An overview of VMware Virtual SAN caching algorithms

October 2015

TECHNICAL WHITE PAPER

vmware®

Purpose and Overview

This document was developed to provide additional insight and information into the operations of VMware Virtual SAN™ (VSAN) caching algorithms. It will explain how VSAN intelligently leverages flash, memory and traditional magnetic disk. VSAN combines the capacity, performance, and endurance of each class of storage.

Introduction

VMware Virtual SAN is a new hypervisor-converged, software-defined storage platform that is fully integrated with VMware vSphere®. Virtual SAN aggregates locally attached storage of hosts that are members of a vSphere cluster, to create a distributed shared storage solution. Virtual SAN can leverage either flash or magnetic drives to provide capacity and a persistent storage layer. This delivers enterprise availability and performance. The distributed datastore of VirtualSAN is a distributed object store that leverages the vSphere Storage Policy Based Management (SPBM) framework to deliver unique availability and performance requirements. The purpose of this document is to provide an explanation of how the caching algorithms are different for read and write caching as well as hybrid and all flash configurations.

VMware VSAN In this document, we use the acronym VSAN to refer to VMware's Virtual SAN product. We refer to flash devices as SSDs, even though they may also be PCIe, NVMe, and DIMM based flash devices. We refer to magnetic (spinning) disks as HDDs.

Read Caching

Read Caching in VSAN exists to separate performance from capacity and deliver great performance and capacity density at a competitive cost.

Read caching is used only in the case of *hybrid* VSAN clusters, where each disk group consists of one SSD and one or more HDDs. VSAN uses the SSD device as the "performance tier" of each disk group. Part of the SSD is used as the Read Cache (RC) of the corresponding disk group. The purpose is to serve the highest possible ratio of read operations from data staged in the RC and minimize the portion of read operations that are served by the HDDs. The reason is obvious – SSDs can serve a large number of IOPS (thousands or tens of thousands) even for random workloads at lower latencies than HDDs. Moreover, the performance 'cost' of operations ($/IOPS) on SSDs is a fraction of its equivalent on HDDs. HDDs serves a very small number of IOPS (order of magnitude 100 IOPS for random workloads). As such, SSDs are much more cost effective for delivering performance, especially for virtualized workloads, which in aggregate are almost always random.
For example, one can buy a 100GB Intel S3700 SSD for $200, which may serve up to 45K IOPS. That is $0.004 per IOPS. On the other hand, a 1TB Seagate Constellation drive, which also costs $200, can only deliver 100 IOPS, yielding $2 per IOPS.

DRIVE TYPE	$ PER IOP	$ PER GB
NL-SAS	$2	$.2
SSD	$.0004	$2

In hybrid disk groups, most of the SSD is used as RC. By default, that is 70% of the device's capacity. This is a configurable parameter, but we do not recommend users to fiddle with it on their own, unless VMware

support has recommended a different setting based on an analysis of the customer's use case and workloads.

The RC is organized in terms of 'cache lines'. A cache line is currently 1MB in size and it is the unit of data management in RC. Data is fetched into the RC and evicted, when needed, at the granularity of a cache line.

In addition to the SSD-resident RC, VSAN also maintains a small in-memory (RAM) read cache, which holds the few most recently accessed cache lines from the RC. The in-memory cache is dynamically sized based on the available memory in the system.

VSAN maintains in-memory metadata that track the state of the RC (both on-SSD and in-memory), including the logical addresses of cache lines, valid and invalid regions in each cache line, aging information, etc. These data structures are designed so that they are compressed and make a very efficient use of memory space while not imposing a substantial CPU overhead on regular operations. There is never a need to swap RC metadata in or out persistent storage. This is one area where VMware holds important IP.

Currently, the contents of the RC are not tracked across power-cycle operations of the host. If power is lost and recovered, then the RC is re-populated (warmed) from scratch. So, essentially RC is used as a fast storage tier, but we do not depend on its persistence across power cycles. The rational behind this approach is to avoid any overheads on the common data path that would have been required if we persisted RC metadata every time the contents of RC were modified, such as cache line fetching and eviction, but more importantly every time a sub-region of a cache line gets invalidated because of a write operation.

Let us see how the actual caching process works:

- When a read operation reaches the VSAN layer that performs I/O to the devices, the RC logic kicks in. It uses the in-memory data structures to detect whether the logical block(s) referenced by the read operation are fully or partially in the RC already. Note that a single read operation may reference a range of blocks that span more than one cache line.

- If some or all the data is not in the RC, a 1MB buffer is allocated for each missing cache line.

- Also, space is allocated in the RC in the SSD for the new cache line(s) by evicting existing cache lines. Cache line eviction and replacement is managed using a variant of the *Adaptive Replacement Cache (ARC)* algorithm, which has been licensed from IBM.

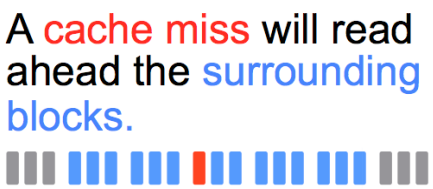
- Each missing cache line is read from the HDD as multiple 64KB chunks. The reason behind this fragmentation is to reduce the queuing penalty incurred for other operations sent to the HDD just after the RC miss.

