INTERLEAVED STORAGE

In recent years, the use of computers to store and process information has risen dramatically. Every major business uses the computer as a tool to compete in industry. The necessity to use computers to compete has driven the need for higher-performance systems. Rapid access to information is critical. Equally important is the safety and availability of information stored in computer systems.

Over the past 20 years, the processing capability of central processing units (CPUs) has increased by as much as 60% per year. Random access memory (RAM) performance has increased at a rate of 40% each year. During this same period, disk storage has doubled in capacity and halved in cost every three years. Unfortunately, due to their electromechanical design, disk-storage performance (seek time, rotational latency, and data transfer rate) improved by less than 50% in the last decade. This gap in CPU and memory and disk input/output (I/O) performance is the limiting factor of today's computer systems.

From Table 1 observe that the performance mismatch between memory and processor bandwidths are an order of magnitude. Typical dynamic RAM (DRAM) chips reach around 30 MHz frequency for random access within a given page. Typical processors operate in a range of 100 MHz to 300 MHz. The performance mismatch between memory storage and magnetic disks are three order of magnitudes. To alleviate the performance mismatch between the processor and memory and memory and secondary storage devices various techniques have been devised to mask the effect of the slower component. In order to understand these techniques we first review the design and architecture of memory and disks.

To quantitatively compare the performance of different devices, a standard method is to measure the access times. The time a program or device takes to locate a single unit of information is called its access times. The access times for different storage devices are given in Table 1.

Memory Architecture

From a logical point of view, memory is just an array of words in which information can be stored. Each location has a unique address. A memory hierarchy consists of multiple levels of memory with different speeds and sizes. The logical view of a memory hierarchy is a cache, primary memory and a secondary memory. Main memory is implemented using DRAM while caches typically use static RAM (SRAM). DRAM is less expensive than SRAM but is slower. In SRAM the value stored in a cell is kept as a pair of inverting gates and remains indefinitely as long as there is power. In DRAM the value stored in a cell is kept as a charge in a capacitor. Thus only a single transistor is used per bit of storage to read or write the stored charge. In comparison, SRAM has four to six transistors per bit. However, in DRAM the charge cannot be stored indefinitely and has to be periodically refreshed.

Table 1. Access Times of Storage Devices

Device	Typical Access Time
Static RAM (SRAM)	10–50 ns
Dynamic RAM (DRAM)	50–150 ns
Erasable programmable read-only memory (EPROM)	55–250 ns
Read only memory (ROM)	55–250 ns
Hard disk drive	9–30 ms
Erasable optical disk	19–200 ms
CD-ROM	100–800 ms
DAT tape drive	20 s
QIC tape drive	40 s
8 mm tape drive	40–500 s

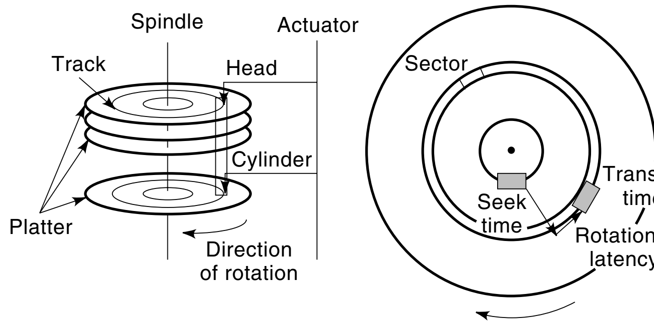

Figure 1. Disk geometry.

The performance of memory is measured by its latency. Memory latency is defined by two measures: (1) access time and (2) cycle time (1). Access time is the time between a read request and when the desired word arrives, while cycle time is the minimum time between memory requests. DRAMs have a larger cycle time compared to the access time as the information in memory has to be refreshed. In contrast, SRAMs have the same access time and cycle time.

Main memory is typically organized with a width of one word. Doubling the width of the memory in turn doubles the memory bandwidth. However, there is an extra cost of a wider bus. Memory chips can be organized in banks such that multiple words can be read or written simultaneously instead of single words. The banks are one word wide so that the width of the bus need not change. The other advantage of memory banks is interleaving sequential access. The interleaving of main memory as a method to improve performance is discussed in detail later.

