Technology Assessment

AFA Performance Testing Framework

Eric Burgener

IDC OPINION

The storage performance requirements of 3rd Platform computing workloads demand more than what traditional hard disk drive (HDD)-based architectures can cost effectively deliver. As a result, flash-based storage is rapidly penetrating the datacenter. Flash operates very differently in terms of read/write performance than traditional HDDs, and this has major implications for how testing should be performed on flash-based arrays. There are several types of flash-based arrays where these considerations apply: all-flash arrays (AFAs), all-flash configurations of hybrid flash arrays (HFAs), and some more flash-optimized HFAs. Although early HFAs exhibited storage performance characteristics more similar to that of HDD-based arrays, these offerings are becoming increasingly flash optimized, narrowing the performance differences between them and AFAs. End users that are testing flash-based arrays for use in their production datacenters need to understand how flash-based arrays and flash-based testing is different. Findings include:

- **Preconditioning.** Flash-based arrays must be transitioned from a fresh-out-of-the-box (FOB) state to a steady state before test results can start to accurately reflect the performance the array will deliver in production. Preconditioning to transition flash-based arrays to steady state operation is a requirement in all testing.
- **Data stream/data set definition.** The workloads that are most often run on AFAs differ from classic workloads in significant ways. In the mixed virtual workloads where AFAs are most often deployed, both data streams and data sets exhibit a variety of read/write ratios, use a wide distribution of block sizes, are heavily skewed toward random I/O, have a high percentage of data that is reducible, and are characterized by I/O bands that drift over time. To understand how an AFA will perform on workloads that exhibit these characteristics, it must be tested with workloads that mimic them.
- **Read/write asymmetries.** While flash-based arrays deliver average latencies and throughput that are an order of magnitude faster than HDD, flash does not exhibit the same type of performance consistency. In fact, there can be wide variability between read/write latencies in flash-based arrays, depending on what is going on in the array at the time. Test administrators need to understand these characteristics, the behavior they produce at a system level, and use a test plan designed for flash, not HDD.
- **Scale of workload generation.** The largest HDD-based arrays are generally delivered on the order of tens of thousands of IOPS, but flash-based arrays can deliver hundreds of thousands to millions of IOPS. To exercise flash-based arrays up to and beyond their performance saturation points, testing equipment must be able to generation sufficient workload.

IN THIS STUDY

The objective of this study is to provide a framework for testing AFAs. Flash media performs very differently than HDD-based media, exhibiting very different behaviors with respect to read/write I/O. Test plans and tools designed for use with HDD-based storage systems do not apply to AFAs and will produce results that do not accurately reflect the capabilities of AFAs in production use.

In this document, we discuss the important considerations in developing and executing a performance testing regimen against AFAs in end-user-focused proof-of-concept testing (rather than vendor development testing). With a better understanding of these considerations, test administrators will be able to more effectively evaluate how various AFAs will be able to perform against their production workloads. The document culminates with a set of suggestions for an accurate yet time-efficient testing methodology that will accurately reflect the performance that AFAs will deliver in production use. The objective of this study is to propose a framework that will produce meaningful results within a 10- to 11-day testing regimen (assuming 8-hour days) plus a 48-hour soak test.

SITUATION OVERVIEW

The Flash-Based Array Market

Although enterprise storage arrays built from flash memory have been available since the 1980s, the extremely high dollar-per-gigabyte costs of these systems relegated them to a very small niche market for decades. Prices dropped slowly over that time period, and by about 2007, some enterprise storage vendors began to offer a limited-capacity flash-based caching layer in their arrays. By 2009, a new set of vendors with all-flash arrays began to emerge. Over the next five years, almost all enterprise storage vendors introduced flash-based arrays in their solutions portfolios, either by adding solid state disk (SSD) drives to their existing HDD-based architectures or by introducing new AFAs of their own. By 2014, the worldwide AFA market grew to $811 million in revenue, while the HFA market hit $8.3 billion.

The rise in the importance of flash in enterprise storage solutions is being driven by evolving I/O patterns in the era of 3rd Platform computing. With the maturity of virtual infrastructure, many existing production applications are being moved to hypervisor-based server platforms, and many enterprises are following a "virtual first" strategy for new application deployment. New applications for mobile computing, social media, big data/analytics, and cloud computing are driving new I/O patterns where steady state and peak requirements can vary by as much as 10 times and the overall IOPS requirements at the datacenter level are measured in the hundreds of thousands. These workloads are all hosted on common virtual infrastructure, producing extremely random I/O at the virtual host level. This new 3rd Platform computing workload is demanding more storage performance than HDD-based storage systems can cost effectively provide, and that more than anything is driving the rapid deployment of flash in the datacenter.

Flash is here to stay, and it is transforming the datacenter. Flash can be integrated either into the server in the form of memory channel–attached storage (e.g., SanDisk ULLtraDIMM), PCIe-based flash cards (e.g., HP ioDrive2), SSDs (e.g., Samsung), or into the network in the form of AFAs or HFAs. AFAs are defined as storage arrays that can use only flash storage to meet performance and capacity requirements, whereas HFAs are defined as storage arrays that can use both flash storage and traditional HDDs to meet performance and capacity requirements. Note that some vendors sell all-flash configurations of HFAs, but IDC still considers those to be HFAs.

Flash-Based Array Testing Requires a New Approach

Flash-based arrays, regardless of whether they are AFAs or all-flash configurations of HFAs, exhibit very different read/write behavior relative to HDD-based arrays. With traditional arrays, the goal of testing was to understand how much I/O could be serviced until the array cache or flash tier was saturated. The performance of these arrays would dramatically diminish when this tier 0 storage layer became saturated as I/O performance would degrade to HDD-type latencies, IOPS, and throughput. While HDDs exhibit relatively constant performance across their entire life cycles, flash storage does not. In fact, flash performance can vary by as much as 80% between when a flash-based system is new without any program/erase (P/E) cycles having been performed and normal, long-term steady state operation. Flash storage also exhibits read/write asymmetries that HDD does not. Therefore, on AFAs or all-flash configurations of HFAs, classic storage testing methodologies will not appropriately characterize the type of storage performance that array will deliver in a production environment with 3rd Platform computing workloads.

For storage administrators familiar with testing traditional HDD-based arrays, key differences to consider when testing AFAs and all-flash configurations of HFAs include:

- **Preconditioning.** Preconditioning should be performed on every flash-based array prior to starting any testing. While traditional HDD-based arrays exhibit similar performance throughout their entire life cycle, flash-based array performance will differ significantly from FOB and after first write on every flash cell. The definition of FOB is that none of the flash cells in the array have been written to. Without comprehensive preconditioning that transitions an array from FOB to steady state, flash-based array performance test results will not reflect real-world results.
- **Read/write asymmetries.** Traditional HDD can overwrite data in place and does not lock data on writes. Flash, on the other hand, has unique attributes, such as the need to erase before rewriting and cell locks on writes, which impact the ability of flash-based systems to deliver consistently predictable performance over time.
- **Endurance.** Flash is a consumable resource. After a defined number of P/E cycles, flash will become read only and ultimately inoperable. How the flash-based array manages PE cycles determines the rate of flash media consumption and flash endurance. Flash endurance has implications on the total cost of ownership (TCO) of flash-based arrays and for enterprise use is expected by customers to be the same as that of HDD-based arrays that are targeted for the datacenter (i.e., five years at a minimum).
- **Data stream/data set definition.** The mixed virtual workloads in 3rd Platform computing are very different than traditional client/server workloads, and it is important to test flash-based arrays using an appropriate workload. A relevant workload will exhibit a variety of read/write

ratios, use a wide distribution of block sizes, is heavily skewed toward random I/O, has a high percentage of data that is reducible, and is characterized by I/O bands that drift over time. It is difficult to reproduce these types of real-world mixed workloads with traditional freeware load generation tools.

