Paper #1014

Caching Disk Subsystems: Bridging the Gap Between High Performance and High Availability

Presented by

Brett Kelleran
SEEK Systems
11014 120th Ave NE
Kirkland, WA 98033
(206) 822-7400
brettk@seek.com

Introduction

The requirements placed on I/O subsystems have grown at a staggering pace over the last several years. Factors in this growth include: an exponential increase in CPU power; a boom in network bandwidth and activity, including internet and intranet applications; and new methods for managing and analyzing data, with techniques such as data warehousing and data mining. Unfortunately, magnetic disk drive technologies, while showing dramatic improvements, have failed to keep pace with the rest of the computer industry.

Several methods have been used to address the limitations of magnetic disk drives. Roughly speaking, these techniques may be broken down into two areas: methods to improve availability, and methods to improve performance. Availability techniques have often centered around RAID (Redundant Arrays of Independent Drives) technology. This includes mirroring (RAID level 1), as well as higher levels of RAID which offer more economical use of disk space. Performance techniques have centered on balancing the load across multiple disk drives, along with implementing disk caching at the host level.

Unfortunately, many of these performance and availability techniques were seen as being mutually exclusive. Early implementations of RAID did not meet performance expectations, with the RAID 5 write penalty becoming particularly infamous. Conversely, spreading data across multiple drives makes this data much more vulnerable to disk drive failures.

Over the past few years several methods have surfaced to try to provide both performance and availability. RAID 0+1 (mirrored stripe sets) offers reasonable performance and high availability, but requires twice as much disk space. Solid State Disk has provided a high performance and availability alternative, but at a cost that is prohibitive to most applications. And most RAID controllers have implemented buffer caches that help improve performance, including write-back cache which can be used to essentially negate the RAID 5 write penalty.

Recently, a new hybrid of Caching Disk Subsystems has surfaced which utilize advanced caching algorithms and large amounts of memory to provide the most effective coupling to date of high availability and high performance. By using these algorithms to store active data in memory while flushing inactive data to disk, the Caching Disk Subsystem can provide performance that nears Solid State Disk, at a fraction of the cost. And by coupling the caching with a RAID (or hybrid RAID) back end, the Caching Disk Subsystem offers high availability as well.

Cache Utilization

In the past few years the speed at which a CPU can load and store data has greatly outpaced the ability of magnetic storage devices to supply this data. In an attempt to lessen the impact of this performance mismatch, most systems now include various caches. A cache is a relatively small amount of very fast solid state memory which sits between the CPU and the data storage subsystem. In cache memory is a duplicate of some of the information which exists in magnetic memory. Since the time required to access the information in the cache is much shorter than the time needed to access information residing on magnetic media, system performance can be greatly improved by maximizing the use of the cache contents, thereby achieving high cache utilization and lower disk activity.

Buffer Cache

Traditionally, most disk cache has been an I/O buffering tool using a Least Recently Used (LRU) algorithm. The result is similar to a stack of cafeteria trays, in which trays are added to the top, and removed from the bottom. This is the most common type of caching algorithm, and as the most recently requested data is read into cache, it pushes out the "oldest" data. Buffer cache is only meant to hold data on its way to or from a disk, so is usually fairly small (8Mb - 64Mb).

Buffer cache may be configured as Write-Through or Write-Back. Write-Through cache passes all writes directly on to disk, so that the data in cache always matches what is on disk. Performance improvements with Write-Through cache are limited to improving read performance through read cache hits. Write-Back cache stores writes in cache, waiting for an optimum time to write the information to disk. Write-Back cache offers the best performance improvements, but requires a non-volatile cache environment as data can be held in cache for an extended period of time before writing it to disk.

For additional performance gains, buffer cache algorithms can be combined with prefetch of reads and concatenation of writes to further optimize performance when a read or write to disk is required. Utilizing prefetch, when the host requests data not in cache, the requested data and data immediately after this data is read into cache. There is a high likelihood that data near requested data will be requested next, and one large operation to disk is much more efficient than two smaller ones. For writes to disk, concatenation of writes increases performance in much the same way that prefetch improves read performance. When writeback to disk is required, the controller organizes data according to where the data resides on the physical disk, and writes it accordingly. The result is fewer, larger, more efficient writes to disk.

