Job Submission and Management through Web Services: the Experience with the CREAM Service

Luigi Zangrando – INFN Padova - ITALY

CHEP 2007 – Victoria, CANADA
2-9 September 2007

www.eu-egee.org

CHEP'07 VICTORIA, BC

International Conference on Computing in High Energy and Nuclear Physics
2-7 Sept 2007 Victoria BC Canada

Information Society and Media

CREAM

- **Modern GRID middlewares as gLite (EGEE project) are composed of a set of components (services) providing basic functionality, such as:**
  - data storage, authentication and security, job management, resource monitoring and reservation, etc.

- **Computing Resource Execution and Management (CREAM) Service is a system for job management and access to computational resources at Computing Element (CE) level developed for gLite by INFN (Padova – ITALY)**
  - we consider a computing resource (defined as Computing Element, CE) typically a cluster of PCs managed by a Local Resource Management System (LRMS)
    - LSF, PBS/Torque, Condor, …

CREAM requirements

- **The main CREAM requirements are basically:**
  - provide a well defined and expressly minimal set of operations (functionality)
  - to be open to emerging standards
  - provide a lightweight, flexible and expansible architecture
  - to be robust (fault tolerant) and scalable
  - guarantee performance and reliability
  - security
  - accounting

CREAM functionality

- **Job submission**
  - Submission of jobs to a CREAM based CE
  - Includes also support for direct staging of input sandbox files
    - actually the operation is split in 2 operations (job register and job start)
  - Job characteristics described via a:
    - JDL (Job Description Language) expression
      - *CREAM JDL is basically the same JDL used by the EGEE gLite Workload Management System (with CREAM-specific extensions)*
    - JSDL (Job Submission Description Language)
      - *OGF specification for describing the requirements of computational jobs for submission to resources in Grid environments*
      - *through the OGSA-BES interface*
  - Supported job types
    - Simple, Sequential batch jobs
    - MPI jobs
    - Support of bulk jobs in progress
      - *DAG jobs, parametric jobs, job collections*

CREAM functionality

- **Proxy delegation**
  - Possibility to automatically delegate a proxy for each job submission
  - Possibility to use a previously delegated proxy for multiple job submissions
    - recommended approach wrt performance, since proxy delegation can be “expensive”
- **Job status**
  - To get status and other info (e.g. creation/submission/start execution/job completion times, worker node, failure reason, e.g.) of submitted jobs
  - Also possible to apply filters on submission time and/or job status
- **Job list**
  - To get the identifiers of all your jobs
- **Job cancellation**
  - To cancel previously submitted jobs
- **Job suspension and job resume**
  - To hold and then restart jobs