- **Load generation.** Flash-based arrays generally deliver at least 10 times better performance, in terms of both latencies and throughput, than traditional HDD-based arrays. Testing and load generation tools that have been designed for HDD-based arrays often cannot deliver sufficient load to adequately stress the performance capabilities of flash-based arrays.

In *Flash Optimized Storage Architectures* (IDC #249295, June 2014), IDC provides a definition of "flash optimized." When arrays have been designed from the ground up for flash, they include functionality that is specifically intended to increase the performance and endurance of flash-based storage that was not included as part of traditional HDD-based arrays. Many of these types of features were initially available only in AFAs, but HFA vendors are increasingly flash optimizing their platforms to make them more competitive with the performance delivered by AFAs. While these flash-optimized HFAs do use storage tiering and caching approaches, they also incorporate unique features like "flash first" for all writes that change their performance characteristics relative to more traditional HFAs. These types of arrays can exhibit performance characteristics that are more similar to AFAs than they are to HDD-based arrays, and many of the recommendations about how to test AFAs in this document apply to them as well.

Understanding Flash Storage Behaviors

Flash storage is built from semiconductor memory components like NAND or DRAM. Flash latencies are on the order of microseconds, whereas HDD latencies are on the order of milliseconds. Individual flash cells are combined into pages, and pages are combined into erase blocks. Individual flash cells can be read, but they cannot be written to – they can only be written to as part of an entire erase block. Note that this is very different from HDDs where individual data blocks can be written to without impacting other data on the drive.

Flash cells can be in one of several states: unprogrammed, programmed, locked, and available to be written. Flash is only ever "unprogrammed" once in its life – after it has been written to once, it will only ever be in one of the other three states. "Programmed" means that the cell contains data that can be read. "Locked" means that the cell is being written to (generally as part of an erase block). And "available to be written" means that the cell can be overwritten (either the data is invalid or it has been zeroed).

Flash-based arrays employ one of two architectures: flash memory based and SSD based. Flash memory-based arrays use custom flash modules, designed to the array vendor's specification and generally manufactured by third parties. Vendors using flash memory-based architectures include IBM (FlashSystem), Skyera (skyHawk), and Violin Memory (Concerto 7000). SSD-based arrays use off-the-shelf commodity SSDs. Vendors using SSD-based architectures include EMC (XtremIO), Kaminario (K2), and Pure Storage (FlashArray). While there may be performance differences based on architectural implementations, the discussion in this document concerning testing to best mimic real-world performance applies equally to both types of architectures.

When handling sustained I/O, flash exhibits behaviors that result in unique performance characteristics at the system level. For those new to flash storage, or for those who want a quick refresher on how flash behavior is different from spinning disk, here is a quick summary of flash behavior attributes:

- P/E cycles. Flash is a consumable resource with a wear rating similar in concept to tread life on tires. Each time an individual flash cell is erased and programmed (i.e., written again), the flash cell incurs a slight amount of damage known as wear. A single iteration of erasing and writing to a flash cell is known as a "program/erase cycle" (P/E cycle). Flash media is rated by how many P/E cycles can be sustained before the flash cell has degraded to the point where it can no longer be written to. Enterprise I/O generally has some mix of read/write, making the flash media at this point basically unusable. If flash continues to be read beyond this point, it will also ultimately become unreadable.
- Asymmetric behavior. Although on average flash storage delivers latencies that are at least an order of magnitude lower than HDDs, read/write I/O can exhibit different latencies depending on the state of flash media at the time an I/O is attempted. Flash media does not have any moving parts and can generally handle random or sequential reads in microseconds unless an erase block is locked. Writes will also occur in microseconds if an erase block is free (in the unprogrammed state or has been freed up by garbage collection [discussed in the bullet points that follow]). If the page is available to be written but has data in it that must be overwritten, a P/E cycle must occur. At a system level, this erase operation can create a significant performance penalty for writes relative to reads, potentially making write operations significantly slower than reads.
- Overprovisioned capacity. Each SSD, flash module, or flash-based array has a set amount of capacity reserved for administrative activities associated with flash performance, endurance, or reliability. This overprovisioned capacity is not available for production use and is not counted as part of the device's usable capacity. All things being equal, more overprovisioned capacity leads to greater endurance at the device level. Although some vendors allow an administrator to change the amount of overprovisioned capacity, each vendor has a recommended level that should be used. Changing the overprovisioned capacity will affect the performance, endurance, usable capacity, and effective dollar per gigabyte an array delivers. Administrators should ensure that a tested and purchased system employs the same level of overprovisioned capacity.
- Wear leveling. Flash cells have a specified wear rating. If certain cells are used frequently, they may wear out significantly sooner than other cells in the same array. Using a technique referred to as "wear leveling," many flash-based arrays attempt to spread writes out as evenly as possible across all of the available flash media within an array, including the overprovisioned capacity. Wear leveling helps improve the overall endurance and reliability of the flash-based array and helps maintain more consistency to operations like garbage collection.
- Free space management. To write even a single byte, an entire erase block must be available. A background process for free space management, referred to colloquially as "garbage collection," works periodically to ensure that systems always have a ready supply of available blocks. As data gets invalidated on the flash media, it is not immediately overwritten but rather marked as "available to be written." Garbage collection conceptually reads data from one or more blocks, discards invalid data, combines valid data to get (if possible) a full block, updates the block map, and then writes the data back to flash. This process is effectively consolidating data onto fewer blocks and in the process freeing up new blocks that are then available to be written.

When garbage collection is occurring, certain blocks may be locked and can be neither read nor written. There is a potential for garbage collection to interfere with real-time I/O performance, thereby unpredictably inserting added latencies into I/O operations that can have a potentially large impact. It is important when testing flash-based arrays to exercise them to the point that garbage collection could occur at any time as this will provide more accurate real-world results.

- Write amplification. Writes generate wear on flash cells. Relative to write I/O requested by an application, there are two types of processes in most flash-based arrays that amplify the number of writes that actually occur within the array: garbage collection and data protection. In most cases, the chunks of data being written are smaller than the defined size of an erase block. Because of the way garbage collection works, a single piece of data may be rewritten multiple times as it is moved around to different blocks before it is changed by an application. Also, data protection schemes like RAID, erasure coding, and replication will generate multiple writes for each application write as data is protected through added redundancies. While administrators should be cognizant of the efficiency of both garbage collection and data protection and the impacts various implementations will have on flash endurance, testing regimens will not be long enough to produce any overt endurance differences between arrays.
- FOB versus steady state performance. When an AFA is FOB, it will be able to perform reads and writes at a noticeably higher level than after it has been transitioned to steady state. This is because garbage collection operations intermittently occur with an array in steady state, potentially increasing in particular write latencies, but they never occur with an array that is FOB. To transition an array from FOB to steady state, all-flash cells (including overprovisioned cells) in the array need to be written to at least once. In normal use, all-flash cells may not be written to for days, weeks, or even in some cases months, depending on the write intensity of production workloads. The purpose of preconditioning is to transition the array to steady state. Steady state performance is representative of the performance the array will consistently deliver throughout its useful life.

