Federated Access to High-Performance Computing and Big Data Resources

Bernd Schuller
Federated Systems and Data division,
Jülich Supercomputing Centre

AFIN 2014, Lisbon November 17, 2014
Outline

- Jülich Supercomputing Centre
- Example use cases
- Solutions for Federated Access
  - UNICORE: services suite
  - Unity: user authentication and identity management
  - UFTP: high-performance data transfer
  - Clients
  - RESTful APIs
JUQUEEN

- IBM Blue Gene/Q
- 28 racks, 458,752 cores
- PowerPC A2 1.6 GHz,
- 16 cores per node
- 5.8 Petaflop/s peak
- 460 TByte main memory
- 5D network
JUST: Juelich Storage Cluster

- IBM-GPFS (General Parallel File System)
- 19.2 PB online storage (15.1 PB net)
- 14,296 disks, MTBF 3 disks per week

- 9.2 PB GPFS Storage System
- Native RAID
- 4,640 NL-SAS + 120 SSD

- Fileserver for
  - HPC systems: JUQUEEN, JUROPA
  - Clusters: JUDGE, JUVIS (visualisation)
  - DEEP (Dynamical Exascale Entry Platform)
  - Big Data collaborations
Tape Libraries

- Automated cartridge systems
- 45 PB (upgrade to 80 PetaByte)

- Used for
  - Backup
  - Long term archive
  - Migration of active (online) data to less expensive storage media

- 2 libraries
  - 16,600 tapes
  - 48 tape drives
Data centric view

- Central storage and archive
  - Mount filesystems
  - Data staging
- Remote compute resources
  - Data sharing, open access
- Site (HPC) systems
  - Upload/download/sync using various access technologies
- End-users
  - Upload/download/sync using various access technologies
Compute centric view

- External users, distributed computing, federations
- Remote compute resources
- Data staging
- Site (HPC) systems
- Mounted filesystems
- End-users
- Launch and manage jobs
- Central storage and archive
Application centric: Simulation Labs

Communities
- HBP
- JARA-HPC
- JÜLICH APPLICATION LAB
- NVIDIA
- EIC
- Big Data
- MONT BLANC
- IDEEP

Exascale

Co-Design

Cross-Sectional Teams
- Application Optimisation
- Math. Methods
- Parallel Performance
- Visualisation
- Kernels
- Tuning

Models

HPC Know-How

Fluid & Solid
- Neuroscience
- Plasma
- Molecular Systems
- Climate
- Biology
- Nuclear & Particle
- Ab Initio
- Terrestrial Systems
Federated Systems and Data

Focus on applications and their requirements in federated environments:

- **Data Management** investigates the data life cycle of applications and strategies, methods, tools and services required for all processing steps.
- **Data Analytics** addresses techniques and methods for analysing Big Data sets.
- **Application Support** deals directly with applications and their integration into distributed environments.
- **Federations** provide a basis for distributed environments by developing the necessary tools and services, e.g. for identity management or data processing models.
- **Standardisation** lays the foundations for the interoperability of federated computing and data infrastructures.
Two use cases from neuroscience
High-throughput brain scans – a Jülich / Univ. Düsseldorf collaboration

- Goal is to create a 3D brain atlas
- Data acquisition
  - Brain section scans (ex vivo)
    (~2000 slices, 500GB per slice → 1 PB)
  - MRT scans (in vivo)
- Processing: image registration, calibration, segmentation, etc
- Image processing using HPC
- Raw data often re-processed (new algorithms, new software versions)
- **Plus**: workflows, metadata, sharing with external partners
SAY BIG DATA

ONE MORE TIME
Human Brain Project

- FET Flagship
- ~10 years, ~1 Billion € (50% EC funding)
- Coordinated by EPFL (Lausanne)
- Huge, multidisciplinary Consortium
  - Neuroscience, medicine, physics, IT, philosophy, ...
  - ~200 partners by Y5
- www.humanbrainproject.eu
HBP Goal

To build an integrated ICT infrastructure enabling a

Global collaborative effort towards understanding the human brain, and ultimately

To emulate its computational capabilities
HBP High performance computing platform

