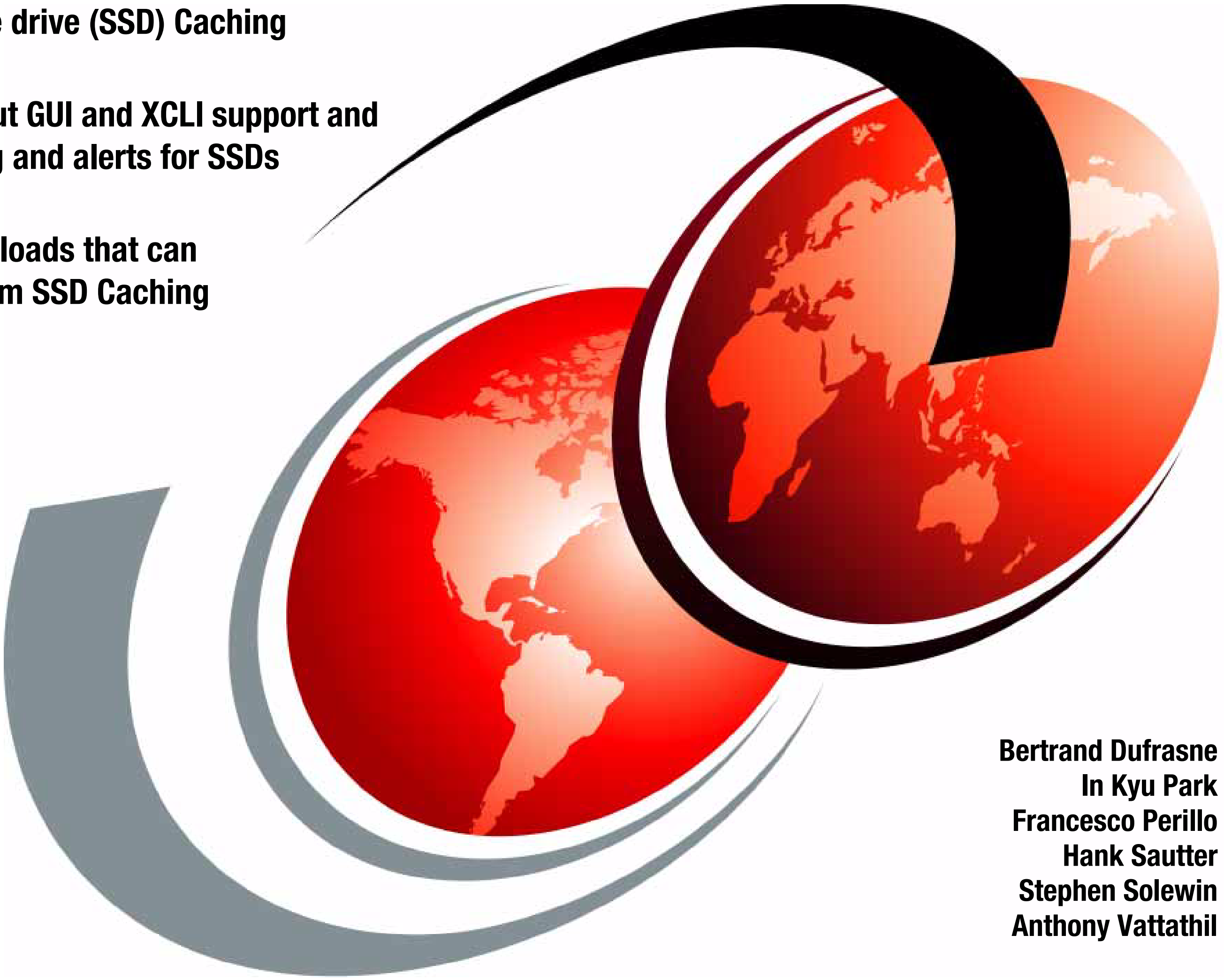
Solid-State Drive Caching in the IBM XIV Storage System

Understand the architecture of solid-state drive (SSD) Caching

Learn about GUI and XCLI support and monitoring and alerts for SSDs

View workloads that can benefit from SSD Caching



Bertrand Dufrasne
In Kyu Park
Francesco Perillo
Hank Sautter
Stephen Solewin
Anthony Vattathil

Redpaper

International Technical Support Organization

Solid-State Drive Caching in the IBM XIV Storage System

May 2012

REDP-4842-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (May 2012)

This edition applies to the IBM XIV Storage System Gen3 with software Version 11.1.x.

Contents

Notices

Trademarks

Preface

This IBM® Redpaper™ publication provides information about the implementation and use of solid-state drives (SSDs) in IBM XIV® Storage System XIV Generation 3 (Gen3), running XIV software version 11.1.0 or later. In the XIV system, SSDs are used to increase the read cache capacity of the existing DRAM memory cache, and are not used for persistent storage.

This paper begins with a high-level overview of the SSD implementation in XIV and a brief review of the SSD technology, with focus on the XIV system. It explains the SSD Caching design and implementation in XIV. Then it examines typical workloads that can benefit from the SSD Caching extension and introduces the tools and utilities to help you analyze and understand the workload. In particular, it highlights the block tracing facility that was designed and developed by IBM Research.

Then this paper explains the process that authorized IBM services representatives use to install SSD Caching. It reviews the changes made to the XIV GUI and the XCLI to support SSD Caching. Finally this paper provides a listing of the new alert-generating events and monitoring options that are provided for SSD support.

This paper is intended for users who want an insight into the XIV SSD Caching implementation and architecture, its capabilities, and usage. For more information about the IBM XIV Storage System, see the IBM Redbooks® publication, *IBM XIV Storage System: Architecture, Implementation, and Usage*, SG24-7659.

The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO) in San Jose, CA.

Bertrand Dufrasne is an IBM Certified Consulting IT Specialist and Project Leader for IBM System Storage® disk products at the ITOS in San Jose, CA. He has worked for IBM in various IT areas. He has authored many IBM Redbooks publications and has developed and taught technical workshops. Before joining the ITSO, he worked for IBM Global Services as an Application Architect. He holds a master degree in electrical engineering.

In Kyu Park is a Storage Technical Advisor and IT Specialist with the Advanced Technical Skills (ATS) team in for the IBM Growth Market Unit (GMU). He has 5 years of experience as a Storage Field Technical Sales Specialist in South Korea and covers storage products for the of financial and industrial sectors. In August 2011, he transferred to IBM China and joined the GMU ATS team. He now focuses on IBM strategic storage products, including the IBM XIV, IBM Storwize® V7000, and IBM SAN Volume Controller.

Francesco Perillo is an IT Storage Specialist in IBM Italy. He has over 6 years of experience in working on EMC and IBM Enterprise Storage and has been involved with the planning, design, implementation, management, and problem analysis of storage solutions. His areas of expertise include the SAN infrastructure, enterprise disk storage, and IBM storage software. Francesco received an engineering degree from Rome Tor Vergata University and holds two technical storage certifications from EMC.

Hank Sautter is a Consulting IT Specialist with Advanced Technical Support for IBM US and has been with IBM for 33 years. He has 20 years of experience with IBM S/390® and IBM disk storage hardware and Advanced Copy Services functions. His previous IBM experience includes working on IBM Processor microcode development and S/390 system testing. Hank's areas of expertise include enterprise storage performance and disaster recovery implementation for large systems and open systems, which he regularly writes and presents on. He holds a Bachelor of Science (BS) degree in physics.

Stephen Solewin is an XIV Corporate Solutions Architect for IBM in Tucson, Arizona. He has 16 years of experience working on IBM storage, including Enterprise and Midrange Disk, LTO drives and libraries, SAN, storage virtualization, and storage software. Steve has been working on the XIV product line since March of 2008. He holds a BS degree in electrical engineering from the University of Arizona, where he graduated with honors.

Anthony Vattathil is a IBM Corporate Architect in California. He has over 12 years of experience in the storage field. His areas of expertise include large scale SAN design and technical knowledge of various storage systems. Tony is also well versed on various Linux and UNIX system platforms. He has written extensively on storage performance and is one of the performance leads for the XIV system.

Thanks to the following people for their contributions to this project:

- For their contribution and support:

  Shimon Ben-David
  Brian Carmody
  Orli Gan
  Haim Helman
  Aviad Offer
  Ohad Rodeh
  Jim Sedgwick
  Yijie Zhang
  IBM Corporation

- For access to equipment and resources

  Rachel Mikolajewski
  George Thomas
  Peter Wendler
  IBM System Level Test Lab in Tucson, AZ

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review 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

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Executive summary

Solid-state drives (SSDs) are still a relatively new addition to storage systems. SSDs are typically used to greatly reduce response time and increase random read input/output (I/O) operations per second (IOPS) when compared to traditional hard disk drives (HDD).

The IBM XIV Storage System Gen3, with software version 11.1, supports the use of SSDs. However, in the XIV system, SSDs are used to increase the read cache capacity of the existing DRAM cache. They are not used for persistent storage, which departs from the approach used by many other storage systems that use SSDs as another, faster storage tier.

Important: In XIV Gen3, SSDs are used solely as an extension of the system read cache.

XIV Gen3 uses 400-GB multilevel cell (MLC) SSDs, increasing the total cache by 6 TB for a fully populated system. SSD Caching in IBM XIV specifically targets I/O-intensive, online transaction processing (OLTP) workloads. With its sophisticated cache algorithms and use of the XIV massive internal bandwidth, this extended cache can be used to accelerate target workload performance from two to three times faster. With certain workloads, it can accelerate target workload performance up to six times faster.

The caching algorithm is embedded in the XIV system software (firmware or microcode) and makes the integration of SSDs transparent to the user or storage administrator. No tuning of the cache is required to achieve the potential performance boost.

Every XIV Gen3 that ships can take advantage of a concurrent code upgrade that enables the use of the SSD extended cache. SSD extended cache is available for all Gen3 systems, both fully and partially populated. Every module in the XIV must be upgraded with the extended cache.

For more information about IBM XIV Storage System Gen3 with software release 11.1, see the announcement letter at:

http://www.ibm.com/common/ssi/rep_ca/1/897/ENUS112-031/ENUS112-031.PDF

For more information about XIV, see the XIV page at:

http://www.ibm.com/systems/storage/disk/xiv/

2

Technology overview of solid-state drives

In storage environments today, the cost of a storage system is as much a concern as the performance that it delivers. Starting with its first generation, the IBM XIV Storage System and its software work with off-the-shelf (therefore, lower cost) components. The need for controlling costs is the reason that the XIV Gen3 software was designed to accommodate the less expensive multilevel cell (MLC) solid-state drive (SSD) technology. The integration of SSDs in XIV increases random read performance and maintains enterprise-level reliability and resiliency. As explained in this chapter, the XIV system software prevents some shortcomings of the MLC SSD technology, which render the XIV system independent of the specifics of any particular SSD vendor.

This chapter provides high-level overview of the SSD technology, strictly focusing on aspects that are relevant to the use of SSDs in the XIV system. This chapter includes the following sections:

- Overview of solid-state drives
- Endurance of solid-state drives
- Solid-state drive type in XIV Gen3

2.1 Overview of solid-state drives

SSDs use semiconductor devices (solid-state memory) to store data. Unlike traditional hard disk drives (HDD), an SSD does not have a mechanical arm to read and write data. An SSD does not have any moving parts.

Currently, most of SSDs are based on NAND flash chips. NAND flash chips are fast, highly reliable, widely available, and are nonvolatile, meaning they can store and retain data even without a power source.

A flash memory stores the information in an array of flash cells. Each cell consists of a single floating-gate transistor (see Figure 2-1) that can store electrons. The cell is programmed by increasing and decreasing the amount of charge placed on it.

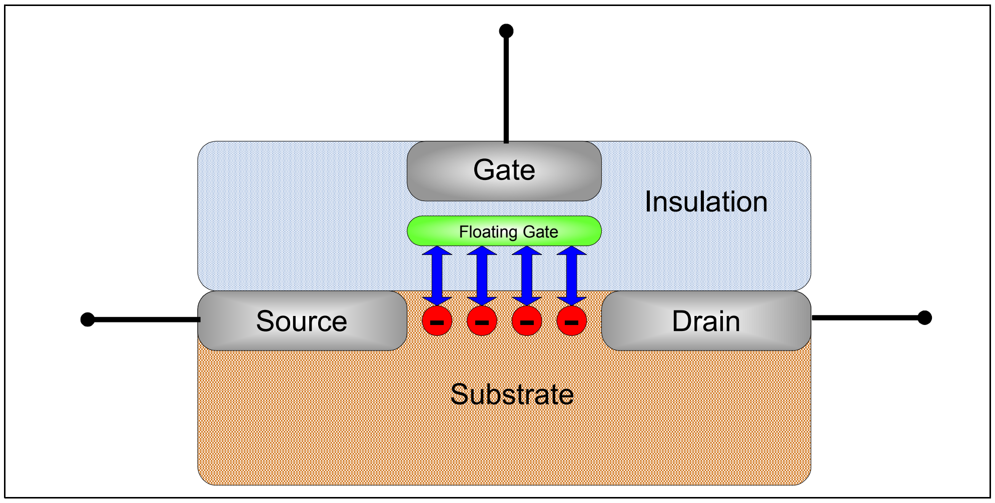


Figure 2-1 Flash transistor cell

Two types of NAND flash memory architectures are available as explained in the following sections.

Single-level cell NAND flash

Single-level cell flash (SLC NAND flash) memory stores 1 bit of data per memory cell. Therefore, a single voltage level is required for the device to detect one of the two possible states of the memory. If a current is detected, the bit is stored as 0, meaning *programmed*). If no current is detected, the bit is 1, meaning *erased*. Thanks to this rather simple architecture, SLC NAND flash can offer quick read and write capabilities, long-term durability, and rather simple error correction algorithms.

Besides the limited programming density, SLC NAND flash has one other disadvantage. That is, it is considerably more expensive per bit when compared to other NAND technologies. The higher cost is due to each cell storing only 1 bit of data. However, SLC NAND flash is the ideal choice when performance is the driver and cost is not an issue.

Multilevel cell NAND flash

Multilevel cell flash (MLC NAND flash) memory can store at least 2 bits of data per memory cell. Therefore, multiple voltage levels are necessary to decipher between the four possible states of memory.

Each memory cell can have the following range of states:

00	Fully programmed
01	Partially programmed
10	Partially erased
11	Fully erased

Two advantages of this increase in bits per cell are that the memory capacity is greater and production costs are reduced. As a result, MLC NAND flash can be sold at a much lower price per bit than SLC NAND flash, and it offers twice the density. However the MLC NAND flash is slower and has a shorter lifetime than the SLC NAND flash.

Important: The XIV Storage System Gen3 uses MLC SSDs.

2.2 Endurance of solid-state drives

The life of a flash is limited by the number of write/erase operations that can be performed on the flash. To extend the lifetime of a drive and to ensure data integrity on the drive, SSDs use a built-in dynamic or static *wear-leveling algorithm* and *bad-block mapping algorithm*. These algorithms are further complemented by the *over-provisioning algorithm*, the *error detection code algorithm*, and the *error correction code algorithm* to ensure data reliability.

These algorithms are implemented in various electronic components of the drive:

- Wear-leveling algorithm

  The goal of the wear-leveling algorithm is to ensure that the same memory blocks are not overwritten too often. With this mechanism, the flash controller distributes the erase and write cycles across all the flash memory blocks.

- Bad-block algorithm

  The bad-block algorithm detects faulty blocks during operations and flags them. These flagged blocks are excluded from the rotation and replaced with good blocks, so that the data does not go into bad blocks.

- Over-provisioning algorithm

  In combination with the wear-leveling algorithm, the over-provisioning algorithm is used to further alleviate write-endurance issues. The over-provisioning algorithm minimizes the amount of write amplification needed. A flash memory must be erased before it can be rewritten, and a whole erase block (512 KB - 2 MB) must be erased in a single erase operation. Therefore, typically more physical space is used than required by the amount of logical data to be written, which is known as *write amplification*. *Over-provisioning* is the difference between the physical capacity of the flash memory and the logical capacity that detected by an application or host. The SSD has more capacity than is usable.

- Error detection code and error correction code algorithm

  The error detection code and error correction code algorithms maintain data reliability by allowing single-bit or multiple-bit corrections to the data that is stored. If the data is corrupted due to aging or during the programming process, the error detection code and error correction code algorithms compensate for the errors to ensure the delivery of accurate data to the host application.

2.3 Solid-state drive type in XIV Gen3

XIV Gen3 uses enterprise-quality MLC NAND flash SSDs. As indicated previously, the MLC NAND flash is slower and has a shorter lifetime than the SLC NAND flash. These disadvantages can be alleviated by sophisticated algorithms, which is the approach taken by XIV.

In addition to the inherent techniques to the SSD and that are listed 2.2, "Endurance of solid-state drives" on page 5, XIV Gen3 with software version 11.1 implements patented algorithms. These algorithms can, for example, restrict direct host access to SSD operations, such as dealing with logical erase blocks (LEBs) when erasing data. (An *erase block* is the smallest unit that a NAND flash can erase.)

The XIV software controls writes to the SSD cache and avoids overwriting a particular spot on the SSD too often. Writing to the SSD cache is sequential. Data writing starts at the beginning of the SSD (first memory page) and then progresses sequentially to the end, without skipping or jumping, as illustrated in Figure 2-2. Each sequential write consists of a 512K block. When the end of the SSD is reached, a new cycle begins, overwriting the previous data, again from beginning to end. This sequential and cyclic writing allows for uniform use of the SSD cells, maximizing the SSD lifetime.

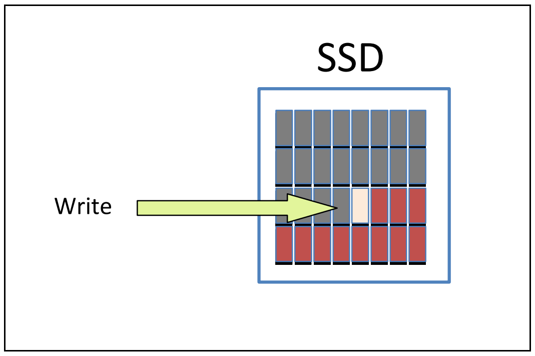


Figure 2-2 SSD written to sequentially

Furthermore, because XIV does only 512K sequential writes to SSD or 4K random reads, it gets a consistent and predictable response time from those SSD operations. XIV understands the performance to expect from the SSDs at any time. This feature is another advantage over systems that do any combination of random reads and writes and sequential reads and writes to SSD, making it impossible for the system to have consistent performance characteristics from the SSDs.

Because XIV uses SSD exclusively as an extension of the read-cache layer, all data on the SSD is written to disk. This way, XIV can sustain multiple SSD failures without any implication of data loss.

For information about the SSD Caching algorithm, see Chapter 3, "Caching architecture and implementation" on page 9.

Table 2-1 summarizes the major MLC and SLC differences. The third column shows the functions that implemented in the XIV system software that compensate for reduced tolerance. These functions are also described in this paper.

