Storage Systems for Data Warehousing

Alexander Thomasian
New Jersey Institute of Technology - NJIT, USA

José F. Pagán
New Jersey Institute of Technology - NJIT, USA

INTRODUCTION

Data storage requirements have consistently increased over time. According to the latest WinterCorp survey (http://www/WinterCorp.com), "The size of the world's largest databases has tripled every two years since 2001." With database size in excess of 1 terabyte, there is a clear need for storage systems that are both cost effective and highly reliable.

Historically, large databases are implemented on mainframe systems. These systems are large and expensive to purchase and maintain. In recent years, large data warehouse applications are being deployed on Linux and Windows hosts, as replacements for the existing mainframe systems. These systems are significantly less expensive to purchase while requiring less resources to run and maintain.

With large databases it is less feasible, and less cost effective, to use tapes for backup and restore. The time required to copy terabytes of data from a database to a serial medium (streaming tape) is measured in hours, which would significantly degrade performance and decreases availability. Alternatives to serial backup include local replication, mirroring, or geoplexing of data.

The increasing demands of larger databases must be met by less expensive disk storage systems, which are yet highly reliable and less susceptible to data loss.

This article is organized into five sections. The first section provides background information that serves to introduce the concepts of disk arrays. The following three sections detail the concepts used to build complex storage systems. The focus of these sections is to detail: (i) *Redundant Arrays of Independent Disks* (RAID) arrays; (ii) multilevel RAID (MRAID); (iii) concurrency control and storage transactions. The conclusion contains a brief survey of modular storage prototypes.

BACKGROUND

The fifty year old magnetic disk drive [technology] remains a viable storage medium because they can accommodate an excess of 500 Gigabytes (GB), are nonvolatile, inexpensive, have an acceptable random access time (10 milliseconds), and exhibit a *Mean Time to Failure (MTTF)* exceeding 10^6 hours. Concurrent with their benefits, disk failures occur frequently in large data centers.

RAID serves to mitigate disk failures in large installations (Chen et al. 1994). RAID level 5 (RAID5) masks the failure of a single disk by reconstructing requested blocks of a failed disk, on demand. Additionally, it automatically reconstructs the contents of the failed disk on a spare disk.

The RAID paradigm is inadequate for *Very Large Disk Arrays (VLDA's)* used in data warehousing applications, because the non-disk components may be less reliable than the physical disks. *Hierarchical RAID (HRAID)* achieves a high reliability by using multiple levels of RAID controllers (Baek et al. 2001). The higher and lower RAID levels of HRAID are specified as RAIDX(M)/Y(N), where X and Y are the RAID level and *M* and *N* denote the number of virtual disks or *storage nodes (SN's)* at the higher level and physical disks at the lower level. *Multilevel RAID (MRAID)* was proposed as an alternative to HRAID, because of its two key differences: (i) disks are organized into *SN's* (or bricks) at the lower level, this constitutes the *smallest replaceable unit (SRU)*, (ii) the association among SN's is logical and dynamic rather than hardwired (Thomasian 2006).

Each SN consists of an array of disks, an array controller, a partially nonvolatile cache, and the capability to interconnect. SN costs are kept low, in some designs, by using mirroring to protect data on each SN.

The internal structure of a brick in IBM's Intelligent Brick prototype is illustrated in (Wilcke et al. 2006)

Figure 2. Bricks are cube shaped and communicate via capacitive coupling between insulated flat metal plates at each of its six surfaces. Higher capacities are attained by stacking bricks on top of each other. Gigabit Ethernet is used to provide connectivity to external cubes. A fail-in-place or deferred maintenance system is also postulated.

MAIN THRUST

We first review RAID systems, before discussing multilevel RAID. We next describe storage transactions, which are required for the correct operation of the system.

