UNICORE in Production is easy to do

Sophia Antipolis
12. Oktober 2005

T-Systems Solutions for Research GmbH
Gert Ohme
UNICORE in Production

Easy to use

ALL YOU NEED TO KNOW IS

just …
UNICORE in Production

Easy to use

how to start your UNICORE client
UNICORE in Production

Easy to use
UNICORE in Production

Easy to use

submit your task
UNICORE in Production

Easy to use

fetch your output
UNICORE in Production

Easy to use

happily walk away with your results
UNICORE in Production

Easy to use

Thanks for listening to all of You,

which share the view of our customers 😊

UNICORE is such as easy to use !
UNICORE in Production

Easy to do: What next?

- UNICORE is easy to use (we have just seen)
- UNICORE is easy to understand
- UNICORE is easy to install
- UNICORE is easy to maintain
UNICORE in Production

Easy to do: What next?

- UNICORE is easy to use
- UNICORE is easy to understand
- UNICORE is easy to install
- UNICORE is easy to maintain
UNICORE in Production

Easy to understand

Two aspects are needed to understand most of UNICORE:

- functionality
- security
UNICORE in Production

Easy to understand: functionality

Simple UNICORE-Workflow for an Abstract Job Object (AJO)

Client sends AJO

Gateway sends valid AJOs to the NJS

NJS compares AJO-Tasks with IDB sends valid AJOs to the TSI

TSI transforms AJO-Tasks into OS-specific Commands uses OS-Batch-System uses OS-specific user-account
UNICORE in Production

Easy to understand: functionality

UNICORE Vocabulary (the 9 technical terms) at a glance

- **Client**
- **AJO** (Abstract Job Object), messages between client and server
- **Server Infrastructure**
  - Gateway to **USite** (UNICORE Site)
  - **NJS** (Network Job Supervisor) manages
    - **UUDB** (Unicore User Database)
    - **IDB** (Incarnation Database)
    - **TSI** (Target System Interfac) on a **VSite** (virtual server)
UNICORE in Production

Easy to understand : functionality

Building Blocks of UNICORE, Term 1

- Client

The User Interface.

You have already seen it.
So we can go on.
UNICORE in Production

Easy to understand: functionality

Building Blocks of UNICORE, Term 2

- **AJO** (Abstract Job Object)

  This one contains either your requests or your results.

  The UNICORE-client first sends the AJO to the target, i.e. the Grid-Resource on the server side, that you asked for.

  The target sends the AJO back at the client, when you are asking for your results.

  Because of its nature, UNICORE-users and maintainers mostly do not notice, that it exists. Only code-diggers see, that it is there.
UNICORE in Production

Easy to understand: functionality

Building Blocks of UNICORE, Terms 3 and 4

- **IDB** (Incarnation Database)
  Data description of the target:
  The IDB describes the available resources at the target system.

- **TSI** (Target System Interface)
  Functional description of the target:
  The functional layer works on the target system and connects the real world operating system with the abstract UNICORE world.
UNICORE in Production

Easy to understand : functionality

Building Blocks of UNICORE, Terms 5 and 6

- **UUDB** (Unicore User Database)
  This database contains a table, which connects your certificate (i.e. your UNICORE-identity) with your user account on the operating system of the target.

- **NJS** (Network Job Supervisor)
  This code works like a servant of your UNICORE-task.
  
  On the front side, he identifies you, talks to your client and (if satisfied) writes down, what your wishes are.
  On the back side he acts as a shepherd, when your (sheepish?) task is handed over to the TSI of the target system.
UNICORE in Production

Easy to understand : functionality

Building Blocks of UNICORE, NJS and its Databases

NJS
Network Job Supervisor
- UUDB Unicore User Database
- IDB Incarnation Database

UUDB
UNICORE User Database connects in a table the Certificates with the OS-specific User-Accounts

<table>
<thead>
<tr>
<th>Certificate 1</th>
<th>User-Account 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate 2</td>
<td>User-Account 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Certificate n</td>
<td>User-Account n</td>
</tr>
</tbody>
</table>

