

Disk Subsystems: The Dell Drive Array

Dell Computer Corporation
9505 Arboretum Boulevard
Austin, Texas 78759-7299

Summary

Rapid advances in processor and memory subsystem performance have allowed for significant imbalance in system performance relative to system components such as disk and network subsystems. The Dell Drive Array (DDA) is designed to counter this potential imbalance and to provide improved fault tolerance characteristics. This article first discusses factors in disk subsystems and then shows how these factors are addressed in the Dell Drive Array.

The Dell Drive Array centers around a high performance disk controller which directly addresses high leverage performance factors to provide a comprehensive solution in a PC based disk subsystem. The Dell Drive Array is designed to maximize compatibility with existing software and current drive technology while providing flexibility to directly address performance requirements and anticipated technology improvements. Especially important is the provision of “data striping,” with redundancy, to provide high performance in a transparent and fault tolerant manner. The conceptual innovation of the Dell Drive Array is to implement disk subsystem technology previously confined to high end systems in a highly effective form suitable for standard PC based systems, while maintaining compatibility with PC hardware and software.

Fundamental Factors in Disk Subsystems

This discussion focuses on personal computers used as workstations, servers and multiuser systems. However, most of the discussion is generic to disk subsystems in general. For a more abstract discussion and specifics associated with mainframes and other larger machines, see Chapter 9 of Hennessy and Patterson[1].

Address correspondence to Charlie Sauer, (512) 343-3310, sauer@dell.com.

Software Factors

Performance realized by a disk subsystem, depends, at least, on the following software controlled factors:

1. The number of applications presenting requests to the subsystem, e.g., one in the case of a DOS workstation or many in the case of a file server or a multiuser system.
2. The nature of the requests made by the applications. (In the case of a file server, the requests may be coming from network attached workstations.) The requests may be small (say, less than a sector), medium (a single sector), or large (multiple sectors). The requests may be in largely sequential patterns, largely random patterns or some combination. The proportion of reads to writes may also be relevant. Conventional wisdom is that reads outnumber writes by a factor of up to 10, but this is clearly not universally true.
3. The extent to which requests are cached (buffered) by the operating system. In the case of a DOS application on a standalone workstation, the default is not to cache, except for buffering of small (sub-sector) requests. (Many utilities exist for providing a disk cache for DOS.) Network operating systems, e.g., Netware, multiuser systems, e.g., Unix, and other systems, e.g., OS/2, provide disk caching by default. A disk cache can have several dramatic benefits to performance:
 - a. Repeated accesses to the same cache block (typically, a cache block is 1–16 sectors) can be satisfied by the contents of the cache, without any disk access after the first one. Where there are intervening accesses to other blocks, the repeated accesses may still be satisfied by the cache as long as the intervening accesses do not cause the desired blocks to be flushed from the cache.
 - b. Reference to part of a cache block (e.g., one sector out of several) causes the entire block to be brought into the cache, allowing subsequent partial block requests to be satisfied without disk access.
 - c. Where blocks are accessed sequentially (or in some other simple pattern), the cache management software can read additional blocks while the application is processing the requested blocks, thus avoiding the full delay of disk access. This assumes that either a multitasking operating system is used or that the disk hardware has bus master (or equivalent)

capability allowing for accesses to be completed while the application is executing.

Unfortunately, cache management requires processor time, so it is possible for a cache to actually degrade performance. In most implementations, a cache requires an extra memory to memory move of the data, so a cache adds latency to any request not satisfied by the cache.

4. The distribution of data across multiple drives, where multiple drives are used. DOS, OS/2 and most Unix systems require all blocks of a file to reside on the same drive, and common practice has related files on the same drive as well. This is unfortunate, because it limits the ability to increase throughput by accessing multiple drives concurrently. Network can spread blocks of a file across multiple drives.

