kaminario.

WHITE PAPER

VMware vSphere 6 Reference Architecture for the Kaminario K2 All-Flash Storage Array

February 2017

TABLE OF CONTENTS

Executive Summary

The Kaminario K2 all-flash array is used to host various environments in the datacenter. Virtualized environments, that are used to increase the utilization of servers and increase productivity, are widely used. VMware vSphere is a great example of a virtual environment that is able to take advantage of what the K2 all-flash array has to offer.

The K2 has various integration points with VMware products, such as VMware ESXi, VMware SRM, VMware and VMware vRealize Log Insight. For each of these products, Kaminario has a unique value proposition and it is highly suggested to follow Kaminario's best practices and reference guides in order to achieve the highest synergies possible.

This reference architecture describes in detail all integration points of the K2 all-flash array with VMware ESXi, including VAAI support, best practices, VMware Native-Multipathing (NMP). In addition, this document describes different integrations Kaminario has with vCenter, Site Recovery Manager (SRM) and Log Insight.

The document includes examples and screenshots that facilitate easy understanding of the different settings and clearly demonstrate the tradeoffs between different options of configurations. Examples are presented in different formats and methods such as esxcli and PowerCLI which can be easily replaced with any other method if desired.

Content in this document is quite technical and is intended to be used by pre-sales engineers, system and storage administrators and customers who want to deploy the Kaminario K2 in vSphere-based environments.

Introduction to K2

The K2 all-flash array is an appliance which is a combination of tested and packaged hardware, software and services. K2's Gen6 hardware platform is based on leading off-the-shelf enterprise components that enable K2's software-defined architecture and software stack. The K2 runs Kaminario VisionOS™, the next-generation flash operating system stack, that provides the core software and advanced data services framework.

VisionOS enables modular components and services that demonstrate a superior value proposition across a real scale-out storage platform, both in innovation and in ease of use:

- **DataShrink** – Data reduction features and capabilities are mandatory for economics of flash storage. With differentiating inline, global, adaptive and selective deduplication, together with inline byte aligned compression, thin provisioning and zero detection, Kaminario is able to establish itself a cost-efficiency leader of flash storage.
- **DataProtect** – Kaminario values its customers' data more than anything. Native array based snapshots and replication allow for returning to any point in time in any site. Data-at-rest AES256 encryption makes sure that data is kept private and safe at all times. A highly resilient design of no single point of failure, non-disruptive upgrades (NDU) and a robust RAID scheme facilitate 99.999% of data availability.
- **DataManage** – The K2 can be managed by various means. Internal management includes an intuitive web-based GUI, a scriptable CLI and a fully programmable RESTful API platform.
- **DataConnect** – K2's RESTful API allows for external applications of the IT eco-system to easily integrate and seamlessly manage the K2. This eco-system is constantly growing and includes: VMware vSphere, Microsoft VSS, OpenStack, Flocker (containers) and Cisco UCS director.

Best Practices for ESXi 6.0 Settings

There are many settings in ESXi that affect the performance and availability of the Kaminario K2 array. The following section describes the major parameters setting and for each parameter details the expected outcome of changing the values.

For the full list of Kaminario's best practices for VMware and for detailed instructions on how to configure ESXi, please refer to the "VMware ESXi 6 Host Configuration Guide" document that can be found on the Kaminario Support Portal. This guide includes some PowerCLI scripts to configure and validate the recommended settings detailed in it.

The following table summarizes Kaminario's recommended ESXi settings and give high level information on the importance of configuring each parameter.

Parameter	Scope	Default Value	Recommended Setting	Importance
Path Selection Policy	Per Volume, per ESXi host	Fixed	Round-Robin	Critical. Better utilization of K2 Ports
Number of Outstanding IOs With Competing Worlds	Per Volume, per ESXi host	32	256	Critical. Allows "pushing" more load to the K2.
Disk.SchedQuantum	Per ESXi host	8	64	Low. Less significant for All-Flash Arrays
VAAI Primitives	Per ESXi host	Enabled	Enabled	Critical. Offloads heavy workloads from the ESXi host to the K2
QLogic HBA Settings	Per ESXi host	Queue Depth=64, ZIO=1, IDT=100	Queue Depth=400, ZIO=6, IDT=1	Medium. Can increase performance significantly but may not always fit the whole eco-system
Emulex HBA Settings	Per ESXi host	lpfc_lun_queue_ depth = 30-128	lpfc_lun_queue_ depth = 30-128	Medium. Can increase performance significantly but may not always fit the whole eco-system

The following sections discuss the importance of each setting, its impact on performance and whether the setting is global or unique to the K2.

VMware Native Multipathing (NMP)

VMware's Native Multipathing (NMP) is the default multipathing plugin provided in vSphere. NMP itself is a plugin of PSA – Pluggable Storage Architecture, a modular framework for managing multipath plugins such as the NMP.

The NMP itself supports sub-plugins such as SATP (Storage Array Type Plugins) and PSP (Path Selection Plugins). The Kaminario K2 uses the K2 SATP to manage K2 volumes and the built-in VMW_PSP_RR PSP to allow multipathing in a round-robin manner.

Kaminario's Storage Array Type Plugins (SATP)

The Kaminario K2 all-flash storage array is an Active/Active storage array. As an Active/Active storage array, the Kaminario K2 gets the most out of its hardware with no compromises or idling resources.

Unlike other storage systems which makes use of ALUA (Asymmetric Logical Unit Access), K2's K-Nodes (storage controllers) can access any LUN in the system at any time from any K-Node – which makes it a true Active/Active system.

To get the best performance out of an Active/Active system, a user would spread the workload across all K-Nodes with all paths connected and active.

Therefore, setting the Path-Policy to Round-Robin for K2 LUNs ensures the best workload distribution. In addition, changing the IO path selection on every second command is Kaminario's recommendation for getting optimal performance.

For more details on how to configure Kaminario's SATP, please refer to Kaminario's Best Practices for VMware.

The following table compares performance between the Round-Robin PSP and the Fixed PSP. To compare the two PSPs, we ran the same workload of IOMeter on a single VM, once configured with Round-Robin and once with Fixed. IOMeter was configured the same in each run with a block size of 16KB, 50/50 R/W and 128 outstanding IOs to get maximum performance. Values in the next table are based on a 3 minutes run for each PSP.

Path Selection Policy	Throughput		IOPS		Latency	
	Average	Maximum	Average	Maximum	Average	Maximum
Fixed	1,062 MB/s	1,085 MB/s	68,197 IOPS	73,163 IOPS	0.68 ms	0.73 ms
Round-Robin	1,505 MB/s	1,566 MB/s	96,456 IOPS	100,257 IOPS	0.43 ms	0.54 ms
% Improvement	41%	44%	41%	37%	36%	26%

As the above comparison suggests, using Round-Robin as the Path Selection Policy can drastically improve performance for all metrics – Throughput, IOPS and Latency, making it the recommended PSP for the Kaminario K2.

ESXi Host Optimizations

Number of Outstanding IOs With Competing Worlds

The Number of Outstanding IOs With Competing Worlds (previously known as `Disk.SchedNumReqOutstanding` in ESXi 5.5 and before) parameter is used to throttle the queue length only when there is more than one virtual machine sending IO to the same Datastore. If, for example, a Datastore is configured with the default of 32 Outstanding IOs and there are 4 virtual machines running on that Datastore, each virtual machine would be reserved with 8 Outstanding IOs. If a Datastore is used by a single virtual machine, that virtual machine is not limited by 32 Outstanding IOs. It will be limited by the HBA's queue depth.

The parameter is configured per Datastore in every ESXi host, making it possible to configure the parameter for K2 volumes only with no impact on Datastores from other vendors.

Configuring the parameter with the same setting across a vSphere Cluster is crucial, otherwise performance degradation may be observed when virtual machines migrate across ESXi hosts in the cluster.

To demonstrate the impact of the parameter we use two virtual machines residing on a single K2 Datastore. Each of the two virtual machines runs an IOMeter workload of 16KB block size, 50/50 R/W and 128 outstanding IOs.

At first, we ran the workload with the parameter default value of 32, and while running, we inspect the K2 performance using the K2 GUI, and the ESXi using the esxtop utility. When looking at the K2 GUI, we can see that the two virtual machines have a total throughput of above 1,000 MB/s, a total of 66,000 IOPS and a consistent latency of 0.3 milliseconds.

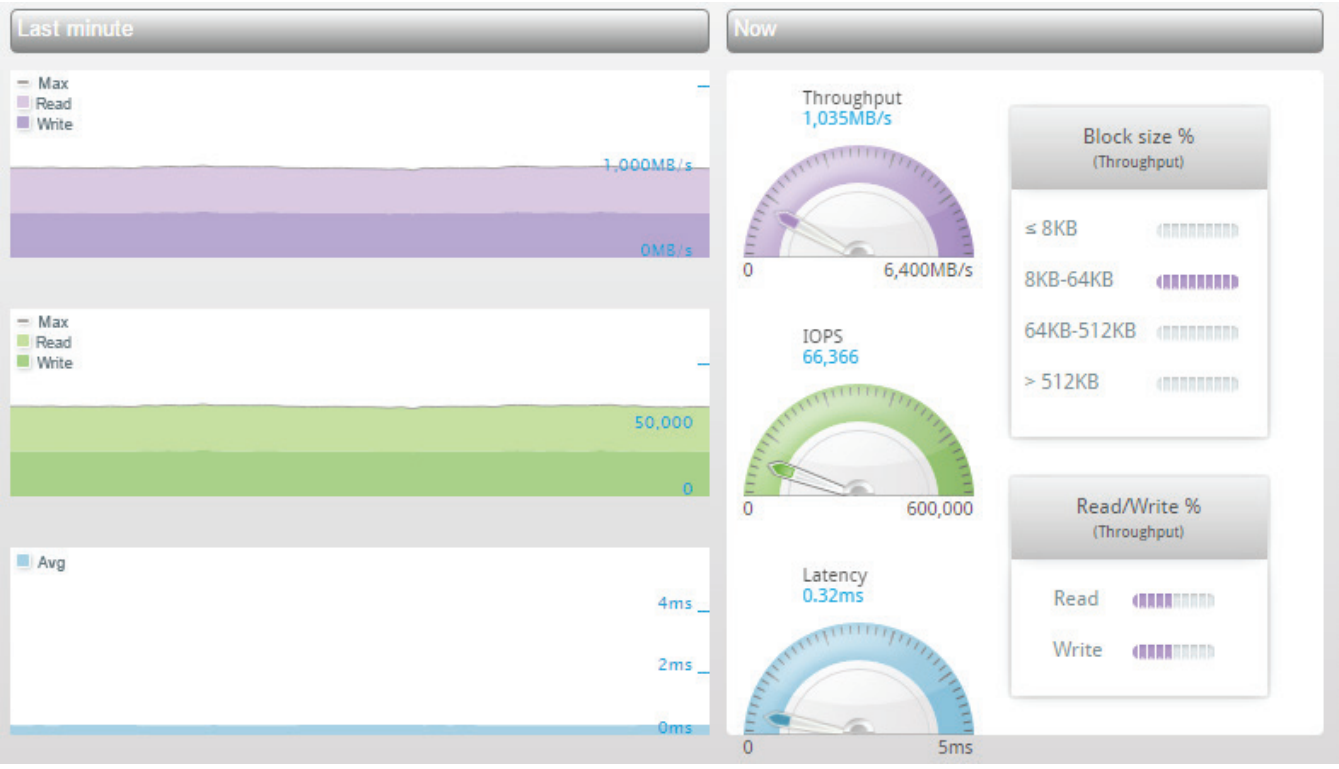


Figure 1: Performance of two virtual machines on a single K2 Datastore with default Outstanding IOs of 32

Using the `esxtop` utility, we can see that all 8 paths of the Datastore are in use, and as expected, load is distributed across all paths equally. An important metric to pay attention to is the KAVG/cmd, which indicates the average amount of time (in milliseconds) a command spends in the ESXi VMkernel. When the Outstanding IOs is configured to 32, we can see that the KAVG/cmd is relatively high – above 1ms, so commands are delayed in the VMkenrel which makes the virtual machines response time (GAVG/cmd) grow larger, and by turn, delay application running in the virtual machines.

```
 3:30:33pm up  1:39, 688 worlds, 4 VMs, 8 vCPUs; CPU load average: 0.21, 0.21, 0.21
```

DEVICE	PATH/WORLD/PARTITION	CMDS/s	READS/s	WRITES/s	MBREAD/s	MBWRTN/s	DAVG/cmd	KAVG/cmd	GAVG/cmd	QAVG/cmd
eui.0024f400d5892b6a	vmhba4:C0:T4:L21	8387.37	4008.45	4378.92	62.63	68.42	0.47	1.21	1.68	0.00
eui.0024f400d5892b6a	vmhba4:C0:T3:L21	8384.53	4107.81	4276.72	64.18	66.82	0.47	1.22	1.68	0.00
eui.0024f400d5892b6a	vmhba4:C0:T2:L21	8385.95	4119.16	4266.78	64.36	66.67	0.47	1.21	1.68	0.00
eui.0024f400d5892b6a	vmhba5:C0:T2:L21	8384.53	4126.26	4258.27	64.47	66.54	0.48	1.18	1.66	0.00
eui.0024f400d5892b6a	vmhba4:C0:T5:L21	8383.11	4238.39	4144.71	66.22	64.76	0.48	1.20	1.67	0.00
eui.0024f400d5892b6a	vmhba5:C0:T5:L21	8383.11	4261.11	4120.58	66.58	64.38	0.48	1.21	1.69	0.00
eui.0024f400d5892b6a	vmhba5:C0:T4:L21	8384.53	4275.30	4109.23	66.80	64.21	0.48	1.20	1.68	0.00
eui.0024f400d5892b6a	vmhba5:C0:T3:L21	8383.11	4356.21	4026.90	68.07	62.92	0.48	1.17	1.65	0.00

Figure 2: Output of esxtop when two virtual machines do IO on a single K2 Datastore with default Outstanding IOs of 32

While still running the IOMeter workload, we change the Outstanding IO of that Datastore to 256, and as expected, a drastic improvement in performance can be observed quite quickly. The total throughput has improved to over 2,000 MB/s, the total IOPS was doubled to 133,000 IOPS and the latency is consistent and still under 1ms.

