dCache, the commodity cache

Patrick Fuhrmann
Deutsches Elektronen Synchrotron
22607 Hamburg, Germany
Notkestrasse 85
Tel: +49-40-8998-4474, Fax: +49-40-8994-4474
e-mail: patrick.fuhrmann@desy.de

1. Abstract

The software package presented within this paper 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, through 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 ftp dialects, including gridftp, as well as by a native protocol, offering regular file system operations like open/read/write/seek/stat/close. 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.

2. Contributors

The software is being developed by the Deutsches Elektronen Synchrotron (DESY) in Hamburg, Germany[1] and the Fermi National Laboratory, Batavia Chicago,IL, USA [2].

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 [9] 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.

3.2Maintenance and fault tolerance

As a result of the name space – 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. Under those conditions, the cache system is extremely tolerant against failures of its data server nodes.

3.3Data 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 calls. This library may be linked against the client application or may be preloaded to overwrite the file system I/O operations. 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) [8] and GsiFtp (GridFtp) [7]. An interface definition is provided, allowing other protocols to be implemented as well.

3.4Tertiary 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. Space is created only when needed. No high/low watermarks are used.

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 /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. Each row of the matrix contains pools with similar attraction. Attraction decreases from top to bottom. Should none of the pools in the top row be available, the next row is chosen, a.s.o.. Optionally, stepping from top to bottom can be done as long as the candidate of row 'n' is still above a certain load. The final decision, which pool to select 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 the overall load pattern. At a certain point, pools may even fetch a file from the Tertiary Storage System again, if 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

A first version of the so called Replica Manager is currently under evaluation. This 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.

4. Data Grid functionality

In the context of the LHC Computing Grid Project [4], a Storage Element describes a module providing mass data to local Computing Elements. To let a local Storage System look like a Storage Element, two conditions must be met. Storage Elements must be able to communicate to each other in order to exchange mass data between sites running different Storage System and Storage Elements have to provide local data through standard methods to allow GRID jobs to access data files in a site independent manner. The first requirement is covered by a protocol called the Storage Resource Manager, SRM [3], defining a set of commands to be implemented by the local Storage System to enable remote access. It mainly covers queries about the availability of datasets as well as commands to prepare data for remote transfer and to negotiate appropriate transfer mechanisms. dCache is providing an SRM interface and has proven to be able to talk to other implementations of the SRM. A dCache system at FERMI is successfully exchanging data with the CASTOR Storage Manager at CERN using the SRM protocol for high level communication and GridFtp for the actual data transfer. The second requirement, to make local files available to Grid Applications, is approached by the G-File initiative, a quasi standard as well. It offers well defined, posix like function calls that allow site independent access to files held by the local Storage Element. Optionally G-File can talk to other grid modules to register imported files or files being exportable. G-File developers at CERN have successfully linked the g-file library against the dCache dCap library.

5. Dissemination

In the meantime, dCache is in production at various locations in Europe and the US. The largest installation is, to our knowledge, the CDF system at FERMI [2]. 150 Tbytes are stored on commodity disk systems and in the order of 25 Tbytes have been delivered to about 1000 clients daily for more than a year. FERMI dCache installations are typically connected to ENSTORE [11], the FERMI tape storage system. CDF is operating more than 10 tape-less dCache installations outside of FERMI, evaluating the dCache Replica Manager. The US devision of the LHC CMS [13] experiment is using the dCache as Grid Storage Element and large file store in the US and Europe. At DESY, dCache is connected to the Open Storage Manager (OSM) and serving data out of 70 Tbytes of disk space. The German LHC Grid Tier 1 center in Karlruhe (GridKa,[12]) is in the process of building a dCache installation as Grid Storage Element, connected to their Tivoli Storage Manager [16] installation.

6. References

[1] DESY : http://www.desy.de
[2] FERMI : http://www.fnal.gov
[3] SRM : http://sdm.lbl.gov/srm-wg
[4] LCG : http://lcg.web.cern.ch/LCG/

[5] CASTOR Storage Manager : http://castor.web.cern.ch/castor/
[6] dCache Documentation : http://www.dcache.org
[7] GsiFtp : http://www.globus.org/datagrid/deliverables/gsiftp-tools.html
[8] Secure Ftp : http://www.ietf.org/rfc/rfc2228.txt
[9] NFS2 : http://www.ietf.org/rfc/rfc1094.txt
[10] Fermi CDF Experiment : http://www-cdf.fnal.gov
[11] Fermi Enstore : http://www.fnal.gov/docs/products/enstore/
[12] GridKA : http://www.gridka.de/
[13] Cern CMS Experiment : http://cmsinfo.cern.ch
[14] Cern LHC Project : http://lhc.web.cern.ch/LHC
[15] Grid g-file :
http://lcg.web.cern.ch/LCG/peb/GTA/GTA-ES/Grid-File-AccessDesign-v1.0.doc
[16] Tivoli Storage Manager :
http://www-306.ibm.com/software/tivoli/products/storage-mgr/