Significant performance improvements can be seen with buffer cache, especially with Write-Back cache. This is particularly true in a RAID 5 environment, where Write-Back cache can effectively negate the write penalty. However, there are limitations associated with the fact that it is essentially a buffering tool. As such it handles bursts of activity, but does not perform as well in a sustained high activity environment.

Adaptive Cache

Adaptive cache, in contrast to buffer cache, looks to store data semi-permanently in memory. By keeping active data in cache and migrating inactive data to disk, an adaptive cache can drastically improve the overall performance of an I/O subsystem. In general, adaptive caches are much larger than buffer caches, typically ranging in size from 64 MB up to 1 GB and beyond.

The prime measure of effectiveness of any cache system is the cache hit rate. Any Caching Disk Subsystem is therefore dependent on its caching algorithms. Because memory is so much more expensive than magnetic disk, it is imperative that active data remain in cache, while inactive data, and only inactive data, be moved to disk. When inactive data pushes active data out of cache, the Caching Disk Subsystem suffers from a condition called cache pollution, a condition that is extremely detrimental to performance.

There are a number of algorithms that may be employed in an adaptive cache to limit cache pollution. The goal of all these algorithms is the same: to identify hot data and keep this data in cache. In general, these algorithms analyze data patterns as they are received from the host in an effort to anticipate what will be accessed in the future. One method implemented by advanced Caching Disk Subsystems is to use a Least Frequently Used (LFU) algorithm along with the Least Recently Used (LRU) method found in most buffering caches.

To see how this might work, consider a Caching Disk Subsystem that actually segments the cache into two parts, a buffered cache and a protected cache. The buffered cache uses an LRU algorithm as previously described. However, the Caching Disk Subsystem also monitors how often each block of data is accessed. Blocks which exceed a certain threshold are declared "hot", and moved to protected cache. Here the active data remains in cache until the data access patterns of the host change, and they are replaced with data that has become more active. Such an approach combines two performance improvements. It buffers all data with an effective LRU algorithm, and it continually monitors, updates, and retains active data in protected cache with a Least Frequently Used (LFU) algorithm. This ensures that the most heavily used data is kept in protected cache, available at solid state speed and safe from other data moving through the buffer cache.

How Does It Work?

To understand how an adaptive cache minimizes cache pollution and optimizes performance, we must define clean, updated and protected data. Clean data is data in cache that exactly matches the same information stored on magnetic disk. Updated data is data in cache that does not match the corresponding data on disk because it has been updated in cache and not yet written to disk. Protected data is data that has been flagged to remain in protected space and may be either clean or updated.

Consider the case of a read. On a read, the controller first looks to see if the data is in cache. If it is, the read can proceed without disk I/O. If the requested data is not in cache, the data plus the extra data in the prefetch, is read from disk to cache and the requested data sent directly to the host. The least recently used data is cleared to make room for the new data (Figure 1).

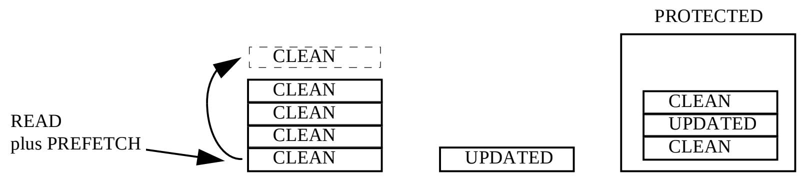

FIGURE 1

On a write, the controller first checks to see if the information is in cache. If it is in cache, the appropriate data is updated and flagged as such and the host immediately released. If it is not in cache, the least recently used clean line is cleared and the new data is written to cache, flagged as updated and the host released (Figure 2).

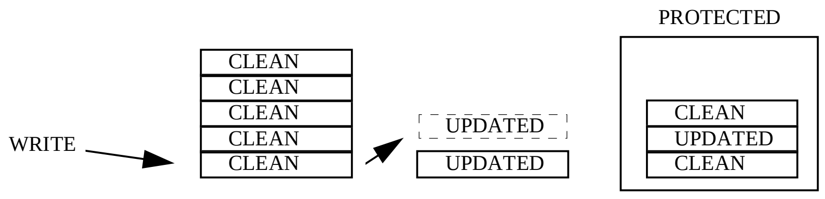

FIGURE 2

When any data in cache is accessed (read or write) more than a set number of times, it is flagged as protected (Figure 3). Once data is protected, the only way it may be cleared from protected space is to be pushed out by more frequently accessed protected data. What this application of the LFU algorithm does is ensure that the hottest data is always in cache and available at solid state speed.

