Redpaper

Barry Whyte

IBM SAN Volume Controller 4.2.1 Cache Partitioning

This IBM® Redpaper discusses the cache partitioning feature added to the IBM SAN Volume Controller (SVC) Version 4.2.1.

In this paper, we briefly describe the functions of a read and write cache, including an explanation of the terminology used.

This paper also discusses the benefits of partitioning cache resources and concludes with an explanation of how it is implemented in SVC Version 4.2.1.

Background

The storage controller or appliance-based read and write caches are primarily used to optimize data access to and from magnetic disk drives. Technology advancements in general computing have given us the ability to process very large quantities of data in a relatively short period of time.

However, the sustainable read and write data rates from magnetic disk drives are not scaled at the same rate. The standard 15 K RPM enterprise class disk drive is capable of around 250 I/O operations per second (IO/s), and around 100 MBps.

Memory-based read and write caches are typically used to speed up I/O requests. Read caches are useful when the same data is reread often, or sequential workloads are detected. Write caching is in use all the time, when it is available. Data is written to cache and written out to physical disk.

A cache usually consists of some physical memory capacity and a set of cache algorithms that are used to manage the contents of the cache. Due to memory cost and constraints, cache sizes are usually limited and require complex algorithms to ensure that the most useful data remains in cache. The job of the cache algorithms is to decide what is useful in the context of its current workload, and to decide what to remove or insert into cache space.

The most efficient cache algorithm, often referred to as Belady's minimum, discards data that is not needed until later. However, this type of predictive algorithm coding is virtually impossible.

Most storage cache algorithms discard the oldest data in cache. However, the method we use to determine what is old data changes:

- **Least Recently Used (LRU)**

  Old data that has not been accessed recently is called LRU data.

- **Least Frequently Used (LFU)**

  Old data that is not accessed frequently. The LFU maintains a count of how frequently data elements are accessed.

- **Adaptive Replacement Caching (ARC)**

  LRU and LFU lists are maintained. If data is accessed more than once, it moves from the LRU list to the LFU. Each list is maintained independently.

There are other additional variations on these algorithms. For example, certain algorithms maintain information about what was recently in the cache. The actual data is discarded, but the metadata describing the element is maintained. ARC-based solutions promote data from the LRU to ARC, even though it has recently dropped off the LRU list.

Primary cache benefit

The primary benefit of a storage cache is to improve I/O response time. Reads and writes to a magnetic disk drive suffer from both seek and latency time at the drive level. This can result in anything from one to 10 ms of response time (for an enterprise class disk).

Without a cache, today's operating system applications that are performing many thousands of concurrent I/O accesses (and each is serialized by the disk drives) have an unacceptable cumulative increase in response time.

Today's storage controllers and virtualization appliances have various methods of spreading workload and utilizing disk drives concurrently. Their primary performance enhancement comes from their cache.

Cache terminology

The following terminology is used in the subsequent sections:

Cache	Cache is a combination of the actual physical cache medium and the algorithms that are managing the data within the cache.
Demote-Ready	A track that is in the demote-ready list is either old read cache data, or write cache data that is successfully destaged to disk.
Destage	The act of removing modified data from the cache and writing it to disk
Modified Data	Write data in cache that is not destaged to disk
Non-owner Node	The non-preferred node for a given VDisk
Owner Node	The preferred node for a given VDisk
Partner Node	The other node in a SVC IO Group

Page	The unit of data held in the cache. In the SVC, this is 4 KB. The data in the cache is managed at the page level. A page belongs to one track.
Track	The unit of locking and destage granularity in the cache. In the SVC, a track is 32 KB in size (eight pages). (A track might only be partially populated with valid pages.)

SVC cache algorithm

The SVC cache is based around the LRU algorithm. It is useful to understand the life cycle of data in an LRU-based cache.

Write life cycle of data in the SVC

When a write is issued to the SVC, it passes through the upper layers in the software stack and into the cache. One of the primary purposes of the cache is to buffer I/O and allow a quick completion to return to the host issuing the I/O. This is many times quicker than writing to physical disk.

