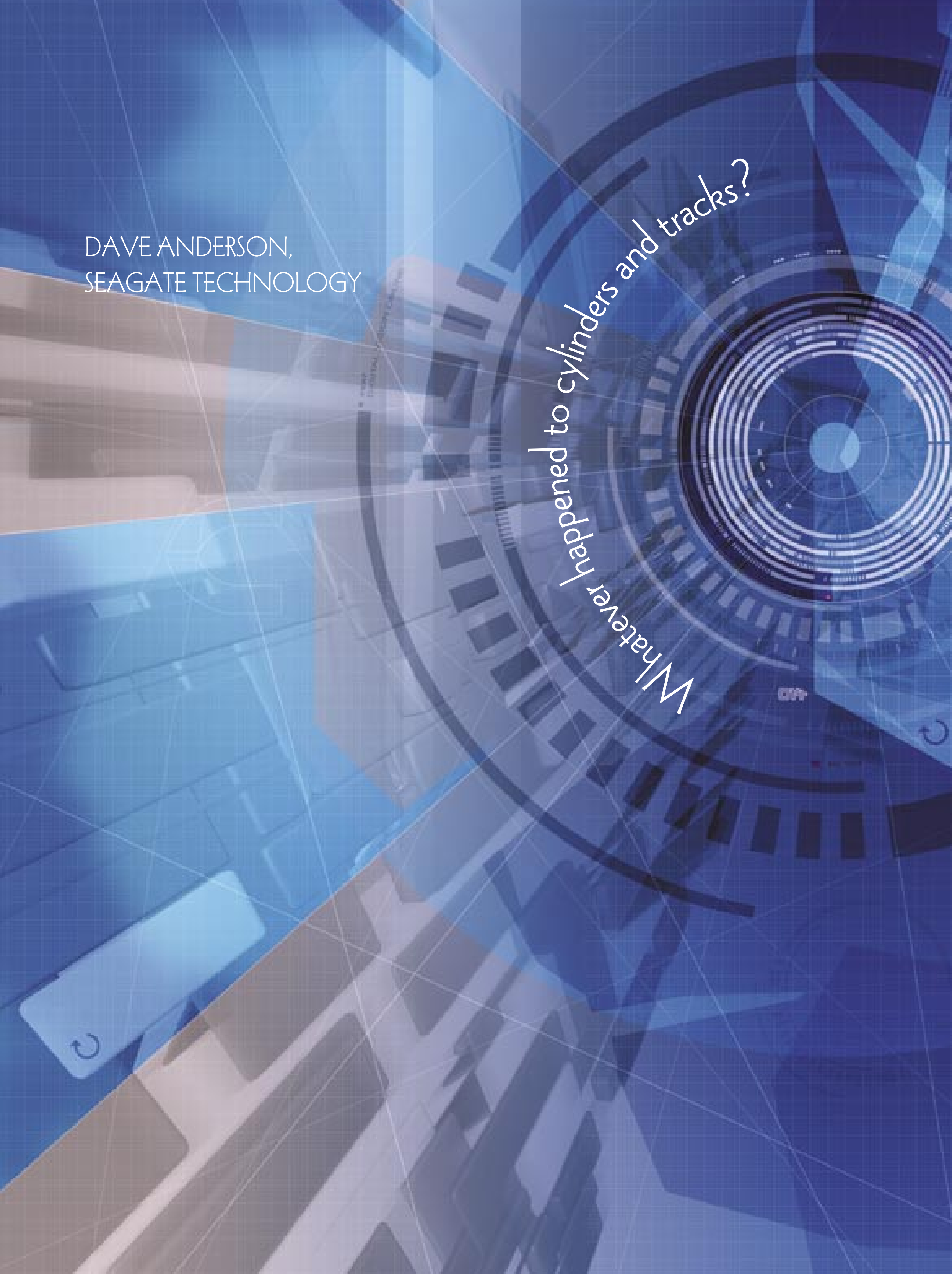
You Don’t Know Jack About Disks

Magnetic disk drives have been at the heart of computer systems since the early 1960s. They brought not only a significant advantage in processing performance, but also a new level of complexity for programmers. The three-dimensional geometry of a disk drive replaced the simple, linear, address spacetape-based programming model.

Traditionally, the programmer’s working model of disk storage has consisted of a set of uniform cylinders, each with a set of uniform tracks, which in turn hold a fixed number of 512-byte sectors, each with a unique address. The cylinder is made up of concentric circles (or tracks) on each disk platter in a multiplatter drive. Each track is divided up like pie slices into sectors. Because any location in this three-dimensional storage space could be uniquely identified by the cylinder number, head (surface) number, and sector number, this formed the basis



Whatever happened to cylinders and tracks?

DAVE ANDERSON,
SEAGATE TECHNOLOGY

You Don't Know Jack About Disks

for the original programming model for disk drives: cylinder-head-sector access.

This raises the question: If that is how data is stored on a drive, why don't we still use that as the programming model? The answer is not an easy one but has its roots in the fact that this geometric model endured until the advent of the intelligent interfaces, SCSI and ATA. [The IBM mainframe world used a slightly different model, allowing tracks to be written with records (blocks) of user-defined length. An individual track could have sectors of different sizes. As one who programmed count key data (CKD) storage, I can attest that it offers the application wonderful flexibility, but the drive design challenges have relegated it to history. Also, a purist might point out that standards etiquette calls for SCSI to use blocks and ATA to use sectors, but I will use these terms interchangeably.]