Disk Architecture

Secondary memory is the least expensive and slowest form of memory. Secondary storage devices include magnetic disks, optical disks, and tapes. The magnetic tape was the first secondary memory that allowed sequential access. The disk is a random access device: it can retrieve the stored data anywhere on the disk in any order. The ability to randomly store and retrieve data is the most important reason disk drives rapidly displaced tape as the primary computer storage technology. Disk drives record data in tracks, or concentric circles, that are numbered from the outermost edge of the disk to the innermost.

Hard disk drives consist of multiple platters. The platter's surface is organized so the hard drive can easily find data. The concentric tracks are divided into units called sectors. (Figure 1 shows the disk geometry.) Information is recorded on the outermost track of all platters first. The design of hard disk drives makes them quite fast, by virtually eliminating friction between the disk and read/write head to increase performance further and reducing wear on the heads and media.

The platters on the hard disk drive are always spinning at 3600 rpm or higher.

The surface of the drive platter is organized with coordinates. Data are stored in concentric tracks on the surfaces of each platter. (A platter has two sides and thus two data recording surfaces.) A typical disk drive can have more than 2000 tracks/in. (TPI) on its recording surface. A cylinder describes the group of all tracks located at a given head position across all platters. To allow for easier access to data, each track is divided into individually addressable sectors.

The process of organizing the disk surface into tracks and sectors is called formatting, and almost all hard disk drives today come preformatted by the manufacturer.

In earlier hard drive designs, the number of sectors per track was fixed and, because the outer tracks on a platter have a larger circumference than the inner tracks, space on the outer tracks was wasted. The number of sectors that would fit on the innermost track constrained the number of sectors per track for the entire platter. However, many of today's advanced drives use a formatting technique called *multiple zone recording* to pack more data onto the surface of the disk. Multiple zone recording allows the number of sectors per track to be adjusted so more sectors are stored on the larger, outer tracks. By dividing the outer tracks into more sectors, data can be packed uniformly throughout the surface of a platter, disk surface is used more efficiently, and higher capacities can be achieved with fewer platters. The number of sectors per track on a typical 3.5 in. disk ranges from 60 to 120 under a multiple zone recording scheme. Not only is effective storage capacity increased by as much as 25% with multiple zone recording, but the disk-to-buffer transfer rate also is boosted. With more bytes per track, data in the outer zones is read at a faster rate.

Based on the organization of data on disks, the access time for a disk is given by the seek latency of the disk head, the rotational latency, and the transfer rate. The seek latency is the time to move the disk arm to the desired track (2). Average seek times are in the range of 10 ms to 15 ms. The time for the requested sector to move under the disk head is called the rotational latency. The transfer time is the time to transfer the bits in the sector under the read/write head. This is function of the block size, the rotation speed, the recording density of the track, and the speed of the disk controller.

Table 2 shows the disk parameters for the current high-end disks. Trends in disk technology are moving toward faster recording density; hence faster transfer rates and lower seek times (about 25%), and spindle speeds up to 10,000 rpm are evident. The speed of a magnetic disk is much lower compared to the main memory. We describe various schemes for reducing the gap in performance in detail later. Having described the architecture of main memory and secondary memory (or magnetic disk) we now discuss interleaving as a method to improve performance.

Table 2. Disk Parameter Values

Feature	Current Range
Form factor	3.5 in.
Size	4.55 Gbyte–18.22 Gbyte
Internal transfer rate	120–190 Mbyte/s
Formatted transfer rate	10–17 Mbyte/s
Track-to-track seek	0.8–1.5 ms
Mean seek	7–8 ms
Rotational latency	4 ms
Spindle speed	5400–7200 rpm

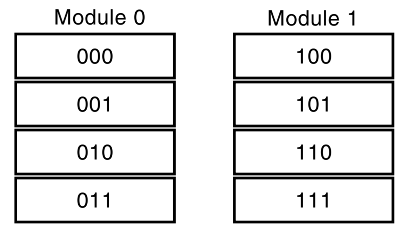

Figure 2. High-order interleaved memory.

MEMORY INTERLEAVING

In an interleaved memory system, the memory is divided into a set of banks or modules to speed up sequential access (1). An interleaved memory with modules is called n-way interleaved. The mapping of memory address to the banks affects the performance of the memory system. The mapping is referred to as the interleaving factor. These are two basic types of memory interleaving based on the assignment of the address to the memory modules.