Technology evaluation and deployment of HPC systems

Main production system at Jülich (Exascale capability around 2021/22) plus facilities at CSCS, BSC, CINECA

Applications requirements analysis, subcontracting for R&D and prototypes

Mathematical methods, programming models and tools

Parallel and distributed programming models, work flows, middleware for resource management, performance analysis & prediction, numerical algorithms for neuroscience

Interactive visualization, analysis and control

In-situ visualization and interactive steering and analysis of simulations

Exascale data management

Scalable querying of datasets, data analytics, data provenance and preservation

Brain-inspired supercomputing
Solutions
UNICORE

Clients
- Web
- Command line
- GUI
- API

Services
- Workflows
- Jobs
- Data Management
- Discovery

Resources
- Compute
- Storage

Federations

Policies

Security

Users
Local batch system LoadLeveller

- Login/Password
- qsub, qstat, mpirun, ...
- /usr/local/apps/myapp/bin/myapp, ...
- ~/mydata/2011/job123/ergebnisse.txt, ...
How can I ... 

- ... use multiple, heterogeneous systems seamlessly, 
- ... manage my job input data and results?
- ... across systems? Workflows?
- This was the original motivation for developing UNICORE (1997)
A federation software suite

- Secure and seamless access to compute and data resources
- Excellent application and workflow support
- Complies with typical HPC centre policies
- Wide variety of clients: GUI, commandline, APIs, ...
- Java/Perl based, supports UNIX, MacOS, Windows and many resource management systems (Torque, Slurm, SGE, …)
- Easy to install, configure, administrate and monitor
- Small, active developer team, responsive to user wishes :-)

- Open source, BSD licensed, visit http://www.unicore.eu
A (subjective) UNICORE timeline

- **1996** (mythical past) : first UNICORE project (Germany only)

- **2002** : UNICORE 4/5 → Eurogrid project, UNICORE goes Open Source, I started to work on the OpenMolGRID project

- **2005-2007**
  - UniGRIDS project : UNICORE WS(RF) interfaces defined
  - UNICORE 6.0 release

- Deployment in PRACE, XSEDE and other HPC infrastructures (national Grids, e.g. PL-Grid)

- **2013** : UNICORE 7.0 release

- … and we're still going (thanks to projects and institutional funding)
UNICORE: Main services

- **Compute**
  - TargetSystemFactory
  - TargetSystem
  - JobManagement
  - Reservations

- **Storage and data**
  - StorageFactory
  - StorageManagement
  - FileTransfer
  - Metadata

- **Workflow**
  - Workflow enactment
  - Task execution
  - Resource Broker

- **Registry**
Default setup

- Access to resource manager and file system via TargetSystemInterface (TSI) daemon installed on the cluster login node(s)
Factory services: virtualisation support

- Can add new types of TargetSystems, e.g. to set up a virtual image during its initialisation phase
- Provide access to the newly started virtual machine
Storage Management Service

- File system
- Apache HDFS
- S3 (under test)
- iRODS (prototype)

Client

mkdir, ls, rm, stat, ...

upload
download

server-to-server
copy
Storage Management Service

- Initiate file transfers
  - Multi-protocol support

- Metadata management
  - Schema-free, key-value
  - Indexed via Lucene, searchable

- Rule-based data processing
  - New files automatically trigger actions
  - e.g. metadata extraction, compression, etc
Factory services: virtualisation support

- Different types of storage backends can be supported
- User can select and provide required parameters

1. createSMS()
   provide parameters
   e.g. access keys

2. return SMS address

3. access backend

4. use

StorageFactory

- File system
- HDFS
- S3

Client

StorageManagement service
UNICORE: under the hood
UNICORE Services Environment

- Implemented in Java

- Based on Apache CXF (http://cxf.apache.org/)
  - Very mature and up-to-date services stack. Current version is 2.7.x, 3.x coming soon
  - SOAP web services
  - REST via JAX-RS

- Numerous other open source libraries
Federated access: security is the key
UNICORE – Basic security flow

- **User invokes a service**, i.e. makes a web service call to a UNICORE service

- **Authentication**: who is the user?
  - Results in the user's X.500 DN („CN=..., O=..., C=...“)

