VM Performance on Flash Storage, Part 2: Thin vs. Thick Provisioning

This post is Part 2 in our series exploring how flash impacts VM performance and deployment strategies. Our first post covered the impacts of VM host / storage array block alignment, this post covers the performance differences between different VM types, and the third post will cover how different LUN and datastore sizes impact performance.

Introduction: The Results Up Front

Storage provisioning in virtual infrastructure involves multiple steps with crucial choices and decisions. One of them is the data format of the virtual disks. VMware vSphere virtual disks or VMDKs (virtual machine disks) can be provisioned in three different formats: Thin, Lazy Zero Thick (aka Flat), or Eager Zero Thick.

The differences lie in the way the data is pre-allocated or whether blocks are zeroed at creation, or run-time. These concepts are well understood in the community, and several years ago VMware set out to prove that there was no performance difference between these three VM types and published comprehensive test results to prove it. Regardless, this remains a topic of ongoing debate, and many customers still believe they see significant performance differences between the three VM types.

We wanted to test how the move to vSphere 5 and the move to all-Flash storage might impact the performance of the different VM types. You can read about our detailed testing methodology below, but the short answer is: **it just doesn't matter, all VM types perform the same.** There are, however, operational differences that Pure Storage introduces, given our support for inline deduplication, pattern removal, thin provisioning, VAAI, and the raw performance of Flash. Our findings are as follows:

- **Same Performance:** Pick any VM type that makes sense for your environment and management processes, they all perform the same (Thin, Lazy Zero Thick, and Eager Zero Thick).
- **Same Provisioning Time:** In additional to all performing the same, because of Pure Storage's VAAI support for Write Same, all VM types can be provisioned in the same amount of time (generally <5 seconds)
- **Same Space Usage:** because of the inline deduplication/compression, pattern (zero) removal, and thin provisioning in the Pure Storage FlashArray, all VM types will also take the same amount of space on disk – it just doesn't matter, Pure Storage always only stores the data, and removes all duplicate data.

Read on to learn more about the testing we did to prove-out these recommendations.

Background Reading: VM Types Explained

One of the first decisions in SAN storage provisioning is making the LUN decision: determining what RAID level for the LUN based on the application I/O profile to provision a physical LUN to ESXi. This task is usually done based on prior experience with the application or some help from vendor's best practice guide.

The next step is to choose a suitable virtual disk format for VM's virtual disk. Here are three formats of virtual disks:

- **Thin:** In thin provisioned disks, the size of the VMDK (at any point in time) is as much as the amount of data written out from the VM. So if you provision a 1 TB virtual drive and the VM only wrote 200GB then the size of the vmdk on disk is 200GB. The key thing to note is that the storage is zeroed on demand and data written out.
- **Lazy Zero Thick (aka Flat):** In Flat or Lazy Zero Thick format, the VMDK is provisioned and whenever a guest issues a write it is zeroed first and then the data is written. The size of the VMDK on the datastore is same as the size of the virtual disk that was created.
- **Eager Zero Thick:** With Eager Zero Thick, the VMDK is pre-zeroed and assigned to the guest during provisioning of the VMFS volume.

With the exception of VMware Fault Tolerance (FT) feature (see VMware KB Article: 1004617 and page 59 of vSphere 5.0 Performance best practice) and Microsoft clustering (page 8 of Setup of MSCS guide) there is no hard requirement to use thick or Eager zeroed data format. The size of the VMDK on the datastore is same as the size of the virtual disk that was created.

The following table summarizes the three VM types, and their implications on space usage, performance, and provisioning time (click to enlarge):

