This book will guide you through tuning, tweaking, and optimizing vSphere in order to improve the performance of your systems. It will also help you identify and fix common performance bottlenecks.

Starting with planning high-performance infrastructure setups, the book covers the core concepts of cluster setups, along with CPU virtualization concepts. Next, it covers the management of physical and virtual memory mechanisms, swapping, and ballooning.

Moving on, you'll get an understanding of how vCenter Server can assist you with recognizing performance issues, and how to utilize advanced technologies such as Storage I/O Control and vSphere Flash Read Cache to tune storage resources.

Finally, the book shows you how esxtop and resxtop can provide advanced metrics to detect performance issues, and how you can optimize storage and network resources inside virtual machines.

Who this book is written for
If you're a system administrator who has some experience with VMware vSphere but would like a quick guide to be able to identify and fix performance issues in your vSphere systems, then this book is for you.

What you will learn from this book
- Choose the right hardware and server components suitable for high-performance scenarios
- Adopt availability and automatic load balancing in your virtual infrastructure
- Understand and monitor CPU and memory resources
- Monitor performance on the vCenter Server, ESXi host, and virtual machine levels
- Configure storage resources to achieve greater performance
- Incorporate latency and performance sensible networking
- Tune and tweak common guest operating systems

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 6 'Performance Monitoring with esxtop'
- A synopsis of the book’s content
- More information on vSphere High Performance Essentials
**About the Author**

*Christian Stankowic* has been working as a certified IT engineer for five years. His interest in Linux and virtualization technology picked up in 2006, when he gained his first experience with openSUSE, Debian, Ubuntu, CRUX Linux, and VMware Server. Since his apprenticeship as an IT specialist for system integration, Red Hat Enterprise Linux, CentOS, Red Hat Satellite/Spacewalk, Nagios/Icinga, and SUSE Linux Enterprise Server became his key qualifications. Recent VMware technologies such as vSphere and the vCenter family are part of his daily work as a virtual infrastructure administrator. He loves to share knowledge. Besides acting as trainer in the company he is working for, he also started writing books to contribute more. He was the coauthor of the book *VMware vSphere Essentials*, Packt Publishing. In his free time, he works on his personal IT blog and develops web applications using recent techniques such as jQuery and PHP. You can check his blog about various IT topics at [http://www.stankowic-development.net](http://www.stankowic-development.net).
Preface

VMware vSphere is all the components and operating system that come together to make VMware’s enterprise virtualization platform unique. When running enterprise-driven application workloads, the VMware vSphere family is the preference of many customers and developers. The vSphere platform is designed to work with numerous servers and infrastructure core components to create virtualized platforms or complete cloud computing fabrics.

This book will guide you through tuning, tweaking, and optimizing vSphere in order to improve the performance of your system. It will also help you to identify and fix common performance bottlenecks.

What this book covers

Chapter 1, Hardware Design Concepts and Best Practices, introduces aspects that are essential to plan virtual infrastructure setups that are suitable for high-performance applications. It will teach you how to choose the right hardware components in accordance with our particular application needs. Also, it will take a deeper look at how to plan a resilient and scalable network setup.

Chapter 2, Cluster Optimization Approaches, takes you through all the possibilities to improve our cluster design, how scale-out setups differ from scale-up alternatives and understand how to choose a concept that is suitable for our requirements.

Chapter 3, Understanding CPU Resources, explores how vSphere utilizes CPU resources and how to optimize our VM configuration in order to achieve greater performance. It will also take a deep look at NUMA and vNUMA.

Chapter 4, Understanding Memory Resources, covers memory resource essentials and how reclamation mechanisms such as page sharing, swapping, ballooning, and compression play a role in workload performance.
Chapter 5, Performance Monitoring with vCenter Server, explains how vCenter Server helps us with investigating our virtual infrastructure's performance. It will teach how statistic metrics can assist us with performance analysis and how to alter them to fit our needs. You will also go through some typical performance issues and possible approaches to get rid of them.

Chapter 6, Implementing High-Performance Storage, covers implementing high-performance storage. You will understand how misalignment can harm our infrastructure's performance. Later, we will focus on optimizing storage resources.

Chapter 7, Implementing High-Performance Networking, focuses on high-performance networking and teaches how to optimize our networking design for more throughput and less latency. After reading this chapter, you will know how technologies such as Network I/O Control and network offloading can assist us with achieving this goal.

Chapter 8, Performance Monitoring with esxtop, teaches about esxtop, an advanced tool to monitor performance at the ESXi host level. It will help you to understand how to make use of the various metrics to analyze performance issues.

Chapter 9, Guest Operating System Performance Monitoring and Tweaking, investigates monitoring and tweaking performance of VMs. It will cover some of the important topics such as storage provisioning types, virtual storage controller types, generic storage tips, virtual network cards and drivers, common file system tweaks, and so on.
Implementing High-Performance Storage

In the last chapter, you learned how to make use of vCenter Server performance charts in order to investigate performance issues. You also learned about some typical performance issue types and how to fix them. In this chapter, we will study about implementing high-performance storage. You will understand how misalignment can harm our infrastructure's performance. Then, we will focus on optimizing storage resources.