Disk-interface protocols implement the programming model for disk drives. The earlier drive interfaces did little more than expose signals to let the host directly manipulate the drive mechanism and initiate a transfer of data at a target location. This put the task of dealing with all the low-level idiosyncrasies peculiar to drives on the programmer charged with developing the firmware or software support.

The introduction of ATA and SCSI fundamentally changed this. Table 1 describes the migration of intelligence from host to drive in the evolution of the more important interfaces. With these intelligent interface protocols, the task of programming the use of disk drives became much easier. Disk-drive designers also gained a freedom of action needed to design higher-capacity and higher-performance drives. I will look at just how drive designers used this freedom of action in their designs, but it is important first to understand the fundamental goal behind drive design: increasing areal density.

THE BASICS: TPI AND BPI

A disk drive is, at its simplest, a delivery mechanism for persistent storage using magnetic recording techniques. The objectives of the drive design process are to improve the capacity, reliability, and performance of this mechanism at a minimal cost. A drive can be thought of as a three-dimensional space of recorded information. The surface of a disk provides two dimensions, and the stack of disks that make up the drive is the third dimension.

Capacity can be increased by adding disks. This drives up cost and causes more difficulties with increasing areal density as disk crowding increases. Increased vibration in the spindle, susceptibility to external vibration, and internal disturbance resulting from turbulent airflow will all increase with more disks in a given space.

For decades disk-drive capacity has been increased by reducing the spacing between data tracks, or track pitch, measured in tracks per inch (TPI), and increasing the linear density of the bits along a track, measured in bits per inch (BPI), as shown in Figure 1.

The product of these terms is areal density, measured today in gigabits per square inch. A state-of-the-art disk drive might have 30-plus gigabits per square inch. The challenge of designing head, media, and signal-processing systems to achieve higher areal density dominates the development of any disk drive.

Figure 2 shows the history of areal density growth.

THE OLD DAYS: THEY WEREN'T THAT GOOD

This original, geometric programming model that coincided with the physical organization of data posed several challenges for the programmer. It helps to understand some of the limitations of this physical model.

The specifics of the geometry seeped into the operating system (OS) software. For example, two drive manufacturers might develop comparable drives of identical capacity, but one might get more of the

TABLE 1 Movement of Intelligence from Host to Drive

Interface	ST-506	ESDI / SMD	SCSI / ATA
Date introduced	1980	1972/1985	1981/1991
MBps	.5	Up to 3	To 320 MBps
Intelligence level	Analog data signal	Digital data signal	Messages
Intelligence moved from host to drive	n/a	Data separator, some geometry control	Geometry abstraction, flaw mapping

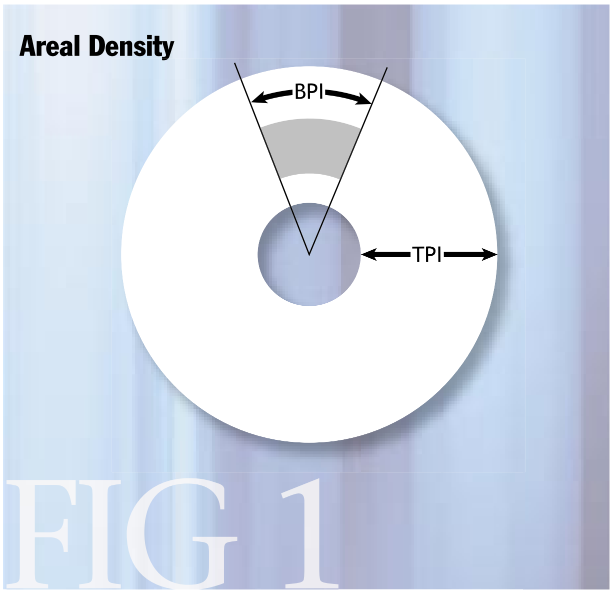


FIG 1

capacity via TPI, the other from BPI, and a third vendor with less capability might require an additional disk in the drive. This would result in drives with very different values for cylinders, tracks, and sectors per track. What's worse—or better, depending on your point of view—some disk controllers were able to get one or two more sectors per track by using more sophisticated electronics, by, for instance, shortening the gaps between sectors. The upshot was that even a given disk drive could have different geometry values depending on the controller to which it was attached.

The complexity of supporting all drive offerings was considerable. As explained previously in the areal density discussion, each generation of disk drive would increase both the sectors per track (BPI effect) and the number of cylinders (TPI effect). This means the OS or controller needed to include a table of drive geometries, the size of which was multiplied by the number of generations it contained.