IDB
Incarnation Database describes Resources of the Target System

| Target has 16 CPU's with 2,4GHz |
| Target has 2 GByte of Memory   |
| ...                             |
| Target has Fortran installed    |
UNICORE in Production

Easy to understand : functionality

Building Blocks of UNICORE, Terms 7, 8 and 9

- **Vsite (Virtual Site)**
  a server with its hardware and software, with least a TSI and the IDB is available, is called Vsite

- **Usite (Unicore Site)**
  a collection of Vsites in an intranet behind a firewall is called Usite

- **UNICORE-Gateway**
  the guard placed in the DMZ of a firewall, which pre-checks incoming AJO’s.
UNICORE in Production

Easy to understand: security

SECURITY

Security Aspects of UNICORE
UNICORE in Production

Easy to understand: security

Following Components, are part of the UNICORE-Security System:
- Client (and plugins)
- Gateway
- NJS

They all use a Keystore containing:
- Trust= Certs of trusted CA's
- Id-Prove= Id-Certs signed by CA's
- Identity= private keys fitting Id-Certs

X.509-Certificates (with Public Keys)
by CA of SfR and/or HWW

Private Keys (secret)

The Keystore is protected by a single password
UNICORE in Production

Easy to understand: security

General Security Handshake between Unicore Components

Component 1

Here is my Certificate

Here is my Identity:
Encrypt secret with private key

Component 2

Signed by trusted CA?

Id fits to Certificate?

Authentication: Identity of Component 1 is proven
UNICORE in Production

Easy to understand: security

Complete UNICORE-Workflow for a simple Abstract Job Object (AJO)

**Client**
sends AJO with Certificate

**Gateway**
rejectsCertificates of untrusted CAs
approvesIdentity
sends valid AJOs to the NJS

**NJS**
rejectsCertificates missing in the UUDB
compares AJO-Tasks with IDB
sends valid AJOs to the TSI

**TSI**
transforms AJO-Tasks into OS-specific Commands
uses OS-Batch-System
uses OS-specific user-account
UNICORE in Production

Easy to understand: example GRS and HWW

- Client@GRS sends AJO with SfR-ID-Cert
  - ClientKeystore contains Private Key
    - SfR-Id-Cert
    - SfR-CA-Cert
    - HWW-CA-Cert

- GRS Gateway trusts SfR-Cert
- DMZ GRS

- USite GRS

- NJS/UUDB Garching
  - VSite with IDB Cluster Garching

- NJS/UUDB Braunsch
  - VSite with IDB Cluster Braunsch

- USite HWW

- Central NJS
  - UUDB has HWW-Certs & SfR-Certs
  - All IDBs

- HWW Gateway trusts SfR-Cert
- trusts HWW-Cert
- DMZ HWW

- VSite w/o IDB NEC-SX8 Stuttgart

- VSite w/o IDB XC1 Karlsruhe
UNICORE in Production

Easy to understand: example GRS and HWW

The UNICORE-Client
shows all
USites/VSites reachable for customers, that are working for GRS
UNICORE in Production

Easy to do: What next?

- UNICORE is easy to use
- UNICORE is easy to understand
- **UNICORE is easy to install**
- UNICORE is easy to maintain
UNICORE in Production

Easy to install

- plan your site
- download the needed components
- install the components
- configure the components
- check the installation
UNICORE in Production

Easy to install: Plan your Site

- which computer will act as what in UNICORE?
- IP-ports, the components shall communicate on?
  - Client – Gateway
  - Gateway - NJS
UNICORE in Production

Easy to install: Download the Software

Download the software from Sourceforge:

- Client
- Plugins
- NJS
- TSI
- UUDB
- Gateway
UNICORE in Production

Easy to install: Install the Components

- **Client**: downloaded file is self installing software
  
  Simply activate the downloaded file

- **Server Parts**: downloaded files are archive-files
  
  Extract them
  
  $ tar -xzf <filename>.tar.gz
UNICORE in Production

Easy to install: Configure the Server Parts

- Change to directory named ./conf

- Adopt entries in *.properties file
  Simply follow the comments to use the right
  - hostnames and ports
  - certificate files
  - etc
UNICORE in Production

Easy to install: Check the Installation

- Change to directory `.conf/logs`

- Watch the logfiles
  
  use a higher level of debug-verbosity at the start
UNICORE in Production

Easy to do: What next?