Table 2-1 SLC versus MLC

Operation	**SLC**	**MLC**	**XIV software features added in v11.1.0**
Write cycles	Higher	Lower	Log structured write (XIV software wear leveling)
Read speed	Higher	Lower	Limits SSD use to only random operations (High-bandwidth workloads are handled by the SAS disk layer.)
Density	Lower	Higher	Reduces costs and increases usable footprints

3

Caching architecture and implementation

SSD Caching is a deterministic cache layer that can adapt to mixed workloads. Extended SSD Caching in XIV Gen3 specifically targets small block, random read input/output (I/O). XIV relies on its InfiniBand back-end to support low latency writes and high throughput.

This chapter explains the solid-state drive (SSD) Caching architecture of IBM XIV Storage System Gen3. It also explains the implementation of SSD Caching and how it cooperates with core caching to extend its capabilities. This chapter includes the following sections:

- SSD tiering versus caching
- XIV versus traditional storage architectures

3.1 SSD tiering versus caching

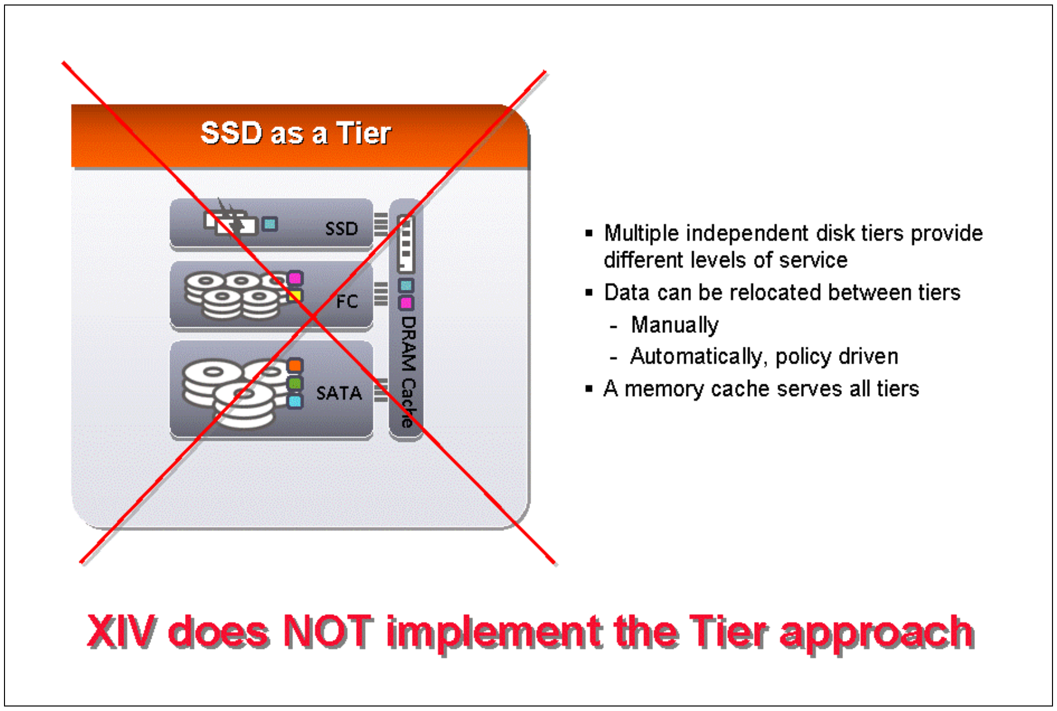
When SSD was first adopted by storage system vendors, they mostly used it as another, faster storage tier. XIV Gen3 departs from that approach by using SSDs exclusively as an extension of the system read cache.

Tiering

Tiering is traditionally how SSDs are integrated in storage systems. With the tiering approach (Figure 3-1), additional storage software intelligence is necessary to unite independent tiers and manage the movement of data between the tiers. This approach can often add cost and complexity.

Important: SSD is not used as a tier in XIV.

Tiering is driven by policies that determine when to move data to a disk type. If your environment is fairly static, this approach can work well. You must also consider the time it takes to relocate data in and out of a tier. If data cannot be relocated quickly, the tiered storage approach is not effective.



- Multiple independent disk tiers provide different levels of service
- Data can be relocated between tiers
  - Manually
  - Automatically, policy driven
- A memory cache serves all tiers

Figure 3-1 SSD used as a storage tier

Caching

By using the SSD as an extension of the storage system cache, the workload benefits from SSD technology in real time. With SSD Caching (illustrated in Figure 3-2), which is the XIV approach, you do not need to relocate data. Furthermore, with XIV, SSDs are used as read cache only. When data in cache is no longer relevant, it can be dropped and replaced by more relevant data.

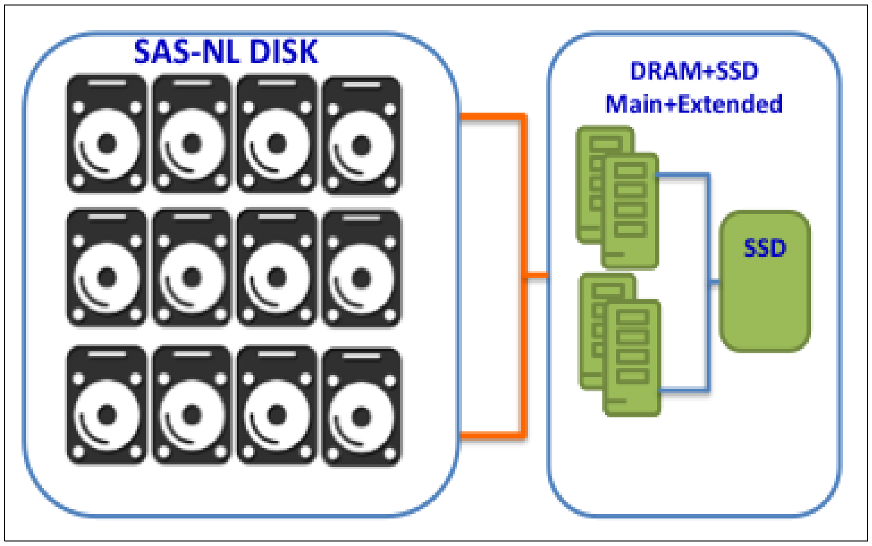


Figure 3-2 SSD used as a cache extension

With the caching approach, the SSD can be used effectively in highly dynamic environments where data patterns are constantly changing. For a holistic solution, the caching architecture must also deliver good write performance.

Important: XIV implements SSDs as an extended read cache.

This unique approach to handling SSD is possible on XIV thanks to the high efficiency of its Peripheral Component Interconnect Express (PCIe) buses between SSD and memory or processor. With this approach, XIV can take full advantage of SSD performance, but on traditional storage, the SSD might saturate the buses, memory, and processor.

3.2 XIV versus traditional storage architectures

Traditional storage systems require the storage administrator to address most of the following questions:

- Do we optimize our system for I/O per second (IOPS) or bandwidth?
- How do we lay out our storage?
- Do we optimize for writes or reads, or do we optimize for capacity?
- How do we cope with different workload characteristics or workloads that shifts from random to sequential. Specifically, how do we handle online transaction processing (OLTP) sessions with random reads versus backups with large block sequential bandwidth requirements?

Consideration of storage tiering as part of the solution then leads to another set of questions:

- What tiering policy should we implement?
- Can we move our large data set to SATA drives in time to service our high bandwidth reads?
- Can we move the relevant blocks back to SSD to facilitate our production workload, which is a random OLTP workload?
- How much SSD capacity do we need?
- Do we have enough budget for a large SSD tier?

The XIV architecture obsoletes all these considerations, providing a solution that you can deploy quickly, with minimal planning.

From inception, the XIV design principles have dictated the use of low-cost components coupled with virtualized and flexible system software (firmware), allowing the integration of emerging technologies. The integration of SSD at the caching layer is another example of adhering to those XIV core values.

Traditional systems with a global cache that must support multiple disk layouts (RAID levels) and disk types (SSD, Fibre Channel (FC), SAS, SATA) make caching less nimble and less aggressive. To support the many variations, the cache must make concessions. For example, optimizing on RAID 6 might have implications on how much read cache is available to do a prefetch on RAID 1.

Consider an example of an application with a highly random OLTP workload that runs from 8 a.m. to 8 p.m. and uses a database. The size of the database is constantly growing and growing quickly. User access patterns vary throughout the day and directly affect which volumes or regions of the databases are most active. In this situation, tiering cannot be proactive enough to move the corresponding blocks in a real-time manner. By the time relevant blocks are moved to SSD, the hot region is elsewhere. To complicate matters more, the same database might be backed up by using a multithreaded copy that shifts the workload characteristic of a large block sequential read.

A storage system equipped with many nearline SAS (NL-SAS) delivers high throughput. Conversely, traditional NL-SAS RAID groups do not perform well for small-block random I/O. An option is to keep the entire database on SSD, but this option is too costly.

In such situations, the XIV architecture helps in the following ways:

- XIV has a predefined, immutable disk configuration that enables the implementation of a single, optimized caching algorithm.
- XIV has a selective read-cache algorithm that tunes itself to reduce random access to the disk layer or forwards bandwidth-intensive I/O operations to the disk layer. XIV can dynamically tune its cache for random and sequential workloads.

The storage administrator must be concerned with only the capacity allocation. After the storage administrator assigns storage to the host, the XIV system software naturally optimizes how to service the host I/Os.

3.2.1 Overview of SSD Caching

SSDs are integrated into the XIV system as an extension of the dynamic random access memory (DRAM) primary cache. Each module has 24 GB of DRAM cache and 400 GB of SSD cache. In a 15-module configuration, the caching layer can be expanded by 6 TB of usable read cache.

XIV SSD Caching is implemented at the lowest level of the caching mechanism, just before the I/O is issued to the disk.

Tip: Cache data movement is inherent to the architecture of the XIV. The caching layer is extended naturally by the introduction of the SSDs.

Figure 3-3 illustrates a cache implementation in XIV.

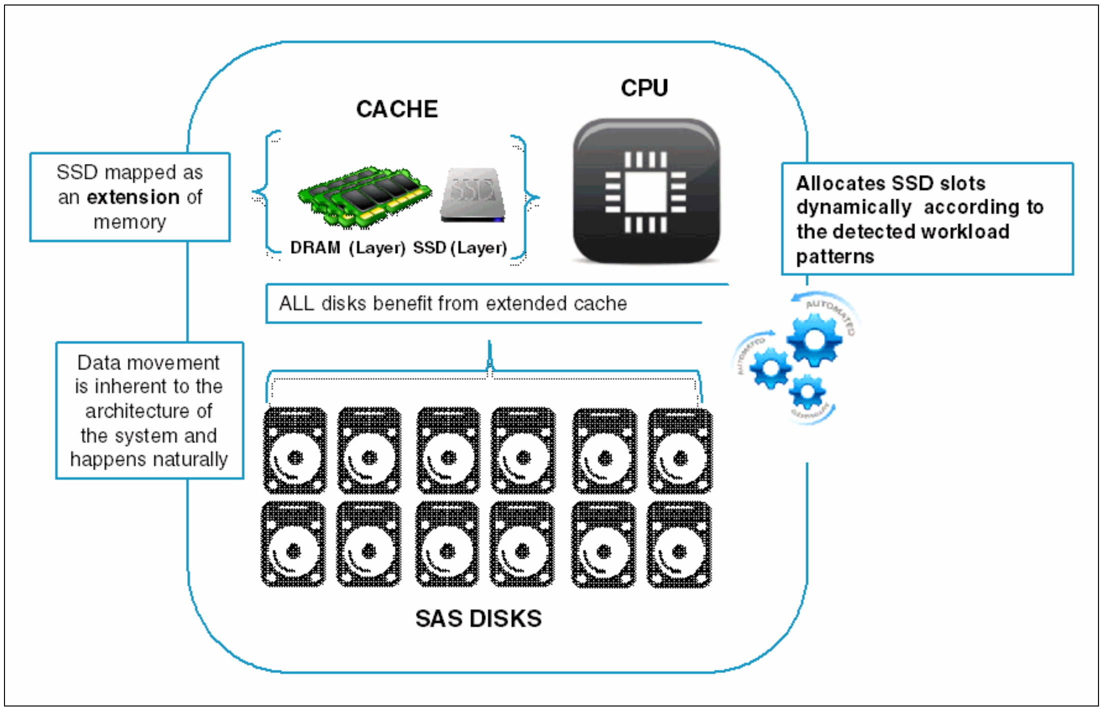

CACHE
CPU
SSD mapped as an **extension** of memory
DRAM (Layer) SSD (Layer)
Allocates SSD slots dynamically according to the detected workload patterns
ALL disks benefit from extended cache
Data movement is inherent to the architecture of the system and happens naturally
SAS DISKS

Figure 3-3 XIV SSD extended caching architecture

The SSD extended cache is targeted exclusively at improving random read performance. The XIV system software (firmware) does not allow a host write request to go directly to an SSD cache. All write I/Os are staged and mirrored in the main cache only (DRAM).

Important: SSDs are used exclusively as a cache extension for random read I/Os. Host writes are *not* cached in SSDs.

When a host sends a write request to the system, the XIV system responds in the following way:

1. Any of the interface modules that are connected to the host can service the request.
2. The interface modules use the system configuration information (metadata) to determine the location of the primary module (data module or interface module) that houses the referenced data. The data is written to the local DRAM cache of the primary module.

3. The primary module uses the system configuration information (metadata) to determine the location of the secondary module that houses the copy of the referenced data. (Again, it can be a data module or an interface module). The data is redundantly written to the local DRAM cache of the secondary module.

Additional information about writes: A write is still acknowledged only after it is committed to the main cache of the secondary cache node. The writes are flushed from the main cache to the disk layer as part of the normal write destage. Destaging to SSD or SAS disks is done asynchronously so that the system does not have to wait for the SSDs or the SAS disks upon a write request. Therefore, when host writes are destaged to the SAS disks, they might be cached in parallel in the SSD cache, to serve future reads (write-through caching).

By using the extended cache for small block random reads only, SSD Caching attempts to greatly reduce the random read access from the disks layer. Therefore, XIV obtains better write performance when read I/Os are served from SSDs, freeing some cycles on the SAS drives.

3.2.2 Selective read cache algorithm for extended cache

Modules (interface or data) are the basic building blocks of the XIV system architecture and are responsible for storing a portion of the overall data set. They manage disk drives and handle all storage features including snapshots, caching, and prefetching.

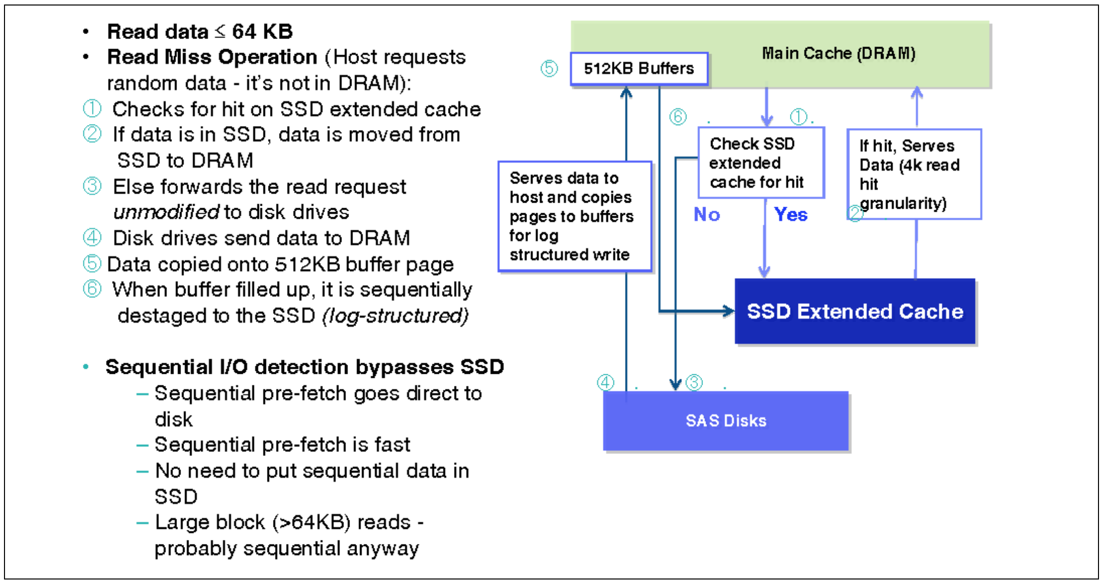
Each module (interface and data) in the XIV system incorporates a cache node. Cache nodes allocate SSD pages when the appropriate workload patterns are detected. Application processing typically generates random I/Os and sequential I/Os. The SSD cache mechanism is triggered only for random reads.

Caching operations are split into two groups: main and extended. The extended cache handles the caching of random read-miss operations of less than 64 KB. All sequential (larger than 64-KB operations) read prefetches are handled in the main DRAM cache. All host writes are handled in the main cache and destaged directly to disk.

For information about internal writes to the SSDs that are inherent to the cache staging mechanism, see “Updates to SSD cached data” on page 17.

3.2.3 Random reads with SSD Caching enabled

An SSD cache map is built as read misses occur in DRAM cache, which is known as *SSD cache learning* and is illustrated in Figure 3-4. The cache node immediately checks the extended cache for the requested I/O. If the requested I/O exists in the extended cache, it is served to the host through the main cache. The I/O operation is now complete and is recorded as an SSD hit.



- **Read data ≤ 64 KB**
- **Read Miss Operation** (Host requests random data - it's not in DRAM):

① Checks for hit on SSD extended cache
② If data is in SSD, data is moved from SSD to DRAM
③ Else forwards the read request *unmodified* to disk drives
④ Disk drives send data to DRAM
⑤ Data copied onto 512KB buffer page
⑥ When buffer filled up, it is sequentially destaged to the SSD *(log-structured)*

- **Sequential I/O detection bypasses SSD**
  - Sequential pre-fetch goes direct to disk
  - Sequential pre-fetch is fast
  - No need to put sequential data in SSD
  - Large block (>64KB) reads - probably sequential anyway

Main Cache (DRAM)
⑤
512KB Buffers
⑥
①
Check SSD extended cache for hit
If hit, Serves Data (4k read hit granularity)
②
Serves data to host and copies pages to buffers for log structured write
No
Yes
SSD Extended Cache
④
③
SAS Disks

Figure 3-4 Cache learning

If the operation results in a true read miss (not in the DRAM cache and not in SSD extended cache), the request is forwarded as unmodified to the disk (SAS layer). The I/O is retrieved from the disk and served to the host by using the main cache. From a host perspective, the I/O operation is now complete and is recorded as a read miss. The related pages are copied to reserved buffers in the main cache.

Important: By design, any read larger than 64k bypasses the SSD cache. Because the ratio of SSD to SAS drives is 1 to 12 in a module, it is more efficient to serve large blocks from the spinning SAS drives given their better overall throughput.