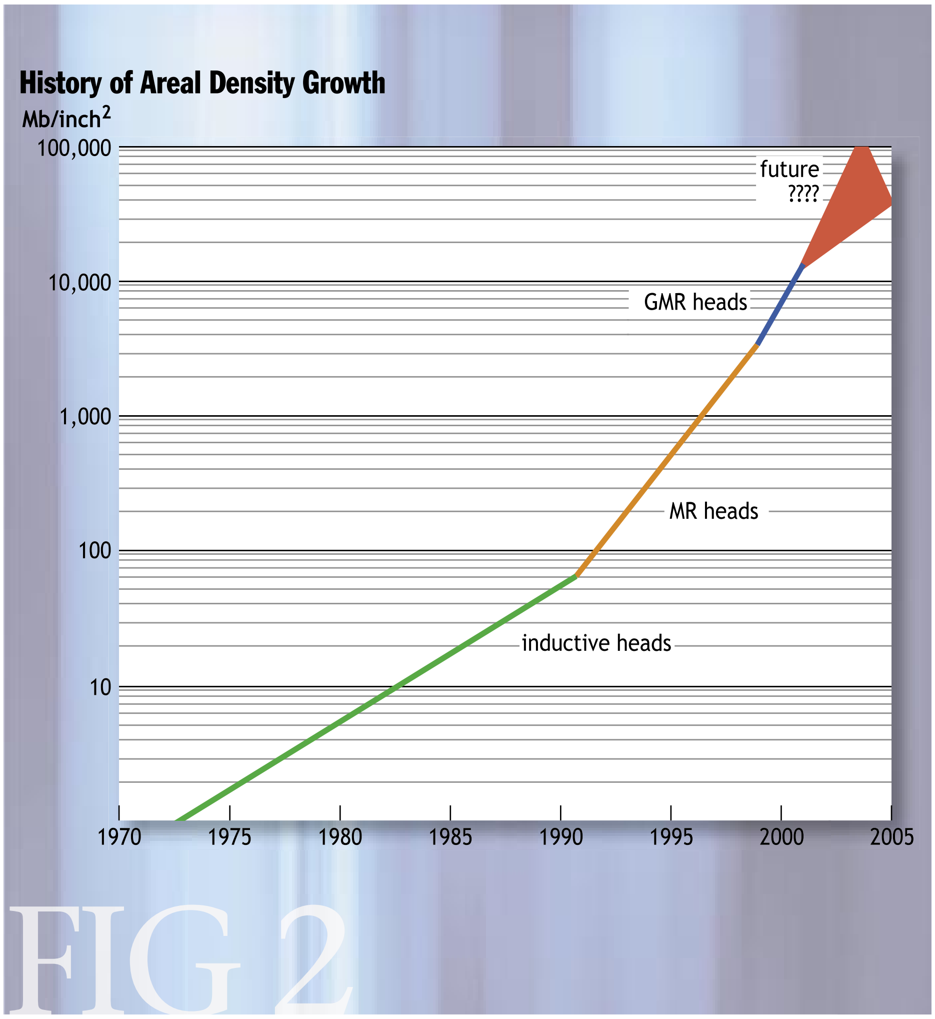


FIG 2

Handling flaws was an even bigger problem. When areal density was low enough (around 2 megabits per square inch, which was state-of-the-art in the mid-1970s), it was possible to build a stack of disks with no flaws, or bad sectors. As areal density increased, this quickly became impossible. Because the OS was looking at the raw device, it was responsible for managing bad-block flaws so they did not jeopardize the user data.

Through many generations of drives, controllers, and operating systems, the developers put up with these problems. But then came the last straw: zoned bit recording (ZBR).

An artifact of the cylinder, head, sector (CHS) model was the assumption that every track had the same number of sectors. As drive designers looked for innovative ways to deliver

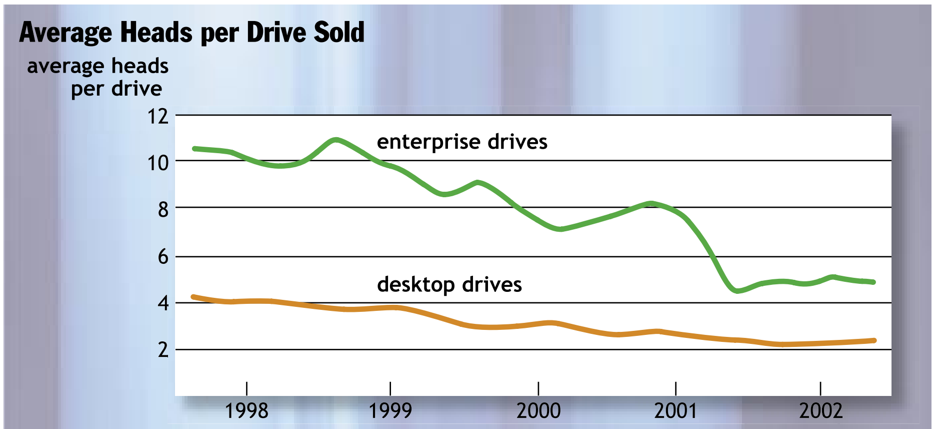


FIG 3

more capacity, they saw that the longer tracks on the outer diameter of a disk could hold considerably more data than an inner track. To take advantage of this, the drive was divided into radial recording bands or zones (i.e., clusters of nearby cylinders). All the tracks within a given zone had the same number of sectors. A track in a zone near the outer diameter of the disk, however, might have 50 percent more sectors than a track in a zone near the inner diameter of the same disk. This would be true for a 3.5-inch drive. The advantage ZBR provides varies by media size and is a function of the relative size of the outer radius of the recording band to the inner. Drives today usually have 15 to 25 zones.