Hardware Factors

Performance depends, also, on the following hardware factors:

1. Main processor performance. This affects not only the delays in operating system overhead, but also the ability of the system to cope with time critical events in the disk subsystem, e.g., to avoid missing revolutions of a disk.
2. Main memory capacity, where software caching is used.
3. I/O bus bandwidth characteristics can be a factor, but even the ISA bus can provide throughput up to 8 MB/second, so other factors usually dominate.
4. Controller register interface. The programming interface presented by the disk controller can limit the performance achievable by the operating system, e.g., by precluding DMA or by precluding opportunities for concurrence where multiple drives are present. Unless it is properly matched to software needs, too high a level of programming interface can also be a problem because of operating system overhead required to meet the interface. Controllers typically have relatively slow embedded processors, so a high level interface may also add unnecessary latency to controller operations.
5. The presence of bus mastering or corresponding capabilities on the disk controller(s). Bus mastering allows the controller to transfer to/from main memory without processor intervention. Assuming adequate performance from the bus mastering hardware, this means that server/multiuser systems can achieve increased throughput by concurrent bus master transfer with other main processor activity.

6. The presence of caching in the disk controller itself. If there is no caching or inadequate caching provided by the operating system, or if there is inadequate main memory for caching, then caching on the disk controller may be effective. On the other hand, if the operating system provides caching, then caching on the controller may result in undesirable latency, especially where the controller processor is slower than the main processor.
7. Ability to access multiple drives simultaneously. This is taken for granted in larger systems, but has typically been frustrated in PC systems by compatibility constraints relative to the original PC/AT ST-506 controller (and controllers derived from the PC/AT controller). There are two common granularities of simultaneous access:
 - a. file/block granularity (“independent seeks”). In a server or multiuser environment, the system administrator, or the operating system itself, may be able to place files, or blocks of files, on separate disks such that multiple drives can be accessed concurrently and independently. Where disk positioning dominates performance, such concurrency can significantly improve throughput.
 - b. sector/byte granularity (“data striping”). In systems without special system administration or built-in operating system support to place files/blocks on different drives to allow concurrency, the default approach is to place file blocks on disk to maximize contiguity. Contiguous placement, i.e., placing consecutive blocks of a file in consecutive positions on the disk, minimizes positioning delays when accessing those consecutive blocks. This is particularly true of operating system and application code files which are installed before fragmentation takes place. (Fragmentation is a characteristic of a disk where files have been created and then deleted, such that the blocks freed by the deletions are not contiguous.)

Where files or file blocks are largely contiguous on disk, there is no natural benefit of concurrent positioning of disks. However, it is possible to construct a “logical” disk from several physical disks by placing consecutive sectors (or even bytes) at corresponding positions on each of the physical disks. Assuming sufficient hardware control to allow synchronous positioning, it is then possible to access several sectors concurrently, achieving dramatic improvements in transfer rates.

There is no simple but meaningful way to quantify the benefits of one approach to granularity relative to another without looking at application, operating system and system administration specifics. Patterson *et al* provide a comprehensive description oriented toward large numbers of disks (10–25) and supercomputers[2]. The Dell Drive Array is oriented toward smaller numbers of drives (2–10) and PC based systems.

8. Drive interface. In PC based systems, the reasonably high performance interfaces are ESDI, IDE and SCSI. ESDI (Enhanced Small Disk Interface) is a low level interface similar to the earlier ST–506 interface but allowing much reduced controller intervention during positioning and much higher transfer rates. Nominal transfer rates of 10 or 15 Megabits/sec. are typical, with 20 Mb/sec. available from some vendors and 24 Mb/sec. anticipated by others. IDE (Integrated Drive Electronics) provides a PC/AT register compatible interface as the drive interface. Typical IDE drives provide a 10 Megabit/sec transfer rate, but 15 Mb/sec. drives are emerging. SCSI (Small Computer Systems Interface) is an even higher level interface designed for other peripherals, e.g., tapes and printers, as well as disks. Drive vendors typically provide the same HDA (head/disk assembly) in two or more of these interfaces, e.g., both ESDI and SCSI or both IDE and SCSI, so in principle there is no fundamental advantage of one over the other. However, as discussed above, higher level interfaces tend to limit performance, so for the same HDA, ESDI should be faster than IDE and IDE should be faster than SCSI, assuming roughly comparable microprocessors are used in the drive electronics.

