CREAM: a WebService based CE

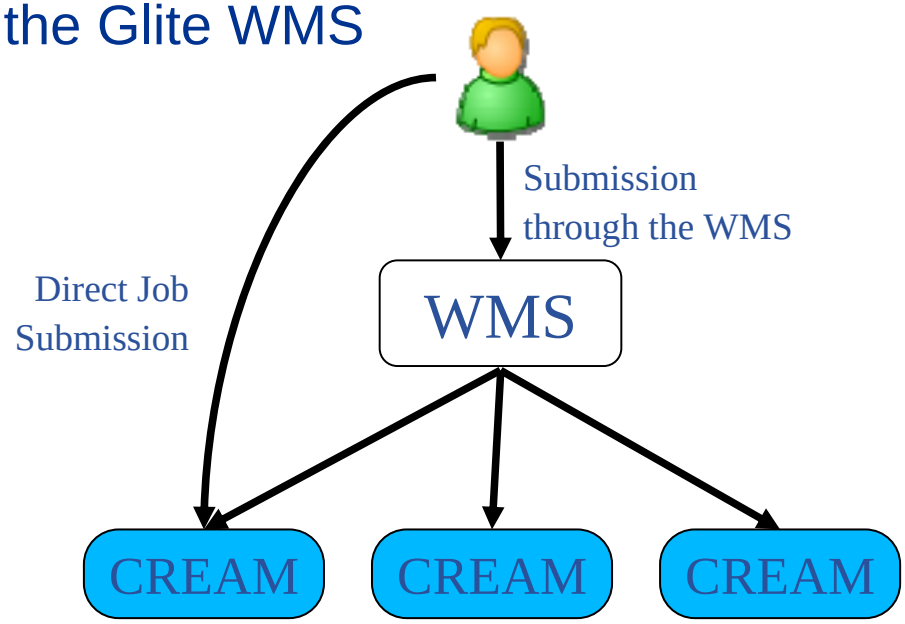
Massimo Sgaravatto
INFN Padova
On behalf of the JRA1 IT-CZ Padova group

CREAM

- **CREAM service: Computing Resource Execution And Management service**
- **Simple, lightweight service for job management operation at the Computing Element (CE) level**
- **Web service interface**
  - WS-I compliant
- **Goals**
  - Trying to address some of the problems/missing functionality of the current implementations, based on the users and admins input and feedback
  - Easy support and maintenance
  - Trying to stick to emerging standards
    - Service oriented architecture
- **Implemented and maintained by the Padova Group of the EGEE JRA1 IT-CZ cluster**
  - Same team developing and maintaining the CEMon service

CREAM usage scenario

- **CREAM should be invoked:**
  - By a generic client (e.g. an end-user willing to interact directly via the CE)
  - Through the Glite WMS



CREAM: functionality

- **Job submission**
  - Submission of jobs to a CREAM based CE
  - Includes also staging of input sandbox files
    - Support for ‘scattered’ sandboxes, as in the WMproxy
  - Actually the operation is split in 2 operations (job register and job start) as done in the WMproxy
  - Job characteristics described via a JDL (Job Description Language) expression
    - CREAM JDL is basically the same JDL used by the Glite WMS
  - Supported job types
    - Simple, batch jobs
    - MPI jobs, as supported by the Glite WMS
      - *Open to revise the current approach if this is not considered satisfactory*
        - E.g. if always using mpirun is not considered appropriated
    - Bulk jobs (parametric jobs, job collections, as defined in Fabrizio’s presentation) planned for the next future
      - *Efficient transfer and management of the sandbox*
        - It is very usual that the jobs of the collections share some sandbox files
        - It doesn’t make sense to transfer these files multiple times

CREAM functionality

- **Proxy delegation**
  - Possibility to automatically delegate a proxy for each job submission
  - Possibility to delegate a proxy, and then using it for multiple job submissions
    - Recommended approach wrt performance, since proxy delegation can be “expensive”
  - Same approach used in WMproxy
- **Job cancellation**
  - To cancel previously submitted jobs
- **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 suspension and job resume**
  - To hold and then restart jobs

CREAM functionality

- **Job purge**
  - To clear a 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)
- **Operations planned but not yet implemented**
  - Job signal
    - To send a signal to a currently running job
  - Job assess
    - To assess how “good” is a specific CE
      - *E.g. how many/which resources on that CE are good for that job*
      - *E.g. estimated time to have the job starting its execution*
      - *To be further discussed*

