L&B and Job Provenance services: What do we know about Grid job

Aleš Křenek, on behalf of CESNET JRA1 team

www.eu-egee.org

Information Society

Overview

- gLite job processing
- Logging and Bookkeeping (L&B)
  - main purpose
  - internals
- Job Provenance (JP)
  - purpose
  - architecture
  - relationship with L&B

gLite job processing

- user prepares JDL and job inputs
- job is submitted via **User Interface**
- **Workload Manager** queues the job
- suitable Computing Element is found
- **Job Controller** sends the job there
- **Log Monitor** tracks the progres, resubmitting eventually
- **Job Wrapper** is started
  - download job inputs
  - execute **job payload**
  - upload outputs
- user retrieves outputs and job is purged

gLite job processing

- user prepares JDL and job inputs
- job is submitted via **User Interface**
- **Workload Manager** queues the job
- suitable Computing Element is found
- **Job Controller** sends the job there
- **Log Monitor** tracks the progres, resubmitting eventually
- **Job Wrapper** is started
  - download job inputs
  - execute **job payload**
  - upload outputs
- user retrieves outputs and job is purged
- **L&B** records important steps from all highlighted components
- user queries L&B for job status

L&B main purpose

- track job progress independently
  - offload user queries from WMS etc.
  - allow WMS services to be stateless
  - provide external view

L&B main purpose

- track job progress independently
  - offload user queries from WMS etc.
  - allow WMS services to be stateless
  - provide external view
- make job information available at single contact point
  - no need to analyze several logs

L&B main purpose

- track job progress independently
  - offload user queries from WMS etc.
  - allow WMS services to be stateless
  - provide external view
- make job information available at single contact point
  - no need to analyze several logs
- process raw information to provide high-level view
  - too detailed for “normal” use
  - difficult to interprete due to complex job life cycle

Jobs and events

- **job** is the primary entity of interest
  - all information in L&B is attached to some job
  - job is assigned a unique **grid job identifier** (jobid)
  - jobid is passed with the job between grid components

Jobs and events

- **job** is the primary entity of interest
  - all information in L&B is attached to some job
  - job is assigned a unique **grid job identifier** (jobid)
  - jobid is passed with the job between grid components
- important points in job life generate **L&B events**
  - transfer between components, match CE, start execution
  - high redundancy
  - multiple *attribute* = *value* pairs
  - fixed schema to reflect gLite jobs
- user annotations (name = value) are events too

Jobs and events (2)

- common event attributes

timestamp	Time the event was generated.
jobId	Grid job id of the job the event belongs to.
seqcode	Sequence code assigned to the event.
user	Identity (certificate subject) of the event sender.
source	Source (software component) which generated this event.
. . .	

- specific attributes for Transfer event

destination	Destination where the job is being transfered to.
dest_host	Hostname of server that takes over control of the job.
job	Job description in receiver's language.
result	Result code of the transfer attempt (START, OK, REFUSED or FAIL).
reason	Detailed description of the transfer, especially reason of failure.
dest_jobid	Job id as assigned by the receiving software component.

Event delivery and processing

- gLite components **call L&B API** to generate events
- L&B library connects to close **local-logger**
- local-logger stores events on disk
- **inter-logger** handles delivery to **L&B server**
- users query the server

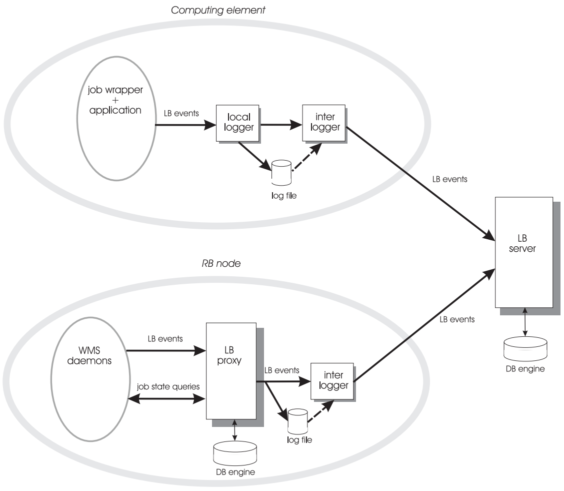
Event delivery and processing

- gLite components **call L&B API** to generate events
- L&B library connects to close **local-logger**
- local-logger stores events on disk
- **inter-logger** handles delivery to **L&B server**
- users query the server
- delivery is **asynchronous store-and-forward**
  - L&B calls are virtually non-blocking
  - high reliability and performance
  - successfully logged event may not be visible immediately