In this chapter, we will investigate the following topics:

- General storage recommendations
- Tuning SAN resources
- Optimizing NFS resources
- Utilizing virtual flash resources
- Storage I/O Control
- Storage DRS
General storage recommendations

The most crucial aspect when implementing storage is to define the storage performance. When designing a shared storage, the goal is to find a perfect mix among throughput, Input/Output Operations per second (IOPS), latency, and capacity—we’re also talking about the isolation and consolidation approaches. Refer to the following table for an overview about both the approaches:

<table>
<thead>
<tr>
<th></th>
<th>Isolation</th>
<th>Consolidation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore capacity</td>
<td>Smaller</td>
<td>Bigger</td>
</tr>
<tr>
<td>VM/storage</td>
<td>Less VMs per datastore</td>
<td>More VMs per datastore</td>
</tr>
<tr>
<td>Throughput</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Latency</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Bad resource sharing</td>
<td>Good resource sharing</td>
</tr>
<tr>
<td>Management</td>
<td>More effort</td>
<td>Less effort</td>
</tr>
</tbody>
</table>

Basically, there is no perfect formula to size storage units; it is rather an individual task. Both approaches have their own advantages and drawbacks. When having a high rate of virtualization in a data center, it is a common procedure to define multiple levels for workloads. Consider the following example:

- **Gold**: These are high-priority workloads with high-grade I/O requirements
- **Silver**: These are the conventional workloads, they are a mix between price and performance
- **Bronze**: These are for development and test VMs, they are cheaper storage media

The following questions might help you in finding the right capacity:

- How long would it take for you to restore all the workloads residing on the Logical Unit Number (LUN)? Keep disasters in mind—large LUN storage units can also lead to more data loss.
- Are all of your workloads I/O-intensive? If you have some I/O-intensive workloads besides conventional workloads, you could separate them. Having all workloads on one storage level might result in unbalanced performance; think about storage tiering.
• What are the requirements of your workloads? Check the application vendor's parameters and evaluating a storage concept.

When going for more LUNs, keep in mind that vSphere 5.5 and 6 offer up to 256 block-based and file-based datastores (with up to 64 TB of capacity) each. This is very important when designing cloud environments with multitenant datastore architecture!

To sum it up, keep the following in mind:

• Find a balance between performance and capacity. Focus more on performance than capacity, keep disasters in mind.
• Try to find a standard sizing for your environment; try to isolate deviating workloads in order to ensure balance.
• Keep VMFS limits in mind (maximum 64 TB capacity, maximum 256 volumes per host, and maximum 2,048 VMs).

Optimizing SAN resources
Shared storage is a very crucial component in virtualization environments, especially for high-performance scenarios. To ensure maximum performance, we need to ensure that all the participating components of a SAN concept such as switches and ESXi hosts are configured properly. In the following sections, we will focus on optimizing the SAN resources.
Hardware basics – components and cabling

When a SAN setup is chosen, it mostly needs to fulfill both performance and availability requirements. To make sure that these requirements are met, it is a common procedure to use at least two SAN fabrics and two storage arrays. A SAN fabric basically describes the interconnectivity of multiple storage switches and storage systems. To ensure both performance and availability on the ESXi host, we need to ensure that multiple HBAs are used. It is advisable to connect the particular HBAs to different SAN fabrics. Using more than four paths per volume is not recommended as it might slow down the performance due to overhead. Refer to the following graphic for a reference implementation:

When configuring iSCSI HBAs, make sure that you utilize the following technologies:

- **VMware NetQueue**: This provides multiple receive queues with interrupts processed by CPU cores that are associated with virtual machines; this significantly optimizes receive performance

- **iSCSI offload**: This offloads iSCSI intelligence on the NIC, reducing the hypervisor’s load
• **Virtual Machine optimized ports (VMOP):** This reduces the hypervisor’s load and virtual network latency as packet classification is offloaded to the NIC.

  Basically, it is a good idea to offload iSCSI and fiber-channel logic in the HBA. Most vendors offer commonly used protocols—refer to Chapter 1, *Hardware Design Concepts and Best Practices.*

Current SAN products usually support up to 16 Gbit/s for fiber channel or up to 56 Gbit/s for iSCSI or FCoE implementations. Make sure that you choose the appropriate medium for data connections.

In the storage subsystem, LUNs should be placed on RAID groups that are offering the required IOPS rates. It is a common procedure to configure multiple RAID groups that address various application's performance and availability requirements and costs (for example, slower storage for development and test and faster storage for production workloads). Even the choice of the RAID configuration can have an influence on the storage performance. Refer to the following table for commonly used RAID groups:

<table>
<thead>
<tr>
<th>RAID 0</th>
<th>RAID 1</th>
<th>RAID 5/6</th>
<th>RAID 0+1</th>
<th>RAID 1+0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration</strong></td>
<td>Data is striped on multiple disks</td>
<td>Data is mirrored on an additional disk</td>
<td>Data is striped on all disks, parity is placed on multiple disks</td>
<td>RAID 1 volume, consisting of two RAID 0 groups</td>
</tr>
<tr>
<td><strong>Disk costs</strong></td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Higher</td>
</tr>
<tr>
<td><strong>Write performance</strong></td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Read performance</strong></td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Parallel I/O operations on all participating disks</td>
<td>Data can be read parallel from any disk (fast), writes are usually executed on both disks (slow)</td>
<td>Data can be read from any disk (fast), disk writes require reading each disk for calculating parity</td>
<td>Two parallel I/O operations per RAID 0 group</td>
</tr>
</tbody>
</table>
Implementing High-Performance Storage

