software application (RunTimeEnv)
  - Computing node
  - Network adapter

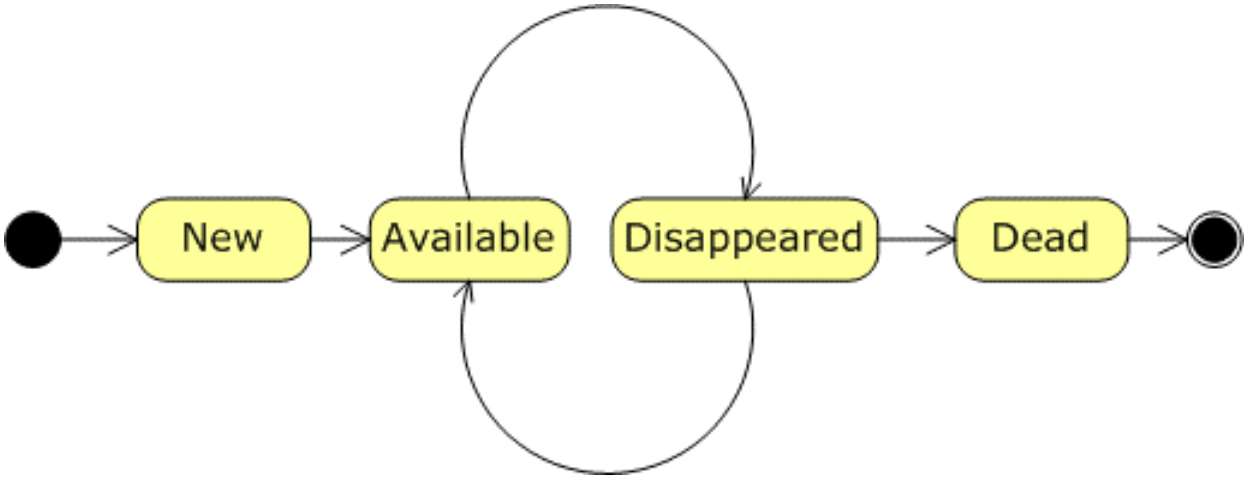
Discovery service: entities

- RESOURCES: are the entities discovered from the GIS, ex:
  - Cluster Head Nodes
  - Storage Services

- COMPONENTS: are the entities belonging to resources and discovered directly from resource itself, ex:
  - Computing Elements
  - Storage Space
  - Network Adapters

Discovery service: purposes

- track the life of the entities: they are characterized by a status (new, available, disappeared, dead).



- Configure the monitoring system accordingly the status of these entities to collect metrics, status and other info.

Discovery service: entities list

This is the list of entities currently tracked by the monitoring system:

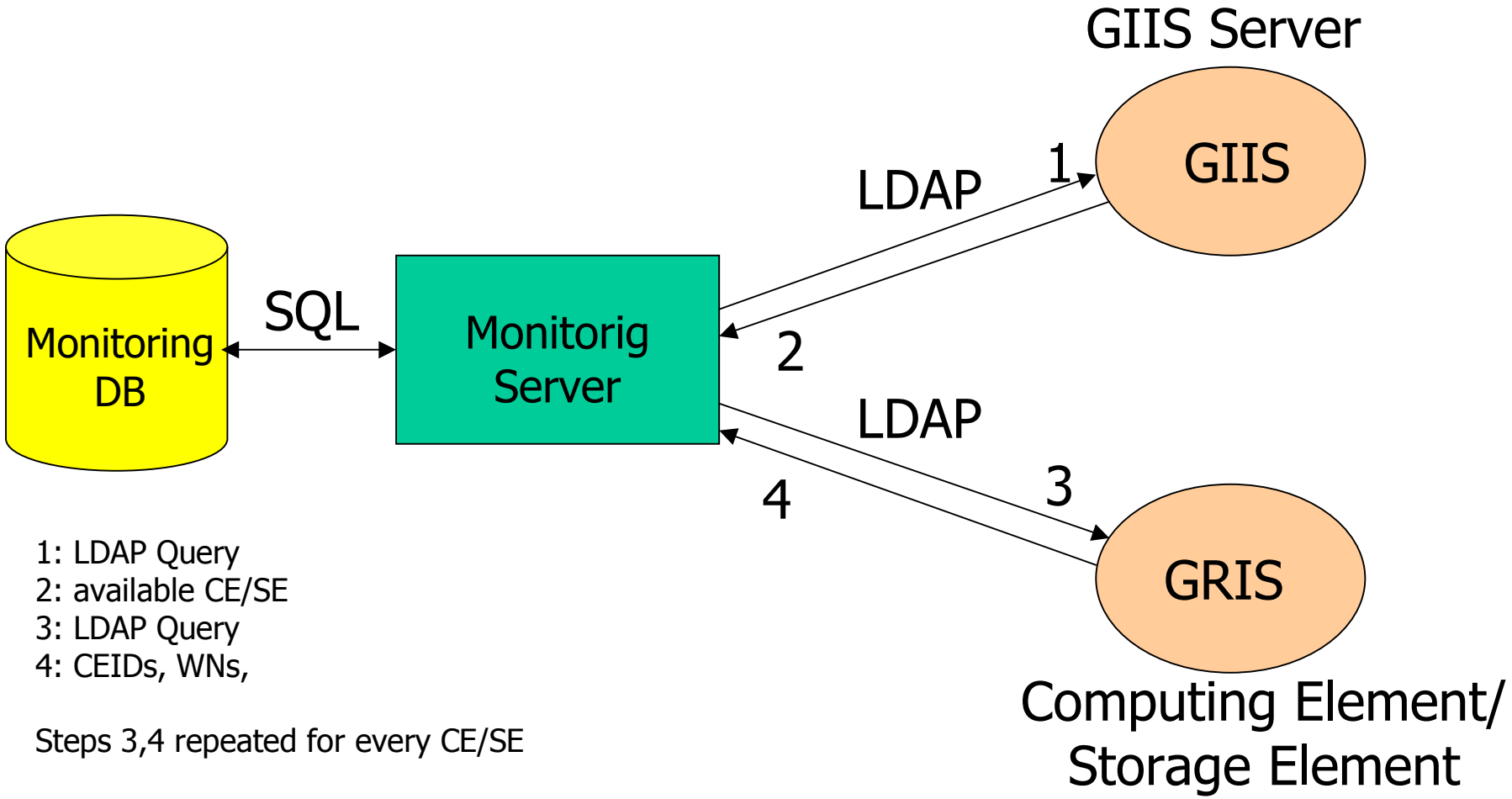
- Clusters
- Storage Services
- Worker Nodes (CL)
- Computing Elements (CL)
- Run Time Environments (CL)
- Virtual Organizations (CE)
- Storage Extents (WN)
- Network Adapters (WN)
- Storage Space (SE)
- Storage Protocols (SE)

CL	= Cluster
WN	= Worker Node/host
SE	= Storage Service

Discovery service: entities (2)

- Every entity (resource or component) is described by a number of characterizing information.
- Entities may be linked together:
  Ex. Network Adapter -> Worker Node -> Cluster
- To track the life of the entities it is used a SQL database where are stored also all the information related to every single entity.

Discovery service: entities (3)