Event delivery and processing

- gLite components **call L&B API** to generate events
- L&B library connects to close **local-logger**
- local-logger stores events on disk
- **inter-logger** handles delivery to **L&B server**
- users query the server
- delivery is **asynchronous store-and-forward**
  - L&B calls are virtually non-blocking
  - high reliability and performance
  - successfully logged event may not be visible immediately
- alternate delivery path goes through **L&B proxy**
  - lightweight L&B server deployed on WMS machine for its internal use
  - synchronous local queries

Event delivery and processing (2)

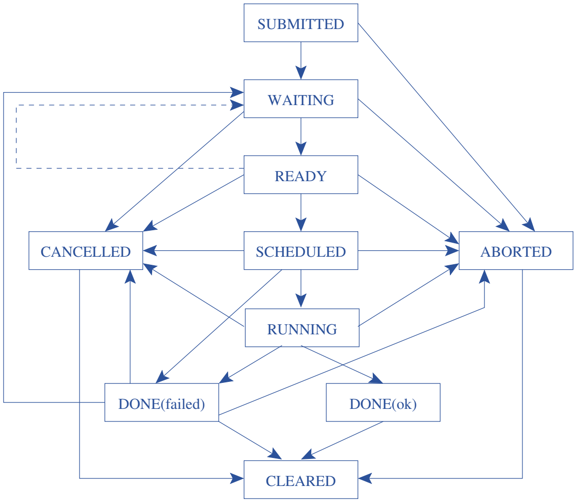


Job state

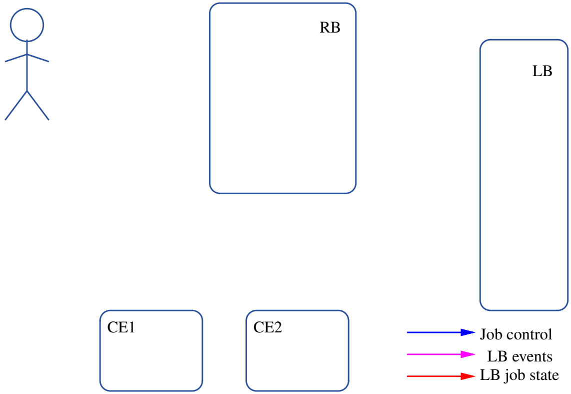
- raw events are stored at L&B server
- **job state** is computed
  - high-level view on the job
  - cope with redundancy in events
  - reflect complex gLite job processing
- job state carries multiple attributes too

owner	Job owner
state	The major state (Submitted, Waiting, ..., Running, Done, ...)
globusId	Globus allocated Id
jdl	User submitted job description
destination	ID of CE where the job is being sent
reason	Reason of being in this status, if any
done_code	Return code (OK, FAILED, CANCELED)
exit_code	Unix exit code
resubmitted	The job was resubmitted
cancelling	Cancellation request in progress

