INSTANT COPIES

Instant copies

Instant copies can virtually copy data sets of several terabytes within a disk subsystem in a few seconds. Virtual copying means that disk subsystems fool the attached servers into believing that they are capable of copying such large data quantities in such a short space of time. The actual copying process takes significantly longer. However, the same server, or a second server, can access the virtually copied data after a few seconds (Figure 2.18).

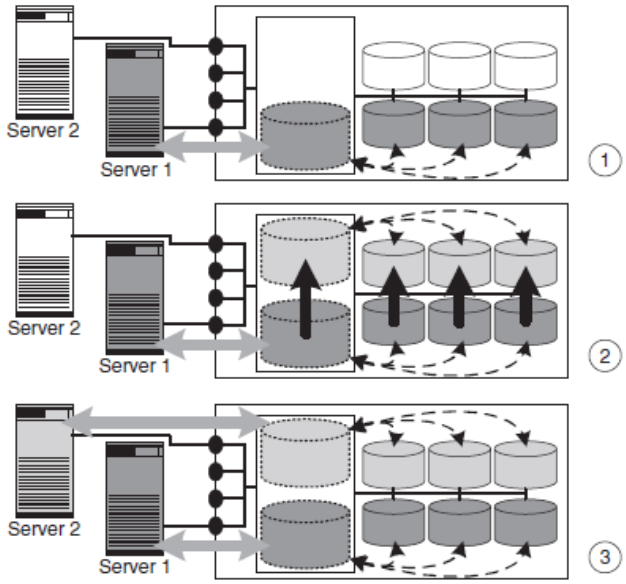

Figure 2.18 Instant copies can virtually copy several terabytes of data within a disk subsystem in a few seconds: server 1 works on the original data (1). The original data is virtually copied in a few seconds (2). Then server 2 can work with the data copy, whilst server 1 continues to operate with the original data (3).

Instant copies are used, for example, for the generation of test data, for the backup of data and for the generation of data copies for data mining. Based upon the case study in Section 1.3 it was shown that when copying data using instant copies, attention should be paid to the

consistency of the copied data. Sections 7.8.4 and 7.10.3 discuss in detail the interaction of applications and storage systems for the generation of consistent instant copies.
There are numerous alternative implementations for instant copies. One thing that all implementations have in common is that the pretence of being able to copy data in a matter of seconds costs resources. All realisations of instant copies require controller computing time and cache and place a load on internal I/O channels and hard disks. The different implementations of instant copy force the performance down at different times. However, it is not possible to choose the most favourable implementation alternative depending upon the application used because real disk subsystems only ever realise one implementation alternative of instant copy.

In the following, two implementation alternatives will be discussed that function in very different ways. At one extreme the data is permanently mirrored (RAID 1 or RAID 10). Upon the copy command both mirrors are separated: the separated mirrors can then be used independently of the original. After the separation of the mirror the production data is no longer protected against the failure of a hard disk. Therefore, to increase data protection, three mirrors are often kept prior to the separation of the mirror (three-way mirror), so that the production data is always mirrored after the separation of the copy. At the other extreme, no data at all is copied prior to the copy command, only after the instant copy has been requested. To achieve this, the controller administers two data areas, one for the original data and one for the data copy generated by means of instant copy. The controller must ensure that during write and read access operations to original data or data copies the blocks in question are written to or read from the data areas in question. In some implementations it is permissible to write to the copy, in some it is not. Some implementations copy just the blocks that have actually changed (partial copy), others copy all blocks as a background process until a complete copy of the original data has been generated (full copy).

In the following, the case differentiations of the controller will be investigated in more detail based upon the example from Figure 2.18. We will first consider access by server 1 to the original data. Read operations are completely unproblematic; they are always served from the area of the original data. Handling write operations is trickier. If a block is changed for the first

time since the generation of the instant copy, the controller must first copy the old block to the data copy area so that server 2 can continue to access the old data set. Only then may it write the changed block to the original data area. If a block that has already been changed in this manner has to be written again, it must be written to the original data area. The controller may not even back up the previous version of the block to the data copy area because otherwise the correct version of the block would be overwritten.

The case differentiations for access by server 2 to the data copy generated by means of instant copy are somewhat simpler. In this case, write operations are unproblematic: the controller always writes all blocks to the data copy area. On the other hand, for read operations it has to distinguish whether the block in question has already been copied or not. This determines whether it has to read the block from the original data area or read
it from the data copy area and forward it to the server.

The subsequent copying of the blocks offers the basis for important variants of instant copy. Space-efficient instant copy only copies the blocks that were changed (Figure 2.19). These normally require considerably less physical storage space than the entire copy. Yet the exported virtual hard disks of the original hard disk and the copy created through space-efficient instant copy are of the same size. From the view of the server both virtual disks continue to have the same size. Therefore, less physical storage space is needed overall and, therefore, the cost of using instant copy can be reduced. Incremental instant copy is another important variant of instant copy. In some situations such a heavy burden is placed on the original data and the copy that the performance within the disk subsystem suffers unless the data has been copied completely onto the copy. An example of this is backup when data is completely backed up through instant copy (Section 7.8.4). On the other side, the background process for copying all the data of an instant copy requires many hours when very large data volumes are involved, making this not a viable alternative. One remedy is incremental instant copy where data is only copied in its entirety the first time around. Afterwards the instant copy is repeated – for example, perhaps daily – whereby only those changes since the previous instant copy are copied. A reversal of instant copy is yet another important variant. If data is backed up through instant copy, then the operation should be continued with the copy if a failure occurs. A simple approach is to shut down the application, copy back the data on the

productive hard disks per a second instant copy from the copy onto the productive hard disks and restart the application. In this case, the disk subsystem must enable a reversal of the instant copy. If this function is not available, if a failure occurs the data either has to be copied back to the productive disks by different means or the operation continues directly with the copy. Both of these approaches are coupled with major copying operations or major configuration changes, and, consequently the recovery takes considerably longer than a reversal of the instant copy.

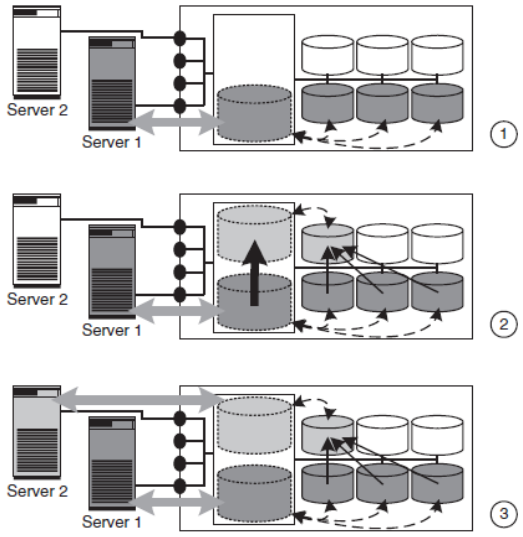

Figure 2.19 Space-efficient instant copy manages with fewer storage systems than the basic form of instant copy (Figure 2.18). Space-efficient instant copy actually only copies the changed blocks before they have been overwritten into the separate area (2). From the view of server 2 the hard disk that has been copied in this way is just as large as the source disk (3). The link between source and copy remains as the changed blocks are worthless on their own.

Source : http://elearningatria.files.wordpress.com/2013/10/cse-viii-storage-area-networks-06cs833-notes.pdf