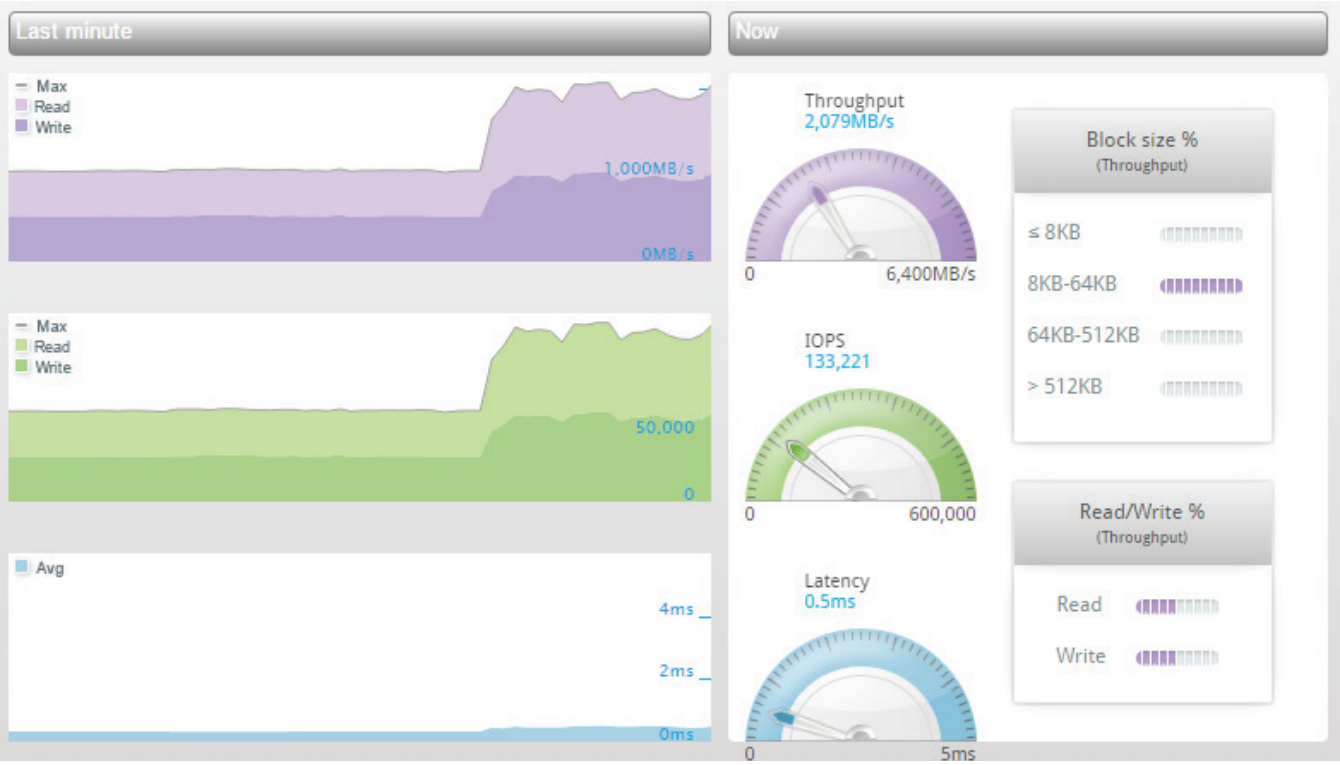


Figure 3: Performance of two virtual machines on a single K2 Datastore with Outstanding IOs configured to 256

In addition, when looking at the output of esxtop, we can see that the KAVG/cmd value is negligible and practically zero, meaning there's no delay in the VMkernel and virtual machines' latency equals the disk latency.

```
 3:51:49pm up  2:00, 689 worlds, 4 VMs, 8 vCPUs; CPU load average: 0.36, 0.27, 0.23

DEVICE                                 PATH/WORLD/PARTITION    CMDS/s  READS/s  WRITES/s  MBREAD/s  MBWRTN/s  DAVG/cmd  KAVG/cmd  GAVG/cmd  QAVG/cmd
eui.0024f400d5892b6a                   vmhba5:C0:T5:L21       16653.18  8231.51  8421.67    128.61    131.57      0.77      0.00      0.77      0.00
eui.0024f400d5892b6a                   vmhba4:C0:T4:L21       16653.56  8286.93  8366.63    129.47    130.73      0.75      0.00      0.76      0.00
eui.0024f400d5892b6a                   vmhba4:C0:T5:L21       16653.56  8299.07  8354.49    129.66    130.54      0.77      0.00      0.77      0.00
eui.0024f400d5892b6a                   vmhba4:C0:T3:L21       16653.18  8307.04  8346.14    129.79    130.41      0.77      0.00      0.77      0.00
eui.0024f400d5892b6a                   vmhba5:C0:T3:L21       16652.42  8308.18  8343.86    129.81    130.35      0.77      0.00      0.78      0.00
eui.0024f400d5892b6a                   vmhba4:C0:T2:L21       16661.15  8325.64  8335.51    130.09    130.24      0.78      0.00      0.78      0.00
eui.0024f400d5892b6a                   vmhba5:C0:T2:L21       16660.77  8352.97  8307.80    130.51    129.80      0.78      0.00      0.78      0.00
eui.0024f400d5892b6a                   vmhba5:C0:T4:L21       16651.28  8400.03  8251.25    131.24    128.93      0.76      0.00      0.76      0.00
```

Figure 4: Performance of two virtual machines on a single K2 Datastore with Outstanding IOs configured to 256

Disk.SchedQuantum

In situations when there are multiple virtual machines on a single Datastore, we might want a virtual machine to request more than the Outstanding IOs configured for that Datastore (as described in the previous section). Such case may be when a virtual machine requests sequential IO like in Database large operations such as full table scan.

If we take for example a single K2 Datastore configured with Outstanding IOs of 256 and 8 virtual machines running on top of that Datastore, each virtual machine would be served with 32 IOs at a time. Let's say that the first virtual machine is running a database and performs a full table scan (a sequential read operation). So as mentioned before, every time the virtual machine is serviced, it is serviced with 32 IOs.

Since full table scan is a sequential operation, we can improve it by allowing more operations (IOs) in each cycle the virtual machine is serviced, and by that we avoid the seek time back to the same location to complete the sequential read in the next cycle the virtual machine is serviced.

How many extra operations are allowed to a virtual machine performing a sequential operation is determined by the `Disk.SchedQuantum` parameter.

`Disk.SchedQuantum` defines the maximum number of consecutive sequential commands issued from a virtual machine before handling other virtual machines' operations. By default, this parameter is configured with a value of 8. Kaminario's recommendation is to set the `Disk.SchedQuantum` parameter to the maximum allowed value of 64.

Back to the example above – the virtual machine performing a full table scan would be serviced with 32 IOs at a time, and in addition, it would be serviced with extra 64 IOs as the `Disk.SchedQuantum` is configured.

To complete the explanation on the `Disk.SchedQuantum`, we should better understand what is considered as a sequential IO. In the context of the discussion here, sequential IO does not mean adjacent, but proximal. A proximal IO is an IO that is in the range of `Disk.SectorMaxDiff` value, which is by default set to 2,000 sectors. So, if the next IO is in the range of 2,000 sectors from the last IO, it would be considered as sequential and will be serviced, otherwise, the next virtual machine will be serviced.

The `Disk.SchedQuantum` parameter is a global parameter, meaning it has effect on all Datastores attached.

As mentioned before, the motivation behind this setting is to prevent the seek time back to the same place when servicing the virtual machine over the next cycles. In all-flash storage arrays such as the K2, seek time is insignificant and has almost no effect on the response time, thus this setting is less critical when working with the Kaminario K2 as the storage provider. However, setting the `Disk.SchedQuantum` may improve performance, especially when the Outstanding IOs is set to the maximum of 256 as recommended by Kaminario.

HBA Configuration

The default settings of an HBA should be satisfying for most workloads and environments. In some cases, depending on performance requirements, changing the HBA queue depth is a recommended method to gain better performance.

Keep in mind that changing the HBA queue depth is a system-wide change, meaning that all traffic on that HBA is influenced from the change. Such a change should be wisely considered if there are multiple different storage arrays connected to the same HBA, especially if storage arrays are from multiple vendors.

Qlogic HBAs

The following table details the description and recommended values for each parameter:

Parameter	Description	Default Value	Recommended Setting
ql2xoperationmode	This parameter enables the Zero Input Output (ZIO) operation mode. ZIO uses an adaptive algorithm to send a single interrupt for multiple SCSI commands instead of sending an interrupt on every SCSI command. Setting this feature minimize the number of interrupts sent.	Mode 1 – Qlogic reserved	Kaminario recommends to set ZIO to mode 6. Setting mode 6 enables the HBA to send an interrupt when the firmware has no commands in flight or when the Interrupt Delay Timer has reached.
ql2xintrdelaytimer	This parameter set the Interrupt Delay Timer (IDT). The Interrupt Delay Timer defines the firmware waiting time before generating an interrupt to the host to notify completion of the request.	100 ms	Kaminario recommends to set the Interrupt Delay Timer to 1 millisecond.
ql2xmaxqdepth	Defines the queue depth for the HBA.	64 (ESXi 5.x and above)	Kaminario recommends to set the queue depth to 400.

To list all available parameters for the `qlnativefc` driver in ESXi, use the following `esxcli` command:

```
esxcli system module parameters list -m qlnativefc
```

Notes:

- It is important to set the same settings in all HBAs in a cluster. Use the Kaminario Best Practices PowerCLI scripts to configure and validate the settings across a cluster.
- The Qlogic HBA parameter "Execution Throttle" is no longer used in VMware ESXi and is ignored by the `qlnativefc` driver.
- Changing the settings above requires to ***reboot*** the ESXi host before changes take effect.

Emulex HBAs

Emulex HBAs default settings should be satisfying for most workloads and environments. If performance fine tuning is required, recommended parameters are queue-depth related only.

Emulex HBAs offer three different queues as described in the following table:

Parameter (Queue)	Description	Default Value	Minimum Value	Maximum Value	Recommended Setting
lpfc_hba_queue_depth	The maximum number of commands queued to a single Emulex adapter.	8192	32	8192	8192
lpfc_lun_queue_depth	The maximum number of commands queued to a single LUN (Volume).	30	1	128	30-128
lpfc_tgt_queue_depth	The maximum number of commands queued to a single target.	8192	10	8192	8192

The *lpfc_**hba**_queue_depth* and the *lpfc_**tgt**_queue_depth* parameters default values are the maximum allowed values, so there's no need to change these.

On the other hand, the *lpfc_**lun**_queue_depth* parameter default value is set to 30, whilst the maximum allowed value is 128. The default setting of 30 can be adjusted as needed.

In general, a high IOPS-loaded environment (i.e. OLTP) may require a higher value (i.e. – 128), while other environments (i.e. OLAP) may be satisfied with a lower value (i.e. – 64).

Notes:

- It is important to set the same settings in all HBAs in a cluster. Use the Kaminario Best Practices PowerCLI scripts to configure and validate the settings across a cluster.
- Changing the settings above requires to reboot the ESXi host before changes take effect.

VMware VAAI Primitives

vStorage APIs for Array Integration (VAAI), is a VMware API that enables storage vendors to better integrate with vSphere, VMware's virtualization platform. The way this integration works is by implementing standard SCSI commands (detailed below) per VMware requirements so that vSphere can offload some of the storage-related tasks (for example, block zeroing) to the storage array without burdening the hypervisor. By utilizing VAAI an ESXi host's CPU, memory, and storage bandwidth can be minimized to allow the ESXi host to manage and run virtual machines in the most optimal conditions.

The Kaminario K2 support the primitives detailed below, allowing a vSphere cluster to get the most out of the Kaminario K2 all-flash storage array.

Zero Blocks (Write Same)

The Zero Blocks VAAI primitive is used to zero out a range of blocks on a LUN by sending WRITE SAME (0x93) SCSI commands to the K2. When provisioning a thick virtual disk, the ESXi host must zero a range of blocks before writing the actual data, thus writing zeros to that range of blocks. Once this range of blocks was zeroed, only then the actual data can be written to the storage array.

By using the WRITE SAME SCSI command, the ESXi host can command the storage array to zero out a range of blocks without sending the actual payload of zeros, so throughput consumption over the storage network is significantly lower.

Not using the WRITE SAME primitive can lead to higher latency and throughput usage. The higher latency can appear in different phases of the virtual machine life cycle, depending on the virtual disk provisioning method:

- **Thick Eager Zero** – The entire virtual disk capacity is reserved and zeroed during the virtual machine provisioning phase. This type of virtual disk may present some slightly higher latency and higher throughput usage during the virtual machine provisioning phase. Using Thick Eager Zero provisioning will allow an application to write data without zeroing blocks since the whole virtual disk is zeroed during the provisioning phase, making it slightly faster to use than other virtual disk types. As can be seen in the comparison table below – Thick Eager Zero virtual disks gain the most benefit from enabling the Zero Blocks primitive. Because of the above, Thick Eager Zero is the recommended virtual disk type by most database vendors.
- **Thick Lazy Zero** – The entire virtual disk capacity is reserved but zeroed-out on-demand. That means that new writes triggers zeroing the required capacity before writing the actual data – just like in Thin provisioned virtual disks, but with the entire capacity reserved. This type of virtual disk provisioning may present some higher latency during the ongoing activity of the virtual machine. When used on the K2, that performance degradation is negligible and almost unnoticed, and might become an issue only in some very latency sensitive applications.
- **Thin** – Capacity is not reserved and zeroed on-demand. New writes to the virtual disk triggers space allocation and zeroing on demand. This type of virtual disk provisioning may present some higher latency during the ongoing activity of the virtual machine. Just like in Thick Lazy virtual disks, when Thin provisioning is used on the K2, that performance degradation is very negligible, and might become an issue only in some very latency sensitive applications.