ZBR added great value: 25 percent or more capacity for no additional material cost in a 5.25-inch drive, the prevailing form factor when ZBR first appeared. It forced the industry to adopt a more intelligent interface—one that would hide the complexities of ZBR and, at the same time, hide the geometry and bad-block flaw problems by pulling that functionality into the drive, as well.

WHAT CHANGED

The advent of intelligent interfaces—first SCSI, then ATA—completely changed the nature of disk support. Disk drives used microprocessors to manage these higher-level interfaces, and programming the host went from a low-level, know-all-the-signal-lines-and-timing engineering task to an inter-computer communication problem that any experienced programmer could manage.

These intelligent interfaces are based on a programming model that is essentially like a tape—sequential sectors or block addresses. Although drives still have block random access capability, the physical geometry is no longer seen in the programming model. Behind this new model or interface, drive vendors have been able to produce higher-capacity, more reliable, and faster-performing storage.

INSIDE A DRIVE TODAY

The cylinder, which was such an important allocation unit when there were lots of heads per cylinder, is fast losing its utility, even if it could be exposed through the interface.

One of the most interesting trends in drives is the rapid reduction in the average number of disks in each drive. Today most drives—dominated by the personal desktop market—ship only a single disk or platter. In fact, a good percentage of them (about 31 percent industry-wide, according to estimates by Seagate market research) ship with a single head. These drives demonstrate one benefit of the terrific increase in areal density. For the cost-sensitive PC market, a capacity sufficient to satisfy most customers can be achieved most economically by using only one recording head on one side of a single disk. The most disks ever put into a 1-inch-high 3.5-inch disk drive is six, with a corresponding 12 heads. The difference between what is possible in theory and what customers buy is dramatic, as shown in Figure 3.

Other factors are further reducing the significance of the cylinder dimension. The long-term trend toward smaller-diameter disks reduces the length of average tracks and shrinks the size of a typical cylinder. In particular, the highest-performance SCSI drives have adopted smaller-diameter media. In a 3.5-inch form factor the largest media possible is 95 millimeters. Although it offers maximum capacity, larger media has several negative side effects. It wobbles more, has more curvature both radially and circumferentially, and consumes more power for a given RPM. These make an increase in areal density more difficult to achieve. For these and other reasons, drives use smaller media as the RPM increases:

RPM	5400	7200	10,000	15,000
Diameter	95 mm	95 mm	84 mm	65-70 mm
Seek time	9	8	5.5	3.5

More recent changes in data layout effectively dissolved the cylinder concept. Logical blocks used to be ordered on a drive so that all sectors on one cylinder were used before going to the next cylinder. This is no longer necessarily the case. Spiraling the sectors along a single surface in one recording zone before moving to another surface in that zone has some advantages, such as sequential transfer performance.

Consider the outer recording zone in a three-disk drive. The first *n* tracks of the drive would be the *n* tracks that make up the first recording zone on the first surface. The next *n* tracks of the drive would be the n tracks on the next surface, but starting at the inner diameter working outward to minimize seek length and preserve sequential performance. The next *n* tracks would be the same as the first *n*, but on the third surface, with the next *n* again going out from the inner track of the recording zone to the outer radius. (See Figure 4.)

There are variations on this serpentine format, and not all drives use it. The point is that our notion of the cylinder as a fixed location of the actuator that would be desirable to use as an allocation unit is not a dependable concept.

The layout of tracks themselves has undergone a transformation. Historically, each sector of user data has been framed by additional information that enables the drive to locate the right one and ensure that it is correct. This extra data used to consume about 20 percent of the total bits on a track. Figure 5 illustrates the important fields.

The address, known as a header, was used to compare with the desired address to locate the target data. The error correction code (ECC) fields provided error recovery information. The gaps were needed to give the head time to turn the writer off or on as needed. Without those gaps, noise from the head could wipe out data or a subsequent sector header field. Because these fields represented overhead and loss of usable capacity, they were candidates for change so that more capacity might be made available to the user.

When drives used inductive heads, every sector had its own separate header, as previously described. Magneto-resistive (MR) and giant magneto-resistive (GMR) heads

Serpentine Ordering of Tracks

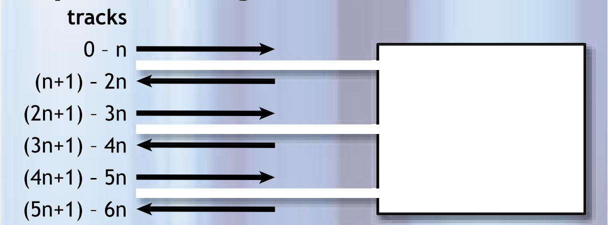


FIG 4

Traditional Layout of Sector on a Track

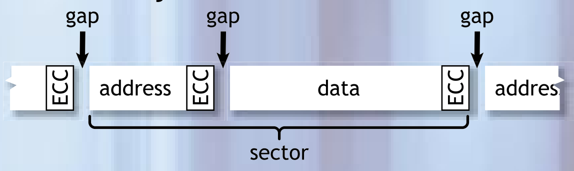


FIG 5

made the headers a more complex problem. Although they had an MR element for reading data, they still had an inductive writer. These two elements doing the reading and writing were next to each other on the head. Because they were on a rotary actuator, they could not both be positioned over a desired track at the same time. This meant that the read element was not on the track when the head was positioned for writing. Drives often used two headers: One was in line with the data sector to validate the address for reading; the other was offset from the data sector to enable the address to be validated when writing. This was clearly going in the wrong direction, consuming more capacity instead of less.