- *High-order memory interleaving.* In high-order interleaving the consecutive memory addresses are stored within the same memory module (except the boundary conditions). For example, for a machine with a 32 bit address space and 16 memory modules, the ith module would contain addresses ranging from $i2^{28}$ to $(i + 1)2^{28} - 1$. Figure 2 shows the interleaving for two memory modules.
- *Low-order memory interleaving.* For the same example of a 32 bit address space and 16 memory modules, with low-order interleaving the ith module contains all addresses whose least significant 4 bits evaluate to i. Thus consecutive memory addresses are stored in consecutive modules. This word interleaving is ideal for sequential accesses to memory. Figure 3 shows the interleaving for 2 memory modules.

Low-order interleaving is useful when the memory cycle is significantly longer than the CPU cycle. If CPU were much faster than memory, and a high-order interleaving is used, then for consecutive memory access, the CPU would have to wait until the previous memory access is completed. If low-order interleaving is used, then consecutive memory locations are in different banks and they can be accessed at the same time. The decision to allocate addresses as contiguous blocks (high-order interleave) or in a striped manner (low-order in-

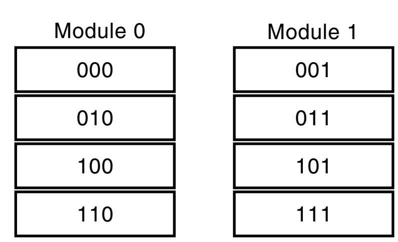

Figure 3. Low-order interleaved memory.

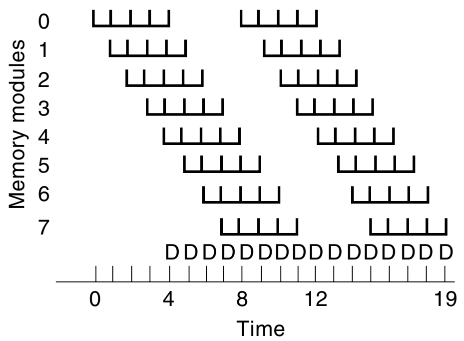

Figure 4. Gantt chart for accessing interleaved memory.

terleave) depends on how one expects information to be accessed. Typically programs are compiled to have instructions stored in successive address locations. Vector elements could also be stored in contiguous addresses. Such linear executions or vector operations benefit from low-order interleaves. However, shared memory multiprocessors use block-oriented schemes and connect an entire memory module to a single processor, thereby preferring a high-order interleave.

Analysis of Memory Access Time with Interleaving

For low-order memory interleaving the access time can be determined using a Gantt chart (3). Let each row in the Gantt chart represent a memory module. The time line represents the units in processor cycles. Let the memory cycle time be m. If a processor requests a word from memory module a at time t, draw a horizontal line in row a starting at time t and continuing for m units. Figure 4 shows the Gantt chart for an eight-way interleaved memory in a system where the processor cycle time is 10 ns and the memory cycle time is 40 ns. The chart shows the memory busy time for each module when the processor requests successive memory cells. If data are available to the processor at every cycle, then the memory is performing to its full potential.

The actual performance gain by interleaving varies from computer to computer. Typical numbers range from 8% to 30%. For better performance it is always better to configure a system with interleaved memory than noninterleaved memory. Thus two 16 Mbyte DIMMs will have better performance than a single 32 Mbyte DIMM. A disadvantage of memory interleaving is that making multiple banks is expensive for the same-sized memory. A second disadvantage is the difficulty of memory expansion. Since memory-controlled hardware will need equal-sized banks, the minimum increment will be to double the memory.

DISK INTERLEAVING

The speed of a magnetic disk is a major bottleneck in the overall system performance. Amdahl's law predicts that large improvements in microprocessor speeds will result in only a marginal improvement in overall system performance, unless accompanied by a comparable improvement in secondary storage performance. Currently disk transfer bandwidths are orders of magnitude slower than memory bandwidths. Table 1 shows the ranges in memory and disk speeds. Although with rapidly changing disk technology the disk capacity and transfer rates have been significantly improved, the overall bandwidth is limited by seek times and is still low. Although disk storage densities have improved by 70% every year and costs have fallen from $11 per Mbyte in 1988 to 5¢ per Mbyte, the total disk access times, which depends on mechanical parts, have improved only by around 10% per year. Memory costs have fallen from $50 to $5 per Mbyte. However, adding more memory is not the solution. Memory is volatile. Thus we will assume that the performance of a system will be limited by the I/O bandwidth of nonvolatile storage.

