



Enabling Grids for E-science

HTTPS/HTTP & xrootd protocols for DPM

Andreas-Joachim Peters

JRA1 All-Hands Meeting CERN 25.10.2007

www.eu-egee.org



Information Society
and Media



- **Transparent Access to DPM files & directories with HTTPS/HTTP protocol**
- **Full support of X509-cert/proxy & VOMS proxies**
- **Full DPM access control - virtual ID mapping**
- **High performance for read operation**
- **High performance for write operations**
 - Write operation via POST from HTTP form or via PUT using e.g. curl or scripting language
- **Optional fully encrypted I/O via https protocol**
- **Lightweight shell client**
- **Configuration via YAIM tools**
- **'dpm-httpd' service based on standard Apache2 daemon**

https with https transport

1. Request URL on headnode Apache port 883 via https
2. SSL session/GridSite decodes additional X509 attributes (VOMS)
3. CGI script interrogates DPM for read/write request (perm. etc.), creates key encoded token and redirects client to disk server on secure port
4. Client connects on disk server Apache port 884/https
5. SSL session/GridSite
6. Key encoded token from head node is verified by disk server (authorization check)
7. Apache serves DPM file via https to client

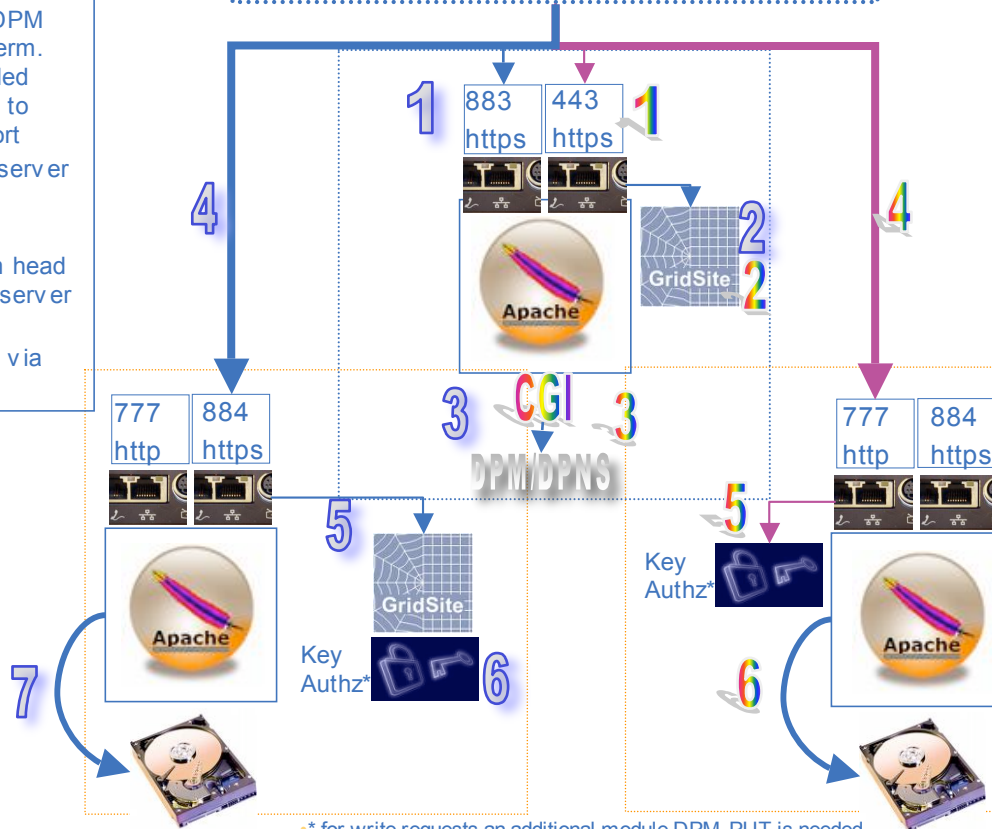


•Clients

- Firefox (X509)
- Explorer (X509)
- Safari (X509)
- wget / curl (X509/Proxies)

https with http transport

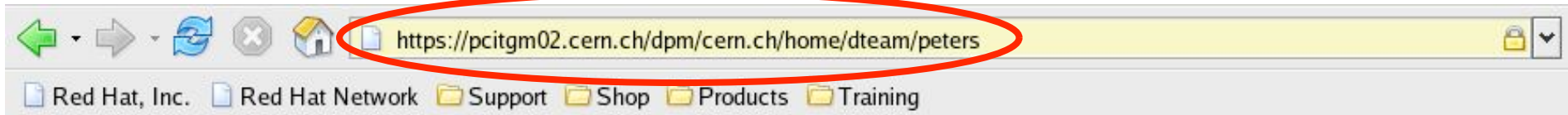
1. Request URL on headnode Apache port 443 via https
2. SSL session/GridSite decodes additional X509 attributes (VOMS)
3. CGI script interrogates DPM for read/write request (perm. etc.), creates key encoded token and redirects client to disk server
4. Client connects on disk server Apache port 777/http
5. Key encoded token from head node is verified by disk server (authorization check)
6. Apache serves DPM file via http to client