Before the cache returns completion to the host, the write must mirror the partner node. This is done for availability reasons. Write data held in cache is not destaged to disk; therefore if only one copy of the data is kept and you have a power failure, you risk losing data.

After the I/O is written to the partner node, and acknowledged back to the owner node, the I/O is completed back to the host[1].

The LRU algorithm places a pointer to this data at the top of the LRU. As subsequent I/O is requested, they are placed at the top of the LRU list. Over time, our initial write moves down the list and eventually reaches the bottom.

If a subsequent write is performed to a track that is held in the LRU list, the new write data is merged into the track, and the track is moved back to the top of the LRU list.

When certain conditions are met, the cache decides that it needs to free a certain amount of data. This data is taken from the bottom of the LRU list, and in the case of write data, is committed to disk. This destage operation is only performed by the owner node.

When the owner node receives confirmation that the write to disk is successful, the control blocks associated with the track are modified to mark that the track now contains read cache data, and is added to the demote-ready list. Subsequent reads of recently written data are returned from cache. The owner node notifies the partner node that the write data is complete, and the partner node discards the data. The data is discarded on the non-owner (partner) node, because reads are not expected to occur to non-owner nodes.

Read life cycle in the SVC

When a read is issued to the SVC, it passes through the upper layers in the software stack and the cache checks to see if the required data resides in cache.

[1] In general, the write is sent to the owner node (preferred node). However, SVC is Active/Active and accepts I/O requests that arrive at the non-preferred node. This requires additional processing and is avoided if possible.

If a read is made to a track already in the cache (and that track is populated with enough data to satisfy the read), the read is completed instantly, and the track is moved to the top of the demote-ready list.

Note: If the track is not fully populated, or the cache does not contain all the data required to complete the I/O, the data is read from disk, and the tracks associated with the read are placed at the top of the demote-ready list.

If a read is satisfied with data held in the demote-ready list, the control blocks associated with the track is modified to denote that the data is at the top of the LRU list. Any reference to the demote-ready list is removed.

There is a distinction made between actual host read I/O requests, and the speculative nature of old writes that are turned into read cache data. It is for this reason that the demote-ready list is emptied first (before the LRU list) when the cache algorithms decide they need more free space.

Cache algorithm life cycle

The cache algorithms attempt to maintain a steady state of optimal population that is not too full, and not too empty. To achieve this, the cache maintains a count of how much data it contains, and how this relates to the available capacity.

As the cache reaches a predefined high capacity threshold level it starts to free space at a rate known as **trickle**. Data is removed from the bottom of the LRU at a slow rate. If the data is a write, it is destaged. If the data is a read, it is discarded.

Destage operations are therefore the limiting factor in how quickly the cache is emptied, because writes are at the mercy of the latency of the actual disk writes. In the case of the SVC this is not as bad as it sounds, as the disks in this case are controller LUNs. Almost every controller supported by the SVC has some form of internal cache. When the I/O rate being submitted by the SVC to a controller is within acceptable limits for that controller, you expect writes to complete within a few milliseconds.

However, problems can arise because the SVC can generally sustain much greater data rates than most storage controllers can sustain. This includes large enterprise controllers with very large caches.

SVC Version 4.2.0 added additional monitoring of the response time being measured for destage operations. This response time is used to ramp up, or down, the number of concurrent destage operations the SVC node submits. This allows the SVC to dynamically match the characteristics of the environment that it is deployed. Up to 1024 destage operations can be submitted in each batch, and it is this batch that is monitored and dynamically adjusted.

Note: The SVC back-end layer controls exactly how many concurrent I/O requests are maintained for a given Managed Disk (MDisk). This queue setting varies depending on the controller type, number of SVC nodes, and number of managed disks presented by the controller. For more details see:

SAN Volume Controller: Best Practices and Performance Guidelines, SG24-7521

If the SVC incoming I/O rate continues, and the trickle of data from the LRU does not reduce the cache below the high capacity threshold, trickle continues. If the trickle rate is not keeping the cache usage in equilibrium, and the cache usage continues to grow, a second high capacity threshold is reached.