Log-structured write

Two 512-KB buffers are available. When a buffer is full, it is destaged to an SSD as a log structured write, which is illustrated in Figure 3-5.

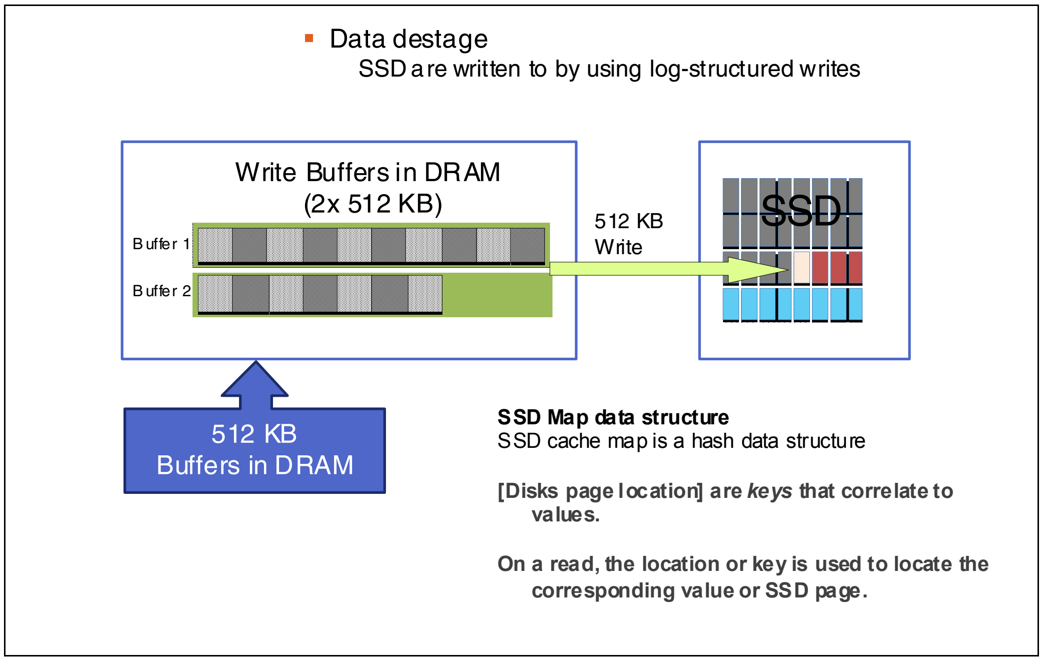

Data destage
SSD are written to by using log-structured writes
Write Buffers in DRAM (2x 512 KB)
Buffer 1
Buffer 2
512 KB Write
SSD
512 KB Buffers in DRAM
SSD Map data structure
SSD cache map is a hash data structure
[Disks page location] are keys that correlate to values.
On a read, the location or key is used to locate the corresponding value or SSD page.

Figure 3-5 Log-structured writes

A *log-structured write* is an IBM XIV patented algorithm that allows buffered I/Os to be destaged sequentially to SSD. This operation minimizes the effects on SSDs of wear-leveling and garbage-collection algorithms.

Random read prefetch

XIV is known for aggressive sequential prefetch capabilities. By isolating a cache node to service only the disks in its respective module, the cache node can be aggressive when doing a prefetch.

This aggressive prefetch capability is extended to random I/Os with the introduction of SSD Caching. When a random read miss is detected, the cache node pulls up to 64 KB from the disk layer and stages it into an SSD. Even a small 4 KB random read miss can trigger a 64-KB prefetch.

The proactive cache builds are then inspected over several cycles. If the additional pages that were pulled into SSD are accessed, they are retained in the SSD cache and marked as *hot*.

Extended cache management

XIV extended cache (layer) is indexed in a memory-resident hash table. XIV reserves 0.3% of the size of the total extended cache capacity to save the hash table. As part of the shutdown process, the contents of the hash table are dumped to the SSDs. Upon startup the data is validated and repopulated into a hash table in memory.

The hash data structure consists of key-value-pair records. In a key value pair, a *key* is the disk page location, and a *value* represents the location where data is stored on the SSD.

On a DRAM read miss, the key is used to determine whether there is an SSD hit. If an SSD hit exists, the value is used to retrieve the data from SSD and satisfy the I/O request.

Updates to SSD cached data

XIV does not allow hosts direct access to the SSD cache. After the initial SSD cache learning occurs, the SSD cache holds a transient copy of the data blocks. When the extended cache must be updated because of a new host write, the corresponding hash table entry is invalidated.

XIV handles the write updates as a normal write operation. During this write operation, the updated main memory pages (DRAM) are copied asynchronously into the SSD buffers and staged for a log-structured write to the SSD. After the I/Os are updated on the SSD, the hash table entry (SSD data map structure) is revalidated, and then subsequent extended cache hits can occur.

This mechanism prevents the extended cache from serving stale data. When the SSD layer is being updated, if a read request for the same block is received, it is serviced from the main DRAM cache. The pages in the main cache are not dropped until the update to the SSD effectively occurs (see Figure 3-6).

Write Operation:

① All writes go to DRAM (Main Cache)

② If SSD has a copy of the data being modified, then the data in SSD is old and stale and a corresponding hash table entry on SSD is invalidated to prevent subsequent hits from the extended cache

③ Writes are de-staged from DRAM to disk normally and asynchronously

④ As writes are de-staged from Main Cache to disk they are asynchronously staged on to the SSD buffer (when less or equal to 64KB)

⑤ Once buffer is full, data is written to SSD and hash table is updated

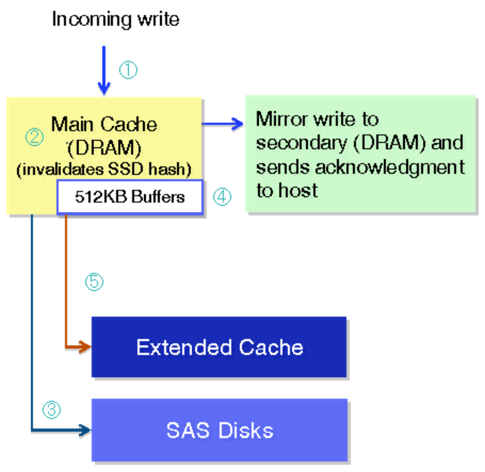

Incoming write
①
② Main Cache (DRAM) (invalidates SSD hash)
512KB Buffers
④
Mirror write to secondary (DRAM) and sends acknowledgment to host
⑤
Extended Cache
③
SAS Disks

Figure 3-6   Cache updates after a disk write

Write performance

To deliver high performance write operations, the XIV system uses the following functions or features that are inherent to its architecture:

- Write operations to XIV volumes always engage all available physical disks (spindles). This core concept of the XIV architecture delivers excellent write performance without generating hot spots.
- A reduced read load on the disks, thanks to SSD Caching, also yields more efficient destaging of dirty data, allowing better write throughput to XIV.
- XIV Gen3 uses InfiniBand as its back-end protocol and can acknowledge writes out of cache three times faster than previous generation.
- XIV uses an intelligent write caching algorithm that allows for small random writes to be combined into larger more spindle friendly writes.

Write performance: By using InfiniBand and intelligent write destage algorithms across all physical disks, XIV Gen3 delivers high performance writes.

Single I/O write performance occurs at DRAM speed because all writes are acknowledged in DRAM. To be precise, writes are acknowledged after they are mirrored safely at the secondary cache node.

3.2.4 SSD failure

If an SSD starts to fail or fails completely, it is phased out similar to any of the other XIV grid components. If the SSD is phased out and not failed, its data is invalidated. If SSD is phased back in, not all data is lost. When an SSD is failed, the system initializes all of the metadata again, so that when the SSD is phased back in, no integrity issues are expected.

Data is not redistributed, which is the case after a module or disk failure. The module with the failed SSD remains the primary module for its partition. However, because of the reduced cache, the module limits its caching duties.

The degraded module continues to serve reads from its DRAM cache and to serve large sequential reads from disk. All small read misses are redirected to the secondary modules, which populate their SSD cache. This behavior distributes the extended caching duties of the module with the failed SSD to all other modules, balancing the use of the remaining extended cache across all modules.

For more information, see also 7.2, “Alerts” on page 53.

3.2.5 SAS disk rebuild

During a rebuild, after a SAS disk failure in a module, the data on an SSD in that module is not invalidated. Rather, it is gradually updated to contain the new data blocks similar to the way that the DRAM does.

4

Workload characteristics

Solid-state drive (SSD) Caching is an optional feature for IBM XIV Storage System Gen3. This chapter highlights the characteristics of workloads that can benefit from SSD Caching. With this information, you can determine whether your critical application workloads can benefit from SSD Caching on the XIV system.

The chapter also highlights the following tools and features that can help in making the determination:

- XIV system statistics
- XIV block tracing facility
- Disk Magic

This chapter includes the following sections:

- Workload for SSD Caching
- Analyzing the workload on XIV

4.1 Workload for SSD Caching

As mentioned previously in this paper, in the XIV Storage System, SSD Caching is implemented as a read cache exclusively and is not used as another storage tier. Moreover, SSD Caching improves small-block random read performance. SSD Caching in the XIV system can improve performance of any workload that is suffering from high latency due to a high amount of small-block random read misses.

Consider a situation where you do not have an XIV system yet, but you want to know whether you might benefit from by extending the XIV with SSD Caching. On most storage systems, you can inspect I/O statistics and determine whether a workload can benefit from SSD Caching. If the storage system has granular reporting, you can take the following steps:

1. Sort your workload by read and write percentages.

   The read portion of the workload can potentially benefit from the SSD Caching. A typical candidate workload for SSD Caching consists of 80% reads and 20% writes.

2. Determine how many of the read requests were misses or rather where the data was not in the cache.

   You can gather this information by looking at the read-miss rate, which determines how many read requests go to disk and are not serviced out of cache. The more read misses you have, the better your workload benefits from SSD Caching.

3. Determine how many of those read requests are small (less than 64 KB) random reads.

   Here, the smaller the read I/O blocks, the better the workload benefits from SSD Caching.

In summary, a workload with I/Os that are small-block random read misses can benefit from SSD Caching in the XIV system.

4.1.1 Workload performance benefits

SSD Caching in the XIV system is quick to respond to changes in workload characteristics and provides increased performance in the following situations:

- I/O operations per second (IOPS) increase of up to three times for an online transaction processing (OLTP) workload
- IOPS increase of up to six times for a highly random workload
- A latency reduction of almost three times for a medical records application

Testing shows that an IOPS increase of six times and 1 ms sublatency is possible when the working data set fits in the SSD cache as illustrated in Figure 4-1.

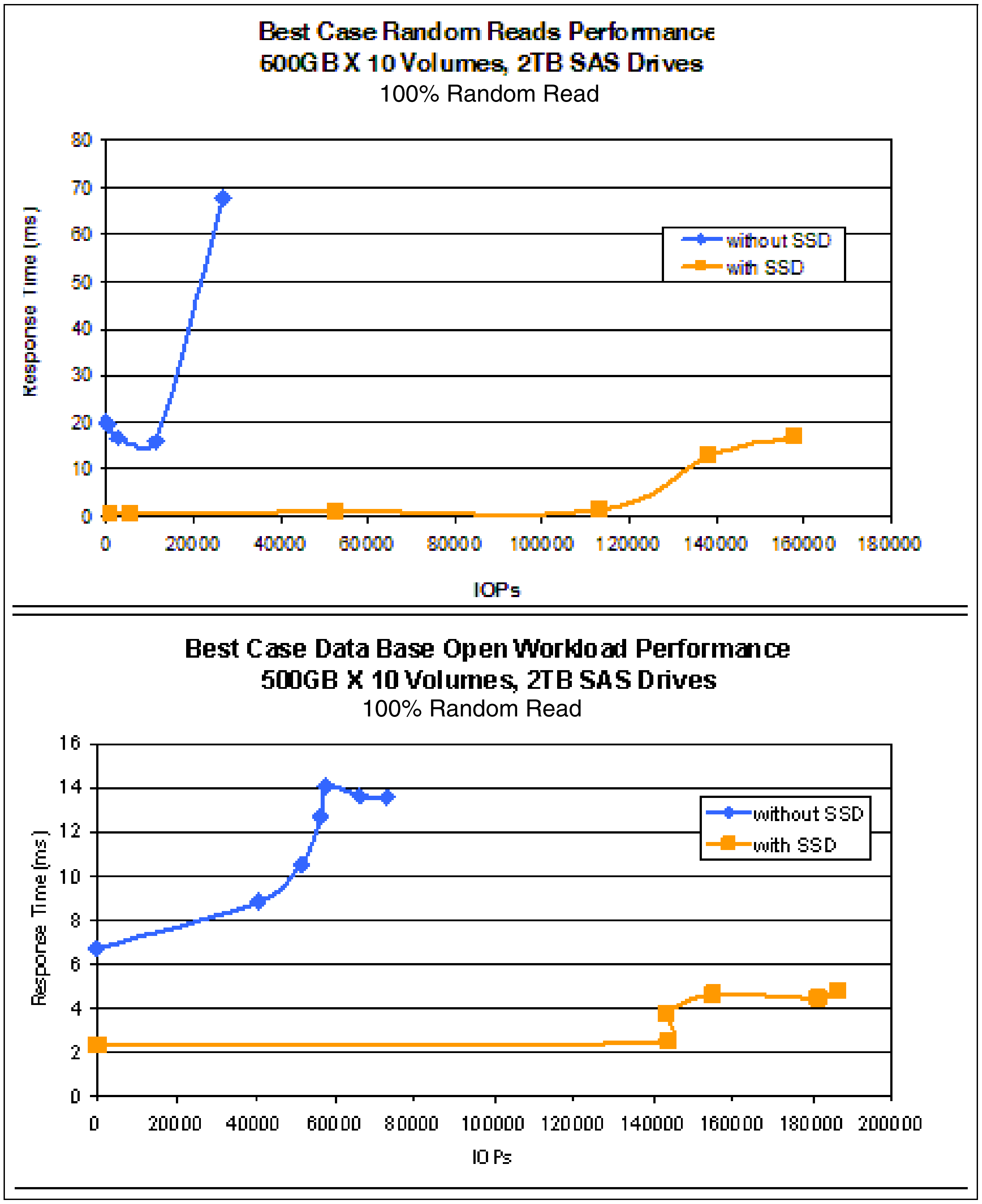


Figure 4-1 Random read performance

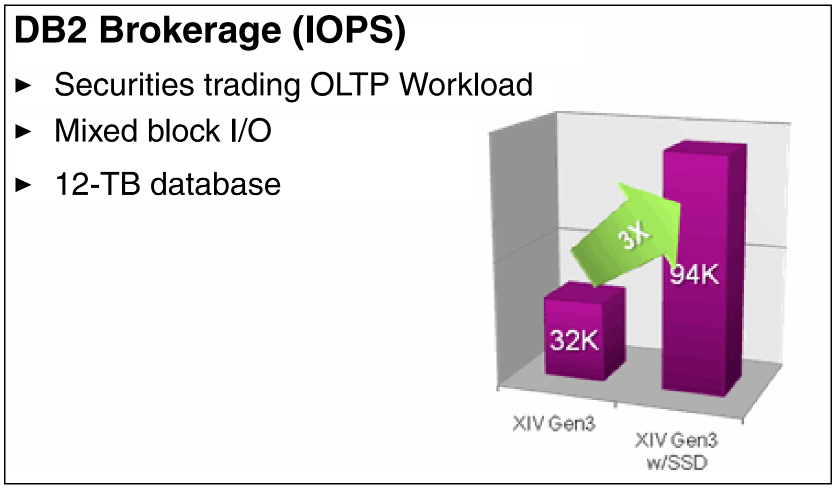
Again, generally, random read activity receives the most benefit from SSD Caching. This optimization happens automatically. However, if needed, it is possible to enable or disable SSD Caching on a per-volume basis. If a specific host requires better performance for a random read workload, the volumes associated with that host can be enabled when other volumes are disabled. This action effectively provides all the SSD cache for the enabled hosts. For more information, see 6.1.2, “Setting SSD Caching” on page 40.

4.1.2 Typical applications and performance benefits

SSD Caching offers performance benefits on several typical applications.

IBM DB2 brokerage applications

Securities trading is an OLTP workload that generates random I/O and benefits from increased IOPs. Figure 4-2 illustrates that SSD Caching for the XIV system was tested with this workload, and an increase of three times was possible in IOPS.



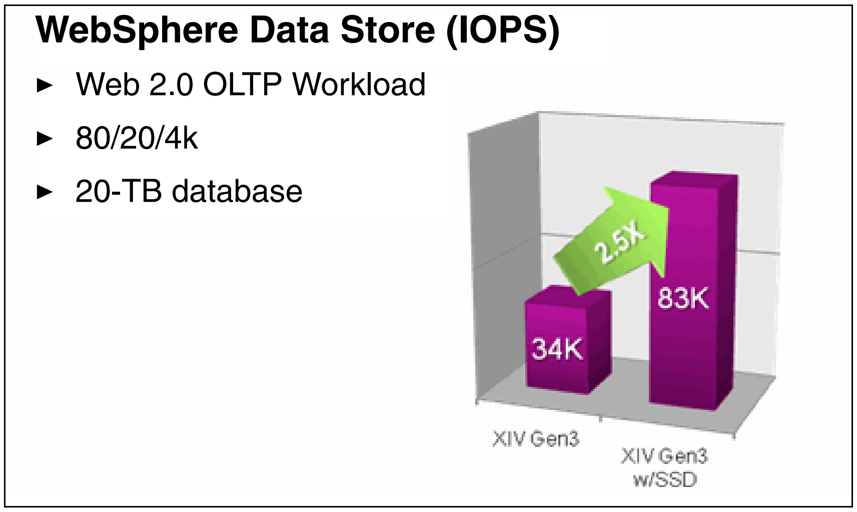
DB2 Brokerage (IOPS)

- Securities trading OLTP Workload
- Mixed block I/O
- 12-TB database

Figure 4-2 IBM DB2® Brokerage application

IBM WebSphere data store

Web access of data is another example of an OLTP random workload. This application benefits from the addition of SSD Caching. Measurements show an increase of 2½ times in IOPS as illustrated in Figure 4-3.



