UFTP
High-performance data transfer for UNICORE

Dr. Bernd Schuller, Tim Pohlmann
Federated Systems and Data division
Jülich Supercomputer Centre
Forschungszentrum Jülich GmbH

July 8, 2011
UNICORE Summit, Toruń
Outline

- Filetransfer in UNICORE
- UFTP
  - Principles
  - Deployment
  - Examples
- Outlook
Data flows using the BFT data transfer

BFT import/export requires three socket connections

BFT staging requires four socket connections

UNICORE basic services

Target System Interface

Local RMS (e.g. Torque, LSF, etc.)

UNICORE basic services

Target System Interface

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

command-line client

Eclipse-based client

UNICORE services

Gateway

authentication, firewall transversal

access to data and management of computational jobs
Ideal data flow

- Import/export without extra socket connections
- command-line client
- Eclipse-based client
- Target System Interface
- Local RMS (e.g. Torque, LSF, etc.)
- RMS (e.g. Torque, LL, LSF, etc.)

Scientific clients and applications

Staging without extra socket connections

Access to data and management of computational jobs
Issues with direct data transfer

- Firewall!
  - Direct connections from the outside to the TSI login node are usually not allowed
  - Statically opening ports (or worse, port ranges) is a security risk

- Port opening technique is required
  - UDP based hole punching (like Skype), but UDP is not directly suited for file transfer
  - TCP based: passive FTP is widely understood, but considered insecure
Basic idea: use passive FTP to open ports

1. "PASV"
2. open 5432 for Client
3. connect to port 5432
4. close control connection
5. close 5432
UFTP: combining passive FTP and UNICORE

- FTP by itself is insecure:
  - Users log in using username/password
- UNICORE provides a highly secure channel from client to server which is used for additional security measures:
  - File transfers are always initiated via UNICORE
  - Client must authenticate using a „secret“ that is exchanged via UNICORE
- Requires an secure „command port“ in addition to the FTP port
UNICORE/X server

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X

UNICORE/X
File transfer using UFTP

1. Create transfer
2. init

Init information sent via command channel includes:
- client IP
- file name
- authn secret
- optional encryption key

3. get filetransfer properties (uftp host and port, # of streams)
4. Upload/download
Security challenges and their resolution

- uftpd server runs with root privileges (because it needs to access files from all users)
  - Switch effective user/group ID before file access
- Sending commands via the Command channel allows local users to read/write files under any user ID
  - Command port not accessible outside the firewall
  - Protect it using client authenticated SSL and ACL file
- Attacker might connect to the newly opened sockets on the uftpd server
  - Client IP is checked, and a secret key is required for authentication
- Data channels might be sniffed
  - Optional symmetric encryption (64 bit key, blowfish algorithm)
Using UFTP from UCC

optionally add UFTP related preferences …

```python
uftp.client.host=localhost
uftp.streams=2
uftp.encryption=false
```

Specify the protocol in file operations

```bash
>ucc put-file -s /home/... -t https://... -P UFTP
```

Specify the protocol in your UCC job

```json
{
Imports: [
  { From: "u6://...?protocol=UFTP", To: ... },
],
}
```
UFTP performance: example

File size (MB) vs. Transfer rate (MB/sec)

- UFTP
- UFTP encrypted
- BFT

localhost system, Core 2, 4GB memory
UNICORE 6.4.1 with Perl TSI
UFTP 1.0.0, 2 streams

> ucc put-file -s /home/... -t https://... -P UFTP -y
UFTP: a high-performance data transfer for UNICORE

- UNICORE FTP (yes, think of "Grid- (rather Globus-)FTP“)
  - *Multiple parallel TCP connections per data transfer*
  - *Client-Server, Server-Server*
  - *Secure, simple, easy to deploy and operate*
  - *Dynamically open ports in the firewall using the „FTP“ protocol*
  - *Platform independent (Java)*
- Widely available with UNICORE version 6.4.1
  - *as rpm, deb, tar.gz thanks to pac(k)man*
Outlook

- Deployment in realistic environments (DEISA/PRACE and others)
- URC support (coming soon!)
- Workflow system support (current version is still 6.3.x)
- Performance testing
- Enhancements
  - Support multi-homed servers
  - Experiment with buffer sizes

- Other uses of the „FTP trick“ not related to data transfer? E.g. when setting up (cloudy, virtual, …) systems dynamically?
Thanks for early testing and feedback on UFTP

- Krzysztof Benedyczak
- Michael Rambadt, Michael Stephan, Björn Hagemeier

... and thank you for your attention!