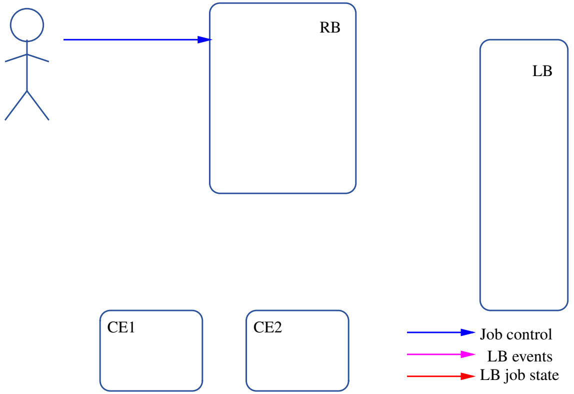
Job state transitions



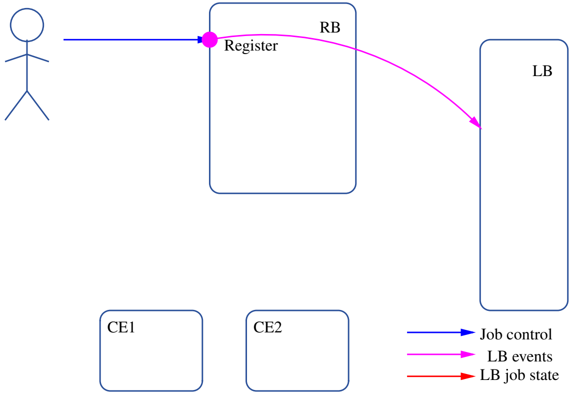
Job state computation



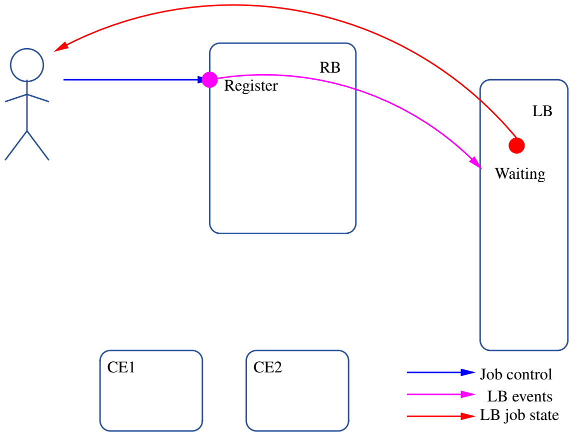
Job state computation



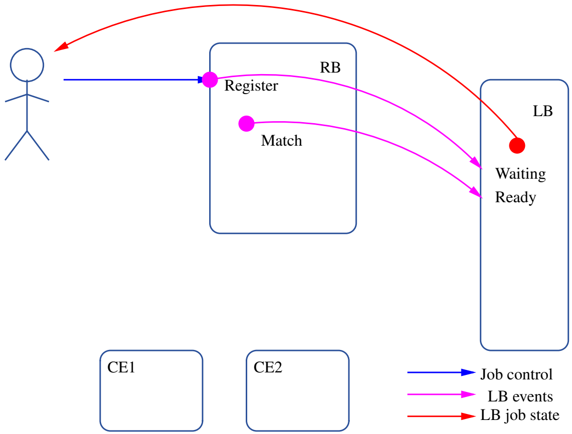
Job state computation



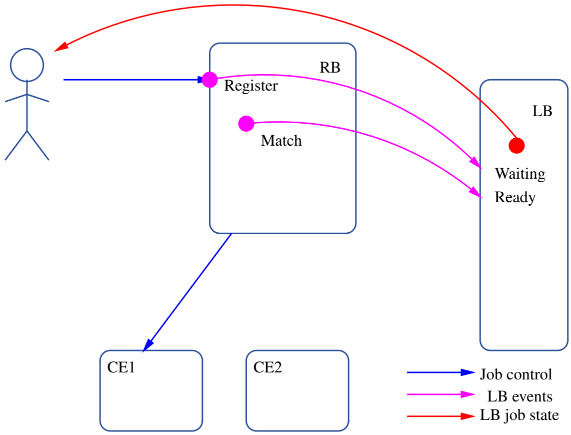
Job state computation



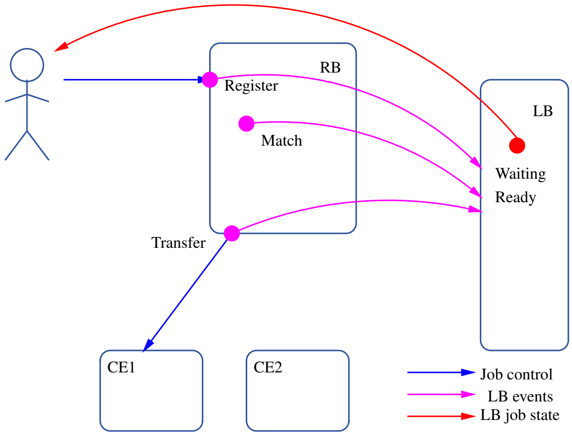
Job state computation



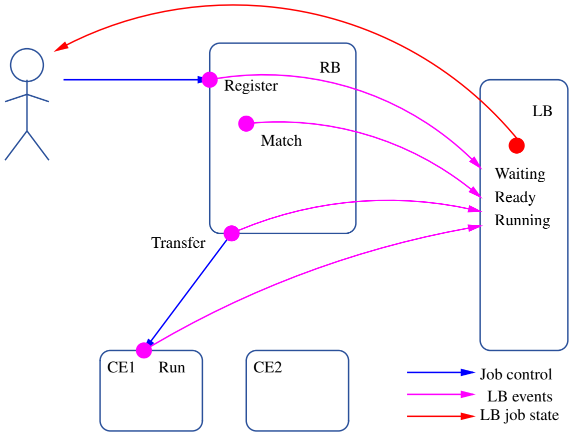
Job state computation



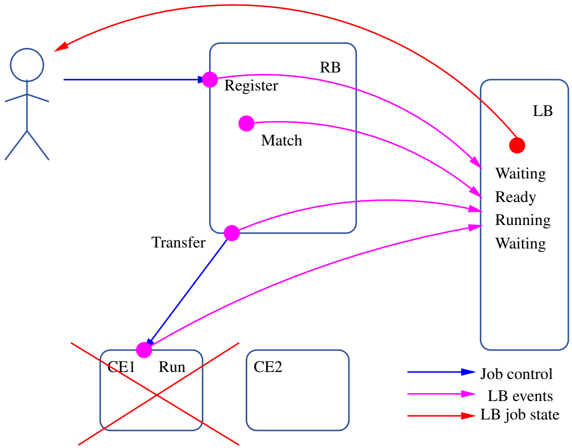
Job state computation



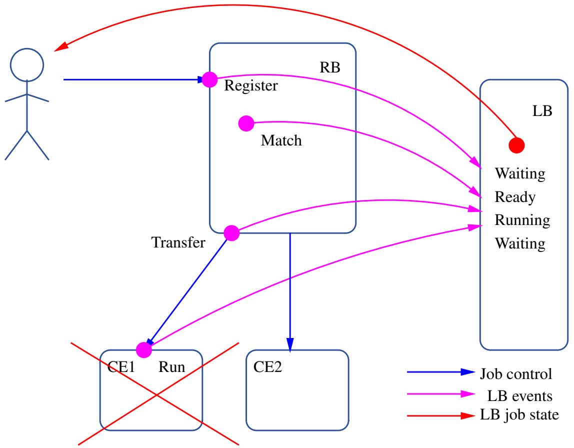
Job state computation



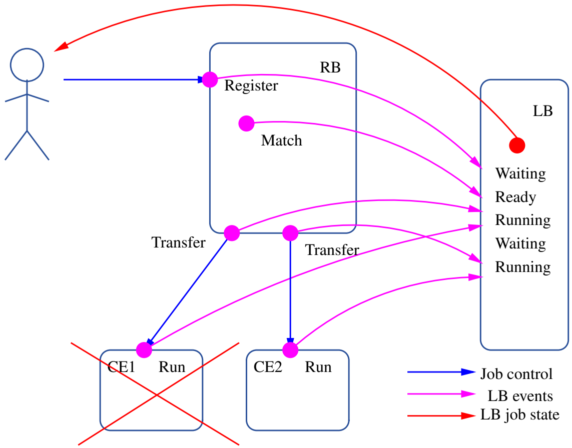
Job state computation



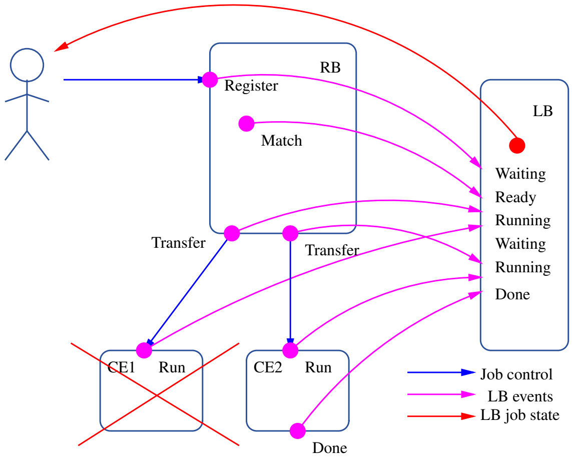
Job state computation



Job state computation



Job state computation



Job types

- Simple

Job types

- Simple
- DAG (directed acyclic graph)
  - composed of several simple jobs
  - arbitrary mutual dependencies
  - handled by Condor DAGMan in WMS
  - both parent job and subjob live their life, generating L&B events etc.
  - performance limitations

Job types

- Simple
- DAG (directed acyclic graph)
  - composed of several simple jobs
  - arbitrary mutual dependencies
