dCache, the Overview

Patrick Fuhrmann[1] for the dCache team

Deutsches Elektronen Synchrotron
Notkestrasse 85, 22607 Hamburg

Abstract. In 2007, the most challenging high energy physics experiment ever, the *Large Hardon Collider(LHC)*, at CERN, will 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 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 to store it at the site specific Storage Systems. A major player in the set of Storage Elements is the *dCache/SRM* system. dCache/SRM has proven to be capable of managing the storage and exchange of several hundreds of terabytes of data, transparently distributed among dozens of disk storage nodes. One of the key design features of the dCache is that although the location and multiplicity of the data is autonomously determined by the system, based on configuration, cpu load and disk space, the name space is uniquely represented within a single file system tree. The system has shown to significantly improve the efficiency of connected tape storage systems, by caching, 'gather & flush' and scheduled staging techniques. Furthermore, it optimizes the throughput to and from data clients as well as smoothing the load of the connected disk storage nodes by dynamically replicating datasets on the detection of load hot spots. The system is tolerant against failures of its data servers which enables administrators to go for commodity disk storage components. Access to the data is provided by various standard protocols. Furthermore the software is coming with an implementation of the *Storage Resource Manager* protocol (SRM), which is evolving to an open standard for grid middleware to communicate with site specific storage fabrics.

1 Technical Overview

The intention of this publication is to descibe features, behaviour and applications of a storage middleware system, called the dCache/SRM[12][10].

The core part of the dCache has proven to combine heterogenous disk storage systems in the order of several hundred tera bytes and let its data repository appear under a single filesystem tree. It takes care of data hot spots, failing hardware and makes sure, if configured, that at least a minimum number of copies of each dataset resides within the system to ensure full data availability in case of disk server maintainance or failure. Furthermore, dCache supports a large set of standard access protocols to the data repository and its namespace.

If dCache is connected to a Tertiary Storage System, it optimized access to such a system by various technics. Currently Enstore[7], the Open Storage Manager (OSM), the High Performance Storage System (HPSS) and the Tivoli Storage Manager (TSM)[9] are supported by the dCache middleware.

Moreover, dCache/SRM supports all interfaces of the LCG storage element definition.

2 Contributors

dCache/SRM is a joined effort between the Deutsches Elektronen-Synchrotron[1] in Hamburg and the Fermi National Accelerator Laboratory[2] near Chicago with significant distributions and support from the University of California, San Diego, INFN, Bari as well as from the GridPP people at Rutherford Appleton Laboratory, UK[4] and CERN[3].

3 Technical Specification

3.1 File name space and dataset location

dCache strictly separates the filename space 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[16] 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.

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 preloaded 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 http like syntax without having the dCache nfs file system mounted. In addition to this native access, various FTP dialects are supported, e.g. GssFtp (kerberos)[15] and GsiFtp (GridFtp)[14]. An interface definition is provided, allowing other protocols to be implemented as well.

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. The cache defines data flow direction as getting the file from a client, delivering

a file to a client and fetching a file from the Tertiary Storage System. 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 refetch 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.

3.7 File Replica Manager

The Replica Manager Module 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 ¡ M interval prior to the shutdown.

3.8 Data Grid functionality

In order to comply with the definitions of a LCG Storage Element 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 mechnism by being integrated into a local filesystem wrapper software provided by CERN, the *Grid File Access Layer, GFAL*[20].

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.

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 independed of the actually distribution mechanism, dCache provides an interface to the *Generic Information Provider, GIP*. GIP 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*[10]. 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 Dissemination

At the time of this publication, dCache is in production at more than 35 locations in Europe and the US. The size of installations span from tapeless, single hosted systems to setups exceeding 200 TBytes of disk storage attached to tertiary storage. Typically LHC Tier I sites, like SARA (Amsterdam), IN2P3 (Lyon), gridKa (Karlsruhe), Brookhaven (US) and FermiLab (US) are running dCache installations connected to a variety of tape storage systems, while Tier II centers make use of the high availability resilient dCache mechanism. Peak throughputs, we learned about, have been in the order of 200 TBytes/day to more than 1000 clients. Large sites report about sustained data rates above 50 TBytes/day.

5 References

References

1. DESY : http://www.desy.de
2. FERMI : http://www.fnal.gov
3. CERN : http://www.fnal.gov

4. Rutherford Appleton Laboratory : http://www.cclrc.ac.uk/
5. Large Hadron Collider : http://lhc.web.cern.ch/lhc/
6. LHC Computing Grid : http://lcg.web.cern.ch/LCG/
7. Fermi Enstore http://www.fnal.gov/docs/products/enstore/
8. High Performance Storage System : http://www.hpss-collaboration.org/hpss/
9. Tivoli Storage Manager : http://www-306.ibm.com/software/tivoli/products/storage-mgr/
10. SRM : http://sdm.lbl.gov/srm-wg
11. CASTOR Storage Manager : http://castor.web.cern.ch/castor/
12. dCache Documentation : http://www.dcache.org
13. dCache, the Book : http://www.dcache.org/manuals/Book
14. GsiFtp http://www.globus.org/ datagrid/deliverables/gsiftp-tools.html
15. Secure Ftp : http://www.ietf.org/rfc/rfc2228.txt
16. NFS2 : http://www.ietf.org/rfc/rfc1094.txt
17. Fermi CDF Experiment : http://www-cdf.fnal.gov
18. GridKA : http://www.gridka.de/
19. Cern CMS Experiment : http://cmsinfo.cern.ch
20. Grid GFAL http://lcg.web.cern.ch/LCG/peb/GTA/GTA-ES/Grid-File-AccessDesign-v1.0.doc
21. D-Grid, The GErman e-science program : http://www.d-grid.de