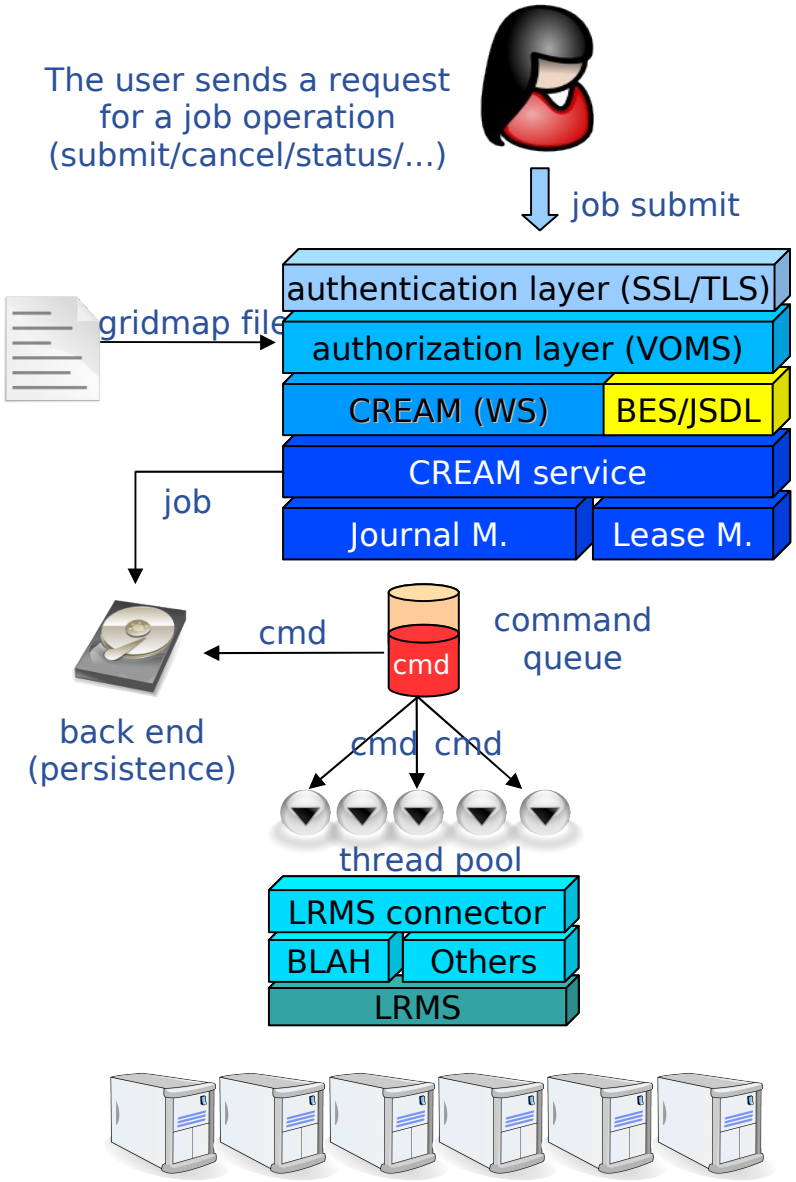
CREAM functionality

- **Job purge**
  - To clear a terminated job from a CREAM based CE
  - Can be explicitly called by the client, or can be called via a cron job (e.g. to clean old jobs)
- **Disabling of new job submissions**
  - Can be used only by CE admin e.g. if the CE has to be shutdown for maintenance
  - Other operations still allowed
  - Also possible to define policies on waiting/pending/running jobs to disable new job submissions
    - e.g. disable new submissions if the number of active jobs is > 3000
    - useful to avoid the CE overloading
- **Job Lease**
  - mechanism for canceling all jobs for which the lease time is expired
- **Accounting**
  - For each job proper information is logged in a log file which is then “managed” by the EGEE accounting service