This will result in two simultaneous operations:

- Any data in the demote-ready list is discarded. This is done in batches of 1024 tracks of data. However, because this is a discard operation, it does not suffer from any latency issues and a large amount of data is discarded quickly. This might drop the cache usage below both high capacity thresholds.
- The LRU list begins to drop entries off the bottom at a rate much faster than trickle.

The combination of these two operations usually results in the cache usage reaching an equilibrium, and the cache maintains itself between the first and second high usage thresholds. The incoming I/O rate continues until the cache reaches the third and final threshold, and the destage rate increases to reach its maximum.

Figure 1 shows the threshold actions.

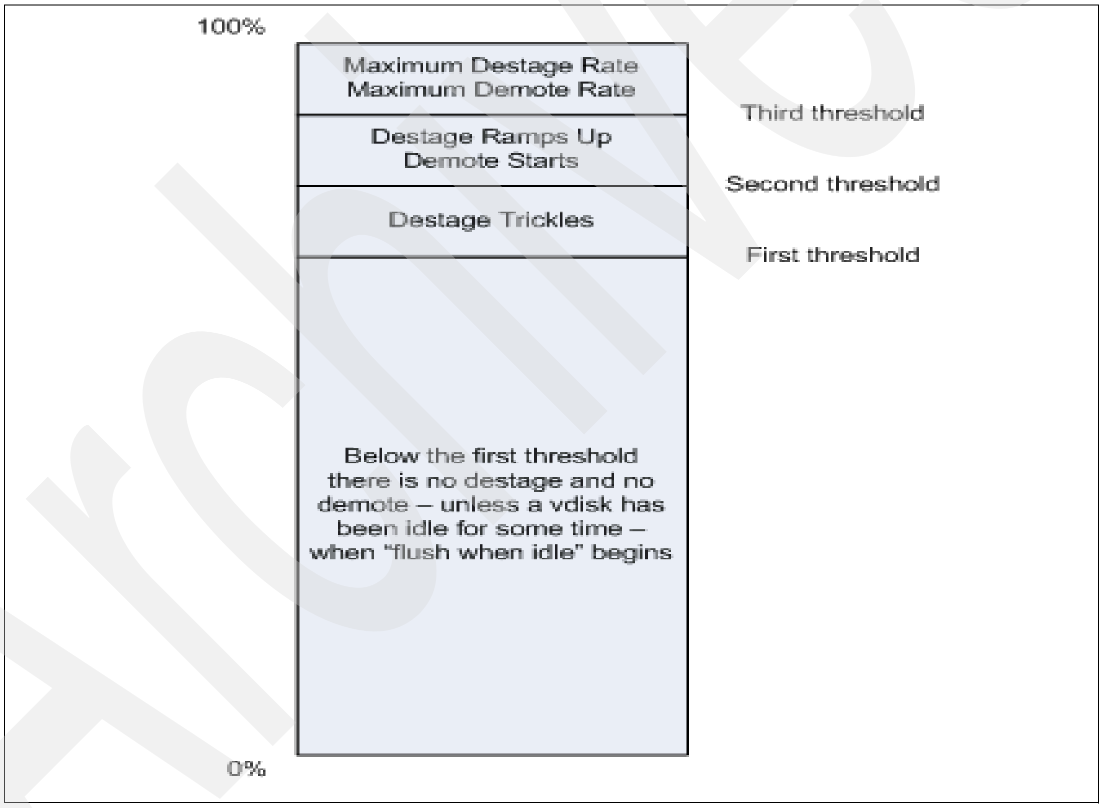
100%
Maximum Destage Rate
Maximum Demote Rate
Third threshold
Destage Ramps Up
Demote Starts
Second threshold
Destage Trickles
First threshold
Below the first threshold there is no destage and no demote – unless a vdisk has been idle for some time – when "flush when idle" begins
0%

Figure 1 Thresholds

Note: The destage rate, and number of concurrent destaged tracks are two different attributes. The rate determines how long to wait between each batch. The number of concurrent tracks determines how many elements to build into a batch.