- In the common case, a read operation references only a fraction of a 1MB cache line. VSAN issues the reads for the 64KB chunks that contain the referenced data first.

- As soon as the data requested from the original read operation are read from the HDD, the original operation is immediately completed upstream. The cache line in the RC is populated asynchronously.

- Once the entire cache line has been read from the HDD and written in the RC, the 1MB in-memory buffer is added to the in-memory RC.

The reason we populate a 1MB-wide cache line in RC when data is accessed for the first time is that workloads typically exhibit locality of reference. That is, if some data is accessed, then the same data or data in its vicinity in terms of block address may be accessed afterwards. The choice of 1MB cache lines (as well as the 64KB chunk size for the operations that populate the cache lines) has been determined experimentally and is based on real-world workloads and traces. The 1MB cache line size is a good tradeoff between achieving effective caching and keeping the size of in-memory metadata reasonable. The detailed analysis is proprietary VMware IP.



VSAN tracks every single write operation in a logical address range that overlaps with some cache line(s) in the RC. When an overlap is detected, a number of steps are performed in a transactional way:

- A record for the operation in persisted on a Transaction Log VSAN maintains in the SSD;
- The write operation's payload is persisted in the Write Buffer area of the SSD;
- The RC metadata is modified to reflect the parts of the cache line(s) that have been invalidated by this write operation.

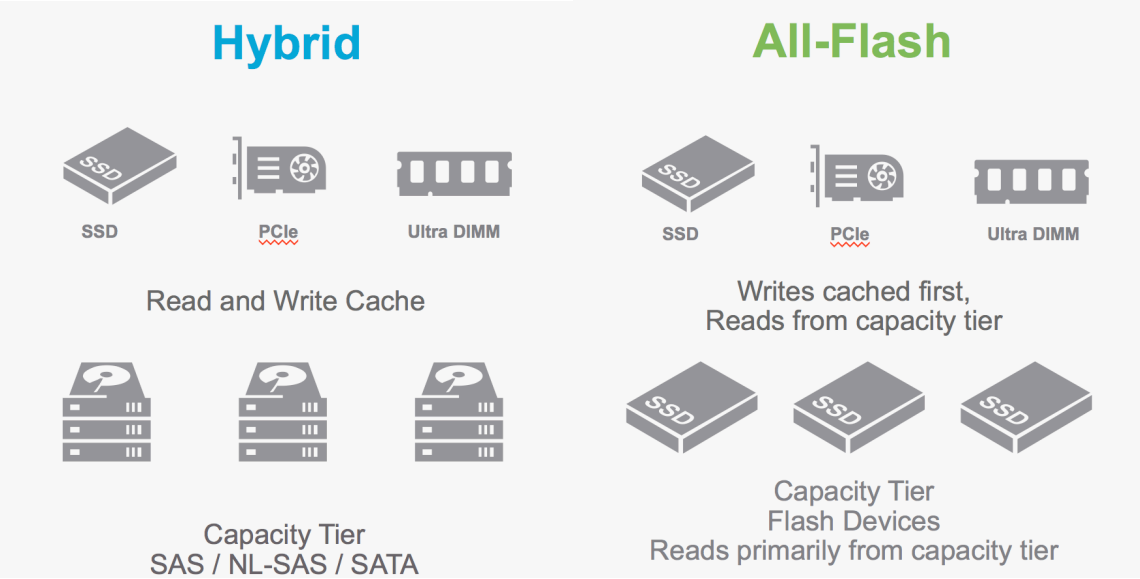
The metadata keep track of the location of the data in the write buffer. The write operation is completed upstream after the transaction is committed successfully.

When a subsequent read operation arrives and references invalidated regions of some cache line(s), the invalidated regions are read from the write buffer utilizing the RC metadata. At a later point, the written data may be de-staged from the SSD to the HDD. At that point and in a transactional way, VSAN uses that data to also replace the contents of the corresponding regions in the affected cache line(s). The regions are tagged as valid again. This process prevents future reads to that portion of the cache line(s) from having to fetch the data from the HDD.

Write Caching

VSAN utilizes write-back caching to achieve different purposes in the case of hybrid and all-flash configurations respectively. However, the underlying algorithm is very similar and in this description we will cover it in a generic fashion.

Why write-back caching? In *hybrid configurations*, this is done entirely for performance reasons. As we discussed, the aggregate storage workloads seen in virtualized infrastructures are almost always random, thanks to the statistical multiplexing of the many VMs and applications that share the infrastructure. We know that HDDs perform badly with random workloads where IOPS and latency are the key performance metrics. So, sending the write part of the workload directly to spinning disks is a very bad idea. On the other hand, magnetic disks exhibit decent performance for sequential workloads where throughput is the key metric. Modern HDDs may exhibit sequential-like behavior and performance even when the workload is not perfectly sequential – 'proximal I/O' suffices.



In hybrid disk groups, VSAN uses the 'Write Buffer' (WB) section of the SSD (by default, the remaining 30% of the SSD's capacity) as a write-back buffer where all the write operations are staged. The key objective is to de-stage written data (not individual write operations) in a way that will create a benign near-sequential (proximal) write workload for the HDDs that form the capacity tier of the disk group.

