Performance Study of VMware vStorage Thin Provisioning

VMware vSphere 4.0

Thin provisioning allows virtual disks to allocate and commit storage space on demand. This paper presents a performance study of thin-provisioned disks in a VMware vSphere™ test environment. The test results show that thin disks perform as well as thick disks, even under I/O-intensive workloads.

Introduction

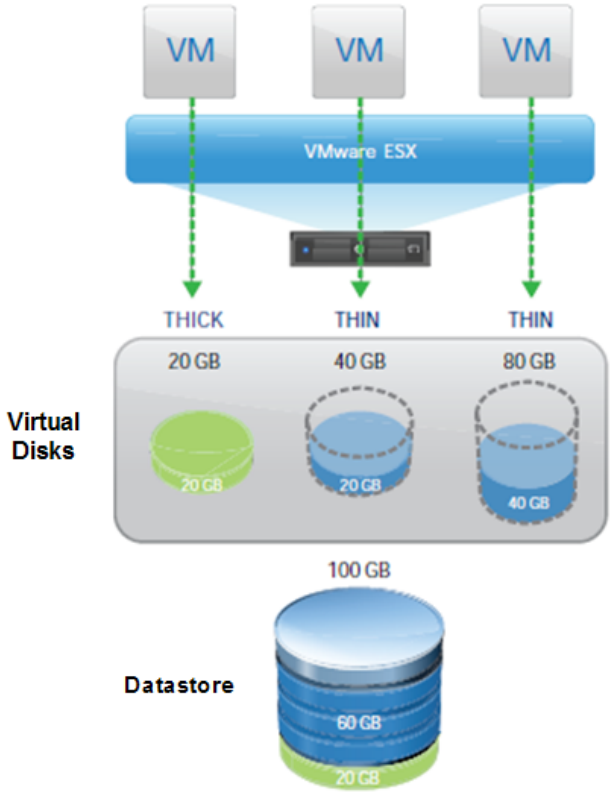
Thin provisioning for VMware vSphere is a technology that redefines how storage space is allocated to virtual disks. Previously, administrators needed to estimate how much storage space their virtual disks would take up to support current usage and future growth, and pre-allocate that entire storage space for each disk.

Thin provisioning, in contrast, allows virtual disks to use only the amount of storage space they currently need. On a vSphere system, a collection of thin-provisioned virtual disks can be created, each of whose upper limit of disk space specified, when combined, is greater than the total amount of storage space currently available. This is known as over-commitment of storage capacity. When administrators employ the technique of over-commitment paired with careful monitoring of actual disk usage, an organization can save a significant amount of money by requiring less hardware to support their vSphere infrastructure.

Figure 1 illustrates the concept of over-commitment. The three virtual machines (one thick and two thin) have disk sizes that together total 140GB. The actual storage space available, however, is 100GB. This size mismatch is possible because the thin disks take only 60GB of actual space, even though they can grow larger.

The 20GB disk, because it is thick, must have the entire 20GB pre-allocated to it. This 20GB of space appears on the storage as a full 20GB, whether or not the data on the virtual disk is actually all utilized. The thin disks, in contrast, only use the actual 20GB and 40GB of disk space on the datastore. Figure 1 shows both thin disks' sizes as a combined 60GB on the datastore.

Figure 1. Thin Provisioning allows storage space to be overcommitted



Allocation Type of Disks

VMware vStorage thin provisioning operates at the virtual machine disk (VMDK) level. When a VMDK file is allocated, it can be allocated as either thick or thin. The performance of writing data to these disks can be affected by the time at which the data is cleared, or zeroed.

Table 1. A virtual machine's disk can be allocated as one of three types

Allocation Type	Pre-allocated	Zeroing
Zeroed thick (default)	Yes	Run-time
Eager zeroed thick	Yes	Create-time
Thin	No	Run-time

Zeroing

Zeroing is the process whereby disk blocks are overwritten with zeroes to ensure that no prior data is leaked into the new VMDK that is allocated with these blocks. Zeroing in the ESX file system (VMFS) can happen at the time a virtual disk is created (create-time), or on the first write to a VMFS block (run-time).

Thin Disks

Thin provisioning involves the creation of thin virtual disks, which are VMDK files. Thin virtual disks are not any larger than they need to be (that is, they are not pre-allocated), and they are not zeroed out until run-time.

Blocks in a thin VMDK file are not written during non-write operations like read and backup. For example, a read to an unallocated block returns zeroes. The block is not backed with physical storage until a write occurs. In this way, thin disks optimize disk space.

Because zeroing takes place at run-time for a thin disk, there can be some performance impact for write-intensive applications while writing data for the first time. After all of a thin disk's blocks are allocated and zeroed out, however, the thin disk is no different from a thick disk in terms of performance.

Thick Disks

There are two types of thick disk allocation types: zeroed thick and eager zeroed thick.

