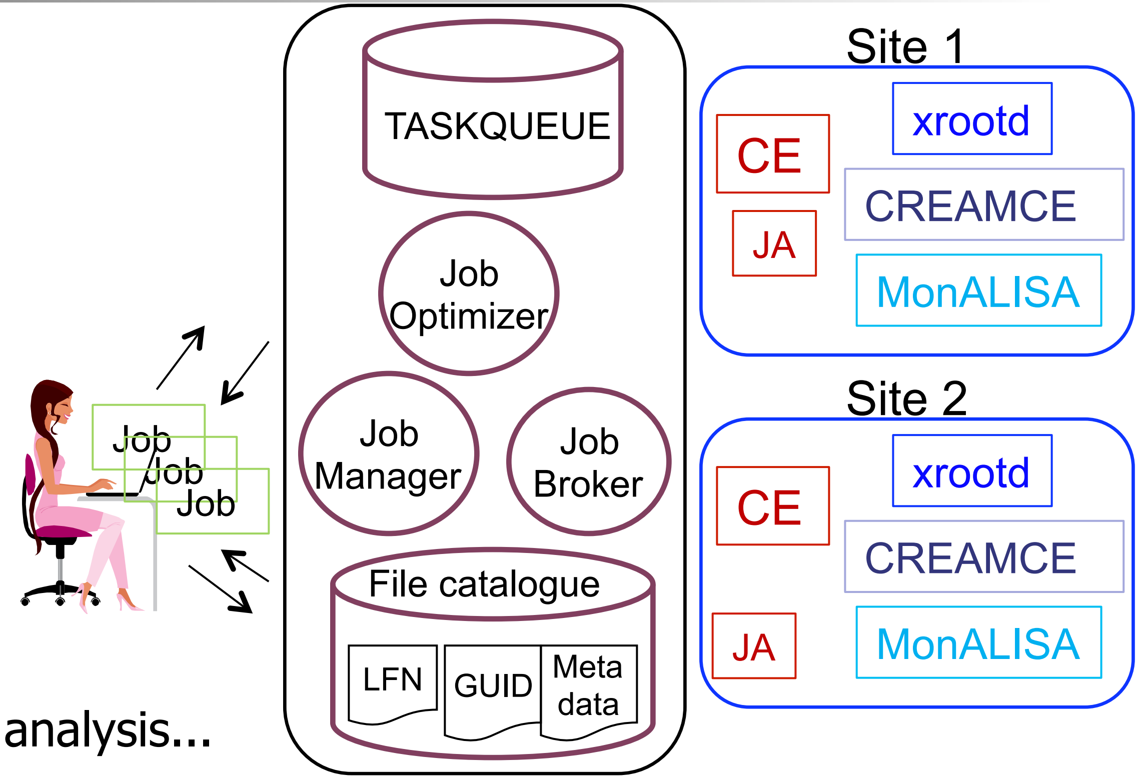
jAlien development report

P. Svirin, A. Grigoras, C. Grigoras, M. Pedreira

ALICE Offline Week – July 2015

AliEn2 – main components

- File Catalogue
  - UNIX-like file system
  - Mapping to physical files
  - Metadata information
  - SE discovery
- Transfer Model
  - With different plugins
- TaskQueue
  - Job Agent & pull model
  - Simulation, reconstruction, analysis...



Perl, C/C++, Bash scripting
~ 150 packages to maintain

JAliEn status

- New implementation used for all central workflow management tasks (productions/data registration/data management/LEGO trains/….)
- Gradually this framework grew in a full implementation of the AliEn objects and interactions with the central databases
  - File Catalogue: LFN, GUID, PFN, SE, booking table
  - Job, JDL and TaskQueue interactions
  - Users, quotas
    - gLite packages for proxy-based authentication
    - Support for signed JDLs by all parties involved
  - Access envelope generation, storage management, transfer methods
    - Parallel replica uploads and downloads for efficient end-user interaction with the federated SE space
  - and the rest of supporting objects and methods in AliEn

