GES 2007 German e-Science

Available online at http://www.ges2007.de

dCache, scalable managed storage

P. Fuhrmann[1], T. Mkrtchyan[1], M. Radicke[1], O. Synge[1]

DESY, Notkestrasse 85, 22607 Hamburg, Germany
email: {patrick.fuhrmann, tigran.mkrtchyan, martin.radicke, owen.synge}@desy.de
phone: (+49 40) 8998 0

Abstract

End of 2007, the most challenging high energy physics experiment ever, the *Large Hadron Collider(LHC)*[9], at CERN, will start to produce a sustained stream of data in the order of 300MB/sec, equivalent to a stack of CDs as high as the Eiffel Tower once per week. This data is, while produced, distributed and persistently stored at several dozens of sites around the world, building the LHC[9] data grid. The destination sites are expected to provide the necessary middle-ware, so called Storage Elements, offering standard protocols to receive the data and optionally store it at the site specific Tertiary Storage Systems. Beside its actual functionality, discussed subsequently, the Storage Element software has to be able to fit into a large variety of environments. They are known to range from sites providing a single storage box of some Tera Bytes of data and nearly no maintenance personnel up to Tier I sites with estimated disk storage capacities reaching into the Peta Byte area. Moreover, sites expected to store data permanently may want to use their already existing Hierarchical Storage Management (HSM) System to drive the robotics. This requires the Storage Element to be aware of HSM Systems and to be able to manage external file copies. The wide range of scalability, from the very small to the limits of affordable storage, is one of the primary goals of dCache, the Storage Element introduced in this presentation. By being strictly compliant to standard data transfer and control protocols, like gsiFtp[17], xRootd[20] and the Storage Resource Manager protocol SRM[29], we are focusing on our second goal which is to make dCache available and useful beyond the borders of the High Energy Physics Community. Beside storing and preparing data for transfer, dCache provides a rich palette of functions to manage the available storage, as will be described subsequently. This includes replication of datasets on automated detection of busy storage components as well as optimization of access to tertiary storage systems.

1 Contributors

dCache is a joined effort between the German Elektronen-Synchrotron, DESY[1] in Hamburg and the Fermi National Accelerator Laboratory[2] near Chicago with significant distributions and support by the US Open Science Grid[8], the UK GridPP[7] organization, the Nordic Data Grid Facility[6] and CERN[3].

2 Introduction

As the abstract of this document implies, dCache is mass storage middleware, currently mainly utilized by the WLCG data handling community. This is basically for historical reasons and not due to any technical dependencies. The system has very little requirements to the underlying storage hardware. The actual storage nodes can be deployed on a variety of platforms, starting with redundant high end systems down to inexpensive mass production boxes. dCache focuses on standard protocols, which is especially true for the storage control protocol and the local and wide area transfer protocols. Those facts make dCache a community independent and generic data management tool. Therefore, this presentation is intended to encourage other communities to evaluate the possibilities of integrating the dCache technology into their mass data chain.

While developing dCache, we closely watch other products offing similar or complementary functionality. They range from systems simply providing fast access to disk storage up to technologies concentrating on driving tape robotics directly. We intentionally avoid comparing dCache which such systems in this presentation because, due the amount of systems out there, this is certainly out of the scope of this document.

The document is split into a section, disclosing a subsample of the dCache technical specification followed by a section, elaborating on upcoming important improvements and extensions.

