New tools for dCache*

Malte Nuhn ,

DESY-IT, RWTH Aachen

Abstract. dCache is one possible implementation of LCG's Storage Element. It offers a comfortable way to access dCache's namespace by exporting metadata via NFS. dCache's current implementation (pnfs) uses NFS2/3 which unfortunately provides weak security and has other disadvantages. To avoid using the NFS-export but still provide straightforward access, a set of tools is developed. These tools query dCache's new Information Provider to provide loadbalancing and automatic retrieving of dCache's doors. In order to achieve good performance, a new caching-service for dCache's IP is implemented. These tools are mainly used by users in the National Analysis Facility (NAF) but also other sites seem to be interested in them.

Key words and phrases: dCache, dcTools, pnfs, information provider, caching

.

1. INTRODUCTION

In order to cope with the huge amout of data produced by the Large Hadron Collider (LHC), the data is distributed among all datacenters participating in the LHC Computing Grid (LCG). Amongst others, these datacenters need to install so called Storage Elements (SE), which provide access to storage capacity via well defined protocols. A popular implementation of SEs is the dCache-System[1].

2. DCACHE

dCache is a disk-caching system, developed by DESY and Fermilab. Eventhough dCache was initially developed to provide disk caching for tertiary storage, it is now often used as SE without tape backend. It provides a single name space across the entire pool of disk servers, making it look like a single filesystem to the user. For more information see (1).

RWTH Aachen, (e-mail: malte(dot)nuhn(at)rwth-aachen(dot)de).

*Supervisor: Andreas Gellrich & Yves Kemp

[1]Other implementations: CASTOR, StoRM, ...

2.1 dCache architecture

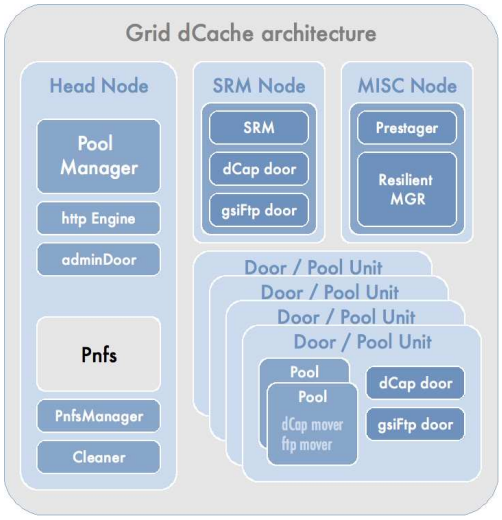
dCache consists of multiple subservices that can - for scalability reasons - be spread among different machines. (see fig. (1) and (6)). The so called *Pool Nodes* manage the data storage. dCache allows you to install multiple pool-nodes, which can be fairly heterogeneous. *Admin Nodes* run administrative services and modules to keep the pools up-to-date and can be put onto different machines, as well. *Door Nodes* provide file access through different grid transfer protocols. Being responsible of serving all of dCache I/O, these machines must have sufficient CPU and memory to handle the user's requests. In larger installations, multiple doors are used. The *pnfs*[2] *Nodes* are responsible for dCache's namespace and mainly do the mapping between logical names and physical locations within dCache's Pool Nodes.

2.2 Problems with pnfs

As mentioned above, pnfs provides a single name space across the entire pool of disk servers so that users can transparently access very large data stores as though a single filesystem. Unfortunately

[2]Pretty Normal File System

FIG 1. *dCache architecture*



dCache's implementation currently uses NFS2/3[3] which provides weak security. NFS2/3 security concept mainly depends on a trusted networks. Besides other problems (5) [4], this requirement can't be satisfied anymore, so that the usage of the current pnfs implementation should be avoided. In order to provide a straightforward access to dCache's storage without using dCache's current implementation of pnfs, a set of tools is needed.

2.3 dcTools

Another way to access the dCache namespace (+ storage), is to access dCache's running doors and performing the needed operations there. Using this method of accessing dCache needs a couple of unhandy operations, e.g.: Checking for valid proxies when using gsidcap and gsiftp; spotting available doors; choosing the best door; loading libraries and calling other tools. In order to gain better performance on performing operations on specific doors, loadbalancing with respect to dCaches new Information Provider(2) should be performed.

3. PROJECTS OBJECTIVES

Until a new implementation of dCache's pnfs is available, dcTools need to be enhanced.

[3]This will hopefully change in the near future!

[4](...) This is really bothering, not to have access to real file info quickly and with ordinary shell commands (...)

3.1 Enhancements

- Enhance overall code
- Make it easier to maintain
- Remove needless requirements
- Produce better output

3.2 New features

In order to improve overall performance, dcTools need new features.