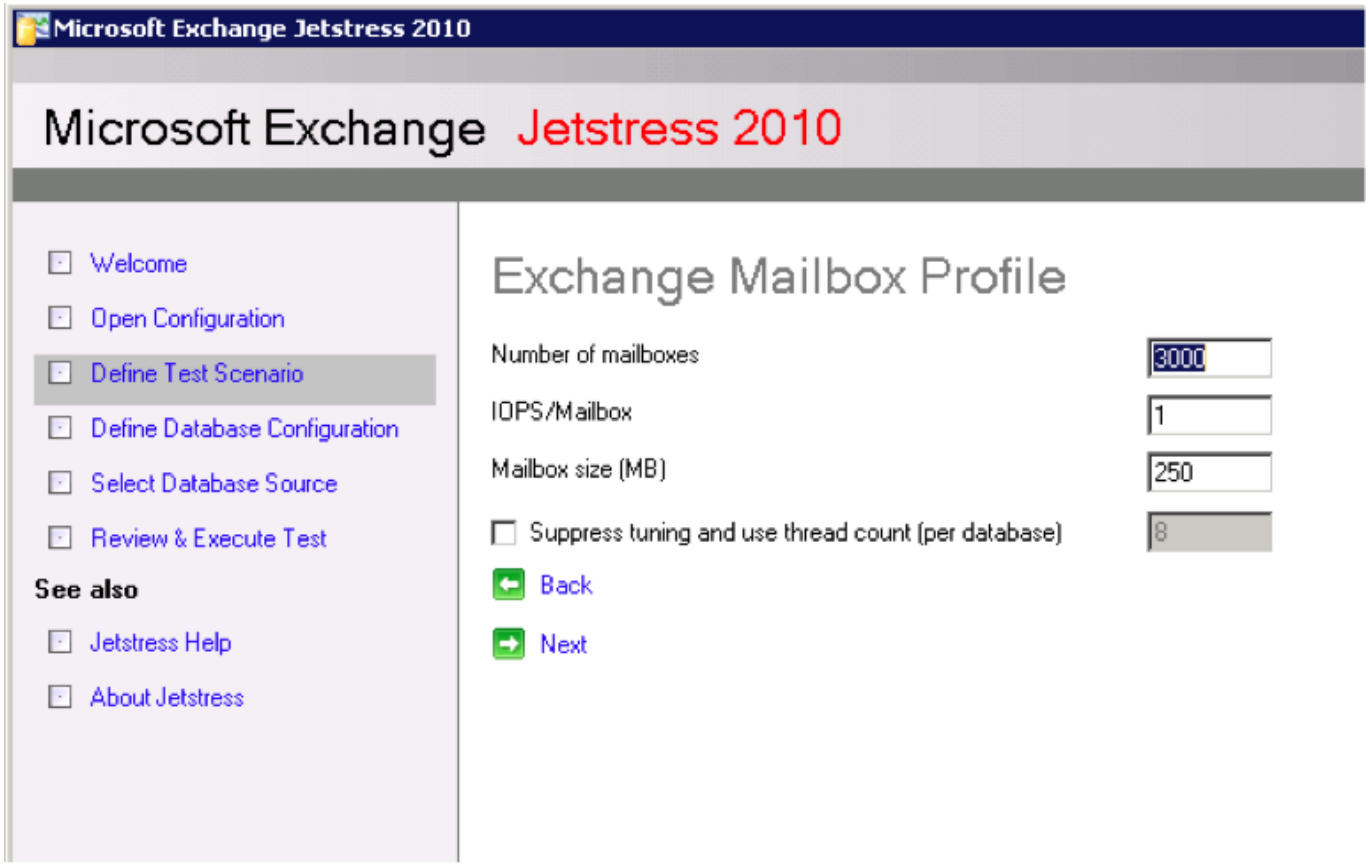
VM Type	Overview		Space Usage (Example 50GB VM with 10GBs Actual Data)					Performance		VM Creation Time		
	Space Allocated	Blocks Zeroed	In VMFS	On the Storage Array: Thick Provisioned Array	On the Storage Array: Thin Provisioned Array	On the Storage Array: Thin Prov Array With Zero Reclaim	On the Storage Array: Pure Storage FlashArray	Disk-Based Storage Array	Pure Storage FlashArray	Disk-Based Storage Array	Disk-Based Storage Array w/ VAAI	Pure Storage FlashArray w/ VAAI
Eager Zero Thick	At VM creation	At VM creation	50GB	50GB	50GB	10GB	2GB	Same (Disk Latency)	Same (Flash Latency)	~10 mins	< 30 sec	4 sec
Lazy Zero Thick	At VM creation	When block first written	50GB	50GB	10GB	10GB	2GB	Same (Disk Latency)	Same (Flash Latency)	~ 1 min	< 30 sec	0.2 sec
Thin	When block first written	When block first written	10GB	10GB	10GB	10GB	2GB	Same (Disk Latency)	Same (Flash Latency)	< 30 sec	< 30 sec	0.1 sec

The Details: Testing Validation

We tested with I/O intensive workload in our lab with the three different data formats on data stores carved out Pure Storage FlashArray and exposed to three similar Windows 2008 R2 VMs.

Workload

Microsoft Exchange server Jetstress 2010 was used to simulate a real life I/O intensive workload. The test run was configured to do a performance test on exchange mailbox profile. The run was done on two databases with 3 threads per database and number of copies per database was 1. IOPS per mailbox was 1. The Jetstress profile is shown below:



Hardware & Software Configuration

- VMware vSphere cluster (vSphere 5.0 build 504890) with 8 hosts with one host actually being used for the test
- The cluster was provisioned with three 1 TB LUNs carved out on Pure Storage FlashArray
- Two SAN switches (one Brocade and another Cisco MDS).
- Both the log volume and the database volume were stored on the same 1 TB LUN that was provisioned to the VMs
- Commodity server with 2 sockets Intel Westmere processors (2 Xeon x5690, 12 X 3.47GHz, total of 24 logical CPUs with Hyper Threading on)
- Memory: 192 GB @ 1333 MHz
- FC HBA: Qlogic ISP2532 based 8 Gb adapter (two ports)
- Windows 2008 R2 SP1 Enterprise 64bit with 4 vCPU and 8 GB of memory (with paravirtual adapter driver)