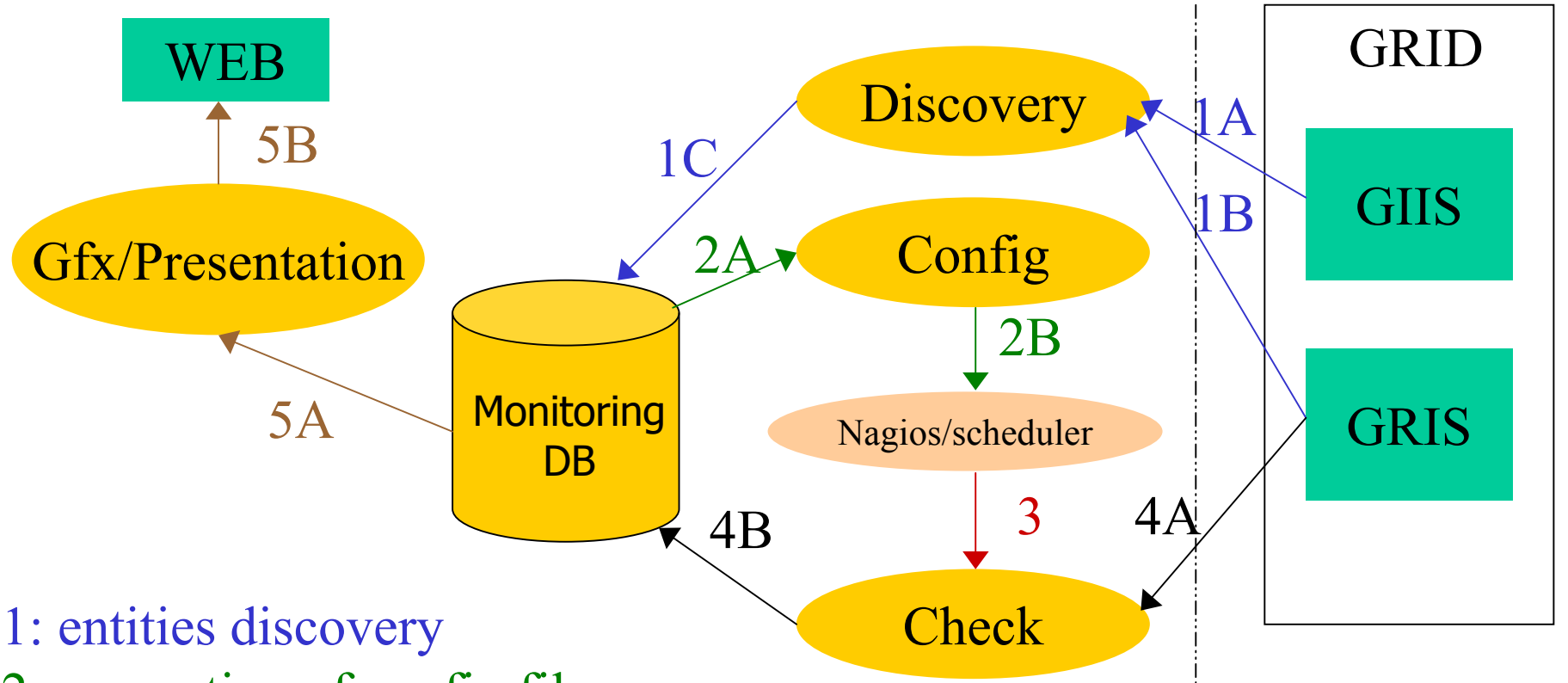
1: LDAP Query
2: available CE/SE
3: LDAP Query
4: CEIDs, WNs,

Steps 3,4 repeated for every CE/SE

Discovery service: config/check processes

- Nagios is a network general purpose monitoring tool
- All the info collected are used to generate a number of *Nagios* configuration files (configuration process).
- *Nagios* schedule, according to some other DB stored parameters, at different interval times, the execution of a number of scripts (*Nagios* plug-ins wrote by the DataTAG WP4) that collect the info associated to every entity (check process) and put those in the DB.

Server Side service layout



1: entities discovery
2: generation of config files
3: check scheduling
4: entities info collection
5: DB info rendering

