Enabling Grids for E-science in Europe

www.eu-egee.org

IT/CZ cluster meeting, Milano, July 20, 2004

L&B and JP WebService design

CESNET team

EGEE is a project funded by the European Union under contract IST-2003-508833

L&B WS Interfaces

- L&B queries
  - simple “job state” of a single job – working prototype, gsoap server, both gsoap and axis (java) client
  - full-featured “jobs” and “events” queries – in progress
- logging into L&B and notifications (Holub) – not addressed now, postponed to the outcome of L&B-RGMA convergence

Job State WS Query

Simple case – single job, known JobId

```
<operation name="JobStatus">
  <input message="JobStatusRequest"/>
  <output message="JobStatusResponse"/>
</operation>

<message name="JobStatusRequest">
 <part name="jobid" element="edgwll:jobid"/>
 <part name="flags" element="edgwll:flags"/>
</message>

<message name="JobStatusResponse">
 <part name="status" element="edgwll:status"/>
 <part name="errCode" element="edgwll:errCode"/>
 <part name="errDesc" element="edgwll:errDesc"/>
</message>
```

Job State WS Query (2)

```
<complexType name="JobStat">
  <sequence>
    <element name="state" type="edgwll:JobStatCode"
       minOccurs="1" maxOccurs="1"/>
    <element name="jobId" type="xsd:string"
       minOccurs="0" maxOccurs="1" nillable="true"/>
    <element name="owner" type="xsd:string"
       minOccurs="0" maxOccurs="1" nillable="true"/>
    <element name="jobtype" type="edgwll:StatJobtype"
       minOccurs="1" maxOccurs="1"/>
    ...
    <element name="children-states" type="edgwll:JobStat"
       minOccurs="0" maxOccurs="1" nillable="true"/>
    ...
  </sequence>
</complexType>

<simpleType name="JobStatCode">
  <restriction base="xsd:string">
    <enumeration value="EDGWLL:JOB-UNDEF"/>
    <enumeration value="EDGWLL:JOB-SUBMITTED"/>
    <enumeration value="EDGWLL:JOB-WAITING"/>
    <enumeration value="EDGWLL:JOB-READY"/>
    <enumeration value="EDGWLL:JOB-SCHEDULED"/>
    ...
  </restriction>
</simpleType>
```

Implementational Issues

- steps in building the service
  - placeholder .h, modified functions prototype suitable for gSoap
  - WSDL generated from the placeholder with soapcpp2
  - client and server implementation generated from WSDL
  - hand written code of “soaped” functions to call original L&B server implementation

Implementational Issues

- steps in building the service
  - placeholder .h, modified functions prototype suitable for gSoap
  - WSDL generated from the placeholder with soapcpp2
  - client and server implementation generated from WSDL
  - hand written code of “soaped” functions to call original L&B server implementation
- extension of legacy L&B server, using gSoap transport plugin interface
- another port ($n + 2$) to serve WS requests
- internal L&B server queues and work distribution among slaves preserved

