Reliability, load-balancing, monitoring and all that: deployment aspects of UNICORE

Bernd Schuller
UNICORE Summit 2016
June 23, 2016
Outline

- Clustering – recent progress
- Monitoring using RESTful APIs
- Ideas for improving and simplifying deployment
- Outlook
UNICORE

Clients
- Web
- Command line
- GUI
- API

Services
- Workflows
- Jobs
- Data Management
- Discovery

Resources
- Compute
- Storage

Security
- Policies

Federations
- Users

June 23, 2016
Clustering
Clustering - motivation

- Different types of clustering
- Fallback (master with a slave as backup)
  - Higher level of availability (software updates, crashes...)
  - Already available (with some data loss when switching)
  - Can be realised „externally“ (e.g. using DNS)
- Round-robin
  - Cluster members are fully equivalent
  - All cluster members have something to do
  - Can deal with higher load than single server
  - Ideally no loss of data when cluster member crashes

Goal of this work!
Basic UNICORE

UNICORE/X

Gateway

Attribute sources

IDB (Resource description)

Target System Interface

UFTPD

Local RMS (e.g. Torque, LL, LSF, etc.)
Clustering – goal

Gateway

Load balancing

UNICORE/X

State

XNJS

Persistence

UNICORE/X

State

Persistence

XNJS

Resources
Clustering – areas of work

- **Persistence**
  - Stores resources
  - Can be shared between UNICORE/X servers (e.g. MySQL DB)

- **State in UNICORE/X**
  - Running file transfer threads
  - Security sessions
  - Internal management information (e.g. number of resources per user)
  - Work queue in the XNJS (jobs currently being processed)
  - …?
Clustering – implementation

- UNICORE/X
- Load balancing
- State
- Hazelnest cluster
- Persistence in shared DB
- Gateway
- Resources
Load balancing

- Gateway has a built-in load balancer
  - Define a site as „multi-site“
  - Both fallback and round-robin

- Other options like *nginx* should work too
Clustering – status

- Update of clustering code using Hazelcast (← awesome!)
  - XNJS work queue
  - File transfers
- Reorganisation of internal management data

- TODO
  - Security sessions
  - BFT file transfers
Example: profile CPU usage
2 node cluster, primary/fallback, run 100 jobs

Primary

![Primary CPU usage graph]

Fallback

![Fallback CPU usage graph]
Example: profile CPU usage
2 node cluster, round-robin, run 100 jobs

Node A

Node B
Clustering – how to deploy

- UNICORE/X nodes must access the same resource(s)
  - Shared database
    - *H2 in server mode*
    - *MySQL (recommended)*
  - Hazelcast config
    - IP address and port for cluster
  - Identical config for UNICORE/X nodes
    - Services, options, etc
    - Same certificate
Monitoring
Monitoring – status

- Monitoring framework developed in EMI
  - Nagios/Icinga plugins

- Advantages
  - Very detaileded checking (applications, storages, etc)

- Disadvantages
  - Relatively complex
  - Dependency on UCC and its (unstable) output
Alternative: monitoring using RESTful APIs

- RESTful APIs cover most of UNICORE's functionality
  - Jobs, data, workflow submission and status checks
  - UFTP authentication server

- Advantages for monitoring
  - Very simple, can be implemented using Python or any other tool that can deal with HTTPS and JSON
  - Username/password authentication
Monitoring the Human Brain Project's HPC platform

- Monitoring user configured at each site (Unity, XUUDBs)
- Gateway, UNICORE/X, Workflow, Service Orchestrator, Registry, UFTP (via Auth server)

![Diagram showing monitoring setup with Icinga, REST API, UNICORE, and Unity]
Outlook – some ideas for deployment
Setup/deployment issues

- High complexity
  - Different services on different physical servers, requiring matching entries in config files
  - Manual adaptation to local BSS (queues, nodes, ...)
  - Non-intuitive format of config files (IDB, xnjs.xml, wsrflite.xml)
  - No config editor

- X.509 server certificates required for production deployments

- UNICORE/X is very large, no module system for deployment
Potential improvements ...

- „Zero-conf“: commandline based tools to simplify setup and configuration
  - Centralised config service e.g. on the gateway
  - CA for the internal services
  - Use host certificates
  - Make trusted CA certs available centrally
  - Auto-accept (or ask admin to confirm) trusted CA on first connect
- Simpler or re-organised config files? (e.g. XNJS config files)
- Lightweight deployment as docker images
- Self-testing features for the TSI
Thank you!