9. Drive characteristics. Usually, the first metric of drive performance is the seek time. Typically, this is quoted as the average of all possible seek times, though actual average seek times depend on actual seek patterns and more detailed seek parameters of the drive. The second metric is the transfer rate, e.g., the 10 or 15 Megabits/sec. figures cited above. Transfer rate is a direct consequence of rotational speed, typically 3600 RPM, and the number of bytes per track. For example, a nominal 15 Megabits/sec. ESDI drive has 31,250 bytes per track unformatted ($31,250 \times 8 \text{ bits} \times 60 \text{ revolutions} = 15,000,000 \text{ bits/second}$). Formatted as 512 byte sectors, this same drive has 54 sectors/track, for a maximum transfer rate of $8 \times 512 \times 54 \times 60 = 13,271,040 \text{ bits/second}$ or 1.58 Megabytes/sec.

There are other hardware factors which have had major impact in the past but can be assumed to be ignored in current high performance PC's and larger systems. (For

example, interleaving of sectors has been a significant factor in the past, but fast processors and track buffers allow 1:1 interleaving to be used by default.)

Other Factors

In addition to the above performance factors, there are at least two other major factors in disk subsystems:

1. Software compatibility. The most widely used controller register interface is the “task file” interface originally implemented in the PC/AT ST-506 controller. However, there are serious performance limits in this interface, as suggested above, so some other interface must be available to achieve full performance. Though there is no other clearly obvious standard interface, the register interface used in the Adaptec 1540 family of SCSI controllers is fairly widely adopted, with device driver support available for most important operating systems. This is a much higher performance design than the task file interface, and thus a reasonable candidate for the primary controller interface.
2. Fault tolerance. Though performance and capacity are the most frequently discussed topics in disk subsystems, fault tolerance is perhaps even more important. Failure of a single drive is a severe problem in a typical system. Without adequate backups, the loss of data is disastrous. Even where appropriate disaster recovery procedures are in place, e.g., frequent backups, there is significant cost in both the procedures used to prepare for the potential of a failure and in the actual recovery process. The work by Patterson *et al* [2] has been highly influential in characterizing approaches to use of redundant drives to provide fault tolerance in disk subsystems.

The Dell Drive Array

The Dell Drive Array (DDA) centers around a high performance disk controller which directly addresses the above factors to provide a comprehensive solution in a PC based disk subsystem. The Dell Drive Array is designed to maximize compatibility with existing software and current drive technology while providing flexibility to directly address performance requirements and anticipated technology improvements. Especially important is the provision of data striping, with redundancy, to provide high performance in a transparent and fault tolerant manner.

Software Compatibility

It is well known that hardware and software compatibility are requirements in PC based systems, so these have been directly addressed in DDA.

BIOS INT 13H. The vast majority of DOS based applications depend on the INT 13H services of the BIOS. A high performance INT 13H interface is standard on the DDA controller. [This addresses the software compatibility factor discussed above.]

Adaptec SCSI Interface Emulation. The Adaptec 1540 register interface provides for bus mastering and significant concurrency, yet strongly leverages existing software. Since the 1540 is a SCSI controller, it is designed with concurrency in mind. A number of important operating systems, e.g., most variants of Unix and Unix-based network operating systems, include device drivers for the 1540. 1540 device drivers for other major operating systems, e.g., Netware and OS/2, are part of the DDA subsystem. Use of the Adaptec interface produces *substantially* improved throughput relative to the task file interface of typical PC disk controllers: (1) *bus mastering* allows the controller to transfer data to/from memory while the processor deals with other devices (e.g., a network) or application processing. (2) *concurrency* allows positioning of several disks while transfers are in progress. (3) *queueing* of requests allows the DDA firmware to optimize processing of the requests by combining and/or reordering requests. [This addresses the software compatibility factor discussed above. It also addresses hardware factors 4, 5 and 7.]

Hardware Compatibility

EISA. The DDA controller is a standard card for the predominant 32 bit PC system bus, EISA (Extended Industry Standard Architecture). The EISA bus allows up to 33 Megabytes/sec transfer rates, so bus bandwidth cannot be a limiting factor until extremely high throughput is achieved. The DDA implements optional high performance EISA characteristics, including bus mastering and bursting. Bus mastering allows DDA to sustain high transfer rates without processor intervention. Bursting is a form of bus transfer which amortizes setup overhead over a large number of bus cycles instead of repeating it for each word transferred. [This addresses hardware factors 3 and 5.]

