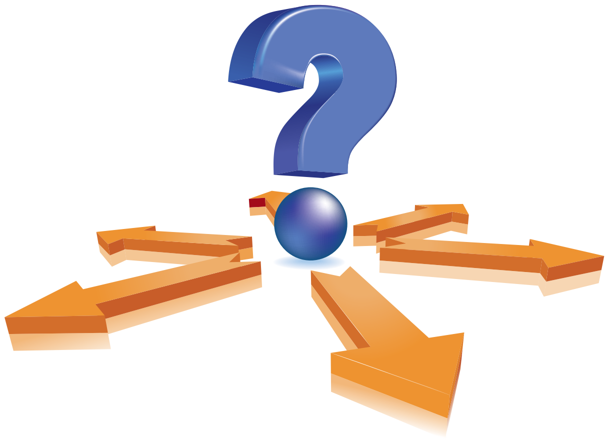
EBOOK



SSD Deployment Guide

Factors You Must Consider When Deploying SSD

VeloBit

Faster Storage Made Easy

Share this ebook

Start ▶

INTRODUCTION

Although relatively low in capacity, solid-state (flash) storage provides extremely high input/output per second (IOPS) performance that can potentially solve most storage I/O related challenges in the data center. There is great excitement around solid state disks (SSD) because they offer great promise for performance improvement and power savings. But those planning an SSD deployment face a bewildering variety of deployment choices. Moreover, choosing poorly can yield large expense, disruption to existing storage environments, and a complex new set of storage management tasks.

This SSD deployment note, provided by VeloBit, is intended to help storage and application storage administrators make wise decisions. The note summarizes the choices available, and outlines some of the pros and cons of different deployment options.

VeloBit is a provider of plug & play SSD caching software that dramatically accelerates business applications at a remarkably low cost. Because we are a vendor, we have a bias. We think deploying SSD as a cache rocks, and that a VeloBit—powered SSD cache is the best solution for most customers. Despite our bias, we have tried to put together an informative, objective, and vendor neutral resource. Occasionally, however, we mention the features and benefits of our product. To make these "commercial breaks" clear, we identify them using

CONTENTS

HOW TO SELECT THE SSD DEPLOYMENT METHOD

In selecting which SSD path is right for you, there are two basic questions:

- Which SSD hardware form factor? Should the SSD be directly attached to the application server, or reside in an array?
- What data should go on the SSD, and how will you get it there?

These choices are summarized in table below.

		Method for Placing Data on SSD		
		Primary Storage	**Data Tier**	**Cache**
SSD Form Factor	Server Attached			VeloBit Faster Storage Made Easy
	Array-Based			VeloBit Faster Storage Made Easy

Table 1 — SSD Deployment Models

Storage and application managers must consider a few factors in making their decision:

- Performance
  - ☐ Which solution delivers the highest performance for my application?
- Cost
  - ☐ What does the solution cost to acquire?
  - ☐ What is the cost of ongoing management and maintenance?
  - ☐ Will the SSD solution complement my current storage environment?
- Operating cost and complexity
  - ☐ What are the system, process & cost for protecting data?
  - ☐ What will it cost to install, manage and maintain the SSD on an ongoing basis?
- Disruption
  - ☐ Will the SSD solution require new backup and data management systems and processes?
  - ☐ Will the solution work with the applications and storage I have in place?
- Reliability and Data Protection
  - ☐ Will my data be safe in the new solution?
  - ☐ Will the system be reliable?
- Ease of data sharing
  - ☐ If my application requires data sharing between servers or instances, will the SSD implementation support that?

Factors Influencing SSD Deployment Choice

- Performance
- Cost
- Operating cost and complexity
- Disruption
- Reliability and data protection
- Ease of data sharing

SSD DEPLOYMENT CONSIDERATION 1 — SERVER-SIDE VS. SSD-IN-AN-ARRAY?

The first question is whether to deploy SSD in a server or in a storage array. In weighing this tradeoff, it is easiest to consider first the case where the SSD will be used as a primary storage volume. In such systems, the SSD holds the primary or only copy of the data.