The Kaminario K2 support the Zero Blocks primitive and zero-out blocks in a 1MB block size. Depending on the K2 configuration (Single K-Block, Dual k-Blocks etc.) and the active workload, the K2 will run WRITE SAME operations in the maximum allowed throughput. The following table describe the time it took to provision a 2TB VMDK with Zero Block enabled and disabled on a Dual K-Blocks K2 system with no throttling:

Provisioning Type	Zero Blocks Enabled		Zero Blocks Disabled	
	Duration (mm:ss)	Avg. Rate	Duration (mm:ss)	Avg. Rate
Thin	Instant	N/A	Instant	N/A
Thick Lazy Zero	00:12	N/A	00:12	N/A
Thick Eager Zero	13:27	2,600 MB/s	28:20	1,230 MB/s

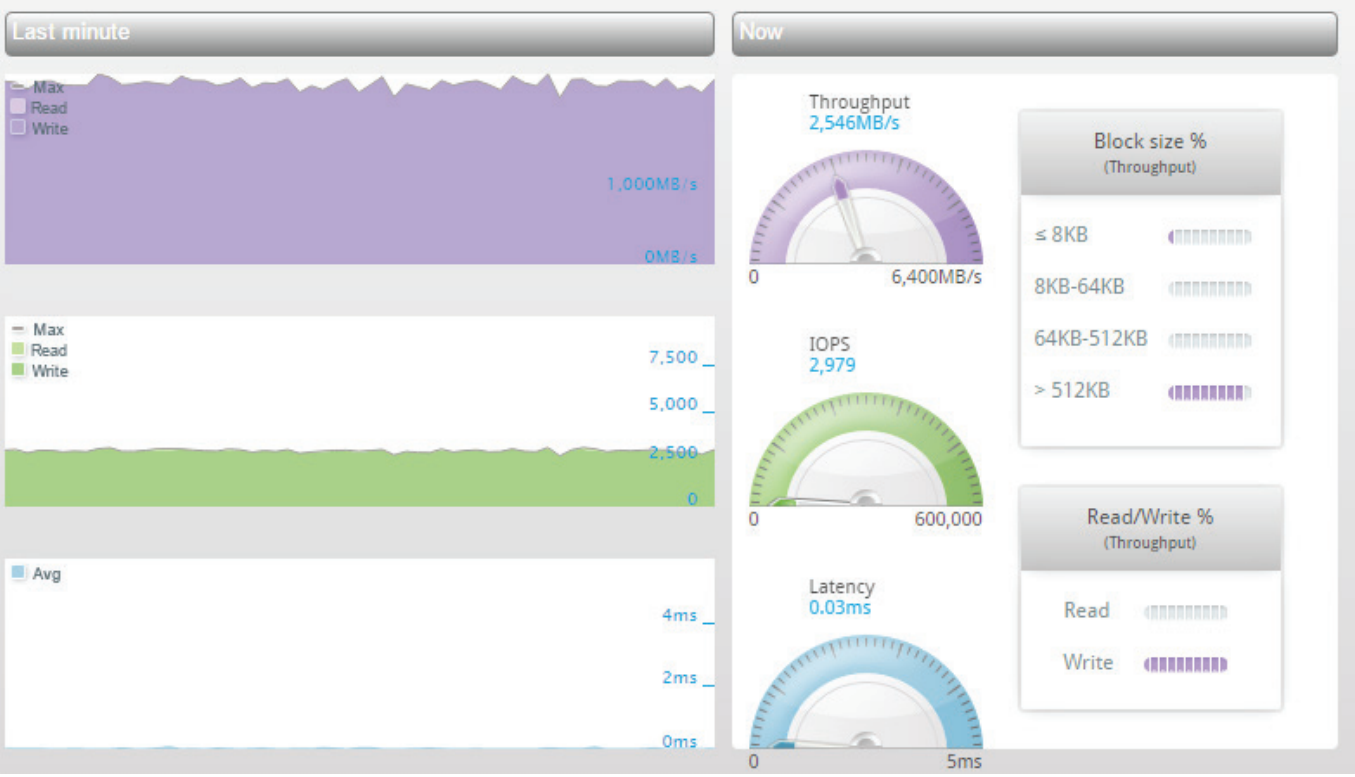


Figure 5: K2 Performance while provisioning an Eager Zero virtual disk with Zero Blocks (Write Same) *enabled*

Full Copy (XCOPY)

The Full Copy VAAI primitive is used to copy a range of blocks from one LUN to another LUN in the same storage array by using the XCOPY (0x83) SCSI command. When a copy operation is required like in Storage vMotion or cloning a virtual machine, the XCOPY SCSI command come in handy.

Without using the XCOPY SCSI command, the ESXi host has to perform all reads and writes to the storage array and utilize its own resources like CPU and RAM for the task, and in addition, it would consume a chunk of the available storage bandwidth.

With XCOPY enabled, the ESXi host sends XCOPY SCSI commands to the storage array to offload the operation. Each XCOPY command orders the storage array to copy a certain number of blocks internally inside the storage array, so the ESXi host does not read or write any data from/to disk with a negligible amount of bandwidth usage. Moreover, using the XCOPY primitive shortens such operations drastically.

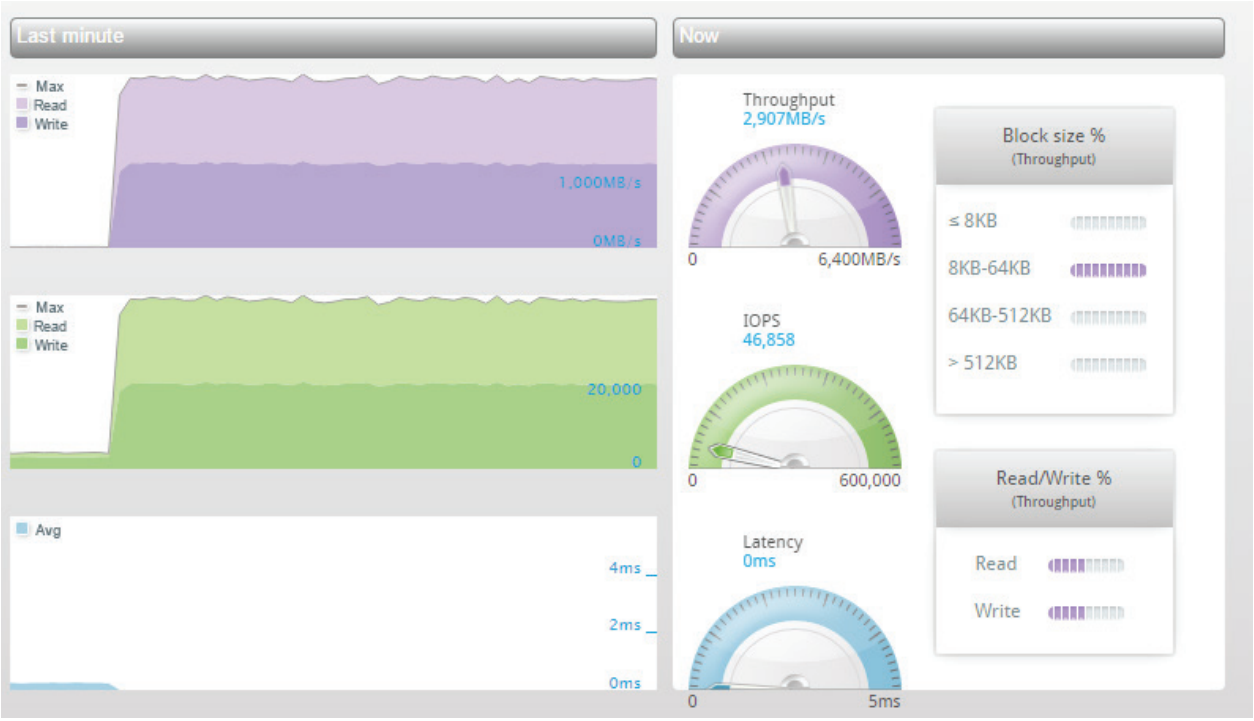


Figure 6: K2 Performance while running Storage vMotion with Full Copy (XCOPY) *enabled*

Looking at `esxtop` we can see commands are sent to the K2, but no throughput consumed:

```
7:06:39pm up 17 days 22:40, 691 worlds, 2 VMs, 20 vCPUs; CPU load average: 0.04, 0.04, 0.03

ADAPTR PATH                NPTH    CMDS/s  READS/s WRITES/s MBREAD/s MBWRTN/s DAVG/cmd KAVG/cmd GAVG/cmd QAVG/cmd
vmhba0 -                      0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba1 -                      1      0.20     0.00     0.20     0.00     0.00    15.81     0.02    15.82     0.00
vmhba2 -                    100 10376.96     1.57    99.77     0.01     0.40     0.46     0.01     0.47     0.00
vmhba3 -                    100 10376.76     0.98   101.14     0.00     0.40     0.49     0.01     0.50     0.00
vmhba32 -                     1      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba33 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba34 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba35 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba36 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba37 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
```

Figure 7: esxtop output while running Storage vMotion with Full Copy (XCOPY) *enabled*. Note the MBREAD/s and MBWRTN/s columns are near zero

While doing the same Storage vMotion operation with Full Copy disabled, we see bandwidth consumption in `esxtop` output, and lower copying rate in the K2 GUI:

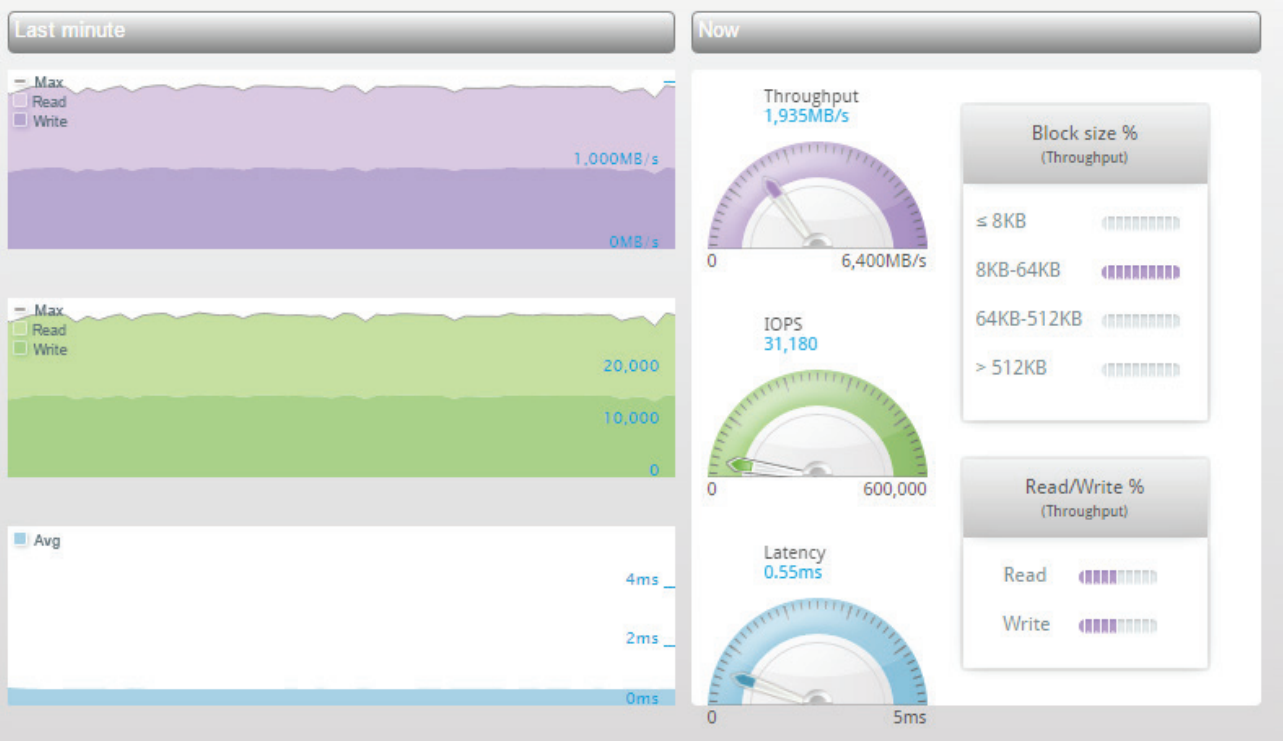


Figure 8: K2 Performance while running Storage vMotion with Full Copy (XCOPY) *disabled*

```
8:00:52pm up 17 days 23:34, 691 worlds, 2 VMs, 20 vCPUs; CPU load average: 0.03, 0.02, 0.02

ADAPTR PATH                NPTH    CMDS/s  READS/s WRITES/s MBREAD/s MBWRTN/s DAVG/cmd KAVG/cmd GAVG/cmd QAVG/cmd
vmhba0 -                      0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba1 -                      1      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba2 -                    100  15880.04  7910.59  7969.45   494.35   494.40     0.99     0.00     1.00     0.00
vmhba3 -                    100  15882.22  7910.59  7971.63   494.35   494.60     0.93     0.00     0.94     0.00
vmhba32 -                     1      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba33 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba34 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba35 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba36 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
vmhba37 -                     0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
```

Figure 9: esxtop output while running Storage vMotion with Full Copy (XCOPY) *disabled*. Note the MBREAD/s and MBWRTN/s columns reflects the K2 reported throughput

The following table shows the timing of moving a 2TB Thick Eager Zero virtual machine using Storage vMotion from one Datastore to another Datastore in the same Dual K-Block K2 system:

Full Copy *Enabled*			Full Copy *Disabled*		
Duration (mm:ss)	Avg. K2 Rate	Bandwidth consumption	Duration (mm:ss)	Avg. K2 Rate	Bandwidth consumption
26:15	2,900 MB/s	Near zero	36:30	1,915 MB/s	1,915 MB/s

Atomic Test & Set (ATS)

The Atomic Test & Set VAAI primitive is used to guarantee atomicity of access to VMware metadata and files on a VMFS Datastore by using the COMPARE AND WRITE (0x89) SCSI command.