<table>
<thead>
<tr>
<th>RAID 0</th>
<th>RAID 1</th>
<th>RAID 5/6</th>
<th>RAID 0+1</th>
<th>RAID 1+0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Availability</strong></td>
<td>No fault tolerance</td>
<td>One disk fault is tolerated</td>
<td>One or two parity bits, allowing one or two devices to fail</td>
<td>One disk fault is tolerated</td>
</tr>
<tr>
<td><strong>Recommendation</strong></td>
<td>Do not use unless your application tolerates complete data loss.</td>
<td>Use for very critical workloads that require more reliability than performance.</td>
<td>Balance of performance and redundancy.</td>
<td>Use for workloads with high performance demands—keep in mind that you have minimum fault tolerance.</td>
</tr>
</tbody>
</table>

To enable faster I/O, most SAN systems can be configured in a way that multiple active paths can be used. Also, altering cache sizes in the storage backend or pooling RAID groups can lead to higher performance. If your SAN design allows it, think about using enterprise flash resources. Ensure that all the participating storage components (such as arrays, SAN switches, and HBAs) are supported along with your vSphere environment. Double-check the interoperability by both VMware and your storage vendor.

When using block-based shared storage with multiple paths, ESXi supports multiple algorithms:

- **Most Recently Used (MRU)**: It is mostly used for active/passive arrays. The primary path is selected during boot; it will be used until it becomes unavailable. In this case, the path is switched and will not return—even if the former path becomes available again.

- **Fixed Path (FP)**: Storage paths are set during boot and might change if paths become down. This is often selected for active/active storage systems using ALUA (for example, IBM SAN Volume Controller, IBM® Storwize, and so on). ALUA (Asymmetric Logical Unit Access) helps ESXi hosts to detect the best paths by presenting storage controllers and preferred nodes. Do not enable FP on non-ALUA storage systems!
• **Round Robin (RR):** It automatically balances the load across paths. When used along with active/passive storage systems, the preferred paths are used. In combination with active/active storage systems, all paths are utilized. For every 1,000 IOPS, another path is chosen to balance the load.

[![](image)](image) Please refer to your storage vendor's manual for virtualization recommendations.

To implement multipathing, ESXi offers a plugin architecture called **Pluggable Storage Architecture (PSA).** It also supports the integrated **Native Multipathing Plugin (NMP)** as a third-party plugin. There are two flavors of NMP plugin types, both are available with VMware and other vendors:

• **Storage Array Type Plugin (SATP)—** Storage Array Type Plugin monitors paths and reports changes, it also does path failover
• **Path Selection Plugin (PSP):** This simply selects a path from a storage system

[![](image)](image) Double-check if your storage vendor offers dedicated NMP plugin or configuration recommendations!

To sum it up, keep the following in mind when optimizing SAN setups:

• Use at least two SAN fabrics and HBAs per ESXi host—even if you’re utilizing dual-port HBAs. Connect the particular HBAs to different fabrics.
• Choose an appropriate RAID level when designing a shared storage for your virtualization environment. Also, make sure that you choose the right RAID group when presenting LUNs to ESXi hosts.
• Use multiple paths in parallel in order to enable faster I/O throughput. Using more than four paths is not recommended due to high overhead.
• Check your storage vendor’s documentation for vSphere configuration recommendations.
• Ensure utilizing all your storage system's features (for example, pooling, SSD caching, and so on).
• Mirror your storage system for high availability scenarios in order to avoid single point of failure (SPOF).
Implementing larger LUNs with VMFS

When extending storage presented to the ESXi hosts, it is not recommended to use extents. Every partition on a hard drive used as datastore is represented as a VMFS extent. Prior to VMFS-5, it was required to span a VMFS volume across multiple partitions in order to have more than two TB capacity. With VMFS-5, this limitation is obsolete—don't use extents as they are not needed. Keep VMFS limits in mind, refer to the VMFS comparison table in the next section for an overview.

Raw device mapping

Instead of using VMFS volumes, ESXi also supports passing through LUNs directly to VMs; this mechanism is called raw device mapping (RDM). Formerly, this feature was sometimes chosen for workloads with extreme I/O requirements. As VMFS-5 is a high-performance filesystem with less overhead, this is not needed anymore.

Check out the following blog post for a performance comparison:

In comparison to RDM's, VMFS's speed is comparable. Typical use cases of RDM's are as follows:

- **Utilizing Microsoft Cluster Service (MSCS) with shared storage**
- **Migrating applications with huge datasets to VMs (P2V) that cannot be migrated to VMDKs in a suitable time frame**

RDM's offers two operating modes, as shown in the following:

- **Pass-through mode**: ESXi hands all SCSI commands to the storage device; no enhanced features such as snapshots are available. This mode is also called physical compatibility or physical RDM; mapped LUNs act like they do on the physical servers.
- **Non-pass-through mode**: ESXi hands the LUN to the VM; however, it still offers some enhanced features such as snapshots. This mode is also called virtual compatibility or virtual RDM.