VeloBit is a software only solution that installs in the application server. It works well with both server-side SSD or array-based SSD.

SSD in Server

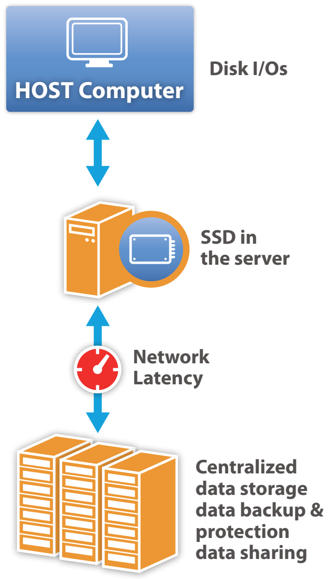
Highest Performance and the Best Price-Performance

Deploying an SSD in the server typically offers the highest performance and the best price-performance. Placing SSD in the server moves data close to the processor, lowering data access times because the latencies of a network and storage network protocols are avoided. As an example, a PCIe card in a server delivers data at speeds from 10 to 20 microseconds. Traversing a storage network takes at least ten times as long.

Lower Cost

With some notable exceptions, a server-side SSD can be less expensive because server-side devices may not require high-end features found in array-based systems such as RAID, snapshots, etc. For these reasons, server-side SSD's typically offer the most "bang for the buck".

SSD In the Server


HOST Computer
Disk I/Os
SSD in the server
Network Latency
Centralized data storage data backup & protection data sharing

Share this ebook

More Granular and Gradual Roll-Out of SSD

The server-side SSD also allows you to pinpoint your SSD deployment to application hot spots. In most enterprises, only a subset of applications needs SSD-level performance. Deploying SSD on only the servers that require high-end performance allows a more granular, and therefore less expensive, roll-out of SSD. Linking SSD purchases to individual servers or applications also facilities a “utility charge-back model”, where only the application users who need higher performance are asked to pay for it. In addition, deploying server-side SSD allows a more gradual start to your SSD roll-out, as you can start with just a few systems and then “pay as you grow” as application needs (and your budget) expand.

Scale Out Capability to Storage I/O Performance

Installing SSD in servers also provides a scale out capability to storage I/O performance. Scaling I/O performance is achieved by simply adding more servers to your network. Each server gets its own SSD cache and there’s no bottleneck at a centralized storage system. This fits the scale out model that is common in virtualized data centers and enterprise cloud environments.

Preserve Your Existing Storage Arrays

Finally, deploying SSD on the server side offers a way to offload — and preserve — your existing storage arrays, extending the life of the infrastructure that is already in place.

There are two primary drawbacks to SSD in the server, both of which have to do with moving from a centralized (and shared) storage model to a decentralized model in which the primary data copy resides in a single server. Both drawbacks can be addressed with Data Tiering or Caching — as we will discuss later.

VeloBit allows server-side SSD to be used as a cache. When SSD on the server is used as a cache, the primary copy of data is kept in your centralized storage, allowing you to leverage your current back-up and data protection systems.

Complexity of Managing and Protecting Data

A first drawback, and the largest issue, relates to the complexity of managing and protecting data in a decentralized model. Most organizations already have well-engineered, centralized storage systems in place that have been tuned to ensure that data is safe, and to minimize operating complexity and cost. Using a server-side SSD as primary storage means that the main copy of data moves from the centralized storage system out to the server. In this decentralized model, the server-side SSD is a single-point-of-failure. If the SSD fails, data is lost.

If your application can tolerate loss of data in a single server, this may not be an issue. But for most applications "data decentralization" requires the creation of new back-up systems. In addition, most server-side SSDs do not support a full set of data protection services (snapshops, replication, etc.) or, if they do, the cost to implement high availability is very high high.

Data Sharing

A second drawback relates to data sharing. If the primary copy of data is held in the server-side SSD (i.e., the SSD is "primary storage"), it is difficult to share the data amongst servers. For applications where data sharing is not needed, or where data sharing is done at a higher level in the application stack, this may be unimportant.