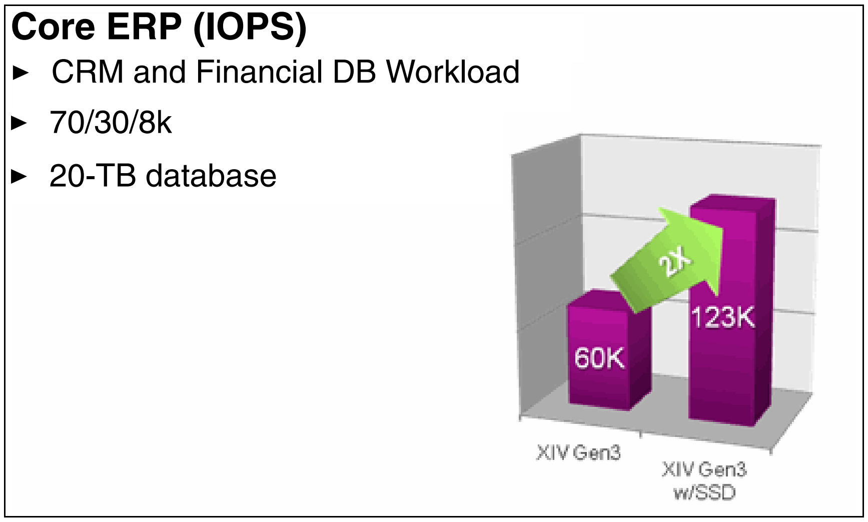
WebSphere Data Store (IOPS)

- Web 2.0 OLTP Workload
- 80/20/4k
- 20-TB database

Figure 4-3 IBM WebSphere® application

Core Enterprise Resource Planning

The Financial Database Workload of Core Enterprise Resource Planning (ERP) shows an increase in IOPS of two times, with the addition of SSD Caching in XIV, as illustrated in Figure 4-4.



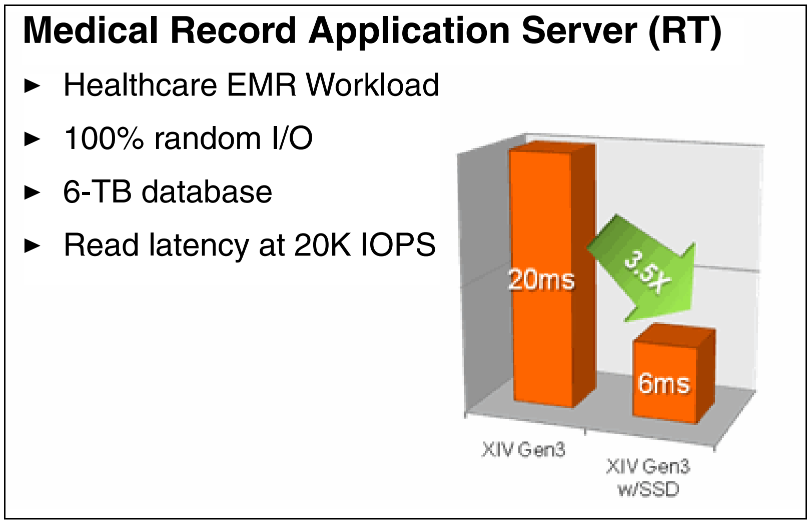
Core ERP (IOPS)

- CRM and Financial DB Workload
- 70/30/8k
- 20-TB database

Figure 4-4 Financial data application

Medical record applications

Medical record applications are used to access patient data, which is a critical activity for the operation of a hospital. Therefore, the application has specific performance requirements. The database used to store the patient data generates a random-read workload that must have good performance. For such a situation, enabling the SSD cache for only the database volumes allows the full benefit of SSD Caching in the XIV system to be available to the medical database. Testing shows that a significant reduction in read latency is possible. See Figure 4-5.



Medical Record Application Server (RT)

- Healthcare EMR Workload
- 100% random I/O
- 6-TB database
- Read latency at 20K IOPS

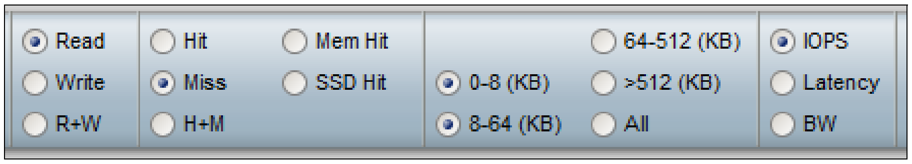
Figure 4-5 Medical records application

4.2 Analyzing the workload on XIV

The XIV system offers several approaches to analyze the workload.

4.2.1 Using the system monitor

If you already have an XIV system, but it is not yet equipped with SSDs, you can use the stem statistics function to analyze a workload. To use this function, click **Monitor** → **Statistics**. In In the area at bottom of the Monitor window that opens (Figure 4-6), you can select filters to separate the data.



[✓ Read] Write R+W

Hit [✓ Miss] H+M Mem Hit SSD Hit

[✓ 0-8 (KB)] [✓ 8-64 (KB)] 64-512 (KB) >512 (KB) All

[✓ IOPS] Latency BW

Figure 4-6 Filter panel for the statistics monitor

These filters are divided by the type of transaction (reads or writes), cache properties (hits versus misses), cache memory hits, SSD cache hits, or the transfer size of I/O as detected by the XIV system. As shown in Figure 4-6, the filter panel now also supports SSD Caching.

To determine if a workload might benefit from SSD Caching, filter the I/Os as follows:

1. For I/Os, select **Read**.
2. For Cache, select **Miss**.
3. For the request size, select I/Os that represents small requests (in the range 0 - 64 KB). To make multiple selections, hold down the Ctrl key.
4. Select **IOPS**.

The resulting I/Os are displayed in the filtered view and are all candidates for SSD acceleration. To help you visualize the results, Figure 4-7 on page 25 and Figure 4-8 on page 25 represent two views of a workload.

Figure 4-7 represents the total IOPS for read and write requests over a period.

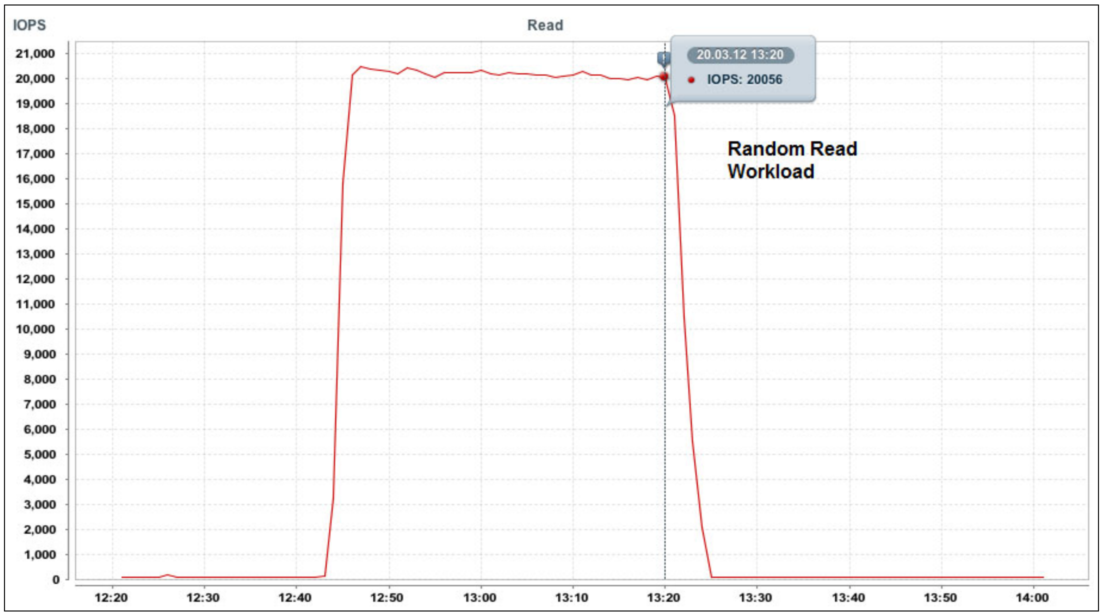


Figure 4-7 Total read and write IOPS

Figure 4-8 shows only read requests, with a size less than 64 KB, that are cache-read misses. This figure represents the portion of the workload that can directly benefit from extended SSD Caching.

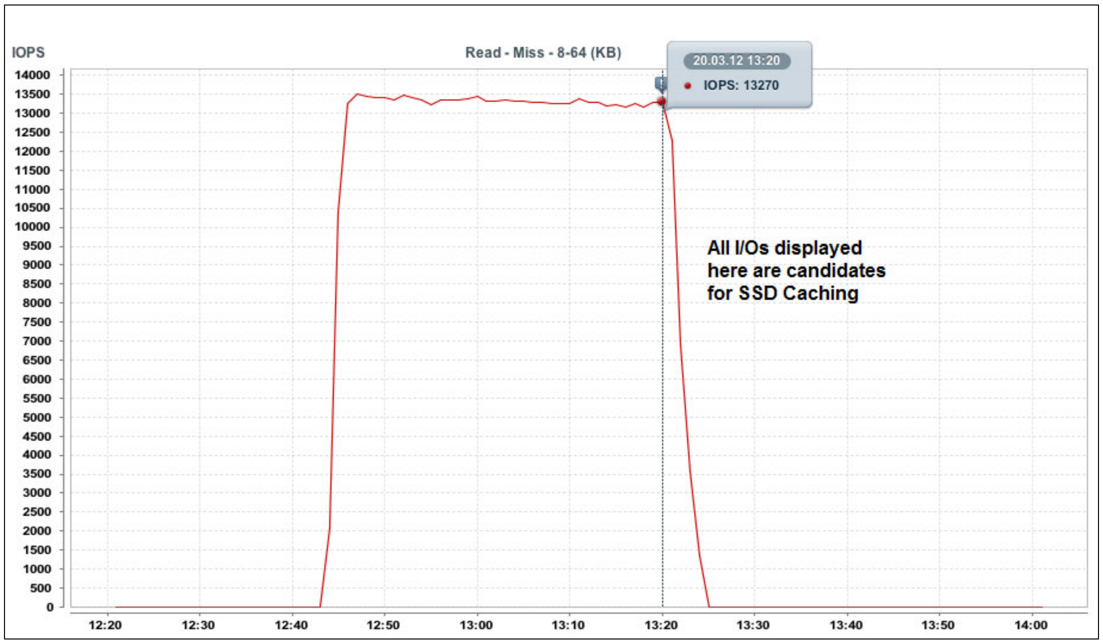


Figure 4-8 Read cache misses

You can also do in-depth analyses of a workload that is running on an XIV system by using the new *XIV block tracing feature*.

4.2.2 Using the XIV block tracing facility

An XIV Gen3 system with software version 11.1, but not yet equipped with SSD Caching, includes a unique feature known as *block tracing*. Block tracing is for use by IBM technical staff and is meant to deliver additional client value, by allowing a deeper understanding of an application workload pattern.

The block tracing facility, which was conceived and developed by IBM Research, collects usage statistics over time to show the application access pattern against a host volume. This pattern helps identify the *moving hit window*. The moving hit windows refers to the active candidates for SSD Caching for a time period. At any time, the area of randomness can shift on the storage system. Knowing the moving hit window for an application is critical in seeding a correct caching architecture.

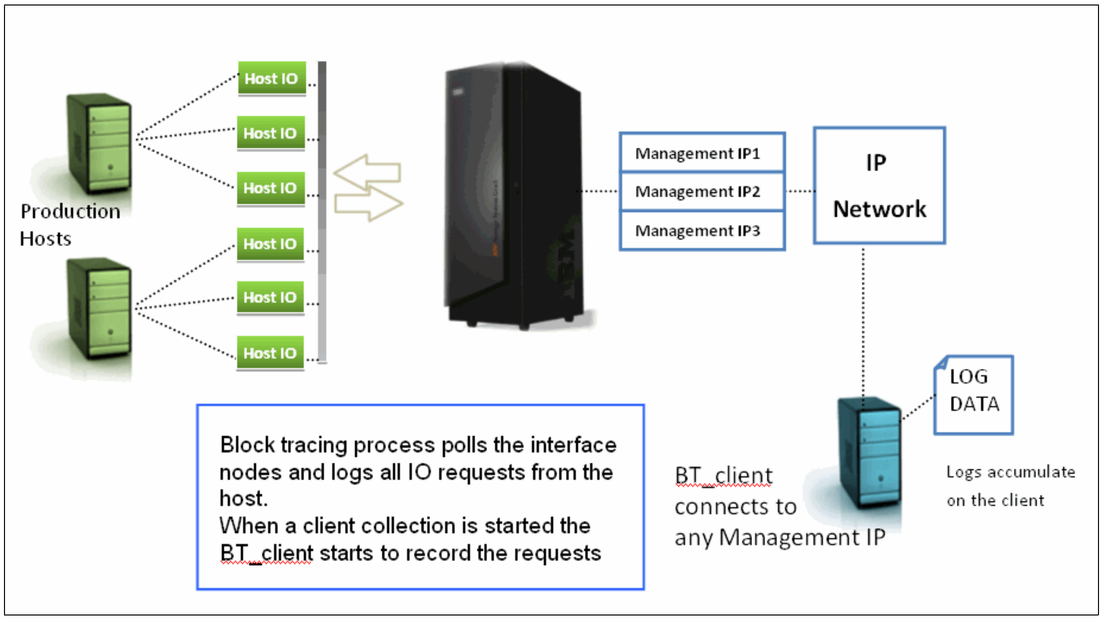
The block tracing facility relies on a cumulative I/O distribution function to determine the areas of a volume that are accessed most frequently. That information can help to predict how to use SSD Caching when it is installed.

The block tracing facility has minimal performance impact on the XIV system. The overall CPU usage is around 0.1%, and no disk I/Os are performed.

In addition, IBM personnel can offload the trace that is taken on an XIV system at a customer site. They can replay it in an IBM lab, on an XIV system of the same capacity, but that has SSD Caching to validate the predictions.

Components and processes of the XIV block tracing facility

Figure 4-9 highlights the components of the block tracing facility.



Block tracing process polls the interface nodes and logs all IO requests from the host.
When a client collection is started the BT_client starts to record the requests

Figure 4-9 Block tracing facility components

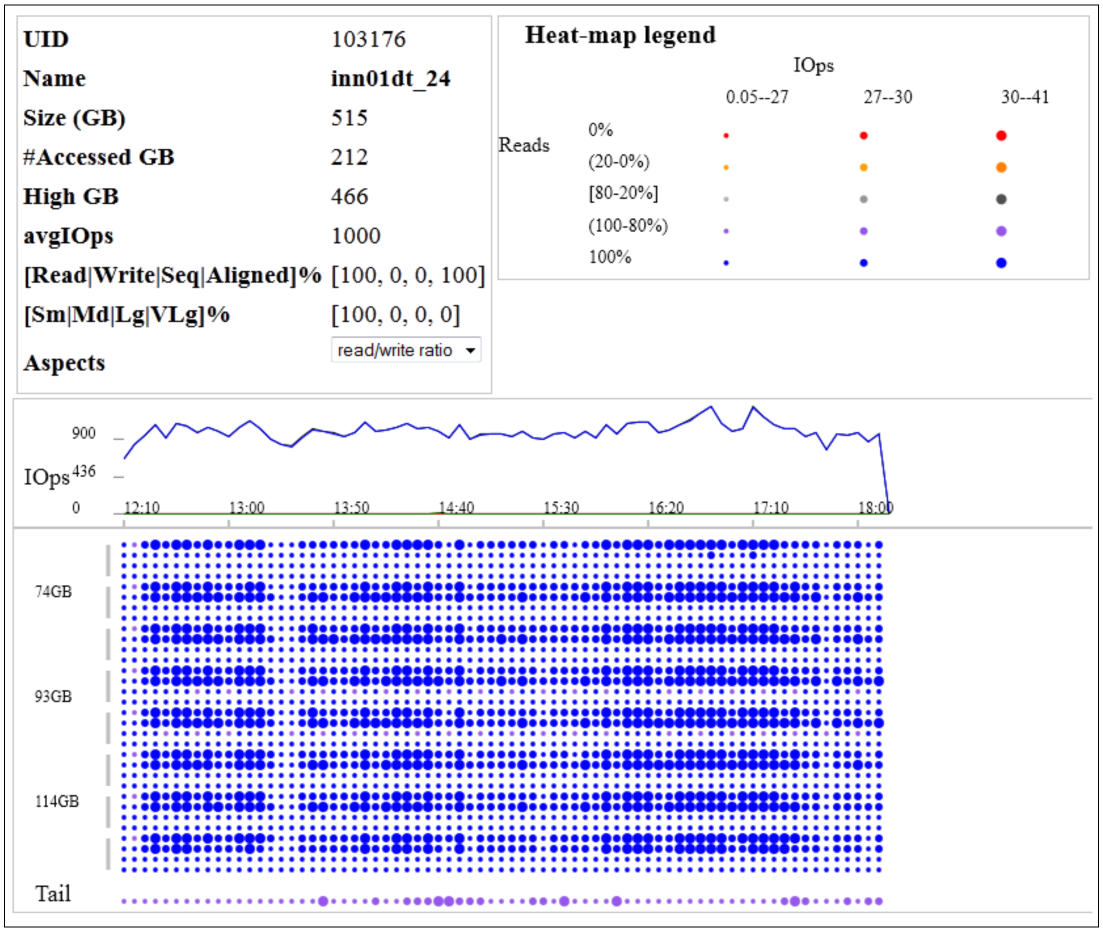
The statistics recording process (BT_server) runs on the XIV system. The XIV system includes up to six interface modules, which are also referred to as *interface nodes* (*inodes*). The interface nodes are gateways to incoming host I/Os. An inode keeps records in-memory for the last million I/Os.

The statistics collections inodes support a special management command that extracts I/O records from a specified, recent time window. This command is triggered by the I/O collection client program. The I/O collection client program (BT_client) runs on a PC that is connected to XIV through an Ethernet connection. The I/O collection program includes a daemon that polls the inodes periodically and collects new I/Os. The daemon also records them in a sorted fashion in a data log that is on the connected PC local disk for offline processing later.

The utility is a separate client program that is written in Java for portability. Typically, an IBM representative or IBM Technical Advisor can run it from their laptop, at a customer site, if needed.

Visualization and cache prediction

The client program also includes a visualization utility. The visualization utility graphically shows the moving hit window for a host volume that is defined on the XIV system (Figure 4-10). By using the I/O-traces, the visualization program calculates a coarse-grained heat map whose granularity is 1 GB per 5 minutes.



UID	103176
Name	inn01dt_24
Size (GB)	515
#Accessed GB	212
High GB	466
avgIOps	1000
[Read|Write|Seq|Aligned]%	[100, 0, 0, 100]
[Sm|Md|Lg|VLg]%	[100, 0, 0, 0]
Aspects	read/write ratio

Heat-map legend

		IOps 0.05--27	27--30	30--41
Reads	0%	●	●	●
	(20-0%)	●	●	●
	[80-20%]	●	●	●
	(100-80%)	●	●	●
	100%	●	●	●

Figure 4-10 XIV host volume heat map

The bottom part of Figure 4-10 shows the heat map. The blue dots show the frequency access pattern for the same volume over 6 hours. These hot spot areas pertain to a host volume, and not an XIV physical disk.

The fundamental XIV concept of uniformly spreading data across all physical disks remains unchanged. Other storage systems target hot spots on disks for relocation to SSDs. However, the XIV system does not relocate any blocks at the physical disk layer, but rather can predict which areas of the logical volume to cache on the SSD.

Cache prediction

By using the same trace data, IBM Research developed algorithms to predict how a workload can reduce read-miss events when read cache is extended to 6 TB. The program does the prediction without needing an XIV system equipped with SSD.

You can use the replay utility to validate the prediction and further validate the model.