- Spot available doors through dCaches Information Provider
- Provide load balancing
- Take blackbox tests into account
- Still support static configruation

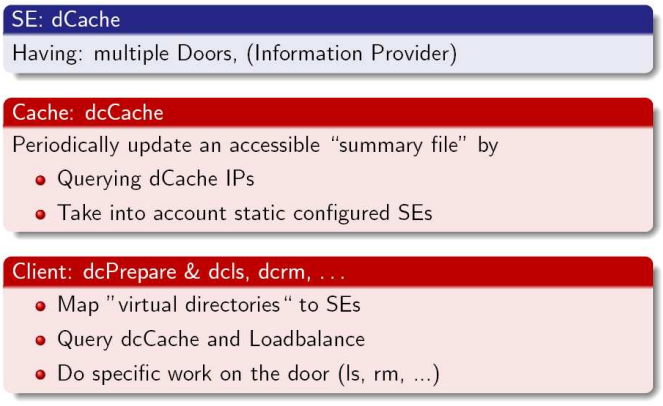
4. SOLUTION

The objectives mentioned cannot be achieved by a straight forward implementation of a better client. A monolithic client, querying dCache's IP to get an up-to-date doorstatus without caching would be too slow. dCache's IP Output is an XML with a size of about 150 Kbyte. Downloading and parsing this file just in time would

- Produce too much traffic on dCache's IP
- Have to much overhead
- Not be able to take blackbox tests into account

Because of this, some sort of caching is needed. Because blackbox tests cannot be effectively done on clientside and a lot of IP queries can be avoided when performing caching on a third location, *dcCache* and *dcPrepare* come into play. (See also fig. 2)

FIG 2. *Basic concept*



4.1 dcCache

dcCache is a cronjob currently running on the PAL-Cluster distilling all information about available SEs. The configuration is mainly done by a directory-structure representing all SEs to be processed.

- Each SE *may* have an update script to process the IPs output
- Each SE *must* have an xml file describing available doors

After that, the cronjob summarizes all these sub files resulting in a file less than 500 Bytes in size. This file is then accessible using http or a distributed file system (e.g. Andrew File System). This makes dcCache lightweight, simple to maintain and keeps it flexible by allowing the update scripts to be customized easily.

4.2 dcPrepare

dcPrepare is a selfcontained tool performing the following tasks

- Mapping between logical filenames and SEs
- Interaction with dcCache
- Loadbalancing (currently weighted random)
- Configuration for environment-variables and possible postprocessing

dcPrepare can be fully configured via an xml file containing information about how to map logical filenames to the SEs. This is done by defining regular expressions and possible substitution rules. Furthermore the location of dcCache's output is specified here. Additionaly, parsing and initialisation scripts can be defined for different protocols. This information will later-on be consumed by the tools performing specific operations like for example *dcls*, *dcrm*, ...

4.3 dcls, dcrm, ...

The specific operations are basically very short scripts wrapping together all needed commands. All they do is

- Ask dcPrepare for a door
- Check for a proxy if needed
- Load possible libraries (e.g. dCache preload libs)
- Execute other tools (e.g. edg-gridftp-ls)
- Pipe output through the configured postprocessing script

5. RESULTS

The concept seems to work out quite well. Because the whole concept is kept strictly modular, different administrators should easily be able to adopt the tools to their needs.

5.1 Timing

The caching algorithm gives an enormous boost in performance. While dcCache's update scripts take up to ≈ 5 seconds to update[5], dcPrepare constantly provides output in under ≈ 0.2 seconds.

5.2 Postprocessing

Being able to provide different postprocessing scripts for different SEs, consistent output along different versions of dCache can be achieved.

5.3 File Locations

The dcTools can be found on
`/afs/desy.de/group/it/services/dcache/bin`
along with the new documentation on
`/afs/desy.de/group/it/services/dcache/doc`

REFERENCES

[1] *dCache website*
http://www.dcache.org/
[2] P. Millar (2008).
New Information Service in dCache 1.8.0-16
DESY/FNAL Announcement
[3] Paul Millar (2008).
New Information Service in dCache 1.8.0-16: XML-Output
https://www.desy.de/~paul/Info/dCache-info-bumblebee.xml
[4] Marcello Barisonzi (2008).
Data Storage at the NAF
https://znwiki3.ifh.de/ATLAS/WorkBook/NAF/Data%20Storage
[5] Mario Kadastik (2008).
pnfs and directory usage
http://www.dcache.org/archive/user-forum/0257.shtml
[6] Christian Forrest (2008).
Overview of Storage Resource Manager (SRM) and dCache for OSG
https://twiki.grid.iu.edu/bin/view/Documentation/StorageDcache

[5] eventhough this is rarely the case