- Zeroed thick is the default allocation type for virtual disks on ESX hosts. This type of disk pre-allocates and dedicates a user-defined amount of space for a virtual machine's disk operations, but it does not write zeroes to a VMFS block until the first write within that region at run-time.
- Eager zeroed thick pre-allocates and dedicates a user-defined amount of space for a virtual machine's disk operations. All the blocks on this disk are pre-zeroed. Pre-allocation plus zeroing makes creation time slower for this disk when compared with a thin or zeroed thick disk, but because the disk is zeroed at create-time, there is no performance impact due to zeroing block segments upon first write. This disk type is typically used with fault tolerance and is created through the command-line using vmkfstools.

Our tests focus on the default type of thick virtual disk, zeroed thick.

Test 1: I/O-Intensive Workload

For this test, we simulated an environment where an I/O-intensive application (one with a large and sustained workload) runs on a virtual machine that is provisioned with a thin disk, and on a virtual machine that is provisioned with a thick disk. The storage for both disks exists on an array of Fibre Channel disks.

Test 1 Setup

The following sections describe the workload and hardware configuration for this performance study.

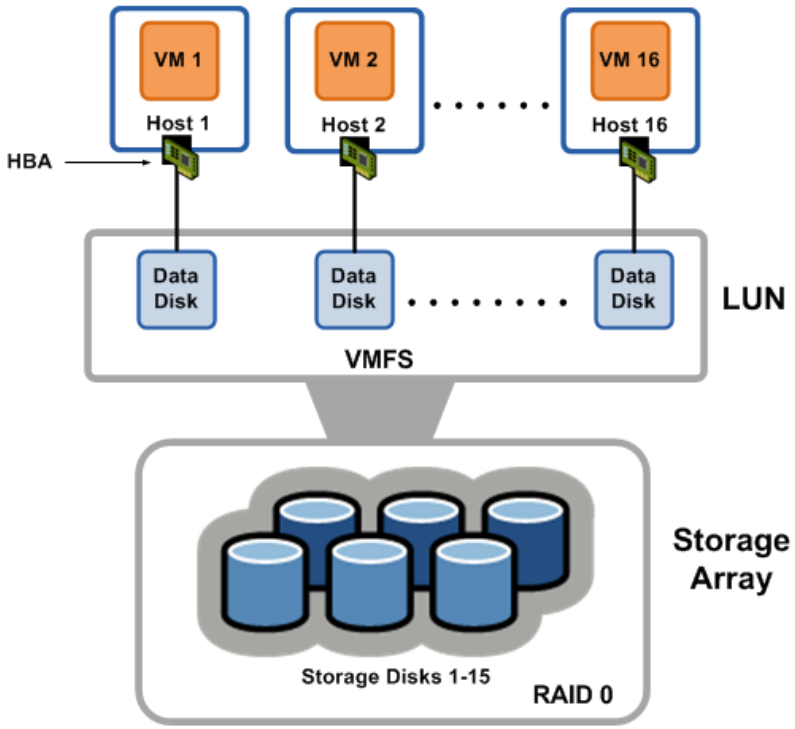
Workload

Iometer (www.iometer.org) is a tool for measuring and characterizing I/O subsystem performance and has a variety of options to generate the desired I/O access patterns. It reports metrics such as I/O per second, throughput, latency, CPU utilization, and so on.

Hardware Configuration

Figure 3 represents the hardware setup used for the performance study of VMware vStorage thin provisioning.

Figure 2. Abstraction of hardware test configuration



The test environment was made up of 16 hosts running as ESX servers, each with a single virtual machine running on it. The backend storage was an array of 15 disks configured as RAID 0.

The 16-host configuration was used to generate the data for the experiment, "How does thin provisioning scale?" The other experiments in Test 1 required only two hosts for comparison.

The following sections provide the relevant hardware and software configuration details.

Servers (16)

Model: HP ProLiant DL380
Processors: Two Intel Xeon processors @ 2.8GHz with hyperthreading
Memory: 4GB RAM
FC HBA: QLogic ISP-2432 based 4Gb adapter

Storage Array

Model: EMC Clariion CX700
Cache: 1GB exclusive read cache per storage processor (SP), 2GB shared write cache
Disks: 15 * ST371460 Seagate 10,000 RPM FC disks
Frontend connections: One 2Gbps FC connection to each storage processor in Active/Passive mode
RAID level: RAID 0

Software Configuration

VMware vSphere : 4.0 build # 164009
VMFS : version 3.33, 1.6 TB size with RAID 0 configuration

Test 1 Methodology

Each vSphere ESX server ran one virtual machine that produced I/O load using Iometer. The data disks for each virtual machine were stored on a shared LUN and each was configured as a thick- or thin-provisioned disk based on the experiments chosen.