  With an array that has been transitioned to steady state, garbage collection operations may kick in at any time. Flash-based array vendors implement different approaches to garbage collection, however, and it is difficult to deterministically state when they will occur. Often they are managed at a systems level by the array, which can schedule them to occur so as to optimize performance and endurance for that particular array configuration and workload. Real-life workloads are generally cyclical, with bursts of high write activity followed by periods of much lower write activity. If an array has been transitioned to steady state and is then subjected to a real-life workload, it will exhibit performance results (over the course of a test run of one or more hours) that are indicative of the performance it will deliver in actual production use. Loading an array so as to force it to go into garbage collection mode before recording test results does not generate a relevant representation of the array's performance on typical 3rd Platform computing workloads (unless the workload you use to force garbage collection operations is very similar to your actual workload).
- Flash translation layer (FTL). Flash storage is generally combined with an FTL (FPGA, ASIC, or software based) that provides a logical block interface for operating systems to access the flash media as if it were a traditional HDD. Traditional data services like file systems have no knowledge of the read/write semantics of flash media, and the FTL provides the appropriate translation between the data service and the flash media. For example, when a file system issues a write I/O, the FTL transparently handles the "erase before write" requirement of flash media. The FTL also abstracts the logical location of the data within the storage device (a flash module or SSD) from the physical location on the actual flash cell pages.

Critical Factors Impacting Flash-Based Array Performance

For testing to produce the most meaningful results, it should as closely mirror how an array will actually be used in production over time as possible. Think about your plans with a flash-based array. What type of flash media are you using (single-level cell [SLC], multilevel cell [MLC])? Will inline data reduction features like compression and deduplication be used? Which protocols and line speeds will be used in production? Will data services like thin provisioning, snapshots, clones, and replication be used? Does the array have configurable quality-of-service (QoS) controls that will be used in your environment? In detail:

- **Flash media type.** Single-level cell is a flash nonvolatile memory cell with only one physical bit of storage in the cell area. SLC is the most durable form of flash media, generally rated at 25+ drive writes per day (DWPD) over its stated useful life. SLC-based devices generally exhibit a higher percentage of overprovisioning to help further improve the overall endurance – one of the reasons for their higher cost. SLC also delivers the best read/write latencies.

  Multilevel cell is a flash nonvolatile memory cell with two physical bits of storage in the cell area. MLC does not offer the same performance or endurance as SLC, but it is quite a bit less expensive on a dollar-per-gigabyte basis. Different MLC implementations exist, including enterprise MLC (eMLC), which can support roughly 10 DWPD, and consumer MLC (cMLC), which supports in the range of 1–3 DWPD. All MLC implementations offer roughly the same performance although there are differences, so ensure that an array that uses one of the MLC implementations can meet your specific performance requirements. The trend in flash-based arrays is to build them from MLC flash media to provide a lower cost solution, offsetting the lower performance and raw endurance of MLC with architectural and software innovations. At this point, appropriately architected MLC-based arrays can easily provide consistent, sub-millisecond latencies and at least the same five-year depreciation life cycle that HDD-based arrays offer.

- **Inline data reduction.** Inline data compression, deduplication, and pattern removal (in repeating character strings) add latency; however, they can be performed on many flash-based arrays without impacting their ability to consistently deliver sub-millisecond response times. Data reduction ratios vary based on workload, but they can easily vary between 0 and 10:1 or more and will have a huge impact on effective dollar per gigabyte and TCO. If you plan to use data reduction in production, you will need to test appropriately. Note that most mixed virtual workloads exhibit a high degree of reducibility, although certain reduction technologies are more effective on different workload types. See *Flash-Optimized Storage Architectures* (IDC #249295, June 2014) for a more complete discussion of this issue. Because of the significant economic benefits of inline (not postprocess) data reduction, IDC strongly recommends this as a baseline feature in any AFAs or all-flash configurations of HFAs you may consider. Most 3rd Platform workloads exhibit a high percentage of reducible data.

- **Protocols.** Storage protocols like PCIe, SAS, and SATA (which tend to be server based) and FC, iSCSI, and FCoE (which tend to be network based) deliver different levels of performance. These protocols vary in their error correction overhead, throughput with different block sizes, and raw and effective bandwidth. Test first with the protocols you plan to deploy in production, and then, time permitting, test other protocols appropriate for your road map.

- **High availability.** If you plan to operate in a highly available environment using features like redundant controllers, multipath I/O, RAID levels that can transparently sustain two or more drive failures, and failover, test with these. Pay particular attention to how dual-controller architectures are implemented and what the implications are for maximum IOPS, overall bandwidth into the array, and performance in degraded modes.

- **Data services.** Many flash-based arrays include enterprise-class data services like thin provisioning, snapshots, clones, and replication that can be very useful for efficient capacity utilization, implementing quick recovery options, provisioning storage faster, and setting up disaster recovery configurations. Depending on how these are implemented by a vendor, there may or may not be performance impacts to their use. Test actual production workflows so you know how storage latency and throughput are affected at a system level when these features are used.
- **QoS.** QoS features let you establish performance parameters on an application-by-application basis that are extremely valuable when you will be running mixed workloads in production. QoS has more of a performance impact at the application level than it does at the array level, but if you plan to use QoS, then run your performance-testing regimen with it configured how you plan to use it, testing its ability to consistently deliver defined performance where specified in mixed workload environments as load is scaled.
- **Metadata management.** Much of what occurs in a flash-based array to optimize the performance, endurance, and reliability of flash media is metadata processing. How the array does metadata processing, and where it keeps the data it uses during this (e.g., in main memory or in flash) can have a large impact on performance. Where metadata is kept can impact the overall performance and scalability of the system as well. A system that keeps all metadata in DRAM may be faster than one that splits it between DRAM and flash, but it may support a significantly smaller maximum capacity (because it has less capacity to store metadata). A smaller maximum capacity can limit the maximum data reduction achievable, potentially also impacting the overall TCO of the array. Keeping all metadata in RAM may also impact resiliency (as in the array's ability to sustain nondisruptive operations during maintenance).
- **Block alignment.** Proper block alignment in workloads with little variability in block size will minimize the number of blocks that have to be accessed to meet any given I/O request. In these situations, improper block alignment can contribute to as much as a 30% loss in performance if not resolved over time and can also negatively impact flash endurance as it increases write amplification. Unfortunately, mixed virtual workloads generally exhibit a high degree of variability in block sizes, so efforts at block alignment (some of which are attempted automatically by the arrays) will deliver more of a performance benefit in situations where a flash-based array is deployed for only a single application.

Finally, not all flash-based arrays will offer all features. Decide what your minimum feature requirements are before you begin testing. Only compare test results between similarly configured arrays. For example, there could be very noticeable latency differences between test results with and without inline data reduction enabled.

Testing Overview and Scope

There are three critical profiles you should understand about your actual production workload: the data stream, the data set, and the workflows. A data set is created by sending an array a certain set of data streams. 3rd Platform workload data streams generally exhibit a variety of read/write ratios, use a wide distribution of block sizes, are heavily skewed toward random I/O, have a high percentage of data that is reducible, and are characterized by I/O bands that drift over time. The resulting data set, and the metadata state of the array, is the result of the array handling this data stream over time. The workflows include tasks that generate specific workload profiles, like (for example) storage provisioning, snapshot and/or clone creation, batch jobs, backups, and snapshot-based replication.

