CACHING: ACCELERATION OF HARD DISK ACCESS

In all fields of computer systems, caches are used to speed up slow operations by operating them from the cache. Specifically in the field of disk subsystems, caches are designed to accelerate write and read accesses to physical hard disks. In this connection we can differentiate between two types of cache: (1) cache on the hard disk (Section 2.6.1) and (2) cache in the RAID controller. The cache in the RAID controller is subdivided into write cache (Section 2.6.2) and read cache (Section 2.6.3).

3.1.1 Cache on the hard disk

Each individual hard disk comes with a very small cache. This is necessary because the transfer rate of the I/O channel to the disk controller is significantly higher than the speed at which the disk controller can write to or read from the physical hard disk. If a server or a RAID controller writes a block to a physical hard disk, the disk controller stores this in its cache. The disk controller can thus write the block to the physical hard disk in its own time whilst the I/O channel can be used for data traffic to the other hard disks. Many RAID levels use precisely this state of affairs to increase the performance of the virtual hard disk.

Read access is accelerated in a similar manner. If a server or an intermediate RAID controller wishes to read a block, it sends the address of the requested block to the hard disk controller. The I/O channel can be used for other data traffic while the hard disk controller copies the complete block from the physical hard disk into its cache at a slower data rate. The hard disk controller transfers the block from its cache to the RAID controller or to the server at the higher data rate of the I/O channel.

3.1.2 Write cache in the disk subsystem controller

In addition to the cache of the individual hard drives many disk subsystems come with their own cache, which in some models is gigabytes in size. As a result it can buffer much greater data quantities than the cache on the hard disk. The write cache should have a battery backup and ideally be mirrored. The battery backup is necessary to allow the data in the write cache to survive a power cut. A write cache with battery backup can significantly reduce the write penalty of RAID 4 and RAID 5, particularly for sequential write access (cf. Section 2.5.4

'RAID 4 and RAID 5: parity instead of mirroring'), and smooth out load peaks. Many applications do not write data at a continuous rate, but in batches. If a server sends several data blocks to the disk subsystem, the controller initially buffers all blocks into a write cache with a battery backup and immediately reports back to the server that all data has been securely written to the drive. The disk subsystem then copies the data from the write cache to the slower physical hard disk in order to make space for the next write peak.

3.1.3 Read cache in the disk subsystem controller

The acceleration of read operations is difficult in comparison to the acceleration of write operations using cache. To speed up read access by the server, the disk subsystem's controller must copy the relevant data blocks from the slower physical hard disk to the fast cache before the server requests the data in question. The problem with this is that it is very difficult for the disk subsystem's controller to work out in advance what data the server will ask for next. The controller in the disk subsystem knows neither the structure of the information stored in the data blocks nor the access pattern that an application will follow when accessing the data. Consequently, the controller can only analyse past data access and use this to extrapolate which data blocks the server will access next. In sequential read processes this prediction is comparatively simple, in the case of random access it is almost impossible. As a rule of thumb, good RAID controllers manage to provide around 40% of the requested blocks from the read cache in mixed read profiles. The disk subsystem's controller cannot further increase the ratio of read access provided from the cache (pre-fetch hit rate), because it does not have the necessary application knowledge. Therefore, it is often worthwhile realising a further cache within applications.

For example, after opening a file, file systems can load all blocks of the file into the main memory (RAM); the file system knows the structures that the files are stored in. File systems can thus achieve a pre-fetch hit rate of 100%. However, it is impossible to know whether the expense for the storage of the blocks is worthwhile in an individual case, since the application may not actually request further blocks of the file.