Implementational Issues

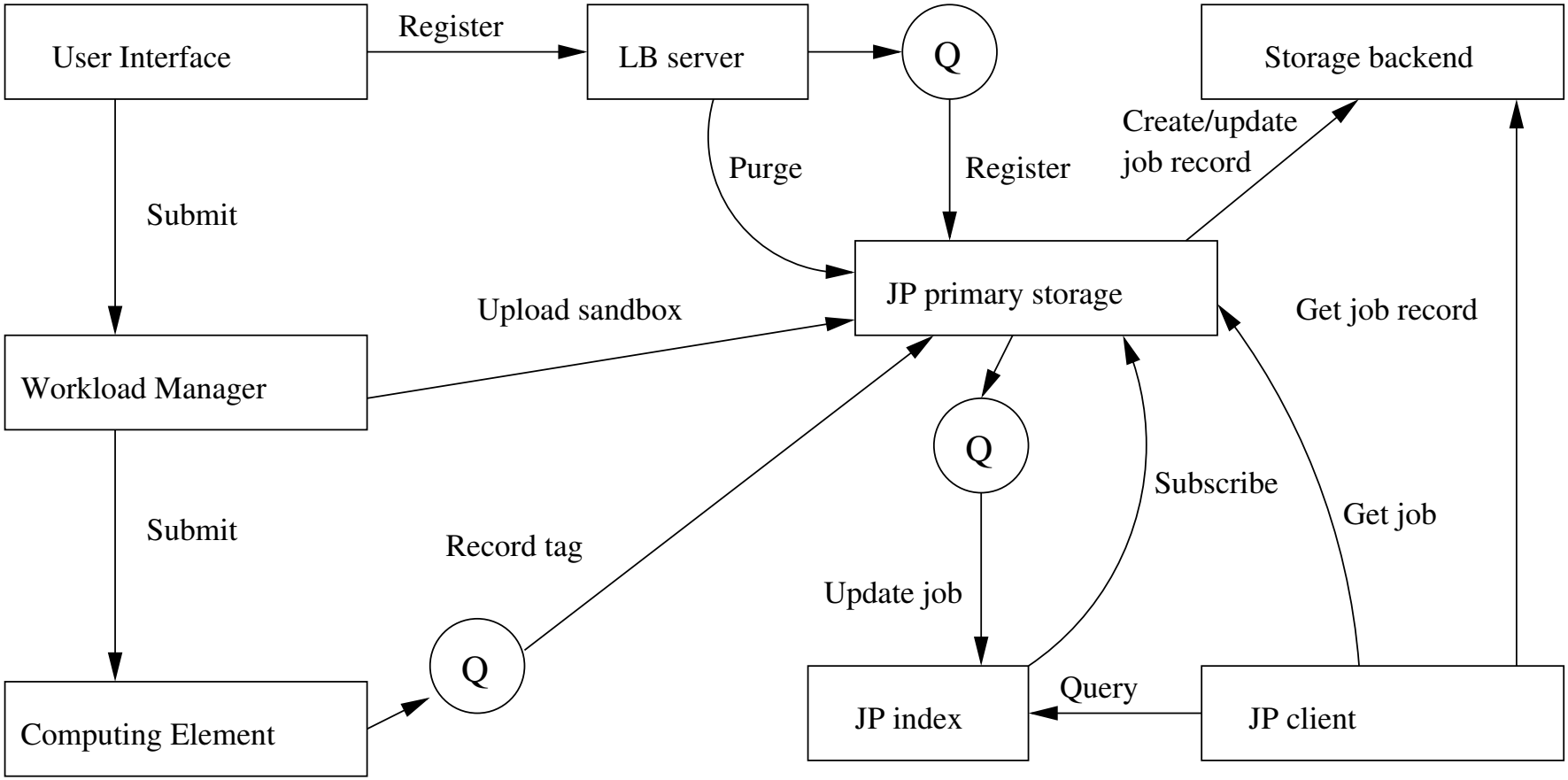
- steps in building the service
  - placeholder .h, modified functions prototype suitable for gSoap
  - WSDL generated from the placeholder with soapcpp2
  - client and server implementation generated from WSDL
  - hand written code of “soaped” functions to call original L&B server implementation
- extension of legacy L&B server, using gSoap transport plugin interface
- another port ($n + 2$) to serve WS requests
- internal L&B server queues and work distribution among slaves preserved
- security via SSL-compatible GSS-API
  - Globus proxies supported via Globus GSS implementation
  - VOMS information accessible via Globus GSS extensions (work in progress)

Job Provenance Design Overview

Enabling Grids for E-science in Europe

- work in progress:
  - high-level design ideas settled
  - being broken down into more detailed components design
  - retrofit of low-level issues to high-level design adjustments still probable
- comments & questions welcome

JP Interaction with Middleware Components



Job registration in JP

- propagated from L&B immediately
- must not block in case of JP unavailability – interlogger-based persistent queue used
- unlike L&B not required to create job record – just best effort information flow to let JP know about the job ASAP

```
<operation name="RegisterJob">
  <input message="RegisterJobRequest"/>
  <fault message="GenericFault"/>
</operation>

<message name="RegisterJobRequest">
  <part name="jobid" type="xsd:string"/>
  <part name="owner" type="xsd:string"/>
</message>
```

Sandbox and L&B dump upload into JP

- NS (or other responsible component) declares the intention to upload

```
<operation name="StartUpload">
  <input message="StartUploadRequest"/>
  <output message="StartUploadResponse"/>
  <fault message="GenericFault"/>
</operation>

<message name="StartUploadRequest">
  <part name="class" type="UploadClass"/>
  <part name="commitTimeout" type="xsd:duration"/>
  <part name="contentType" type="xsd:string"/>
  <part name="contentLength" type="xsd:long"/>
</message>

<simpleType name="UploadClass">
  <restriction base="std:string">
    <enumeration value="INPUT-SANDBOX"/>
    <enumeration value="OUTPUT-SANDBOX"/>
    <enumeration value="JOB-LOG"/>
  </restriction>
</simpleType>

<message name="StartUploadResponse"/>
  <part name="destination" type="xsd:string"/>
  <part name="commitBefore" type="xsd:dateTime"/>
</message>
```