The drawbacks of server-side SSD may not apply in all applications. Applications that are cluster oriented, as an example, may not require data sharing. And in some applications, data loss is acceptable or data is replicated amongst servers at a higher level in the stack. But for most applications, using server-side SSD as primary storage is not an effective option. In these cases, a data tiering or caching solution can be used in combination with server-side SSD to resolve the challenges.

SSD in Array

Data Protection and Data Management

When an all-SSD storage array is used, the customer retains the benefits of centralized storage. Data protection/data management tasks are easier to perform.

Data Sharing

With an array, data can be shared amongst applications more easily. In addition, the high performance of SSD can be made available to all servers attached to the array.

High Cost

The biggest issue for customers deploying SSD as a primary store is cost. According to Forrester, SSDs can be up to 10 times more expensive than hard drives. At this cost premium, few customers can afford to use an SSD for all of their data. Placing all of your data on an SSD, as in the "all or nothing" SSD array model, is not a viable option.

Performance

In addition, because of the network lags outlined above, SSD arrays may offer less performance for the buck than server-side SSD.

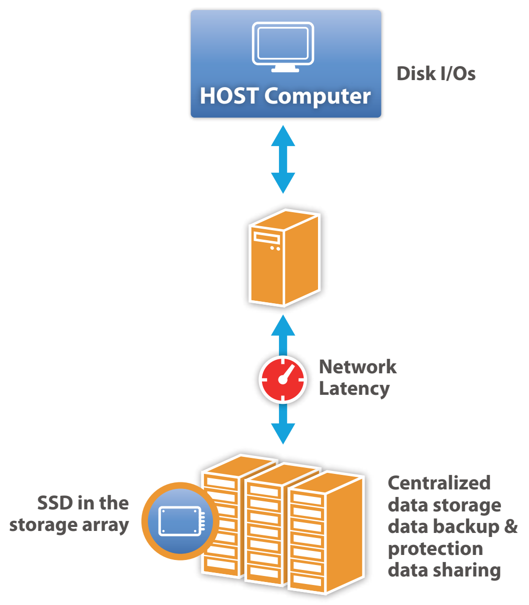
Disruption

Deploying an all-SSD array creases disruption, as data must be moved from the array you have in place to the new array. This forklift upgrade is expensive as well.

Capital Commitment and Changes to Operations

Finally, purchasing a new SSD array requires a very large commitment — both in capital expense and in changes to operations — in comparison to the "pay as you grow" approach of server-side SSD.

SSD In the Storage Array


HOST Computer
Disk I/Os
Network Latency
SSD in the storage array
Centralized data storage data backup & protection data sharing

SSD DEPLOYMENT CONSIDERATION 2: TIERING VS. CACHING?

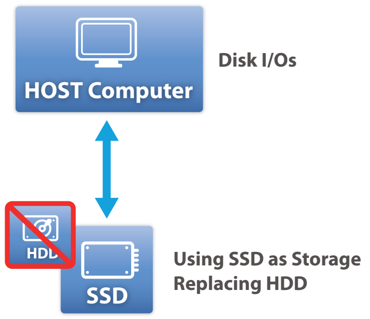
For the vast majority of customers, SSD's are too expensive relative to disk to warrant using them as primary storage for all applications. Further, many customers who choose to deploy server-side SSD need a mechanism to move or copy data from the server to a centralized pool where it is easier to protect and share. As a result, for most users a "hybrid" solution that mixes SSD (either on the server or in the array) with disk is appropriate. To implement hybrid storage, users must have a mechanism to decide which data to put on an SSD, when to place it there, and how to move the data between SSD and disk. There are two classes of solution to this data-movement / data-placement problem — tiering and caching.