- UNICORE is easy to use
- UNICORE is easy to understand
- UNICORE is easy to install
- UNICORE is easy to maintain
UNICORE in Production

Easy to maintain

- Monitoring the System
- Updating the System
UNICORE in Production

Easy to maintain: Monitoring

- Monitoring the System
  - Use external tools
  - Big Brother (DWD, GRS)
  - Nagios (GRS and HWW; with reporting)

our experience:

OS-specific tools break much more often, than UNICORE does
UNICORE in Production

Easy to maintain: Monitoring with Big Brother
UNICORE in Production

Easy to maintain: Update

- Update the System

  Watch, if something has changed on SourceForge,
  Download and reinstall.
  Keep the configuration files.

OR …
UNICORE in Production

Easy to maintain: Quality Managed Update

- Update the System T-Systems Style

  Take advantage of QM-based Release-Management

  But this leads to another story! ...
UNICORE in Production

Is Easy to do

THANKS FOR LISTENING
Release Management for UNICORE

1. Customer Aspects
   • Deliver innovative and stable GRID-Software to the customer (just an update)

2. Developer Aspects
   • Synoptical check of UniGrids with prerequisite packages
   • Formalize dependencies and versioning
   • Early warning system of malfunctions
   • Continuous Integration Tool: Apache Gump
Release Management: Customer Aspects (just an update)

Customer needs:
- Stability

Developer wants:
- Innovation

T-Systems SfR offers Service:
- Testing
- Debugging
- Change Integration

Institutes:
- DWD
- GRS
- DLR
- UniGrids
- NextGrid
- IN-Grid
Release Management: Developer Aspects

Unigrids

AXIS

LOG4J

commomns-discovery

commomns-httpclient

SAXON8

XERCES

ws-addressing

commons-logging

uses projects, which are still under definition and development themselves
Unigrids

Unigrids breaks if another project fails or changes definitions

Jar-Hell threatens:
thttp://www.krysalis.org/version/jar-hell.html

Approved Solution: Continuous Integration Tool

AXIS

comics-discovery

LOG4J

comics-httpclient

SAXON8

XERCES

ws-addressing

commons-logging
Release Management
Solution: Continuous Integration Tools

There are many Continuous Integration Tools

- CruiseControl
- DamageControl
- Tinderbox
- BuildBot
- Anthill
- BeetleJuice
- LuntBuild
- **Gump (Apache) <= Unigrids depends on Apache-Projects mostly**
- Draco.NET
- Continuum
- Sin
- Parabuild
Release Management: How Gump works

- Unigrids
- AXIS
- LOG4J
- commons-discovery
- commons-httpclient
- SAXON8
- XERCES
- ws-addressing
- external software

Uses projects, which are still under definition and development themselves.
Release Management: How Gump works

Phase 1: Download and Integrate changes into testing environment

- CVS
- Subversion
- External software
- Unigrids
- Pilot-Appl. Quabox / Cubbox

Testing environment
Release Management: How Gump works
Release Management: How Gump works

Phase 2: Build- and functionality
Testing
building tools
• ant
• maven
• make

testing tools
• junit
• user defined
Release Management: How Gump works

Phase 3: Results, Presentation and Notification

- **Fileservers**
  - Pool of tested software

- **Webserver**
  - Webpages
    - for developer: logfiles
    - for customer: observable metric (ex. error-count)

- **testing environment**
  - Email to developer if new error occurred
Release Management:
Work already done with Gump

For Phase 1: Download
  – Description of Repositories

For Phase 2: Building and Testing
  – Description of Unicore/UniGrids Package Dependencies

For Phase 3: Results, Presentation and Notification
  – Web-Server
    [http://tdb.grs.de/gump](http://tdb.grs.de/gump) (URL will change)

  – File-Server with central Download-Pool for the Customers is in Preparation
Thank You

Gert Ohme
T-Systems SfR GmbH
Customer Center GRS

mailto: Gert.Ohme@t-systems.com
Internet: http://t-systems.com