Various techniques have been used to improve the performance of disks. These include the following. (1) *Minimizing the mechanical delays:* To reduce seek delays multiple disks heads are used per surface, the entire cylinder is accessed in parallel by using tracks-in-parallel moving disk heads, or the bit density is increased along a track to improve the transfer rate. The zoned bit recording with fixed density storage is used to fully utilize the capacity of larger tracks. (2) *Minimizing the effect of mechanical delays:* Disk caching and disk scheduling are used to mask the effect of mechanical delays. Caching improves the performance for reads. The disk write performance is improved by writing to cache and delaying the actual disk write. The inertia of the disk head is used to write the cached data on a power failure. Disk scheduling is used to reduce the seek time component of disk delay. Some disk scheduling algorithms used are shortest seek time first (SSTF) and SCAN.

Just as multiple CPUs and instruction pipelining can be used to improve a system's processing capability, multiple disk drives improve a system's I/O capability. For many years, minicomputers and mainframe computers have employed high bandwidth controllers and multiple disk drives. Furthermore, the ever-increasing requirement of space requires the need for using multiple disks.

When using multiple disks, the data or files have to be suitably placed on the disks to utilize disk bandwidth fully. In most cases, perfect file placement is not possible. This is because, on most systems for a given period or time, approximately 80% of all I/O requests go to 20% of the available disk drives. Therefore, the storage system is never balanced. The result is storage system "hot spots" that cause I/O requests to back up in disk queues. This results in an inefficient storage system with one or more disks becoming the bottleneck.

To help solve this problem, the concept of *disk interleaving* was developed. Disk interleaving was first used in the Cray supercomputer to improve the performance of very large data arrays (4). It was later used for large database systems (5) and in implementations of Unix (6). Disk interleaving or striping is a method of coupling a group of disks together (7). Groups of disks are interleaved if consecutive portions of data are on different disks. Data are broken down into fixed size chunks and distributed across the stripe set volume. The result is an even distribution of "hot spots" across the set of drives. In this way, the full I/O bandwidth capability is available and the system's aggregate performance improves.

The granularity of interleaving (or the stripe unit size) is the size of a contiguous unit of data stored on each disk. The degree of interleaving (or striping width) is the number of disks used to store the data. The granularity of disk interleaving can be chosen at any level. It could be at the attribute level or at the record level, at the block level or at the byte level. Whatever the level of interleaving chosen the goal is to utilize the inherent parallelism provided by disk interleaving (8).

Synchronized Disk Interleaving

With synchronized interleaving, byte B_i in a block of data is assigned to disk unit (B_i mod n). Thus byte 0 is assigned to disk 0 and byte 1 to disk 1 and so on. Since adjacent bytes of a block of data are at the same place on each disk, the rotation of all disks can be synchronized. The granularity of synchronized interleaving can be byte level, sub-block level, or block level (9). By synchronizing multiple disks they can be treated as a single disk unit thus simplifying the control. However, as more disks are added, the performance may suffer significantly from possible interference. The advantages of synchronized disk interleaving are (1) simplified control, (2) parallelism through interleaving, (3) single logical image of interleaved disks, and (4) facilitating uniform distribution of access requests over multiple disks.

Asynchronous Disk Interleaving

In asynchronous interleaving the blocks of data are placed independently of each other on the disks (10). This is in contrast to synchronous interleaving, where the data are placed at the same physical location or a predetermined location on disk. In an asynchronous system the disks are independent of each other and the data belonging to a block are also stored independently. As a result, the seek and rotational latencies involved in the same transfer will be different for each disk. Asynchronous interleaving is more suitable when the number of disks in the system are large and the reference patterns are not regular and structured.

Although interleaving is a proven technology that increases parallelism and reduce hot spots, it has several drawbacks. First and foremost, striping makes a large set of data vulnerable to disk failure. Because stripe set data are distributed, when a disk in a stripe set fails, all data in the stripe set are lost. The time to restore a failed stripe set, especially if it contains a large number of disks or high capacity disks, can be significant (11). Second, if disk striping is implemented in software on the host CPU, the system incurs the additional processing overhead of the striping driver.

Redundant Array of Inexpensive Disks

The key problem of interleaving is that as the number of disk drives in a stripe set increases, the aggregate mean time between failure (MTBF) of the stripe set drops dramatically. An MTBF of 200,000 h (or 23 years) for a single disk implies an MTBF of 2000 h (or three months) for an array of 100 disks. The conclusion is that performance significantly improves at the expense of availability.