Developed by **DataTAG WP4**

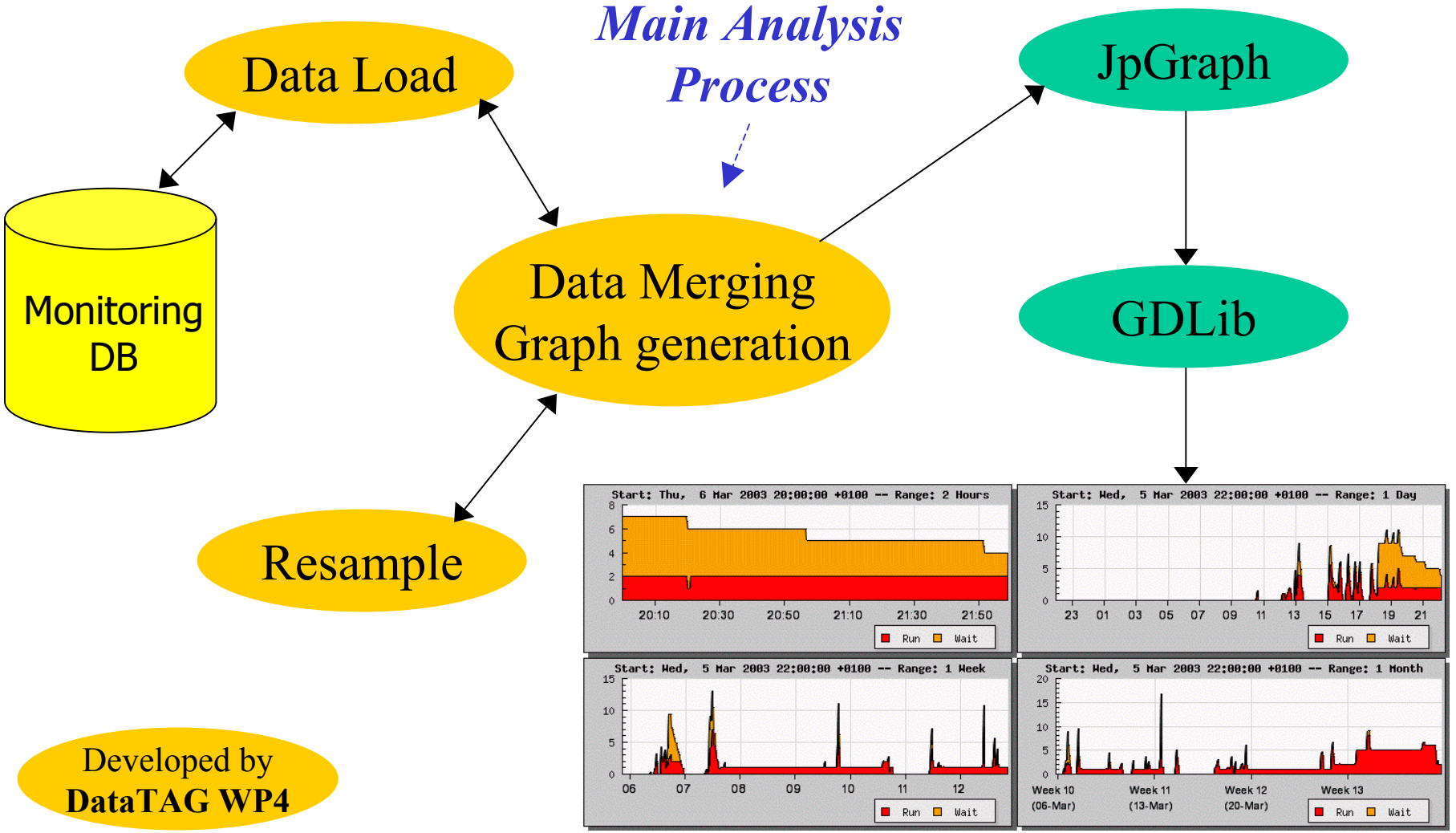
Discovery service: Scheduling

- Discovery and config generation run as cron jobs; although the two processes can be scheduled independently at different time intervals, a discovery is just followed by a config generation.
- Check plug-ins are scheduled by *Nagios*; the interval for each one is set by a corresponding parameter in the DB.

DataBase stored info

- Three types of information are stored in the database (~50 tables):
  - Entities: actual status, historical status (fed by discovery process)
  - Info about entities (fed by check process)
  - Monitoring configuration parameters (fed manually by monitoring administrator)