- **Assign attributes** to the DN
  - Standard attributes: role, Unix ID, groups, etc.
  - Custom attributes: (e.g. S3 access and secret keys)

- **Authorisation**
  - Add context: e.g. who owns the service?
  - Check local policies (XACML)

- **Allow or deny** the request
Delegation

- Allow Service to work on behalf of the user

- UNICORE solution based on SAML
  - Use chain of signed assertions
  - Trust always delegated to particular server
  - Can be validated and audited

User

1. submits job → Resource A

2. uploads results → Resource B
End-user authentication in UNICORE

- Pre-UNICORE 7: X.509 client certificates REQUIRED for end-users
- Users tend to hate them
  - All sorts of usage issues
- Lack of understanding leads to lack of security (sending keys via email etc)
- Users understand passwords
  - and it is relatively easy to teach basic security measures
Certificate-less end-user authentication

- **Goal**: no end-user certificates (not even short-lived)

- **Approach**
  - Use *signed SAML assertions*
  - Issued and signed by the trusted server (Identity Provider, IdP)
  - MANY options, e.g. support for existing SAML IdPs, federations like DFN AAI, etc
  - Flexible solution is required

- **Implications**
  - Client – server TLS is not client-authenticated any more
  - End-user cannot sign anything (no more „non-repudiation“)
Introducing Unity

- Complete **Authentication and Identity Management** solution
- Manage users and user attributes, group membership
- Developed by **ICM / Univ. of Warsaw** (PL)
- Separate product: [www.unity-idm.eu](http://www.unity-idm.eu)
- Increasing take-up: e.g. HBP

![Diagram showing the process of using Unity]

1. **User** provides credentials
2. **Unity** authenticates
3. **User** uses the service
Unity architecture
Unity admin: managing content

UNITY administration interface

Logged as: Default Administrator [entity id: 1]

Groups

- Root (/)
  - A
  - D
  - portal

Group /portal members

- Group by entities
- Show targeted identities
- Search:

<table>
<thead>
<tr>
<th>ENTITY</th>
<th>IDENTITY TYPE</th>
<th>IDENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>userName</td>
<td>demo</td>
</tr>
<tr>
<td>[3]</td>
<td>persistent</td>
<td>5c1e8334-e268-43dd-a7c7-3097bc320813</td>
</tr>
<tr>
<td>[3]</td>
<td>x500Name</td>
<td>CN=Demo User,O=UNICORE,C=EU</td>
</tr>
</tbody>
</table>

Group /portal details

- GROUPS ATTRIBUTES CLASSES
  - UNICORE portal attributes

Attributes of entity [3] in group /portal

- Effective
- Internal
- Required in bold

<table>
<thead>
<tr>
<th>ATTRIBUTES</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td><a href="mailto:test@example.com">test@example.com</a></td>
</tr>
<tr>
<td>cn</td>
<td></td>
</tr>
</tbody>
</table>
Unity admin: managing endpoints

UNITY administration interface

<table>
<thead>
<tr>
<th>Endpoints</th>
<th>Authenticators</th>
<th>Translation profiles</th>
<th>Message templates</th>
<th>Database backups</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNITY SOAP SAML service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status: ✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNITY UNICORE SOAP SAML OIDC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status: ✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type: SAMLUnicoreSoapIdP</td>
<td>Type description: SAML 2 UNICORE identity provider web endpoint</td>
<td>Paths:</td>
<td>Description: Metadata of the SAML 2 identity provider web endpoint</td>
<td></td>
</tr>
<tr>
<td>Binding: webservice-cxf2</td>
<td>Context address: /unicore-soapidp-oidc</td>
<td>Authenticators:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: oidc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNITY UNICORE SOAP SAML</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status: ✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: authentication assertion

```xml
<urn:Assertion>
  <dsig:Signature/>
  <urn:Subject>
    <urn:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">CN=Demo User,O=UNICORE,C=EU</urn:NameID>
    <urn:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches">
      <urn:SubjectConfirmationData NotOnOrAfter="2014-11-16T10:30:23.334Z"/>
    </urn:SubjectConfirmation>
  </urn:Subject>
  <urn:AttributeStatement>
    <urn:Attribute Name="cn">
      <urn:AttributeValue>Demo User</urn:AttributeValue>
    </urn:Attribute>
    <urn:Attribute Name="email">
      <urn:AttributeValue>test@example.com</urn:AttributeValue>
    </urn:Attribute>
    <urn:Attribute Name="memberOf">
      <urn:AttributeValue>/portal</urn:AttributeValue>
      <urn:AttributeValue>/</urn:AttributeValue>
    </urn:Attribute>
  </urn:AttributeStatement>
</urn:Assertion>
```
UFTP – high performance data transfer
Requirement: efficient data transfer through firewalls
Common data transfer issues