As TPI increased, servo information was placed on the data tracks instead of on a separate dedicated servo surface. Typical drives today might have three to five sectors per servo burst, depending mostly on whether the drive is SCSI or ATA. (More servo bursts consume capacity but provide more performance and reliability in the presence of external influences such as vibration.) These servo bursts took the form of short blocks of information that consume from 5 to 10 percent of the bits on a track. As drive designers developed experience with the embedded servo bursts, they realized that no headers were really needed. They could use the positioning information in the servo and just count until they passed enough sectors to arrive at the desired one. (See Figure 6.)

RELIABILITY AND PERFORMANCE

More abstract interfaces that isolate the host from drive error recovery and flaw management contribute to a more reliable storage system. One way drives take advantage of the intelligent interface is with sophisticated error recovery. Because error correction codes and recovery techniques are now handled inside the drive, the drive has abilities that no host has. For example, if a particular sector is difficult to read, a drive can make a complex series of adjustments as it repeatedly attempts to recover the information on the disk. It can move the head slightly to the left and then to the right or it can adjust the timing, essentially starting the reading a fraction of a millisecond sooner or later. A drive might do 20 to 30 retries in various combinations of offset and timing to give itself the best opportunity to pick up the data. This procedure does not happen often, but it can make the difference between data that can be read and data that cannot. Once it recovers the information, it can then, optionally, mark that section of the drive as bad and rewrite those logical blocks to another section. A certain percentage of the raw capacity of the drive is reserved for remapping of sections that go bad later on. This space is neither addressable nor visible through the interface.

Intelligent interfaces brought an immediate performance benefit in that they could buffer data until the host was ready to accept it. Similarly, they could accept data regardless of when the drive was positioned to write it, eliminating the need for the host to synchronize its connection to the drive with positioning of the heads for a data transfer. Although this buffering proved to be an important performance improvement, drives go well beyond this in enhancing performance.

In a demanding workload environment, a high-performance drive can accept a queue of commands. Based on the knowledge of these specific commands, the drive can optimize how they are executed and minimize the time required to complete them. This assumes that multiple commands can be issued and their results either buffered in the drive or returned out of order from the initial request. The longer the command queue is, the greater the possibility for throughput optimization. Of course, host file systems or controllers do this as well, and many argue that it can manage more commands. The drive offers some special advantages, however.

Consider a drive attached to a single host. If the host is managing the I/O queue, it can be confident that the read/write head is at roughly the location of the last I/O it issued. It will usually select another command as close as possible to the previous one to send to the drive, often working from one end of the logical block address (LBA) range to the other and back again. The LBA range is that sequential "tape" model mentioned earlier. This is about as good an LBA scheduling model as the host allows.

The drive, on the other hand, knows the actual geometry of the data. This includes exact information about the radial track position and the angular or rotational position of the data. If it has a queue to work on, it selects the next operation based on the one nearest in time, which can be quite a bit different from the nearest LBA. Consider the following work queue:

Operation	Starting LBA	Length
Read	724	8
Read	100	16
Read	9987	1
Read	26	128

The host might reorder this to:

Operation	Starting LBA	Length
Read	26	128
Read	100	16
Read	724	8
Read	9987	1

Sector Formats

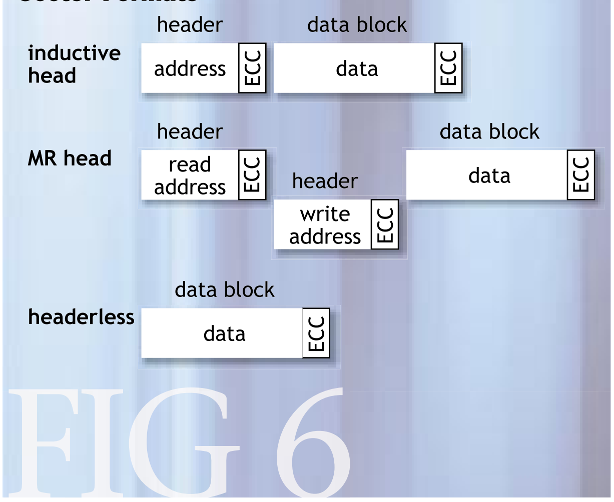


FIG 6

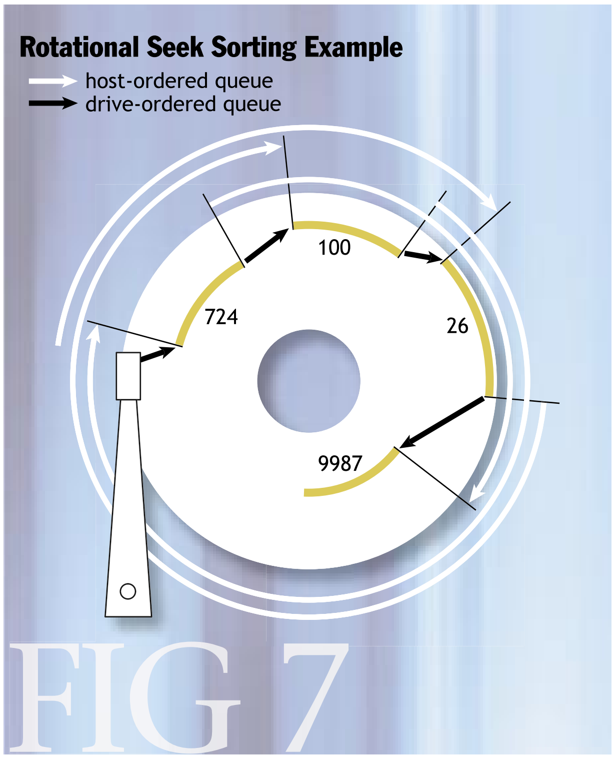


FIG 7

This seems to make sense. The actual rotational location of the data, however, may make this the worst ordering. The drive can take advantage of both seek distance and rotational distance to produce the optimal ordering shown in Figure 7.

The drive-ordered queue would complete in three-quarters of a revolution, while the host-ordered queue would take three revolutions. The improvement when the drive orders a queue can be impressive.

As Figure 8 shows, a drive that is able to do a throughput of about 170 random I/Os per second with no queue can achieve more than three times that number with a sufficiently long queue (admittedly, a queue of 256 I/Os would be rather large). Note also that as the queue lengthens, the variation in response time can go up. The average service time for an individual operation will decrease, but some commands will sit in the queue longer, victims of the overall optimization. This results in both an increase in average response time and—what is usually of more concern to users—an increase in the variation in response time. The interface provides a timer to let the user limit the maximum time any I/O can sit before it must be serviced.

If the drive is servicing work from more than one system, the benefits from the drive managing the queue can be especially valuable. Two hosts may each operate as though it is the only source of I/O requests. Even if both do the best they can in ordering their queues, the conflict of the two independent queues could at times produce interference patterns that result in degraded performance. The drive can efficiently coalesce those workloads and reap the benefits of a single, longer queue. In enterprise-class disk arrays, where four or eight storage directors might be accessing each drive, the benefit is even clearer.

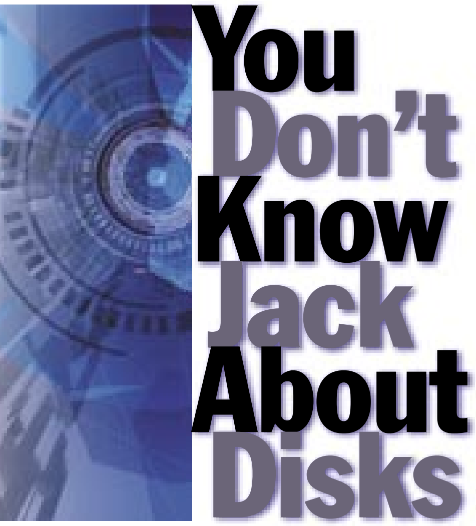
ATA VERSUS SCSI

ATA and SCSI have much in common and use essentially the same addressing model, but they have subtle protocol differences that reflect the differing applications of the two interfaces. [The most important differences have nothing to do with the interface, but everything to do with the head disk assembly (HDA), which the interface is connecting to the system. A complete discussion of these differences is available in "ATA versus SCSI: More Than an Interface," by Dave Anderson, Erik Riedel, and Jim Dykes, FAST, 2003.]

Both interfaces typically use a 512-byte sector. SCSI supports other sector sizes, including incrementally longer lengths. These are used extensively in disk arrays, where the host may read and write 512-byte sectors, but the array controller reads and writes perhaps 520 or 528 bytes, as shown in Figure 9. (The drives have to be formatted for this longer sector.) The controller will append data information it uses to validate the sector—both its address and contents. This prevents any firmware error or electronics bug in either the drive or array controller from corrupting the data without the array being aware of it. This is a key feature in enterprise subsystems, which feature the highest level of reliability and data integrity.

SCSI interfaces (Fibre Channel, parallel SCSI, and serial attached SCSI) allow multiple hosts to connect to a drive. This creates more complexity for the drive firmware, but enables fault-tolerant configurations—i.e., having no single point of failure. It also entails some special commands. SCSI protocols have, for example, reserve and release commands that let a host take exclusive control of a drive when necessary. These are essential to managing a multi-host storage system. Again, this is functionality that would not be justified in a drive for a low-end array or a personal computer, but more than worth the cost in a system that is keeping a business up and running.

The SCSI interface includes other management features that allow more detailed control over the drive's operations. The respective standards committees, T10 for SCSI [www.t10.org] and T13 for ATA [www.t13.org], are good sources for information.



THE MOST PRESSING ISSUE AND WHAT ELSE MIGHT CHANGE

The last vestige of that old geometric model for disk drives is the fixed 512-byte sector. This still maps directly to the physical unit written to the media. It is probably what drive manufacturers would most like to get rid of next.