IDE Drives. There were three obvious options for the drive interface for DDA: ESDI, IDE and SCSI. Though still dominant in the installed base, ESDI is being dominated by IDE and SCSI in new offerings from drive vendors. ESDI would have required significantly higher hardware cost and complexity on the controller and thus was inappropriate. SCSI provides no special advantages as a drive interface, potentially interferes with performance optimizations and would have required additional firmware support on the controller. So the obvious design choice was IDE, allowing DDA to take advantage of the recent growth in performance and capacity characteristics of IDE drives. The small form factors associated with IDE suggest

that it is the preferred choice in configuring relatively large numbers of drives. [This addresses hardware factor 9.]

Native Controller Characteristics

A number of the performance factors discussed above depend on either direct support in the lowest levels of the operating system or in firmware on the disk controller. The best leverage for these factors in DDA was to provide an extremely high performance microprocessor based subsystem, with specifications rivaling main processor performance. The processor is a 32 bit Intel 960KA RISC processor with a 16 MHz clock. This is in stark contrast to the 16 bit CISC processors, e.g., the 186, typically used on PC disk controllers in the past. 512KB of 32 bit burst mode ROM has been provided for the firmware, and 256KB of static RAM provides for data storage and the option of dynamically loaded firmware. The internal bus structure of the controller is designed to maximize bandwidth to the drives, including simultaneous transfer from multiple drives. [This addresses hardware factors 4, 6, 7 and 8.]

Connections are provided for five IDE drive pairs. In traditional IDE implementations, drives pairs are master/slave in the sense that only one drive may be positioned at a time. The DDA interface allows a compatible extension which allows concurrent positioning of each member of a pair, in a “master/master” fashion. Thus Dell Drive Arrays configured for multiple logical volumes or for independent seeks can get unexpectedly high levels of concurrency. [This addresses hardware factors 7 and 8.]

Connections are also provided for possible future hardware extensions, e.g., for additional RAM for caching or for ports to other peripheral devices.

To allow for diagnostics and future extensions, a native programming interface is presented to the operating system in addition to the emulation interfaces described above. The BIOS INT 13H support in DDA uses the native interface to maximize DOS performance.

Caching

The majority of the 256KB SRAM on the controller is not needed for tables or other code purposes and is used as a “small” cache. 200KB+ is really not “small” historically, as far as disk caches go, and is clearly useful for file caching. Aggressive read ahead strategies are used to optimize throughput for sequential access patterns. The architecture allows for extension to optional larger caches in the future, as sug-

gested above. [This addresses hardware factor 6 and obviates the need for software factors 2 and 3.]

Concurrency (Striping)

DDA provides concurrency at both of the granularities described above. As the discussion above indicates, the striping mode is likely to be more generally applicable because it does not depend on direct operating system or administrative support.

Data striping is implemented at the sector granularity using 512 byte sectors. A logical drive consists of up to five physical drives. The rotation of the drives is synchronized, so concurrent transfer from all the logical drives is possible with a single positioning delay (seek, if any, plus rotation). For sequential transfers of contiguous files, extremely high transfer rates are feasible. An individual drive can provide transfer rates of over 1Megabyte/sec., so a five drive stripe has the potential for 5 MB/sec transfer rate. [This addresses software factor 2 and hardware factor 7b.]

Concurrency (Independent Seeks)

However, concurrency at the file or block granularity may be achieved, and preferable, in some environments. DDA allows independent seeking of all drives at once, due to the “master/master” implementation described above. (The native DDA interface allows for a full 10 drives to be seeking concurrently. Due to traditional SCSI architecture limits, the 1540 emulation interface allows a maximum of 7 concurrent seeks, as opposed to the task file limitation of 2 concurrent seeks.) [This addresses software factors 2 and 4 and hardware factor 7a.]

Other Performance Optimizations

Some of the performance features described above assume “nice” behavior of the operating system. For example, if an application makes a read request for four sectors, then that request should be presented intact to the controller. Unfortunately, it is not safe to assume that this will be the case. With some operating systems, multi-sector operations are broken into single sector operations before they are presented to the controller. The DDA firmware looks for single sector operations that can be combined into multisector operations. Note that this will be effective even where the application makes single sector requests that it could have combined into a larger request. DDA also looks for opportunities to reorder requests to improve performance in a multitasking or server environment. [This addresses software factor 2.]