- Firewall
  - Direct connections from the outside to the login node(s) are usually not allowed
  - Statically opening ports (or worse, port ranges) is a security risk
    → *need dynamic port opening technique*

- User management
  - Authentication and authorization
  - User ID / group IDs mapping
  - External / anonymous users
Solving the firewall issue: using passive FTP to open ports

1. "PASV"
2. open 5432 for Client
3. connect to port 5432
4. close control connection
5. close 5432
FTP by itself is insecure:
- Users log in using username/password

UFTP adds a **secure control channel** which is used for additional security measures:
- Authenticate clients
- Map user ID / group IDs
- Initiate data transfers

Requires an secured „command port“ in addition to the open FTP port
UFTP components

- UFTPD server
  - Pseudo-FTP port (open in firewall) for clients
  - Local command port (SSL protected) used by Auth server
  - Run as root w/ setuid

- UFTP client
  - Authenticate
  - Connect to UFTPD
  - Send/receive data

- Auth server
  - Client authentication
  - User ID mapping
Standalone „Auth server“

- **Authentication**
  - Password check
  - sshkey check
  - Unity is supported

- **Attribute mapping**
  - uid, gid
  - QoS e.g. rate limit

- **RESTful service**

**Diagram:**
1. Authenticate
2. Check
3. Initiate client transfer. Pass secret and client IP

---

AFIN 2014, Lisbon November 17, 2014
Standalone UFTP Client

- **Authentication**
  - Username/password (HTTP basic auth)
  - sshkey incl. support for ssh-agent

- **Commands**
  - `ls` – list remote files
  - `cp` – copy file(s)
    - supports reading/writing parts of files (byte ranges)
  - `sync` – synchronize single remote/local files

- **Requirements**: Java 7

- **Available as tgz archive**
UFTP features

- Fast file transfer library similar to FTP
- Firewall friendly and secure
- Optional encryption and/or compression
- Multiple TCP streams per connection
- Fully integrated into UNICORE for data staging and client/server data movement
- Standalone client is available
- Flexible integration options (portals, …) or separate authentication server
- Implemented in Java, available as tgz, rpm, deb
UFTP - Some applications and use cases

- File transfer and data staging in UNICORE
  - Built into standard UNICORE clients
  - Java applet for the UNICORE web portal

- Standalone use (client plus separate AuthN server)
  - Secure, high-performance data upload/download

- Integrate UFTP functionality into web applications

- Planned master thesis: Data access and sharing at JSC (UFTP+AAI+HPC storage cluster)
2012: Testing UFTP on a 100 GBit/s testbed
TU Dresden – TU Freiberg

- Up to 10 GBit/sec per cluster node
- Up to 100 GBit/sec aggregated transfer rate
Single client, single server

- Up to 1.2GB/sec
- 98% of line rate
Multiple clients, single server

- Up to 8 clients
- (roughly!) parallel transfers (50GB each)
Multiple client/server pairs

- Up to 11 (roughly!) parallel transfers (50GB each)
- 12 GB/sec
- 98% of line rate

![Diagram showing multiple client/server pairs with throughput and parallel transfers](image)
UNICORE Clients

- „Rich client“ based on Eclipse
- Commandline client
- Web portal via Browser
- APIs
  - Java
  - RESTful (work in progress)
Rich client

- Building, submitting and monitoring jobs and workflows
- Integrated data and storage management
- X.509 and Unity for AuthN
- “Simple view” for novice users
- Based on the Eclipse framework
- Extensibility through plug-ins
- Installation/update mechanism for plug-ins and Application GUIs
Integrated storage management in the UNICORE Rich client Grid browser

- Create files
- Drag and drop from/to desktop environment
- Copy and paste
- Remote file editing
What is a „portal“ anyway?