Note: If the back-end disk controllers cannot cope with the amount of data being sent from the SVC, the cache might reach 100% full. This results in a one-in, one-out situation where the host I/O is only serviced as quickly as the back-end controllers can complete the I/O, essentially negating the benefits of the SVC cache. Too much I/O is being driven from the host for the environment in which the SVC is deployed.

You can avoid these situations by using the IBM Disk Magic™ tool for determining the amount of back-end storage required. For regular monitoring of SVC statistics, you can use the IBM TotalStorage® Productivity Center (TPC for disk) tool.

Cache partitioning

In the previous section we describe a situation where the cache reaches 100% full when more I/O is driven to an SVC node than can be sustained by the back-end storage.

A shared resource, such as a global cache resource, can suffer from this problem even if only one storage controller is struggling to cope with the I/O load.

Cache partitioning provides a mechanism to protect a shared cache from not only seriously overloaded controllers, but also misbehaving controllers.

SVC cache partitioning details

SVC Version 4.2.1 first introduced cache partitioning to the SVC code base. This decision was made to provide flexible partitioning, rather than hard coding a specific number of partitions. This flexibility is provided on a Managed Disk Group (MDG) boundary. That is, the cache automatically partitions the available resources on a MDG basis.

Most users create a single MDG from the Logical Unit Numbers (LUN)s provided by a single disk controller, or a subset of a controller/collection of the same controllers, based on the characteristics of the LUNs themselves. For example, RAID-5 compared to RAID-10, 10K RPM compared to15K RPM, and so on.

The overall strategy is provided to protect the individual controller from overloading or faults. If many controllers (or in this case, MDGs) are overloaded then the overall cache can still suffer.

Table 1 shows the upper limit of **write cache data** that any one partition, or MDG, can occupy.

Table 1 Upper limit of write cache data

Number of MDGs	Upper limit
1	100%
2	66%
3	40%
4	30%
5 or more	25%

You can think of the rule as, no single partition occupies more than its upper limit of cache capacity with write data.

Upper limits are the point at which the SVC cache starts to limit incoming I/O rates for Virtual Disks (VDisks) created from the MDG.

If a particular partition reaches this upper limit, the result is the same as a global cache resource that is full. That is, the host writes are serviced on a one-out one-in basis - as the cache destages writes to the back-end disks. However, only writes targeted at the full partition are limited, all I/O destined for other (non-limited) MDGs continue normally.

Read I/O requests for the limited partition also continue normally. However, since the SVC is destaging write data at a rate that is obviously greater than the controller can actually sustain (otherwise, the partition would not reach the upper limit), reads are serviced equally as slow.

Note: Due to the relationship between partitions and MDGs, you must be careful when creating large numbers of MDGs from a single controller. This is especially true when the controller is a low or mid-range controller.

Enterprise controllers are likely to have some form of internal cache partitioning and are unlikely to suffer from overload in the same manner as entry or mid-range.

Changes to SVC cache algorithm for partitioning

Previously, we explored the general life cycle of an I/O through the SVC cache. The algorithm is further enhanced to cater to partition limiting, and multiple streams of destage.

Instead of the destage algorithm acting purely on total cache utilization, it now has to take into account individual cache partition utilization.

The same rules apply for the overall cache utilization, that is; if the cache reaches the first, second, or third capacity thresholds, the algorithm destages and demotes data as it did before. However, now the algorithm must also monitor each partition. If any partition reaches a high capacity threshold (defined as a high percentage of its relative allocation), destage begins. Only write data in any partition over its respective threshold limit is destaged.

It is easier to understand these concepts using the examples that are in the next sections.

Destage example

Assume, we have four MDGs. Therefore, each partition has an upper limit of 30%. Assume that only two of these partitions are active. At most, we have 60% of the cache full of write data for these partitions. For this example, we also assume that we have no read data, only write data in the cache.

Partition 1 is performing very little I/O, and is only 20% full (20% of its 30% limit - so a very small percentage of the overall cache resource).