The use of error correction codes is an important drive component for protecting data and ensuring that it can be recovered from a disk and returned accurately. ECC information is appended to each 512-byte sector of data. This additional information means that a little more disk space is required to store the information. As data bits have become smaller, flaws have become effectively larger—that is, each flaw can corrupt more bits. The ECCs that enable the drive to recover from these flaws have had to become longer, from perhaps 16 bytes of ECC per sector in 1995 to more than double that today. To continue increasing areal density, it will get longer yet. To minimize the proportion of disk space dedicated to ECC overhead, drive vendors universally desire a longer sector size—4,096 bytes is the proposed length. Whereas the work required to support 4K sectors in a drive is straightforward, the difficulty of getting the rest of the computer system to employ it is daunting. It is not clear when this might happen, but companies are

How Much Storage is Enough?

Peter Lyman and Hal R. Varian, U.C. Berkeley

In 2000, faculty and students at the School of Information Management and Systems at the University of California at Berkeley initiated a study to measure how much information is produced in the world each year. They looked at several media and estimated yearly production, accumulated stock, rates of growth, and other variables. The report is about 200 pages long and tries to pull together virtually everything known about information production. Since it was first published in October 2000, technology and practice have changed dramatically. An update of the study is in progress and should be available online in summer 2003. Details about the report are available at www.sims.berkeley.edu/how-much-info. The following contains excerpts from that report.

In 1999, the world produced about 1.5 exabytes of storable content (10^{18} bytes). This is 1.5 billion gigabytes, and is equivalent to about 250 megabytes for every man, woman, and child on earth. Printed documents of all kinds make up only .003 percent of the total. Magnetic storage is by far the largest medium for storing information and is the most rapidly growing, with shipped hard-drive capacity doubling every year. Magnetic storage is rapidly becoming the universal medium for information storage.

units	
kilo	10^3
mega	10^6
giga	10^9
tera	10^{12}
peta	10^{15}
exa	10^{18}
zeta	10^{21}
yotta	10^{24}

How did we ever count the number of bytes produced? Much of the world's content is easy to count: books, magazines, newspapers, movies, CDs, and so on all have unique identifiers (ISBN numbers, ISSN numbers, and the like). Good production data exists about photographic film, hard drives, and other media.

The tough part is digital content, though that's the most important component. According to our estimates, more than 90 percent of information content is born digital these days. Conversion to ASCII, MP3, MPEG, and other compression technologies dramatically reduces storage requirements by one to two orders of magnitude. If all printed material published in the world each year were expressed in ASCII, it could be stored in less than 5 terabytes.

Individuals produce significant amounts of non-digital information; as photos and videos move to digital formats, however, households will have to manage terabytes of data. Print and film content is rapidly moving to magnetic and optical storage. This is true for professional use now and will become increasingly true for individual users. More than 80 billion photographs are taken every already investigating the OS and driver problems.

Some research argues for exciting future changes in the programming model, including hiding the physical sector size from the host altogether. Exposing the geometry of a drive to the host operating system in a useful way is impossible, because it is subject to change and variation. When a specific method for laying out files is being determined, however, knowledge of the

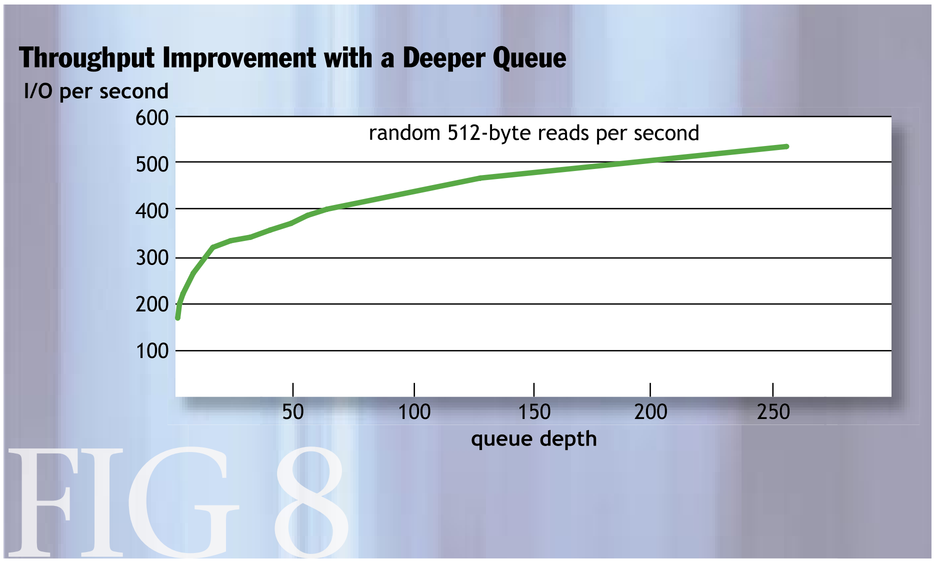


FIG 8

year; this would consume more than 400 petabytes of storage—more than 80 million times the storage requirements for text.

Our project is primarily concerned with content that is stored, either by institutions or by individuals. A lot of material, however, is communicated without being systematically stored. Some of this material is born digital, such as e-mail, Usenet, and Web information. Some of it is non-digital, such as telephone calls and letters.

We expect that digital communications will be systematically archived in the near future and thus will contribute to the demand for storage. In 1999 we estimated e-mail to be about 12 petabytes per year, Usenet about 73 terabytes, and the static HTML Web about 20 terabytes. Many Web pages are generated on-the-fly from data in databases, so the total size of the "deep Web" is considerably larger.