  - handled by Condor DAGMan in WMS
  - both parent job and subjob live their life, generating L&B events etc.
  - performance limitations
- Collection
  - simple container for many subjobs
  - no dependencies, still shared inputs etc.
  - handled more efficiently in WMS
  - parent job disappears after break-up
  - its state is deduced by L&B from subjobs only

Event ordering

- event order is critical for computing job state
- timestamps are not reliable on the grid
  - strict clock synchronization cannot be enforced
- event counter is not sufficient
  - dead branches of job submission

Event ordering

- event order is critical for computing job state
- timestamps are not reliable on the grid
  - strict clock synchronization cannot be enforced
- event counter is not sufficient
  - dead branches of job submission
- addressed by **hierarchical sequence code**
  - `NS=4:WM=7:JSS=3:LM=6:LRMS=0`

Event ordering

- event order is critical for computing job state
- timestamps are not reliable on the grid
  - strict clock synchronization cannot be enforced
- event counter is not sufficient
  - dead branches of job submission
- addressed by **hierarchical sequence code**
  - `NS=4:WM=7:JSS=3:LM=6:LRMS=0`
- it's even worse . . .
  - shallow resubmissions allow parallel execution of submission branches
  - only one wins, but it may not be the most recent one
  - not the highest sequence code becomes the authoritative one

Event ordering

- event order is critical for computing job state
- timestamps are not reliable on the grid
  - strict clock synchronization cannot be enforced
- event counter is not sufficient
  - dead branches of job submission
- addressed by **hierarchical sequence code**
  - `NS=4:WM=7:JSS=3:LM=6:LRMS=0`
- it's even worse ...
  - shallow resubmissions allow parallel execution of submission branches
  - only one wins, but it may not be the most recent one
  - not the highest sequence code becomes the authoritative one

- *don't worry if you don't understand fully*
  - *just demonstrating complexity of the problem*

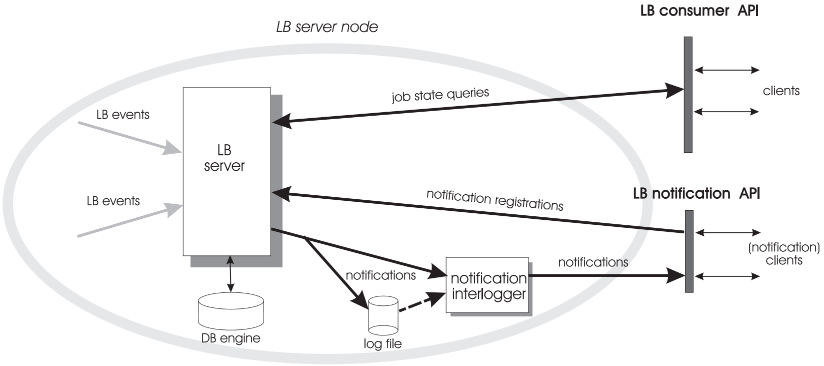
User queries

- active query (C/C++ API, WS)
  - retrieve job states or raw events
  - AND-ORed conditions on attributes
    - eg. *which of my 'apple' or 'pear' annotated jobs are scheduled for execution at CE 'garden.somewhere.org'*

