dCache, a managed storage in grid

Patrick
for the dCache Team

support and funding by
Topics

Project Topology

Why do we need storage elements in the grid world?

The idea behind the LCG (gLite) storage element.

Available Solutions

The dCache implementation

dCache in a nutshell

Weak points and outlook

Usage

Selected Topics
What do we need a grid storage element for?

We need to serve large amounts of data locally

- Access from local Compute Element
- Huge amount of simultaneously open files.
- Posix like access (What does that mean?)

We need to exchange large amount of data with remote sites

- Streaming protocols.
- Optimized for low latency (wide area) links.
- Possibly controlling 'link reservation'.
We need to allow storage control

- Space reservation to guarantee maximum streaming.
- Define space properties (TAPE, ONLINE, ...)
- Transport protocol negotiation.

We need to publish SE specific information

- Clients need to select 'best' SE or CE for a job.
- Availability
- Available Space (max, used, free ...)
- Supported Spaces (Tape, disk ...)
- Which VO owns which space?
The Idea of a Grid Storage Element

Wide area dataset transport
Possibly managed by high level tools (FTS)
The Idea of a (LCG) Grid Storage Element

- Information Publishing
  - Content: GLUE
  - Transport: LDAP

- SRM Storage Resource Management
  - Space/Protocol Management

- Wide Area Transport Protocol
  - In use: gsiFtp
  - Discussed: http(s)

- Local Access Protocol
  - (gsi)dCap or rfio and xRoot
Available Solutions

Common Protocols
- infoProvider
- SRM
- gsiFtp
- rfio
- dCap
- xRoot

Available Solutions

- dCache
- CASTOR
- DPM
- SToRM

Common Protocols
- infoProvider
- SRM
- gsiFtp
- rfio
- dCap
- xRoot
The dCache SE implementation

Black Box View

dCache Core

High Level Services
- Resilient Manager
- Admin Module (ssh, jpython)
- Maintenance Module
- Flush Manager
- Hopping Manager

Information Protocol(s)
- (NFS 4.1)

Storage Management Protocol(s)
- SRM 1.1 2.2

Data & Namespace Protocols
- dCap
- gsiFtp
- xRoot
- (http)

Namespace ONLY
- NFS 2 / 3

Tape Storage
- OSM, Enstore
- Tsm, Hpss, DMF

Namespace provider

ACLs

The dCache SE implementation

Black Box View

dCache Core

High Level Services
- Resilient Manager
- Admin Module (ssh, jpython)
- Maintenance Module
- Flush Manager
- Hopping Manager

Information Protocol(s)
- (NFS 4.1)

Storage Management Protocol(s)
- SRM 1.1 2.2

Data & Namespace Protocols
- dCap
- gsiFtp
- xRoot
- (http)

Namespace ONLY
- NFS 2 / 3

Tape Storage
- OSM, Enstore
- Tsm, Hpss, DMF

Namespace provider

ACLs

The dCache SE implementation

Black Box View

dCache Core

High Level Services
- Resilient Manager
- Admin Module (ssh, jpython)
- Maintenance Module
- Flush Manager
- Hopping Manager

Information Protocol(s)
- (NFS 4.1)

Storage Management Protocol(s)
- SRM 1.1 2.2

Data & Namespace Protocols
- dCap
- gsiFtp
- xRoot
- (http)

Namespace ONLY
- NFS 2 / 3

Tape Storage
- OSM, Enstore
- Tsm, Hpss, DMF

Namespace provider

ACLs

The dCache SE implementation

Black Box View

dCache Core

High Level Services
- Resilient Manager
- Admin Module (ssh, jpython)
- Maintenance Module
- Flush Manager
- Hopping Manager

Information Protocol(s)
- (NFS 4.1)

Storage Management Protocol(s)
- SRM 1.1 2.2

Data & Namespace Protocols
- dCap
- gsiFtp
- xRoot
- (http)

Namespace ONLY
- NFS 2 / 3

Tape Storage
- OSM, Enstore
- Tsm, Hpss, DMF

Namespace provider

ACLs
\textbf{dCache in a Nutshell}

- Strict name space and data storage separation, allowing
  - consistent name space operations (mv, rm, mkdir e.t.c)
  - consistent access control per directory resp. file
  - managing multiple internal and external copies of the same file
  - convenient name space management by nfs (or http)
In a Nutshell

