A Distributed Rate-Adapting Buffer Cache for Mass Storage Systems

P. Fuhrmann[1], *M. Gasthuber*[1], *C.G.Waldman*[2]

[1] Deutsches Elektronen-Synchrotron (DESY),Germany
[2] Fermi National Laboratory (FNAL),USA

Abstract

To address mass storage needs common to both DESY and Fermilab, a rate-adapting buffer cache is under development. The aim is to maximize utilization of limited tiary storage resources by the use of prefetching (read-ahead) and caching of files based on access statistics, as well as decoupling network transfer rate from tape I/O rate. A modular architecture allows cache capacity to be dynamically reconfigured by adding or removing ”cache pools”, as well as allowing the system to work with several different mass storage systems. PNFS is used to provide the underlying namespace and to store cache-related metadata.

Keywords: cache, mass storage, FNAL, DESY, enstore, osm, Eurostore, pnfs

1 The premise

As service providers for huge amounts of tape space, the DESY and Fermilab mass storage groups and certainly many other storage devisions in the HEP world are facing an overproportional increase in storage space requirements as well as in the number of concurrent accesses to their tape repositories. These requirements are by no means compensated by the transaction rate of the currently available robotic tape libraries nor by their prices. Essentially, Tape Storage providers are permanently facing two powerful enemies. The Storage clients, treating the system as a large disk, and the robotic tape libraries, suffering from the intrinsic three-folded unshareability of its components (robot, drive and tape). The ideal client would access tape data at the maximum speed the drive is capable of. It would adjust its average filesize to minimize the NonI/O versus I/O ratio. For the same reason it would accumulate files up to a reasonable amount before writing them to the Storage Manager and possibly it would sort read requests according to the order the files reside on tape. In addition clients should store files themself on disk if these files are known to be accessed again. On the other hand, an ideal Storage Manager should be able to simulate unlimited diskspace with no time offset accessing a particular file. These opposing requirements are hidden, as long as the available resources cover the ongoing requests. At DESY we are already observing disharmonies every here and then which still can be solved by helping our clients to make better use of the storage system. We assume that within year 2000 this approche will certainly fail. Fermilab is expecting the same trend within this year as well.

It’s pretty obvious that neither the speed of the robots nor the requirements and the behaviour of the storage clients can easily be influenced by the local storage groups. Consequently, DESY and Fermilab took a joined effort to produce an intermediate layer which helps to disarm the converging requirements listed above.

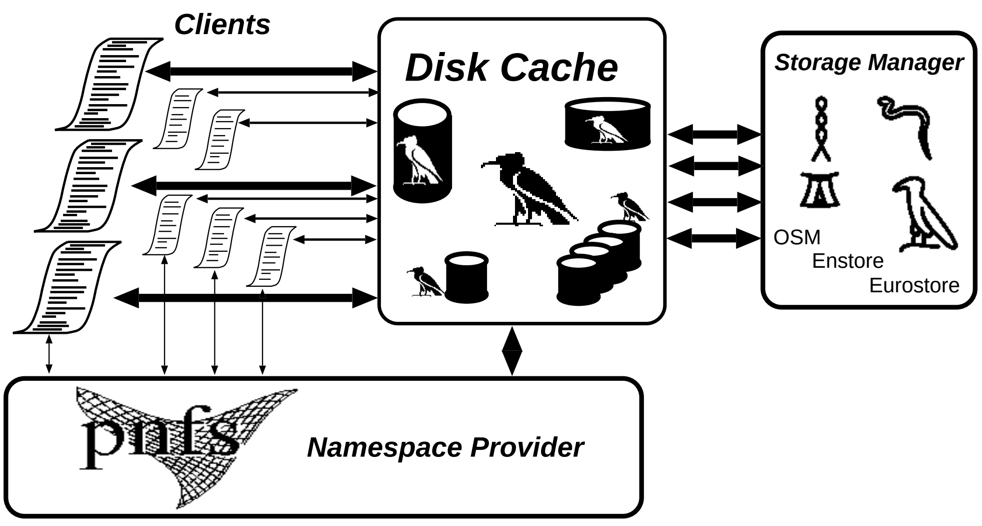

Figure 1: The initial Setup

2 Specification, Requirements and Constraints

2.1 The Specification

From the preliminary considerations, DESY and Fermilab agreed on having the following properties implemented in a production version of the *Disk Cache*, subsequently denoted as *DC*. They listed properties are discussed later on in more detail.