Partition 2 is being written to heavily. When it reaches the defined high capacity threshold within its 30% limit, write data begins to destage. The cache itself is below any of its overall thresholds. However, we are destaging data for the second partition.

Figure 2 on page 8 shows what happens to partition 2.

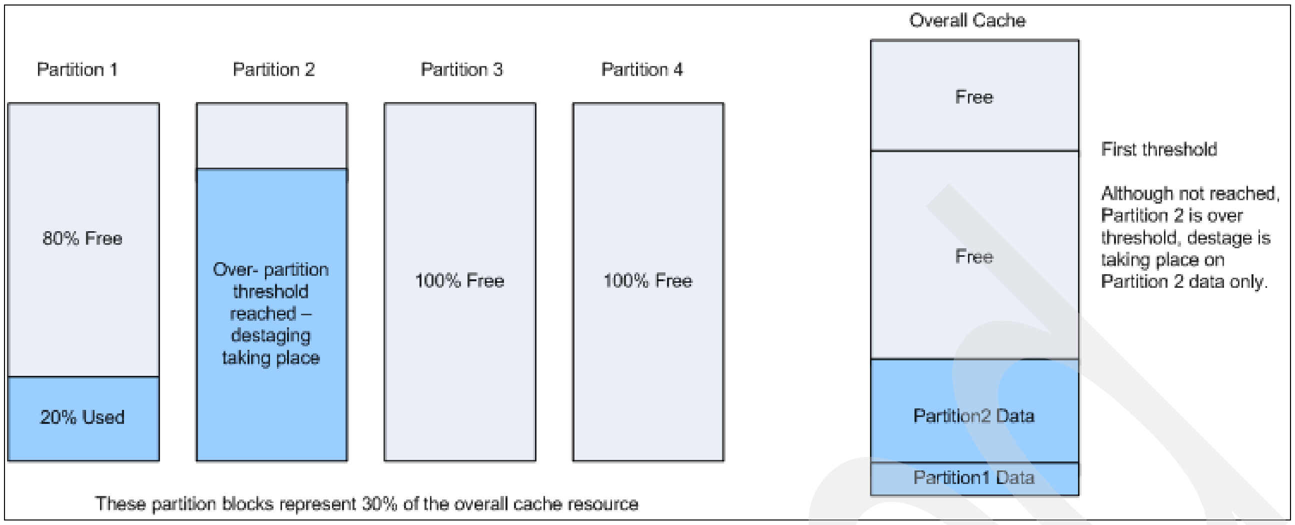
Overall Cache
Partition 1
Partition 2
Partition 3
Partition 4
80% Free
20% Used
Over- partition threshold reached – destaging taking place
100% Free
100% Free
Free
Free
Partition2 Data
Partition1 Data
First threshold
Although not reached, Partition 2 is over threshold, destage is taking place on Partition 2 data only.
These partition blocks represent 30% of the overall cache resource

Figure 2 Partition 2

We see that the controller servicing partition 2 is struggling, cannot cope with the write data that is being destaged, and the partition is 100% full, and it is occupying 30% of the available cache resource. In this case, incoming write data is slowed down to the same rate as the controller itself is completing writes.

Partition 3 begins to perform heavy I/O, goes above its high capacity threshold limit, and starts destaging. This controller, however, is capable of handling the I/O being sent to it, and therefore, the partition stays around its threshold level. The overall cache is still under threshold, so only partitions 2 and 3 are destaging, partition 2 is being limited, partition 3 is destaging well within its capabilities.

Figure 3 shows what happens to partition 3.