JAliEn, key differences

- More efficient communication infrastructure
  - Persistent, compressed, SSL channels
  - Exchanging Java serialized objects
  - Multi-layered channel multiplexing and object caching
  - Logging and error propagation
- No tokens for authentication, simply use the Grid certificate
- Better real-time monitoring of all components
- Platform-independent, easily maintainable code

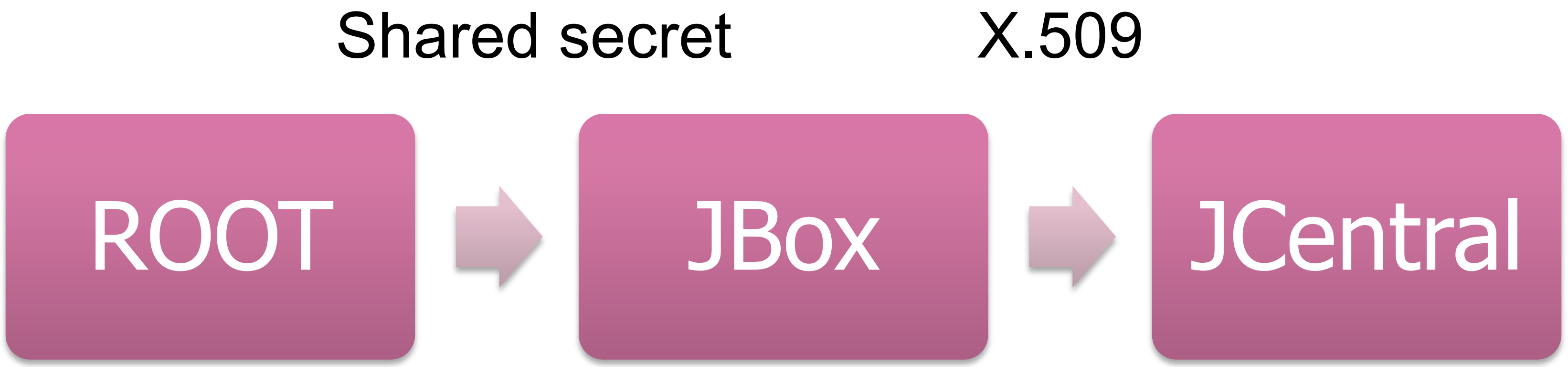
Components

- JCentral – one service processing generic "Request" objects
- JSite – one or more levels of multiplexing
- JBox
  - Persistent user-level daemon/service
  - X.509 grid certificate auth (no tokens)
  - ALiEn interface to the localhost ROOT sessions/processes
- JSh – command-line shell

JAliEn improvements

- Security libraries upgraded
- Support for package dependencies
- Improvements for some commands for sake of performance
- Xrootd4+ support (native 3rd party copy +CERN gateways support)
- Commands ported from ALiEn
- REST interface introduced

Authentication methods



JAliEn commands

- 188 commands present in AliEn, around 60 were identified to be necessary in JSh
- 13 new commands added to JSh (fquota,jquota,chmod,mirror, deleteMirror,kill), now total – 45
- 33 commands were added to JAliEn-ROOT plugin

