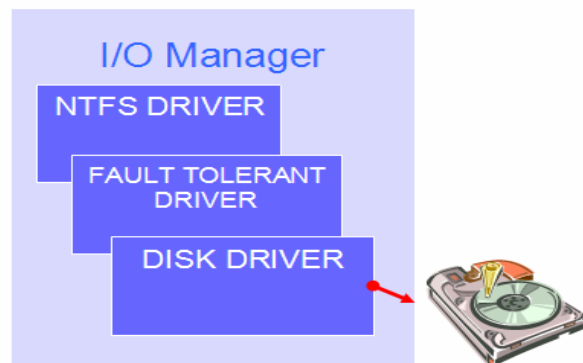


File Fragmentation, SANs, NAS and RAID

This document will explain the behavior and benefit of implementing defragmentation software with intricate modern hardware technologies such as RAID, NAS and SANs. SANs, NAS devices, corporate servers, and even high end workstations and multimedia-centric desktops characteristically implement multiple physical disk drives in some form of fault tolerant disk striping (RAID). Because the purpose of fault tolerant disk striping is to offer redundancy, as well as improved disk performance by splitting the I/O load, it is a common misconception that fragmentation does not have a negative impact.

As this data will show these devices do suffer from fragmentation. This is attributed to the impact of fragmentation on “logical” allocation of files as well as some degree as to their “physical” allocation.

The file system driver, NTFS.sys, handles the logical location (what the operating system, and a defragmenter affect). The actual ‘writing’ is then passed to the fault tolerant device driver (hardware or software RAID), which then, according to its procedures, handles the placement of files, and generating parity information, finally passing the data to the disk device driver (provided by drive manufacturer).



As noted, stripe sets are created, in part, for performance reasons. Access to the data on a stripe set is usually faster than access to the same data would be on a single disk, because the I/O load is spread across more than one disk. Therefore, an operating system can perform simultaneous seeks on more than one disk, and can even have simultaneous reads or writes occurring.

Stripe sets work well in the following environments:

1. When users need rapid access to large databases or other data structures.
2. Storing program images, DLLs or run-time libraries for rapid loading.
3. Applications using asynchronous multi-threaded I/O's.

Stripe sets are not well suited in the following situations:

1. When programs make requests for small amounts of sequentially located data. For example, if a program requests 8K at a time, it might take eight separate I/O requests to read or write all the data in a 64K strip, which is not a very good use of this storage mechanism.
2. When programs make synchronous random requests for small amounts of data. This causes I/O bottlenecks because each request requires a separate seek operation. 16-bit single-threaded programs are very prone to this problem.

It is quite obvious that RAID can exploit a well written application that can take advantage of asynchronous multi-threaded I/O techniques. Physical members in the RAID environment are not read or written to directly by an application. Even the Windows file system sees it as one single "logical" drive. This logical drive has (LCN) logical cluster numbering just like any other volume supported under Windows. As an application reads and writes to this virtual environment (creating new files, extending existing ones, as well as deleting others) the files become fragmented. Because of this fact, fragmentation on this logical drive *will have* a substantial negative performance effect. When an I/O request is processed by the file system, there are a number of attributes that must be checked which cost valuable system time. If an application has to issue multiple "unnecessary" I/O requests, as in the case of fragmentation, not only is the processor kept busier than needed, but once the I/O request has been issued, the RAID hardware/software must process it and determine which physical member to direct the I/O request. Intelligent RAID caching at this layer can mitigate the negative impact of physical fragmentation to varying degrees but will *not* solve the overhead caused to the operating system with the logical fragmentation.

So the question now becomes how does a defragmenter affect this? The defragmenter sees the RAID environment just as the file system does. That is, it defragments the logical drive, improving the speed and performance of a RAID environment by eliminating wasteful and unnecessary I/Os from being issued by the file system. This occurs because the file system sees the files and free space as being more contiguous. The file system will spend less time checking file attributes which means more processor time can be dedicated to doing real useful work for the user/application. In addition, these I/O requests are now more likely, depending on the RAID, to fill the entire 64K chunk (RAID stripe) size with the I/O now taking full advantage of the RAID.

To gauge the impact of fragmentation on a RAID system employ performance monitoring technologies and examine Average Disk Queue Length, Split I/O's, and % Disk Time. Additional disk performance tuning information can be found in Microsoft's online resources.

Executive Software • 7590 North Glenoaks Boulevard • Burbank • California 91504-1052 • USA
Toll Free 800-829-6468 • Phone 818-771-1600 • Fax 818-252-5514 • www.executive.com