In all-flash disk groups, VSAN utilizes the tier-1 SSD entirely as a write-back buffer (100% of the device's capacity up to a maximum of 600GB). The purpose of the WB is very different in this case. It is used to absorb the highest rate of write operations in a high-endurance device and allow for only a trickle of data to be written to the capacity flash tier (tier 2). This approach allows for the use of low endurance and thus lower cost (and larger capacity) SSDs for the capacity tier, making for very cost-compelling all-flash configurations.

Nevertheless, capacity tier SSDs are capable of serving very large numbers of read IOPS. Thus, there is no read caching done in the tier-1 SSD, other than when the most recent data referenced by a read operation happen to be still in the WB. For typical workloads, the big majority of reads are served directly from the capacity tier.

In either case (hybrid or all-flash), every write operation is handled through a transactional process (parts of the process were discussed in the RC section):

- A record for the operation is persisted in the Transaction Log in the SSD.
- The data (payload) of the operation is persisted in the WB.
- In-memory tables are updated to reflect the new data written and their logical address space (so the latest data can be tracked) as well as the physical location of their final destination in the capacity tier.

The write operation completes upstream after the above transaction has committed successfully.

In the common case (under typical steady-state workload), the log records of multiple write operations are coalesced before they are persisted in the log to significantly reduce the amount of metadata write operations performed to the SSD. By definition, the Log is a circular buffer, which is written and freed in a sequential fashion and thus write amplification is avoided (good for device endurance).

Block allocation in the WB region is performed in a round-robin fashion keeping wear leveling in mind. Even when a write operation is an overwrite of data that is already in the WB, VSAN never does an in-place re-write of an existing SSD page. Instead, a new block is allocated and metadata are updated to reflect that the old blocks are invalid. VSAN fills an entire SSD page, before it moves to the next one. Eventually, entire

pages are freed when all their data are invalid (it is very rare that VSAN may have to re-buffer data to allow for SSD pages to be freed). Also, because the device firmware does not have visibility into invalidated data, it sees no "holes" in pages. Thus, internal write leveling (by moving data around to fill holes in pages) is practically eliminated. This has a very positive impact on the overall endurance of a device. In general, VSAN's design has gone to great lengths to impose a workload that is very benign to the endurance of SSDs. As a result, we expect that the life expectancy of SSDs in VSAN may exceed the manufacturers' specifications, which are developed with more generic workloads in mind. For a discussion of Solid-State Disk (SSD) design, the concept of a 'page' and SSD endurance considerations, the reader is referenced to the literature. For example, there is a related Wikipedia article.

The WB region of an SSD is carved into 'N' logical buckets, which are used by the data de-staging algorithm. The most recent data is written in bucket N-1. The most stale or coldest data live in bucket 0. The allocation of data blocks to buckets is purely a logical operation reflected in metadata and it does not have anything to do with actual physical partitioning of SSD space. When VSAN needs to de-stage data due to resource constraints (more on this later), it starts from bucket #0, i.e., from the coldest data.

The de-staging process follows an 'elevator' algorithm which writes non-invalidated data to each of the capacity tier devices in increasing physical LBA order, one bucket at a time. For typical workloads, the data accumulated in a bucket are enough to create a near-sequential (proximal) workload for each HDD. The algorithm minimizes seek times and maximizes the throughput achieved from each HDD. Even in all-flash cases, the proximal workload results in better throughput from the capacity SSD.

The key difference between the process in hybrid and disk groups is in the number of buckets utilized. The details of this analysis are beyond the scope of this paper. As a principle, the number of buckets and the size of the WB are used as a tool to reduce the probability of 'warm' data to be de-staged in all-flash cases. On the other hand, with HDDs in the capacity tier, the de-staging process has to be much more aggressive. The reason is that due to the lower performance of HDDs, the algorithm must address resource pressure at a much earlier stage. In practice, warm data may remain in the WB for perpetuity in the all-flash case, but there is a higher probability that may be de-staged in hybrid configurations.

The key aspects of the de-staging process are two-fold

- The timing criteria that trigger the next round of the elevator algorithm;
- The number of buckets that need to be processed.

Both aspects are related to VSAN running out of resources for the WB. These resources include:

- Memory consumption by metadata structures;
- Log build up;
- Write buffer build up.

As soon as the utilization of any of these resources exceeds certain (internal) thresholds, the de-staging process is initiated. If the elevator algorithm cannot catch up with the incoming rate of write operations, then back-pressure is put on the I/O stream which may reach all the way back to the 'ingress' point to the storage, e.g., the VSCSI devices of the virtual machines. This 'congestion management' situation is perceived as higher latencies to the application. It is a meaningful response to the occasional workload spike. However, if it persists for longer period of time, it means that one's system is not sized appropriately for their workload demands. The sizing of their system should be re-visited using VMware sizing and capacity planning tools for VSAN.

October 12, 2015
The VSAN Engineering Team

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com

Copyright © 2010 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at http://www.vmware.com/go/patents. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.