Hardware Strategies to Improve Database Application Performance

Whitepaper

Summary
As processing loads grow and storage demands expand, a primary challenge for Database Administrators (DBAs) is to extract maximum performance from applications, especially where performance is dependent on disk I/O. Improvements in drive densities and spindle speeds have been impressive, but they have not kept pace with demands for ever-better performance.

The database application performance challenge can be met without an extraordinary investment in people or capital equipment using new caching, cache management, and RAID technologies.

The Changing Problem
I/O performance can be improved through software and hardware techniques. The traditional software approach has relied on DBAs tuning application software, specifically by indexing disk drive activity, balancing loads, and changing application code. This approach fueled a robust consulting market and it can be effective, but the process is long, tedious, and always requires thoughtful experimentation. In practice, lack of time and talent often makes this option unfeasible.

The traditional hardware approach to improving I/O performance is to adopt a Solid State Disk (SSD). A SSD is silicon that emulates a disk drive. Initially for mainframes and later used in server environments, SSDs increase performance by placing data in solid state storage instead of on a disk. SSDs increase application performance, but at a very high price. SSDs are expensive to buy and require a DBA with accurate knowledge of which specific data should migrate from disk to the SSD. Since SSDs are not "adaptive," meaning their contents must be manually configured; the ever-changing data needs of applications require constant upkeep by an SSD administrator. Without full-time attention, the performance benefits of SSD disappear.

Hands-off Hardware
What if there was a way to increase I/O and improve database application performance that did not require in-depth analysis by DBAs or labor-intense solid state memory? Is a set & forget storage appliance possible?

"Adaptive Cache™" is a memory management technology used by FasFile RAID to intelligently manage cache contents and deliver exceptional performance for database applications, at a fraction of the cost of SSD. Adaptive Cache, without administrator intervention, adjusts its contents to retain the most frequently used information and maximize ongoing application performance.

Traditional Cache
Virtually every storage system contains cache. Cache is used in disk drives, servers, and in RAID controllers -- as little as 32 Megabytes to as much as 16 Gigabytes. Cached data can be retrieved 20 times faster than data on disk, but cache capacity is much more expensive than disk capacity. It is therefore important that critical data reside in cache and less important data reside on disk. This means that the size of the cache in is less important than how cache contents are utilized.

Traditional cache is also called FIFO (First In, First Out) cache because it employs a Least Recently Used purging algorithm. This algorithm continuously places the newest or Most Recently Used data in cache and purges the oldest or Least Recently Used data. This approach is effective for buffering I/O, but it creates a problem called "Cache Pollution." Cache Pollution occurs when cache is filled with recently but infrequently used data. Cache Pollution displaces data that applications use repeatedly, but not recently. The undesirable result is that important data is remanded to slower disk storage, negating much of the performance benefit.

Adaptive Caching
Seek Systems employs Adaptive Cache in its FasFile RAID storage systems. Adaptive Cache is unique because it divides cache into two discrete segments, Dynamic Space and Protected Space, and intelligently manages their contents to maximize performance and eliminate Cache Pollution.

Dynamic Space and Protected Space
Similar to the operation of FIFO cache, FasFile RAID's Dynamic Space uses the Least Recently Used algorithm and serves as an I/O buffer. Unlike FIFO cache, FasFile RAID's Protected Space cache uses a Least Frequently Used purging algorithm.

The Least Frequently Used algorithm tracks the usage of each data block and promotes frequently used data from Dynamic Space to Protected Space. Data in Protected Space is accessed at cache speeds and cannot be overwritten by a simple stream of new data requests.

Protected Space is cache for "hot" data. Unlike a SSD, Protected Space is automatically monitored to keep it populated with the "hottest" data. As applications demand different data at different frequencies, Adaptive Cache repopulates Protected Space to keep it up-to-date.

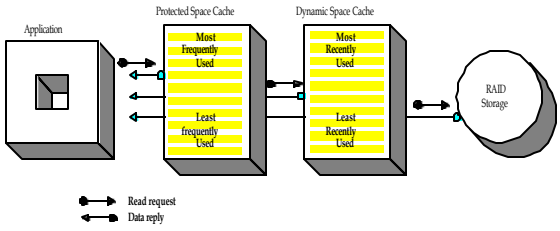


Figure 1 shows the hierarchy of Adaptive Cache.

The database application uses Protected Space cache just as it would RAID disk storage. In other words, Adaptive Cache requires no changes to code or databases design. By constantly monitoring activity and keeping critical data in Protected Space, FasFile RAID delivers sub-millisecond I/O performance for most operations.

Other Acceleration Techniques

Moving critical data from disk to cache is the most important step to faster database application performance, but not the only step. Seek Systems' FasFile RAID also accelerates the disk read and disk write operations.