Although the social impact of the Web has been phenomenal, about 500 times as much e-mail is being produced per year than Web pages. It appears that about 610 billion e-mails are sent per year, compared with 2.1 billion static Web pages. Even the yearly flow of Usenet news is more than three times the stock of Web pages. As Andrew Odlyzko [see "Content is not Kin," *Technical Report,* AT&T Labs, 2000, www.research.att.com/~amo/doc/networks.html] puts it, *"Communication, not content, is the killer app."*

PETER LYMAN is professor and associate dean at the School of Information Management and Systems at the University of California at Berkeley. He received his bachelor's degree in philosophy from Stanford University, master's in political science from U.C. Berkeley, and Ph.D. in political science from Stanford. He serves on the boards of the Council on Library and Information Resources (CLIR), Sage Publications Inc., and the Internet Archive; the advisory boards of the Getty Information Institute and Chadwyck-Healy Inc.; on the Association of American Universities Committee on Digital Networks and Intellectual Property; and on the editorial boards of the *American Behavioral Scientist*, and the *Electronic Publishing Journal.*

HAL R. VARIAN is the dean of the School of Information Management and Systems at the University of California, Berkeley. He is also a professor in the Haas School of Business and the Department of Economics. He received his bachelor's degree from MIT and his master's in mathematics and Ph.D. in economics from U.C. Berkeley. He has taught at MIT, Stanford, Oxford, Michigan, and other universities around the world. Varian is a fellow of the Guggenheim Foundation, the Econometric Society, and the American Academy of Arts and Sciences. He has served as co-editor of the *American Economic Review* and is on the editorial boards of several journals. His current research involves the economics of information technology and the information economy. He is the co-author of a book on business strategy, I*nformation Rules: A Strategic Guide to the Network Economy*, and writes a monthly column for *The New York Times.*

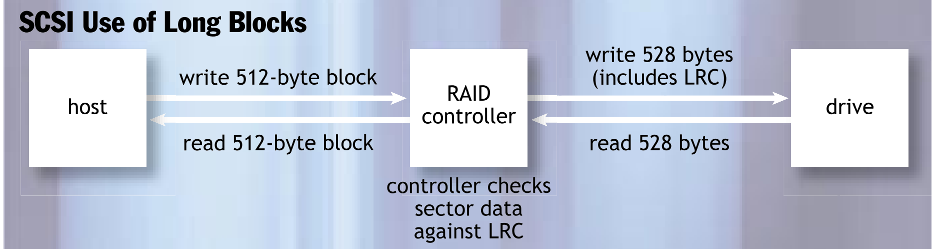


FIG 9

exact drive geometry, including information such as flaw location, would benefit performance.

One of the motives behind object-based storage device (OSD) research by the Storage Networking Industry Association (SNIA) and the T10 Technical Committee of the International Committee of Information Technology Standards is the recognition that the drive could be assigned the responsibility for organizing the information storage space [refer to "Working Project Draft T10/1355-D Revision 06 Information Technology—SCSI Object-Based Storage Device Commands (OSD)," Ralph Weber, editor, Aug. 2, 2002]. It could then employ an exact geometric model, because it understands the unique specifics of its three-dimensional space and could in theory allocate data intelligently based on that knowledge. OSD would also solve the long-sector problem; it completely abstracts the underlying drive format by transferring data in byte lengths.

Security is a growing concern among users. OSD research advocates an appealing security model. Recent research at MIT found that a significant amount of sensitive data could still be recovered from drives that had been "erased" and discarded ["Selling a computer? Be sure to erase hard drive files," by Justin Pope, *Minneapolis Tribune,* Jan. 20, 2003]. OSD includes a security model that would marry the security policy to the data at the drive level. Special information such as a unique password would always be required to decode the information stored on the media. Because the data cannot be separated from the security control, this sort of vulnerability could be avoided.

Concurrency control is another area of research that could have implications with respect to the programming model. Two projects suggest that the drive, because it is at the point of convergence for accesses from multiple systems, could help make clusters more scalable [see "Scalable Concurrency Control and Recovery for Shared Storage Arrays," by Khalil Amiri, Garth Gibson, and Richard Golding, CMU-CS-99-111, February 1999; and "The Global File System," by Steven R. Soltis, Thomas M. Ruwart, and Matthew T. O'Keefe, *Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies*, Sept. 17-19, 1996]. Other examples include applications for disk locks such as fencing and revocation of access rights. Seagate and other drive manufacturers have participated in several aspects of this ongoing research, and most feel that many of these ideas are worthwhile candidates for future disk-drive architectures. Q

BIO

DAVE ANDERSON, director of strategic planning for Seagate Technology, has more than 20 years of experience in the computer field. His responsibilities include overall strategy for all disk interfaces. He has been involved in the architecture and planning of Fibre Channel since it was first proposed as a disk interface. He was also one of the principal architects of the disk XOR commands that are now a part of the standard SCSI interface specification and was the author and editor of the original object-based storage device (OSD) proposal being developed by SNIA for submission to the SCSI standards committee. Anderson was one of the original nine elected members of the SNIA Technical Council. He was also one of the founding members of the serial attached SCSI working group. The author thanks Zip Cotter for his technical and editorial acuity.