Replay utility

IBM service representatives have a program to replay the collected traces from the customer's XIV system. The replay program is used to issue the I/Os as recorded in the log and within the same time frame on an XIV system that has the same capacity as the one on which the logs were recorded. When the replay is done, the IBM service representative can use the XIV statistics monitor to see how the workload performed.

Figure 4-11 shows a comparison of the original (that is without SSDs) read cache hits, the predicted read cache hits, and the actual read cache hits (replay) when the system is equipped with SSD.

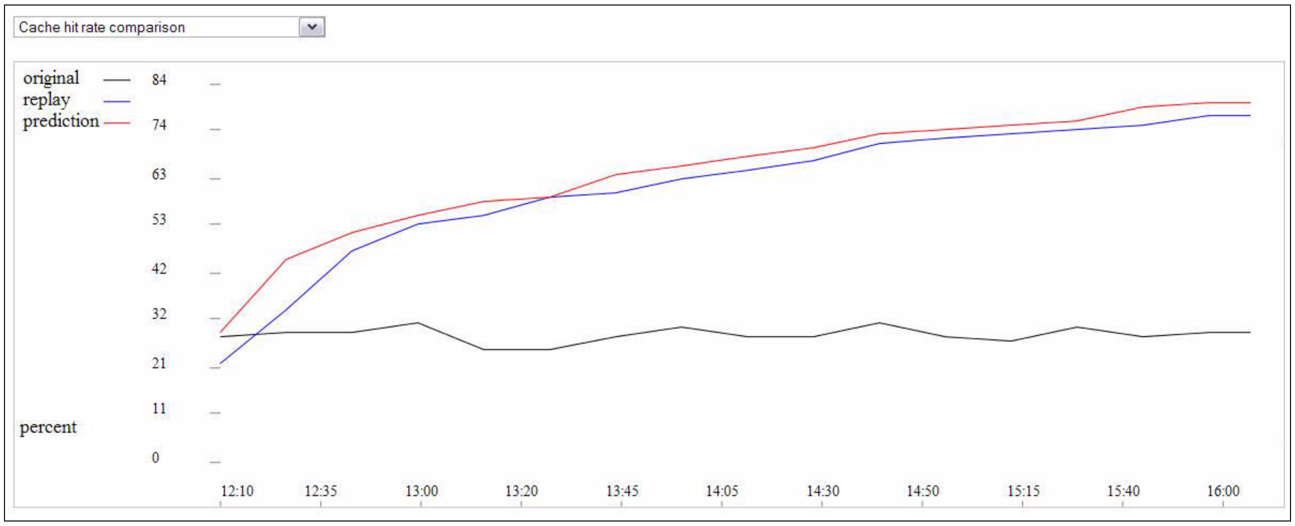
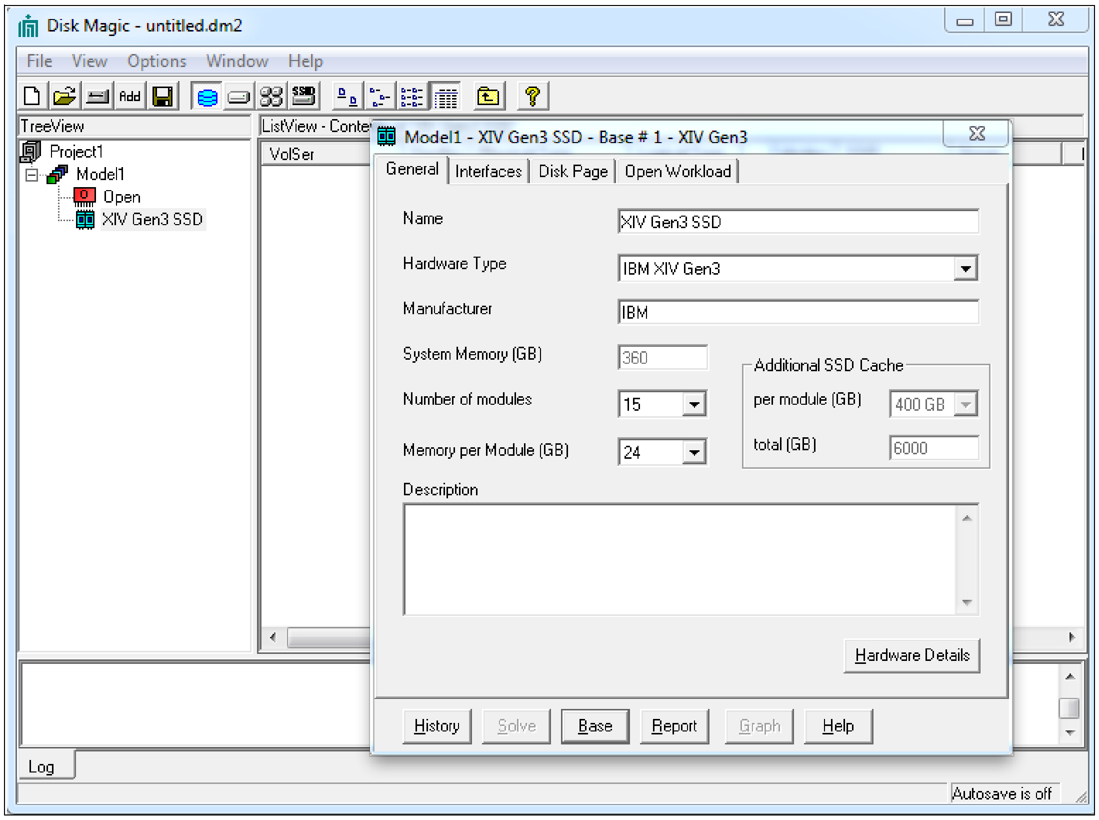


Figure 4-11 Predicting SSD cache hits compared to actual (replay)

4.2.3 Disk Magic

Disk Magic now supports modeling the effect of an SSD read cache in XIV Gen3 configurations as of Disk Magic v9.2.0. Automatic cache modeling also provides an estimation of the expected SSD cache hit rate based on an existing workload. By using the existing workload and the input or estimated SSD cache hit rate, Disk Magic can predict the expected benefit to response time and throughput in any XIV configuration when SSD read cache is added (Figure 4-12).



Name	XIV Gen3 SSD
Hardware Type	IBM XIV Gen3
Manufacturer	IBM
System Memory (GB)	360
Number of modules	15
Memory per Module (GB)	24
Additional SSD Cache: per module (GB)	400 GB
Additional SSD Cache: total (GB)	6000
Description	

Figure 4-12 Using Disk Magic to model XIV Gen3 with SSD Caching

5

Installing and upgrading solid-state drives

This chapter outlines how an IBM service representative installs the solid-state drive (SSD) Caching feature on an IBM XIV Storage System that is deployed in a client environment. After they install and enable the XIV system modules, the SSDs are ready for use. No additional monitoring, configuring, or tuning is necessary. The SSD Caching feature is supported automatically by the unique XIV caching algorithm, which dynamically adapts to detected I/O request patterns.

Ordering information: You can order a system that has SSDs already installed, which applies to fully or partially populated racks. You can also order SSDs for a Gen3 system that is already installed.

Important: SSD installation and upgrades are nondisruptive procedures.

This chapter includes the following sections:

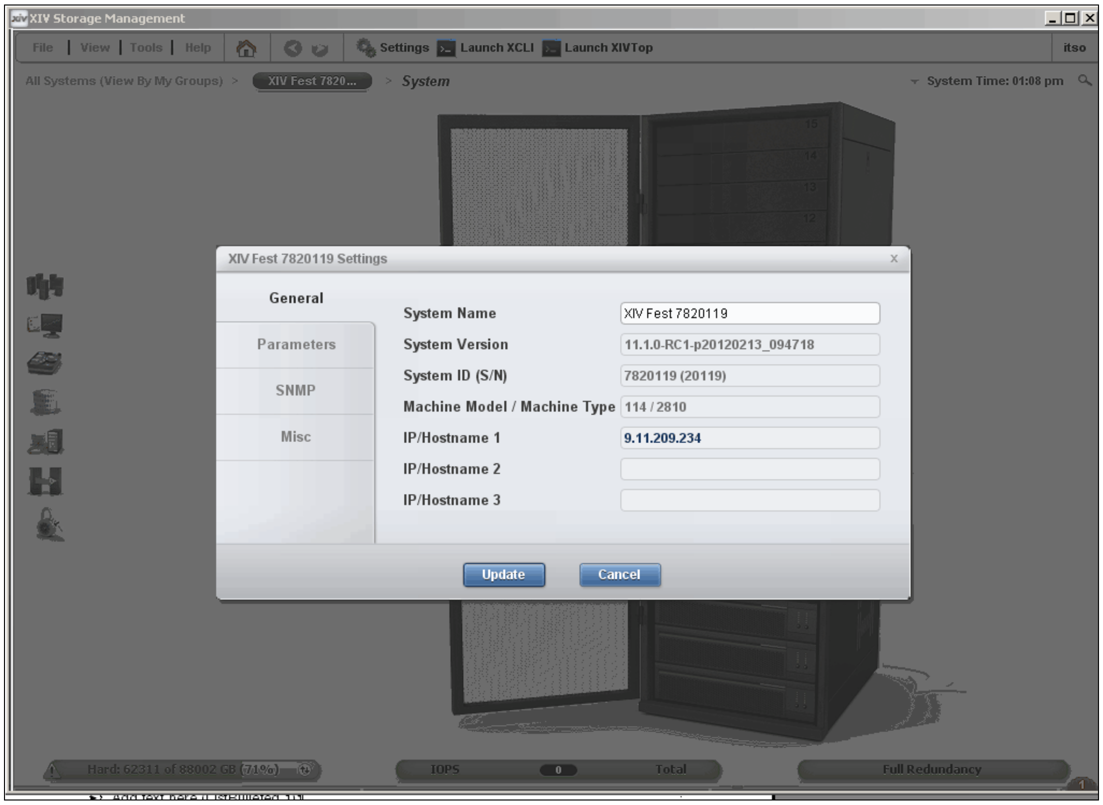
- Prerequisites
- Installing SSD
- Upgrading SSD after a module upgrade

5.1 Prerequisites

Before the IBM service representative can install SSDs, ensure that XIV storage software (firmware) Version 11.1.0 or later is installed on the XIV system. The IBM service representative also upgrades the XIV system code. However you can check your system software level before ordering the SSDs, by using the XIV Storage Management GUI or the XIV command-line interface (XCLI):

- Checking the system version by using the GUI

  To check the code level on GUI, click **Setting**s on system dashboard (larger window in Figure 5-1). In the Settings window that opens, you can read the system version (inset in Figure 5-1).



System Name	XIV Fest 7820119
System Version	11.1.0-RC1-p20120213_094718
System ID (S/N)	7820119 (20119)
Machine Model / Machine Type	114 / 2810
IP/Hostname 1	9.11.209.234
IP/Hostname 2	
IP/Hostname 3	

Figure 5-1 Checking the system code level by using the GUI

- Checking the system version by using the XCLI

  To check the system version, open the XCLI window and enter the **`version_get`** command as shown as Example 5-1.

Example 5-1 Checking the system code level by using XCLI

```
>>version_get
Version
11.1.0-RC1-p20120213_094718
```

5.2 Installing SSD

After the system has the appropriate XIV software version, the IBM service representative can install the SSDs in your XIV Gen3 system.

Restriction: Only an authorized IBM service representative can install SSDs.

The IBM service representative installs SSD by using the following tasks:

1. From the IBM XIV Storage Management GUI, verify that the XIV system status on the lower-right side of the window shows *Full Redundancy*.
2. From the rear of the rack, insert an SSD into each data and interface module as illustrated in Figure 5-2:
   - Orient the SSD so that the arrow on the pull tab points upward.
   - Push the release tab to the side, and insert the SSD into the opening in the carrier assembly. Release the tab to secure the SSD in place.

   Important: Install SSDs in *every* data and interface module, starting from the lowest module in the rack and working upward.

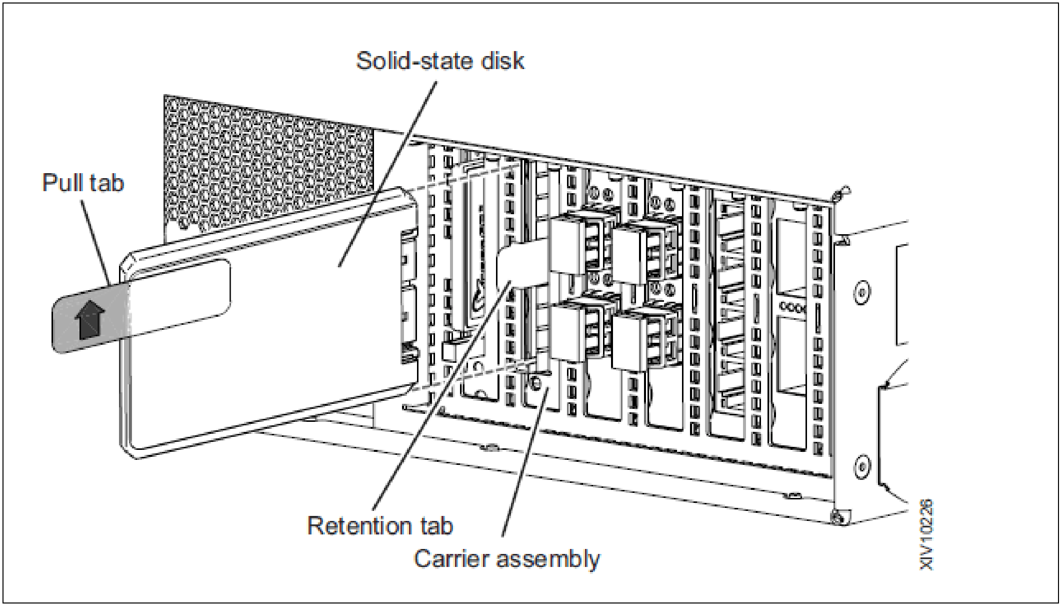


Figure 5-2 Inserting an SSD into the module

3. Using an XCLI command strictly reserved for the IBM service representative, enable the flash cache and discover the SSD modules.
4. Run a component test on each SSD. When the component test is over and is successful, all SSDs are ready.
5. Phase in all SSDs by using the XCLI or the GUI.

The SSD installation by the IBM service representative is now completed.

The XIV user or system administrator can now verify that the SSDs have a status of *OK* by using either of the following options:

- Run the **`ssd_list`** command on the XCLI (Example 5-2).

Example 5-2 Verifying the SSD status by using the CLI

```
>>ssd_list
Component ID Status Currently Functioning  Capacity   Target Status   Vendor   Model              Size   Serial   Firmware    Fru      Group   Temperature
1:SSD:2:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB87  MA40      99Y0720          41
1:SSD:3:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB84  MA40      99Y0720          41
1:SSD:8:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB7E  MA40      99Y0720          42
1:SSD:6:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB96  MA40      99Y0720          40
1:SSD:4:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB8D  MA40      99Y0720          41
1:SSD:9:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200C237  MA40      99Y0720          45
1:SSD:1:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB90  MA40      99Y0720          40
1:SSD:5:1      OK       yes                  512GB                    XIV     MTFDDAA512MAR-1KAAB  488378  0200BB93  MA40      99Y0720          42
1:SSD:7:1     OK       yes                   512GB                   XIV      MTFDDAA512MAR-1KAAB  488378  0200C184  MA40     99Y0720          43
```

- Using the GUI as shown in Figure 5-3.

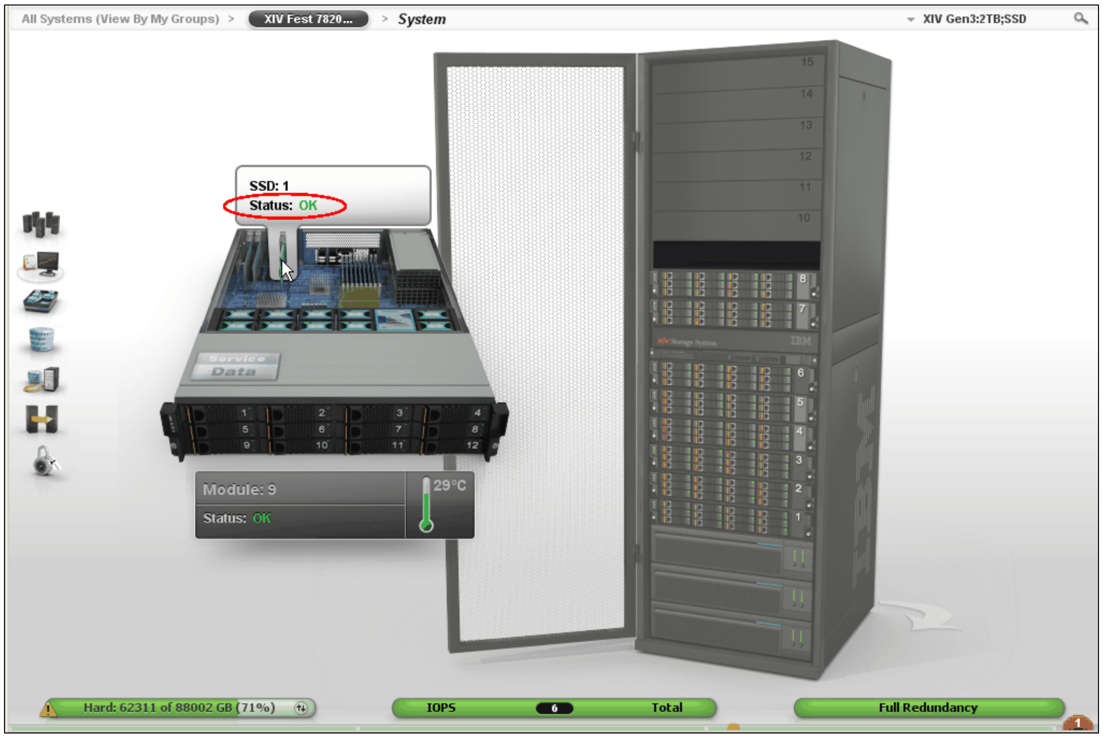


Figure 5-3 SSD Cache Status verify

After the SSDs are enabled, you do not need any additional software or tuning considerations for efficient SSD utilization. The SSD Caching feature is supported automatically by the unique XIV caching algorithm, which dynamically adapts to detected I/O request patterns.