Data presentation service

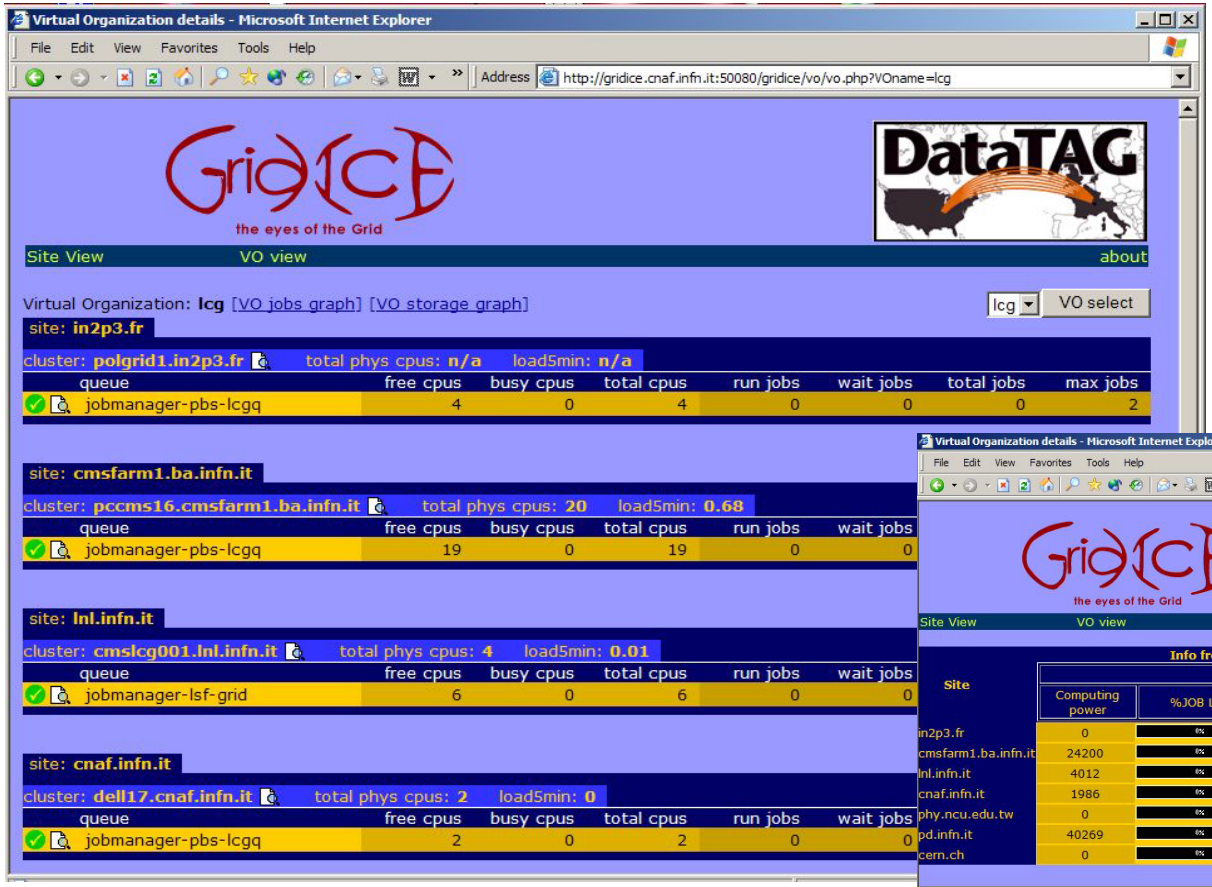


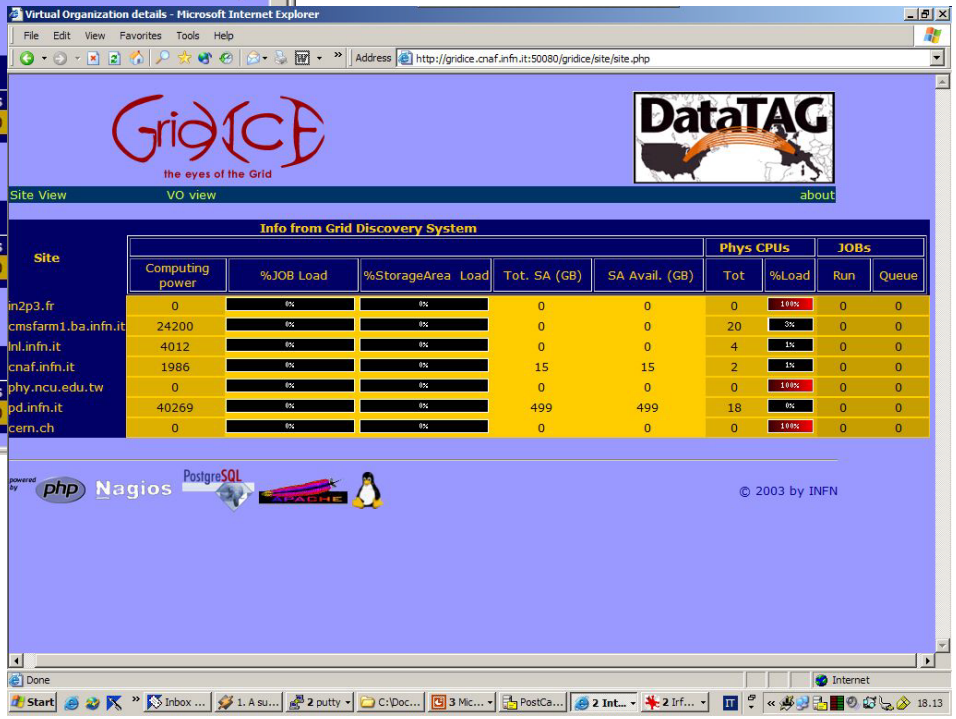
Data presentation service (2)

