Karlsruhe Institute of Technology

dCache

dCache Beginners Course
Introducing dCache

An overview to dCache and how it is deployed in grid environments.

Karlsruhe Institute of Technology (KIT), Steinbuchcentre for Computing (SCC)

Storage Management Systems

- dCache is a so called *storage management system*
- Storage management systems are characterised by the following main attributes:
  - Manage much higher amounts of data/files like “normal” filesystems (for example btrfs, ext4 or XFS) do, but usually on a much higher logical level.
  - Integrate many storage media that can be of different type into one system by utilising *Hierarchical Storage Management* (HSM), e.g. disks in front of a tape archival system.
  - Provide mechanisms to automatically balance the load (e.g. auto-replication), ensure resilience and high availability and means for advanced control systems to manage the data as well as the data flow.
  - Supports several access-protocols (e.g. POSIX or grid protocols like SRM)

Overview And History

- dCache is a highly sophisticated storage management system written in Java.
- It is being actively developed at Deutsches Elektronen-Synchrotron (DESY) since 2003. Major contributions come from Fermi National Accelerator Laboratory (FNAL) and Nordic Data Grid Facility (NDGF).
- It is one of the main storage management systems used within the European Middleware Initiative (EMI), the Worldwide LHC Computing Grid (WLCG) and in many other places.
- It uses standard open source technologies in several places.
- Support is provided by the developers and dedicated support groups. Communication is done mainly via mailing lists.

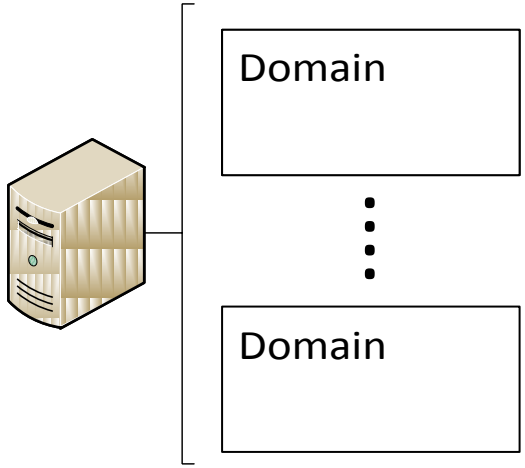
Functionalities And Features

- Detailed logging and debugging as well as accounting and statistics.
- XML information system with detailed live information about the cluster.
- Web-interface with live summaries of the most important information.
- Checksumming of data.
- Resilience and high availability can be automated in different ways.
- Powerful cost calculation is consulted in matters of data flow control (from and to pools, between pools and also between pools and tape) load balancing and performance tuning.
- Garbage collection of replicas, depending on their flags, age, et cetera.
- Space management and support for space tokens.

This course does not cover the HSM-capabilities ("writing to tape") of dCache.

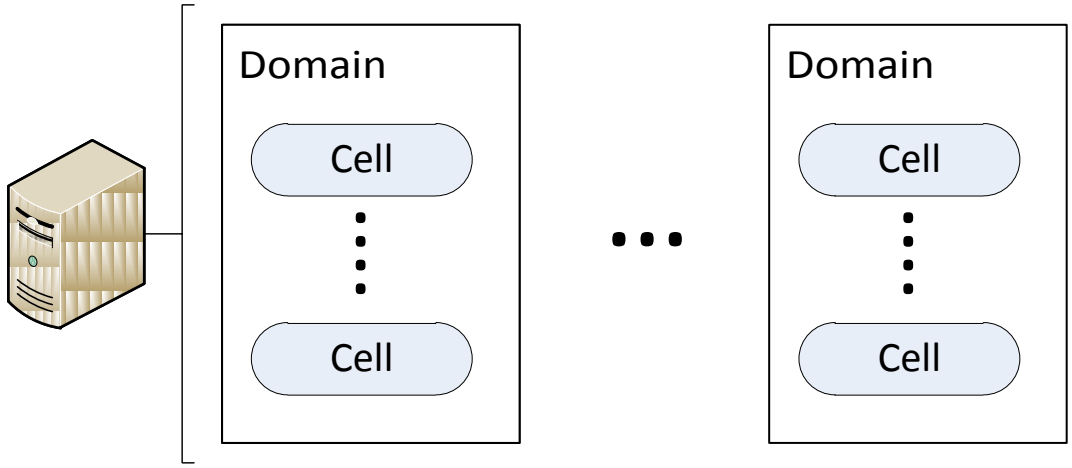
dCache's structure

- A dCache cluster is structured by the following three classes of objects:
  1. Domains
     - Each domain corresponds to exactly one Java VM and is thus bound to exactly one node of the cluster.
     - Nodes can generally run multiple domains.
     - A domain groups cells together.