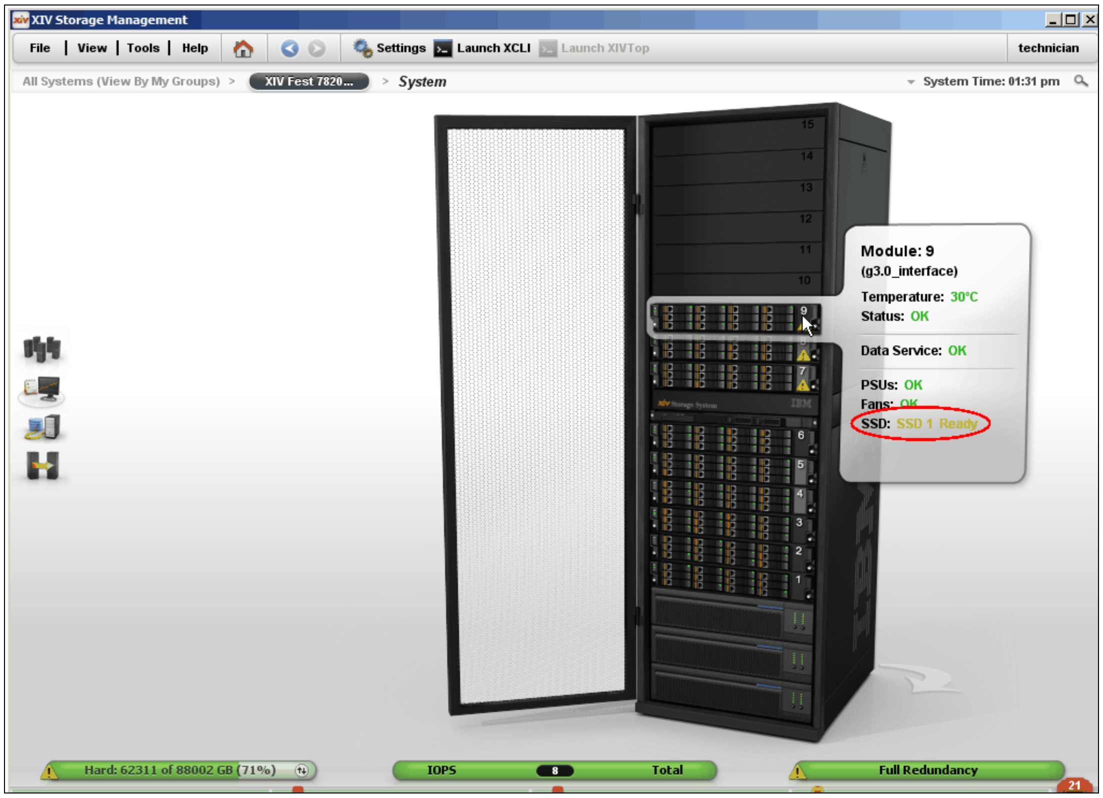
5.3 Upgrading SSD after a module upgrade

SSDs can also be installed on partially populated XIV system with fewer than 15 modules. The only constraint is that SSDs must be installed for every module in the rack.

If you decide to order additional modules to complement a partially populated rack that is already configured with SSDs, you must also order SSDs for the additional modules.

For example, consider a situation where you have a 6-module XIV Gen3 System already equipped with SSDs. Then you order three additional modules with SSDs.

The IBM service representative first installs the additional modules in the rack. When the modules are phased in, the XIV system starts a redistribution procedure. After the redistribution procedure is finished, the additional modules show an *OK* status in green, but the SSD status is displayed as *Ready* (not *OK*) as illustrated in Figure 5-4.



Module: 9
(g3.0_interface)
Temperature: 30°C
Status: OK
Data Service: OK
PSUs: OK
Fans: OK
SSD: SSD 1 Ready

Figure 5-4 Module status after additional module installation

As explained in 5.2, "Installing SSD" on page 33, the IBM service representative runs a component test for the additional SSDs and then enables them.

The status of the additional SSDs is now displayed as *OK*. You can verify the status in the XIV GUI as shown in Figure 5-3 on page 34 or by using the `ssd_list` XCLI command as shown in Example 5-2 on page 34.

6

GUI and XCLI support for solid-state drives

This chapter describes the changes made to the IBM XIV Storage Management GUI and XIV command-line interface (XCLI) commands to accommodate the new solid-state drive (SSD) Caching feature. It includes the following sections:

- SSD Caching support in XIV
- SSD Caching management with XCLI

6.1 SSD Caching support in XIV

After the IBM service representative installs and enables the SSDs, the XIV system can use them as an extension of the read cache. The SSD cache automatically adjusts to the workload, providing a performance boost for small I/O random reads. The storage administrator does not need to perform any additional configuration or tuning.

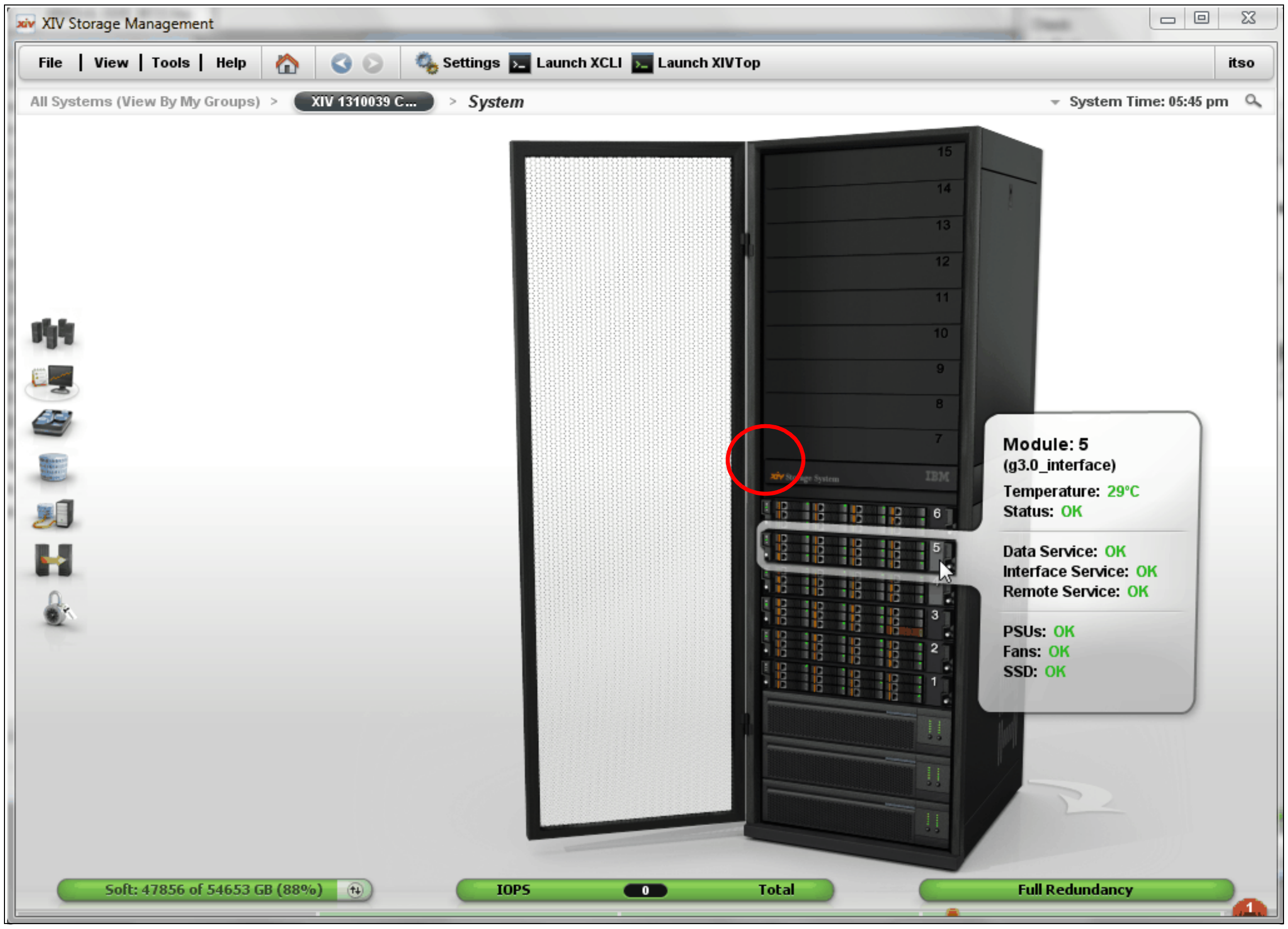
However, the storage administrator has the flexibility to perform optional actions, such as the following examples:

- Check the SSD status in each module.
- Enable or disable SSD Caching at the system level.
- Enable or disable SSD Caching at the volume level.

6.1.1 Viewing and managing the SSD by using the GUI

To view and manage the SSDs, the XIV Storage Management GUI Version 3.1 is required. You can check the health and the status of the SSDs in the main system view.

When you move the cursor over a module, a window is displayed that shows the temperature and status of major module components. When SSDs are present, the SSD status is displayed at the bottom of the window. If the SSD is operational, a green OK status is displayed as shown in Figure 6-1.



Module: 5
(g3.0_interface)
Temperature: 29°C
Status: OK
Data Service: OK
Interface Service: OK
Remote Service: OK
PSUs: OK
Fans: OK
SSD: OK

Figure 6-1 Module status with SSD

If SSDs are not installed, the window does not include the SSD label at the bottom, as shown in Figure 6-2.

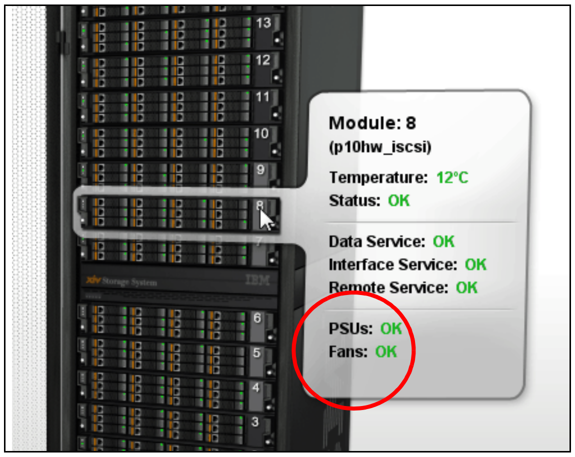
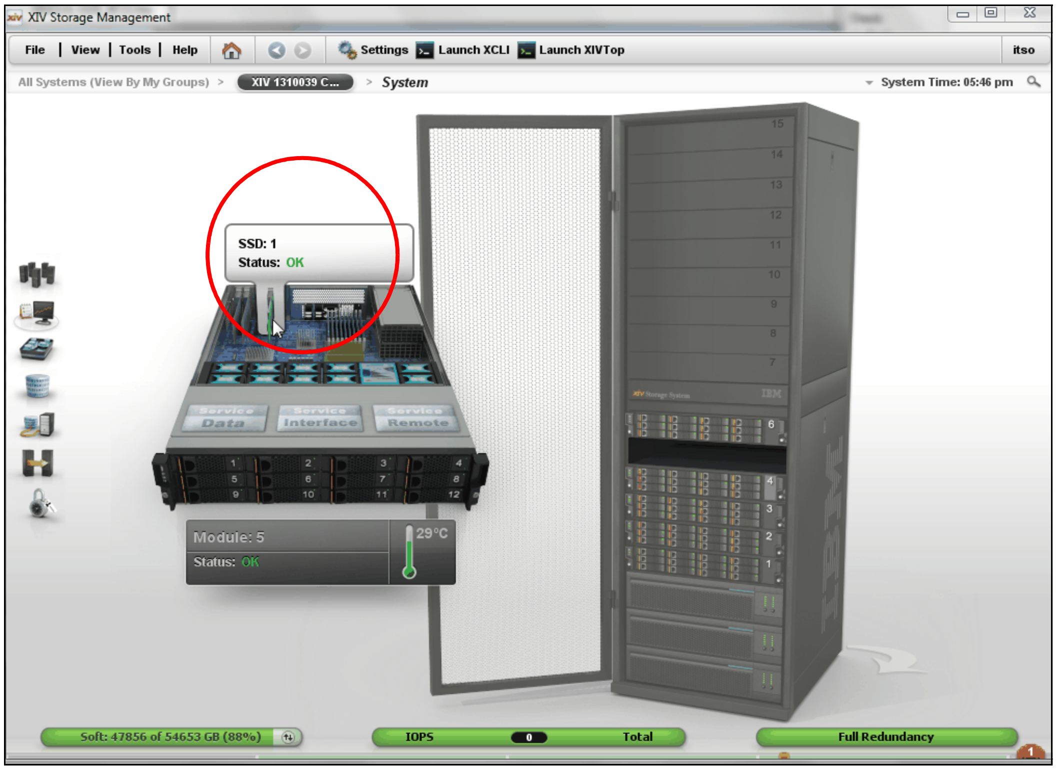

Module: 8
(p10hw_iscsi)
Temperature: 12°C
Status: OK
Data Service: OK
Interface Service: OK
Remote Service: OK
PSUs: OK
Fans: OK

Figure 6-2 Module status without SSDs

Clicking a module number in the main system view (Figure 6-1 on page 38), opens a full perspective view of the module and its components (Figure 6-3).



SSD: 1
Status: OK
Module: 5
Status: OK
29°C

Figure 6-3 SSD status

From this view, you can also check the SSD status. If an SSD is installed, place the mouse cursor over the SSD in the module view. The SSD can have one of the following statuses:

- A *green status* indicates that the SSD is OK and phased in.
- A *yellow status* indicates that the SSD is phased out.
- A *red status* indicates that the SSD failed.

6.1.2 Setting SSD Caching

When the IBM service representative initially enables SSD Caching after installation, SSD Caching becomes effective and enabled for all volumes that are defined in the XIV system. New volumes that are created afterward, by default, also have caching enabled.

The storage administrator can change the default status to disabled or to specifically enable or disable SSD Caching at the volume level. You can manually change SSD Caching state for specific volumes from the **Volumes and Snapshots** view.

Tip: You can dynamically enable or disable SSD Caching at any time.

Setting SSD Caching at the system level

To enable or disable the SSD Caching for the entire XIV system:

1. From the toolbar in the main system view, click **Setting** (Figure 6-4).

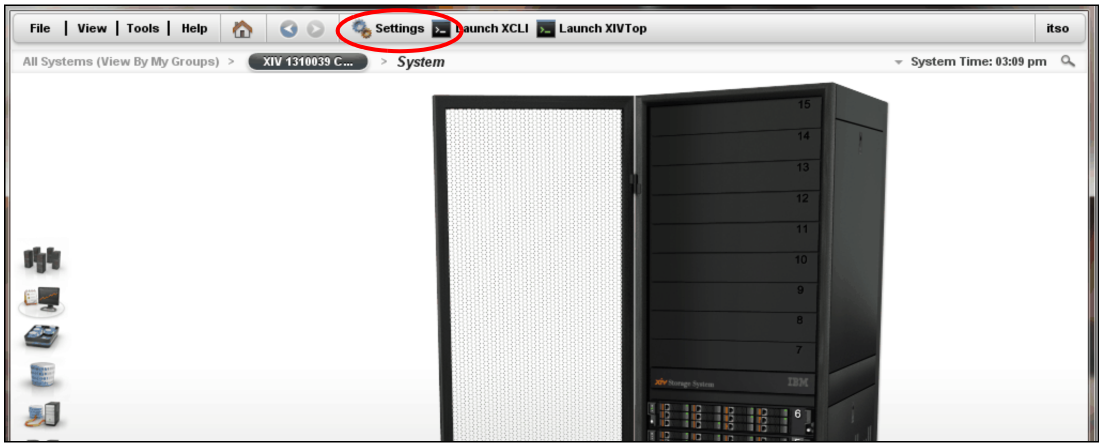
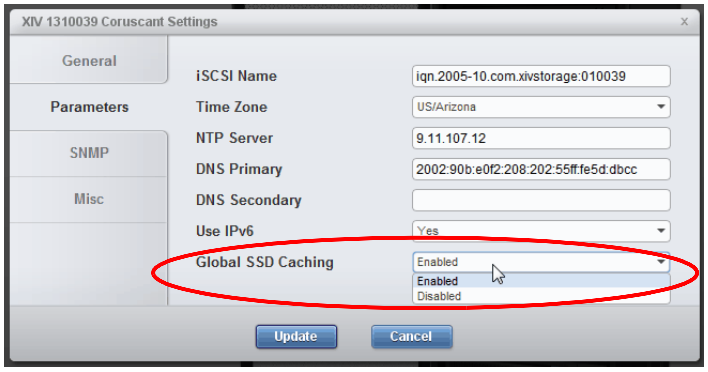


Figure 6-4 Opening the system settings

2. In the Settings panel (Figure 6-5), on the **Parameters** tab, set Global SSD Caching to **Enabled** or **Disabled**. Then click **Update**.



XIV 1310039 Coruscant Settings

iSCSI Name	iqn.2005-10.com.xivstorage:010039
Time Zone	US/Arizona
NTP Server	9.11.107.12
DNS Primary	2002:90b:e0f2:208:202:55ff:fe5d:dbcc
DNS Secondary	
Use IPv6	Yes
Global SSD Caching	Enabled

Enabled
Disabled

Figure 6-5 System settings for Global SSD Caching

By default, the newly created volume follows the system-level setting.

System-level settings: If SSD Caching is *disabled* from the system-level settings, it is disabled by default for every new volume that is defined. If SSD Caching is *enabled* from the system-level settings, it is enabled by default for every new volume that is defined.

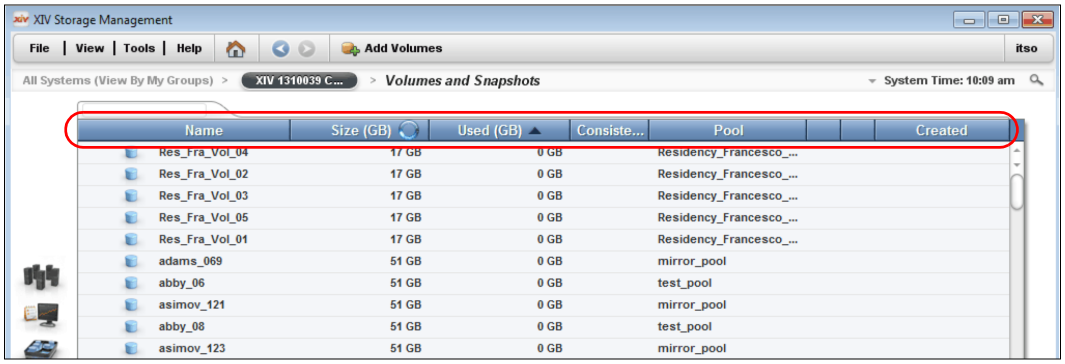
Setting SSD Caching at the volume level

The Volume and Snapshots view in the XIV Storage Management GUI includes a new SSD field that indicates the SSD Caching status for the volumes and defined in the system. By default, the new SSD field is not included in the view.

Adding the SSD field

To include the SSD field in the view:

1. Open the **Volumes and Snapshots** view from the main XIV GUI.
2. Right-click anywhere on the field bar (Figure 6-6), and the Customize Columns window opens.