From a usability perspective, it is recommended to use pass-through mode in case RDM's needs to be utilized. For typical workloads, there is no need to utilize RDM's.
To sum it up, consider the following while maintaining storage LUNs:

- Always use the latest VMFS version, don't use extents
- Keep VMFS limits (for example, maximum files) in mind
- Use VMFS instead of RDM whenever possible

**I/O alignment**

A common reason for poor I/O performance of virtualized environments is the I/O alignment. Physical storage such as hard drives and filesystems, for example VMFS, have an allocation unit describing the smallest unit that is used to store data. These units can be called chunk size, block size, or cluster size. These are used for different types of storage, as follows:

- Storage systems have chunk sizes
- VMFS used by ESXi have a fixed block size
- Guest filesystems such as NTFS and ext4 use cluster or block sizes

Basically, all these words describe the smallest unit of storage allocation. It is important to bring the particular allocation units in a reasonable relation in order to ensure optimal I/O. All current hard drives use 4-kByte blocks to allocate storage. Operating systems use the logical block addressing (LBA) method to assign data to the unique disk sector numbers. Previously, operating systems assigned 512-byte blocks to store the data in a combination of a hard drive's cylinder, head, and sector number. For better compatibility, hard drives use internal 4-kByte blocks; however, they emulate 512-byte blocks for the connected bus (512e compatibility).

The goal of alignment is to set filesystem's block sizes in a way that they match the physical storage media boundaries. Therefore, in a perfect setup, a filesystem block matches exactly one physical block. If one filesystem block is read or written, only one physical block needs to be accessed by the hard drive, which speeds up performance. In this case, we're talking about aligned I/O. Let's take a look at the following image for Aligned IO:

<table>
<thead>
<tr>
<th>Aligned IO</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system (4K)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical blocks (512B)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Physical sectors (4K)</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Implementing High-Performance Storage

As you can see, the filesystem's block size matches the **Physical sectors** — independent of **Logical blocks**. Every File system block that is read/written is mapped to **Logical blocks**, matching exactly one **Physical sector**.

Now, let's focus on an example for unprofitable I/O, also called **Misaligned IO**:

<table>
<thead>
<tr>
<th>Misaligned IO</th>
<th>1</th>
<th>2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system (3K)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical blocks (512B)</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Physical sectors (4K)</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

The File system is using 3-kB blocks, which is unprofitable as the Physical sectors size is bigger. If the filesystem block 2 is read or written, two physical sectors need to be accessed, resulting in a doubled time effort and degraded I/O performance.

With virtualization coming into the picture, these block sizes also need to be considered for VMFS and used storage media such as shared or local storage systems.

Let's focus on the block sizes and file limitations VMFS offers, as follows:

<table>
<thead>
<tr>
<th></th>
<th>VMFS-3</th>
<th>VMFS-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block sizes</td>
<td>1, 2, 4, and 8</td>
<td>1 MB (fixed)</td>
</tr>
<tr>
<td>Maximum file size</td>
<td>1 MB block size: 256 GB</td>
<td>62 TB</td>
</tr>
<tr>
<td></td>
<td>2 MB: 512 GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 MB: 1 TB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 MB: 2 TB</td>
<td></td>
</tr>
<tr>
<td>Maximum files per volume</td>
<td>Approximately 30,720</td>
<td>Approximately 130,690</td>
</tr>
</tbody>
</table>

With vSphere 5, VMware decided to drop the possibility of selecting particular block sizes. When upgrading VMFS-3 volumes to VMFS-5, the former block size is inherited; therefore, this should be avoided. If a VMFS-5 volume is created using the vSphere Web Client or legacy client, the starting offset is automatically set to 2,056 to be aligned. When manually using `fdisk` or `vmkfstools`, the offset needs to be set manually. To be aligned, the offset needs to be dividable by eight.
Keeping VMFS block sizes in mind, let's take a look at the following block sizes of some popular guest file systems:

<table>
<thead>
<tr>
<th>Operating system</th>
<th>NTFS</th>
<th>Ext4</th>
<th>btrfs</th>
<th>ZFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Microsoft</td>
<td>Linux</td>
<td>Solaris,</td>
<td></td>
</tr>
<tr>
<td>Block sizes</td>
<td>Windows</td>
<td></td>
<td>BSD, and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>512, 1,024, 2,048, 4,096, or 8,192 bytes</td>
<td>1, 2, 4, or 64 kB</td>
<td>16, 32, or 64 kB</td>
<td>8 kB — 128 kB in 1 kB steps</td>
</tr>
<tr>
<td>Recommendation</td>
<td>Greater than 16 TB volume size: 4 kB</td>
<td>Greater than 4 TB volume size: 1 kB</td>
<td>Use automatic block-size selection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16–32 TB size: 8 kB</td>
<td>4–8 TB size: 2 kB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32–64 TB: 16 kB</td>
<td>8–16 TB: 4 kB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64–128 TB: 32 kB</td>
<td>16 TB–256 PB: 64 kB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>128–256 TB: 64 kB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that changing the block size after the filesystem is not possible. Consider the block size when creating the filesystem. Modern guest operating system versions automatically select the most reasonable block size when formatting volumes.