In the universe of performance testing, there are five basic types of tests that could potentially be performed, and these are ranked in order of relevance to understanding how an array will perform on your actual production workload:

- Class A: Workload testing using your actual data streams, data sets, and workflows
- Class B: Workload testing using data streams, data sets, and workflows that are generated using application-specific testing tools like SLOB (Oracle), Jetstress (Exchange), Login VSI (VDI), and others
- Class C: Workload testing that uses data streams, data sets, and workflows generated by general-purpose workload generation tools like Load DynamiX, FIO, or vdbench that are capable of very accurate modeling
- Class D: An intelligent performance corners test using generic data sets, data streams, and workflows designed to closely model 3rd Platform computing workloads
- Class E: Hero testing that encompasses most classic performance corners testing

Class A testing, if you could do it, would provide the most accurate way to forecast how an array would perform in production with your actual workload. Unfortunately, privacy and other security concerns can make this very difficult to do and most organizations cannot do it. Class B testing can produce very accurate results as well but can be very time consuming if you model each application separately and can add additional cost. Organizations may do this type of testing for a specific application but generally will not take the time to do this to model mixed workloads that are consolidated onto a single array.

Class C testing is more of a generic testing methodology and is generally more time efficient than Class B, but it can require some additional expense (for something like Load DynamiX) and proficient scripting skills to use tools like FIO or vdbench to create the data sets, data streams, and workflows that you need. With mixed workloads, Class D testing can be less time intensive than Class C testing and can produce very relevant results but does require good scripting skills as well. Class E testing produces results that do not accurately represent the performance flash-based arrays will deliver for most 3rd Platform computing workloads and should *not* be used to compare the performance of flash-based arrays for these types of workloads.

With this document, IDC hopes to inject reality into the performance testing of flash-based arrays. Historically, Class E testing has been used with HDD-based arrays, generally because it has been the easiest to do. But because of the significant differences between data streams and data sets with 3rd Platform workloads, IDC does *not* recommend using Class E testing. It will not produce results that will allow meaningful comparisons between flash-based arrays if you will be using them with mixed virtual workloads. If you can do Class A, B, or C testing right, you will be able to achieve a very realistic representation of the performance different flash-based arrays will offer in your environment. But Class D testing will produce results that are also meaningful for comparisons and may be easier to do than the other three higher classes. The specific recommendations in this document, assuming that you cannot do Class A, B, or C testing, fit the Class D test definition.

At its core, IDC's proposed Class D test regimen is intended to very efficiently determine how a particular array will perform in your environment and to provide results that allow you to accurately compare different options. Planning your testing regimen and scope out up front is the best way to get a comprehensive set of results that accurately represent what your experience with an array will be in your production environment over the long term. Your planning should include these steps:

- **Determine your data stream and data set mix.** If you know the characteristics of your actual workload, use that. But this document assumes that you do not know that in detail and is proposing instead a limited data stream and data set that will allow you to very efficiently replicate actual 3rd Platform workloads in a very time-efficient manner. What is important is that the test data stream and data set exhibit a variety of read/write ratios, use a wide distribution of block sizes, are heavily skewed toward random I/O, have a high percentage of data that is reducible, and are characterized by I/O bands that drift over time.
- **Define and document your test plan.** The intent is to use data streams, data sets, workflows, and in general a test plan that will stress the array *in the same manner* that you will use it over time in your production environment. Documenting the test plan ensures not only that you comprehensively cover needed areas but also that you can easily recreate that test plan on other arrays if you will be doing a bake-off. In this document, IDC is recommending a test plan that will likely span between 10 and 11 days per array (plus a 48-hour soak test), covering array setup and preconditioning, data set preparation, performance testing, functional testing, and fault injection testing. Regardless of how you choose to do it, if you are comparing performance between two or more arrays, you must execute the same test plan across all arrays to be able to perform accurate comparisons.
- **Execute the test plan.** Given the workloads for which flash-based arrays are typically deployed, it is a challenge to adequately reproduce them with ad hoc scripting and traditional freeware tools like iozone and iometer. If you are going to go the freeware route for general-purpose storage performance testing with flash-based arrays, you may want to consider FIO, vdbench, or BTEST because they at least have a built-in ability to generate compressible and dedupable data (which is a large component of most virtual workloads). Or this may be an area where you want to consider purpose-built workload-generating appliances that can accurately characterize and reproduce complex mixed storage workloads (e.g., Load DynamiX) or very application-specific workload generation tools like LoginVSI (for VDI environments) or any of the various Oracle testing tools like SLOB, Hammerora, or Swingbench.
- **Evaluate the results.** If you establish objectives and define what "success" looks like up front, it will be easier to quickly evaluate if an array meets your needs or which array meets your needs most cost effectively.

The AFA Performance Testing Framework

HDD-based arrays generally produce tens of thousands of IOPS, whereas flash-based arrays can produce hundreds of thousands to millions of IOPS. The IOPS for any given array will vary depending on the block size, so it's difficult to specify a single IOPS target that will be appropriate for all customers. Understand the stated performance of the array you are testing at various block sizes, and take your workload requirements and projected growth into account as you set up your test harness. Ensure that your test equipment can generate sufficient workload to push the performance boundaries of the array you are testing.

Table 1 provides a summary of the AFA test plan.

TABLE 1

AFA Test Plan Summary

Phase	Description	Tasks	Estimated Elapsed Time
1	Setup and preconditioning	2 x 85% overwrite	24 hours
	Data set preparation	2 x 5% overwrite 5 x 1% overwrite	6 hours
2	Baseline workload testing	Develop scripts	6 hours
		Thread/queue depth optimization	6 hours
		IOPS ramp testing (4 runs at 3 hours per run)	12 hours
3	Functional testing	Document workflows	6 hours
		Test workflows (4 runs at 4 hours per run)	16 hours
4	Fault injection testing	Inject failures (4 runs at 2 hours per run)	8 hours
5	Soak test	Run 4 baseline workloads consecutively (12 hours each)	48 hours
			132 hours

Source: IDC, 2014

Phase 1: Workload Definition, Setup and Preconditioning, and Data Set Preparation (One to Two Days)

Workload Definition

This document assumes that you will be doing Class D testing of your array. To approximate a typical 3rd Platform workload, you will need to work with data streams and data sets that exhibit a variety of read/write ratios, use a wide distribution of block sizes, are heavily skewed toward random I/O, have a high percentage of data that is reducible, and are characterized by I/O bands that drift over time. For workload definition, identify and use actual recurring workflows in your existing environment.

In IDC's experience, read/write ratios vary between write intensive with a mix of reads and read intensive with a mix of writes but rarely include 100% write or 100% read ratios. Block sizes will vary between 4K and 256K or larger but will likely average out to somewhere between 24K and 32K across the entire distribution of block sizes. It is highly recommended that you model the block size distribution in your actual production workload and plan to use that mix in your performance testing. The I/O profiles tend to be 60-80% random, with the remainder made up of sequential streams. Data reduction ratios, assuming compression, deduplication, and repeating character string removal (but not thin provisioning), generally vary between 4:1 and 6:1 for mixed workloads, although for certain workloads (like databases) they can be lower, and for other workloads (like VDI), they can be much higher. There will generally be I/O bands with drift over time, and the data stream and data set recommendations we make take both temporal and spatial locality into account.