In 1987, redundant arrays of inexpensive disks (RAID) was proposed by Patterson, Gibson, and Katz (12). (RAID was subsequently renamed to redundant array of independent disks.) To solve the MTBF problem, RAID introduced the concept of using redundancy to ensure data availability. Redundancy, however, has its disadvantages. The write of data requires the update of redundant information, slowing down writes.

The different types of redundancy and striping schemes were originally classified into five RAID levels, RAID 1 through RAID 5 (13). Subsequently, levels 0, 6, and 7 were added. The RAID schemes differ in two respects: (1) the granularity of interleaving and (2) the pattern in which redundant information is distributed across disks (14).

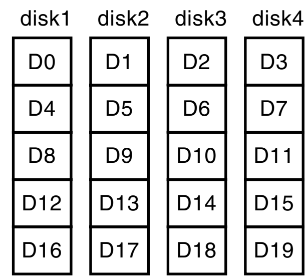

Figure 5. RAID level 0.

RAID 0 is interleaving without storing any redundancy information. Figure 5 shows the interleaving across multiple disks without any redundant data. RAID 1 (mirroring) is the simplest form of RAID that stores redundant information. It entails using disk mirroring (shadowing) to duplicate the information stored on a disk. Whenever data are written to a disk the same data are also written to a mirror disk so that there are always two copies of the information. Figure 6 shows the placement for an eight-disk system with four of the disks used to store the mirrored blocks. The read performance of RAID 1 can be very good. When used in conjunction with an intelligent controller, multiple read commands can be processed simultaneously by a shadow set. It also is possible to select the disk whose read/write heads are closest to the desired data, thereby reducing access time and improving performance. Conversely, the write performance of a RAID 1 system is slightly worse than a single-disk write operation. This is because both disks in the shadow set must be written to for each write operation.

Because most systems have a much higher percentage of reads than writes, mirroring can significantly improve system I/O performance. However, it does not solve the "hot spot" problem. Furthermore, shadowing is expensive. In essence, each component of the disk storage system must be duplicated (i.e., disks, controllers, cables, cabinets, power). For this reason, RAID 1 only is practical for remote mirroring, where maintaining system availability during a catastrophic disaster (such as a fire or flood) is imperative.

RAID 2 (memory-style ECC) uses a memory-style Hamming error-correction code (ECC) that can be used for data reconstruction in the event of a disk failure. The Hamming code technique was developed in the 1950s for large arrays of DRAM. Hamming codes contain parity information for distinct overlapping subsets of components. The RAID 2 method stripes bits of data across multiple disk drives. The number of redundant disks is proportional to the logarithm of the total number of disks in the system. The storage efficiency of RAID

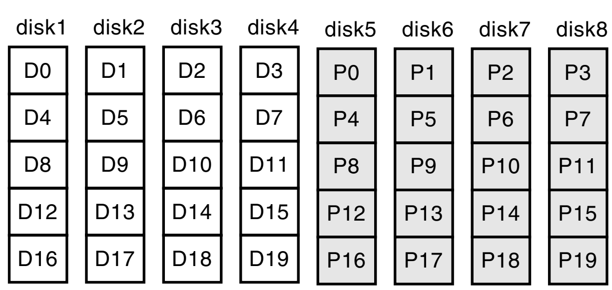

Figure 6. RAID level 1.

2 increases as the number of disks increases. A typical RAID 2 configuration uses 10 data drives and four Hamming ECC drives. Using RAID 2, a single I/O operation accesses all drives. For this reason, the drive spindles must be synchronized. In this configuration, rotational latency (the delay time from when a read/write head is on-track and when the requested data passes under it) is the same as a single drive.

Because data bits are read in parallel, performance of RAID 2 for large data transfers can be excellent (transfer rare is the sum of the data disks). However, this is not true for small data transfers. With the disks operating completely in parallel, small transfer requests have the same performance characteristics as a single disk. Thus, for most systems, performance gain is not realized. In fact, compared with 14 individual disks, performance of RAID 2 for small to medium size data requests is considerably less. Further, because additional disks are required for the Hamming ECC information, storage efficiency is significantly reduced.