* for write requests an additional module DPM-PUT is needed

- **HTTP(s) Access Model**


- Authenticate via HTTPS with Certificate, X509 or VOMS-Proxy to the DPM head node
 - *Do file transport over*
 - HTTP (high-performance / low client load) - 'insecure' I/O
- or -
 - HTTPS (high-performance with high client load) - 'secure' I/O
- In multi-server setups the HTTPS client gets redirected to the disk server responsible for file serving
 - DPM head node
 - *HTTPS port 443 for initial contact with redirection to HTTP port*
 - *HTTPS port 883 for initial contact with redirection to HTTPS port*
 - DPM disk server
 - *HTTP port 777 for HTTP transport*
 - *HTTPS port 884 for HTTPS transport*
- All nodes run a standard Apache web server
- Redirection is verified via a special key authorization module for Apache which verifies a signature in the redirection URL from the head node



DPM HTTPS BROWSER

• Web Browser Interface:

- Read
- Write
- Browse
- Rm
- Chmod
- Stat
- Mkdir
- Rmdir

Author:  Andreas.Joachim.Peters@cern.ch 2007 - IT-GD

New File-/Dirname ... Mode

permissions	nlinks	owner	group	size	mtime	filename	
						[parent dir] ..	
-rw-rw-r--	1	101	102	115	Aug 23 2007	higgs-candidate01.root	<input type="button" value="post"/> <input type="button" value="rm"/> <input type="button" value="chmod"/> <input type="button" value="stat"/> Mode <input type="text" value="664"/>
-rw-rw-r--	1	101	102	1558550	Aug 23 2007	higgs-candidate02.root	<input type="button" value="post"/> <input type="button" value="rm"/> <input type="button" value="chmod"/> <input type="button" value="stat"/> Mode <input type="text" value="664"/>
-rw-rw-r--	1	101	102	2162348	Aug 23 2007	higgs-candidate03.root	<input type="button" value="post"/> <input type="button" value="rm"/> <input type="button" value="chmod"/> <input type="button" value="stat"/> Mode <input type="text" value="664"/>
-rw-rw-r--	1	101	102	6122265	Aug 23 2007	higgs-candidate04.root	<input type="button" value="post"/> <input type="button" value="rm"/> <input type="button" value="chmod"/> <input type="button" value="stat"/> Mode <input type="text" value="664"/>
-rw-rw-r--	1	101	102	1730675	Aug 23 2007	higgs-candidate05.root	<input type="button" value="post"/> <input type="button" value="rm"/> <input type="button" value="chmod"/> <input type="button" value="stat"/> Mode <input type="text" value="664"/>

Shell Interface based on 'wget'

```
[pcitgm02] /home/peters > /opt/lcg/bin/dpm-httpd-cp https://pcitgm02.cern.ch/dpm/cern.ch/home/dteam/uploadedfile /tmp/localfile
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100    713    100    713     0     0    1936       0  --:--:--  --:--:--  --:--:--   696k
100   6433    100   6433     0     0    660k       0  --:--:--  --:--:--  --:--:--   628k
```

- **dpm-httpd-cp** <src> <dest>
 - <src>, <dst> URL like 'https://<host>/dpm/cern.ch/....' or local file
- **dpm-httpd-cmd** <cmd> <url>
 - <cmd> can be
 - **ls**
 - **mkdir**
 - **rm**
 - **chmod**
 - **stat**
- Return values:
 - 0 - success
 - != 0 - error code

