Seek Systems, Inc.

A Seek Systems, Inc. Whitepaper: Technology Summary of FasFile Xcelerator

Summary

This whitepaper outlines the technology behind FasFile Xcelerator. Xcelerator delivers important cost and performance improvements to SCSI and RAID storage. While comparable to Solid State Disk (SSD) systems, Xcelerator solves many drawbacks of SSD, including administrative overhead and buy-in cost.

FasFile Xcelerator attaches directly to the SCSI bus, working with any standard SCSI or RAID storage system to accelerate the performance of applications running on one or several host computers. A combination of unique architectural and advanced algorithms, FasFile Xcelerator improves the performance applications, particularly database applications, at a fraction of the cost of conventional SSDs.

Dynamic Control of Cache

Developed by Seek Systems, FasFile Xcelerator is dynamically managed cache that supplements existing RAID or standard SCSI disk storage. Unlike conventional SSDs which require that an administrator specify data sets and move them to solid state memory, FasFile Xcelerator automatically adjusts to retain the most frequently used information in Adaptive Cache™.

Alternatives in Solid State Storage

Anyone suffering through I/O logjams has probably explored moving some data to a memory based storage subsystem. The performance gains possible through SSDs are well documented. Relational database applications in particular have made good use of SSD technology with temporary tables and transaction log files, increasing query speeds by more than 400 percent.[1] Unfortunately, the total cost of ownership of SSDs makes them impractical for many applications. In addition to the buy-cost and administrative burden, the data allocated to the SSD are fixed, wasting expensive memory on inactive data.

In contrast, FasFile Xcelerator maximizes memory usage by monitoring data patterns and storing the more active data in Adaptive Cache and storing less active data on disk. This reduces the amount of cache memory required for temporary tables and transaction log files and allows frequently accessed files to be stored in solid state memory.

CPU-based caching is another approach to solid state acceleration, but it has practical disadvantages. It consumes host CPU cycles in sequential search operations, which can degrade application performance. CPU-based caching is also expensive, often requiring proprietary hardware sold only by CPU vendors. In addition, CPU- based caching is limited by the physical capacity of the host platform. FasFile Xcelerator's Adaptive Cache can expand in increments of 128MB up to 1GB. FasFile Xcelerator's cache is also non-volatile, in contrast to CPU cache whose contents are obliterated if power is lost.

FasFile Xcelerator's algorithms and tuning characteristics lead to a much higher hit rate for accessing data from solid state memory, improving the performance of all I/O intensive applications.

Inside FasFile Xcelerator

The performance of FasFile Xcelerator can be traced to its advanced memory management processes. Since data in solid state memory can be retrieved as much as 20 times faster than data on magnetic disk, it is critical that active data stays in memory and inactive data be moved to magnetic disk. To accomplish this, FasFile Xcelerator uses a combination of *Least Recently Used* and *Least Frequently Used* algorithms to maximize performance combined with the division of memory into *Normal Space* and *Protected Space*.

Normal Space: Least Recently Used Data

Similar to the operation of CPU-based cache, FasFile RAID Normal Space uses a *Least Recently Used* algorithm. As memory is needed for reads and writes it replaces the least recently used data in the Normal Space with this new data. The operation can be compared to a stack of cafeteria trays, in which new trays are added to the top and old ones removed from the bottom.

The Least Recently Used approach is effective for simple buffering of I/O. In interactive applications, the Least Recently Used algorithm leads to a problem called "Cache Pollution." Cache Pollution occurs when a high volume of reads or writes come through with data that is only to be used once or a few times. This data replaces data in cache that is being used repeatedly by the host – data that should reside in solid state memory.

Protected Space: Least Frequently Used Data

To avoid Cache Pollution, FasFile Xcelerator uses Protected Space: cache memory driven by a *Least Frequently Used* caching algorithm. FasFile Xcelerator tracks how often each data block is being accessed and moves the most popular data from Normal Space into Protected Space where it cannot be overwritten by recently used data.

Data in Protected Space is safe from any Cache Pollution and will remain there until FasFile Xcelerator detects other data that is more active. FasFile Xcelerator automatically divides cache between Protected Space and Normal Space. This allocation can be manually adjusted, although this is rarely necessary.

