The Tivoli Storage Manager in the Large Hardron Collider Grid world

Patrick Fuhrmann, for the dCache team

DESY, Hamburg, HH 22607, Germany

In 2007, the most challenging high energy physics experiment ever, the LHC at CERN, will go online and will produce a sustained stream of data in the order of the content of one ordinary CD every two seconds, which would result in a stack of CDs as high as the Eiffel Tower once per week. For various political and technical reasons, this data is, while produced, distributed and persistently, tape based stored at several dozens of sites around the world, building the LHC data grid. Those sites are expected to provide the necessary middle-ware, abstractly called a Storage Element, talking the agreed protocols to receive the data and storing it at the site specific Hierarchical Storage Systems. One implementation of a Storage Element, installed at the majorities of sites, is the dCache/SRM. The dCache/SRM system has been designed and implemented in close collaboration between the Deutsches Elektronen-Synchrotron in Hamburg and the Fermi National Accelerator Laboratory near Chicago. Beside a rich set of Hierarchical Storage Systems, like Enstore, OSM and HPSS, dCache supports the TSM to store and retrieve data to/from Tape. At the time of this talk, the major German LHC site, GridKa in Karlsruhe, has connected its dCache/SRM system with its TSM instance. This talk will concentrate on the opportunity for the TSM system to become the tape backed for LHC Storage Elements.

High energy and particle physics background

Although already more than 2000 years ago, philosophers suspected matter to consist of an endless number of very small solid particles, deeper insight into the building blocks of our known world could only be achieved after having technologies at hand, allowing to resolve structures below the subatomic scale. Governed by physics laws, those structures only unfold providing energy densities reciprocally proportional to the size of those structures. The worlds largest installation of such a microscope, using superconducting magnets, is the **L**arge **H**ardron **C**ollider at **CERN**, next to Geneva in Switzerland. A 27 Km tunnel holds two ring pipes equipped with supercooling magnets, accelerating bunches of protons to nearly the speed of light and letting them collide at an energy of 14 TeV. This is the highest energy achieved ever by any accelerator in the world as well as the most intense beam. At four locations within the beam, huge detectors are placed detecting particles produced during beam collisions.

The LHC Computing Grid

Knowing that beam collisions will happen at a rate of 800 million times a second and that one detector event may hold up to 20 physical events, it becomes clear that computer science faces the challenge of processing and storing data two orders of magnitude larger than they did for known physics experiments. This in mind, the LHC Computing Grid Group was formed, targeting computing challenges common to all LHC experiments. Although a couple of computing patterns would have covered those issues, a **Tier** approach has been chosen. Within this design, the raw data source, namely CERN, builds the **Tier 0** centre surrounded by only very few **Tier 1** centres per country. Those deliver data to **Tier 2** centres still providing significant CPU and storage resources. The main difference between Tier 0/1 and Tier 2 sites, at least under the storage perspective, is certainly that Tier 1 centres are supposed to persistently store precious data while Tier 2 centres may remove data when space is

running short. The current understanding of persistent storage is still tape technology. All those Tier centres are already existing sites running their own compute farms and storage fabrics. So, sufficiently flexible interfaces to compute and storage systems had to be defined to allow interoperability of the tier tree without forcing the local sites to change their existing software stack. The storage unit supposed to store permanent as well as temporary data is interfaced by the so called **S**torage **E**lement (SE). All upper layer middle ware systems expect a local storage system to comply with this interface.

The Storage Element definition

Being a LCG Storage Element mainly covers four areas.

There must be a protocol for locally accessing data. This is mostly done by nfs mounting a server for file name operations but transferring the actual data via faster channels.

A secure wide-are transfer protocol must be implemented which, at the time being, is agreed to be GsiFtp, a secure Ftp dialect.

To allow central services to select an appropriate Storage Element for the 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.

The forth area, defining a LCG Storage Element, is a protocol which makes a storage area a manageable one. The interface is called the Storage Resource Manager (SRM). Beside name space operations it allows to prepare datasets for transfers directly to the client or to initiate third party transfers between Storage Elements. The 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.

As already mentioned, LCG Tier 0/1 Storage Elements are required to be connected to Tape Storage Systems or Hierarchical Storage Managers to allow persistent storage. The following paragraphs describe a piece of middle ware, called the dCache®, satisfying all Storage Element requirements and being able to use the Tivoli Storage Manager (TSM®) as its persistent storage system.