Tiered Storage

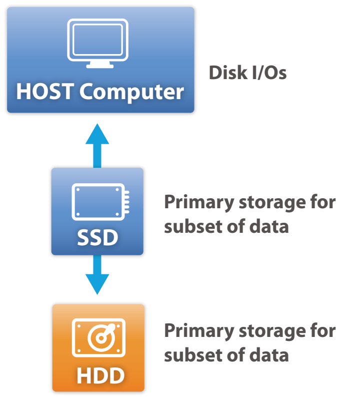
Balance High-Performance/High Cost SSD with High-Capacity/ Low Cost HDD

In a tiered storage solution, SSD is used as a top tier of storage. Tiered storage balances the high-performance / high-cost SSD capabilities and the high-capacity / lower-cost aspects of hard drives. The idea is to put frequently accessed ("hot") data on high performance SSDs and migrate less frequently accessed ("cold") data to low-cost disk drives. In a tiered storage system the SSD is typically in an external storage array but vendors are beginning to introduce products that extend tiering to include a server-side SSD. In a tiered storage system, there is one copy of the primary data, and that copy resides on the storage element to which it has been assigned.

SSD as Primary Storage



SSD as a Tier


HOST Computer
Disk I/Os
SSD
Primary storage for subset of data
HDD
Primary storage for subset of data

Complexity of Data Tiering

In tiering, the user must determine which data to place on the SSD and when to place it there. The user must take a specific action to move the data to the SSD, and the application that uses this data must then be told its correct location.

Determining what data to place on the SSD can be difficult. Sometimes the choice is obvious. Database indexes and busy database tables are examples of data elements that should be "hot" enough to warrant placement on SSD. More often, the decision does not have a fixed answer. Which data are "hot" changes constantly, requiring ongoing monitoring and data movement as application needs change. For example, the reports finance runs may put great demands on the storage on the last day of the month, but that data could sit dormant for the rest of the period. For storage administrators, continually monitoring data usage patterns and rearranging data can be a time-consuming and expensive chore.

To address the complexity and burden of manual data placement, storage vendors are introducing new products for automated data tiering. These solutions automatically move sections of data to SSD as the sections become hot and then demote them as they become less active. While this strategy can function, it has problems.

Implementation Cost

First, the storage system must support automated tiering. For many customers, this requires an expensive upgrade to a new storage system.

SSD Wear

Second, the granularity of sections of data that are moved can be large. As a result, data movement to and from the SSD tier can create a heavy write load on the SSD, wearing the SSD and potentially causing the SSD to fail. The data movement also puts a load on the storage controller, impacting performance for other activities.

Inefficiency of Tiering Software Algorithms

A fourth challenge is that the decisions used to promote data are backward looking. Data that was hot recently is not necessarily going to be hot again.

Vulnerability of Data / Data Protection Cost

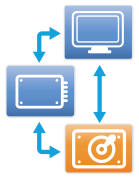
A significant issue is that since in data tiering the SSD tier holds the primary copy of data, failure of the SSD causes loss of data. As a result, the SSD tier must be protected through RAID-like schemes. These schemes add computational overhead that impacts performance, and require extra SSD capacity. The increased capacity makes an expensive storage media even more costly. Finally, if the SSD in a tiered system is deployed on the server, data protection is complicated to implement as the server represents a single point of failure.

SSD as a Cache

Caching is a technique where a controller - whether it's software, a RAID controller inside a server or a high-end external disk controller - uses SSD as a cache in front of disk storage. As in tiering, the caching controller identifies "hot data" and stores it on SSD. *A major difference, and one that drives many advantages for SSD cache, is that in caching the SSD holds a copy of the data rather than the primary copy.*

Before the SSD era, caching implementations typically meant using a small amount of memory to accelerate disk access. The cache typically held only very recently accessed data — perhaps a few minutes worth. The cache hit rate was relatively low, meaning that many requests were served from mechanical disk and performance improvements were somewhat limited. With the falling cost of SSDs, a larger cache can be created, increasing the cache hit rate. It also means that data can be in cache for much longer.

SSD as a Cache



A big difference between cache and automated tiering is that the data in cache is always a second copy of the data that is on the hard drive. An SSD cache leaves primary data in your pre-existing storage array. This has several advantages.

Reduced Implementation Complexity and Disruption

First, failure of the cache rarely produces a data loss. As a result, the cost and complexity of implementing data protection for the SSD tier (either on the server or in the array) is avoided. Second, there is no change to data protection or data management, so the complexity and disruption of implementing an SSD-specific data protection solution is reduced. An SSD cache works with your existing disk storage, boosting its performance. This may enable you to delay or avoid expensive disk array upgrades.