Open Standards

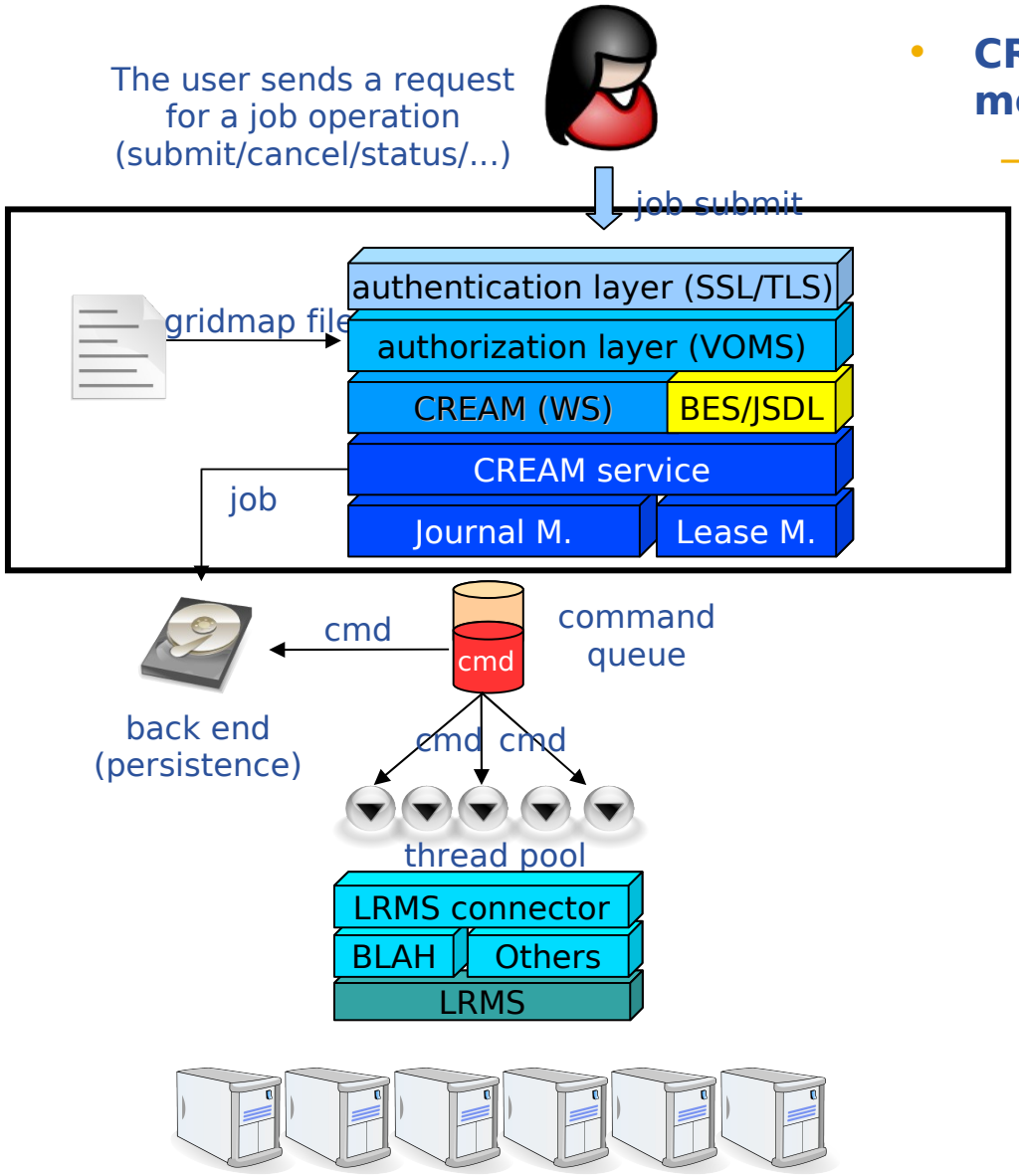
- CREAM is open to emerging standards
- Open Standards adoption is fundamental for CREAM strategies
  - they guarantee a high degree of interoperability
  - **Web Services** and **Grid Services** related technologies
- CREAM exports a well defined Web-service interface (WS-I compliant)
- CREAM supports also OGSA-BES (Basic Execution Service) interface
  - CREAM interface and BES interface coexist
  - CREAM-BES developments done in collaboration with the OMII-EU project
  - Shown at SC'06 (Tampa-FLORIDA) in the interoperability demo with other computational services
  - next interoperability demo will be at SC'07 (Reno - Nevada)

CREAM architecture



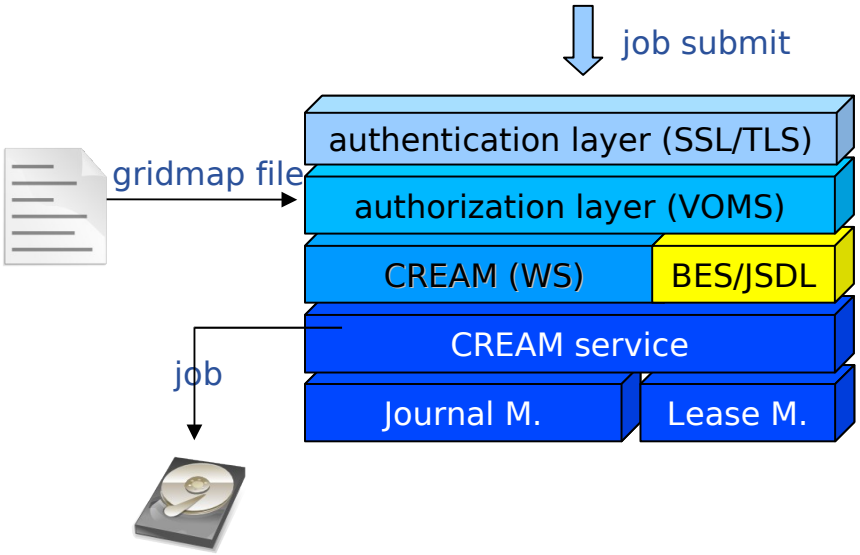
- **CREAM adopts the SOA architectural model**
  - interoperability and flexibility