Back to the 1990s?

... or „Web 2.0“?
UNICORE Portal

- Aim for a simple, easy-to-use web application
- Flexible authentication and user registration
  - support Unity
- Implementation choices
  - Java-based, VAADIN web framework
  - Use UNICORE Java APIs
UNICORE Portal – Job creation view

Job name: Script.job
Select application: Bash shell
Select version: 3.1.16
Command line arguments:

Input parameters
- DEBUG:
- VERBOSE:
- OPTIONS:
- SOURCE: input.sh

Submit
Several „list“ views, e.g. jobs, sites

- Workflow creation
- JavaScript
- Initially only simple graphs
UNICORE Portal: Data manager

Select: **local** or **remote** storage

Initiate **data movement** (direct, not via portal)
WS(RF) – in use since 2004/2005

■ Pros
  ■ Strongly typed
  ■ Messages can be validated
  ■ SOAP: headers/envelope mechanism
  ■ WS-Security, SAML well established

■ Cons
  ■ CPU intensive (XML processing, XML signatures)
  ■ Complex interface (look at a typical WSDL!)
  ■ Only Java and C# can be realistically used on the client side
RESTful – pros and cons

- **Pros**
  - Weakly coupled
  - HTTP benefits (error codes, caching, …)
  - Several authentication options (HTTP basic, OAuth, …)
  - Multiple message formats and resource representations can be used
    - JSON, XML, HTML, …
  - Clients in all languages (even *curl* or *wget*)

- **Cons**
  - No standard solution for trust delegation (yet)
RESTful APIs

- Concrete requirements from the Human Brain Project
  - Authentication via OpenID Connect
  - Simple job submission and management
  - Data movement

- REST APIs available with UNICORE 7.1
- OIDC under development, will be available in UNICORE 7.2

- Dedicated talk tomorrow!
Putting it all together: the Human Brain Project's HPC platform

1. authenticate
   returns OIDC token

2. access REST APIs
   pass OIDC Bearer token

3. OIDC Bearer token
   returns signed SAML

3.1 validate OIDC

Unified Portal

REST API

UNICORE

BSC
HPC site

CINECA
HPC site

CSCS
HPC site

JSC
HPC site

KIT
S3 storage

1. authenticate
   returns OIDC token

3. OIDC Bearer token
   returns signed SAML

3.1 validate OIDC

Unified Portal

REST API

UNICORE

BSC
HPC site

CINECA
HPC site

CSCS
HPC site

JSC
HPC site

KIT
S3 storage
Summary

Main challenges
- Concrete needs to access HPC compute and data resources through federations
- More users and more diverse usage of HPC resource
- Data sharing, open access and all that

Solutions
- UNICORE – compute and storage abstractions
- Unity – federated identity management
- UFTP – high-performance data transfer with sharing capabilities
Outlook

- Current and future trend: web-style
  - Authentication via OAuth2
  - RESTful APIs
  - Portals and science gateways
  - Data sharing
  - Maximise end-user friendliness, driven by applications

- Add/extend support for
  - Cloud resources (OpenStack, S3, EC2, …)
  - Hadoop / YARN jobs
  - Virtualised applications (Docker)
Team / Thank you

- Björn Hagemeier, Valentina Huber, André Giesler, Boris Orth, Mariya Petrova, Jedrzej Rybicki, Rajveer Saini and many others at JSC
- Krzysztof Benedyczak, Marcelina Borcz, Rafał Kluszczynski, Piotr Bała and others at ICM / Warsaw University
- Richard Grunzke and others at Technical University Dresden
- Students: Burak Bengi, Maciej Golik, Konstantine Muradov
- … many others who reported bugs, suggested features, contributed code and provided patches

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 604102 (Human Brain Project)