jAliEn status and plans

Vova, Miguel, Costin
AliEn communication

Central services

- Brokers (jobs, transf.)
- Optimizers (quota, prio)
- Authen (all else)

Catalogue
TQ
Transfers
LDAP

VOBox Services

- SOAP

Site services

- Worker node
- JobAgent
- ROOT on WN slot
- ROOT on User Machine
- Xrootd protocol
- Token authentication

Worker node

- SOAP proxy certificate
- O(20K)

Job APIs

- Xrootd protocol
- Token authentication
- O(10)

User APIs

- Proxy certificate
- aliensh
- O(10)
### jAliEn

**Central services**
- Catalogue
- TQ
- Transfers
- LDAP

**Site services**
- jSite
- JobAgent
  - :jBox
- SSL (Compressed (Java serialized object stream))
- WebsocketS, JSON serialization of requests/replies

**Worker node**
- ROOT on WN slot
- ROOT on User Machine

**Default uplink**

**Optional uplink**

**O(10)**
- jCentral

**O(100)**
- jCentral to jSite

**O(1000)**
- JobAgent to jSite

**O(1000)**
- JobAgent to :jBox

**O(10)**
- jBox to jsh
Flexible deployment

Same codebase, each level multiplexes connections and caches objects

Dual personality servers
  Java binary serialization + SSL and compression
  Efficient channel for inter-service communication
  Asynchronous messages passed between endpoints

Websockets + SSL
  End-clients (ROOT, custom clients)
Both are long-lived, persistent connections
Scenarios

Jobs connect to the JobAgent-provided socket
  or the VoBox service directly, or jCentral...
Option to scale horizontally the site services (for large sites)

Users have several alternatives
  Use full Grid certificate in ROOT to connect centrally (ask for password at every connection)
  Same with proxy certificates (1 password/proxy generation)
  Run jBox to act like ssh-agent, asking the password ~1/reboot
Early adopters

git clone https://gitlab.cern.ch/jalien/jalien.git
cd jalien
./jalien setup
./compile.sh
./jalien

... > cp -T 32 output/*/*.zip file:/tmp/

A much faster way to download the internet (multiple threads, skipping existing files to resume/retry)
Websockets provide full-duplex communication channel over a single TCP connection
Persistent channel, suitable for heavy load, low latency applications
ROOT implementation: based on libwebsockets, an open source library available in all popular linux distributions
Secure connections based on OpenSSL
Deployment options

Embedded Tomcat server providing the websocketS server endpoint
  fixed port no. for central services
  dynamic port no. for WN/user desktop instances

ROOT plugin loads identity and server addr:
  from environment (child process of JobAgent)
  from $TMPDIR/jalien_token_<uid> (user desktop)
  default locations (~/.globus/user{cert, key}.pem and alice-jcentral.cern.ch:443, for standalone ROOT instances)
Client connections

Similar ROOT call to the current one

TGrid *jalien = TGrid::Connect("jalien");

New clients can be built in any language

wss://<host>:<port>/websocket/json

wss:// = Secure WebSocket protocol
JSON for serialization

JSON is generic, compact and easy enough to work with.

ROOT plugin uses JSON-C to compose and decode the messages from upstream.

In Java we use json-simple.

Any other language has an easy to use JSON library to use.
JobAgent for Titan

Full stack, from BQ submission to job execution
Specialized version of the jAliEn JobAgent
Tackling network-less worker nodes
Filesystem-based communication channel between WNs and interactive nodes

Implements all steps of the job execution:
- Download input files, prepare sandbox, setup the environment
- Execute the actual job, monitor resource usage
- Upload job output files, clean up
Caching service for AliEn

Share cached data across all central AliEn and API services
- `find` command results - OCDB searches
- `access/whereis` - file locations
- LFN tree configuration
- JobToken generator
- User groups, database locations...

A Tomcat-based solution serving 5-10kHz of requests
# Site services

<table>
<thead>
<tr>
<th>VoBox (CE)</th>
<th>WN (JobAgent)</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Done</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logger redirection</td>
<td>Logger redirection</td>
<td>JobManager and JobBroker calls</td>
</tr>
<tr>
<td>LDAP tree configuration</td>
<td>Job matching</td>
<td></td>
</tr>
<tr>
<td>Startup script</td>
<td>CVMFS and env setup</td>
<td></td>
</tr>
<tr>
<td>CE slots retrieval</td>
<td>Payload execution and control</td>
<td></td>
</tr>
<tr>
<td>Number jobs matching</td>
<td>Monitoring info (process info: cpu, mem, disk, same for control)</td>
<td></td>
</tr>
<tr>
<td>Monitoring info</td>
<td>Configuration through VM env (instead of LDAP in AliEn)</td>
<td></td>
</tr>
<tr>
<td><strong>To-Do</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify proxy utils (timeleft,</td>
<td>MessagesMaster (msgs for JAs: e.g. <em>kill payload</em>)</td>
<td>Embedded</td>
</tr>
<tr>
<td>re-create…)</td>
<td>Spy on job execution</td>
<td>Startup scripts for services</td>
</tr>
<tr>
<td>Batch interfaces</td>
<td></td>
<td>Full-scale tests</td>
</tr>
</tbody>
</table>


Optimizers

GUID tables
  Automatically switch the catalogue to a new GUID table when content grows too much in the active one

LFN tables
  Send alarms when parts of the tree grow to more than 50M rows
  Still splitting the tree manually but this too could be automatized

Gradually plan to move the rest
  Relying on Java thread pools to do the heavy lifting
  Still need the ability to monitor and control them individually
Data movers

Transfer agents in production
Handling all bulk data operations

- Raw data migration to T1s
- Storage decommissioning / recovery
- Physical removal of SE files
- SE crawler / dark data cleanup
Misc points

New catalogue (Cassandra-based) implemented only in the new framework

Drops the proprietary Xrootd protocol between services
  Xrootd client as command line tools (v4+ syntax ready)
  Standard protocols lower the bar for writing new clients

All services are IPv6-ready

Simplify the firewall requirements (only upstream connections)
  The only exception is the bandwidth/traceroute test
Trello dashboard - link
Last slide