For optimal performance, filesystems larger than 4 GB should be aligned with 1 MB sizes. To accomplish this, we need to ensure that the starting offsets and block sizes are dividable by 2048 without any remainder (mapping between LBA and physical sectors), as follows:

\[
\text{LBA} = 8 \times \text{physical sector size}
\]
\[
\text{LBA} = 8 \times 512
\]
\[
\text{LBA} = 2048
\]

It's a common procedure to use a starting offset of 2,048 bytes. Bringing all these factors in one big picture, let's think about the following example:

- Local/shared storage with 4-kB block size
- VMFS-5 volume with 1-MB fixed block size
- Virtual disk with 4-kB sectors and 512-byte logical blocks
- Filesystem in guest operating system with 4-kB sectors
Refer to the following picture for an example of end-to-end aligned IO:

<table>
<thead>
<tr>
<th>Aligned full-stack I/O</th>
<th>VM File system (4K)</th>
<th>Host Filesystem, physical sectors and DAS/SAN storage: 4096/2048 = 2, aligned</th>
<th>VMFS block size: 1048576/2048 = 512, aligned</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Logical blocks (512B)</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
<td>255 2033 2034 2035 2036 2037 2038 ... 2047</td>
<td>256</td>
</tr>
<tr>
<td>Host DAS/SAN storage (4K)</td>
<td>1 2</td>
<td>1 (1 MB block)</td>
<td>255</td>
</tr>
</tbody>
</table>

Keeping the formula in mind, you can see that the storage is aligned, as follows:

- Filesystem, physical sectors and DAS/SAN storage: 4096/2048 = 2, aligned
- VMFS block size: 1048576/2048 = 512, aligned

To sum it up, consider the following:

- Don't upgrade VMFS-3 volumes to VMFS-5; reformat them
- Choose block sizes wisely when creating guest filesystems; most modern operating systems are virtualization-aware and automatically detect the best setting
- When using legacy guest operating systems, such as Microsoft Windows prior to 7, Server 2008 R2, or Red Hat Enterprise Linux 5, make sure to validate the block size

**Verify VM alignment**

In this section, you will see examples of how to verify the I/O alignment in guest operating systems. Keep in mind that if I/O is misaligned, then there is no supported non-disrupting way to change the filesystem design. To fix misalignment, it is a common procedure to create a new partition and copy the data.

**Microsoft Windows**

Starting with Windows 2008 R2 Server, filesystem alignment is provided automatically. To verify the I/O alignment under older Microsoft Windows releases, proceed with the following steps:

1. Click **Start** and **Run**, enter `msinfo32.exe`.
2. Select **Components** | **Storage** | **Disks**.
3. Check the **Partition Starting Offset** column information. The number should to be dividable by 4,096 without any remainder. Refer to the following screenshots for misaligned (32,256/4096 = 7,875) and aligned (1,048,576/4096 = 256) I/O:
For Linux-based systems, the `fdisk` utility is used to verify I/O alignment. Run the following command and check the start blocks—in order to be aligned, they need to be dividable by 2048 without any remainder (beginning LBA offset):

```
# fdisk -lu /dev/mmcblk0
```

Disk /dev/mmcblk0: 4025 MB, 4025483264 bytes
4 heads, 16 sectors/track, 122848 cylinders, total 7862272 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000bc88c

<table>
<thead>
<tr>
<th>Device</th>
<th>Boot</th>
<th>Start</th>
<th>End</th>
<th>Blocks</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/mmcblk0p1</td>
<td></td>
<td>2048</td>
<td>133119</td>
<td>65536</td>
<td>c</td>
<td>Linux</td>
</tr>
<tr>
<td>/dev/mmcblk0p2</td>
<td>133120</td>
<td>7337983</td>
<td>3602432</td>
<td>83</td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td>/dev/mmcblk0p3</td>
<td>7337984</td>
<td>7862271</td>
<td>262144</td>
<td>83</td>
<td>Linux</td>
<td></td>
</tr>
</tbody>
</table>

As you can see, the partition design is aligned. Refer to the following output for misalignment:

```
# fdisk -lu /dev/sda
```

```
... |
<table>
<thead>
<tr>
<th>Device</th>
<th>Boot</th>
<th>Start</th>
<th>End</th>
<th>Blocks</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sda1</td>
<td>*</td>
<td>63</td>
<td>208844</td>
<td>104391</td>
<td>83</td>
<td>Linux</td>
</tr>
<tr>
<td>/dev/sda2</td>
<td>208845</td>
<td>167766794</td>
<td>83778975</td>
<td>8e</td>
<td>Linux LVM</td>
<td></td>
</tr>
</tbody>
</table>
```

The starting blocks are not divisible by 8 and 2,048.
Implementing High-Performance Storage

Optimizing NFS resources

NFS is often chosen as the cost-effective alternative to SAN concepts. Due to the technologies such as affordable 10-GbE networks and Jumbo Frames, NFS can reach I/O throughputs that are comparable to SAN setups or they are even better at a lower price. To ensure high performance, we will focus on configuring Jumbo Frames along with NFS.