CREAM architecture



- **CREAM adopts the SOA architectural model**
  - interoperability and flexibility

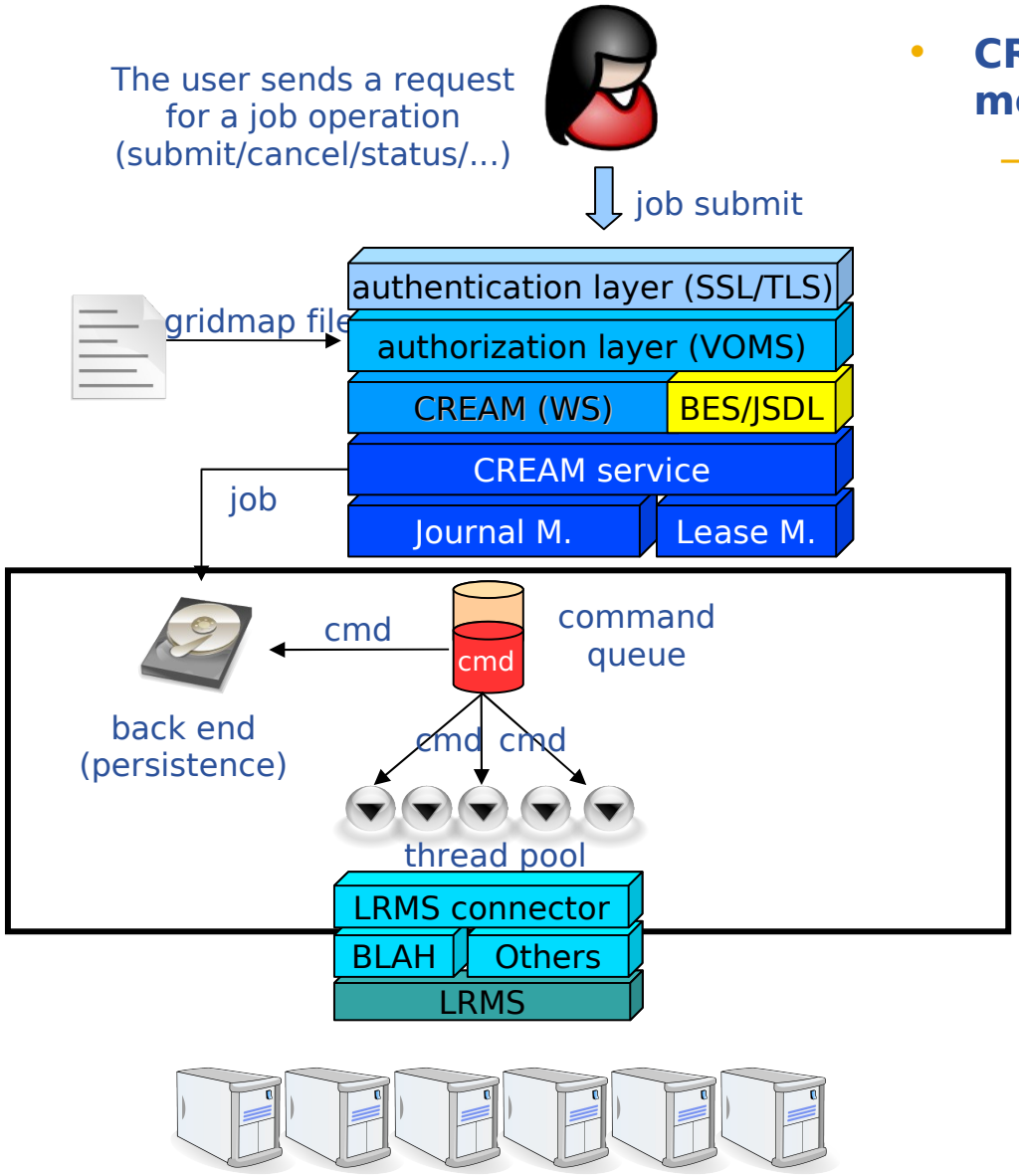
CREAM architecture



- To get access to the CREAM service it is needed to cross the AuthN and AuthZ layers;
- The DN and VOMS attributes are extracted from the user's proxy certificate;
- The AuthZ is based on VOMS attributes and on the gridmap file;