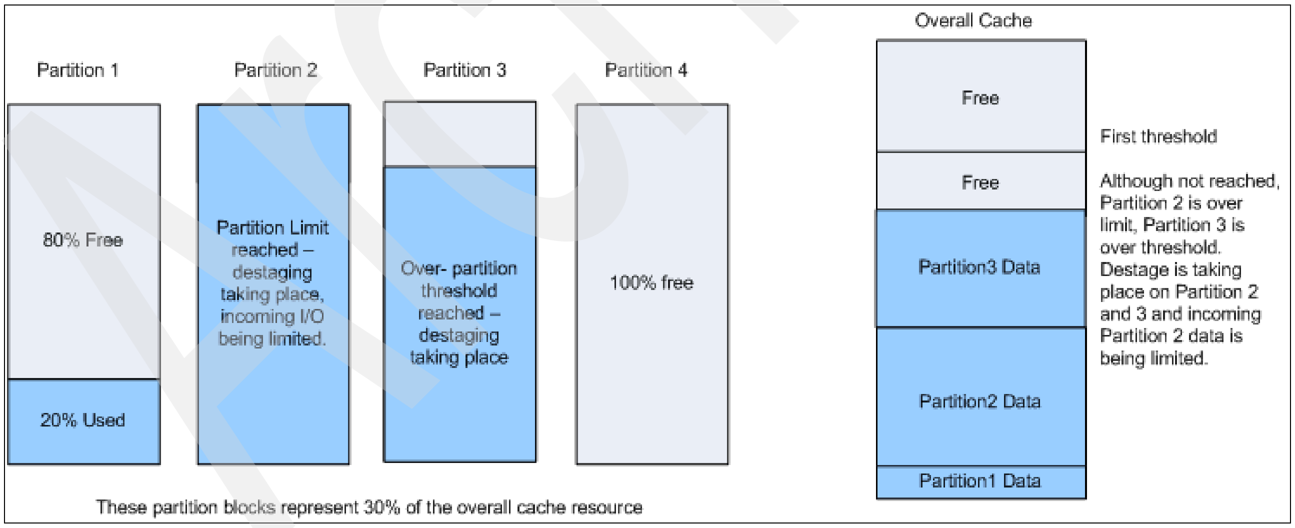
Overall Cache
Partition 1
Partition 2
Partition 3
Partition 4
80% Free
20% Used
Partition Limit reached – destaging taking place, incoming I/O being limited.
Over- partition threshold reached – destaging taking place
100% free
Free
Free
Partition3 Data
Partition2 Data
Partition1 Data
First threshold
Although not reached, Partition 2 is over limit, Partition 3 is over threshold. Destage is taking place on Partition 2 and 3 and incoming Partition 2 data is being limited.
These partition blocks represent 30% of the overall cache resource

Figure 3 Partition 3

Partition 4 begins to perform heavy I/O, when it reaches just over a third of its partition limit, and the overall cache is now over its first threshold limit. Destage begins for all partitions that have write data - in this case, all four partitions. When the cache returns under the first threshold, only the partitions that are over their individual threshold allocation limits continue to destage.

Figure 4 on page 9 shows what happens to partition 4.