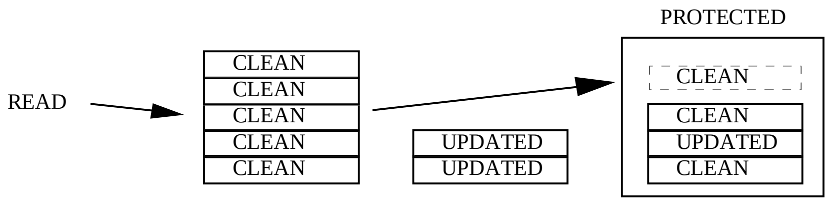

FIGURE 3

Eventually, and periodically, data must be written to disk. Typically three events can initiate a write to disk: the updated stack gets too large; the clean stack gets too small; the controller senses inactivity from the host. When any of these occur the updated data is sent to the write-back queue where they are sorted for the most efficient writes. Once updated data is written to disk it is flagged as clean and placed on the clean stack.

Bottom Line Performance Gains

The following chart (chart 1) underscores the importance of the caching algorithms just discussed. As the data clearly demonstrates, it is the cache hit rate that determines the I/O performance of the Caching Disk Subsystem.

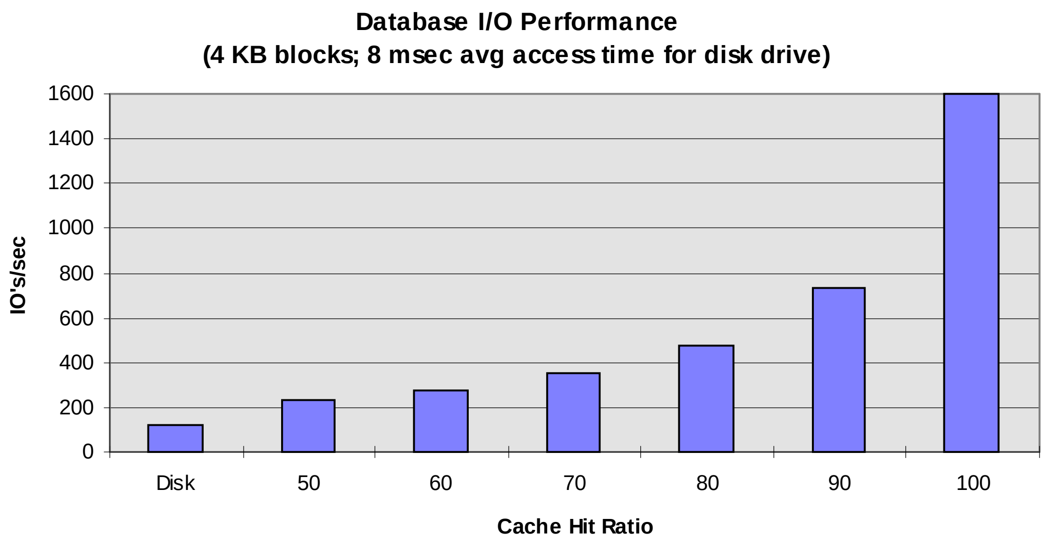

Chart 1

A Caching Disk Subsystem offers the user a chance to choose the performance required for an application. By adjusting the cache to disk ratio of the subsystem, the cache hit ratio can be increased or decreased. This also allows scalability as applications and the data they require grow.

Many variables go into sizing Caching Disk Subsystems. I/O access patterns, size of data storage, and the requirements of the endusers are all important variables that must be taken into consideration. Most vendors will provide a consultation service to help customers size a system properly. When evaluating the performance of a system, it is always important to have a good metric by which to judge success or failure. This is especially true with Caching Disk Subsystems. Whether this metric is an important batch job, screen refresh time, or the percentage of time the CPU spends waiting on I/O, it is essential to have something definite and repeatable on which to perform benchmarks.