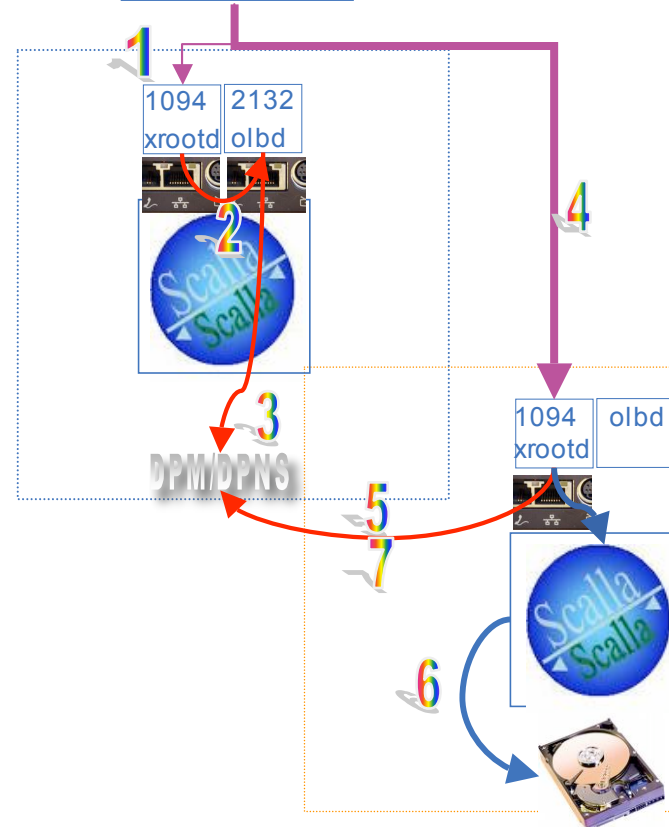
- **Providing access to DPM files with xrootd protocol via the native 'xrootd' daemon.**
- **Architecture:**
 - OLBD XMI plugin in olbd daemon for redirection
 - DPMFS OFS plugin in xrootd daemon for file access
 - xrootd & olbd daemons run like rfiod on disk server and head node
- **Limitations:**
 - All user mapped to single DPM identity in restricted DPM namespace ('root')
 - Currently no GSI/VOMS support due to missing support of client information in the olbd framework possible - although plugin would support it
 - Will be fixed in future xrootd version

xrootd access on DPM

1. Request path on DPM headnode xrootd port 1094 via (x)root protocol
2. xrootd interrogates olbd to 'locate' path
3. olbd communicates with DPM service to redirect 'get' or 'put' request with a DPM token and SFN
4. Client get's redirected to xrootd on disk server
5. xrootd verifies DPM token and permissions
6. xrootd gives access to path
7. xrootd closes the DPM transaction when file gets closed



- xrdcp
- root> TFile::Open("root:...");



- **Test configuration:**

- DPM node: 8 GB Mem. / 2 x double-core 2.4 GHz Pentium (R) - Linux 2.6.22.6 SMP x86_64
- Client Node: 3.4 GB Mem. / 2 x 3.4 GHz Pentium (R) Linux 2.6.9-55.0.2.EL.cernsmp x86_64
- Gigabit interconnection - same switch



- **File size $\Rightarrow 0$ (minimal transport, mainly service IA)**
 - Read 1024kb files with single clients:

https/http-io	0.27 s / file
rfio	1.61 s / file
xrootd	1.02 s / file

- **File size $\Rightarrow \infty$ (transport dominating, service IA min.)**
 - Read 1 GB files with single clients:

https/http-io*	17.2 s / file
rfio*	18.9 s / file
xrootd*	20.7 s / file

* data not cached on file server

- Parallel clients reading randomly 1024kb files (time seen by an individual client)

#Clients	https/http-io	xrootd	rfio
5	0.27 s / file	0.99 s / file	1.61 s / file
10	0.32 s / file	1.57 s / file	1.64 s / file
20	0.37 s / file	3.21 s / file	1.76 s / file
40	0.71 s / file	6.69 s / file	2.52 s / file *

* dpm & dpns daemon take 100% CPU each on one core

- **Parallel clients reading 1Gb files**
 - all files un-cached
 - measurements limited by ‘random’ seek on server disk
 - all measurements give the summed I/O rate for 10 clients

Clients	xrootd	rfio	https/http-io
10	13.3 Mb/s	13.3 Mb/s	13.3 Mb/s

- **Parallel clients reading single identical 1Gb file**
 - No disk I/O limitation
 - always sum of I/O for all clients given

Clients	https/ http-io	rfio	xrootd	https/ https-io
1	100 Mb/s	97.5 Mb/s	96.0 Mb/s	98.5 Mb/s [1 core 100% CPU]
10	99.1 Mb/s	100.7 Mb/s	95.0 Mb/s	
40	95.2 Mb/s	93.4 Mb/s	88.0 Mb/s	
80	100 Mb/s	97.5 Mb/s	95.3 Mb/s	
120	96.6 Mb/s	96.6 Mb/s	96.6 Mb/s	
120	60 %	80 %	65 %	Client Load
10/40/80/120	27 %	27 %	27 %	Server Load

- **HTTPS protocol implemented for DPM**
 - High bandwidth file streaming possible over HTTP
 - High bandwidth file streaming possible over HTTPS encrypted but with higher server/client load
- **xrootd protocol**
 - Native xrootd integration into DPM setup
 - No grid-like Authentication yet
- **Performance comparisons**
 - No evident difference from client point of view between HTTPS/rfio/xrootd protocol
 - Lowest file open latency for HTTPS implementation (read)
 - DPM in test setup can handle ~16 reads (dpm_get) per second via rfio or 56 reads (!dpm_get) per second via HTTPS
 - CPU usage of dpm & dpns daemon play a role in limitation to <files/s>
 - To be studied further

- **Some documentation & sources about HTTPS access can be found under**
 - <https://twiki.cern.ch/twiki/bin/view/LCG/DpmHttpsAccess>

Thanks for your attention !