We suggest a range of four baseline workload profiles in the Phase 2: Baseline Workload Testing (One to Two Days) section that should closely approximate 3rd Platform workloads and allow for very time efficient yet very accurate performance testing of flash-based arrays in these environments.

Setup and Preconditioning

As you set up your test harness, keep a few things in mind. First, to accurately understand the limits of the array, it must be the bottleneck for peak performance measurements. This means you need to configure sufficient capability for workload generation and network bandwidth to ensure that is true. Exchanging an HDD-based array for a flash-based array could possibly move the performance bottleneck to some other location in your infrastructure besides the storage array. Make sure you check for any needed adjustments in your test harness to ensure that the flash-based array is the performance bottleneck. You should also make sure that your test harness is well isolated from your production environment so that test activities do not impact business operations.

Second, if you plan on consolidating workloads currently running on separate storage systems onto the flash-based array, the interaction between workloads when run on a single array, regardless of its performance, could easily create unexpected results that are not easily extrapolated from workload profiles in isolation. This is often referred to as the "noisy neighbor" problem. If you plan to use your flash-based array for mixed workload consolidation, you do need to test with this type of workload to best gauge the performance it can provide in your environment. Because of this, it may be difficult to accurately configure your flash-based array for optimal performance on this workload without some trial and error during initial testing. Keep an eye out for this.

If you will be performing comparisons between two or more flash-based arrays, be sure you follow your written test regimen in the same manner for all arrays. Configure all arrays similarly in terms of LUN sizes, RAID levels, and the enablement of various data services. Note that even on the same array, IOPS, throughput, and latency can vary significantly against the same data streams and data sets when data reduction is enabled as compared with when it is not.

Finally, most flash-based array vendors have a recommendation for a maximum addressable array capacity during normal operation, which is often in the 80-85% range of maximum physical capacity. All of your baseline workload testing (Phase 2), functional testing (Phase 3), and fault injection testing (Phase 3) should be performed with this amount of addressable capacity configured in the array. Configure relatively large LUNs (at least two to three times larger than the array RAM cache), allocate 90% of the array's addressable capacity to them, and use the RAID level(s) you plan to use in production.

Now we turn to preconditioning. A flash-based array goes through three distinct phases in its life cycle: FOB, transitioning, and steady state. For array performance to be consistent with real-world results, tests must be run in steady state. To move a flash-based array from FOB to steady state, every flash cell in it must be written at least once. Preconditioning moves a flash-based array from FOB to steady state.

Although it can take days, weeks, or months in normal production use to write to every cell within an array, a focused approach can accomplish this within a 24-hour period for many arrays. To ensure you write to every cell, including the overprovisioned capacity, plan to do a two-pass preconditioning run. Calculate 85% of the maximum physical capacity (not the actual configured capacity) of the array you will be testing and multiply that by 2. That is the amount of data you will need to write to complete preconditioning. For the most accurate estimates, use base 2 (1TB = 1,024GB) calculations. To complete preconditioning quickly, use a 100% large block (256K) write workload with nonreducible data. Track IOPS and latencies so you can see the point where the array, over time, moves from the higher performance of FOB to the lower performance of steady state. If you do not see this inflection point drop in performance, your array has not yet been transitioned to steady state so start another 85% overwrite run.

You can estimate how long it will take to transition your array by multiplying 85% of its maximum physical capacity (not the addressable capacity you configured) in gigabytes by 2 and then dividing by 86,400 (the number of seconds in a day). That will give you the gigabyte per second you need to write to complete both passes in a 24 hour period. You can then adjust based on how much write throughput you can generate during the array preconditioning process. Do *not* do an SSD reset after you have completed this step.

Data Set Preparation and Artificial Aging

Now that the array is transitioned, you will need to transform the data with which it is filled to look more like a typical 3rd Platform workload. If your workload is typical of other 3rd Platform configurations, you want to transform the data set that you will use for testing to look like what a data stream that exhibits a variety of read/write ratios, uses a wide distribution of block sizes, is heavily skewed toward random I/O, has a high percentage of data that is reducible, and is characterized by I/O bands that drift over time would produce over a period of time.

The data with which the array is filled from the two initial preconditioning passes does not match this, and it must be modified to make it look more like an actual 3rd Platform production data set that has been in use for awhile. Do two additional passes, using sequential, small block writes of data that is reducible to a 5:1 ratio and exhibits a mix of block sizes that average out to your average block size. In virtual environments, the I/O blender effect combined with server consolidation pretty much eliminates the possibility of having a uniform block size, so to best mimic real-world environments, you should not test with one (even one that matches your average block size). Plan to write at least 5% of the array's maximum physical capacity (not 5% of the actual capacity you configured) on each run, but ensure that the amount of data written on each run is larger than the array cache. If 5% of the array capacity is not larger than the array cache, then increase the percentage as needed.

Now we want to modify this data further to make it look like it has been in production use for awhile and has created a very rich metadata set. You want to "artificially age" the data in the array, producing in essence a time-compressed version of a real-world 3rd Platform workload. To do that, you will perform five additional runs, each of which will write 1% of the array's maximum physical capacity. Each run should have a defined skew in terms of spatial locality, and that skew must be varied across each of the five runs. Use 100% random writes with the same block size distribution you used for the data set preparation as previously mentioned. When these five runs are complete, you should have a data set with very rich metadata that closely approximates what an actual 3rd Platform workload that has been run for a week or so would have produced. Generally in testing an array, you want to know how it will handle complex metadata states over time against a workload very similar to your own, and these preliminary steps should allow you to create that in the array within a 1-2 day period.

Phase 2: Baseline Workload Testing (One to Two Days)

To perform our Class D baseline workload testing, it is suggested that you use the four following profiles:

- 20/80 read/write ratio, all sequential, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift
- 65/35 read/write ratio, all random, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift
- 35/65 read/write ratio, all random, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift
- 80/20 read/write ratio, all sequential, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift

Note that we are not recommending any testing using either 100% writes or 100% reads. Because real workloads generally do not do that, the results from this type of test would not be relevant to understanding a flash-based array's performance on a typical 3rd Platform workload. Note, however, that hero testing often uses these types of unrealistic workloads to produce eye-popping performance results for marketing purposes. These types of results should not be used to compare the performance of flash-based arrays that would be running your 3rd Platform workloads because they are not a good indication of real-world performance. Just because an array performs extremely well on a workload of 100% random small block writes does not mean that it will perform better than another array on a real-world 3rd Platform computing workload.