The dCache/SRM Storage Element

The dCache core software essentially allows to combine a huge amount of individual disk storage systems, called pools, and let them appear under a single file system tree. Directory and file structure are independent of the actually location of the datasets within the storage server array. Although each file has a single entry within the file system tree, it may have various copies inside and outside of dCache. Moreover, dCache has various automatic mechanisms to replicate files based on load or the need for secure copies.

The file system tree is exposed by the nfs2/3 protocol allowing all regular name space operations and a slightly limited set by the FTP and dCap protocol. Posix like file transfers are supported by the dCap protocol. A c-language implementation of dCap is provided with the dCache system. Streaming transfers, tuned for wide area connections, allowing multiple channels per dataset, are supported though FTP. Both, dCap and FTP support gsi and kerberos authentication.

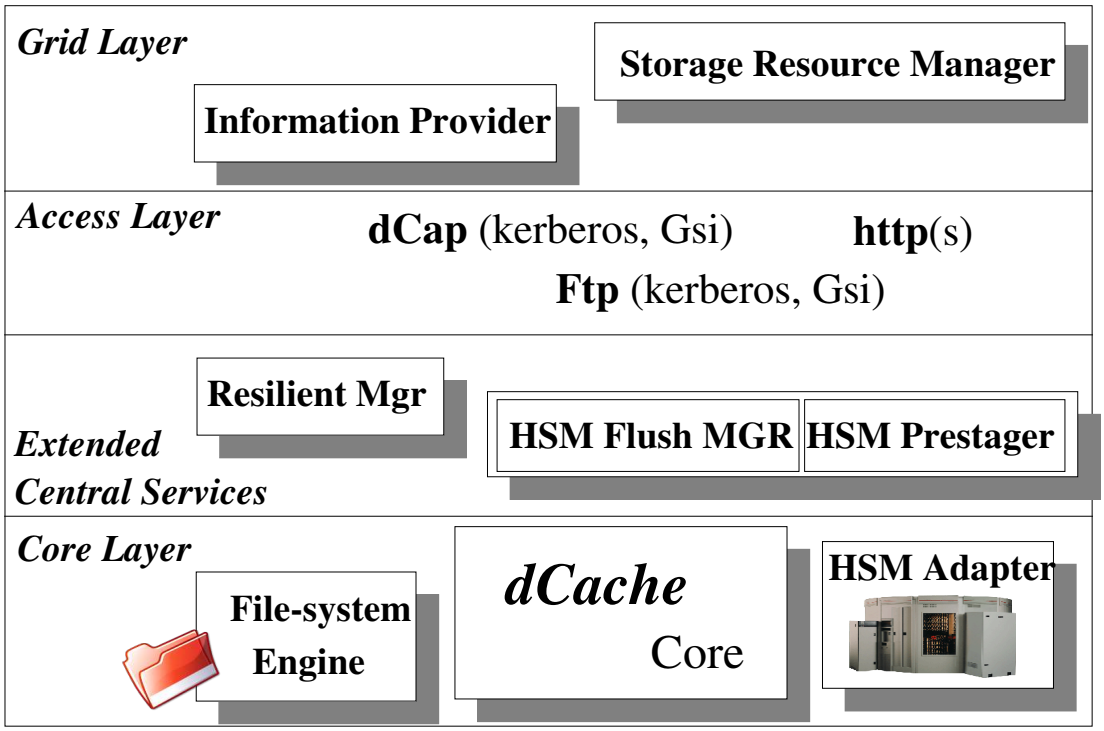
dCache offers sophisticated methods for assigning storage servers to request properties, like client ip number, file system tree location and data transfer direction. This mechanism includes the definition of fall back servers for high availability purposes. Moreover, dCache uses space and cpu load information from all active servers to finally find an appropriate one for each request.

dCache/SRM supports the agreed set of SRM functionality to interact with other SRMs to exchange data in the LCG context. Nevertheless, the SRM definition, and consequently the implementation of the SRM within dCache, is in permanent flux to meet upcoming requirements.