Other Acceleration Techniques

FasFile Xcelerator employs algorithms to improve the performance for disk read and disk write operations. For read operations, an algorithm called *Prefetch* is performed. When an application requests data that is not in Protected Space cache, FasFile Xcelerator reads the requested data from disk as well as the data that immediately follows. This is because there is a high probability that the application will request the subsequent data. One large read from disk is much more efficient than two small ones.

Prefetch may be adjusted for best performance. For OLTP applications a value of 2 is optimal (for every block of data requested an additional 2 Prefetch blocks are returned), while for sequential operations (e.g., backups) a Prefetch multiple of 4 or 5 is preferred.

For disk write operations, FasFile Xcelerator uses an algorithm called *Concatenation of Writes* to increase performance. Before FasFile Xcelerator writes information to disk, it pre-organizes the data according to where it resides on the physical disk. This batching activity results in fewer, larger, write operations.

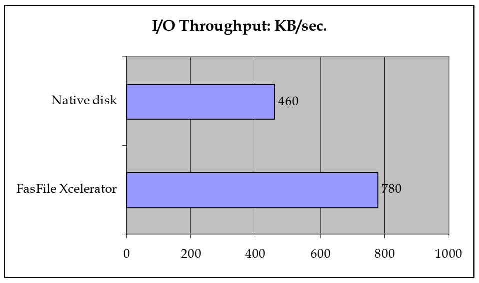
Actual Acceleration Results

The following are descriptions of the results from using FasFile Xcelerator in several different applications.

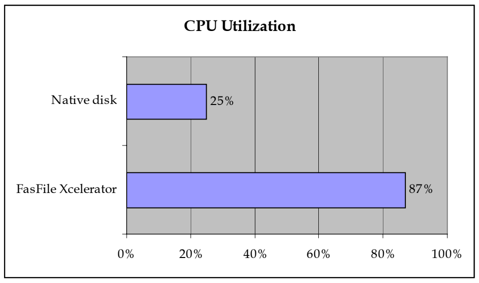
1. NFS I/O Enhancement

FasFile Xcelerator was employed because of the inefficiencies in NFS writes to disk in a data acquisition application. Local writes were fine, but when the same writes were performed over a FDDI cluster, performance fell dramatically. The causes were increased CPU overhead, an increase in inter-operation time (the time between SCSI commands from the host), and the breaking of single writes into many small NFS writes.

FasFile Xcelerator increased throughput on these writes by 67%. Primarily by concatenating the smaller NFS writes, FasFile Xcelerator removed the disk bottleneck.



Another telling statistic was CPU usage, which increased from 25% without the Xcelerator to 85% with the Xcelerator.

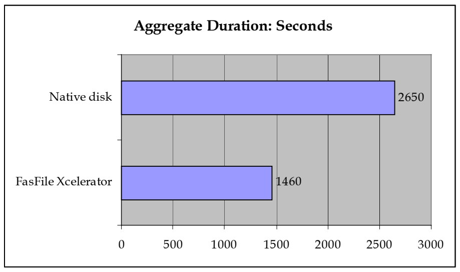


After removing the disk bottleneck, the CPU went back to work processing data.

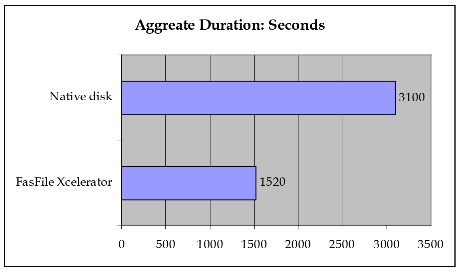
2. Relational Database Acceleration

Relational database applications enjoy dramatic improvements from FasFile Xcelerator. FasFile Xcelerator moves temporary tablespace, transaction log files and heavily used indexes and tablespaces into Protected Space. Temporary tablespace is where all intermediary processing occurs. For complex queries the activity to TempSpace is intensive, as data is written and updated repeatedly. Table sorts, updates, joins and similar commands are acutely taxing on TempSpace.