3 Technical Specification

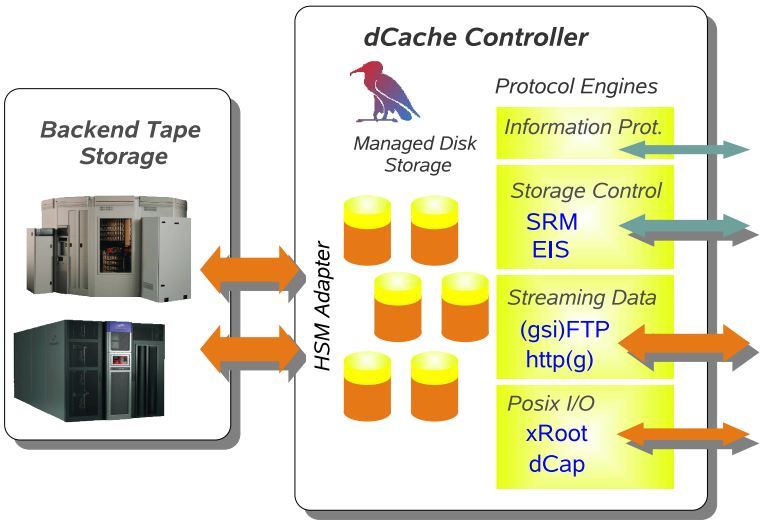

Figure 1: The big picture

Picture 1 provides a brief insight into the major dCache functional blocks.

- dCache offers common grid standard protocols
  - gsiFtp for wide area data transport.
  - dCap and xrootd for local area random data access.
  - SRM[14] for storage control
  - LDAP to publish status and storage attributes embedded in the GLUE schema.
- dCache provides an interface to various tertiary mass storage systems, e.g. TSM, enstore, OSM, dmf and HPSS.
- dCache utilizes regular filesystems (preferable xfs and zfs) on highly distributed, so called, pool nodes to store its data.
- The dCache core makes sure all transfers are scheduled correctly according to the rules provided to the dCache setup.

Subsequently we will discuss some of the features dCache already provides in more detail.

3.1 File name space and dataset location

dCache strictly separates the filename space[35][26] of its data repository from the actual physical location of the datasets. The filename space is internally managed by a database and interfaced to the user resp. to the application process by the nfs2/3[19] protocol and through the various ftp filename operations. The location of a particular file may be on one or more dCache data servers as well as within the repository of an external Tertiary Storage Manager. dCache transparently handles all necessary data transfers between nodes and optionally between the external Storage Manager and the cache itself. Inter dCache transfers may be caused by configuration or load balancing constrains. As long as a file is transient, all dCache client operations to the dataset are suspended and resumed as soon as the file is fully available. See the section on Chimera, later in this document, on an update of the file system engine.

3.2 Maintenance and fault tolerance

As a result of the name space and data separation, dCache data server nodes, subsequently denoted as pools, can be added at any time without interfering with system operation. Having a Tertiary Storage System attached, or having the system configured to hold multiple copies of each dataset, data nodes can even be shut down at any time. In both setups, the dCache system is extremely tolerant against failures of its data server nodes.

3.3 Data access methods

In order to access dataset contents, dCache provides a native protocol (dCap), supporting regular file access functionality. The software package includes a c-language client implementation of this protocol offering the posix *open, read, write, seek, stat, close* as well as the standard filesystem name space operations.

This library may be linked against the client application or may be pre-loaded to overwrite the file system I/O. The library supports pluggable security mechanisms where the GssApi (Kerberos) and ssl security protocols are already implemented. Additionally, it performs all necessary actions to survive a network or pool node failure. It is available for Solaris, Linux, Irix64 and windows. Furthermore, it allows to open files using an URL like syntax without having the dCache nfs file system mounted. A second posix like protocol, we support, is xRoot which is currently evaluated at various sites. In addition to this random access, various FTP dialects[29] are supported, e.g. GssFtp (kerberos)[18] and GsiFtp (GridFtp)[17]. An interface definition to dCache is provided, allowing other protocols to be easily implemented.