dCache and the Tivoli Storage Manager

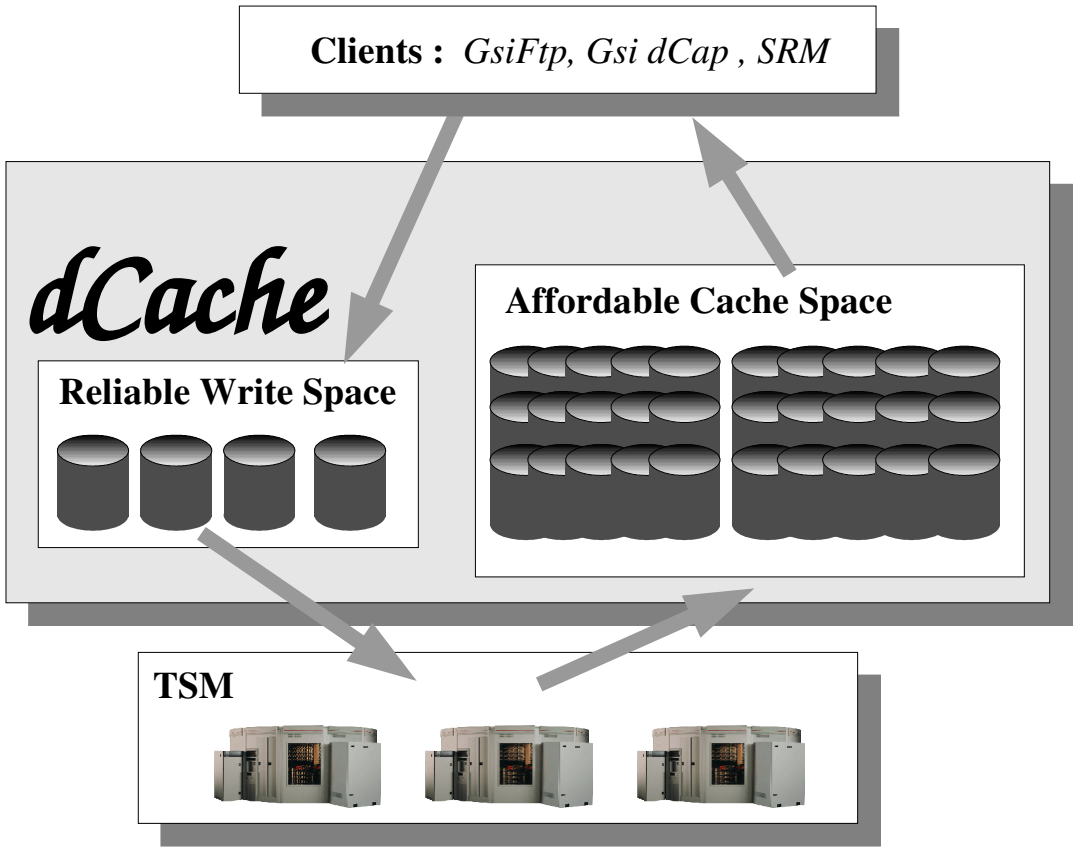
dCache can be configured to have a connection to a Tape Storage Library, like the **TSM** in archive mode, **HPSS** and many more. Very little requirements need to be met allowing dCache to interact with such a system. Having the TSM connected, dCache collects incoming data on previously defined **safe**

disks, grouped by ***TSM File Spaces,*** assigned to sections within the dCache file system tree. According to customizable rules, dCache flushes those files, based on time and/or space considerations. After having data flushed into the TSM, the data is declared 'cached' and is automatically removed by the system when space is running short. On a request for a file, no longer on disk within dCache, the TSM is asked to restore the file to one of dCache file servers. The original file request is put on hold until the restore operation succeeds and proceeds afterwards. No manual interaction is required for any of the described operations. Although the current functionality is fine for most applications, we are going to review the dCache - Tape Storage interconnection modules to make the dCache more aware of Tape System behaviours and requirements.



System installations

During the last two years, dCache has becoming increasingly popular for various reasons. One of the advantages, often mentioned, is the ability of dCache, to make site specific storage systems aware of Grid transfer and management protocols without modification of the local system which is a strong requirement, sites has to meet when wanting to become part of the upcoming LHC data storage and analysis chain. This is a great opportunity for the TSM to be the default Tape Back-end for dCache because TSM is available at a lot of sites as backup system anyway. The first successful dCache/TSM installation has been set up at the German LHC Tier 1 centre (gridKa) in Karlsruhe. Having proved this to be successful, other sites have shown interest in that approach as well. Very likely the Central Institute for Applied Mathematics (ZAM) in Juelich will follow next. There are at least three Institutes in France, Canada and the Netherlands from which we know that they are evaluating the dCache/TSM approach.



Acknowledgements

Beside all the people working on the dCache I would like to thank Ralf Lueken and Tigran Mkrtchyan from DESY for doing the TSM integration and Doris Ressmann from gridKa, Karlsruhe for being the guinea pig.

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 GFAL 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/