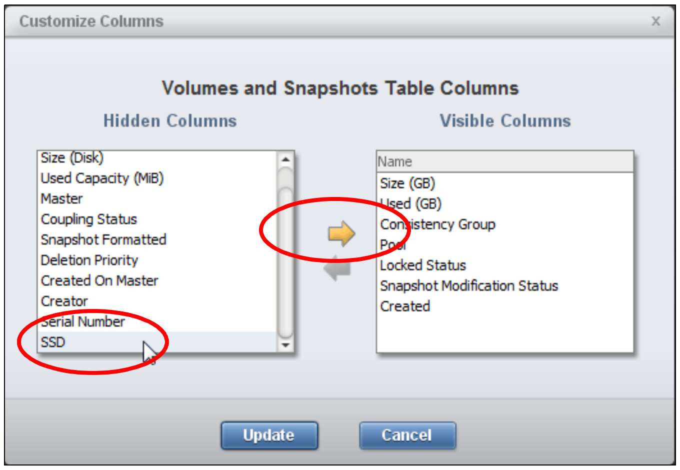

XIV Storage Management
File | View | Tools | Help
Add Volumes
itso
All Systems (View By My Groups) > XIV 1310039 C... > Volumes and Snapshots
System Time: 10:09 am

Name	Size (GB)	Used (GB)	Consiste...	Pool			Created
Res_Fra_Vol_04	17 GB	0 GB		Residency_Francesco_...			
Res_Fra_Vol_02	17 GB	0 GB		Residency_Francesco_...			
Res_Fra_Vol_03	17 GB	0 GB		Residency_Francesco_...			
Res_Fra_Vol_05	17 GB	0 GB		Residency_Francesco_...			
Res_Fra_Vol_01	17 GB	0 GB		Residency_Francesco_...			
adams_069	51 GB	0 GB		mirror_pool			
abby_06	51 GB	0 GB		test_pool			
asimov_121	51 GB	0 GB		mirror_pool			
abby_08	51 GB	0 GB		test_pool			
asimov_123	51 GB	0 GB		mirror_pool			

Figure 6-6 Volumes and Snapshots view

3. In the Customize Columns window (Figure 6-7):

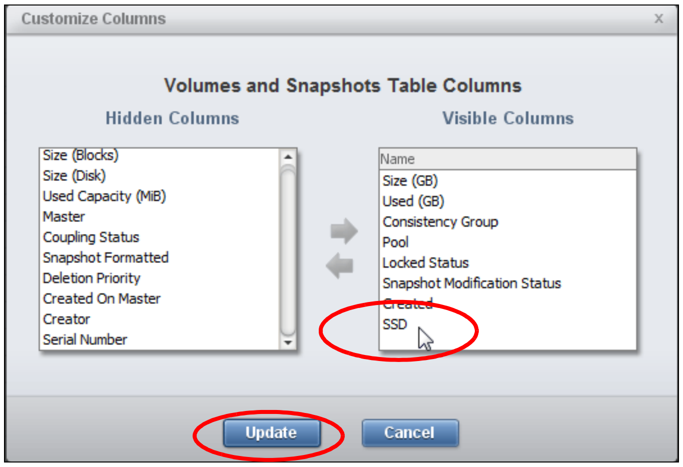
   a. In the Hidden Columns area, select **SSD**, and then click the yellow right arrow indicator.



Hidden Columns	Visible Columns
Size (Disk)	Name
Used Capacity (MiB)	Size (GB)
Master	Used (GB)
Coupling Status	Consistency Group
Snapshot Formatted	Pool
Deletion Priority	Locked Status
Created On Master	Snapshot Modification Status
Creator	Created
Serial Number	
SSD	

Figure 6-7 Customize Columns window

The SSD field moves to the right box as shown in Figure 6-8.



Customize Columns

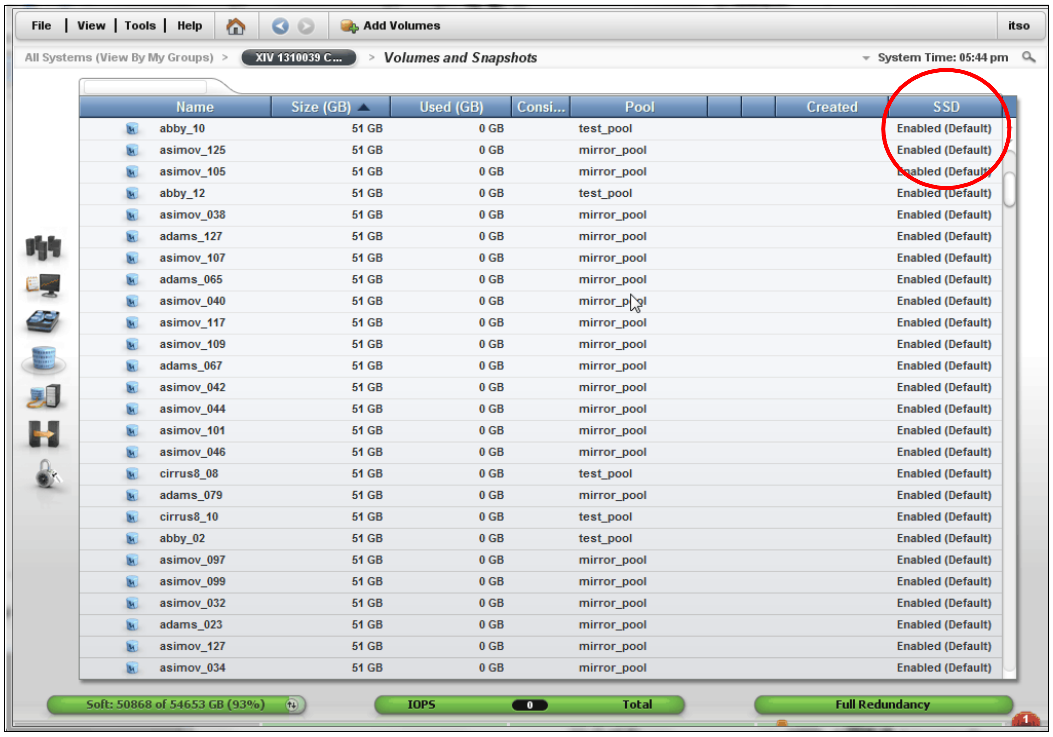
Volumes and Snapshots Table Columns

Hidden Columns	Visible Columns
Size (Blocks)	Name
Size (Disk)	Size (GB)
Used Capacity (MiB)	Used (GB)
Master	Consistency Group
Coupling Status	Pool
Snapshot Formatted	Locked Status
Deletion Priority	Snapshot Modification Status
Created On Master	Created
Creator	SSD
Serial Number	

Figure 6-8 SSD field moved to the Visible Columns area

b. To commit the change, click **Update**.

You return to the Volumes and Snapshots view, with the new SSD column added (Figure 6-9).



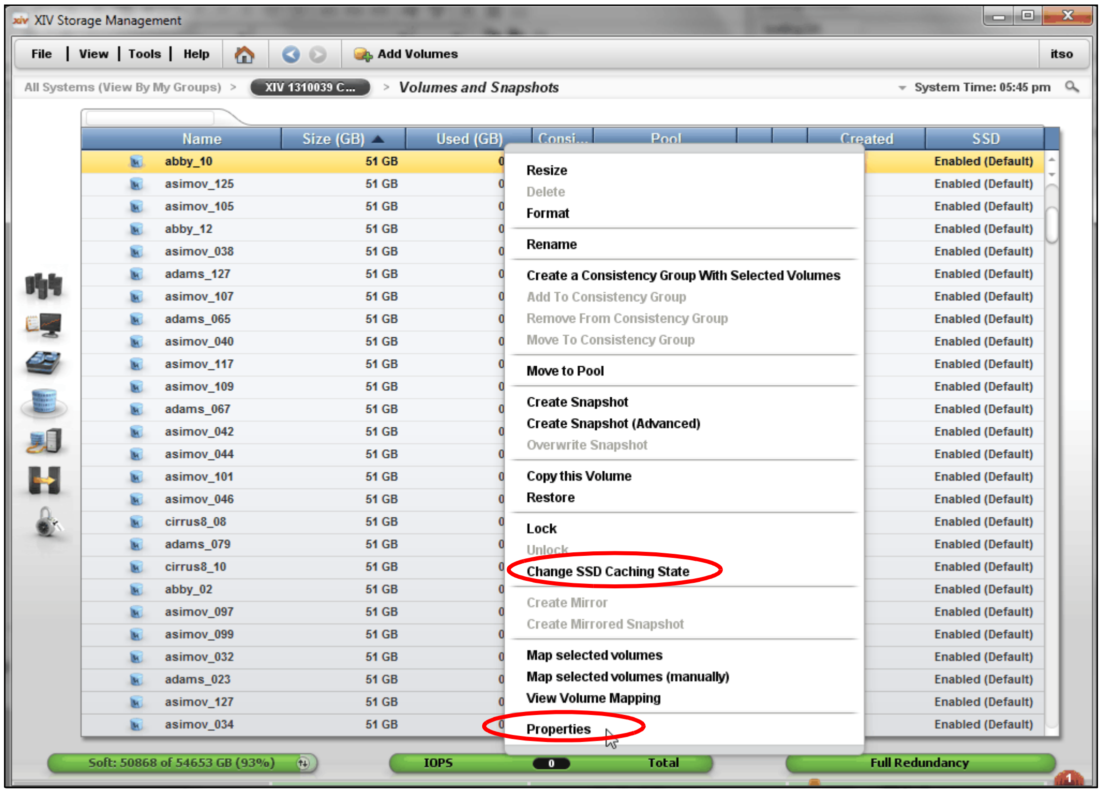
File | View | Tools | Help
Add Volumes
itso
All Systems (View By My Groups) > XIV 1310039 C... > Volumes and Snapshots
System Time: 05:44 pm

Name	Size (GB)	Used (GB)	Consi...	Pool	Created	SSD
abby_10	51 GB	0 GB		test_pool		Enabled (Default)
asimov_125	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_105	51 GB	0 GB		mirror_pool		Enabled (Default)
abby_12	51 GB	0 GB		test_pool		Enabled (Default)
asimov_038	51 GB	0 GB		mirror_pool		Enabled (Default)
adams_127	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_107	51 GB	0 GB		mirror_pool		Enabled (Default)
adams_065	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_040	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_117	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_109	51 GB	0 GB		mirror_pool		Enabled (Default)
adams_067	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_042	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_044	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_101	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_046	51 GB	0 GB		mirror_pool		Enabled (Default)
cirrus8_08	51 GB	0 GB		test_pool		Enabled (Default)
adams_079	51 GB	0 GB		mirror_pool		Enabled (Default)
cirrus8_10	51 GB	0 GB		test_pool		Enabled (Default)
abby_02	51 GB	0 GB		test_pool		Enabled (Default)
asimov_097	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_099	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_032	51 GB	0 GB		mirror_pool		Enabled (Default)
adams_023	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_127	51 GB	0 GB		mirror_pool		Enabled (Default)
asimov_034	51 GB	0 GB		mirror_pool		Enabled (Default)

Figure 6-9 Volumes and Snapshots view showing the new SSD column

Changing the SSD Caching state

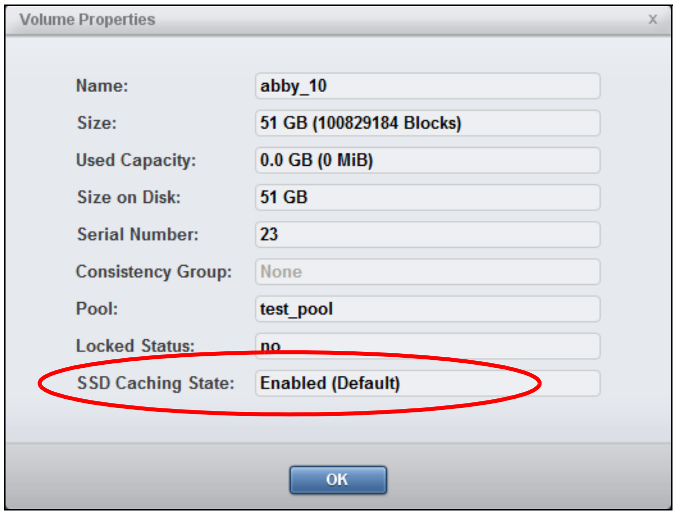
If you right-click a volume row, you can click either **Change SSD Caching State** or **Properties** to see the volume properties (Figure 6-10).



Name	Size (GB)	Used (GB)	Consi...	Pool	Created	SSD
abby_10	51 GB	0				Enabled (Default)
asimov_125	51 GB	0				Enabled (Default)
asimov_105	51 GB	0				Enabled (Default)
abby_12	51 GB	0				Enabled (Default)
asimov_038	51 GB	0				Enabled (Default)
adams_127	51 GB	0				Enabled (Default)
asimov_107	51 GB	0				Enabled (Default)
adams_065	51 GB	0				Enabled (Default)
asimov_040	51 GB	0				Enabled (Default)
asimov_117	51 GB	0				Enabled (Default)
asimov_109	51 GB	0				Enabled (Default)
adams_067	51 GB	0				Enabled (Default)
asimov_042	51 GB	0				Enabled (Default)
asimov_044	51 GB	0				Enabled (Default)
asimov_101	51 GB	0				Enabled (Default)
asimov_046	51 GB	0				Enabled (Default)
cirrus8_08	51 GB	0				Enabled (Default)
adams_079	51 GB	0				Enabled (Default)
cirrus8_10	51 GB	0				Enabled (Default)
abby_02	51 GB	0				Enabled (Default)
asimov_097	51 GB	0				Enabled (Default)
asimov_099	51 GB	0				Enabled (Default)
asimov_032	51 GB	0				Enabled (Default)
adams_023	51 GB	0				Enabled (Default)
asimov_127	51 GB	0				Enabled (Default)
asimov_034	51 GB	0				Enabled (Default)

Figure 6-10 Right-clicking a volume row

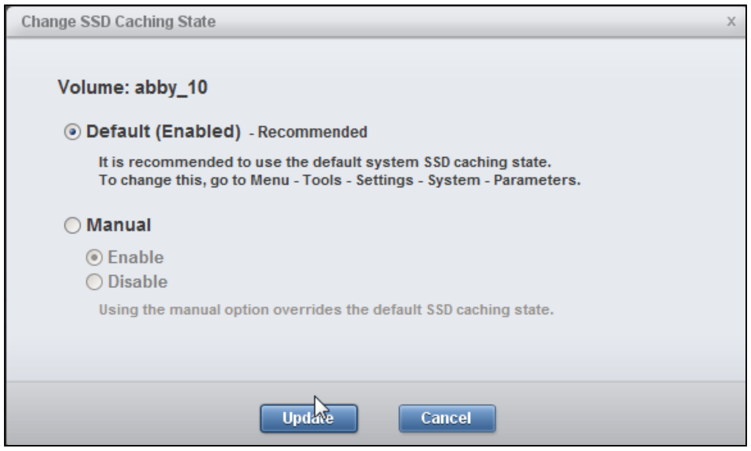
If you select **Properties**, the Volume Properties window (Figure 6-11) opens. This panel shows a new *SSD Caching State field*, which indicates whether SSD Caching for the volume is enabled or disabled and whether it is the system Default.



Name:	abby_10
Size:	51 GB (100829184 Blocks)
Used Capacity:	0.0 GB (0 MiB)
Size on Disk:	51 GB
Serial Number:	23
Consistency Group:	None
Pool:	test_pool
Locked Status:	no
SSD Caching State:	Enabled (Default)

Figure 6-11 Volume Properties window

If you select **Change SSD Caching State** (in Figure 6-10), the Change SSD Caching State window (Figure 6-12) opens. In this window, you can change the status of the SSD Caching for that volume.


Change SSD Caching State

Volume: abby_10

[✓ Default (Enabled)] - Recommended

It is recommended to use the default system SSD caching state.
To change this, go to Menu - Tools - Settings - System - Parameters.

Manual

[✓ Enable]
Disable

Using the manual option overrides the default SSD caching state.

Figure 6-12 Changing the SSD Caching state of the volume

If the **Default** setting is selected, the volume follows the current system settings. You can override the **Default** setting by selecting **Manual**. Then the volume no longer follows the current default system settings. In this case, you can select either of the following options:

- **Enable** to enable SSD Caching for the selected volume
- **Disable** to disable SSD Caching for the selected volume

Tip: The overall system status setting for SSD Caching is **Default (enabled)** as shown in Figure 6-12.

You can select more than one volume in the Volumes and Snapshots view, if you need to change the SSD Caching State for a list of volumes:

- Click the first volume to select it. The volume becomes highlighted in yellow as shown in Figure 6-10 on page 43.

- To select a sequential list of volumes, hold down the Shift key and click the last volume that you want to include. All the volumes in the range are selected and highlighted as shown in Figure 6-13.

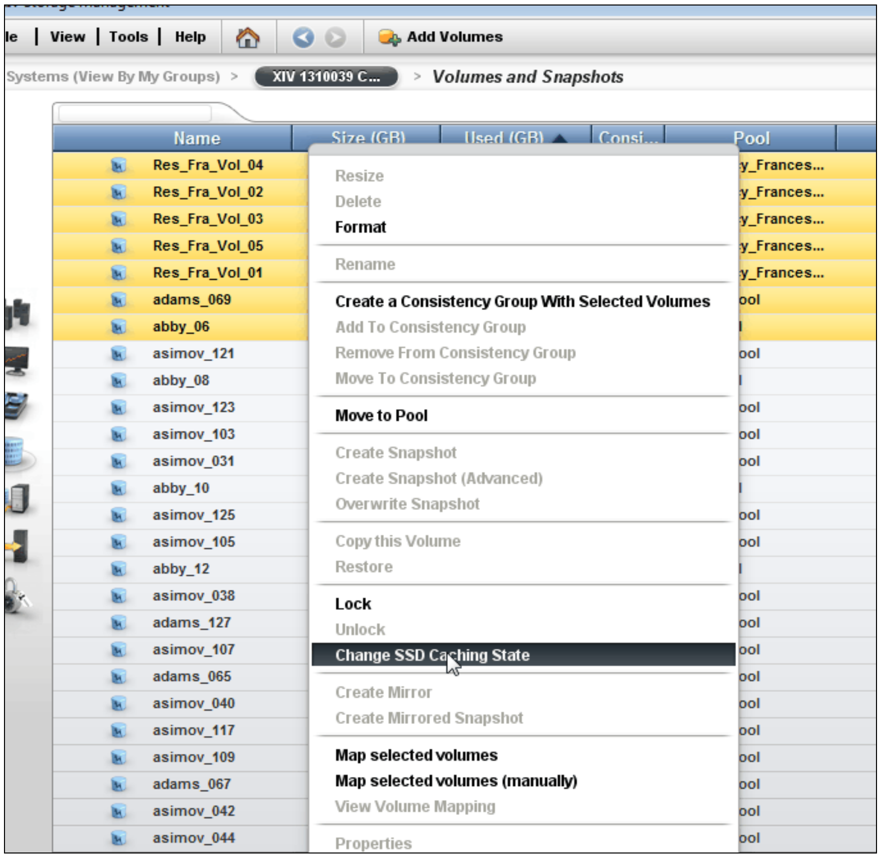
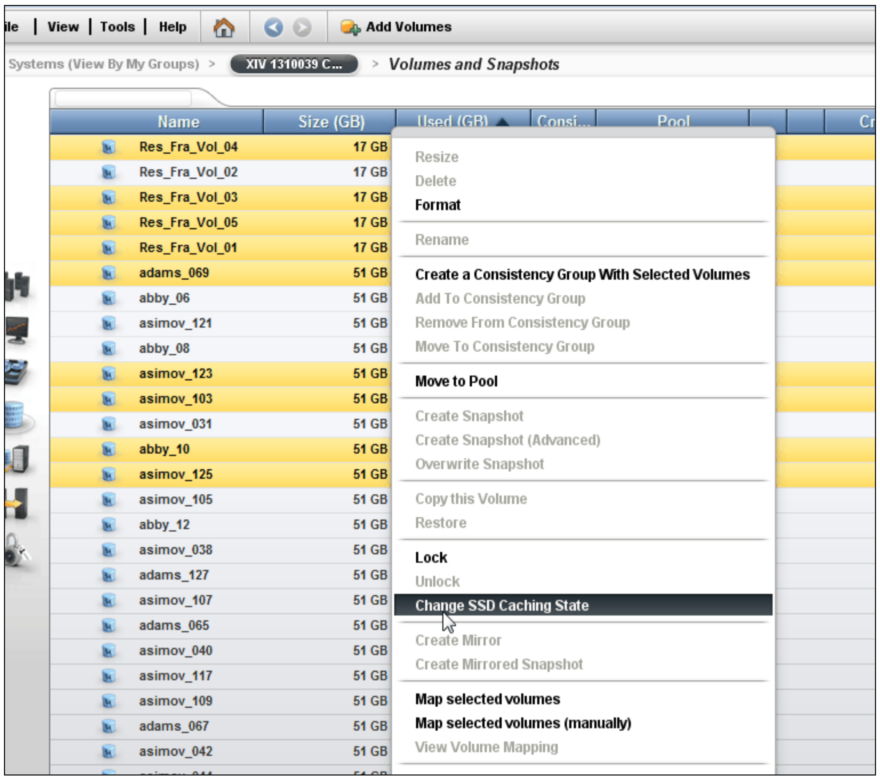


Figure 6-13 Continuous Volumes Selection

- To select specific volumes, hold down the Ctrl key and click one volume at a time, on as many volumes you want to include in your selection. See Figure 6-14.