We ran the individual Iometer tests for 2 minutes each and then power cycled the guest. The complete iteration of the Iometer run included going through all the blocks sizes from 1K-512K and using both sequential and random read/write workloads wherever appropriate. We used 2 minutes for each test in order to clearly identify two distinct phases that both the thick and thin disks undergo:

- **Zeroing,** when the VMFS blocks need to be zeroed before they are written
- **Post-zeroing,** when all the VMFS blocks are zeroed, so no zeroing takes place before writing

The following metrics were collected during the tests:

- Guest I/O throughput and latency measured by Iometer
- Host CPU consumption
- Array side throughput and latency numbers using EMC Navisphere analyzer
- SCSI reservation conflicts[1]
- VMDK information such as to-be-zeroed blocks and number of blocks from the VMFS layer before and after the test[2]

Test 1 Results

The results of this test answer the following questions:

1. How does thin provisioning perform relative to thick?
2. How does thin provisioning scale?
3. What is the performance impact of external fragmentation?
4. What is the performance of a thick disk when a thin disk shares the same volume?

How does thin provisioning perform relative to thick?

Our tests showed that both thin and thick disks perform with similar throughput when compared. Figure 4 shows the results after running sequential writes.[3]

The figure shows that the aggregate throughput of the workload is around 180 MBps in the post-zeroing phase of both thin and thick disks, and around 60 MBps when the disks are in the zeroing phase. Due to empty blocks needing to be zeroed on first write, the zeroing phase number is much lower, as expected, for both disks.

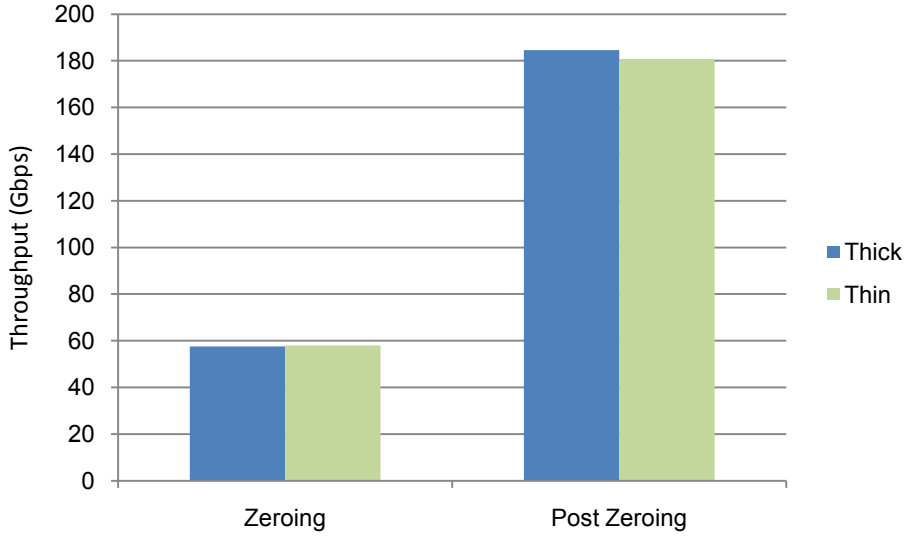
---

[1] When the guest issues its first write to an unallocated file block, VMFS needs to allocate the file block. This operation requires VMFS to briefly reserve the LUN and can lead to other initiators experiencing some SCSI reservation conflicts.

[2] For more information, see "To-Be-Zeroed Blocks and Number of Blocks" in the Appendix.

[3] We used 64K sequential writes to generate the charts for these tests. Performance trending was the same for sequential writes of various sizes, from 1K through 512K.

Figure 3. Aggregate throughput of thin disks is similar to that of thick disks

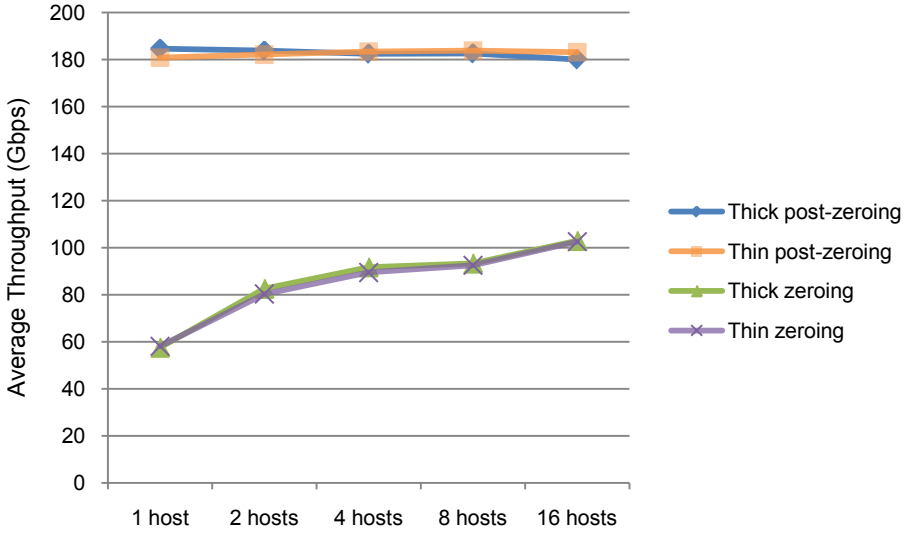


For more common workloads (that is, those that are not I/O-intensive), see "Test 2: File Copy Workload." Note that the aggregate throughput of disks in the zeroing and post-zeroing phases does not show such a wide disparity as in this test.