Before VMware added the ATS capability, it used the SCSI-2 reserve and release lock mechanism to enable a vSphere cluster to lock and release a LUN. When using the SCSI-2 mechanism, a lock is done on an entire LUN, making other nodes in the cluster wait for the lock holder to release it before any other operation can be done.

With ATS enabled an ESXi host can lock a very small and specific number of blocks, allowing nodes in the cluster to lock other blocks concurrently. Using ATS, a cluster can provision more virtual machines and manage them in parallel with no lock conflicts or delays.

Using ATS is encouraged in all vSphere environments, but in large scale environments with high density of virtual machines per Datastore like VDI for example, using ATS is a must.

When creating a new K2 volume which is targeted to be a VMFS Datastore, the "Enable VMFS (SCSI-2 Reservations)" option should be checked to support VMFS formatting. When ESXi formats a new VMFS Datastore, it uses a SCSI-2 Reservation to lock the entire LUN during the format phase. Once the LUN is formatted with VMFS, the SCSI-2 lock is released and locking mechanism switches to ATS automatically.

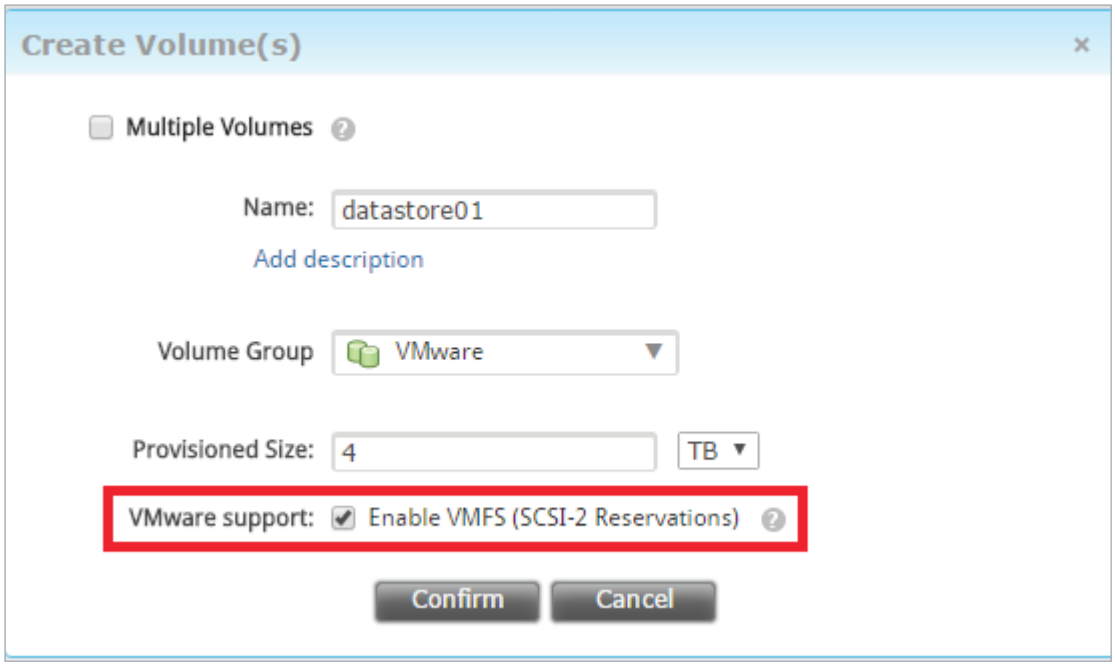


Figure 10: Creating a VMFS LUN in the K2 with ATS support

Block Delete (UNMAP)

The Block Delete VAAI primitive is used to mark deleted space as available to allow the K2 to reclaim that space by using the UNMAP (0x42) SCSI command.

The VMFS file system is managed by vSphere, meaning that the vSphere cluster is responsible for moving data around, creating new files and deleting files not needed any more. When such operations take place, the storage array is not aware of storage spaces that were freed up and treats them as allocated and used capacity.

By sending the UNMAP SCSI command to the storage array, the vSphere cluster informs the storage array that specific storage spaces can be reclaimed and are not in use anymore by the vSphere cluster, allowing the storage array to use that space for other purposes.

Moreover, having the storage array synced with the VMFS state, the Kaminario K2 capacity usage is correlated with vSphere reports and vice versa. The UNMAP operation can create high throughput internally in the K2 as metadata is updated. Starting with Kaminario K2 v5.8, it is possible to manually throttle the available throughput for UNMAP operations using the K2 CLI:

```
\=>> system help scsi-unmap-throttling-set
28-Dec-2016,10:25:09 IST
Set throttling rate for SCSI UNMAP commands
```

Named arguments:

```
  rate - A value between 0-9 to determine the throttling rate (0 - no throttling, 9 - max
throttling)
```

The following example demonstrate how to recognize a Datastore's UUID and run an UNMAP job for that Datastore. First, we'll use the `esxcli` command line to list the available Datastores and their UUIDs:

```
[root@sol-esx16:~] esxcli storage vmfs extent list
```

From the output of the command above, we'll copy the UUID of the Datastore we want to UNMAP and use it in the next esxcli command line:

```
[root@sol-esx16:~] esxcli storage vmfs unmap --volume-uuid=585c1143-ef4b0064-aabb-0026b9fa0b9c
```

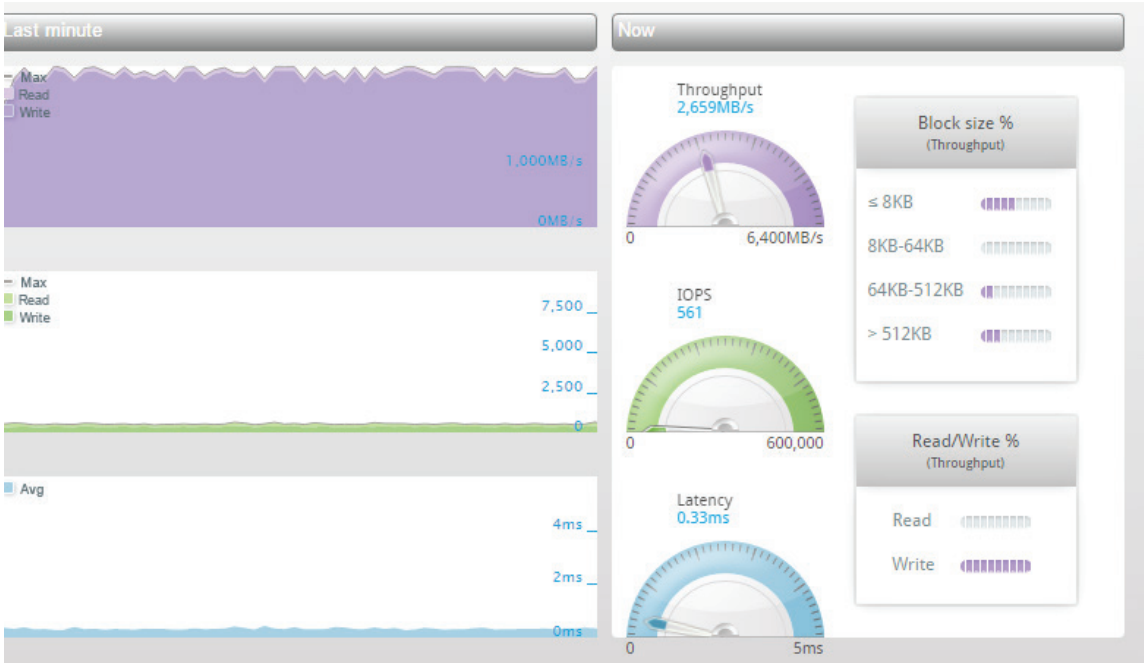


Figure 11: K2 Performance while running an UNMAP command from an ESXi host

In the next example, we limit the K2 UNMAP throughput to the minimum by setting the throttling mechanism as demonstrated in the following example from the K2 CLI:

```
\=>> system scsi-unmap-throttling-set rate=9
28-Dec-2016,10:29:16 IST
SUCCESS
```

Once the UNMAP throttling was set in the K2, we'll run an UNMAP job the same way we did before:

```
[root@sol-esx16:~] esxcli storage vmfs unmap --volume-uuid=586288f5-78061f93-149a-0026b9fa0b9c
```

Looking at the K2 GUI, we can see that UNMAP workload is throttled down 370 MB/s:

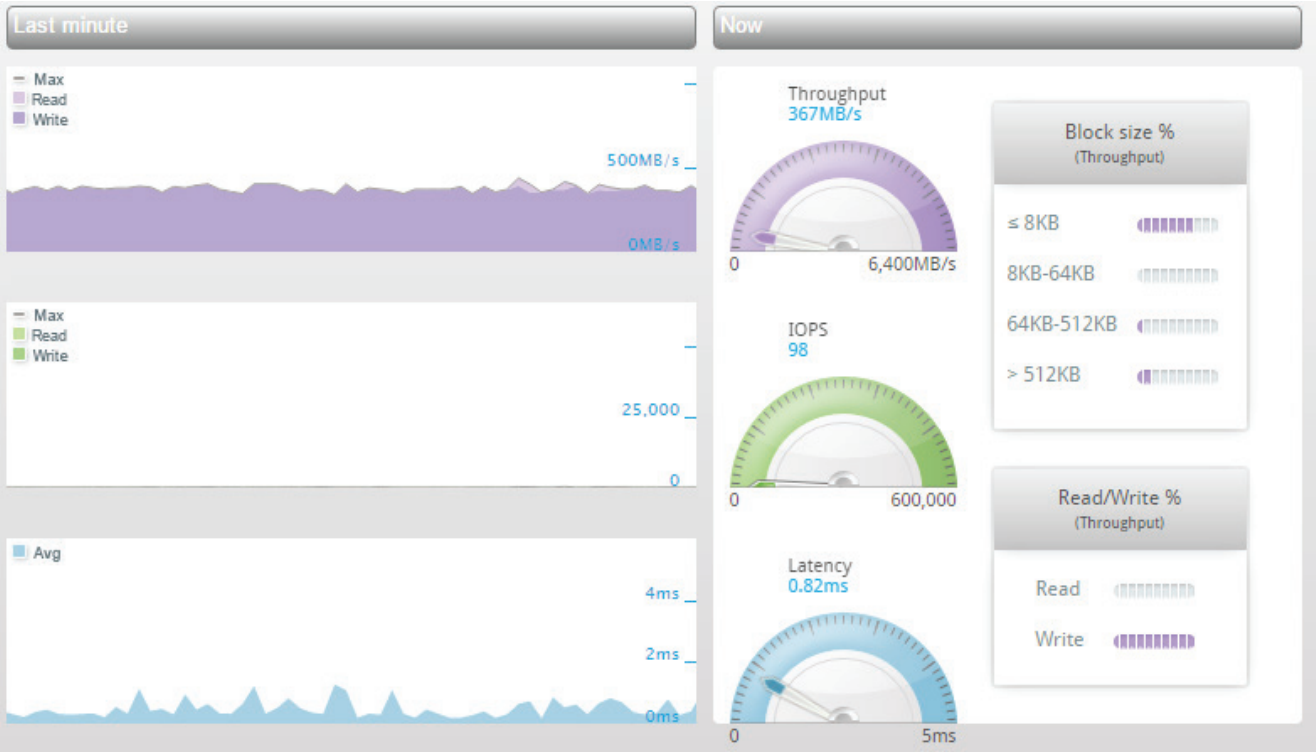


Figure 12: K2 Performance while running an UNMAP command from an ESXi host with UNMAP throttled down in the K2

In-Guest Space Reclamation

The section above (Section 4.1.4 Block Delete (UNMAP)) discussed the UNMAP primitive in relation to the VMFS filesystem and not the filesystems inside guest virtual machines like ext4 or NTFS for example.

Starting with ESXi 6.0 and Virtual Machine Hardware version 11 and above, it is possible to issue UNMAP commands from the virtual machine to a thinly provisioned virtual disk. When data is deleted inside the guest operating system it can send out UNMAP commands to the storage provider – in this case, a thinly provisioned virtual disk. After the UNMAP command was received by the ESXi host, the virtual disk can be shrunk per the actual used capacity inside the guest. Having the virtual disk shrunk, the common UNMAP procedure described in the previous section kicks in and UNMAP commands are sent out to the storage array.

In the following example, we use a Windows 2012 R2 virtual machine with a thin virtual disk. The ESXi host is running version 6.0 and the Virtual Hardware version is 11. We use a single K2 volume which is configured with compression only and no deduplication.

Before trying any UNMAP operations from a guest virtual machine, make sure that the `/VMFS3/EnableBlockDelete` option is enabled in ESXi.