Algorithms called Prefetch are used for disk reads. When an application requests data that is not in cache, FasFile RAID accesses the disk, reads the requested data, and then reads the data that immediately follows. The additional data is Prefetched because it is likely that the application will subsequently request that data from disk. By performing one large disk read instead of two small ones, FasFile RAID saves resources and pre-positions relevant data in cache for fast access.

If the application requests data that is not sequentially addressed, the Prefetch algorithm uses the addresses to ascertain the address of follow-on data. The amount of Prefetched data may be adjusted for optimal performance. OLTP applications use a Prefetch multiple of 2: for every block of data requested, an additional 2 Prefetched blocks are returned. Sequential operations use Prefetch multiple of 4 or 5.

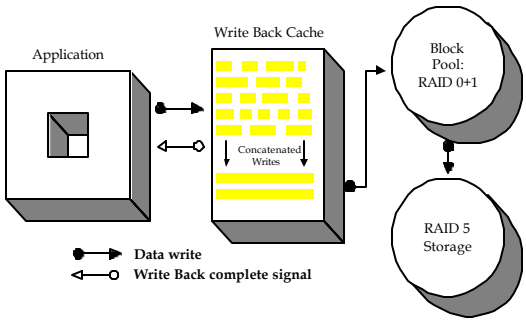
For writes to disk, FasFile RAID improves performance using Write Back cache. Write Back cache holds written data in a buffer, but signals the CPU that the disk write operation is complete. This frees the CPU to perform other tasks instead of idling while data is deposited on disk. FasFile RAID then further accelerates disk writes using a Concatenation process. When an application writes data to disk FasFile RAID's Concatenation algorithm sequences that data in cache based on where it resides on the physical disk. It then writes data to disk in fewer, larger, write operations. Every avoided disk write operation saves precious milliseconds of disk seek time, improving performance for end-users.

RAID Performance Trade-offs

Conventional RAID demands a performance versus efficiency trade-off. This trade-off is rooted in RAID Level 5, the most frequently implemented and commercially successful variant of RAID. RAID 5 makes efficient use of disk space but often uses four distinct write operations to put data onto disk. This is referred to as the "RAID 5 Write Penalty." This penalty consumes resources and impedes overall application performance.

FasFile RAID uses a technique called the Block Pool to avoid the RAID 5 Write Penalty. The Block Pool is reserved space on disks that functions as "virtual cache" for disk write processes. When data is to be written to RAID 5 disk, FasFile RAID avoids the RAID 5 Write Penalty by first writing that data to the Block Pool using swifter RAID 0+1. Then in a background process, FasFile RAID relocates the data from the Block Pool to disk using the more space-efficient RAID 5. The Block Pool eliminates performance-robbing "backpressure" on the application from sluggish RAID 5 dk write operations.

Figure 2 shows the interaction of Concatenated Writes, Write Back Cache and the Block Pool.



The Prime Beneficiary: Database Applications

Interactive applications (e.g., database queries) enjoy the most impressive performance gains from FasFile RAID. Critical database files are moved into Protected Space cache to increase I/O. Commonly cached files include Temporary TableSpace (or Workspace), Transaction Log Files, and heavily used Indexes and TableSpaces. Temporary TableSpace is where intermediary processing occurs. Complex queries that repeatedly write and update data create intense write activity to TempSpaces. Table Sorts, Updates, Joins, and similar commands are also taxing on TempSpace.

Transaction Log files can be very I/O intense with a high percentage of disk writes. Log Files are where the database records its transactions so it can perform a Rollback to previous states, should that be necessary. How these files are used varies by database engine, but queries using commands such as Rollback, Begin, Save, and Commit are heavy users of the Transaction Log files.

FasFile RAID increases database and application server CPU utilization by offloading I/O tasks to the RAID controller and decreasing aggregate disk I/O processes.

Better Performance Is Your Bottom Line

Improvements in drive densities and spindle speeds have not kept pace with demands for higher performance. Until now DBAs had to invest in analysis and administration (people) or expensive capital equipment that requires more administration (more people).

FasFile RAID, gives DBAs a storage solution that increases database application performance and obviates the need for SSD administration or frequent in-depth analysis.

FasFile RAID's combination of caching, cache management, and Block Pool technologies make it an easy way to improve database application performance. FasFile RAID is application independent and is compatible with UNIX, NT, Windows 2000, AIX , and Linux servers.

To Learn More

Visit the Seek Systems Web site at www.seeksystems.com or phone us at
(800) 790-7335 or (425) 806-7335.

Seek Systems, Inc.
11715 North Creek Parkway South
Suite 110
Bothell, WA 98011