How does thin provisioning scale compared to thick?

To answer this question, we collected data from sequential writes on 1 host and scaled up through 16 hosts.

Figure 4. Throughput of thin and thick disks compared in the zeroing and post-zeroing phases, and scale up data from 1 host to 16 hosts



The resulting data displays similar throughput for both thin and thick virtual disks, in both the zeroing and post-zeroing phases.

It is also worthwhile to note that we did not see any effect of SCSI reservation conflicts on the performance as we scaled from a lower to a higher number of hosts.

What is the performance impact of fragmentation on thin disks?

To answer this question, let's first take a look at two types of fragmentation that can affect the VMFS-3 file system: internal and external.

Internal Fragmentation

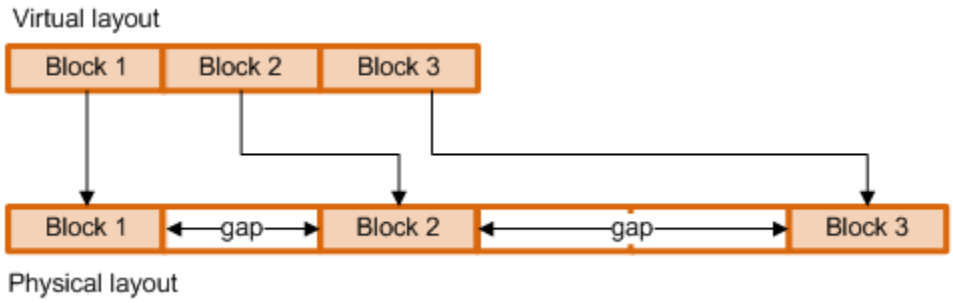
Internal fragmentation occurs when the file system allocates a block for a file, but the file doesn't use the entire block. VMFS-3 addresses this issue by using a sub-block allocator. Small files use sub-blocks instead of file blocks. A sub-block is 1/16th the size of a file block on 1MB file block-size volumes, and 1/128th the size of a file block on 8MB volumes.

External Fragmentation

External fragmentation occurs when blocks belonging to the same file are scattered across different locations on the volume. Such distributed data can affect performance by increasing seek time and rotational latency, which is the time it takes to physically move the disk head to the right track.

For example, a file from a single virtual machine might write to physical disk storage on three, non-consecutive block locations, as shown in the following figure. Fragmentation is measured as the distance between two blocks, labeled as each gap in the figure.

Figure 5. Virtual machine data becomes fragmented when it is written to discontiguous regions on VMFS



External fragmentation is unlikely to have much impact on vStorage thin-provisioned disks for the following reasons:

- The default block size in VMFS is 1MB. Because most of the I/O requests are of 64KB in size, the 1MB VMFS block size is sufficiently large enough that most of the I/O requests do not straddle block boundaries. So even if blocks are discontiguous (not next to each other), I/O requests execute to locally contiguous (nearby) regions.
- When the thin-provisioned disk is fully grown, it behaves the same as a thick disk that is pre-allocated and already zeroed (eager zeroed thick). On a fully-zeroed and grown disk, the gaps between two fragments of data are at least as large as the size of the block.
- Disk arrays have huge caches, and most of the disk writes are absorbed there. This aspect of SAN devices makes it difficult for fragmentation to have a noticeable effect on disk write performance.

How do the fragments grow for a particular thin disk?

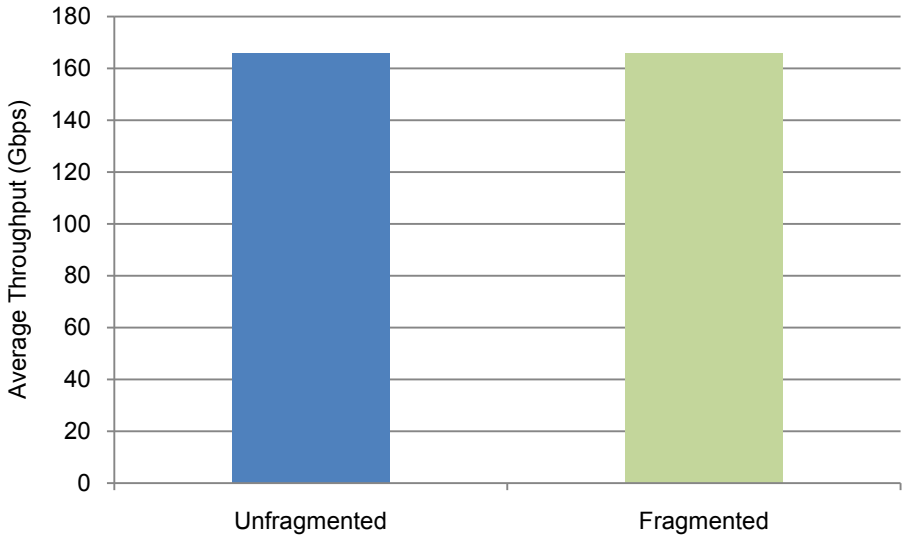
Typically, virtual machines are registered to a given host and blocks from a given host are allocated closer together so that there is a spatial locality for a given virtual disk's file blocks on disk. This allocation process also depends on the workload that is running on a given ESX host.