User queries

- active query (C/C++ API, WS)
  - retrieve job states or raw events
  - AND-ORed conditions on attributes
    - eg. *which of my 'apple' or 'pear' annotated jobs are scheduled for execution at CE 'garden.somewhere.org'*
- notifications (C/C++ API)
  - specify similar conditions on job state
    - eg. *my job is done or aborted*
  - receive notification handle (transferable among machines)
  - block until conditions are satisfied
  - avoids unnecessary L&B server polling

User queries



Security

- all network communication is authenticated and encrypted with SSL
- user query authorization
  - job owner
  - per-job specified users and groups (VOMS)
  - privileged users (configured on L&B server)
- inter-service authorization
  - currently all authenticated services are trusted
  - improved model being discussed (VOMS, LCAS, . . .)

Some caveats

- event delivery is asynchronous
  - logged events are not visible immediately

Some caveats

- event delivery is asynchronous
  - logged events are not visible immediately
- interpretation of events is quite complex
  - presence of Done/Fail does not necessarily mean overall failure

Some caveats

- event delivery is asynchronous
  - logged events are not visible immediately
- interpretation of events is quite complex
  - presence of Done/Fail does not necessarily mean overall failure
  - presence of Done/OK does not necessarily mean job success

Some caveats

- event delivery is asynchronous
  - logged events are not visible immediately
- interpretation of events is quite complex
  - presence of Done/Fail does not necessarily mean overall failure
  - presence of Done/OK does not necessarily mean job success
  - L&B state machine implementation is the only fixed semantics :-(

Some caveats

- event delivery is asynchronous
  - logged events are not visible immediately
- interpretation of events is quite complex
  - presence of Done/Fail does not necessarily mean overall failure
  - presence of Done/OK does not necessarily mean job success
  - L&B state machine implementation is the only fixed semantics :-(
- L&B is not a permanent storage
  - jobs may disappear (be purged) after timeout
  - they migrate to Job Provenance eventually

Job Provenance – motivation

- L&B is not designed for long-term storage
  - data stored as rich structure
  - database size affects performance
  - index reconfiguration partially interferes with operation

Job Provenance – motivation

- L&B is not designed for long-term storage
  - data stored as rich structure
  - database size affects performance
  - index reconfiguration partially interferes with operation
- Job Provenance – contradictory requirements
  - compact storage of huge data
  - still allow efficient and frequent queries
  - handle changes in time
    - data format
    - query patterns

JP design

- two specialized classes of components to split responsibilities

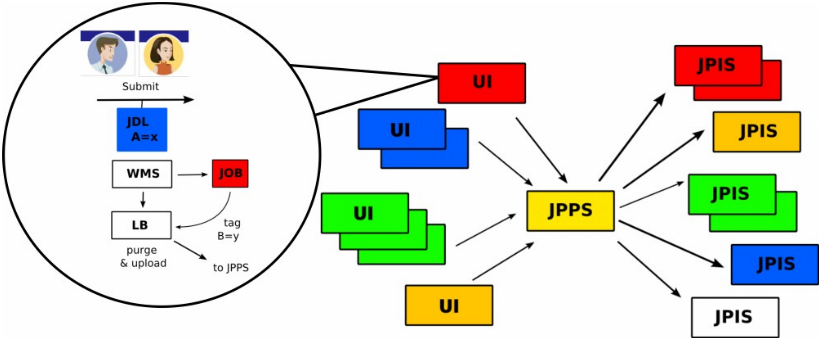
JP design

- two specialized classes of components to split responsibilities
- **JP Primary Storage**
  - keep data permanently
  - compact format – compressed files, minimal metadata
  - no restriction on format – plugins handle it
  - write-once, read-many – do not drop any information
  - don't allow direct user queries