3.4 Tertiary Storage Manager connection

Although dCache may be operated stand alone, it can also be connected to one or more Tertiary Storage Systems. In order to interact with such a system, a dCache external procedure must be provided to store data into and retrieve data from the corresponding store. A single dCache instance may talk to as many storage systems as required. The cache provides standard methods to optimize access to those systems. Whenever a dataset is requested and cannot be found on one of the dCache pools, the cache sends a request to the connected Tape Storage Systems and retrieves the file from there. If done so, the file is made available to the requesting client. To select a pool for staging a file, the cache considers configuration information as well as pool load, available space and a *Least Recently Used* algorithms to free space for the incoming data. Data, written into the cache by clients, is collected and, depending on configuration, flushed into the connected tape system based on a timer or on the maximum number of bytes stored, or both. The incoming data is sorted, so that only data is flushed which will go to the same tape or tape set. Mechanisms are provided that allow giving hints to the cache system about which file will be needed in the near future. The cache will do its best to stage the particular file before it's requested for transfer. Space management is internally handled by the dCache itself. Files which have their origin on a connected tape storage system will be removed from cache, based on a Least Recently Used algorithm, if space is running short. Less frequently used files are removed only when new space is needed. In order to allow site administrators to tune dCache according to their local tape storage system or their migration and retrieval rules, dCache provides an open API to centrally steer all interactions with Tertiary Storage Systems.

3.5 Pool Attraction Model

Though dCache distributes datasets autonomously among its data nodes, preferences may be configured. As input, those rules can take the data flow direction, the subdirectory location within the dCache file system, storage information of the connected Storage Systems as well as the IP number of the requesting client and the data transfer protocol, the client is able to support.

Data flow direction are defined as
- getting the file from a client
- delivering a file to a client
- fetching a file from the Tertiary Storage System
- and transferring data between internal dCache pools.

The simplest setup would direct incoming data to data pools with highly reliable disk systems, collect it and flush it to the Tape Storage System when needed. Those pools could e.g. not be allowed to retrieve data from the Tertiary Storage System as well as deliver data to the clients. The commodity pools on the other hand would only handle data fetched from the Storage System and delivered to the clients because they would never hold the original copy and therefore a disk resp. node failure wouldn't do any harm to the cache. Extended setups may include the network topology to select an appropriate pool node. Those rules result in a matrix of pools from which the load balancing module, described below, may choose the most appropriate candidate. The final decision, which pool to select out of this set, is based on free space, age of file and node load considerations.

3.6 Load Balancing and pool to pool transfers

The load balancing module is, as described above, the second step in the pool selection process. This module keeps itself updated on the number of active data transfers and the age of the least recently used file for each pool. Based on this set of information, the most appropriate pool is chosen. This mechanism is efficient even if requests are arriving in bunches. In other words, as a new request comes in, the scheduler already knows about the overall state change of the whole system triggered by the previous request though this state change might not even have fully evolved. System administrators may decide to make pools with unused files more attractive than pools with only a small number of movers, or some combination. Starting at a certain load, pools can be configured to transfer datasets to other, less loaded pools, to smooth out the overall load pattern. At a certain point, pools may even re-fetch a file from the Tertiary Storage System rather than an other pool, assuming that all pools, holding the requested dataset are too busy. Regulations are in place to suppress chaotic pool to pool transfer orgies in case the global load is steadily increasing. Furthermore, the maximum numbers of replica of the same file can be defined to avoid having the same set of files on each node.

For various reasons (See figure 2), dCache might decide to internally replicate files. This process is handled transparently to the end user. Some of the reasons for replication are :
- As already described, dCache may consider to redistribute files from a highly loaded pool in order to reduce the access load of this particular pool.
- A dataset could reside on a pools from which a client is not allowed to read. (e.g. firewall setup).

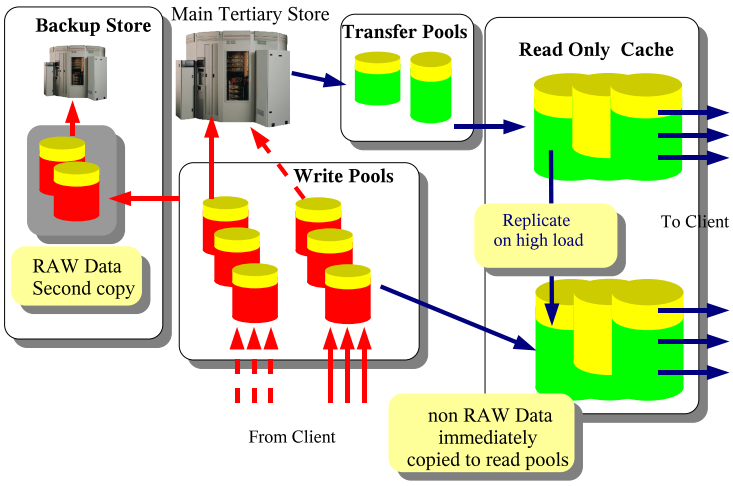

Figure 2: File hopping

- A second, safe copy of a dataset might be required until the system decides to store the data onto a Tertiary Storage System.
- In order to optimize Tertiary Storage System access, it might be required to support intermediate transfer pools, to decouple Tape Backend reads from client read operations.