Thin-provisioned disks do not have their storage pre-allocated. Because of this, if there are several thin-provisioned virtual disks that are being used, then even though as a whole the allocated blocks from that

ESX host would be close together, there is no guarantee that an allocation for a single thin-provisioned virtual disk would be close together.

The following figure shows that the number of fragments has very little effect on the performance of the VMware vStorage thin-provisioned disks.

Figure 6. Fragmentation does not noticeably affect throughput performance for thin disks

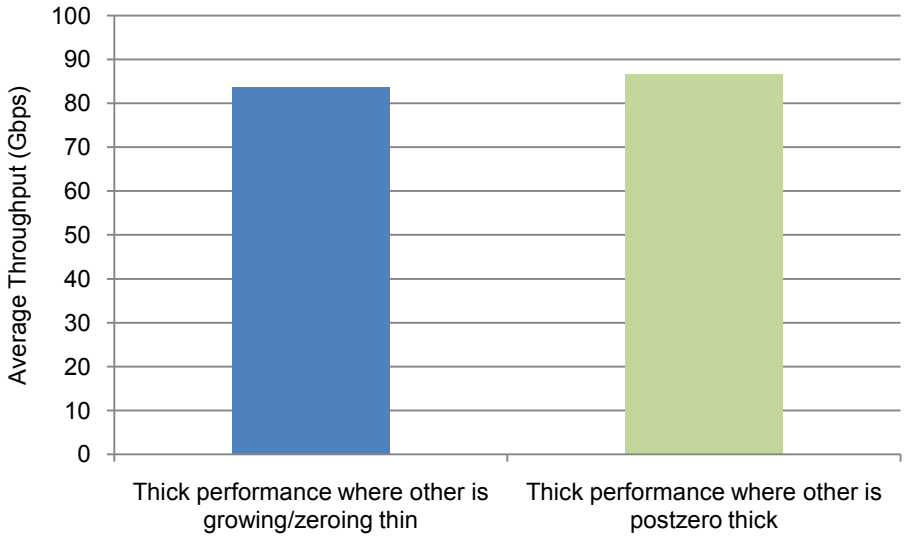


The data shown in the previous figure is from a 64K random write. The fragments were created by picking an appropriate workload while growing the thin disk to 12GB. Growing an empty 12GB thin disk to capacity using a 1KB random write workload resulted in a maximum of 12,288 fragments on 1MB block size VMFS volumes. A 1KB sequential write created the least number of fragments, so that data is not included.

What is the performance of a thick disk when a thin disk is on the volume?

The following figure shows that thick disk throughput performance is only slightly impacted while the thin disk is in the zeroing phase.

Figure 7. Effect of thick performance when a thin disk co-exists on same host



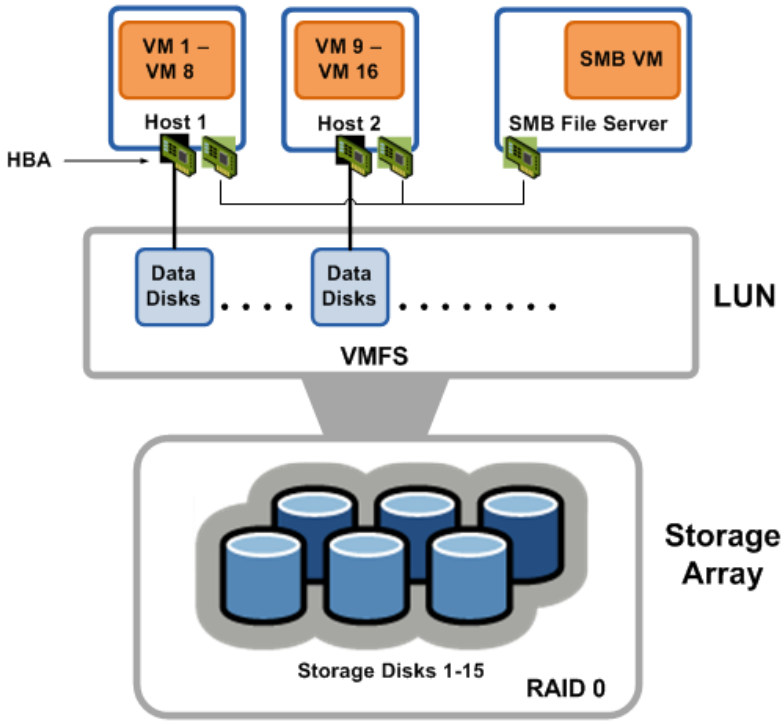
Test 2: File Copy Workload

Test 2 measures the performance of thin-provisioned disks with a relatively light-weight file copy workload.