For the sake of this example and before starting any data movements, we'll compare the reported used capacity by the ESXi host and the K2. Using the vSphere Web-Client, we can navigate to the Datastore section and see that the total used capacity by the Datastore is 9.81 GB:

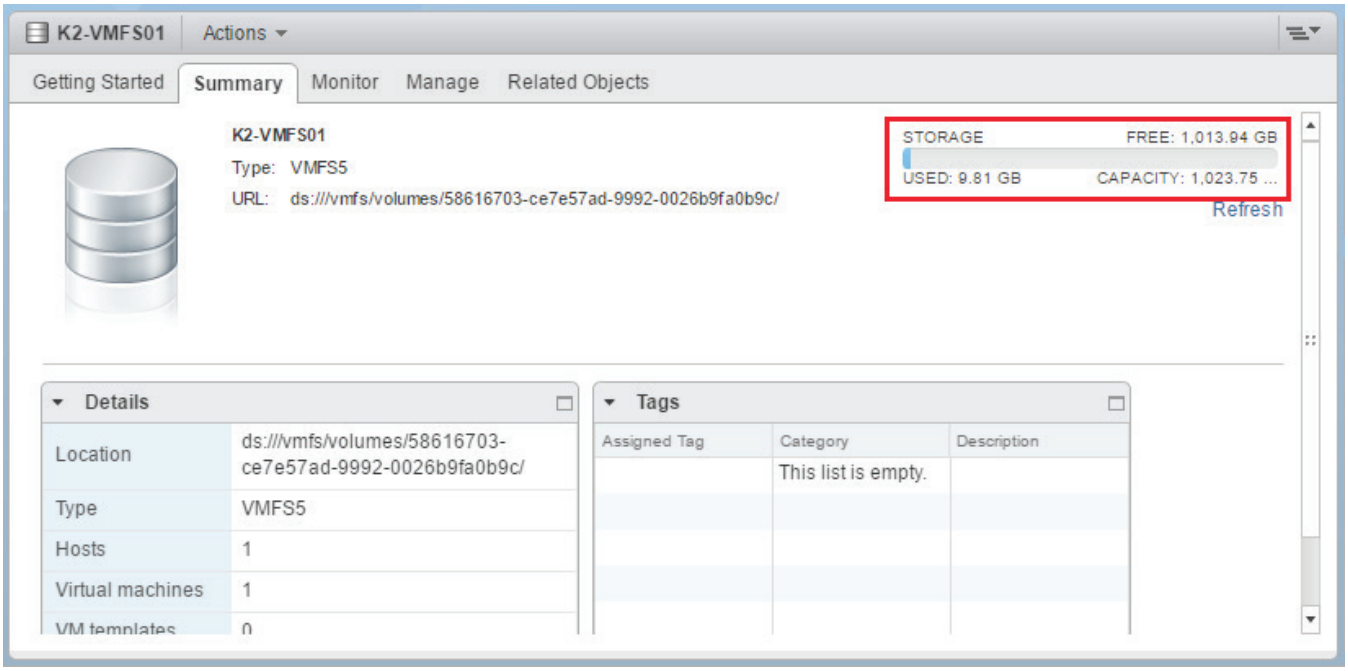


Figure 13: Datastore used capacity as reported in vSphere Web-Client

Looking at the same volume from the K2 GUI, we can see that the volume has 9.8 GB allocated capacity – meaning that the reported capacities by the ESXi and the K2 are aligned.

Figure 14: Volume used capacity as reported in the K2 GUI

We can also use the Kaminario vCenter Plug-In (VCP) to see the allocated capacity in the K2 of that volume:

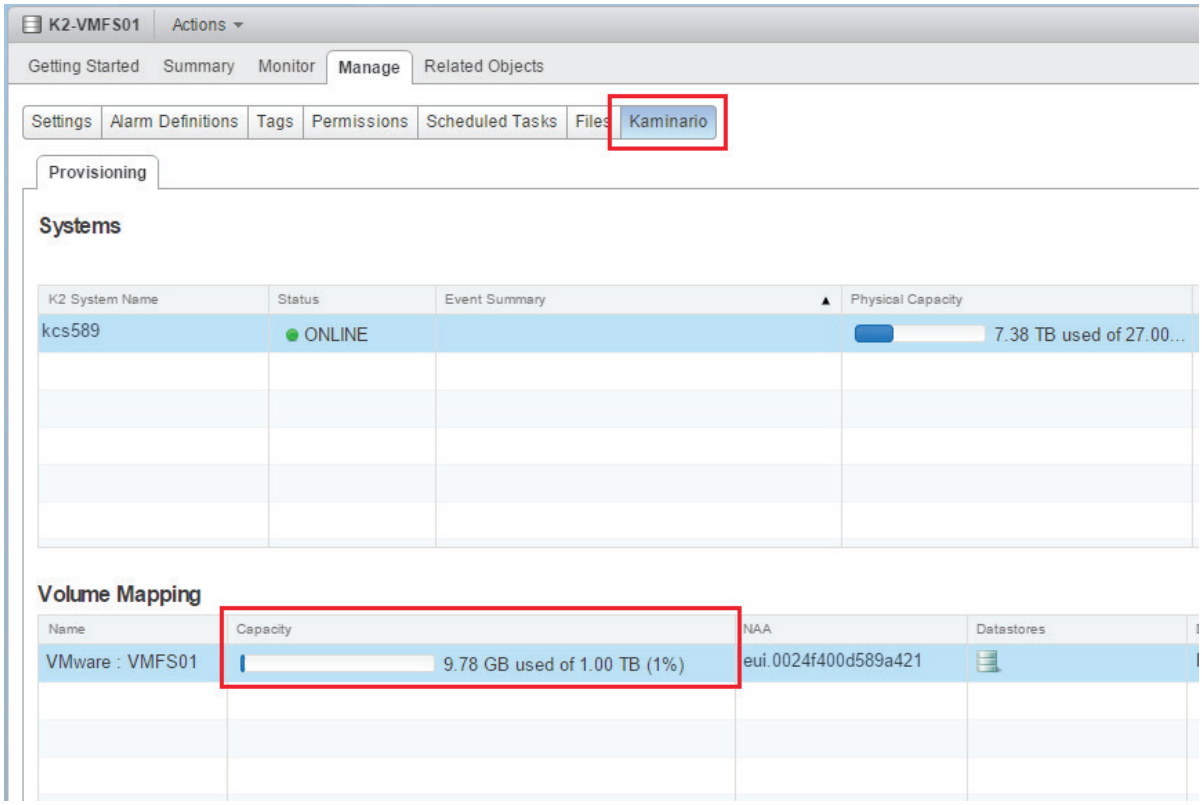


Figure 15: Volume used capacity as reported in the K2 vCenter Plug-In

From the 9.8 GB used on the Datastore, the Windows 2012 R2 virtual machine consumes 8.85 GB:

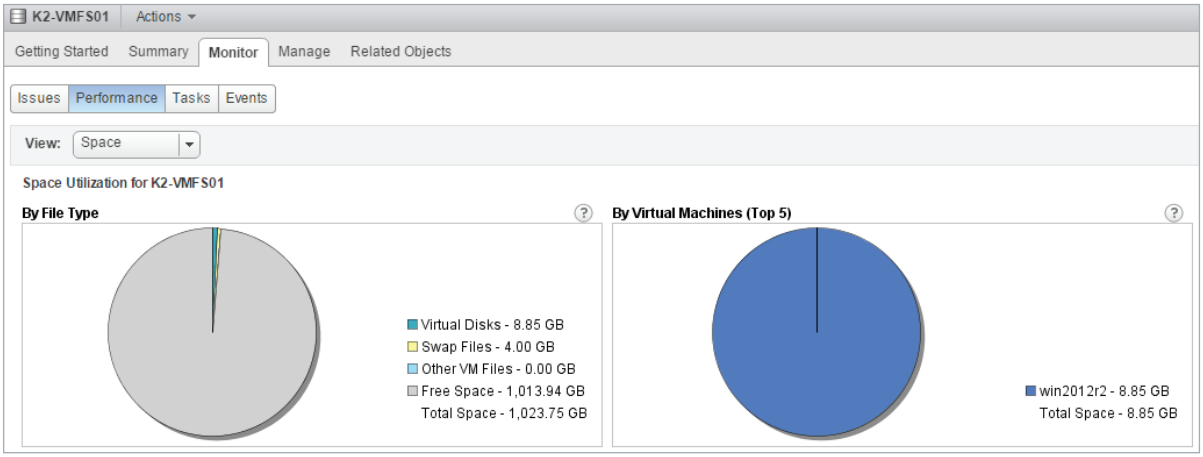


Figure 16: Volume and virtual machine used capacity - vSphere Web-Client

Next, we'll add some large files to the Windows 2012 R2 virtual machine to make it consume more capacity. In total, we add 12.7 GB to the virtual machine:

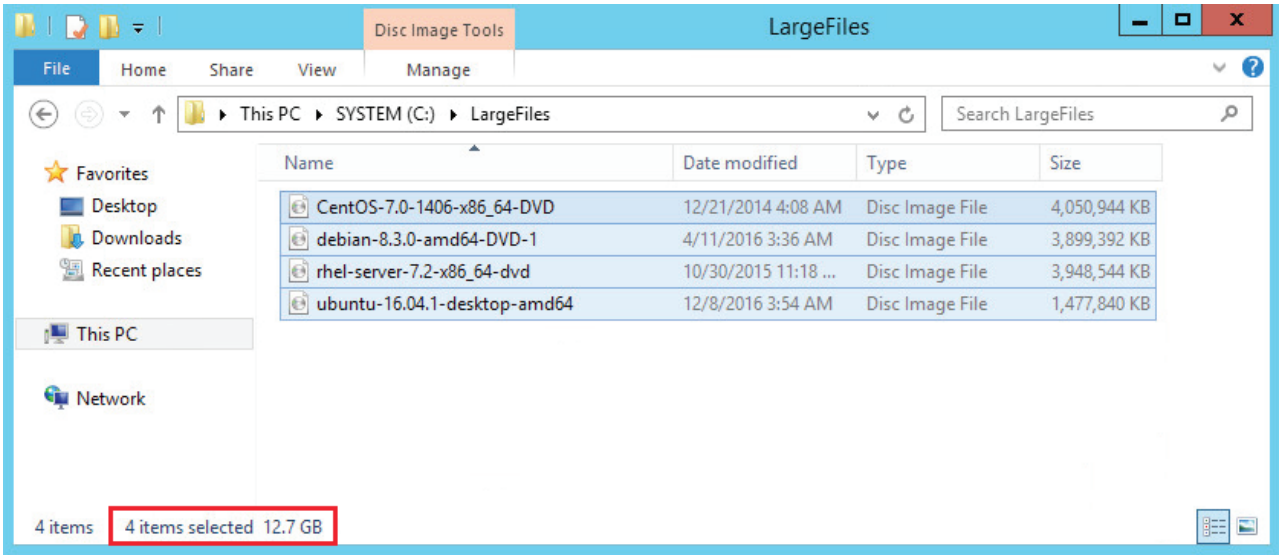


Figure 17: 12.7 GB of Large files added to virtual machine

As we can see in Figure 10 above, the virtual disk file has consumed 8.85 GB of capacity before adding any files. After we added 12.7 GB of large files to the virtual machine, we can see that the virtual disk has increased to 21.6 GB:

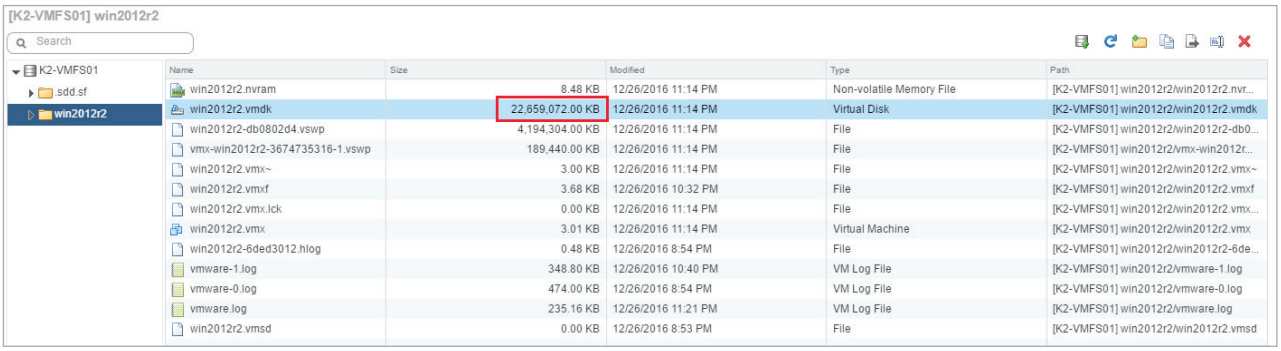


Figure 18: Windows 2012 R2 virtual machine files sizes after adding large files – vSphere

Since there are some more files on the Datastore than just the virtual disk file (like swap files, configuration files etc.), we can see that the K2 is reporting that 23.3 GB are allocated on that volume:dsssssssssssssssssssssssssssssssssssssssssss

Figure 19: Volume used capacity after adding large files – K2

Next, we will permanently remove the files from the Windows 2012 R2 virtual machine (Shift + Delete or a normal delete and emptying the Recycle Bin afterwards).

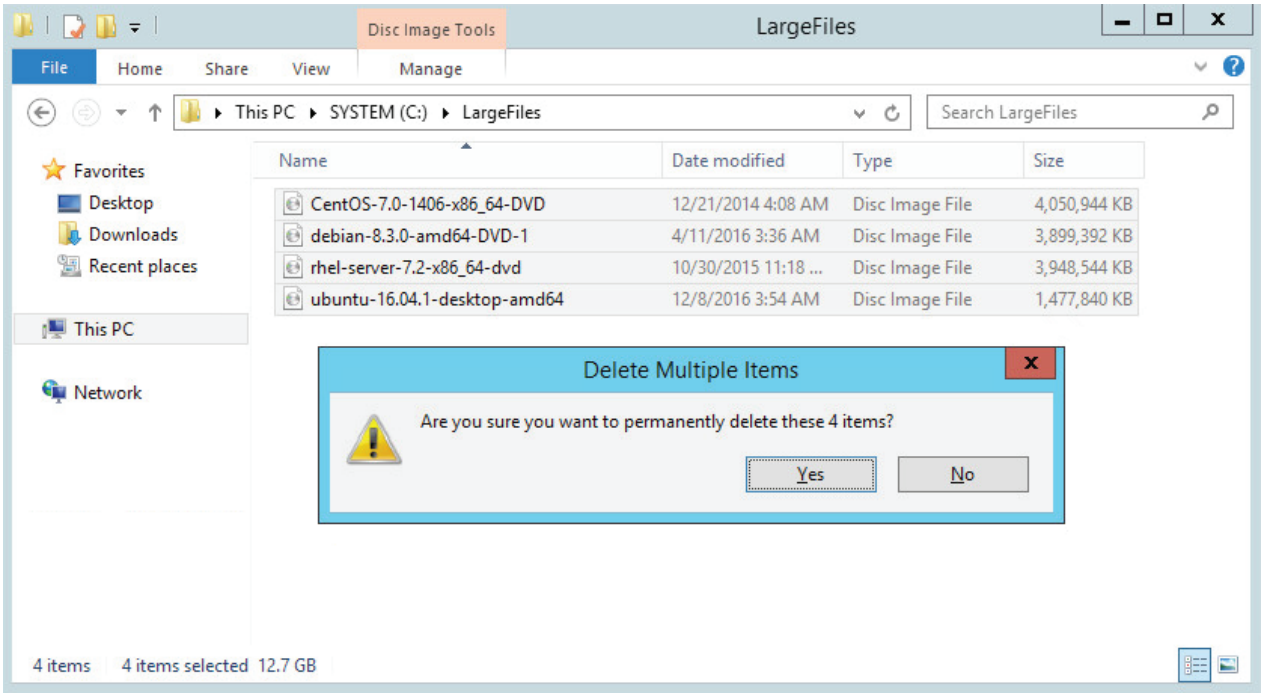


Figure 20: Permanently delete all large files from the virtual machine

Deleting the files is only the first step – we will not see any changes in used capacity or any UNMAP commands sent out. At the next step, we will have to run UNMAP inside the guest, or 'Optimize' the drive as it is called in Windows 2012 R2. The process is very quick and after selecting the 'Optimize' option the drive status is OK.