Availability

While the caching algorithms of the Caching Disk Subsystem provide the performance, there are many factors which provide availability. These implementations will vary significantly between systems, but in general one should expect to see availability addressed at three levels: the system level, the controller level, and the drive level.

SYSTEM AVAILABILITY:
System availability includes power, cooling, maintenance, and monitoring capabilities. Power is often handled by redundant power supplies, and should be backed up with some type of UPS when Write-Back cache is implemented. Cooling will often use redundant fans as well as some type of temperature monitor inside the cabinet or Caching Disk Controller. Maintenance may be designed to be handled by either technical or non-technical personnel, but in either case most work should be able to be done on line (e.g., via hot-swappable components). Monitoring tools vary greatly from device to device, so it is important to choose a system that meets one's particular requirements. Many Caching Disk Subsystems may be monitored remotely through a dial-up or network connection.

CONTROLLER AVAILABILITY:
The controller of the Caching Disk Subsystem is a solid state device and is therefore the most reliable piece of the system. However, in truly high availability applications a dual controller may be used. In an active-passive configuration one controller waits idly, ready to take over should the other controller fail. In an active-active configuration both controllers are functional, and each can assume the function of the other should it fail. When Write-Back cache is implemented, any dual controller configuration requires mirrored caches to ensure data integrity.

DRIVE AVAILABILITY:
Magnetic disk drives, with all of their moving parts, are the most common point of failure in a Caching Disk Subsystem. As a result, most implementations will build in some redundancy at the drive level. Typically this will be a mirroring (RAID 1) configuration for smaller systems, and higher RAID levels for larger systems.

RAID Levels and Caching Disk Subsystems

In the past, choosing a RAID level has typically meant choosing between RAID levels 3 and 5. In a single-user, sequential environment, RAID 3 was selected, while in a multi-user, semi-random environment (such as database applications), RAID 5 was chosen. However, two factors have recently begun to change this. The first is the fact that in a Caching Disk Subsystem, the controller manages all I/O between the disks and the host, thereby changing the I/O patterns seen by the disks. The second is the progress that has been made with hybrid RAID levels.

A Caching Disk Controller, when it needs to go to disk, will try to group data in an organized, efficient manner. As part of this process most controllers will try to maximize I/O transaction size, which requires less overhead and increases data rates. This means that in a Caching Disk Subsystem that is operating effectively, a multi-user semi-random environment (such as database applications) can look much more sequential to the disks. In some cases that may appear to be RAID level 5 environments, it may be more effective to run at RAID level 3.

There has also been a surge recently in hybrid RAID levels. Some of these combine existing RAID levels to harness the positive attributes of each, while using cache to effectively mask deficiencies. Others use an adaptive RAID level which will actually adjust RAID levels (usually 1, 3, and 5) on the fly, depending on the access patterns. While all of these implementations offer protection against disk failures, the performance of these new RAID levels is directly tied in to the caching and disk access algorithms of the controller.

Application

A Caching Disk Subsystem is effective in any environment that demands high performance and high availability. System bottlenecks will generally occur in four areas: network, CPU, memory, or I/O. Areas suffering from I/O bottlenecks will see the greatest performance gains with Caching Disk Subsystems. Many system analysis tools exist to pinpoint bottlenecks, both at an application level and at an operating system level. On UNIX systems SAR and IOSTAT are two utilities that can provide valuable insight. Some vendors also offer an initial performance consultation free of charge.

Relational Database Applications

Relational database applications in particular put extreme pressure on I/O response, while at the same time demanding high availability. They are thus strong candidates for Caching Disk Subsystems. Files that drive the highest I/O include temporary tablespace (or workspace), transaction log files (also referred to as before image files), and heavily used indexes. Temporary tablespaces are where all intermediary processing is carried out. For complex queries, such as those found in data warehousing applications, the activity to these tempspaces is particularly write intensive, as data is written and updated repeatedly. Table sorts, updates, joins and similar commands make intensive use of tempspace. Transaction log files are used to keep track of database updates, and are also very write intensive.

Conclusion

Cached Disk Subsystems are playing an increasingly important role in providing the high I/O bandwidth required by today's CPUs and transaction intensive database applications. When coupled with availability techniques such as RAID, these subsystems can be an effective component of the most critical production systems.