Test 2 Setup

The following figure depicts the hardware configuration designed for this test.

Figure 8. Hardware configuration for Test 2



Servers (3)

Model: HP ProLiant DL380
Processors: Two Intel Xeon processors @ 2.8GHz with hyperthreading
Memory: 4GB RAM
FC HBA: QLogic ISP-2432 based 4Gb adapter

Storage Array

Model: EMC Clariion CX700
Cache: 1GB exclusive read cache per SP, 2GB shared write cache
Disks: 15 * ST371460 Seagate 10,000 RPM FC disks
Frontend connections: One 2Gbps FC connection to each storage processor in Active/Passive mode

VMs

Clients: Windows Server 2003 Enterprise Edition, 32-bit, 1 vCPU 512MB RAM (16 client VMs, 8 VMs per host)
Server: Windows Server 2003 Enterprise Edition, 32-bit, 2 vCPU 1GB RAM

Test 2 Methodology

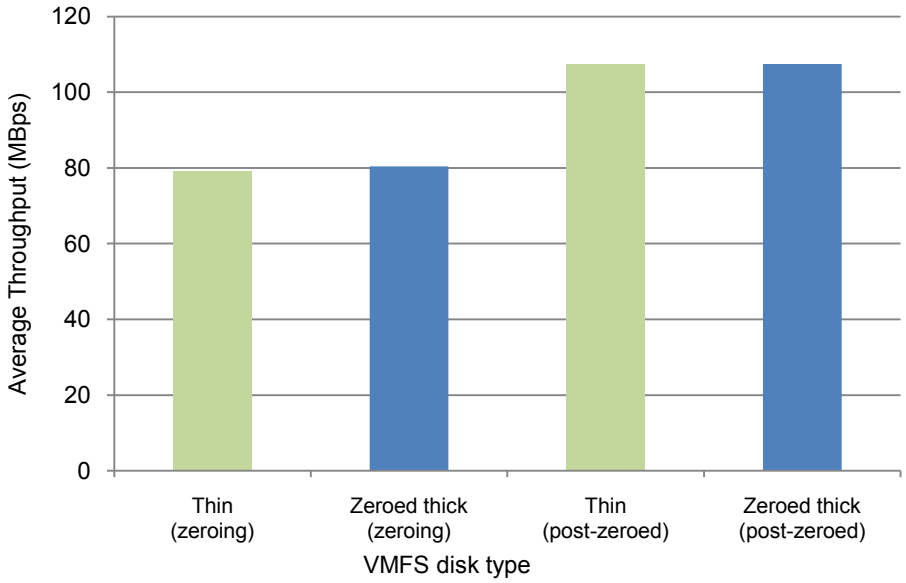
Test 2 measures the performance of a file copy operation that happens simultaneously on multiple client virtual machines that run on two ESX hosts, while the files are served by an SMB file server VM that runs on a separate ESX host. The network communication occurs across a dedicated 1Gbit network. Each client VM copies files from the network iteratively to its own virtual disk until it fills up. About 290MB worth of files is copied during each iteration. The virtual disks of all client VMs reside on a single VMFS volume.

With such a setup, we measured the average file copy time for each iteration across the client VMs while varying the backing disk type of the virtual disks to thin (zeroing), thin (post-zeroing), zeroed thick, and eager zeroed thick.

Test 2 Results

The following figure shows the results from test 2.

Figure 9. Test results of a file copy workload from multiple virtual machines



The results reinforce what we have shown earlier:

- Thin-provisioned disks have virtually no performance overhead compared to the default disk format (zeroed thick) during the block allocation and zeroing phase. In the graph, compare **Thin (zeroing)** and **Zeroed thick**.
- Thin-provisioned disks have no performance overhead due to fragmentation and perform similarly compared to a fully-allocated and zeroed thick disk (eager zeroed thick). In the graph, compare **Thin (post-zeroed)** and **Eager zeroed thick**.

Performance Recommendations

This section provides recommendations for array-side thin provisioning, benchmarking, and monitoring.

Array-Side Thin Provisioning and VMware vStorage Thin Provisioning

Some storage array manufacturers implement thin provisioning behind the LUN. This type of thin provisioning can include on-demand provisioning and initialization of the requested block on its first access. Arrays can also encompass other space-saving measures such as de-duplication of zeroed blocks.