- **Rate-Adaptation** The datatransfer between the *DC* and the Storage Manager has to run on the drives maximum bandwidth, while the client may chose any convenient data rate to the *DC*.
- **Deferred Write** On a PUT[1], the *DC* has to store data up to a particular amount or until a time limit is reached before the data is actually sent to the Storage Manager in one chunk.
- **Read-ahead** The *DC* reads the requested file from a tape together with a number of subsequent files which may not have been requested yet.
- **Staging** Files or file groups may stay in the *DC* for a configurable amount of time if it's assumed that these files may be used again within a reasonable time interval.

2.2 The Requirements

In addition to the actual specification of the properties of a *DC*, there are requirements which reflects the usage of such a device in the real world.

- The usage of a *DC* only makes sense in an environment where there is a particular pressure on the Storage Manager. Consequently, because the purpose of the *DC* is to buffer this pressure, it has to be hightly scalable. It's our impression that this can only be achieved by allowing the *DC* to be spread among an arbitrary number of hosts.
- On a PUT, the *DC* will acknowledge the reception of the data on behave of the Storage Manager before the data is actually on tape. This behaviour requires that the data is at least as securely stored as on tape. Which means that the particular *DC storage areas* which are

[1]The notation PUT/GET in conjunction with a Storage Manager describes the dataflow seen from the client. So, a PUT means data going into the Storage Manager.

allowed to store PUT data must provide sufficently secure disks (RAID, MIRROR), while other parts may only serve read requests and therefore are not even required to be under control of the central Mass Storage Service. The system must be able to distinguish among these different pools.

2.3 The Constraints

Obviously, the *DC* must fit into the existing environment at both development sites, DESY and Fermilab, which on one hand limits the possibilities but on the other hand enforces an amount of flexibity which will make it easier to adopt the *DC* to other Storage Managers as well.

- The interconnection of the *DC* with the on-site Storage Manager should be limited to the absolute necessary. This allows us to support the Fermilab **enstore[5]**, the DESY **OSM** as well as **Eurostore[3]**, a Storage System currently under development at DESY.
- To use a raw Storage Manager there will be always some kind of filesystem-like namespace provider necessary. At Fermilab and DESY *Pnfs[1]* is in charge of this task. So it's not surprising that the *DC* requires some of its features.

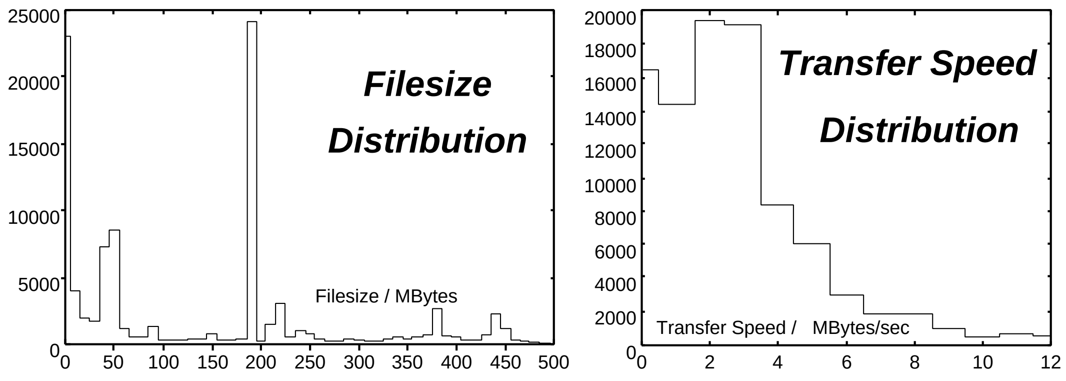

Figure 2: A typical Filesize/TransferSpeed Distribution

3 Proving the theory

Figure 2 represents nearly 100000 consecutive requests assembled within 3 weeks from the DESY STK powderhorn robot equiped with 10 9840 Drives. The datarate distribution, which peaks around 3 MBytes/sec for a drive type, which provides an average performance of about 9 Mbytes/sec, gives an impression of the pathological client behaviour. The same is true for the filesizes. About half the number of files do not exceed 50 Mbytes, which means a transfer time of 6 seconds against a mount/load/unload/dismount time of about 30 seconds.