CREAM interfaces

- **C++ CLI user interface**
  - Very similar and “homogeneous” with the WMproxy CLI
    - We try to stay synchronized
      - *Also at WSDL level*
- **Java client also available**
  - Implemented as requested by the EU funded GRIDCC project
    - Integrated in their portal
- **CREAM C++ API available**
  - Used by the C++ CLI and by ICE (see later)
  - We will make these APIs public soon
    - When we feel that they are stable enough
- **Of course possible to implement a “custom” user interface, using the CREAM WSDL**

CREAM: some internal details

- **Web service application**
  - Implemented in Java
  - Use of Axis
  - Use of Tomcat as application server
- **Job management requests saved on 'journal manager'**
- **Pool of threads serving in parallel the requests saved on this journal manager**
  - Number of threads is configurable
- **Interface with underlying resource management system**
  - "Hidden" by a proper "abstraction layer"
  - Implemented via BLAH
    - Already used in the current EGEE Condor based CE
  - Manages job management operations on behalf of CREAM
  - Notifies CREAM about job status changes
    - Very good performance in being able to promptly detect job status changes
  - All batch systems supported by BLAH (currently LSF and PBS/Torque) are automatically supported by CREAM

CREAM: some internal details

- **Security**
  - CREAM security architecture follows the guidelines of the Global EGEE security architecture, relying on the official tools provided by the EGEE security group (JRA3)
  - Authentication
    - PKI based infrastructure
    - X.509 certificates
  - Authorization
    - Ban/allow (grid-mapfile) PDP currently used
    - VOMS PDP (recently released) being integrated
    - Integration with G-PBOX planned
      - *For policy enforcements*
    - 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*
  - Proxy delegation
    - Using the official EGEE port delegation stuff
  - Credential mapping
    - To map Grid credential on local accounts
    - Implemented via glexec (JRA3 tool)
      - *Glexec uses LCMAPS and LCAS*