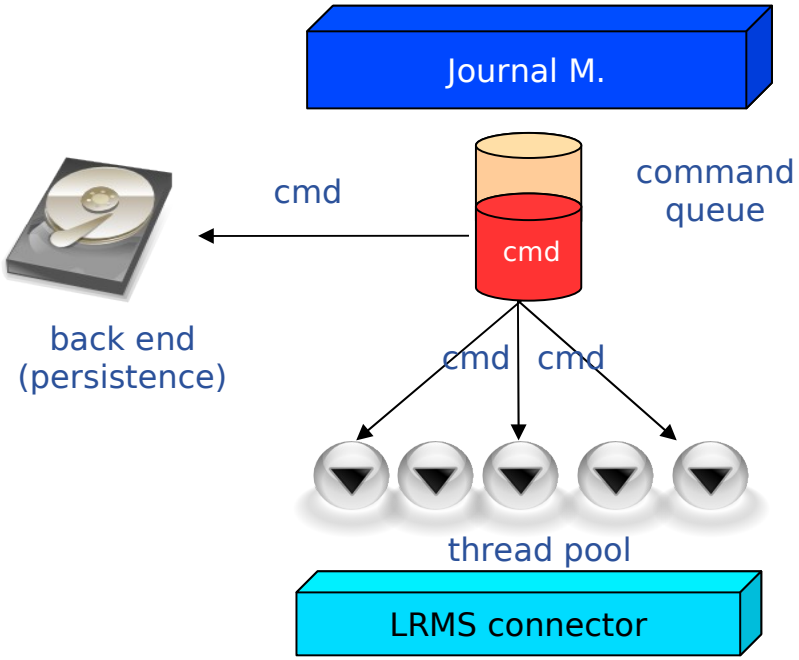
- CREAM is fault tolerant (job info and user's requests are persistent)
- The Journal manager handles all user's asynchronous requests (cmd) and inserts them on a FIFO queue.