Name	Size (GB)
Res_Fra_Vol_04	17 GB
Res_Fra_Vol_02	17 GB
Res_Fra_Vol_03	17 GB
Res_Fra_Vol_05	17 GB
Res_Fra_Vol_01	17 GB
adams_069	51 GB
abby_06	51 GB
asimov_121	51 GB
abby_08	51 GB
asimov_123	51 GB
asimov_103	51 GB
asimov_031	51 GB
abby_10	51 GB
asimov_125	51 GB
asimov_105	51 GB
abby_12	51 GB
asimov_038	51 GB
adams_127	51 GB
asimov_107	51 GB
adams_065	51 GB
asimov_040	51 GB
asimov_117	51 GB
asimov_109	51 GB
adams_067	51 GB
asimov_042	51 GB

Figure 6-14 Random Volumes Selection

After you select the necessary volumes, right-click the highlighted volumes and select **Change SSD Caching State**.

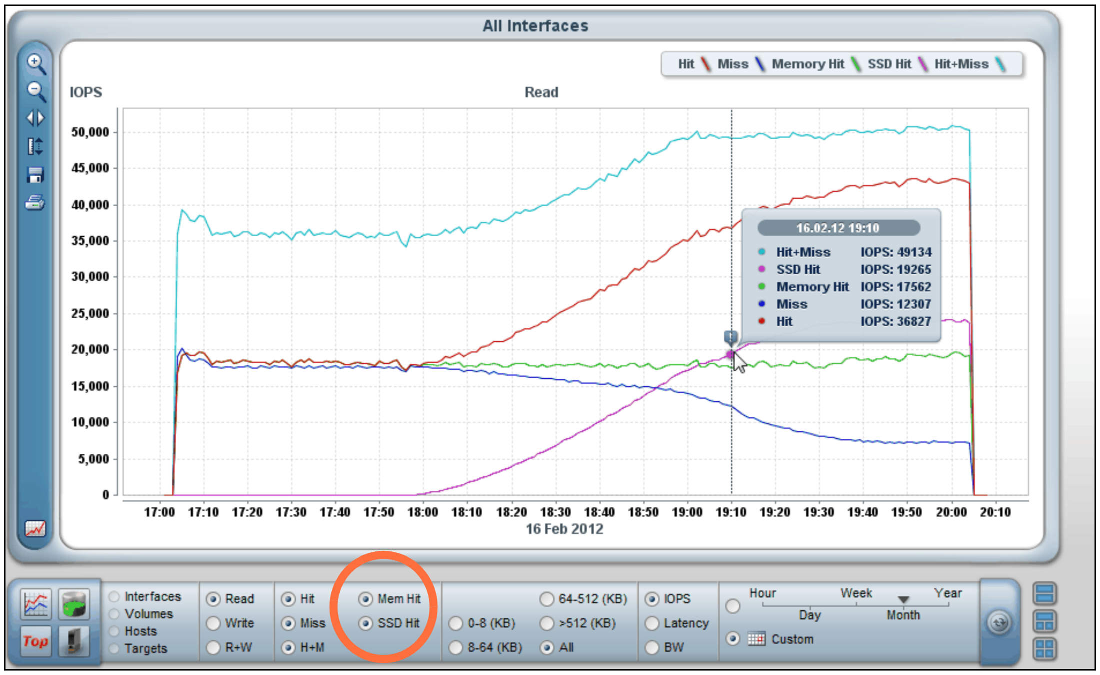
Tip: If volumes are part of a consistency group, they do not all need the same SSD Caching setting.

6.1.3 SSD Caching performance statistics

The Performance Statistics view (Figure 6-15 on page 47) was updated to reflect support for SSD Caching. Two new metrics (and filters) are included for read I/O transactions:

- Mem Hit parameter, which indicates read I/O from the main DRAM cache
- SSD Hit parameter, indicates read I/O from the extended SSD cache

Figure 6-15 illustrates a common trend when you enable SSD Caching. Total Memory Hit (red line) goes up thanks to SSD Caching, Read Miss (blue line) goes down, and SSD Hit (purple line) goes up proportionally.



16.02.12 19:10

Hit+Miss	IOPS: 49134
SSD Hit	IOPS: 19265
Memory Hit	IOPS: 17562
Miss	IOPS: 12307
Hit	IOPS: 36827

Figure 6-15 Performance Statistics view showing the Mem Hit and SSD Hit parameters

For more information about SSD performance, see Chapter 4, “Workload characteristics” on page 19.

6.2 SSD Caching management with XCLI

New commands are included in the XCLI that match the possible actions described for the GUI. For example, the **help search=ssd** command shows a list of all commands related to SSDs (Example 6-1).

Example 6-1 List of commands related to SSDs

```
XIV 1310039 Coruscant>>help search=ssd
Category   Name                        Description
system     ssd_caching_disable         Disables Flash Caching
system     ssd_caching_enable          Enables SSD Caching
system     ssd_list                    Lists SSDs used as flash cache in the
                                         system
system     vol_default_ssd_caching_get Gets the Default State of the SSD Caching
system     vol_default_ssd_caching_set Sets a Default State for SSD Caching
system     vol_ssd_caching_set         Overrides the Default SSD Caching State
                                         for a Volume
```

The **ssd_caching_disable** and **ssd_caching_enable** commands listed in Example 6-1 are restricted commands that can be used only with authority from an IBM service representative. The IBM service representative uses these commands to bring SSDs online when they are phased into the XIV system at installation time. For more information, see Chapter 5, “Installing and upgrading solid-state drives” on page 31.

Use the **`ssd_list`** command to access a list of SSDs that are used as flash cache in the system. Example 6-2 shows the results of using this command.

Example 6-2 List of SSDs

```
XIV 1310039 Coruscant>>ssd_list
Component ID   Status Currently Functioning   Capacity   Target Status   Vendor   Model                 Size   Serial     Firmware   Fru      Group   Temperature
1:SSD:3:1      OK       yes                   512GB                      XIV      MTFDDAA512MAR-1KAAB   488378 0200BB81  MA40       99Y0720           42
1:SSD:2:1      OK       yes                   512GB                      XIV      MTFDDAA512MAR-1KAAB   488378 0200BB78  MA40       99Y0720           40
1:SSD:6:1      OK       yes                   512GB                      XIV      MTFDDAA512MAR-1KAAB   488378 0200BB71  MA40       99Y0720           47
1:SSD:4:1      OK       yes                   512GB                      XIV      MTFDDAA512MAR-1KAAB   488378 0200BB95  MA40       99Y0720           47
1:SSD:1:1      OK       yes                   512GB                      XIV      MTFDDAA512MAR-1KAAB   488378 0200BB7A  MA40       99Y0720           39
1:SSD:5:1       OK       yes                  512GB                       XIV     MTFDDAA512MAR-1KAAB   488378 0200BB7D  MA40        99Y0720          43
```

The **`vol_default_ssd_caching_get`** command checks the default SSD Caching setting, either enabled or disabled, as shown in Example 6-3.

Example 6-3 Checking the SSD Caching state

```
XIV 1310039 Coruscant>>vol_default_ssd_caching_get
Command executed successfully.
default=enabled

XIV 1310039 Coruscant>>vol_default_ssd_caching_get
Command executed successfully.
default=disabled
```

If the default status is enabled, SSD Caching is *enabled* on all the volumes in the system unless manually changed. Otherwise, if the default status is *disabled*, SSD Caching is disabled for all the volumes in the system.

Options for an enabled state: With the SSD Caching state enabled, you can choose to explicitly disable any volume that must not be included in the extended caching.

To change the SSD Caching state globally for the system, issue one of the following commands:

- Set the SSD Caching to enabled:

  ```
  vol_default_ssd_caching_set default=enabled
  ```

- Set the SSD Caching to disabled:

  ```
  vol_default_ssd_caching_set default=disabled
  ```

Use the **`vol_ssd_caching_set`** command to set the SSD Caching status for a specific volume and eventually override the system default setting. You must specify the **`vol`** and **`state`** parameters when issuing the command, as shown in Example 6-4.

Example 6-4 SSD Caching set for a volume

```
XIV 1310039 Coruscant>>vol_ssd_caching_set vol=Res_Fra_Vol_01 state=enabled
Command executed successfully.

XIV 1310039 Coruscant>>vol_ssd_caching_set vol=Res_Fra_Vol_01 state=disabled
Command executed successfully.
```

You can also use the **vol_list** command with the **-x** flag and the **vol** parameter to view all of the volume properties. The properties now include two additional values, `ssd_caching` and `use_ssd_caching_default` (Example 6-5).

Example 6-5 State of a volume

```
XIV 1310039 Coruscant>>vol_list -x vol=Res_Fra_Vol_02
<XCLIRETURN STATUS="SUCCESS" COMMAND_LINE="vol_list -x vol=Res_Fra_Vol_02">
  <OUTPUT>
          <volume id="bec1240012f">
              <creator value="itso"/>
              <creator_category value="storageadmin"/>
              <id value="bec1240012f"/>
              <name value="Res_Fra_Vol_02"/>
              <size value="17"/>
                ............
                ............
             <ssd_caching value="disabled"/>
              <use_ssd_caching_default value="yes"/>
          </volume>

  </OUTPUT>
</XCLIRETURN>
```

The value of **use_ssd_caching_default** parameter indicates whether the volume follows the default system state for SSD Caching:

- If the value is `yes`, the volume follows the default system state for SSD Caching.
- If the value is `no`, the volume does not inherit the default setting for SSD Caching. If the global system setting for caching is changed, the volume keeps its current `ssd_caching` value.

7

Monitoring and alerting

This chapter highlights the additional information that the IBM XIV Storage System can provide about installed solid-state drives (SSDs). It includes information about performance and basic operational status. It also provides information about alerts that were added to the XIV system software.

This chapter includes the following sections:

- Monitoring SSDs
- Alerts

7.1 Monitoring SSDs

In version 3.1 of the XIV Storage Management GUI and XIV command-line interface (XCLI), you can view specific performance information about installed SSDs. You can also differentiate between DRAM cache parameters and SSD cache parameters. Figure 7-1 shows the Mem Hit and SSD Hit parameters.

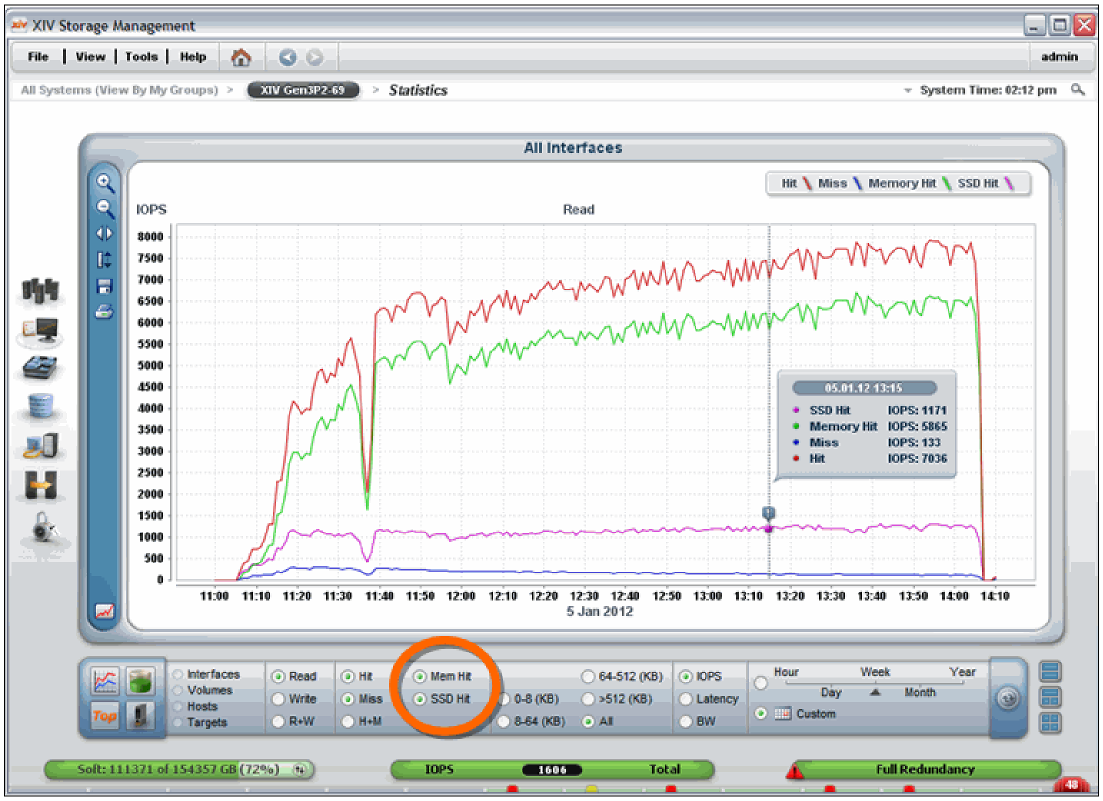


Figure 7-1 Mem Hit and SSD Hit parameters

Regardless of the other parameters that are being examined, the Mem Hit and SSD Hit parameters are valid only for read data.

You can enable or disable SSD Caching at the system level or on a volume-by-volume basis. To see the status of the SSD Caching for each volume, when using the XIV GUI, follow the steps in 6.1.2, “Setting SSD Caching” on page 40. Alternatively, when using the XCLI, see 6.2, “SSD Caching management with XCLI” on page 47.

7.2 Alerts

With adding SSDs comes the requirement to generate alerts about those SSDs. Starting with version 11.1.0, the XIV system software adds the following events about SSDs:

COMPONENT_TEST_OF_SSD_HAS_FAILED
A major event that is seen only when IBM service personnel are working on the XIV.

SSD_CACHING_DISABLED
An informational event that is generated when IBM service personnel issue the command to no longer use SSDs.

SSD_CACHING_ENABLED
An informational event that is generated when IBM service personnel issue the command to start using SSDs.

SSD_COMPONENT_TEST_STARTED
An informational event that is seen only when IBM service personnel are working on the XIV.

SSD_HAS_FAILED
A major event that is issued when the module detects that the SSD is no longer functioning.

VOLUME_SET_DEFAULT_SSD_CACHING
An informational event that is issued when the default state for all volumes is changed. The Description column shows if the action taken was to set the default to *enabled* or *disabled*.

VOLUME_SET_SSD_CACHING
An informational event that is generated when the status of a single volume is changed. The Description column shows if the action taken was to enable or disable SSD Caching for that particular volume.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM XIV Storage System: Architecture, Implementation, and Usage*, SG24-7659
- *IBM XIV Storage System: Copy Services and Migration*, SG24-7759
- *IBM XIV Storage System: Host Attachment and Interoperability*, SG24-7904

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *IBM XIV Storage System Planning Guide,* GC27-3913
- *IBM XIV Storage System: Product Overview*, GC27-3912
- *IBM XIV Storage System User Manual*, GC27-3914
- *IBM XIV Storage System XCLI Utility User Manual*, GC27-3915

Online resources

These websites are also relevant as further information sources:

- IBM XIV Storage System Information Center

  http://publib.boulder.ibm.com/infocenter/ibmxiv/r2/index.jsp
- IBM XIV Storage website

  http://www.ibm.com/systems/storage/disk/xiv/index.html
- System Storage Interoperability Center (SSIC):

  http://www.ibm.com/systems/support/storage/config/ssic/index.jsp

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Solid-State Drive Caching in the IBM XIV Storage System

Understand the architecture of solid-state drive (SSD) Caching

Learn about GUI and XCLI support and monitoring and alerts for SSDs

View workloads that can benefit from SSD Caching

This IBM Redpaper publication provides information about the implementation and use of solid-state drives (SSDs) in IBM XIV Storage System XIV Generation 3 (Gen3), running XIV software version 11.1.0 or later. In the XIV system, SSDs are used to increase the read cache capacity of the existing DRAM memory cache, and not used for persistent storage.

This paper begins with a high-level overview of the SSD implementation in XIV and a brief review of the SSD technology, with focus on the XIV storage system. It explains the SSD Caching design and implementation in XIV. Then it examines typical workloads that can benefit from the SSD Caching extension and introduces the tools and utilities to help you analyze and understand the workload. In particular, it highlights the block tracing facility that was designed and developed by IBM Research.

Then this paper explains the process that authorized IBM services representatives use to install SSD Caching. It reviews the changes made to the XIV GUI and the XCLI to support SSD Caching. Finally this paper provides a listing of the new alert-generating events and monitoring options that are provided for SSD support.

This paper is intended for users who want an insight into the XIV SSD Caching implementation and architecture, its capabilities, and usage. For more information about the XIV Storage System, see the IBM Redbooks publication, *IBM XIV Storage System: Architecture, Implementation, and Usage*, SG24-7659.

REDP-4842-00