CREAM: some internal details

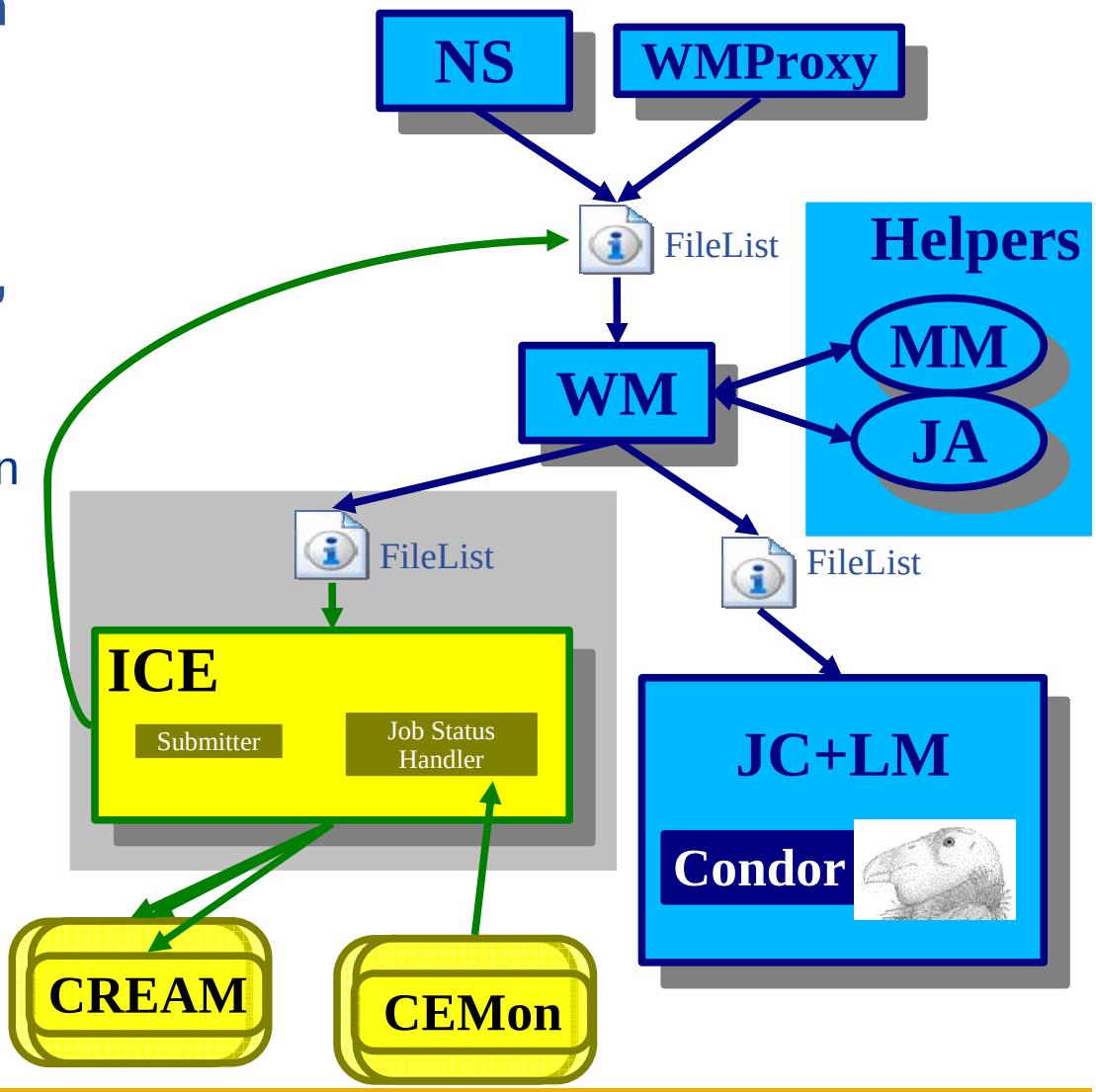
- **GridFTP server LCAS-LCMAPS enabled deployed on the CE node**
  - Used for the Input Sandbox transfer
  - Explored also DIME mechanisms but not considered satisfactory
    - Performance problems
    - Problems with interoperability with GSOAP
- **Paying attention to robustness and fault tolerance**
  - E.g. trying to be crash proof, saving persistently vital information
    - E.g. job information data (job repository) saved on permanent storage
    - E.g. job management requests to be served saved on permanent storage as well
  - E.g. trying to be resilient to BLAH parser crashes

ICE

- **ICE: Interface to Cream Environment**
- **Software component acting as an interface between the WMS and CREAM CEs**
- **Daemon running on the WMS node**
  - It will be investigated if it can be a WM thread for the future
- **Basically has the role played by JC+LM+Condor in the submission to non-CREAM CEs**
- **Isn't "ICE-CREAM integration" nice ? ☺**