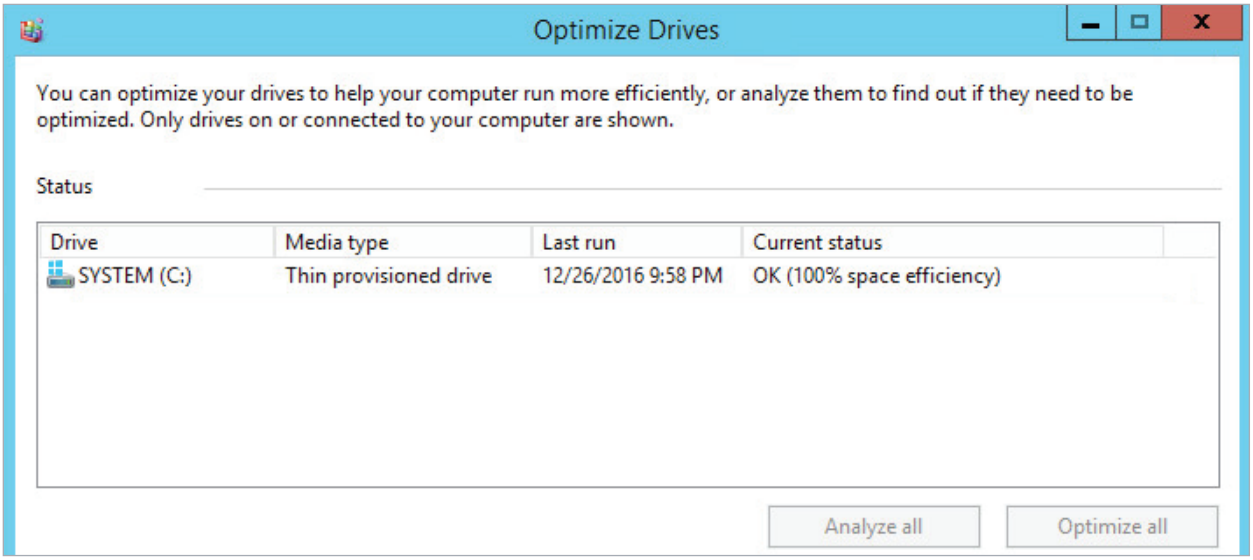


Figure 21: Optimizing drives in Windows 2012 R2

Once the guest has completed the operation, we can see that the virtual disk file has shrunk from 21.6 GB to 8.8 GB - just a little less than what it used to be before the UNMAP process:

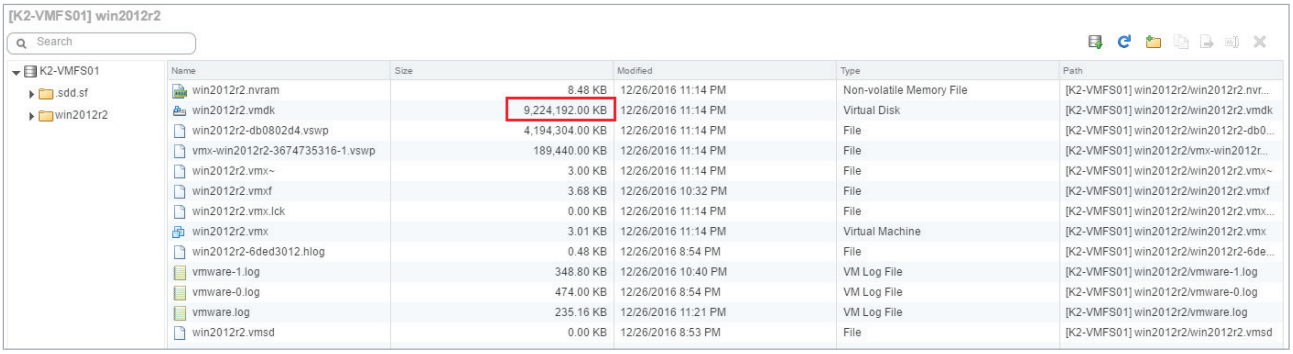


Figure 22: The VMDK file has shrunk after Windows Drive Optimization

After the virtual disk file has shrunk, the UNMAP commands are automatically sent out to the K2 by the ESXi host. This operation can be seen in the performance view in the K2 GUI, just as demonstrated in section 4.4.4 above.

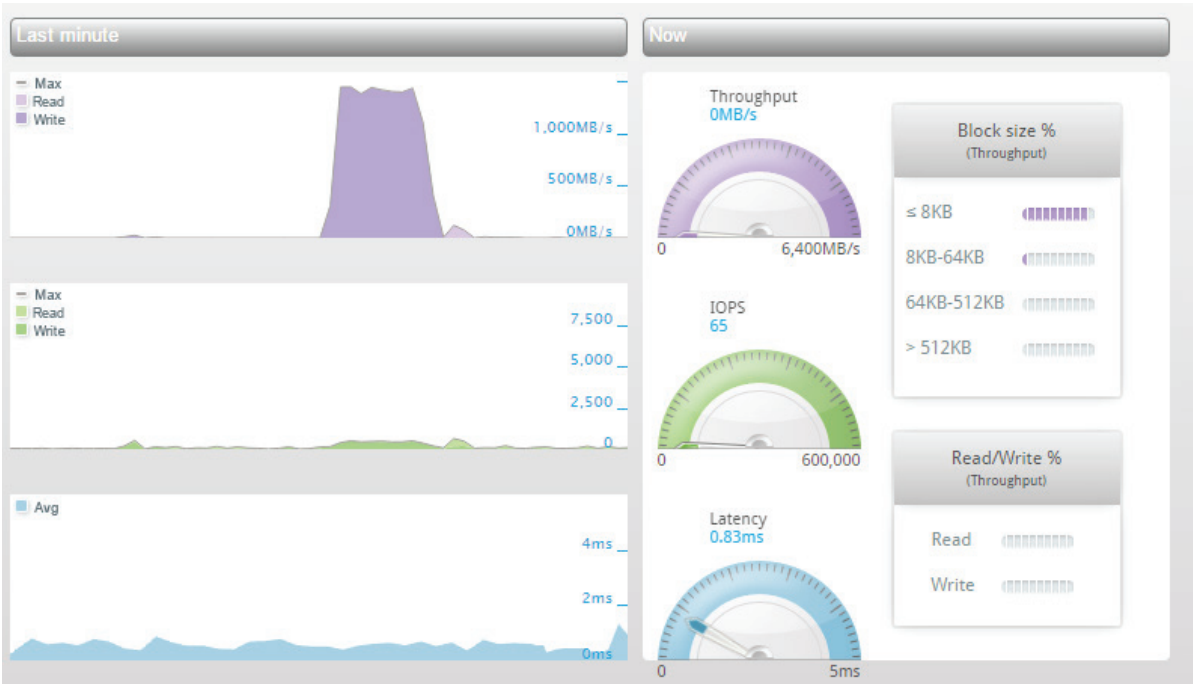


Figure 23: Kaminario K2 UNMAP workload

Last, we'll monitor the volume capacity in the K2 GUI to make sure that capacities are aligned:

Figure 24: K2 reported allocated capacity after UNMAP

As we can see in the figure above – the reported capacities by the K2 and the ESXi host are aligned.

Thin Provisioning Stun

As mentioned before in section 4.4.1 Zero Blocks (Write Same), one of the virtual disk types is a thin-provisioned virtual disk. When using thin-provisioned virtual disks, it is possible to over-provision capacity, meaning that having the total capacity of all virtual disks greater than the Datastore capacity.

A possible scenario in such a configuration is that virtual machines "see" that capacity is available while the underlying Datastore is running out of space. Before VMware added the Thin Provisioning Stun VAAI primitive, all virtual machines would pause in such scenario. After the new VAAI primitive was added, the behavior has changed so only virtual machines requesting more storage would pause and others would not.

Once the K2 volume and VMFS filesystem are extended, the paused virtual machines can be resumed.

The following example includes a single 100 GB Datastore which contains three virtual machines called TP-VM01, TP-VM02 and TP-VM03. Each of the virtual machines has a 40 GB thin-provisioned virtual disk for the operating system, and a second 40 GB thin-provisioned virtual disk for application data. The total provisioned capacity of all virtual disks is 240 GB, while the Datastore available capacity is only 100 GB.

Having Windows 2012 R2 installed in all virtual machines and nothing but that, the Datastore used capacity is about 27 GB as reported by the vCenter Server and the K2.

Figure 25: - vCenter used capacity for Datastore K2-VMFS02

Figure 26: - K2 allocated capacity for volume VMFS02

We use IOMeter to generate some data and workload on the additional 40 GB virtual disk in each of the virtual machines. In the first virtual machine (TP-VM01) on the Datastore, we run IOMeter with a 16KB block-size and 100% read workload profile. Then, in the other two virtual machines (TP-VM02 and TP-VM03), we run IOMeter with a 16KB block-size and 100% write workload profile.

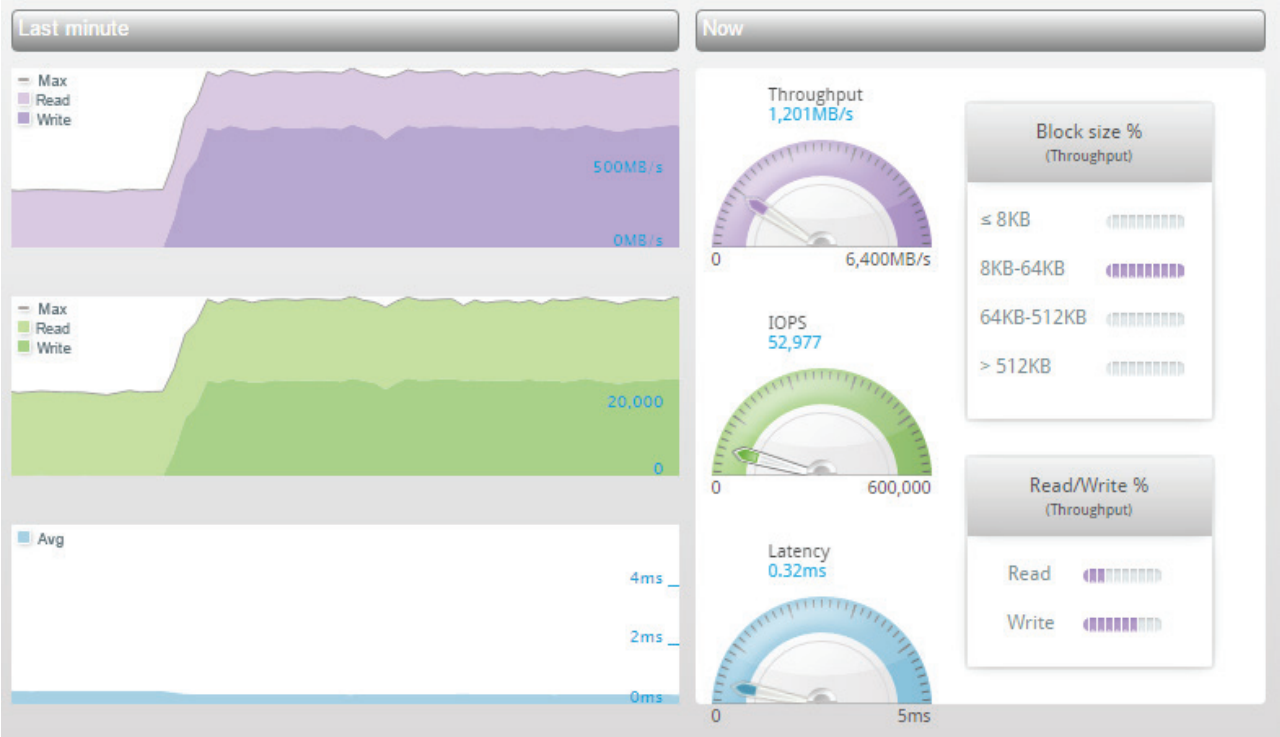


Figure 27: - Starting with a 100% read workload and then adding the 100% write workload

While the workloads are running, we monitor the volume's allocated capacity in the K2 and the Datastore's used capacity in the vCenter Server.