3.7 File Replica Manager

The Replica Manager Module[28] enforces that at least N copies of each file, distributed over different pool nodes, must exist within the system, but never more than M copies. This approach allows to shut down servers without affecting system availability or to overcome node or disk failures. The administration interface allows to announce a scheduled node shut down to the Replica Manager so that it can adjust the $N < n < M$ interval prior to the shutdown.

3.8 Data Grid functionality

In order to comply with the definitions of a WLCG Storage Element, as sketched in picture 3, the storage fabric must provide the following interfaces :

- There must be a protocol for locally accessing data. dCache provides this by nfs mounting a server for file name operations but transferring the actual data via faster channels. Local Storage Elements, including dCache, hide this mechanism by being integrated into a local filesystem wrapper software provided by CERN, the *Grid File Access Layer, GFAL*[22].
- A secure wide-are transfer protocol must be implemented which, at the time being, is agreed to be GsiFtp, a secure Ftp dialect. Furthermore dCache offers kerberos based FTP as well as regular and secure http access.

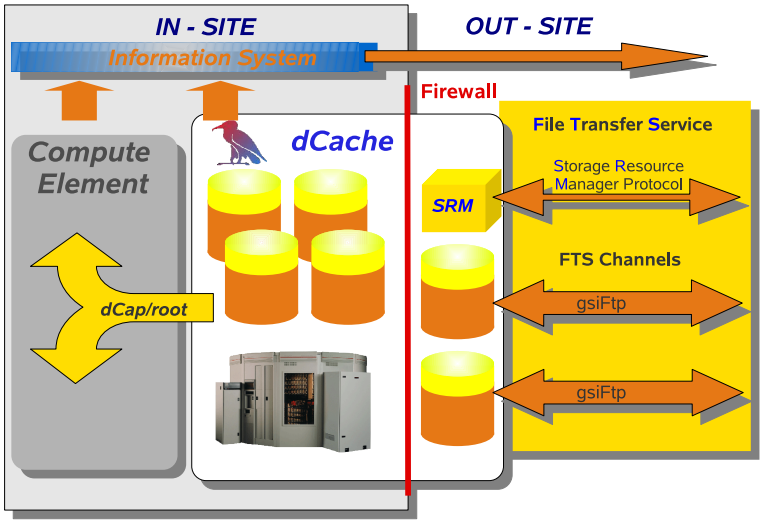

Figure 3: dCache as part of the WLCG grid

- To allow central services to select an appropriate Storage Element for file copy or file transfer requests, each Storage Element has to provide sufficient information about its status. This includes its availability as well as its total and available space. Currently this information is provided via the ldap protocol but this, for scalability reasons, is in process of being redesigned. In order to be independent of the actually distribution mechanism, dCache provides an interface to the *Generic Information Provider, GIP*. GIP[25] is responsible to make this information available to the connected grid middle ware.
- The forth area, defining a LCG Storage Element, is a protocol which makes a storage area a manageable. The interface is called the *Storage Resource Manager,SRM*[14]. Beside name space operations, it allows to prepare datasets for transfers directly to the client or to initiate third party transfers between Storage Elements. SRM takes care that transfers are retried in case they didn't succeed and handles space reservation and management. In addition, it protects storage systems and data transfer channels from being overloaded by scheduling transfers appropriately. The SRM doesn't do the transfer by itself, instead it allows to negotiate transfer protocols available by the data exchanging parties.