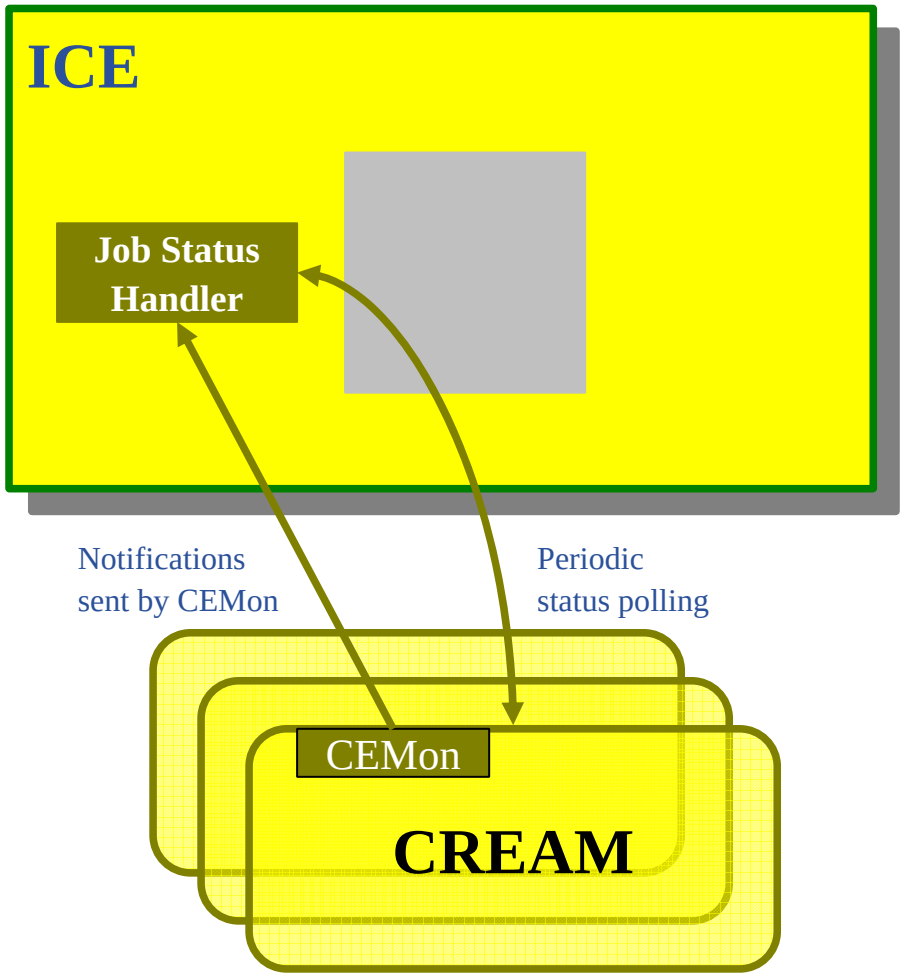
WMS-CREAM integration

- **ICE takes the job management requests from its filelist**
- **Submitter handles job submissions, job removals, proxy renewals**
  - Also job suspend and job resume when implemented in WMS
- **Failed submissions are reinserted into the WM's filelist as in the current implementation (JC+LM)**



Job Status Changes

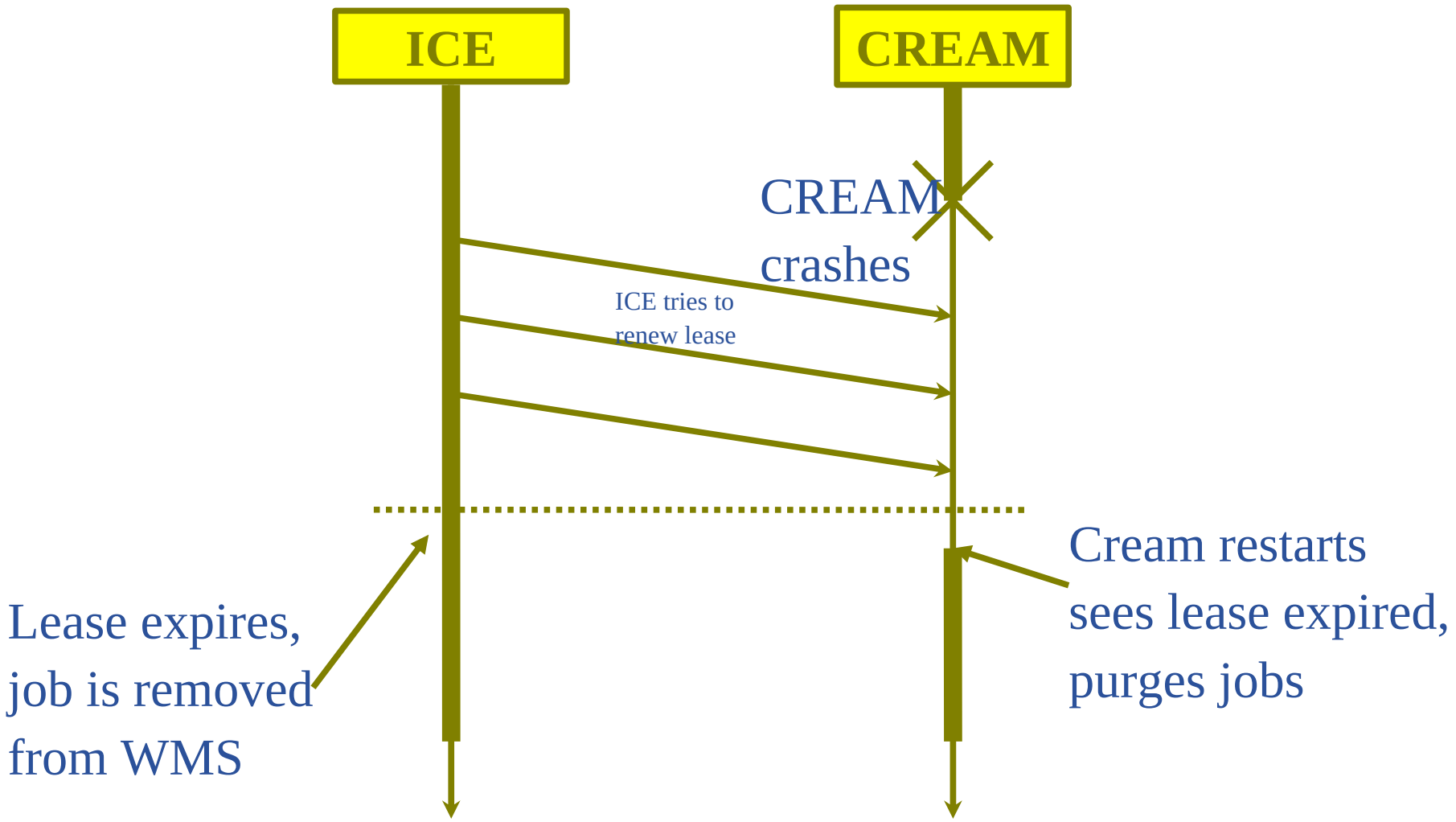
- **A thread of ICE receives notifications about the job status changes from CEMon closely coupled with CREAM CE**
  - CEMon: already used in the current Glite CE to provide CE information
- **As a fail-safe mechanism, another thread is needed to poll the status of jobs still alive**
  - If the relevant notifications are not received via CEMon