JP design

- two specialized classes of components to split responsibilities
- **JP Primary Storage**
  - keep data permanently
  - compact format – compressed files, minimal metadata
  - no restriction on format – plugins handle it
  - write-once, read-many – do not drop any information
  - don't allow direct user queries
- **JP Index Server**
  - fraction of primary data only
  - create and populate semi-dynamically
  - "super-query" anticipating specific usage pattern
  - indices to serve user queries efficiently

JP design


Submit
JDL
A=x
WMS
JOB
LB
tag
B=y
purge
& upload
to JPPS
UI
UI
UI
UI
JPPS
JPIS
JPIS
JPIS
JPIS
JPIS

JP data representation

- job-centric approach again
  - any data must be assigned to a job

JP data representation

- job-centric approach again
  - any data must be assigned to a job
- raw form
  - small size **tags**: “name = value”, added any time
  - **bulk files**: uploaded “as is” and compressed

JP data representation

- job-centric approach again
  - any data must be assigned to a job
- raw form
  - small size **tags**: "name = value", added any time
  - **bulk files**: uploaded "as is" and compressed
- logical view
  - **JP attributes**: "namespace:name = value"
  - tags map directly
  - files are processed by **plugins**

JP data representation

- job-centric approach again
  - any data must be assigned to a job
- raw form
  - small size **tags**: "name = value", added any time
  - **bulk files**: uploaded "as is" and compressed
- logical view
  - **JP attributes**: "namespace:name = value"
  - tags map directly
  - files are processed by **plugins**
- JPPS: raw form and plugins
- JPIS and user queries: logical view only

JPIS setup

- volatile content
- configured semi-dynamically according to user needs
- JPPS's to query
- "super query"
  - superset of data assumed to be retrieved by user queries
  - list of attributes
  - conditions – which jobs to retrieve
- query mode
  - one-time – retrieve currently matching data
  - continuous – subscribe for future updates
  - combined
- set of indices
  - restrict end-user queries – index usage is required

L&B – JP interaction

- job purged from L&B after timeout (several days)
- complete data dumped in ASCII format and uploaded to JP
- L&B plugin for JPPS
  - read dumped data and sort by sequence code
  - process by copy of L&B state machine – final job state
  - provide JP attributes on job state level
- similar access to raw data is not implemented currently but feasible

JP deployment

- services running at **JRA1 preview testbed**
  - not-yet certified versions
  - maintained by developers
  - semi-production operation
  - user feedback welcome, fast reaction

JP deployment

- services running at **JRA1 preview testbed**
  - not-yet certified versions
  - maintained by developers
  - semi-production operation
  - user feedback welcome, fast reaction
- two major application usecases (demos)
  - molecular docking study
  - Atlas experiment
- see poster #30 for details

Summary

- L&B gathers many details on grid job life
  - widely deployed in EGEE production
  - correct interpretation is tricky due to complex job life
  - not designed for direct data-mining
- JP archives job information for long time
  - full L&B information is preserved
  - stored for long time
  - more suitable for data-mining
  - available at Preview testbed

References

- L&B and JP overview
  - F. Dvořák et al., Services for Tracking and Archival of Grid Job Information, Proc. Cracow Grid Workshop'05, 255–263, 2006.
  - F. Dvořák et al., gLite Job Provenance, Proc. IPAW'06, LNCS 4145, 246–253, 2006.
- Detailed technical report on the services
  - L. Matyska et al., Job Tracking on a Grid—the Logging and Bookkeeping and Job Provenance Services, CESNET technical report, 2007, soon at `http://www.cesnet.cz/doc/techzpravy/`.
- Papers on JP demos
  - A. Křenek et al., Experimental Evaluation of Job Provenance in ATLAS Environment, Proc. CHEP'07, J. Physics: Conf. Series, 2007, accepted.
  - A. Křenek et al., Multiple Ligand Trajectory Docking Study—Semiautomatic Analysis of Molecular Dynamics Simulations using EGEE gLite Services, PDP 2008, accepted.
- JRA1 preview testbed
  - `https://twiki.cern.ch/twiki/bin/view/EGEE/EGEEgLitePreviewTestBedComposition`