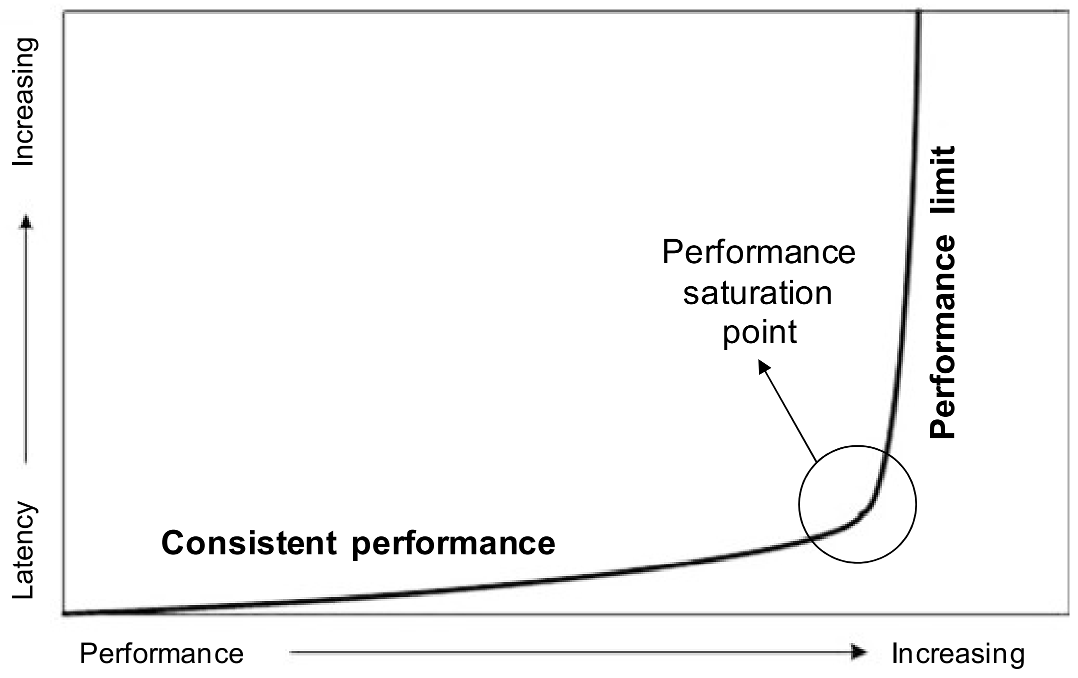
During workload-oriented testing, the goal is to add workloads with specific I/O profiles to the array over time to generate an understanding of how IOPS, throughput, and latency react as a varied load is increased. After you have developed your four threads, start by determining the maximum performance for each thread. Make sure you design the threads to randomly touch all logical block addresses within the test LUNs you have configured but also to exhibit drift over time. Start the thread in isolation and vary the queue depth until you find the optimal one for your performance objectives (IOPS, throughput, latency, etc.). Each vendor will have its own set of recommended queue depths for its array, so start from there and adjust as necessary. If you are familiar with HDD-based testing, note that flash-based arrays exhibit significantly lower latencies than HDD-based arrays, and they will achieve optimum performance per thread with relatively shorter queue depths (Little's law).

Once you've determined the optimal queue depth, you can determine your starting thread count. You want to pick the starting thread count to ensure that you will see results during your test that cover both below, at, and way above what you expect your actual workload will initially require when you deploy while at the same time making the most efficient use of time. If a thread generates, for example, 5,000 IOPS, and your flash-based array is rated at 300,000 IOPS, and you think your initial workload will require 140,000 IOPS, you may want to start with 22 threads and then scale it up to beyond 60 threads (to ensure that you go beyond the performance saturation point). Keep in mind that the lower you start, the longer each test run will take, but in any event you need to cover the performance range that is relevant to your environment.

Figure 1 shows an example conceptually of the type of graph you should end up with if you plot IOPS (or load) on the x-axis and latency (or response time) on the y-axis. The performance saturation point is the point where latency increases exponentially with additional load. The inflection point in the right-hand portion of Figure 1 is the performance saturation point.

FIGURE 1

Example of Real-World AFA Performance Graph



Source: IDC, 2014

You will start test runs at your determined starting worker thread count and "ramp test" by adding load at several minute intervals until you have well exceeded the performance saturation point. Ideally, you'd like to see the array operate in steady state at each worker thread count before adding another thread. Most likely, many threads and multiple servers will be needed to generate sufficient load to saturate a flash-based array. If you use Class C workload testing equipment like Load DynamiX, that may not be true since a single load generation appliance can produce as much as 7 million IOPS, making it much easier to generate the loads you'll need. You want to test far enough beyond the performance saturation point to, one, ensure that you have in fact passed it, and two, to understand where and how failure occurs. Does performance gracefully degrade? Does latency start to vary widely? Are there other impacts that occur at this point of which you need to be aware? Individual test runs are likely to be at least one to two hours, if not longer, depending on your starting worker thread count and the performance ceiling of your array. Test runs of this length will ensure that you are getting results that accurately reflect what the flash-based array will deliver against real-world 3rd Platform workloads.

Do *not* perform SSD resets in between test runs. SSD resets clear metadata and will affect the performance the array will deliver — making it higher — until the metadata again starts to look like what the metadata will look like from real-world use over time. The most relevant results for performance comparisons using real-world 3rd Platform workloads are produced when the arrays are operating in steady state.

Many AFAs track and display the data reduction ratios they are achieving on average across production workloads. It is strongly suggested that you confirm these ratios on your own during the testing. Different vendors measure data reduction ratios differently. Any short-term space savings due to thin provisioning should not be included in your data reduction ratios. This may require some extra calculations, but doing them ensures that you understand up front (particularly if you are able to use a data set that closely mimics your actual workload as we recommend) exactly what you will be getting. If you will be using thin-provisioned storage all the time, then you should test with thin-provisioned storage.

Phase 3: Functional Testing (One Day)

Most datacenters use automation to perform recurring workflows on a daily, weekly, or monthly basis. These workflows may leverage storage management features such as snapshots, clones, replication, encryption, and QoS. Your goal here is to test these recurring workflows in conjunction with several different baseline mixed workloads to determine what, if any, performance impacts there are when you use the array in the manner you intend to use it in production. Run these tests using the same four different baseline mixed workloads that you used in Phase 1 testing, performing the functional testing while they are running. Plan to perform all the functional tests you want to do for each baseline mixed workload run. You can create your own tests to reflect your workflows, but some common ones are:

- Snapshots and clones. If you create, retain, and use snapshots and/or clones on a regular basis, you will want to understand the performance implications of this mixed use. Test your workflows against the four baseline mixed workloads using three static snapshot/clone configurations: 50% volumes and 50% snapshots; 50% volumes and 50% clones; and 50% volumes, 25% snapshots, and 25% clones. Note that readable snapshot performance can differ from clone (writable snapshot) performance, so if you will be using clones, test their read/write performance. Snapshots and clones may provide degraded performance relative to the source volumes, and/or performance may degrade over time as more are retained and used. Understand the behavior of the array as snapshots and clones are created and deleted, both in order and out of order.
- Snapshot backup. If you create snapshots of certain applications at four-hour intervals to maintain a disk-based, quick recovery option, always keep the most recent six snapshots and then explore the performance implications of this. Start each baseline mixed workload and then create six different snapshots in quick succession. If you plan to use snapshots APIs like VMware VADP or VAAI, Windows VSS, or Oracle RMAN, then perform the snapshots that way. Then manually shut down the workload generation for one "application" and, while the

others continue to run, perform a recovery from the most recent snapshot. Note any performance impacts on other workloads running in the array.