Figure 28: - Allocated capacity as reported on the K2 grows as workloads are running

Figure 29: - Used capacity as reported in the vCenter Server grows as workloads are running

Once we reached the point that the Datastore has ran out-of-space, we notice two things immediately:

- The K2 has stopped processing any write workload and now is running only read operations coming from VM-TP01:

Figure 30: K2 GUI shows only 100% read workload

- The virtual machines that were running 100% write workload (TP-VM02 and TP-VM03) have paused in the vCenter, waiting for extra capacity before continuing any operation:

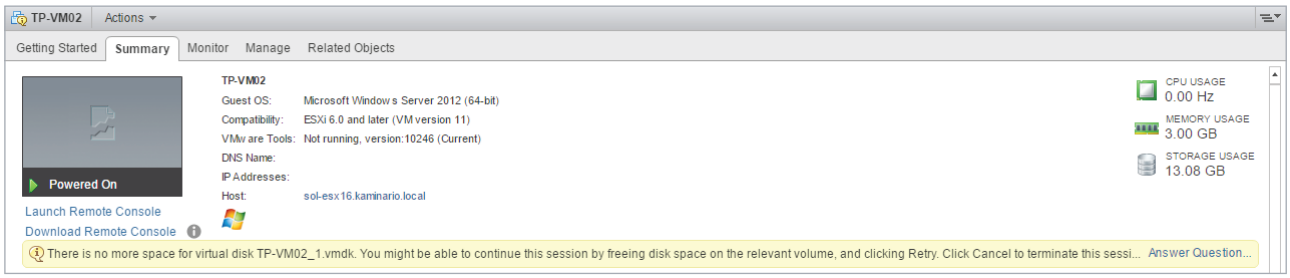


Figure 31: Virtual machine TP-VM02 has paused as the Datastore is out-of-space

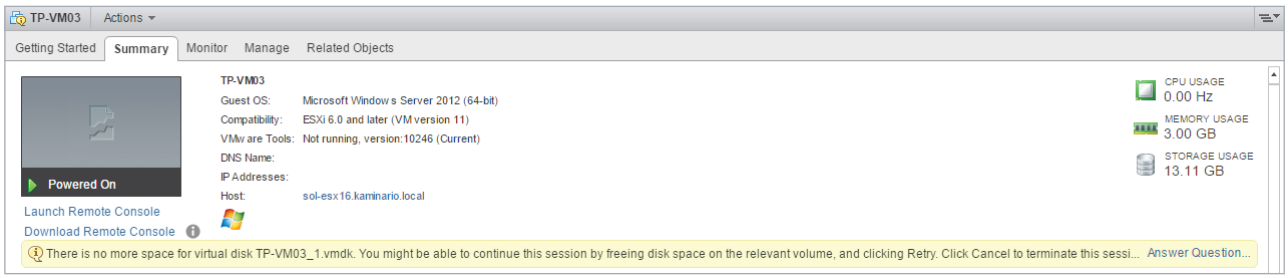


Figure 32: Virtual machine TP-VM03 has paused as the Datastore is out-of-space

As a next step, using the K2 GUI, we will increase the provisioned capacity of the volume to 300 GB. Now that the provisioned capacity in the K2 has grown, we can extend the VMFS filesystem. To do so, use the Rescan Storage option in vCenter to rescan the ESXi host for the new volume capacity, and then use the 'Increase Datastore Capacity' wizard in vCenter.

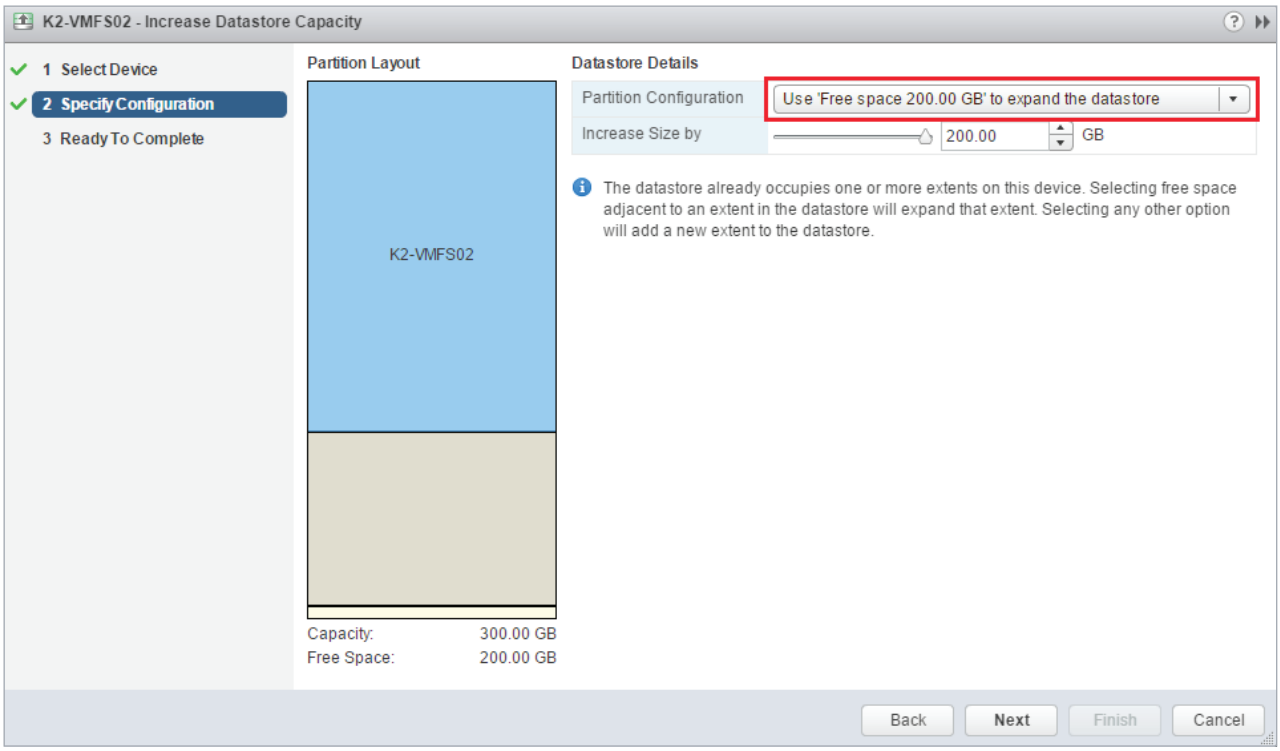


Figure 33: Increase Datastore Capacity wizard in vCenter

Now that there's some available capacity in the Datastore, we can answer the virtual machine question with the 'Retry' option to resume the operation.

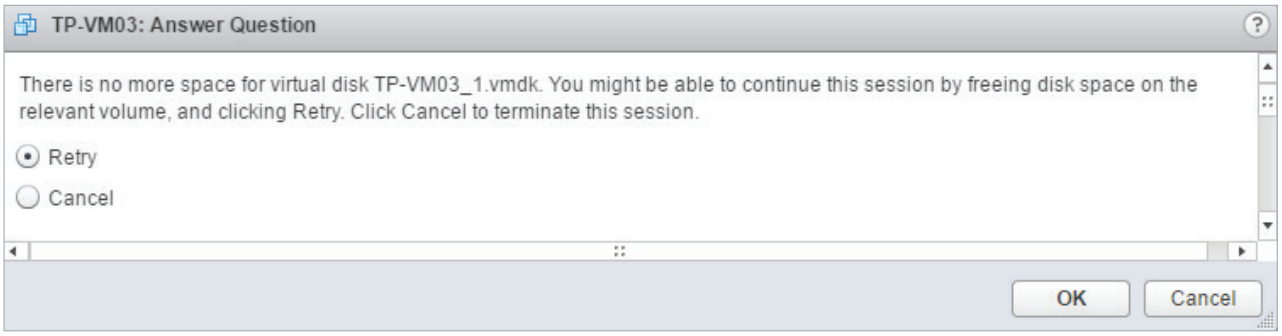


Figure 34: Select retry to resume the virtual machine operation

As expected, the virtual machines resume their operation and we can notice the write workload coming back in the K2:

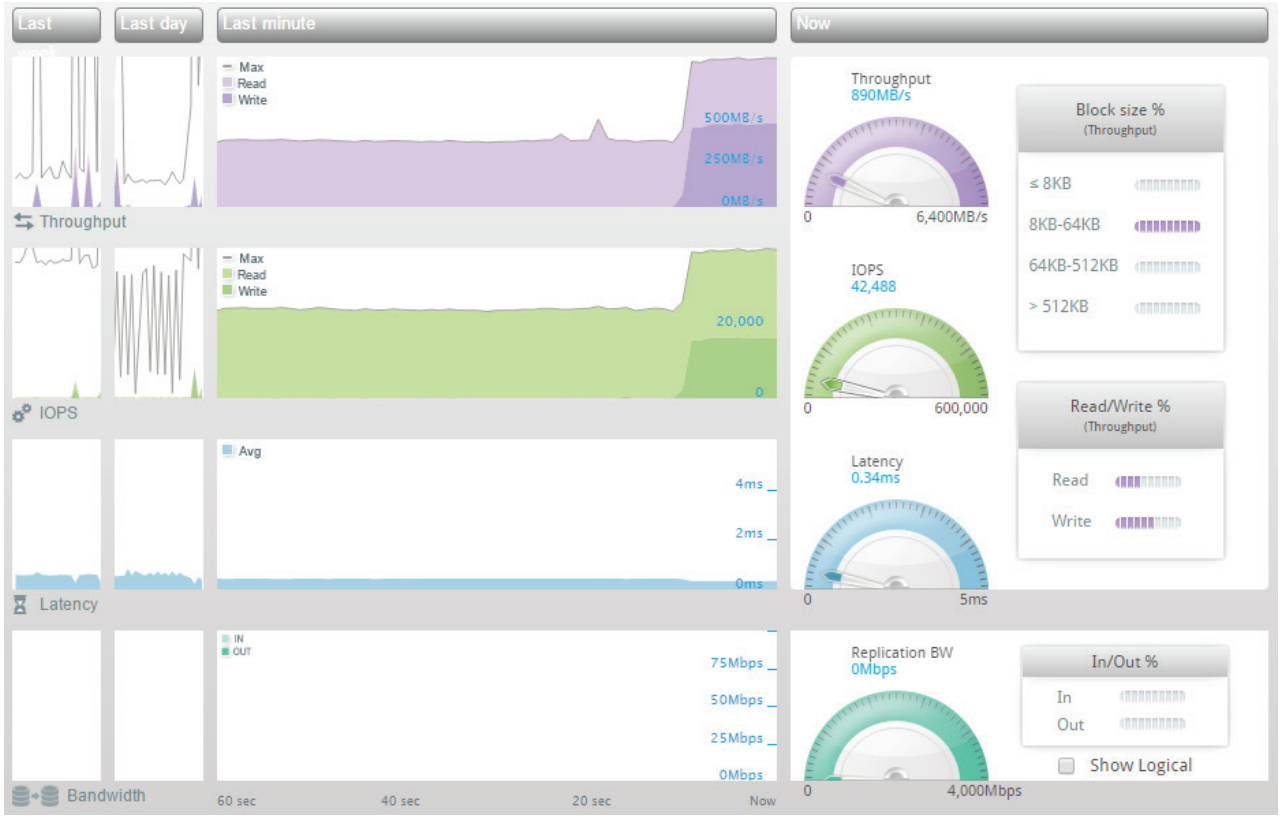


Figure 35: Write workload comes back as can be seen in the K2

Quota Exceeded Behavior

The Quota Exceeded Behavior primitive allows a storage array to inform an ESXi host that a Datastore reaches a specific threshold. In the Kaminario K2, the threshold is set at a volume group (VG) level by the VG Quota mechanism.

When not using this primitive, a Datastore can fill up and pause virtual machines that request more storage (as demonstrated in the previous section – 4.1.6 Thin Provisioning Stun).

If the Quota Exceeded Behavior primitive is in use and the VG threshold has reached, an event is sent out to vSphere informing it that the threshold was reached. Following such event, an administrator can use Storage vMotion to free up a Datastore, extend a Datastore, extend the VG quota or add a new Datastore.

In the next example, we'll set a quota on a K2 VG and write some data to the single volume in that VG. Once the quota has reached we'll see that a warning event is sent out to the vSphere, informing it that the quota was reached.

In the current state, the VMware VG has a single volume in it with 9.7 GB of capacity allocated. We'll set the VG Quota to 20 GB so we can quickly reach it for the purpose of this example.

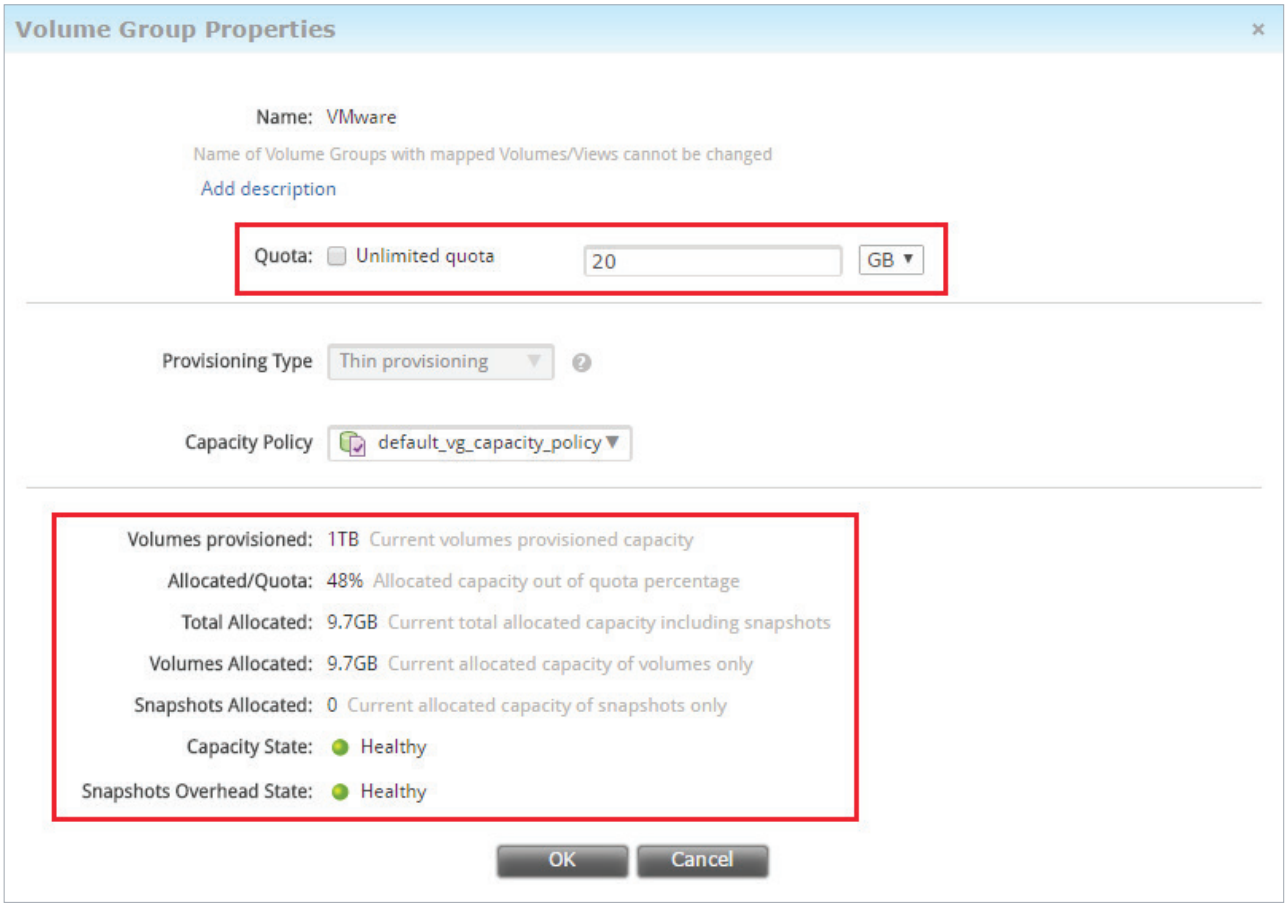


Figure 36: Configuring a VG Quota in K2

To reach the 20 GB Quota configured, we'll have to write at least additional 10 GB of data to that volume. To do so, we'll copy some large files to a virtual machine running on that Datastore.