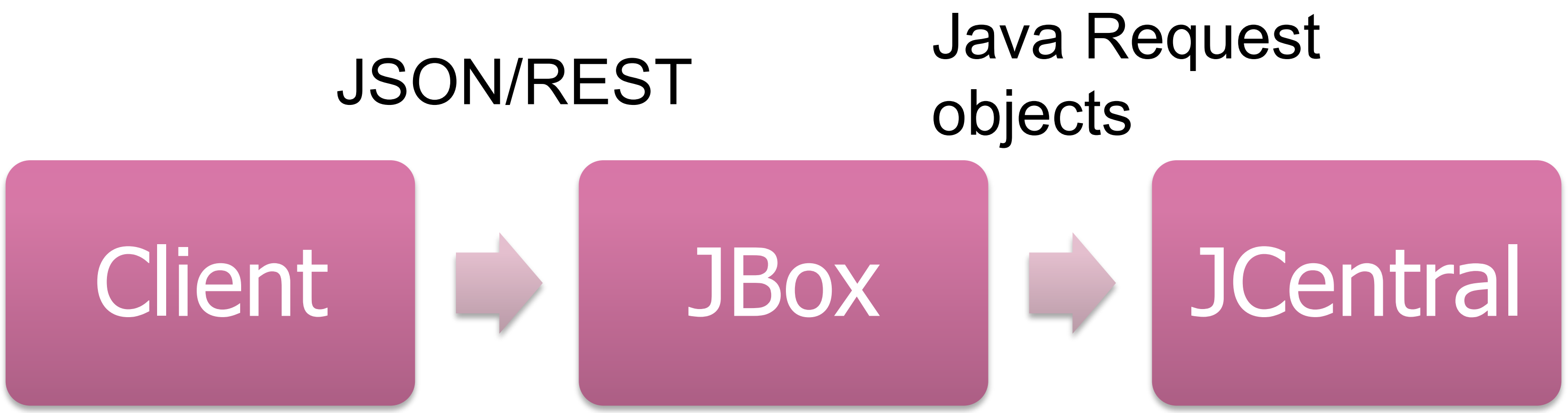
REST API for JAliEn

Shifting for an old transmission format Plain XML-over-TCP to HTTP/JSON calls between client and JBox

Advantages:

- following HTTP standards
- it is possible to use HTTP features like redirects, error codes, etc.
- we use standard libraries (libcurl/libjson-c)
- possible to easily implement clients in any language

Protocols



REST API for JAliEn

The following controllers will be created for the commands:

- catalogue
- user
- jobs
- transfers
- se
- packages

REST API for JAliEn

ls: GET /catalogue/<LFN>/ls

touch: PUT /catalogue/<LFN>/touch

rm -r: DELETE /catalogue/<LFN>?recursive=true

mv: POST /catalogue/<LFN>/mv

REST API for JAliEn

ls: GET /catalogue/%2Fetc%2Fpasswd/ls

touch: PUT /catalogue/%2Fetc%2Fpasswd/touch

rm -r: DELETE /catalogue/%2Fetc%2Fpasswd?recursive=true

mv: POST /catalogue/%2Fetc%2Fpasswd/mv

REST API for JAliEn-ROOT

```
root [3] TGrid *alien = TGrid::Connect("jalienjson://localhost:8000/");
....
root [2] alien->ls("filename", "", false);
Info in <TJAliEnJson::Final URL to be used: %s>: http://localhost:8000/catalogue/filename/ls
Info in <TJAliEnJson::Data received: %s>: {"exitcode": 0, "results": [{"owner": "agrigora", "name": "/alice/users/a/agrigora/test.jdl", "permissions": "600"}, {"owner": "grigoras", "name": "/alice/users/a/agrigora/root.jdl", "permissions": "777"}], "exitmessage": "All OK", "metaInfo": {"pwd": "/alice/users/a/agrigora", "role": "admin", "port": "8080", "user": "agrigora"}}
```

REST API for JAliEn-ROOT

```
Info in <TJAliEnJson::Line ===============>:
Info in <TJAliEnJson::key=%s>: owner
Info in <TJAliEnJson::val=%s>: agrigora
Info in <TJAliEnJson::key=%s>: name
Info in <TJAliEnJson::val=%s>: /alice/users/a/agrigora/test.jdl
Info in <TJAliEnJson::key=%s>: permissions
Info in <TJAliEnJson::val=%s>: 600
Info in <TJAliEnJson::Line ===============>:
Info in <TJAliEnJson::key=%s>: owner
Info in <TJAliEnJson::val=%s>: agrigora
………
```

Java Spring/Gradle integration

- Enhances modularity. Provides more readable code.
- Provides loose coupling between different modules.
- Effective in organizing the middle-tier applications.
- Flexible use of Dependency injection. Can be configured by XML based schema or annotation-based style.
- MVC for REST
- Different security implementations
- Gradle integration (pulls dependency packages on-the-fly,creates single JAR with all dependencies)

Summary

- Appended new commands to JCentral, more to come
- HTTP requests introduced into JAliEn-ROOT plugin
- Many options for commands have to added to REST requests
- Integrate Xrootd as command line tool used by JAliEn, which will allow to use any version
- Need to check for compatibility with REST standards
- Polishing, packaging, testing …