- The presentation of the date was made addressing different user types :
  - Vo views, for a VO manager
  - Site views, grid manager
  - Single entity grid/site manager

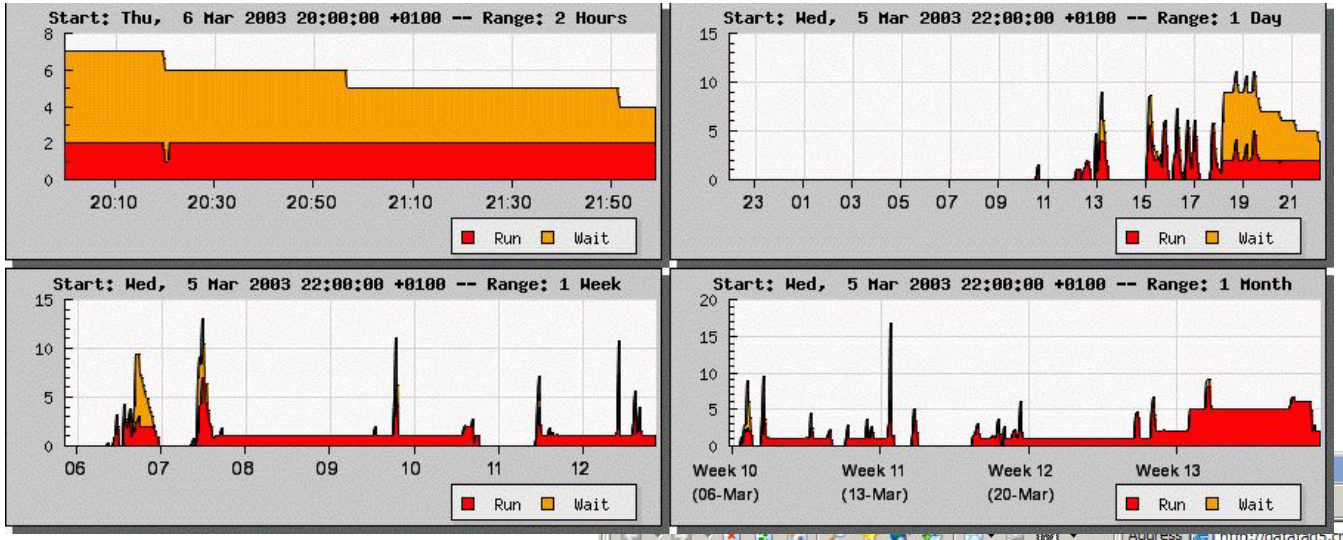
(see next slides / following there is a live session that demonstrate the features just discussed)

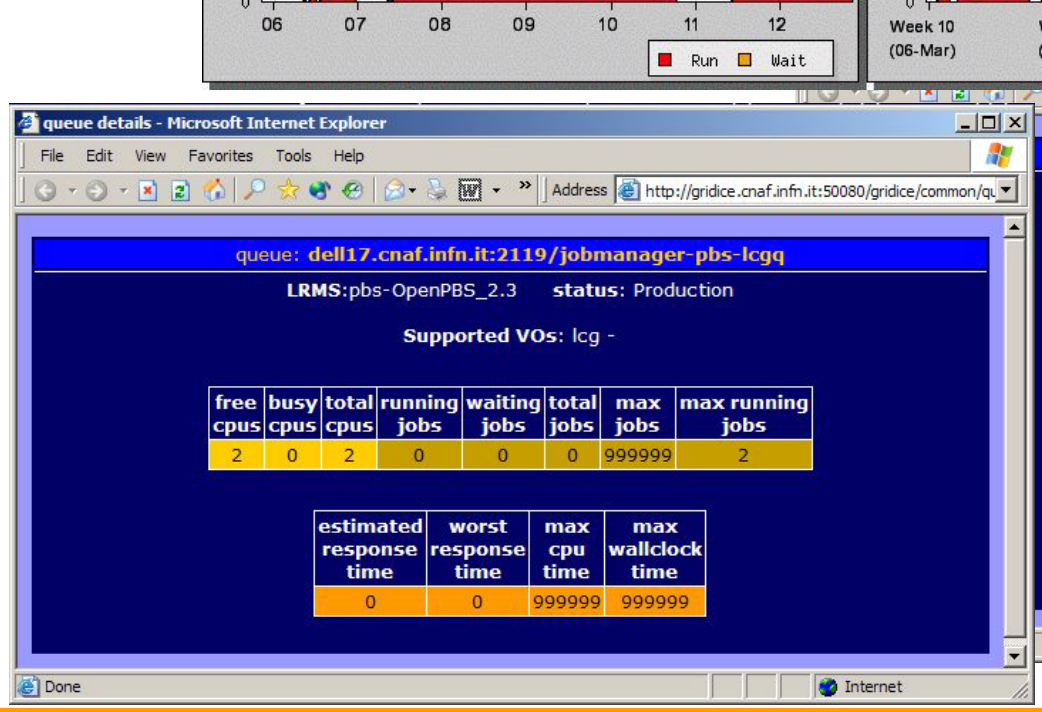
Data presentation service (3)