CREAM architecture



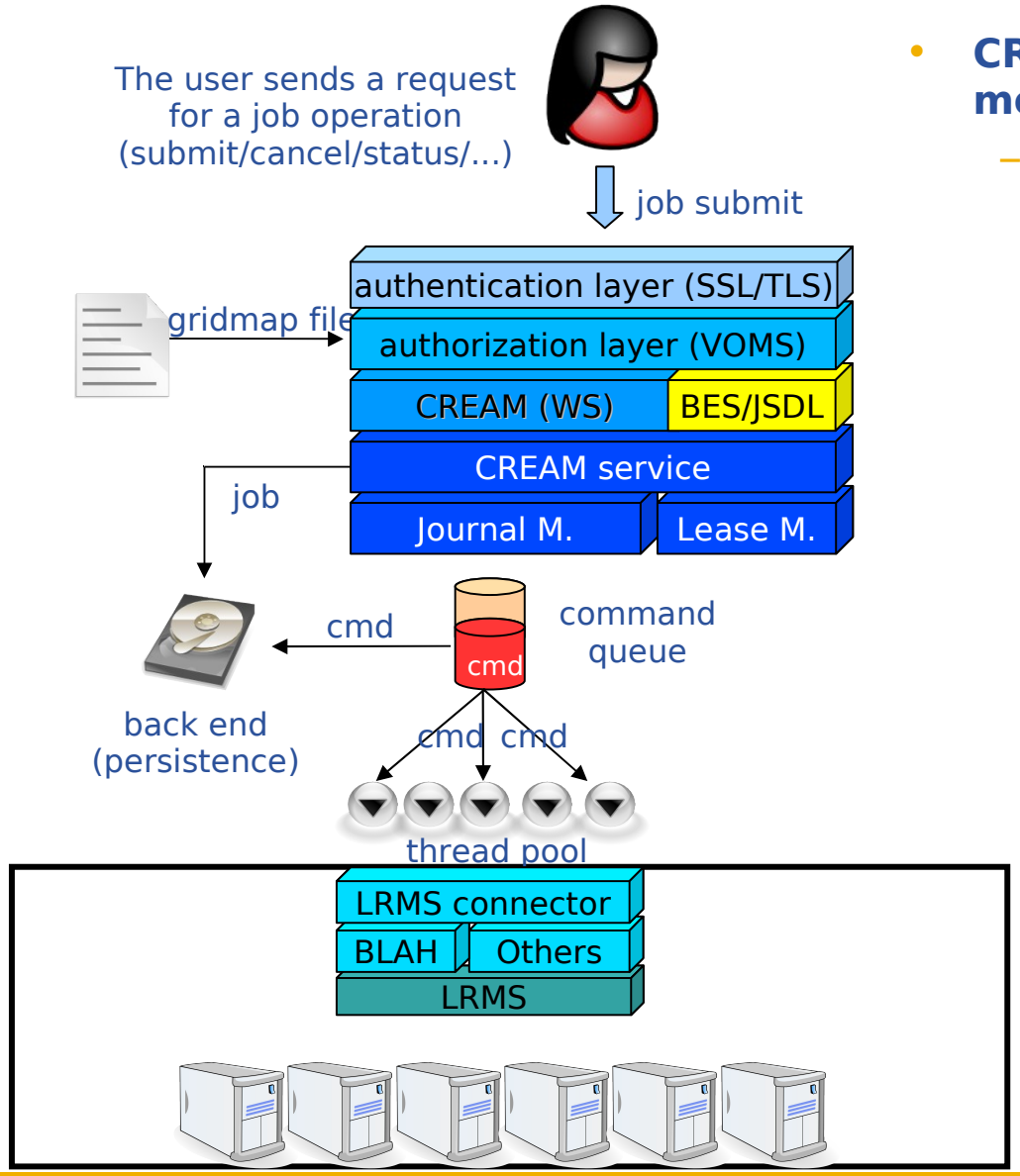
- **CREAM adopts the SOA architectural model**
  - interoperability and flexibility

CREAM architecture



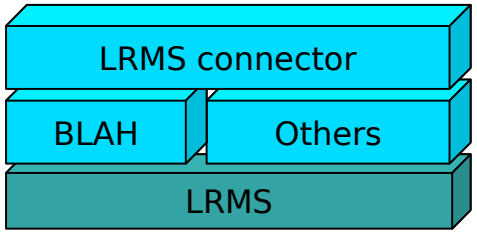
- All user commands are persistently stored on the JM's back-end (queue);
- A pool of threads fetches the commands from the queue and processes them (in parallel);
  - number of threads is configurable
- All threads act with the LRMS through an abstract layer (LRMS connector);
  - interface with underlying resource management system

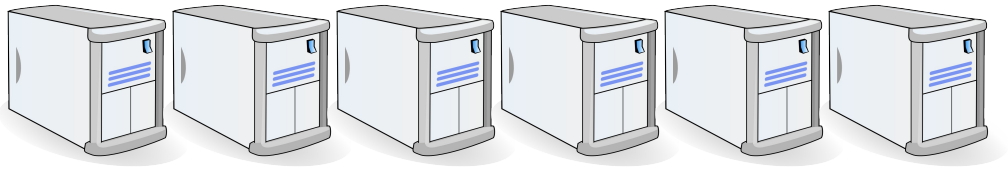
CREAM architecture



- **CREAM adopts the SOA architectural model**
  - interoperability and flexibility

CREAM architecture





- The LRMS connector interface provides an abstraction of the LRMS functionality;
- Ad-hoc connectors for new LRMS can be implemented;
- The connectors are pluggable;
- CREAM provides the BLAH (Batch Local Ascii Helper) connector
  - Light component accepting commands to manage jobs on different resource management system
  - Support for LSF and PBS/Torque currently provided (support for Condor is on-going)
  - BLAH manages job management operations on behalf of CREAM
  - BLAH also notifies CREAM about job status changes
    - good performance and efficiency because the job status changes are detected promptly