Sandbox and L&B dump upload into JP (2)

- packed sandbox (ZIP) is uploaded using bulk file transfer protocol (gridftp) to the `destination` URL
- upload is committed with another JP call

```
<operation name="CommitUpload">
  <input message="CommitUploadRequest"/>
  <fault message="GenericFault"/>
</operation>

<message name="CommitUploadRequest">
  <part name="destination" type="xsd:string"/>
</message>
```

Record JP Tag

- analogy to L&B user tags
- also binary data supported (cannot be indexed)
- “override” L&B tags but still are distinguishable

```
<operation name="RecordTag">
  <input message="RecordTagRequest"/>
  <fault message="GenericFault"/>
</operation>

<message name="RecordTagRequest">
  <part name="jobid" type="xsd:string"/>
  <part name="tag" type="TagValue"/>
</message>

<complexType name="TagValue">
  <sequence>
    <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="sequence" type="xsd:int"/ minOccurs="0" maxOccurs="1">
    <element name="timestamp" type="xsd:dateTime"/ minOccurs="0" maxOccurs="1">
    <element name="stringValue" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <element name="blobValue" type="xsd:base64Binary" minOccurs="0" maxOccurs="1
  </sequence>
</complexType>
```

Querying and retrieval of information from JP

Enabling Grids for E-science in Europe

- If JobId is known query the primary storage directly

```
<operation name="GetJob">
  <input message="GetJobRequest"/>
  <output message="GetJobResponse"/>
  <fault message="GenericFault"/>
</operation>

<message name="GetJobRequest">
  <part name="jobid" type="xsd:string"/>
</message>

<message name="GetJobResponse">
  <part name="jobLog" type="xsd:string"/>
  <part name="inputSandbox" type="xsd:string"/>
  <part name="outputSandbox" type="xsd:string"/>
  <part name="tags" type="xsd:string"/>
</message>
```

- the response contains URL's to job record parts, the client downloads what is required from the storage backend

Querying and retrieval (2)

Enabling Grids for E-science in Europe

- JobId is not known – query suitable index server
- each index server is configured with a set of queryable and indexed attributes
  - L&B attributes (job owner, submission time, terminal state, . . .)
  - JP attributes (sandbox file names, . . .)
  - tags (both L&B and JP values)
- query is similar to L&B query,
  - must contain only only queryable attributes
  - at least one must be indexed
- server responds with a list of matchin JobId's and URLs to download from storage backend

Feeding JP index servers

- many-to-many relationship index server – primary storage
- two feed modes
  - batch – all matching jobs are retrieved
  - incremental – only newly created/modified jobs are sent

Feeding JP index servers

- many-to-many relationship index server – primary storage
- two feed modes
  - batch – all matching jobs are retrieved
  - incremental – only newly created/modified jobs are sent
- in either case, index server issues a query

```
<operation name="FeedIndex">
  <input message="FeedIndexRequest"/>
  <output message="FeedIndexResponse"/>
  <fault message="GenericFault"/>
</operation>

<message name="FeedIndexRequest">
  <part name="destination" type="xsd:string"/>
  <part name="attributes" element="Attributes"/>
  <part name="conditions" element="PrimaryQuery"/>
  <part name="continuous" type="xsd:boolean"/>
</message>

<message name="FeedIndexResponse"/>
  <part name="feedId" type="xsd:string"/>
  <part name="expires" type="xsd:dateTime"/>
</message>
```

Feeding JP index servers (2)

- primary storage starts feeding the index
- batch feeds go directly, in chunks of reasonable size for performance reasons
- incremental feeds are handled by persistent queue (interlogger) to cope with index server temporary unavailability

```
<operation name="UpdateJobs">
  <input message="UpdateJobsRequest"/>
  <fault message="GenericFault"/>
</operation>

<message name="UpdateJobsRequest">
  <part name="feedId" type="xsd:string"/>
  <part name="data" element="UpdateJobsData"/>
  <part name="feedDone" type="xsd:boolean"/>
</message>

<element name="UpdateJobsData" type="UpdateJobData" minOccurs="1"/>
<complexType name="UpdateJobData">
  <part name="jobid" type="xsd:string"/>
  <part name="attributes" element="AttrValues"/>
  <part name="tags" element="Tags"/>
</complexType>
```