RAID 3 (bit interleaved parity) replaces RAID 2's Hamming ECC disks with a single parity disk. RAID 3 improves upon memory-style ECC disk arrays by noting that unlike a memory controller, disks controllers can easily identify the failed disk. Thus a single parity disk can be used instead of a set of parity disks used in RAID 2. The ECC (parity) is produced by performing an exclusive OR (XOR) operation on the data. The result of the XOR is stored on the parity disk. In the event of a disk failure, data from the failed disk can be reconstructed by reading the remaining disks and calculating the missing bits using the parity data. Using this method, storage efficiency is significantly increased. RAID 3 storage efficiency is calculated as $n/(n + 1)$, where n is the number of data disks in the array. Like RAID 2, RAID 3 provides excellent large transfer I/O characteristics, but small and medium I/O transfers are not efficient. Bit interleaved parity disks are used in applications that require high bandwidth but not high I/O rates.

RAID 4 (block interleaved parity) uses a different approach. Rather than storing individual bits of data on separate disks, data are stored in fixed block sizes called striping units. Each block of data is stored on a different disk as used in disk striping. Blocks are read and written independently. Also, the spindles are not synchronized.

RAID 4 redundancy is obtained through the use of a parity disk. When block is written to a data disk, parity for that block is written to a corresponding block on the parity disk. Because a block on the parity disk contains the parity for the corresponding blocks on all data disks, whenever data are written the existing XOR data must be read, updated, and rewritten (called the read-modify-write procedure). This results in an extra rotation of the parity disk. Because of the amount of activity on the parity disk, it can easily become a bottleneck.

RAID 4 read performance is good. Because I/O is independent to each drive, performance is improved through the use of multiple head actuators for small data transfers and through parallelism on large data transfers. RAID 4 write performance is poor due to the implementation of parity. Storage efficiency is the same as RAID 3. Figure 7 shows the placement for RAID 4 with each parity group consisting of four data blocks and one parity block.

RAID 5 (block interleaved distributed parity) resolves the RAID 4 parity disk bottleneck. RAID 5 distributes (stripes) the parity blocks among all disks in the array thereby, evenly distributing the load. Since the data are distributed across all disks instead of all but one disks in RAID 4, it allows for all the disks to participate in servicing read requests. RAID 5 read performance is similar to RAID 4, while write performance is significantly improved. RAID 5 has one of the best small read, large read, and large write performance compared to any other RAID scheme. Small writes are somewhat inefficient compared to schemes like mirroring due to the read-modify-write operation used to update parity. The methods used to distribute parity have an impact on the performance of RAID 5. The left-symmetric parity placement shown in Fig. 8 has one of the best distributions of parity. A property of left-symmetric placement is that on a sequential traversal of blocks each disk is accessed once before any disk is accessed the second time. This property reduces conflicts for large reads. Storage efficiency of RAID 5 is the same as RAID 3 and RAID 4. With parity distributed across all drives, data integrity of RAID 5 is excellent. For data to become unavailable in a RAID 5 system, two drives in the array must fail.

The drawback of standard RAID 5 is that the performance degradation after failure may be unacceptable for various applications like transaction processing and real-time video service. In the worst case a workload of small reads will double the effective load per disk on each of the functioning disks due to the extra disk accesses needed to reconstruct data for reads to the failed disk. In systems that stripe data across multiple parity groups the average increase in load is significantly less than in RAIDs with one large parity group. However, the parity group with the failed disk still experiences a 100% increase in load after failure in the worst case. The declustered parity organization solves this problem by uniformly distributing the load over all disks. The scheme used to distribute load uniformly in a declustered parity arrangement is to create a set of parity groups that includes every

disk1	disk2	disk3	disk4	disk5
D0.0	D0.1	D0.2	D0.3	P0
D1.0	D1.1	D1.2	D1.3	P1
D2.0	D2.1	D2.2	D2.3	P2
D3.0	D3.1	D3.2	D3.3	P3
D4.0	D4.1	D4.2	D4.3	P4

Figure 7. RAID level 4.

disk1	disk2	disk3	disk4	disk5
M0.0	M0.1	M0.2	M0.3	P0
M1.0	M1.1	M1.2	P1	M1.3
M2.0	M2.1	P2	M2.2	M2.3
M3.0	P3	M3.1	M3.2	M3.3
P4	M4.0	M4.1	M4.2	M4.3

Left-symmetric data organization in
RAID level 5 disk array with G=D=5

Figure 8. RAID level 5.