To check the improvements of a hypothetical disk cache, we first ran the *Rate-Adaptation* on the sample. As a result we found that the drive/tape usage was reduced to 45% of its original value, mount/dismount included. What appears to be a good result for the drive/tape will certainly push the robot to its limit because the number of mount/dismounts will roughly double. To partially relieve the robot of this burden, we assumed the *DC* to store data up to 400 Mbytes or until 4 hours have passed per StorageGroup[2], before writing the data to the Storage Manager in one chunk. (*DeferredWrite*). At no time, more than 30 GBytes of disk space had been necessary to store the data intermediately. This simple scheme reduced the mount/dismount cycles to a third of

[2]A StorageGroup denotes a set of tapes. It's the smallest unit the client can specify when writing data.

its original value. The *staging* feature has not been investigated yet, because each experiment is running its own staging system. Our intend is to replace those implementations by the *Disk Cache*.

Nevertheless, the results we were able to retrieve, convinced us of putting more efford into converting the *disk cache prototype* into a production version.

4 Under the hood

Finally there are some details listed about the *Disk Cache Project* and its implementation.

- The *Disk Cache Prototype* has been designed and implemented in Java by Charles G. Waldman[3] at DESY/Hamburg.
- The *Disk Cache* is built on top of a communication package, the *DESY-CELLS*, which has been developed by DESY-IT and which is the master building block of the *Eurostore[2] Software* as well.
- The Storage Areas of the *Disk Cache* can be distributed among an arbitrary number of hosts. Areas can be added and removed dynamically without interference with the running system. In addition, a mapping can be specified between the *Disk Cache Storage Areas* and some units (*StorageGroup,File Family*) within the Storage Manager. It allows to direct PUT requests to secure disks (e.g. RAID) and to have different user groups using their private storage areas.
- Charles has chosen an exponential aging function for files in the *Disk Cache Storage Areas*. Frequently a *remove-candidate* list is produced from which files are taken to free disk space when actually needed.
- Charles has implemented an FTP server to access the prototype. DESY will add an *RFIO-like* access method. It is planned to produce a shared library for the clients talking this protocol. This library will overwrite the libc i/o functions to allow arbitrary applications to access the *DC* without beeing recompiled.
- Statistical information about files are exclusively stored in *Pfns* (no additional database), which cares about the synchronization with the namespace and keeps these information even if the file is currently not in the cache.
- It is planned to allow the assignment of a kind of *StickyBit* to files or directories[4] to inform the *DC* that these files should stay on disk regardless of the aging function.

References

1 P.Fuhrmann, “A Perfectly Normal Namespace for the DESY OSM”, CHEP’97, Berlin, Spring 1997.

2 D.Roweth, P.Fuhrmann, M.Gasthuber, ”Eurostore - Design and First Results”, 16th IEE Symposium on Mass Storage Systems, 7th NASA Goddard Space Flight Center Conference on Mass Storage, San Diego, USA, March, 1999

3 D.Roweth, P.Fuhrmann, M.Gasthuber, ”The EuroStore Project - Results and deployment @DESY”, CHEP2000, Padova, Spring 2000.

4 S.Brand,P.Fuhrmann, ”A distributed disk layer for mass storage at DESY”, Computer Physics Communications 110(1998) 131-133

5 D.Petravick et al. , ”Enstore Technical Design Document”, Joint Projects Document JP0026, Fermi National Laboratory.

[3] Charles is a member of the IT devision of the Fermi National Laboratory. He spent about three months at DESY/Hamburg to get the prototype finished.

[4] Pnfs allows to assign any kind of information to the filename entries.