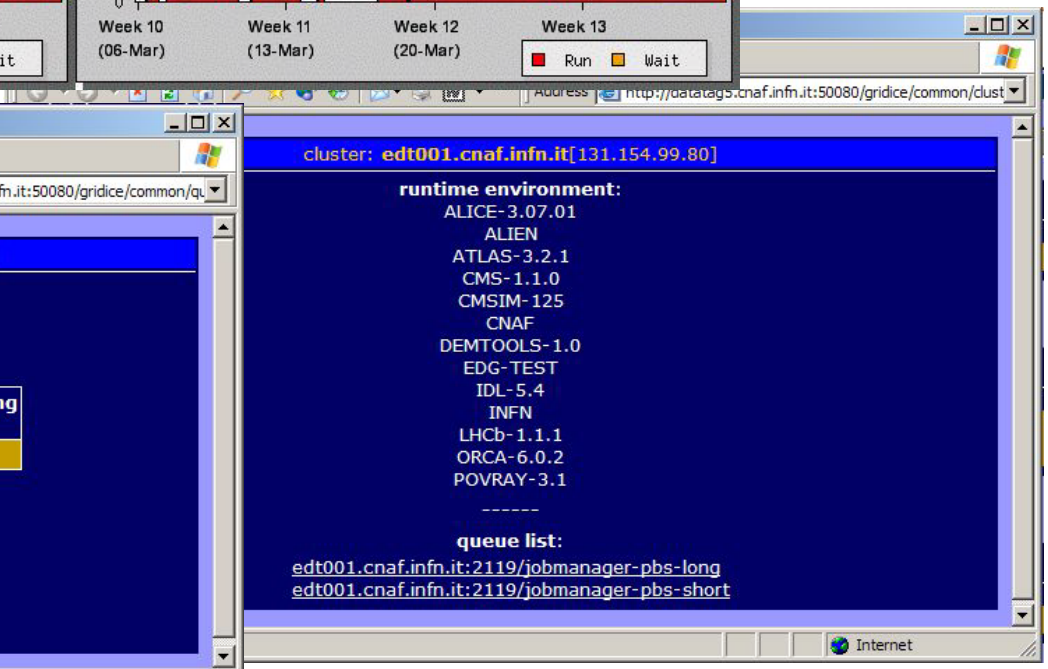




Data presentation service (4)







Next steps, short term

- Check plug-in refactoring: we made some tests with LDAP and to improve the performance we must aggregate the queries (less queries, more date to be transferred).

  Data reduction with the activation of the thresholds

  We are thinking to introduce some kind of caching for last data pushed in the DB to less stress the DB

- DB schema improvement: dynamic discovery of the URL GRIS (at the moment with GlueInformationServiceURL). Introduction of new components: CESEBind, SECEBind.

- Activation of the service (GRIS, GIIS, gridftp,…) checking

Next steps, short term (2)

- Grid Collective Service Monitoring (e.g. edg-broker, edg-replica-location-service)
- Job Monitoring at queue level (some open issues, ex. VO)
- Native R-GMA support as GIS: we need a working and stable testbed with R-GMA as GIS, extend the CE GIN to support the new metrics.
- Hosts Role (via GlueHostService) in order to associate service state to proper host state