VMware vStorage thin provisioning optimizes its space based on the workload, allowing more virtual machines to fit on the datastore. Array-side thin provisioning can optimize physical space further. You can use VMware vStorage thin provisioning and array-side thin provisioning together in your vSphere environment. However, we recommend careful monitoring and management of thin-on-thin systems.

Benchmarking Recommendation

If you are benchmarking using thin-provisioned disks, make sure you have given sufficient time to warm up the thin disk so that it is fully grown, or make sure you track the values of **nb** and **tbz** for your benchmarks so that you are aware of which phase the thin disk is going through (see "Acquiring Actual Values" under "To-Be-Zeroed Blocks and Number of Blocks" on page 13).

Other Recommendations

We recommend the use of alerts to help monitor thin disk oversubscription. In particular, the following alarm triggers will be helpful: **Datastore Disk Usage %** and **Datastore Disk Overallocation %.** Using VMware vCenter Server, you can provide reports and set thresholds to proactively manage growth and capacity. For more information about alarm triggers, see the vSphere 4.0 *Basic System Administration* guide.[4]

Thin provisioning can lead to the oversubscription of physical disk space. You can manage oversubscription with Storage VMotion, which enables the dynamic migration of VMDKs, or VMFS volume grow, which provides the ability to dynamically increase the size of your datastore.

Note: Swap files, redo logs, and linked clones for snapshots do not make use of thin provisioning.

Conclusion

In this paper, we determined that a workload run on a thin-provisioned disk performs very closely to that of a thick-provisioned disk, in both the zeroing and post-zeroing phases of disk growth. Additionally, scaling up the servers from 1 host to 16 hosts showed improved throughput as more hosts were added. After the disks were all allocated and zeroed, the throughput remained stable, as expected. Next, we showed that the performance impact of external fragmentation is negligible. Finally, we saw that the performance of a thick virtual disk is barely affected when a thin disk co-exists on same host. An additional test for a file copy workload showed thin provisioning to be comparable with thick provisioning. Workloads with low I/O place a negligible delay in the zeroing and post-zeroing phases of disk growth.

The data we have collected reveals that VMware vStorage thin provisioning (the usage of thin virtual disks) is at par in performance with the default method of provisioning in VMware vStorage (the usage of zeroed thick virtual disks). Moreover, VMware vStorage thin provisioning offers increased storage utilization, which results in savings on physical storage hardware.

Appendix

This section contains information relevant to re-creating our tests, including:

- How to employ thin-provisioned disks.
- A description of to-be-zeroed blocks (**tbz**) and number of blocks (**nb**), and how to find this information (see "Acquiring Actual Values" on page 13).

Employing Thin-Provisioned Disks

Creating virtual machines as thin disks allows these disks access to a larger amount of storage than their actual footprint. The storage space is allocated and expanded on-demand by the VMFS-3 driver when the guest operating system requests more space.

Virtual disks are easily specified as thin during the disk creation process. On the Create a Disk page, select the checkbox next to **Allocate and commit space on demand (Thin Provisioning)**.

Existing thick disks can be transformed to thin during a migration using Storage VMotion. On the Disk Format Page, select **Thin Provisioned Format.**

Existing thick disks with underutilized storage capacity can be changed to thin from the command-line using vmkfstools with the following options:

```
vmkfstools –i <thick_disk>.vmdk <thin_disk>.vmdk –d thin
```

To-Be-Zeroed Blocks and Number of Blocks

Zeroed-thick disks and thin disks both go through a zeroing phase and a post-zeroing phase. For the purposes of data collection and interpretation in our experiment, we defined variables for **to-be-zeroed blocks (tb)** and **number of blocks (nb)** for zeroed thick disks and for thin disks. Because zeroed thick is the default allocation type for thick virtual disks, we do not include statistics for eager zeroed thick in this study.

For both thin and zeroed-thick disk types, we defined the following:

- **tbz** is blocks to be zeroed
- **nb** is the number of blocks in the current disk and **nb = disk size (in MB) / VMFS block size (in MB)** defines the pre-allocated state of a disk

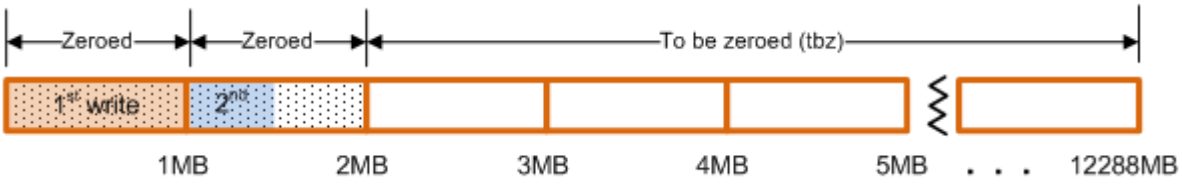
Thick Virtual Disk

Before entering the zeroing phase, the following is true for a thick disk:

- **nb = disk size / VMFS block size** because its blocks are pre-allocated
- **tbz = nb** to start

In the zeroing phase of a thick disk, **tbz** decreases as all of the blocks are zeroed out, eventually reaching 0 (no blocks left to be zeroed).