Redundancy

As electronic components and assemblies have improved in reliability, the failure of disk drives, especially the mechanical/magnetic aspects of the drives, have become increasingly prominent as a source of hardware system failures. Further, recovery from drive failure is often significantly slower when it is even possible, and loss of data is a serious concern. As storage capacity increases, either through large individual drives, collections of independent drives or through striped collections of drives, the loss of data becomes more likely. Frequent backups are necessary to cope with anticipated media defects, head crashes, etc.

One of the advantages of striped drives, and the best known aspect of Patterson's RAID proposals [2], is that some of a logical volume may be dedicated to redundant data for recovery from drive failures. In a properly designed system, loss of a single drive within a volume incurs no loss of data. The other drives are used to reconstruct the data from the lost drive. This reconstruction of data can be done during normal operation of the system, without loss of availability and with acceptable degradation of performance. When the failed drive is repaired or replaced, the appropriate data for that drive is reconstructed, returning the fault tolerant capability of the system. Thus the system can operate indefinitely in the presence of single drive failures within each volume, as long as at most a single failure occurs within a volume and as long as the interruptions to replace drives are tolerated.

These are the concepts used in the redundancy implementation in the DDA. The specific approach used in DDA is referred to as "level 4" by Patterson *et al*[2]. In a level 4 logical volume, one of the drives is used as a "parity" drive. Say there are four "data" drives and one parity drive. The sectors of the parity drive are constructed as the exclusive or of the corresponding sectors on the data drives, e.g.,

$$parity = data1 \oplus data2 \oplus data3 \oplus data4$$

for an array of four data drives and one parity drive, where \oplus represents the exclusive or operation. If, say, the second data drive were to fail, then the data would be recovered by the exclusive or of the remaining drives, i.e.,

$$data2 = data1 \oplus parity \oplus data3 \oplus data4.$$

The level 4 approach has the advantages, relative to some of the alternatives, of simple implementation, very high read performance, and good write performance, as-

suming a clever implementation. Assuming that reads dominate writes, level 4 is probably the best general solution. [This addresses the fault tolerance factor.]

Flexibility and Extensibility

As described above, DDA has been designed to allow flexibility and extensibility to meet future opportunities and requirements. The firmware allows for boot time loading of code into the SRAM, either to patch the firmware in ROM or to add additional functions. The ROM itself is large enough to allow for substantial additional functionality in the future. The option connector allows for additional memory and/or devices to be added in the future, as well as for development and manufacturing diagnostic functions.

One set of obvious options for future firmware consideration is alternate forms of redundancy. Two of the more promising forms in the taxonomy of Patterson *et al* are level 1 and level 5, in addition to the level 4 form currently provided by DDA[2].

Level 1 is full duplexing, or mirroring. Level 1 has the advantage of potentially higher write performance for small writes, since there is no need for a read/modify/write cycle. In level 4 and level 5, read/modify/write cycles are required for writes if the transfer is not aligned on appropriate boundaries or if the number of sectors is less than the number of drives in a logical volume. However, level 1 spends 50% of the disk space on redundancy, vs. 1 drive out of several for level 4 or level 5.

Level 5 is very similar to level 4, but there is no designated drive for redundancy — the redundancy sectors are distributed alternately across all of the drives. In level 4, the parity drive can be a bottleneck for small writes, and the level 5 approach eliminates this bottleneck. However, the level 5 approach precludes optimizations which give special treatment to the parity drive, since there is no designated parity drive, so the level 5 implementation has lower performance for some common write scenarios, e.g., where the number of sectors per file block is the same as the number of drives per volume.

Conclusion

By integrating advances in microprocessor, memory and firmware technology, the DDA provides very high performance *and* fault tolerance. This is achieved while maintaining compatibility with industry standard hardware and software interfaces.

The design is very general both in the initial implementation and in provision for future extensions.

References

1. J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 1990.
2. D.A. Patterson, G. Gibson and R.H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *ACM SIGMOD* conference proceedings, Chicago, June 1988, pp. 109–116.