ICE: some internal details

- **Written in C++**
- **Multithreaded**
- **Fault tolerance and robustness**
  - Persistent saving of vital information
  - Reliable lease based mechanisms in the submission protocol
    - General idea
      - *Each job has an attribute (the lease) which is basically the time to live of the job*
      - *When the lease expires, the job is removed on both sides: CREAM and WMS*
      - *Leases are renewed by ICE as long as ICE and CREAM can talk to each other*
    - To handle failure scenarios and avoid to “lose” jobs (zombies)
      - *ICE failures*
      - *CREAM failures*
      - *WMS → CREAM connection failures*

Job Lease


ICE
CREAM
CREAM crashes
ICE tries to renew lease
Lease expires, job is removed from WMS
Cream restarts sees lease expired, purges jobs

Status

- **CREAM, as described (also with the described known problems/missing functionality) is ready and continues to evolve**
  - Just deployed a CREAM certification testbed where tests in a clean environment are being performed
    - See CREAM web site
  - Also stress tests and performance measures
  - Going to open this testbed to interested users in the next future
- **ICE working prototype with core functionality exists**
  - Missing functionality
    - Logging to LB
    - Lease protocol
    - Proxy renewal
  - Integrated tests with WMS needed

Some next steps

- **Complete on-going developments**
  - In particular support of bulk jobs
- **Finalize WMS-CREAM integration (ICE)**
- **Address problems raised during testing and certification process**
- **JSDL support**
  - JSDL: GGF based language to describe job characteristics
  - Emerging as standard
- **Follow interface standardization efforts**
  - Several initiatives: GGF BES, CRM initiative, Multi-Grids interoperation, OMII-Europe, …
- **Support of non homogeneous CE**
  - Using the functionality that BLAH is going to offer to pass some directives to the underlying batch systems
  - Exploring matchmaking within CREAM CE
- **Interactive access to job**
  - Interactive read-only access to a running job's environment
  - Remote `ps`, `top`, `ls`, `cat` and `tail`-like functionality on the Worker Node
  - Intelligent browsing of remote files: client-side hex viewer and `view`-like functionality only trasfers needed chunks of the remote file as needed
  - Some work already done

Other info

- **Please visit CREAM – ICE web site: http://grid.pd.infn.it/cream/field.php**
- **Software**
  - CREAM and CREAM CLI sw
  - Installation and configuration guides
  - Release notes
- **Documentation**
  - User's guide
  - JDL specification

Conclusions

- **Tried and trying to pay attention to users and admins requirements**
- **Open to suggestions and recommendations**
- **We think that some of the achieved/planned functionality can be really useful and can address some current problems**
  - E.g. bulk job submission at CE level
- **First results are really encouraging**
  - E.g. wrt performance
- **Having full control on the software without many external dependencies is facilitating the process**