4 Scalability and maintenance efforts

Scalability is certainly a buzzword which is interpreted by implementors of large systems at their convenience. dCache is luckily in the good position having installations out which prove that the technology copes with overall storage

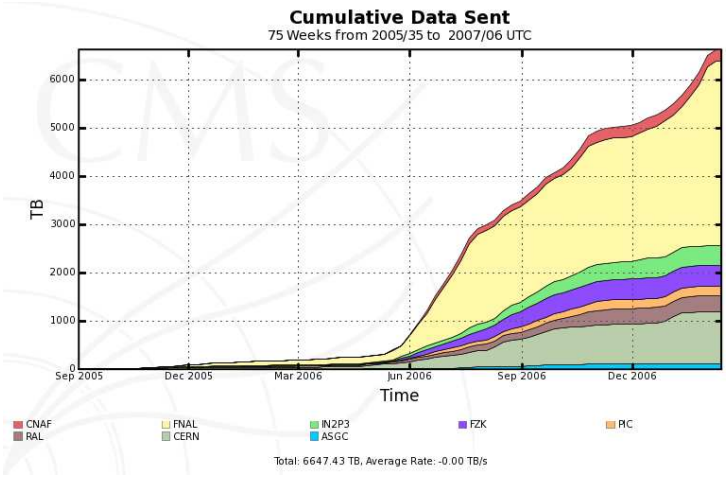

Figure 4: Data sent by Tier I centers (By courtesy of Frank Wuerthwein, UCSD)

spaces in the order of petabytes and which can deliver data out of those repositories in the several hundred Terabyte range per day.

Although we actively investigate in upcoming bottlenecks in case the amount of storage or the transfer rates are increasing significantly, currently our main areas of investigations are :

- Does dCache scale in the small ? Does it make sense to have dCache running on a very small amount of data with moderate overall transfer rates ?
- Is the number of sysadmins needed to run small to very large installations justifiable with the achieved profits ?

Lots of effort is currently going into solving those issues, not only by dCache.ORG itself but by groups like the d-grid initiative and the Open Science Grid[8]. We are convinced of having suitable solutions at hand before end of 2007.

5 Ongoing development

dCache is currently encountering two classes of limitations. Due to the fact that the user community has permanently increased, there are new requirements which haven't been though about initially. Second, because of the tremendous increase in data storage and access profile, certain bottlenecks could be identified. This section picks four of the most prominent planned improvements addressing the issues described above.

- **Chimera** Again, for historical reasons, the current file name space engine is only accessible through its nfs2/3 interface. The consequence is that even the dCache core has to traverse the nfs client, network and nfs server layer

in order to access dataset meta-data. It's obvious that this approach is a significant bottleneck, especially because arbitrary node/users, mounting the dCache nfs2 system can, by flooding the nfs layer with requests, interfere with the dCache core system. To overcome this deficiency, Chimera, a new file name space engine has been developed. Chimere essentially is a meta-data catalogue providing a file system view of its maintained objects.

- **Access control lists** The complexity of already existing grid communities, resp. Virtual Organizations, concerning data access permissions, exceeds the possibilities of the rather simple Unix authorization schema. Therefor an Access Control List module for Chimera is under development.
- **Improved Tertiary Storage System interface** The data stream, injected into the WLCG Tier I sites has to be forwarded to the back-end Tertiary Storage System. In order to cope with the special requirements of such systems and the concurrent performance deficiencies of available disk storage hardware a central component in dCache will be introduced to optimize access to the Tertiary Tape Storage Systems.
- **NFS 4.1** Although dCache provides two posix like data access protocols, it would certainly be desirable to support a protocol for which the client drivers are already part of the various operating systems. With the upcoming nfs 4.1 specification, this will become possible. Nfs 4.1 can make optimized use of highly distributed data sources and as to our current information, industry is very interested in this protocol and a lot of OS vendors already have nfs 4.1 clients ready. Moreover nearly all storage box vendors are working on server implementations. A major part of the nfs 4.1 server specification is already build into dCache. If nfs 4.1 will become widely accepted by storage system vendors and customers, this would significantly simplify the access to dCache data.