- Desktop provisioning. If one of your workloads is a VDI environment, then you may want to test clone performance with high numbers of clones. Start each baseline mixed workload and allow it to reach steady state. While the baseline workload continues to run, determine the time it takes to create 1 clone, 20 clones, and maybe even 100+ clones. There are multiple ways that clones can be created, so make sure that you test the workflow(s) that are relevant to your environment. Mount them on VMs and compare their performance with the original source clone. Does performance degrade as more clones are in use?
- Replication. This is an optional test, since you are not likely to have both a local source and a remote target array available to test actual replication performance across a network. However, you may want to test local replication and evaluate how easy it is to configure and use it. You may also want to perform some fault injection testing with local replication in Phase 4. If so, start each baseline mixed workload, allow it to reach steady state, and then enable replication. Configure and use realistic consistency groups as you test this. Test recovery with replicated snapshots and consistency groups.
- Encryption. If you plan to use encryption, you will want to understand if there is a performance hit when it is enabled. Perform a single baseline workload test run with encryption enabled, and compare the results to the performance of that same baseline workload test run you performed in Phase 2. Don't perform other functional tests during this run.
- QoS. If you will be running mixed workloads at high levels of density on your array, you will want to test the array's ability to consistently deliver predictable performance as array headroom decreases. Use the QoS feature to define the level of performance you want for each volume or volume group. Just like in your real-world environments, some volumes will have very high priority, and others will not and may only receive performance if it is available. Increase the load by increasing the thread count to well beyond the performance saturation point of the entire array. Observe what happens with volumes of different priority. What you want to see is that your highest-priority volumes receive their full complement of performance, based on the rules you have established, until you have reached the performance saturation point.

If you only ever plan to run one application on your array, then you will not need to test QoS.

Phase 4: Fault Injection Testing (One Day)

Failures are a fact of life in datacenters. Many of the flash-based arrays employ redundant designs that make failures transparent and allow for the online replacement of failed components without impacting read/write I/O. You will want to verify this, however. Your goal on this day is to inject various failures into your array while it is running the baseline mixed workloads to determine what, if any, performance impacts there are when failures occur. Run these tests using the same four different baseline mixed workloads that you used in Phase 1 testing, performing the fault injection testing while they are running.

To begin testing, start a baseline mixed workload and allow it to run until you know steady state has been reached. Then begin fault injection testing. For components that are online replaceable, you should fail that component, remove it and replace it with a separate component, and then monitor how long it takes the array to recover to steady state. You should plan to perform multiple fault injection tests per baseline mixed workload run, waiting until the array recovers to steady state before moving to the next simulated failure. In some cases (like with a controller failure), the new steady state may be at a lower performance level. Take note of any performance impacts. Each baseline mixed workload test run may take between one and two hours to simulate all faults. Here are the scenarios you should test:

- Fail a path
- Fail a drive (or flash module)
- If the array claims to support it, fail two drives (or flash modules) in the same RAID group simultaneously
- Fail a controller
- Fail a power supply (and fail a fan separately if the array supports that)
- Perform a controller firmware upgrade
- Expand capacity by adding a shelf online

If you can set up and use the array vendor's remote support capability during these tests, do so to test that on at least one of the runs. Let them know up front if you plan to do this though, so they will understand why they are seeing so many different failures in quick succession on one array. All vendor products may not be able to support all suggested tests, but if they do not make sure that whatever level of nondisruptive operations and overall availability the array supports, you are comfortable that it can meet your production availability requirements. Although it is not suggested as a fault injection test, you may also want to ask about drive firmware upgrades.

Phase 5: Soak Test

Other than array preconditioning, the majority of tests will run for several hours. While this is substantially longer than traditional HDD-based testing, it is short for AFA testing. Plan to run the four baseline mixed workloads you used in Phase 1, without any functional or fault injection testing, consecutively over the weekend. Target to run them statically at 85% of your array's performance saturation point, but do not scale them up to the performance saturation point as you did in ramp testing. This is effectively your soak test. Start with the first test right before you leave work on Friday, and have the tests run in succession for 12 hours each for a total testing time of 48 consecutive hours. When you return on Monday morning, record and analyze the results to ensure that performance was consistent and no anomalies were recorded.

Interpreting the Test Results

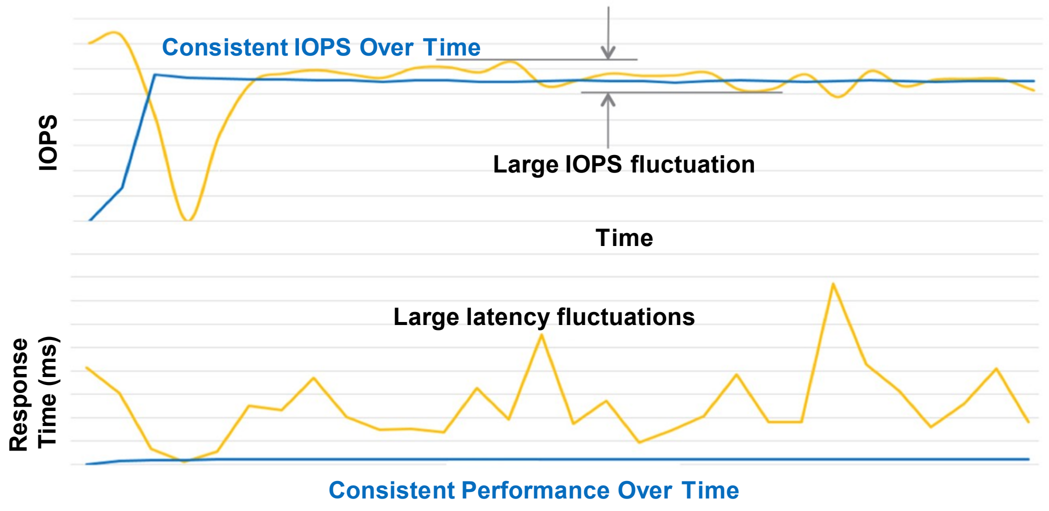
As you start to view and interpret the test results, refer to your original test objectives. Given flash-based arrays' emphasis on performance, it is likely you are at least looking for consistent sub-millisecond latencies up to the array performance saturation point for a select group of high-priority applications. Or you may be looking for consistently lower latencies. You are also looking for the array's ability to consistently deliver the IOPS and throughput that your mixed workload requires. Ideally, you want to see latency staying the same or increasing only very slightly – still staying under your target – as load (IOPS) increases, and then at the performance saturation point the latency increasing exponentially (refer back to Figure 1).

What you don't want to see is unpredictable latency as load increases. Figure 2 shows an example of that. As load increases, garbage collection periodically kicks in, and write latencies may spike during that period. When you first start to see significantly increased latencies, you may think this is the performance saturation point, but then as you continue to increase load, the latencies return to the target range. This is why it is important that you continue to increment load until you are absolutely sure that you have passed the performance saturation point.

While Figure 2 does not include specific numbers to quantify the variableness, understand what your performance requirements are. Small variations around a target IOPS or latency may still be acceptable, but smoother, more predictable performance is better. You'll need to be your own judge on how much variability is acceptable.

FIGURE 2

Example of Good and Bad AFA Performance Testing Results



Source: IDC, 2014

If you are not using and testing QoS controls, you want to see a smooth curve that looks very similar to the real-world AFA performance graph (refer back to Figure 1). If you are using QoS, you want that same curve at the system level but will want to delve deeper to understand whether or not all volumes are receiving the specified number of IOPS and are getting those IOPS with consistent latencies that meet your target requirements.

If your business is growing, then you will have some growth projections that drive future performance and capacity planning. Note how much of an array's maximum performance is required to run your current workload, and how much room for growth it provides. Are you comfortable that the system has sufficient room for expansion to continue to meet your requirements over its depreciation life cycle?