Security in CREAM

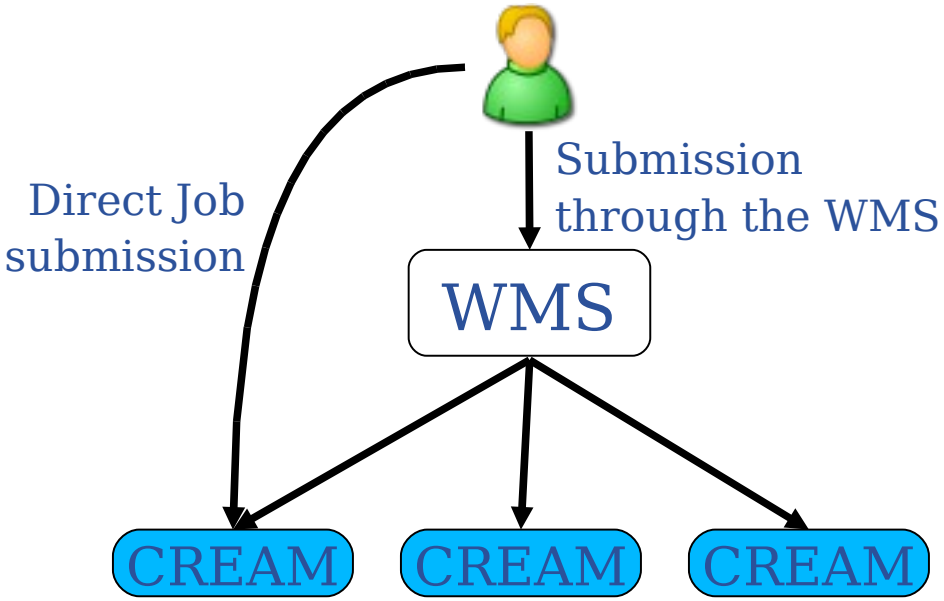
- **CREAM security architecture follows the guidelines of the Global EGEE security architecture, relying on the official tools provided by the EGEE security group**
- **Authentication**
  - PKI based infrastructure
  - X.509 certificates
  - Authentication implemented via EGEE gLite trustmanager for the server side
  - gSoap-plugin, gridsite libraries for the client side (for delegated-proxy creation)
- **Authorization**
  - Based on the EGEE gLite authorization framework (gJAF)
  - Virtual Organization based AuthZ and/or possibility to enable/disable specific Grid users
    - VOMS attributes stored as AC into the proxy-certificate
    - VOMS attributes into SAML assertion (next step)
  - local authorization based on gridmap-file
  - A user can manage (e.g. cancel, monitor) only her jobs
    - but possibility to define CE admins, who can manage also jobs submitted by other users
    - implemented via an AdminPIP, plugged in the Authorization Framework