Ease of Use

An SSD cache can also be simpler to operate than a tiered storage solution. While there may be tuning opportunities, in general an SSD cache automatically chooses which data to hold in SSD. There is no need for an administrator to decide what data should be placed on it. In addition, caching is more likely than tiering to accelerate “hot” data in real-time, since the hot data is likely to be held in cache. In a tiering system, there can be a delay as data that recently became hot is moved from the disk tier to the SSD tier.

Lower Cost

Additionally, when SSD cache is implemented in a server, it offloads the network. It also alleviates processing load from your storage array, improving I/O performance for other workloads and increasing the length of time before an expensive storage array upgrade is required. SSD cache is typically pinpointed at one server or application, allowing a granular, gradual, and less expensive introduction of SSD’s.

VeloBit compresses data as it is cached, dramatically increasing the effective cache size. This boosts the cache hit rate and yields large gains in performance.

VeloBit software is deployed in a server, and utilizes server resources for processing. However, it can be used with an SSD volume attached in any way, including SSD served from an array.

In summary, SSD caches — particularly when deployed in a server — offer advantages in performance, cost, ease of use, administration costs, and in offloading an array. They deserve strong consideration in most environments.

Data Sharing

However, there are three considerations in deploying an SSD cache. The first issue is "cache coherence" and the impact on data sharing. Caches can be used to hold both read and write operations (a "write-back" cache), or they can hold only read data while passing write requests directly through to disk (a "write-through" cache). When a write-back cache is used, writes may reside only in cache until the cache is flushed to disk. Consider the case of two applications, on two different servers, that share data on an array. In this scenario, the two applications will not have a consistent view of the data in the period between when data is written to a cache and the cache is flushed. Data consistency can be achieved by using the SSD as a write-through cache, but then only read operations are accelerated.

Vulnerability of Data in a Write Cache

A second consideration is the vulnerability of data in a write cache. If a write is held in a server-side write-back cache (and not yet committed to disk), the data in cache is lost if the cache hardware fails. If the cache hardware is reliable and persistent, the vulnerability is small. But if not, the "depth" of the write cache must be limited or the cache must be a write-through cache.

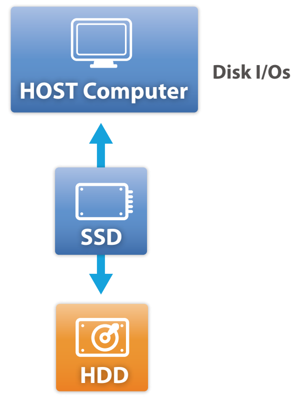
VeloBit software can be used for write-through (read) or write-back (read/write) caching. When used as a write-back cache, VeloBit allows the user to control the depth of the cache, limiting how long write data is held before it is committed to disk. VeloBit supports all "flush" and "sync" commands issued by an operating system or an application.

SSD Write Load (Depending on Cache Implementation)

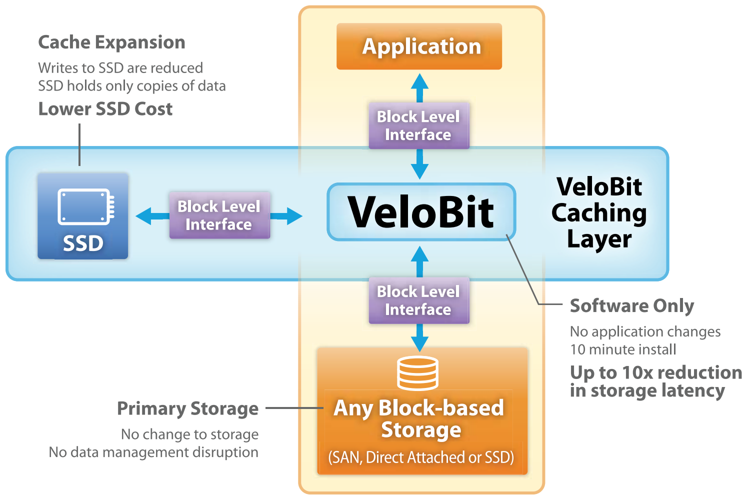
A third consideration is the write-load inherent in some caching implementations. In typical caching systems, the cache is deployed in a "vertical" architecture with the SSD in front of disk (see Figure 1, below). In this configuration, most or all of the data passes through the cache before it is written to disk. This can put a heavy write-load on the SSD, subjecting it to excessive wear and shortening its life. In addition, write performance can degrade if the SSD reaches the "SSD Write-Cliff" due to excessive garbage collection.