6 Conclusions

As shown in picture 4, dCache is already storing and transferring huge amounts of data for several years. We expect this technology to handle the majority of the LHC data in the years to come, mostly in the Tier I centers but as well in larger Tier II's. This fact, and the impressive feature-set of dCache should make other communities curious to invest some time to evaluate dCache as their storage middle-ware as well. Last but not least, with the described current developments in dCache, this technology is well prepared for the future

References

1. DESY : http://www.desy.de
2. FERMI : http://www.fnal.gov
3. CERN : http://www.cern.ch
4. Rutherford Appleton Laboratory : http://www.cclrc.ac.uk/
5. GridKA : http://www.gridka.de/

6. The Nordic Data Grid Facility : http://www.ndgf.org/
7. UK Computing for partical Physics http://www.gridpp.ac.uk/
8. The US Open Science Grid http://www.opensciencegrid.org/
9. Large Hadron Collider : http://lhc.web.cern.ch/lhc/
10. LHC Computing Grid : http://lcg.web.cern.ch/LCG/
11. Fermi Enstore http://www.fnal.gov/docs/products/enstore/
12. High Performance Storage System : http://www.hpss-collaboration.org/hpss/
13. Tivoli Storage Manager : http://www-306.ibm.com/software/tivoli/products/storage-mgr/
14. SRM : http://sdm.lbl.gov/srm-wg
15. dCache Documentation : http://www.dcache.org
16. dCache, the Book : http://www.dcache.org/manuals/Book
17. GsiFtp http://www.globus.org/ datagrid/deliverables/gsiftp-tools.html
18. Secure Ftp : http://www.ietf.org/rfc/rfc2228.txt
19. NFS2 : http://www.ietf.org/rfc/rfc1094.txt
20. NFS2 : http://www.ietf.org/rfc/rfc1094.txt
21. Cern CMS Experiment : http://cmsinfo.cern.ch
22. Grid GFAL http://lcg.web.cern.ch/LCG/peb/GTA/GTA-ES/Grid-File-AccessDesign-v1.0.doc
23. D-Grid, The German e-science program : http://www.d-grid.de
24. Patrick Fuhrmann et al. dCache, the Upgrade. Spring 2006, CHEP06, Mumbai, India
25. Lawrence Field et al. Grid Deployment Experiences: The path to a production quality LDAP based grid information system. Spring 2006, CHEP06, Mumbai, India
26. Tigran Mkrtchyan et al. Chimera. Spring 2006, CHEP06, Mumbai, India
27. Lars Schley, Martin Radicke et al. A Computational and Data Scheduling Architecture for HEP Application. Spring 2006, CHEP06, Mumbai, India
28. Alex KULYAVTSEV et al. Resilient dCache: Replicating Files for Integrity and Availability Spring 2006, CHEP06, Mumbai, India
29. Timur Perelmutov et al. Enabling Grid features in dCache Spring 2006, CHEP06, Mumbai, India
30. Abhishek Sinh Rana et al. gPLAZMA : Introducing RBAC Security in dCache Spring 2006, CHEP06, Mumbai, India
31. Patrick Fuhrmann et al. The TSM in the LHC Grid World Sep 2005, TSM Symposium , Oxford, UK
32. Patrick Fuhrmann, dCache, the commodity cache. Spring 2004, Twelfth NASA Goddard and Twenty First IEEE Conference on Mass Storage Systems and Technologies. Washington DC, USA
33. Timur Perelmutov, Storage Resource Managers by CMS,LCG. Spring 2004, Twelfth NASA Goddard and Twenty First IEEE Conference on Mass Storage Systems and Technologies. Washington DC
34. Michael Ernst et al. Managed Data Storage and Data Access Services for Data Grids. Sep 2004, CHEP04, Interlaken, Switzerland
35. Tigran Mkrtchyan et al. Chimera, the commodity namespace service. Sep 2004, CHEP04, Interlaken, Switzerland
36. Patrick Fuhrmann et al. dCache, LCG SE and enhanced use cases. Sep 2004, CHEP04, Interlaken, Switzerland
37. Michael Ernst, Patrick Fuhrmann et al. dCache. March 2003, CHEP03, San Diego, USA