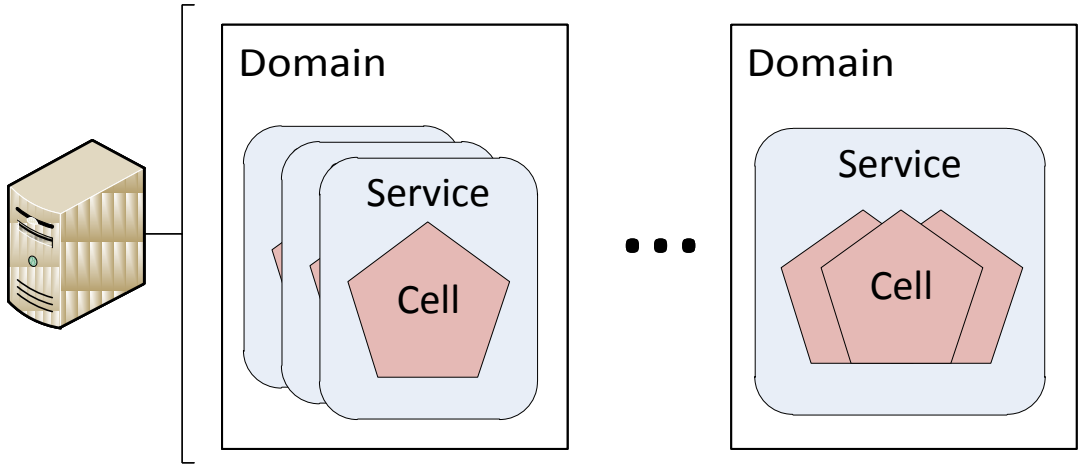
dCache's structure

2. Cells
   - A cell is the entity that provides services.
   - Every cell is always a member of exactly one domain.
   - Depending on the cell-type there may be multiple instances per cluster, possibly each running with different settings.



dCache's structure

3. Services
   - A service is provided by one or more cells (possibly from different domains).
   - Depending on the service-type there may be multiple instances per cluster, possibly each running with a different set of settings.
   - Some services require several cells to run properly.



About Cells And Services

- “Core”-services and –cells
  - Implement basic, common and central functionality.
  - Often these kind of cells/services are singular throughout a dCache setup.
- “Door”-services and -cells
  - Serve as gateways to the data stored in dCache.
  - Protocol specific.
  - Multiple doors allow for load balancing.
    - Setting up more than one SRM door should be done only by experts.
- “Pool”-services and -cells
  - Manage the actual data stored in dCache.
  - Of course, many pools are allowed in one dCache setup.
- There are a couple of additional cells/services, that are not vital to a functional dCache installation.

Supported Access Protocols

- NFS 3 – limited access to the file hierarchy and file meta data over network.
- NFS 4.1 (pNFS) – “full” network filesystem access.
- WebDAV – a superset of HTTP.
- gsi-/DCAP - *dCache Access Protocol* is dCache’s original and native access protocol, possibly GSI-secured.
- gsi-/FTP – the *File Transfer Protocol*, also in a GSI-secured fashion (a.k.a. GridFTP).
- xroot – the protocol from xrootd, the *eXtended ROOT Daemon*.
- SRM – *Storage Resource Manager* is a meta-protocol targeted at storage management systems, including many specific features like space tokens, lifetime and pinning. For the actual data transport another transfer-protocol (like GridFTP or gsiDCAP) is used.

File Hierarchy Provider

- dCache gathers all files in a traditional file hierarchy, which allows files being identified by (path-)name.
- For this a name space abstraction layer on top of a relational database called *Chimera* is used.
- Internal to dCache, files are identified by unique *PNFS-ID*s and Chimera keeps track of the mapping pathname ↔ PNFS-ID.
- Chimera also manages file's storage information and meta data, e.g. actual location in pools, replicas, creation time, POSIX file permission modes, etc.
- The cell offering the services to interact with Chimera in dCache is called the *PnfsManager*.
- The NFS 3 or NFS4.1 door services are there to allow NFS-mounts of the name space hierarchy.

dCache Setup On Clusters

- Thanks to the modular build of dCache, the cells can almost freely be grouped or separated in domains as desired.
- Everything is possible between the two extremes.
  - Having all cells running in a sole domain on one single machine, or
  - letting every cell run in its own domain on a dedicated machine.
- Typically, the first core services that are granted dedicated hardware are the SRM service and PnfsManager. All other services are somewhat lightweight and can be grouped on one host.
- Usually, the doors are next to be set up on dedicated machines.
- Pools are running on machines that are capable of managing heavy I/O to disk and via network.
- In order to ease the administration it is generally beneficial to run services/cells each in an own domain (as long as there is enough RAM to spend).
  - This way, every service can be restarted independently from the others

First Chapter Completed!

- Are there any questions?