Results

As summarized above, the results showed substantially similar performance between the three VM types (click to enlarge):

Total I/O Performance

	MSExchange Database ==> Instances	I/O Database Reads Average Latency (msec)	I/O Database Writes Average Latency (msec)	I/O Database Reads/sec	I/O Database Writes/sec	I/O Database Reads Average Bytes	I/O Database Writes Average Bytes	I/O Log Reads Average Latency (msec)	I/O Log Writes Average Latency (msec)	I/O Log Reads/sec	I/O Log Writes/sec	I/O Log Reads Average Bytes	I/O Log Writes Average Bytes
Eager Zero Thick	Instance3444.1	1.372	2.020	1029.092	750.548	39987.069	34275.476	0.000	0.543	0.000	343.162	0.000	5271.147
	Instance3444.2	1.401	2.022	1027.562	751.004	39988.805	34276.204	0.000	0.542	0.000	344.444	0.000	5255.333
Lazy Zero Thick (Flat)	Instance2372.1	1.320	2.009	1052.023	769.030	39827.645	34244.835	0.000	0.569	0.000	345.088	0.000	5347.880
	Instance2372.2	1.354	2.011	1045.664	766.588	39870.256	34251.038	0.000	0.568	0.000	347.091	0.000	5339.350
Thin	Instance2208.1	1.333	2.032	1050.119	764.410	39842.670	34254.579	0.000	0.564	0.000	343.959	0.000	5297.460
	Instance2208.2	1.374	2.035	1046.202	761.276	39865.743	34271.439	0.000	0.564	0.000	345.181	0.000	5309.216

Eager Zero Thick

Host System Performance

Counter	Average	Minimum	Maximum
% Processor Time	6.608	5.067	7.822

Lazy Zero Thick (Flat)

Host System Performance

Counter	Average	Minimum	Maximum
% Processor Time	6.757	5.457	7.952

Thin

Host System Performance

Counter	Average	Minimum	Maximum
% Processor Time	6.725	5.223	8.290

Observations

- There were no LUN decisions while creating the LUN on Pure Storage FlashArray as there is no user defined RAID setting to worry about. More on this topic in a previous blog article here.
- The results confirm that there is no difference in performance even while running an I/O intensive workload like Microsoft JetStress on Pure Storage FlashArray! Isn't that one less decision you need to worry about?
- Also it is worthwhile to note that there was no increase in CPU utilization on the guest (they were more or less the same with average CPU around 6.7 %)
- VAAI supported arrays, like Pure Storage FlashArray, that support block zero primitive, so zeroing blocks is a non-issue. Block zeroing is fast on the FlashArray to begin with and it is even faster with the VAAI integration. This makes provisioning eager zeroed thick format in matter of seconds rather than minutes.
- In the case of Zeroed thick luns provisioning, when the VM the on demand zeroing is issued we don't incur any performance penalty.
- Fragmentation is a non-issue with thin provisioned disks in Pure Storage FlashArray as the data is spread across all the available flash drives. There is no penalty for allocating the data blocks on demand.
- With VMware VAAI atomic-test-and-set (ATS) providing smaller grain locking, which is supported on FlashArray, metadata updates in a clustered host scenario in a thin provisioned disks is not a performance concern any more.
- VMware vSphere alarms can be used in setting alerts on thin provisioned disks to take corrective action once a soft threshold is reached. The VAAI thin provisioning API also helps in issuing a warning message when a soft threshold is hit instead of panic'ing the guest VM.
- On the Pure Storage FlashArray side, alarms and alerts accomplish the same job so thin on thin is not an issue any more.
- In the FlashArray backend, the average latencies on read/writes were less than 1 msec during the test
- On a personal note, I have run jetstress in the past with various storage vendors, flash cards and have not seen 1 -2 msec read/write response time for I/Os.

All in all it does not really matter what data format you select for the application, you can just create a LUN and expose to a cluster in two-step process. Then carve out a vmdk in any data format and you are done. No more LUN decisions, and data format decisions; that's the beauty of all flash based FlashArray.

Conclusion

Yet another benefit of using the Pure Storage FlashArray in your data center: you never have to worry about the data formats on your virtual machine disks.

Related work

1. Although a bit dated, this VMware paper – Performance study of VMware vStorage thin provisioning (vSphere 4.0) – provides a nice comparative study of thin and thick data formats.
2. Best practices for using virtual disk thin provisioning white paper
3. EMC's take on thin vs. thick provisioning on Chad's blog
4. VMware documentation