Security in CREAM

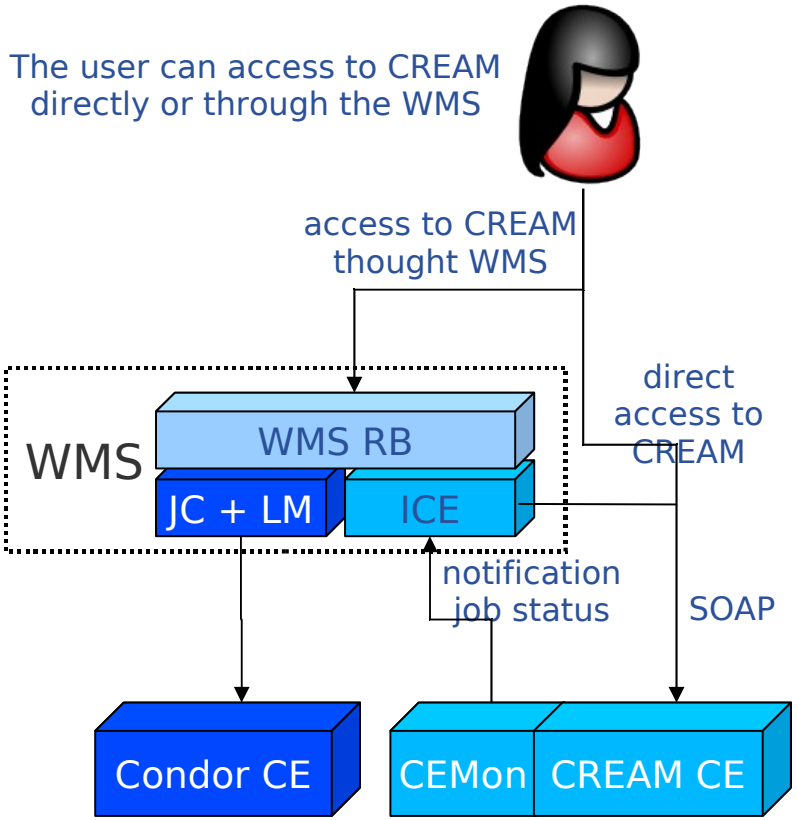
- **Delegation**
  - Using the EGEE gLite port delegation stuff
  - Delegation port embedded into CREAM
    - Not a standalone service
- **Credential mapping**
  - To map Grid credential on local accounts
    - Implemented via glexec (another EGEE gLite product)
      - *glexec is a thin layer to change Unix credentials based on Grid identity and attribute information*
      - *glexec is based on lcas and lcmaps*
    - glexec used in BLAH
      - *The commands to interact with the underlying batch system (submit, cancel,et.) are glexec-ed*
    - glexec used also in the CREAM service itself
      - *For specific operations that should be executed as the local user mapped to the considered Grid user*

CREAM usage scenario

- **CREAM should be invoked:**
  - By a generic client (e.g. an end-user willing to interact directly via the Computing Element)
  - Through the EGEE gLite Workload Management System (WMS)



gLite WMS – CREAM integration



- The CREAM integration with **WMS** is allowed by **ICE** (Interface to CREAM Environment);
- ICE: is an intermediate layer (gSOAP/C++) that must considered as a client of CREAM.
- ICE subscribes to the **CEMon** service in order to asynchronously receive notifications about the job status changes.
- In case some notifications are lost, ICE performs synchronous status polling for jobs for which it hasn't received status for some time.
- To maintain its subscriptions ICE periodically checks them and renews the expiring ones.

CEMon[itor] is a general purpose notification framework working in synchronous and asynchronous mode, that virtually supports any kind of monitoring thanks to its plug-in architecture. CEMon gets informations about the job status from the CREAM data persistence back end through JNDI APIs.

Performance/Reliability tests

- **In July and August 2007 CREAM passed the acceptance tests defined by the EGEE project**
  - performance and reliability tests

- **Test criteria:**
  - performance test:
    - 5000 simultaneous jobs per CE node (scheduled + running)
    - job submission rate of 10000 jobs/day
    - 50 different users submitting jobs to a single CREAM-CE node
  - reliability test:
    - job failure rates in normal operations due to the CE <0.5%
    - job failure due to restart of CE services or reboot <0.5%
    - 5 day unattended running with performance on day 5 equal to that on day 1

- **test results:**
  - > 8 days long
  - > 86000 jobs submitted via gLite WMS
  - no error due to CREAM
  - no performance degradation

Future plan for CREAM

- **The EGEE TCG recently decided to increase the effort on CREAM to make it production ready on SL4 with VDT 1.6.**
- **next steps:**
  - complete and verify the packaging and the accessory functionalities (accounting, monitoring, etc...) needed by the EGEE infrastructure
  - consolidate CREAM: further stronger tests
    - e.g. unlimited number of submitting users
    - Job failure rates due to the CE < 0.1%
    - 1 month unattended running without significant performance degradation
    - Self-limiting behaviour when the CE load reaches its maximum (e.g. 5000 running jobs)
    - ...
  - integration with Condor-G
  - participation to next interoperability demo at SC'07