VeloBit software includes a number of features that combine to dramatically lower the write load on an SSD. It uses a "horizontal" architecture, so that not all writes pass through the SSD before written to disk. It compresses data that is written to the SSD, reducing the volume of writes. Additionally, a variety of algorithm optimizations including the Conservative Insertion Protocol lower the write load to SSD and decrease the frequency of garbage collection.

Vertical Architecture SSD Cache



VeloBit SSD Cache Architecture



Share this ebook

Warm Up Period for SSD Caches

A final consideration is the "warm-up period" for SSD caches. An SSD cache delivers its highest performance once it has been populated with enough "hot" data to yield a high cache hit rate. There are select cases where an administrator can predict that a data set will be "hot" in advance of it being cached. An example would be reports that are run infrequently, but at a regular interval — as in a monthly financial analysis. In these cases, a manual move of the data — as in a tiering system — would enable data to be pre-positioned in the SSD more quickly than an SSD caching solution.

VeloBit uses content locality caching which caches blocks based on the popularity of their contents, in addition to how recently or frequently the blocks were accessed. As a result, the VeloBit HyperCache has a shorter warm-up period in many cases.

CONCLUSION

Solid state storage offers great promise for I/O performance improvement and power savings. Each SSD deployment method offers advantages and disadvantages, depending on the application and operating environment. There are two main considerations when determining the SSD deployment method:

1) Should the SSD be directly attached to the application server, or reside in an array?

Deploying the SSD in the storage array simplifies data protection and data sharing but is expensive, has high operating costs, and offers lower performance compared to deploying the SSD in the server. Deploying SSD in the server offers higher performance and lower deployment costs. However, data sharing and data protection are more complex unless a caching solution is used.

2) Should the SSD be deployed as primary storage, a storage tier, or a cache?

Deploying SSD as primary storage offers the simplest implementation and management but is disruptive and expensive. SSDs can be up to 10 times more expensive than disk-based solutions and changing primary storage is a disruptive process. SSD as a storage tier is a cost effective solution, but is complex to deploy and operate. An SSD cache is inexpensive, non-disruptive, and simple to operate. Caches, however, limit data sharing, may increase data vulnerability (when used as write cache), and could increase SSD wear.

Table 2 summarizes the advantages and disadvantages of the main SSD deployment choices:

Decision Criteria	Array			Server		
	Primary	Tier	Cache	Primary	Tier	Cache
High Performance	◐	◐	◐	●	●	●
Low Cost	○	○	○	◐	●	●
Low operating cost	○	○	○	●	●	●
Low Complexity	●	○	●	●	○	●
Low Disruption	○	○	●	○	○	●
High Reliability and Data Protection	●	◐	●	◐	○	◐/●*
Ease of Data Sharing	●	●	●	○	○	○

**depending on implementation*

VeloBit provides plug & play SSD caching software that dramatically accelerates databases and other business applications at a remarkably low cost. VeloBit uses Solid State Disk (SSD) to create a transparent application acceleration layer that boosts performance by 10x. VeloBit employs Content Locality Caching technology which overcomes most limitations of traditional caching algorithms. VeloBit increases SSD performance, reduces SSD write load and SSD wear, and has a shorter 'warm up' period than other caches. The software installs seamlessly in less than 10 minutes and automatically tunes for fastest application speed. VeloBit deploys and operates transparently to existing applications, storage, or data management.

Learn more about VeloBit

Free Trial

See a Demo

Share this ebook