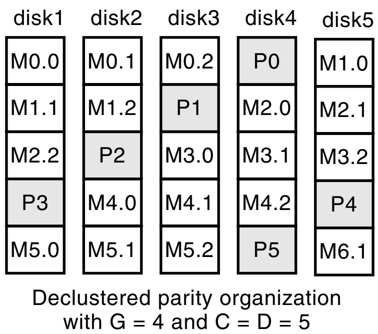

Figure 9. Declustered parity in RAID level 5.

possible mapping of parity groups members to disks (15–17). For eight disks and a parity group of size four it would create $\binom{8}{4}$ distinct mappings. Figure 9 shows a declustered parity placement.

RAID 6 (P + Q redundancy) uses two-dimensional parity computation to handle multiple failures. Conceptually the disks are considered to be in a matrix formation and the parity is generated for the rows and columns of the disks in the matrix. The P × Q redundancy scheme uses the Reed–Solomon codes to protect against two disk failures using a minimum of two redundant disks. The disk array is structured similar to the RAID 5 array.

Raid 7 supports heterogeneity, where the disks are asynchronous and independent with differing characteristics. It is the most recent development in the RAID taxonomy. The RAID 7 architecture has an independent structure with a separate device cache, device control and an embedded operating system. It allows easy configuration since drives of different capacities, access times, transfer speeds, and form factors can interconnect, allowing expandability to suit future requirements. Another important feature of RAID 7 is dynamic mapping where a block of data need not be written to the same location after an update.

BIBLIOGRAPHY

1. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach,* San Mateo, CA: Morgan Kaufmann, 1990.
2. C. Ruemmler and J. Wilkes, An introduction to disk drive modeling, *IEEE Comput.,* **27** (3): 17–29, 1994.
3. K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability,* New York: McGraw-Hill, 1993.
4. O. G. Johnson, Three-dimension wave equation computations on vector computers, *Proc. IEEE,* **72**: 905, 1984.
5. R. Agrawal and D. J. DeWitt, Whither hundreds of processors in a database machine, *Proc. Int. Workshop High-level Arch.,* 1984.
6. J. R. Lineback, New features tune unix for high-end machines, *Proc. Electron.,* August 1985.
7. K. Salem and H. Garcia-Molina, Disk striping, *Proc. IEEE Data Eng. Conf.,* 1986, pp. 336–342.
8. S. Khoshafian, M. Livny, and H. Boral, Multidisk management algorithms, *Proc. ACM SIGMETRICS,* 1987, pp. 69–77.
9. M. Y. Kim, Synchronized disk interleaving, *Proc. IEEE Trans. Comput.,* **C-35**: 978–988, 1986.
10. M. Y. Kim and A. N. Tantawi, Asynchronous disk interleaving: Approximating access delay, *Proc. IEEE Trans. Comput.,* **40**: 801–810, 1991.
11. J. Chandy and N. A. L. Reddy, Failure evaluation of disk array organizations, *Proc. Int. Conf. Distributed Computer Syst.,* May, 1993.
12. D. Patterson, G. Gibson, and R. Katz, A case for redundant arrays of inexpensive disks (RAID), *Proc. ACM-SIGMOD Int. Conf. Manage. Data,* Chicago, 1988.
13. G. A. Gibson and D. A. Patterson, Designing disk arrays for high data reliability, *J. Parallel Dist. Comput.,* **17** (1–2): 4–27, 1993.
14. P. M. Chen et al., RAID: High-performance, reliable secondary storage, *ACM Comput. Surveys,* **26** (2): 145–188, 1994.
15. M. Holland and G. A. Gibson, Parity declustering for continuous operation in redundant disk arrays, *Proc. Architectural Support for Programming Lang., Oper. Syst.,* 1992; also *SIGPLAN Notices,* **27** (9): 23–25, 1992.
16. M. Holland, G. A. Gibson, and D. P. Siewiorek, Fast, on-line failure recovery in redundant disk arrays, *Digest Papers FTCS-23 23rd Int. Symp. Fault-Tolerant Comput.,* Los Alamitos, CA: 1993, pp. 422–431.
17. M. Holland, G. A. Gibson, and D. P. Siewiorek, Architectures and algorithms for on-line failure recovery in redundant disk arrays. *J. Distr. Parallel Databases,* **2** (3): 295–335, 1994.

RENU TEWARI
HARRICK M. VIN
The University of Texas at Austin