Figure 10. Thick disk pre-allocated to 12GB, in the process of zeroing and writing; nb = 12288 (each VMFS block is 1MB, with 12288MB in the 12GB disk)



Thin Virtual Disk

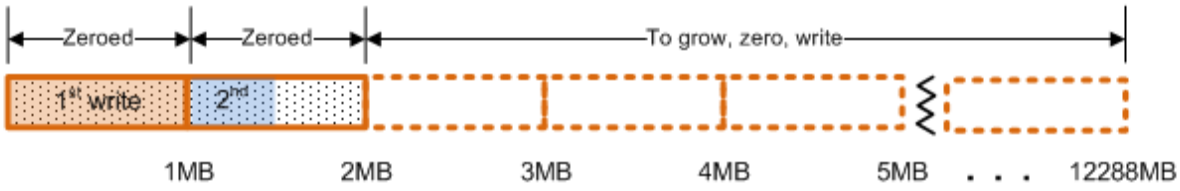
Before entering the zeroing phase, the following is true for a thin disk:

- **nb = 0** to start, because the disk is not pre-allocated
- **tbz = nb** to start

In the zeroing phase of a thin disk:

- **nb** increases from 0 to **disk size / VMFS block size**
- **tbz** is 0 as blocks are added and zeroed as needed

Figure 11. Thin disk (1MB VMFS block size) in the process of growing, zeroing, and writing until it reaches 12GB; nb = 2 at this point



After the zeroing phase is completed for a thin disk, it enters the post-zeroing phase where

- **nb = disk size / VMFS block size** because its blocks are now allocated
- **tbz = 0**, because the blocks have been zeroed

In this phase, a thin disk is similar in behavior and performance as a zeroed thick disk. Notice in Table 2, that the values in the Post-Zeroing column are the same.

Table 2. The values of **tbz** and **nb** in the zeroing and post-zeroing phases for a thin and thick disk each 12GB in size—VMFS block size in this example is the default 1MB

Allocation Type	Pre-allocated	Zeroing
Zeroed thick (12G)	tbz = 12288 to 0, nb = 12288	tbz=0, nb = 12288
Thin (12G)	tbz = 0, nb = 0 to 12288	tbz=0, nb = 12288

Acquiring Actual Values

During testing, the values for **tbz** and **nb** for a disk were determined as follows:

1. We used the `vmkfstools` command:

```
vmkfstools -D <flat-disk.vmdk>
```

We examined `/var/log/messages` for **nb** and **tbz** data:

```
vmkernel: FS3: 142: <START flat-disk.vmdk>
vmkernel: Lock [type 10c00001 offset 64833536 v 734, hb offset 3940352
vmkernel: Addr <4, 121, 193>, gen 592, links 1, type reg, flags 0x0, uid 0, gid 0, mode 600
vmkernel: len 12884901888, nb 12285 tbz 0, cow 0, zla 3, bs 1048576
```

The last line of the log displayed the **nb** and **tbz** blocks. In the above example, the disk is fully grown and zeroed.

References

[1] "Does fragmentation affect VMFS datastores?" VMware knowledge base. Aug 14, 2009. http://kb.vmware.com/kb/1006810

[2] *Performance Best Practices for VMware vSphere 4.0*. VMware guide. Jun 30, 2009. http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf

[3] "VMware VMFS." VMware product datasheet. Accessed Sep 29, 2009. http://www.vmware.com/pdf/vmfs_datasheet.pdf

[4] *vSphere Basic System Administration*. VMware guide. Oct 21, 2009. http://www.vmware.com/support/pubs

About the Authors

Mandar Kulkarni is a performance engineer at VMware. In this role, his primary focus is to analyze and help improve the performance of the storage stack in VMware vSphere. He received his master's degree in computer science and engineering from the University of Florida at Gainesville.

Sunil Satnur has been working for VMware performance engineering since early 2006. He has studied, published, and presented ESX storage stack scalability for both server and desktop workloads. He received his master's degree in computer science from Stony Brook University.

Acknowledgements

This performance study required the careful designing and running of various experiments that involved discussions between team members in the VMware Performance, Core Storage, and VMFS teams. The authors would like to thank Scott Drummonds, Glen McCready, Abhijit Paithankar, and Devaki Kulkarni for providing very useful insights on various components of the study right from experiment design to analysis of results. The authors would also like to thank Julie Brodeur from the Outbound Performance team for writing the paper in a simple and comprehensible way.