FUTURE OUTLOOK

Flash will result in enterprise storage, as well as application architectures, undergoing a significant transformation over the next five to seven years. During this period, we will see the performance of the new "commodity" storage array (which will heavily leverage flash) go up significantly, while the TCO will drop, relative to traditional HDD-based solutions. IDC expects the AFA market to grow to over $3 billion by 2018, while the HFA market will surpass $16 billion. Expect the following over the next few years:

- **Increasing flash optimization.** Flash optimization of both AFAs and HFAs will continue at a rapid pace, narrowing the performance gap between the two. For application environments needing consistent sub-500 microsecond latencies, however, AFAs will retain their distinct advantage.
- **Application architectures.** Within the next 12–18 months, we will start to see applications come on the market that have been specifically designed around the unique performance characteristics of flash media, further increasing the performance gap between 3rd Platform applications run on flash and client/server applications running on HDD-based arrays. These applications will perform best on AFAs.
- **Evolution of flash testing and tools.** Testing methodologies will evolve to accommodate the storage efficiency technologies that will become checkbox items for flash-based arrays marketed to the datacenter (repeat pattern removal [tied to the thin-provisioning feature], compression, deduplication, and ultimately virtual machine–aware storage management [enabled by hypervisor APIs like VMware's VVOLs and Microsoft's ODX]). Freeware tools will evolve to accommodate these common characteristics of 3rd Platform data sets.
- **Flash testing expertise.** As flash-based arrays become more popular, the knowledge concerning how to accurately test them will become much more widespread. Within 2015, standards bodies such as SNIA will have produced a set of flash-based array testing guidelines.
- **Preconditioning.** By the end of 2015, we will start to see flash-based array vendors ship a canned array of preconditioning tests to help simplify the initial installation process. These tools will leverage the array's system-level awareness of flash cell state to shorten the preconditioning process while making it slightly more efficient (each cell will be written to once and only once).

ESSENTIAL GUIDANCE

If you are a medium-sized to large enterprise, it is very likely that as you consider refreshing your primary storage array, you will consider flash-based arrays. As flash-based arrays evolve over the next five to seven years to become the enterprise storage workhorses for primary storage environments, storage administrators need to understand how to effectively test them. Items to keep in mind include:

- **Create a realistic test plan.** Understand the I/O profile of the workload you plan to run on the flash-based array, and plan to test it as closely to how you will use it in production as possible. Take varying read/write ratios, mixed block sizes, a heavy random I/O skew, compressible and dedupable data, and spatial and temporal locality (I/O banding with drift) into account. Install and configure the array so that it meets any infrastructure standards in your shop that will drive how it is used in production. Test your production workflows and understand how the array behaves in failure modes. Don't waste time doing "hero tests."

If you are only considering a flash-based array to run a single application, you are selling the technology short. Mixed workload consolidation is the future of flash-based arrays, and you will achieve the best TCO with these solutions when they are used in that manner. Running multiple applications, flash-based arrays deliver a TCO that HDD-based arrays cannot touch. Given that, you should strongly consider a flash-based array for mixed workload consolidation, and if you are going to do that, you will need to test it in that kind of environment.

- **Execute to the test plan.** If you are doing a bake-off with multiple arrays, ensure that you are testing all of these arrays consistently so that you can accurately compare results. Performance when inline data reduction is enabled and disabled is very different, and the same may be true with encryption and other data services. Having a written test makes this easier.
- **Perform preconditioning.** Ensure that you have transitioned the flash-based array from FOB to steady state prior to beginning any tests that you will use to evaluate its performance on your workload. Performance between an FOB flash-based array and a preconditioned one can vary by as much as 80%.
- **Know the performance saturation point.** Scale your baseline mixed workloads up to and beyond the performance saturation point to understand not only the capability of the array running your workload but also what failure looks like. This helps with capacity planning and avoids any unexpected surprises.
- **Perform a soak test.** Today's production systems run 24 x 7 with consistent load (although there is a certain amount of cyclicality in terms of read/write intensity). Perform a soak test of at least 48 hours to understand how the array will behave under longer-term testing with consistent workload pressure.
- **Evaluate test results in light of your objectives.** Define "success" before you start your testing in terms of latencies, overall response time, and throughput. Know what features you expect the array to provide in terms of data services, scalability, and reliability. Look for test results that show consistent, predictable performance as workload increases up to the performance saturation point. And evaluate the array's ability to scale to accommodate your business growth over your anticipated depreciation life cycle.

LEARN MORE

Related Research

- *IDC's Worldwide Flash Storage Solutions in the Datacenter Taxonomy, 2014* (IDC #250560, September 2014)
- *Executive Interviews on Flash Array Deployments in the Enterprise* (IDC #249884, July 2014)
- *Flash-Optimized Storage Architectures* (IDC #249295, June 2014)
- *Violin Memory: Laying the Foundation for the All-Silicon Datacenter* (IDC #249324, June 2014)
- *Drivers of Enterprise Storage Purchases* (IDC #248165, April 2014)

- *IDC Worldwide Storage Predictions 2014: Storage Disruption — Flash, Cloud, and Software-Based Storage* (IDC #WC20140109, January 2014)

Appendix: Solid State Standards Associations

Storage Networking Industry Association

Storage Networking Industry Association (SNIA) has adopted the role of industry catalyst for the development of storage solution specifications and technologies, global standards, and storage education. SNIA, a nonprofit trade association, has over 400 members dedicated to developing and promoting standards, technologies, and educational services to empower organizations in the management of information. The SNIA Solid State Storage System Technical Working Group (S4 TWG) focuses on flash storage, and among other goals is working toward defining a set of vendor-neutral flash array testing guidelines. The S4 TWG can be found at members.snia.org/apps/org/workgroup/s4twg.

JEDEC Solid State Technology Association

JEDEC, formerly known as the Joint Electron Devices Engineering Council, is a global leader in developing open standards for the microelectronics industry, bringing together manufacturers and suppliers to create standards to meet the diverse technical and developmental needs of the industry. JEDEC's official Web site is www.jedec.org. Two JEDEC documents may be of interest to administrators working on flash-based array testing projects:

- *Solid State Drive (SSD) Requirements and Endurance Test Method* (JESD218, September 2010)
- *Solid State Drive (SSD) Endurance Workloads* (JESD219, September 2010)

Synopsis

This IDC study provides a framework for end users to test flash-based arrays. It is primarily designed for use with AFAs, all-flash configurations of HFAs, and highly flash-optimized HFAs. IDC performed substantial research to understand and document the unique performance characteristics of flash-based arrays, evaluated various storage performance and other testing tools, and worked with array vendors as well as independent testing labs. The result is a proposed test plan that will allow end users to accurately characterize how flash-based arrays will perform in their environments against their workloads.

"Flash-based arrays exhibit unique performance characteristics that are very different from HDD-based storage architectures and are used with workloads that are very different from those of the recent past," said Eric Burgener, research director, IDC Storage Systems. "As they execute test plans with these arrays, these differences need to be taken into account in order to be relevant for real-world 3rd Platform workloads."

About IDC

International Data Corporation (IDC) is the premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications and consumer technology markets. IDC helps IT professionals, business executives, and the investment community make fact-based decisions on technology purchases and business strategy. More than 1,100 IDC analysts provide global, regional, and local expertise on technology and industry opportunities and trends in over 110 countries worldwide. For 50 years, IDC has provided strategic insights to help our clients achieve their key business objectives. IDC is a subsidiary of IDG, the world's leading technology media, research, and events company.

Global Headquarters

5 Speen Street
Framingham, MA 01701
USA
508.872.8200
Twitter: @IDC
idc-insights-community.com
www.idc.com