Workload Management System and Reliability

WLCG Reliability Workshop
CERN, 26-30 November 2007

Francesco Giacomini – INFN

Design Decisions

- **Decouple Workload Management System from Logging&Bookkeeping Service**
  - Independent services running on different nodes
  - The WMS processes requests
  - The LB stores significant events logged during job lifetime and computes job state
  - LBProxy (on a WMS node) provides a cache for jobs still active on that WMS
- **The WMS's and the LB's are independent**
  - N:M mapping
  - Easier scalability

Design Decisions /2

- **Decouple interaction with clients from the processing of the requests**
  - Interaction with clients requires higher reliability and availability w.r.t. the back-end
- **Isolate different functionalities in different processes**
  - Persistent communication among them (e.g. *filelist*)
- **Decouple collection of information from external sources from its use**
  - e.g. *Information SuperMarket*, to store info on the available computing and storage resources found in the Information System
    - Info kept in memory already in the right format to do matchmaking

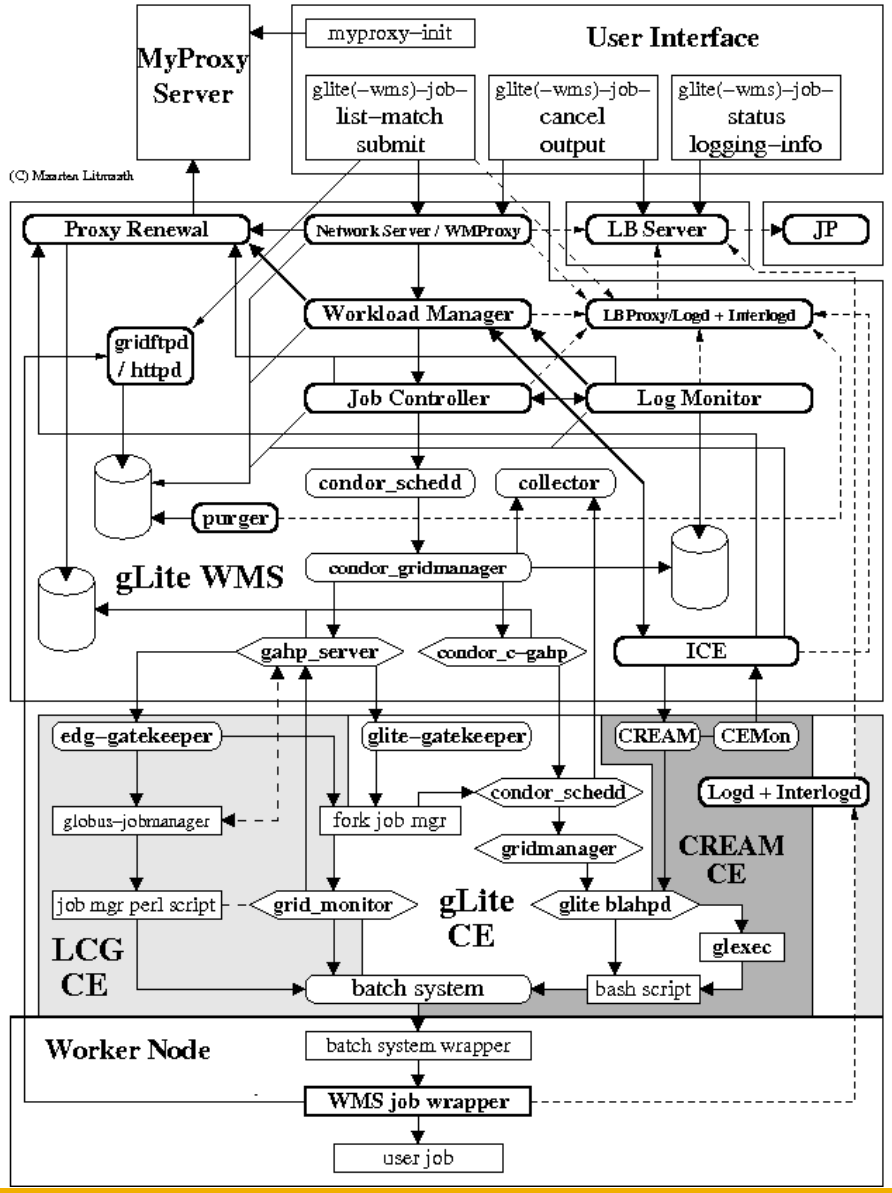
Implementation techniques

- **Mostly written in C++**
- **Exploit *Resource Acquisition Is Initialization* (*RAII*)**
  - Acquire a resource in the constructor, release it in the destructor
  - Prevent leakage of memory, locks, temporary files, ...
  - Useful to undo operations in case of errors
- **Use advanced libs**
  - standard C++ lib
  - boost libs

Tricks

- **Commit periodic suicide of services known to leak**
- **Inadequate memory management in the C++ runtime**
  - Use of alternative allocators (e.g. google-malloc)
- **Need to spawn a large number of processes with a large memory footprint**
  - Spawn a smaller proxy process instead, whose only purpose is to start the real process when needed
- ***Maradona* file**
  - copy the JobWrapper stdout back with an alternative mechanism
- ***Shallow* resubmission**
  - Require a job to acquire a token before start running

Complexity and Reliability



Complexity and Reliability /2

- **Complexity is a major enemy of reliability**
- **Some complexity is intrinsic in the system, in order to support the provided functionality**
- **Many dependencies on components we don't control**
  - e.g. much stability gained implementing support for job collections (and bulk matchmaking) directly in our code
- **Too many variants of similar services to support**
  - Information System: BDII, RGMA, CEMon
  - File/Replica Catalogs: RLS, DLI, SI
- **Too much persistency points**
  - Simplification going on with the code restructuring activity