- dCache partitioning for very large installations
- File hopping on
  - automated hot spot detection
  - configuration (read only, write only, stage only pools)
  - on arrival (configurable)
File Hopping

- Backup Store
- Main Tertiary Store
- Transfer Pools
- Read Only Cache
- Write Pools

RAW Data 2. copy

From Client

Replicate on high load

non RAW Data immediately copied to read pools
dCache in a Nutshell

- Overload and meltdown protection
  - Request Scheduler.
  - Separate I/O queues per protocol  (load balancing)
- Supported protocols: (gsi)ftp, (gsi)dCap, xRoot, SRM, nfs2/3
- xRoot support
  - Vector read
  - Currently working on asyn I/O
Technical Introduction

Scheduler and I/O queues

I/O Request

Space Manager

Dispatcher by request Attributes

Pool Candidates selected by Protocol
Client IP number/net
Data Flow Direction
Name Space Attributes (Directory)
SRM Spaces

List of candidates

Dispatcher by Pool Cost

Pool Protocol Queues

xRoot  dCap  gsiFtp
In the Nutshell

- **HSM Support**
  - TSM, HPSS, DMF, Enstore, Osm
  - Automated migration and restore
  - Working on Central Flush facility
  - support of multiple, non overlapping HSM systems (NDGF approach)

- **Misc**
  - Graphical User Interface
  - Command line interface
  - Jpython interface
  - SRM watch
  - NEW : Monitoring Plots
Is this useful for non LCG applications?

Weak points:

- Posix like is NOT posix (file system driver)
- Http(s) not really supported
- Security might not be sufficient
**Posix like is NOT posix**

**Linked Library**

- `App`
  - `dCap`
  - `libC`
  - `iNodes`
  - `xfs`
  - `ext3`

Application needs to be linked with the dCap library.

**Preload Library**

- `App`
  - `preload dCap`
  - `libC`
  - `iNodes`
  - `xfs`
  - `ext3`

Application stays unchanged but doesn't work in all cases. (Static linked, Some C++ apps.)
And this is real posix

Application doesn't need to be changed. NFS 4.1 driver comes with OS.
Solution is on the way....
We are currently putting significant efforts in the NFS 4.1 protocol

**Deployment Advantages:**

Clients are coming for free (provided by all major OS vendors).

**Technical Advantages:**

- NFS 4.1 is aware of distributed data
- Faster (optimized) e.g.:
  - Compound RPC calls
  - 'Stat' produces 3 RPC calls in v3 but only one in v4
- GSS authentication
  - Built in mandatory security on file system level
- ACL's
- OPEN / CLOSE semantic (so system can keep track on open files)
- 'DEAD' client discovery (by client to server pings)
NFS 4.1 in dCache
Goal: Industry standards in HEP?
dCache usage

In use at 9 Tier I centers
- fzk (Karlsruhe, GR)
- in2p3 (Lyon, FR)
- RAL (Rutherford, UK)
- BNL (New York, US)
- FERMILab (Chicago, US)
- SARA (Amsterdam, NL)
- PIC (Spain)
- Triumf (Canada)
- NDGF (NorduGrid)

About 40 Tier II's

dCache is part of VDT (OSG)

We are expecting > 20 PB per site > 2011

**dCache will hold the largest share of the LHC data.**
Some more hot topics
The NDGF Challenge: gsiFtp Protocol Version II

NDGF Tier I

- Finland
- Denmark
- Sweden
- Norway

Denmark

Chimera

Head-node

SRM

CERN

Patrick Fuhrmann et al.   BE-grid, Brussels, BE   October 16, 2007
The NDGF Challenge: Multi Site HSM support

Single Site approach

Flush to HSM

Restore to any Pool

Multi Site approach

Not all pools can access all HSM systems

Patrick Fuhrmann et al.  BE-grid, Brussels, BE  October 16, 2007
The wonderful world of

**SRM 2.2**

*Only if there is a lot of time left*
The SRM in dCache supports

- CUSTODIAL (T1Dx)
- NON-CUSTODIAL (T0D1)
- Dynamic Space Reservation
- late pool binding for spaces
- and more
SRM 2.2 (The space token)

As it used to be (<= 1.7)

As it will be with 1.8

Remark: The space of a Space Token is assigned to a pool at the moment the file is opened and not when the space token is created.
Further reading

www.dCache.ORG