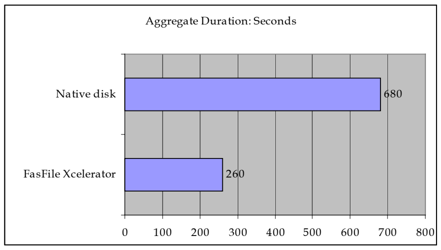
Moving TempSpace into FasFile Xcelerator's Protected Space led to a demonstrable performance improvement. The first benchmark was based on a Create/Sort of 100,000 records, with one primary key and five alternate keys. There were three simulated users performing this concurrently, with over 62,000 operations per user. FasFile Xcelerator increased performance by over 40%.



The second benchmark was based on a complex update operation involving three tables and over 72,000 operations. FasFile Xcelerator increased performance over 50%.



Transaction Log files can be I/O intensive, with many writes. Log files are where the database engine records transactions, so that it can Rollback tables to previous states, if needed. The database engine, determine how these files are used, but commands such as Rollback, Begin, Save, and Commit stress the Transaction Log files. The following table shows a 60% gain in performance for a database when the Transaction Log files were moved to FasFile Xcelerator.



Operational Considerations

FasFile Xcelerator uses fully non-volatile memory. Should it detect any problems with power it switches into Write Thru mode, ensuring that all data is written directly to disk. It remains in this state until power is restored, when it will switch to Write Back mode.

FasFile Xcelerator is transparent to the host and installation is as simple as adding a disk. Installation can usually be completed in under an hour. FasFile Xcelerator generates performance monitoring statistics that include SSD Hit Ratio, Average Inter-Operation Time, Read/Write Ratio, Average I/O Size, and total I/O. To optimize performance, the tunable parameters include Prefetch, Disk WriteBack Thresholds, Read and Write Cache Size Limits, and Protected Space.

More Performance in Less Memory

The advanced algorithms of FasFile Xcelerator ensure the active data is kept in solid state memory, where it may be accessed many times faster than if it resided on disk. In addition, I/O for inactive data on magnetic disk is accelerated by Prefetch of Reads and Concatenation of Writes. FasFile Xcelerator optimizes the performance of your disk storage, applications and host computers.

Glossary of Terms

Adaptive Cache- Seek Systems' solid state memory technology that improves the I/O performance of disk storage.

Cache Pollution- The process by which recently used, but infrequently used data crowds "hot" data out of cache. Cache Pollution can severely degrade performance of cache memory.

Concatenation of Writes- The process by which many small writes to disk are grouped together to produce a few large writes. This results in significant I/O improvement.

Least Frequently Used- An algorithm in which the least frequently used data in cache memory is replaced by more frequently used data. A method for ensuring that active data remains in solid state memory.

Least Recently Used- An algorithm in which the least recently used data in cache memory is replaced by the most recently used data. An effective method for buffering data, but it is prone to Cache Pollution.

*Normal Space***-** The portion of FasFile Xcelerator memory that uses the Least Recently Used algorithm. It acts as a buffer for all reads and writes to disk.

Prefetch- The process by which more data is read from the disk than is actually requested. It works on the principle that there is a high probability this data will be accessed next, and that single large reads are more efficient than multiple small reads.

Protected Space- The portion of FasFile Xcelerator cache used by the Least Frequently Used algorithm. It acts as semi-permanent SSD storage for active data.

Solid State Disk (SSD)- High capacity, relative to standard cache, solid state memory that functions as a disk drive would. SSD is usually attached to a host via SCSI.

Write Back- A mode in which a disk write is signaled as complete, even though data has yet to be written to disk. It provides the greatest performance but requires non-volatile memory to ensure that data is written to disk.

Write Thru- A mode in which the write to disk must be completed before the write is acknowledged as complete.

Footnotes

1 For information on Relational Databases and how they manage different tablespaces and queries, SEEK suggests number of books by C.J. Date, including An Introduction to Database Systems (Addison-Wesley, 1995). For information on UNIX file systems and performance monitoring tools such as IOSTAT, VMSTAT, and SAR, see UNIX System Administration Handbook, by Evi Nemeth, Garth Snyder, Scott Seebass, and Trent R. Hein (Prentice Hall PTR, 1995).