Jumbo frames

When accessing storage resources served by NFS resources, the typical Ethernet package limitations are given. By default, Ethernet uses a Maximum Transmission Unit (MTU) of 1,500 bytes. In other words, an Ethernet package can contain up to 1,500 bytes of payload. For modern high-speed networks, such as 10 GbE or higher, this is not suitable. Using the default MTU on an Ethernet network serving NFS resources results in approximately 3 to 5 Gbps throughput. The reason for this is that the hypervisor or network card (if using offloading functions) checks the integrity sums of network packages before handing them to the hypervisor's storage layer. The default MTU forces a higher load and overhead. Using a higher MTU in combination with NFS also means lower overhead as the packages are bigger and need fewer integrity checks. As a result, the storage throughput is higher.

When using an MTU higher than 1,500, the term Jumbo Frames is used. For iSCSI and NFS, ESXi supports MTU sizes of up to 9,000 bytes. When configuring higher MTU sizes, the switch configuration also needs to be evaluated as all the participating network components (such as switches and NICs) need to support it. Some switches also allow MTU sizes to be higher than 9,000 bytes; however, ESXi does not.

To configure a higher MTU size for NFS storage, proceed with the following steps:

1. Log in to vSphere Web Client.
2. Ensure that all VMs running on NFS storage are powered-off or moved to other datastores. During the following configuration, the connection will be reset. Also ensure that you have configured the switches and switch ports in order to enable higher MTU size.
3. Move to the networking pane and select the virtual switch or distributed virtual switch used for NFS communication.
4. Click Manage and Settings. In the Properties pane, click Edit.
5. Select Advanced and set the MTU size. Click OK to save the settings.
6. Wait for the NFS connection to be established or remount the share.
The following screenshots shows the configuration of custom MTU sizes for distributed virtual switches:

As MTU sizes are set in virtual switch level, it is recommended to have dedicated network cards and range for NFS or iSCSI communication. For conventional VMs, using larger MTU sizes is not recommended for all applications as most applications are not designed to utilize bigger MTU sizes.

Depending on your workloads, setting a higher MTU might have a smaller or bigger gain on performance. Make sure that you consider the following when configuring higher MTU sizes:

- Ensure that all the participating NICs and switches support higher MTU sizes; also, configure the switch ports
- Use dedicated virtual switches and NICs for storage backends

**Utilizing virtual flash resources**

In vSphere 5.5, a technology called vSphere Flash Read Cache (vFRC) was introduced; this makes it possible to utilize SSDs that are attached to the ESXi hosts as page target for the host itself and as read cache for VMs running on it. Locally attached SSD drives are grouped in a vFRC pool where various virtual flash resources can be located. This technique can significantly improve the performance of read-intensive workloads.
Implementing High-Performance Storage

The vFRC feature requires a vSphere Enterprise Edition license.

The vFRC uses a special filesystem to store data, it is not replicated across the ESXi hosts. When using for an ESXi host, the flash resources are used in case the host is overloaded and needs to page out the memory resources. An ESXi host should be designed in a way that it never needs to page out; if this scenario arises, Virtual Flash Host Swap Cache can ensure better performance than with conventional hard disk storage.

On a per virtual disk basis, a fixed cache size is defined. This cache offloads blocks requested very often into flash storage to enable faster access. As a result, the storage that the VM resides on will have more resources for other I/O operations such as write requests. The vFRC is vMotion-aware ([Enhanced vMotion using vSphere Web Client](#)); when moving the VM, it is also possible to migrate the read cache. Using this flavor of cache does not remove data on the conventional storage—if the accelerated cache is lost or not migrated, it will be rewarmed. DRS is also able to leverage vFRC technology during evaluating the placement designation. As vFRC is implemented at hypervisor level, it is also completely transparent to the VM. When a vFRC-enabled VM fails over to another host, it will only power on if the sufficient flash resources are present. If a vFRC pool on an ESXi host reaches 90 percent usage, it will start to evict pages based on the usage statistics. Refer to the following table for limitations:

<table>
<thead>
<tr>
<th>Type</th>
<th>Configuration</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system</td>
<td>Virtual Flash File System (VFFS), a SSD-optimized VMFS derivate</td>
<td></td>
</tr>
<tr>
<td>Use cases</td>
<td>Host memory swap and VM Flash Read Cache</td>
<td></td>
</tr>
<tr>
<td>Supported interfaces</td>
<td>PCI Express, SAS, and SATA (no remote devices)</td>
<td></td>
</tr>
<tr>
<td>Number of devices per host</td>
<td>1–8 SSDs</td>
<td></td>
</tr>
<tr>
<td>Maximum SSD size</td>
<td>4 TB</td>
<td></td>
</tr>
<tr>
<td>Maximum vFRC pool size per host</td>
<td>32 TB</td>
<td></td>
</tr>
<tr>
<td>Maximum host swap cache size per host</td>
<td>4 TB</td>
<td></td>
</tr>
<tr>
<td>Maximum cache size per virtual disk</td>
<td>200 GB (using Web Client)</td>
<td>400 GB (using advanced host settings)</td>
</tr>
<tr>
<td>Supported block sizes per VM cache</td>
<td>4, 8, 16, 32, 64, 128, 256, 512, or 1,024 kB</td>
<td></td>
</tr>
</tbody>
</table>
Also, keep the following details in mind when planning to use vFRC:

- The vHW 10 or higher is required for virtual flash resources in VMs. Currently, Fault Tolerance is not supported along with vFRC.
- Be careful when sizing virtual flash caches; only assign it to VMs that require it. If a VM fails over to another host, it won't power on if the virtual flash resources are exhausted.
- Make sure to check the VMware Hardware Compatibility List (HCL) when selecting SSDs for your servers: http://www.vmware.com/resources/compatibility/search.php?deviceCategory=vfrc.

Preparing virtual flash resources
To prepare using virtual flash resources, proceed with the following steps:

1. Log in to vSphere Web Client.
2. Click Hosts and Clusters and select a particular ESXi host.
4. Click Add Capacity. Select the SSDs and click OK.
5. Repeat the preceding steps for all the remaining hosts in your cluster. The following screenshot demonstrates adding a SSD as Virtual Flash resource:
Configuring Virtual Flash Read Cache

To configure Virtual Flash Read Cache for a particular VM, proceed with the following steps:

1. In the navigation, click VMs and Templates and select a particular VM.
2. Click Manage and Settings. Next to VM Hardware, click Edit.
3. Expand the hard disk column and enter a cache size. When clicking Advanced, it is also possible to set a particular block size. The higher the block size, the latency also increases. The default of 4 kB is usually the best setting as it offers the best granularity. Depending on the VM and guest filesystem and application design (for example, database server), it might also be more efficient to set higher cache sizes. Make sure to run performance tests when experimenting with different sizes.
4. Click OK to save the changes.

The following screenshot demonstrates the configuration of a particular virtual disk for using vFRC resources:

![Virtual Flash Read Cache Configuration](image)

Configuring Virtual Flash Host Swap Cache

To configure Virtual Flash Host Swap Cache, proceed with the following steps:

1. In vSphere Web Client, click Hosts and Clusters and select a particular ESXi host.
2. Select Manage and Settings. Under Virtual Flash, click Virtual Flash Host Cache Configuration.
3. Click Edit and select Enable virtual flash host swap cache.
4. Specify the cache size in GB and click **OK**.
5. Repeat the preceding steps for all the remaining hosts in your cluster.

Refer to the following screenshot for an example configuring **Virtual Flash Host Swap Cache**:

![Virtual Flash Host Swap Cache](image)

---

**Storage I/O Control**

**Storage I/O Control (SIOC)** is a technology that ensures that all the running workloads on multiple ESXi hosts, using shared storage, achieve the same fair amount of resources. In other words, it makes sure that a VM doesn't affect the performance of other VMs (noisy neighbor scenario), independent of the ESXi host they are running on. To do this, SIOC monitors the average latency and adjusts—if necessary, it also monitors the virtual disk device queues of particular VMs in order to ensure fair resource usage. In combination with disk shares and limitations, priorities can be implemented. SIOC is supported with block-level storage and NFS.

SIOC requires a vSphere Enterprise Edition license.

It is possible to configure SIOC in two ways, as shown in the following:

- Manual threshold
- Percentage of peak throughput

Before vSphere 6, a manual threshold of 30 ms was the default. Consider the following about deviant values:

- A **lower value** ensures higher throughput and, possibly, higher latency; however, it also leads to less isolation. Isolation describes how often share controls are enforced.
Implementing High-Performance Storage

• Using **higher values** results in lower latency and stronger isolation. Keep in mind that extremely high values might reduce throughput.

When latency is very crucial for your environment, you should set thresholds of 20 ms or less. Refer to the following table for frequently used latency thresholds:

<table>
<thead>
<tr>
<th>Media</th>
<th>Threshold recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td>10-15 ms</td>
</tr>
<tr>
<td>FC/SAS</td>
<td>20-30 ms</td>
</tr>
<tr>
<td>SATA/Nearline-SAS</td>
<td>30-50 ms</td>
</tr>
</tbody>
</table>

However, setting this value is not an easy task; it is possible to easily configure SIOC inadequately. Setting values need to be chosen very carefully, depending on the storage type setting inaccurate values will result in suboptimal behavior. Beginning with vSphere 6, VMware decided to make a percentage setting of peak throughput the default behavior. Once enabled, ESXi will automatically find a threshold based on the throughput percentage. The higher the value, the higher the threshold will be. You can compare results with higher and lower values when using manual thresholds (as seen previously).

SIOC is not supported with **Raw Device Mappings (RDMs)**, VSAN, or old VMFS volumes with multiple extents. Therefore, again: make sure not to use VMFS-3 – use VMFS-5 instead.