RAID LEVELS

There are seven RAID levels (Chen et al. 1994) which are classified as *k disk failure tolerant (kDFT)* arrays (Thomasian et al. 2007). RAID0 is 0DFT because it has no redundancy. Data is divided into *striping units (SUs)*, which are written to the disks in a round-robin manner. This technique of writing data, in small units, across multiple disks is termed striping. Striping also serves to balance the load across an array of *N* disks.

RAID1 is also referred to as *Basic mirroring (BM)*. The simplest RAID1 configuration requires two disks, where data is written such that both disks are exact copies of each other. RAID1 has *M = N/2* disk pairs, where *N* is the number of physical disks. RAID1 can tolerate up to M disk failures (one disk of a pair), however failure of both disks (in a pair) will lead to data loss. Hence, RAID1 is 1DFT. BM has twice the read transaction rate of a single disk because reads are duplexed between the disk pairs. Similarly, a single disk failure will shift the load to the working disk and essentially double the load on the surviving disk.

In *Interleaved Declustering (ID)* the total number of disks (*N)* are divided into *c* groups, so that there are *M = N / c* disks per group. In this scheme, disks are partitioned into primary and secondary areas. Data is written completely to a primary area of a single disk. Additionally, this data is divided into blocks and written to the secondary areas of the remaining *M – 1* disks in the group. Each group is a 1DFT array and a single disk failure will increase the load on the surviving disks by *M / (M – 1)*.

In *Chained Declustering (CD)* the primary data on each disk is replicated to the secondary area of the next disk modulo *N*, e.g., for $i < N$ $(N+i) \bmod N = i$.

In *Group Rotate Declustering (GRD),* data is striped across a group of disks (*M = N / 2)* and replicated (in a rotated manner) across the remaining *M* disks. The load on surviving disks in CD and GRD may be balanced with appropriate routing of read requests. The tradeoff between load balancing and reliability in these RAID1 configurations is discussed in (Thomasian and Blaum 2006). The aforementioned paradigms can be applied to bricks and to arrays of multiple bricks, thus facilitating a balanced load on failures.

RAID2 uses Hamming codes which are inefficient and increases the number of check disks in proportion to the logarithm of existing data disks.

RAID3, RAID4, and RAID5 are 1DFTs because they require one disk for parity. RAID3 uses small SUs which are intended for parallel data transfers, thus increasing the transfer rate. RAID3 is most efficient with synchronized disks. RAID4 allocates the parity SU on the Nth disk which is computed from the SUs on the first *N-1* disks. On disk *N*, let D_i, $1 \le i \le N\text{-}1$ denote the first *N-1* SUs. The parity of the SUs is: $P_{1:4} = D_1 \oplus D_2 \oplus D_3 \oplus D_4$ where $\oplus$ is the XOR operator, such that $1 \oplus 1 = 0$. If disk 1 fails then block d_1 in SU D_1 can be reconstructed as $d_1 = d_2 \oplus d_3 \oplus d_4 \oplus p_{1:4}$.

When data block d_1 is updated, instead of reading all corresponding blocks, we can update the parity block by computing $d_1^{diff} = d_1^{old} \oplus d_1^{new}$ and using d_1^{diff} to compute $p_{1:4}^{new} = p_{1:4}^{old} \oplus d_1^{diff}$. RAID4 has two disadvantages: (i) the parity disk is not accessed by read requests, (ii) it becomes a hot-spot when write requests dominate. RAID5 uses the left-symmetric distributed parity layout. Here, blocks are arranged in left to right diagonals. Reading a block on a failed disk, in RAID5, will generate reads to corresponding blocks on the surviving *N - 1* disks, thus doubling the load on these disks. A solution to this problem is *clustered RAID – CRAID*, which uses a parity group of size *G < N*, so that load increase to the other disks is $\alpha = (G - 1)/(N - 1)$.

RAID6 is a 2DFT with two parity disks, P and Q parities. Reed-Solomon codes and specialized parity codes: EVENODD, *Row Diagonal Parity - RDP*, X-codes, and RM2 (Thomasian and Blaum 2007) are typically used. X-codes require *N* to be prime, while