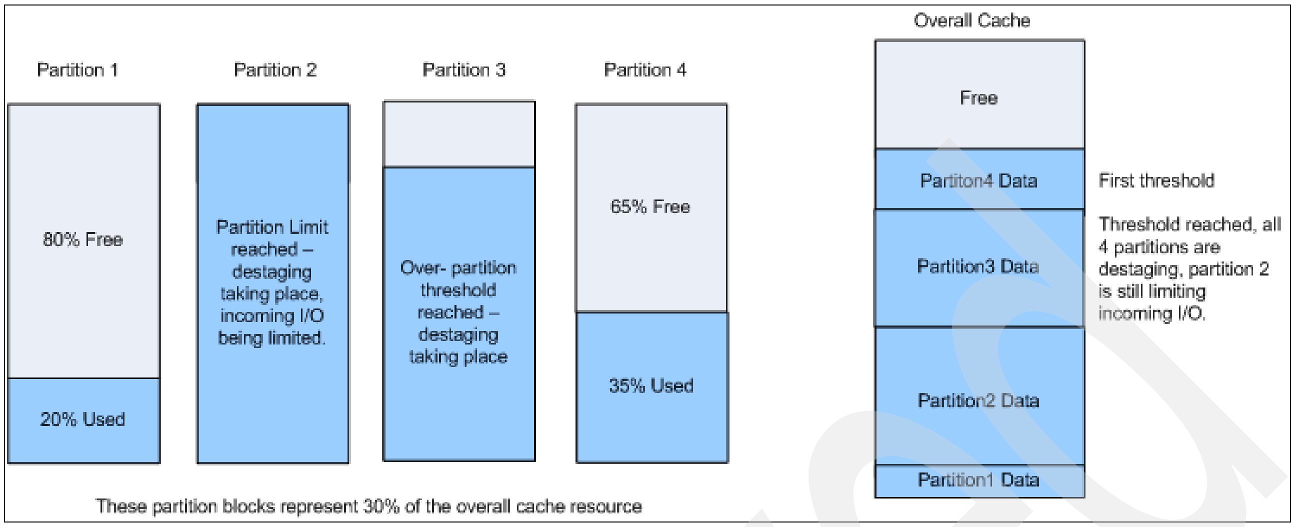
Partition 1
80% Free
20% Used
Partition 2
Partition Limit reached – destaging taking place, incoming I/O being limited.
Partition 3
Over- partition threshold reached – destaging taking place
Partition 4
65% Free
35% Used
These partition blocks represent 30% of the overall cache resource
Overall Cache
Free
Partiton4 Data
Partition3 Data
Partition2 Data
Partition1 Data
First threshold
Threshold reached, all 4 partitions are destaging, partition 2 is still limiting incoming I/O.

Figure 4 Partition 4

You can see the controller that struggles is limited, while the other partitions (controllers) are unaffected.

In general, this model of cache partitioning can protect up to three MDGs (controllers) suffering a serious problem. Partitioning, however, is better than a global cache model, where a single MDG might end up occupying the entire cache resource.

In lab testing, we see that in the worst case, degraded, cache disabled RAID-5 test case, the partitioning improves overall I/O Group throughput by over 200%.

Am I still getting efficient cache usage

The most common question that comes up since the introduction of cache partitioning relates to the upper limits. If you are only performing I/O to one or two MDGs, then is the cache space being reduced?

The main thing to keep in mind is that the partitioning is only limited on write I/Os.

In general, a 70/30 or 50/50 ratio of read to write operations are observed. Of course, there are applications, or workloads that perform 100% writes, however write cache hits are much less of a benefit than read cache hits. A write always hits the cache. If modified data already resides in the cache, it is overwritten, this might save a single destage operation. However, read cache hits provide a much more noticeable benefit, saving seek and latency time at the disk layer.

The overall LRU nature of the cache is still maintained, so while it is true that write data might be destaged earlier than in a non-partitioned scheme, it is still the oldest write data in the cache that is destaged. It is also only partitions that are over their allocation limit that are destaged.

In all benchmarking tests performed, even with single active MDGs, good path SVC I/O group throughput remains the same as it was before the introduction of cache partitioning.

Conclusion

This Redpaper discusses the cache partitioning feature added to the IBM SVC Version 4.2.1, the reason for its introduction, and the details of how this function is implemented. The reader should now be aware of the benefits of partitioning cache resources, and the details of how this is implemented in SVC Version 4.2.1.

The team that wrote this IBM Redpaper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Barry Whyte is a Master Inventor working in the Systems and Technology Group (STG) based in IBM Hursley, UK. Barry primarily works on the IBM SAN Volume Controller virtualization appliance. Barry graduated from the University of Glasgow in 1996 with a B.Sc (Hons) in Computing Science. In his 10 years at IBM he has worked on the successful Serial Storage Architecture (SSA) range of products and the follow on Fibre Channel products used in the IBM DS8000™ range. Barry joined the SVC development team soon after its inception, and has held many positions before taking on his current role as SVC performance architect.

Thanks to the following for their help and comments during review of this Redpaper:

Lee Sanders
Bill Scales
IBM Hursley

Notices

This document REDP-4426-00 was created or updated on April 30, 2008.

Send us your comments in one of the following ways:

- Use the online **Contact us** review IBM Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Trademarks