SIOC is enabled per datastore basis and executed every four seconds. Basically, it is recommended to use SIOC as it is one of the most advanced vSphere features. However, for some high-performance setups, disabling this feature can also be a way to optimize I/O performance. Think about very latency-sensitive workloads running on dedicated resources that are not shared (for example, standalone storage per VM in isolation setups). In this case, using SIOC will not improve the performance as a single workload is flooding the storage.

To sum it up, keep the following in mind:

• SIOC is enabled on a per datastore basis. Make sure that you run performance benchmarks to check the difference. Enable SIOC in consolidation setup approaches.

• Start with default settings and make changes if required. Be careful when setting manual thresholds; make changes stepwise.

• Utilize SIOC to prioritize your important VM disks.
Configuring SIOC

To configure SIOC, proceed with the following steps:

1. Log in to vSphere Web Client and select the Storage pane on the left.
2. Select a datastore from the list and click Manage, Settings, and Edit next to Datastore Capabilities.
3. Click Enable Storage I/O Control. If necessary, also select a custom threshold.
4. Click OK to save the changes. Repeat these steps for other shared storage units. Refer to the following screenshot for an example of configuration:

![Configure Storage I/O Control](image)

Storage DRS

In addition to conventional DRS, **Storage DRS (SDRS)** can be used to balance storage usage and latency. It can also be used to simplify storage management. Without Storage DRS, the administrator manually needs to ensure that all the virtual disks are placed in optimal storage. Storage DRS can assist by offering the following features:

- **Logical grouping**: Coherent datastores are grouped as datastore cluster (for example, test LUNs as test datastore cluster). However, a datastore cluster is also a new storage object that is used by many tasks and tools in vSphere. Using datastore clusters heavily simplifies the storage management. Instead of specifying a particular datastore, SDRS can select the most appropriate data storage when deploying a VM.
Implementing High-Performance Storage

- **Load balancing**: SDRS is able to balance the storage consumption and latency between datastores. It offers two independent automation levels: **Manual** and **Fully Automated**. While manual mode only makes recommendations, fully-automated mode will also migrate virtual disks. SDRS monitors datastores in order to create metrics that are used for evaluating migrations. In fully-automated mode, SDRS will check imbalances every eight hours. By default, datastores utilizing more than 80% capacity or having latency higher than 15 ms are considered imbalanced. For migrating virtual disks, Storage vMotion is used. Only datastores having at least 5% more free capacity are evaluated as target. SDRS is also capable of reading the I/O statistics by SIOC.

- **Initial placement**: When deploying new workloads, SDRS considers virtual disk parameters and finds an appropriate datastore for it. Besides storage utilization, latency values are also evaluated.

- **Affinity**: SDRS is able to consider the affinity rules during migration evaluations. In other words, it is possible to keep all the virtual disks of a VM on the same datastore, which is the default.

- **Easy maintenance**: Before storage backend maintenance, SDRS can automatically migrate virtual disks online in order to avoid downtime. It is comparable to an ESXi host's maintenance mode.

---

Storage DRS requires a vSphere **Enterprise Plus Edition** license.

Using Storage DRS is highly recommended if your vSphere license allows it; it is one of the most advanced vSphere features. When planning to use Storage DRS, consider the following:

- When using Storage DRS in fully-automatic mode along with SAN resources, make sure that your storage supports VAAI full copy (XCOPY primitive).

- Is keeping all VMDKs of a particular VM together suitable for your environment? Disabling it can make SDRS even more granular and efficient as virtual disks are spread across the cluster. Especially, for big VMs, this can be a huge benefit.

- Choose custom intervals to check the imbalance wisely. Setting an extremely low value might result in too many migrations. Keep in mind that every Storage vMotion process temporarily affects the performance slightly for the time of the migration. The default setting of eight hours is suitable for most environments. Extremely tampering landscapes might require setting a lower value.
Configuring Storage DRS
Before configuring Storage DRS, ensure that you have at least two datastores to configure the datastore cluster. Proceed with the following steps:

1. Log in to vSphere Web Client and move to the Storage pane.
2. Select the data center where the datastore cluster should be created. Click Actions, Storage, and New Datastore Cluster.
3. Enter a name for the datastore cluster. Also, make sure that Turn ON Storage DRS is enabled.
4. Select No Automation (Manual Mode) or Fully Automated.
5. In the next screen, set Storage DRS thresholds. Under the Advanced options, there are additional options such as Default VM affinity and IO imbalance threshold, which control the sensitivity of automatic migrations. It is also possible to control how often SDRS will check for imbalance (1-720 hours). Click Next.
6. Select clusters or standalone ESXi hosts participating in the datastore cluster.
7. Click all the datastores that should join the datastore cluster. Already existing datastores with VMs running on it are also supported for this. Click Finish.

Summary
In this chapter, you learned how to design and optimize block-based and file-based shared storage. We now know how advanced vSphere features, such as SIOC, vFRC, and Storage DRS, can assist us in achieving greater I/O performance.

In the next chapter, we will focus on high-performance networking. You will learn how to optimize our networking design for more throughput. After this chapter, you will know how technologies such as Network I/O Control (NIOC) and TCP Segmentation Offload (TSO) can assist us in achieving this goal.
Where to buy this book

You can buy vSphere High Performance Essentials from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers. Click here for ordering and shipping details.