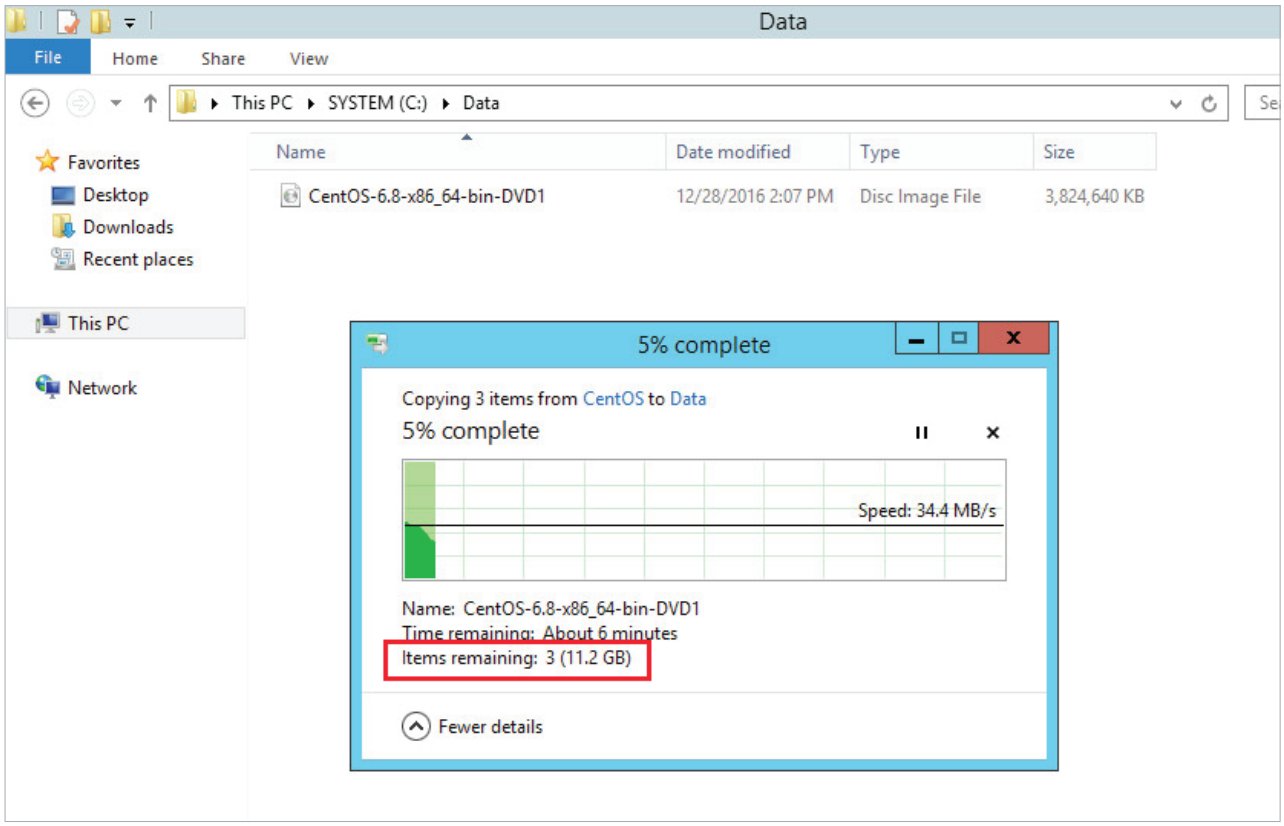


Figure 37: Writing data to reach quota limit

While we write some new data to the Datastore, we monitor the K2 VG quota usage:

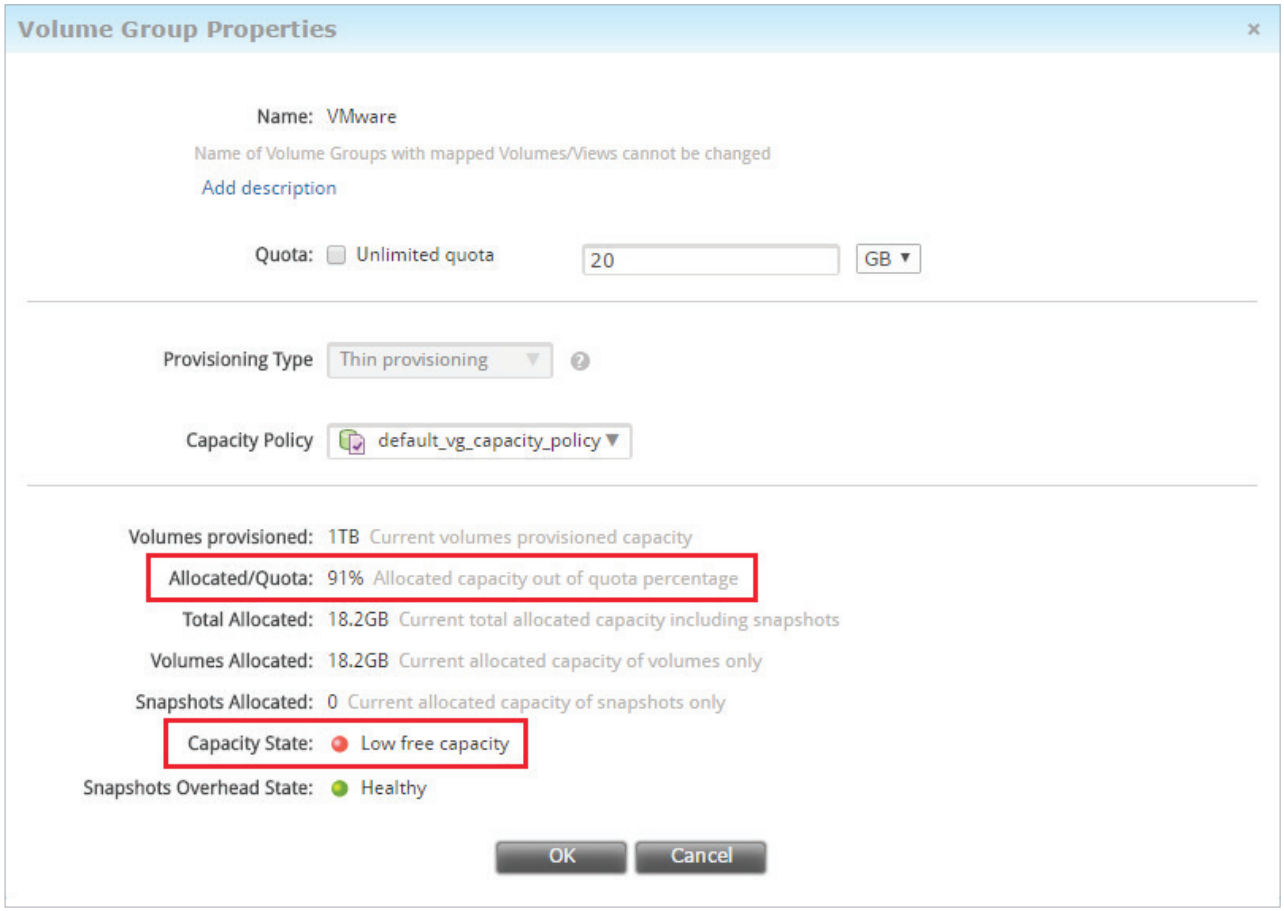


Figure 38: K2 Volume Group Low free capacity

As the quota limit was reached, the K2 informs vSphere about the reached quota which in turn triggers an alarm in vCenter:

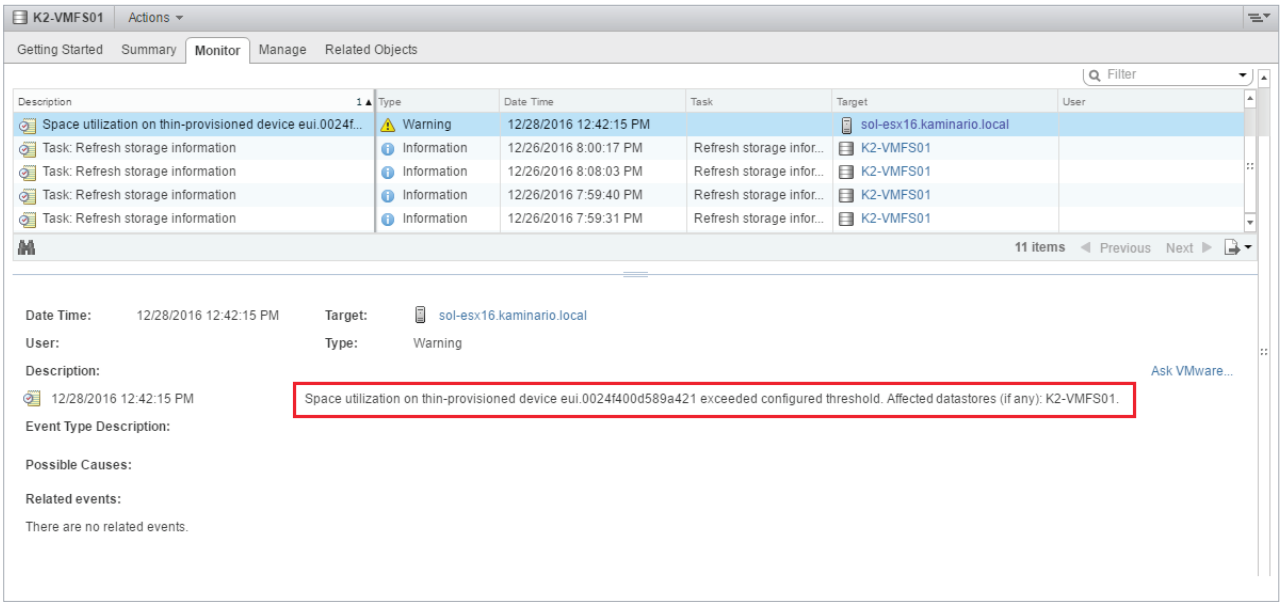


Figure 39: Space utilization event appears in vCenter because of a reached quota

At this step, an administrator can increase the VG quota, extend a volume, add a new volume or move the virtual machine to a different Datastore according to the policy and needs implemented in the virtual environment.

Achieving High Performance in ESXi - Performance Comparison

As explained in previous chapters, several ESXi parameters can significantly improve virtualized applications' performance. This section of this Reference Architecture compares performance of mixed workload between multiple different configurations. At first, we ran a workload with no best practices configured at all and recorded the performance results. At the next step, we configured the best practices for ESXi Host Optimizations without setting the HBA recommended settings, and again, we recorded the performance results. At the last step, we configured the HBA (Qlogic QLE2560) recommended settings and recorded the results.

The purpose of this section is to study the effect of changing settings on an existing application. For that reason, we decided to use a single ESXi host for our testing while we can achieve significant better performance by splitting the workload on several ESX hosts.

The workload is the same through all the steps of the comparison and based on IOMeter. We used four virtual machines for benchmarking, each with the following spec:

- 2 vCPUs
- 4GB RAM
- Windows Server 2012 R2 64 bit
- 40GB Operating System Virtual Disk, Thick Provisioned Eager Zeroed
- 15GB IOMeter Virtual Disk, Thick Provisioned Eager Zeroed

The mixed workload we use in this comparison is composed from the following, each virtual machine with its own specific workload:

Virtual Machine	Block Size	R/W	Outstanding IOs	Sequential/Random
Mixed01	4KB	40/60	64	50/50
Mixed02	8KB	75/25	64	80/20
Mixed03	16KB	50/50	64	60/40
Mixed04	32KB	80/20	64	20/80

All four virtual machines are running on a single ESXi host with 24 logical processors and 96GB of RAM. During the comparison, there were no changes to resources allocation in vSphere or any Storage I/O Control (SIOC) changes and default settings were in use. In each of the three steps, we ran the workloads specified in the table above for a duration of 10 minutes.

In each of the three steps of the comparison we examined the results from three points of view:

- Kaminario K2 - The K2 performance was exported from the K2 Analysis screen to a CSV file to compare the performance between the different steps and for the different calculations of averages and maximums.

- vSphere ESXi – We used esxtop to monitor the response time using three different metrics (From VMware's KB1008205):
  - DAVG/cmd - The average response time in milliseconds per command being sent to the device.
  - KAVG/cmd - The amount of time the command spends in the VMkernel.
  - GAVG/cmd - The response time as it is perceived by the guest operating system. This number is calculated with the formula: DAVG + KAVG = GAVG
- Guest VMs IOMeter - The IOMeter results are the averages produced by IOMeter for each step by every one of the four virtual machines for the workload duration.

Default Settings

In this part of the comparison our test environment was configured as follows:

- Fixed Path Selection Policy (PSP)
  - Number of commands to switch path = N/A
- Disk.SchedQuantum = 8
- Number of Outstanding IOs With Competing Worlds = 32
- Qlogic HBA with default settings (QD=64, ZIO=1, IDT=100)

From looking at the K2 Analysis screen, we can see performance was very consistent in all metrics – Throughput, IOPS and Latnecy, however, performance is not satisfting as we know the K2 can withstand higher loads.

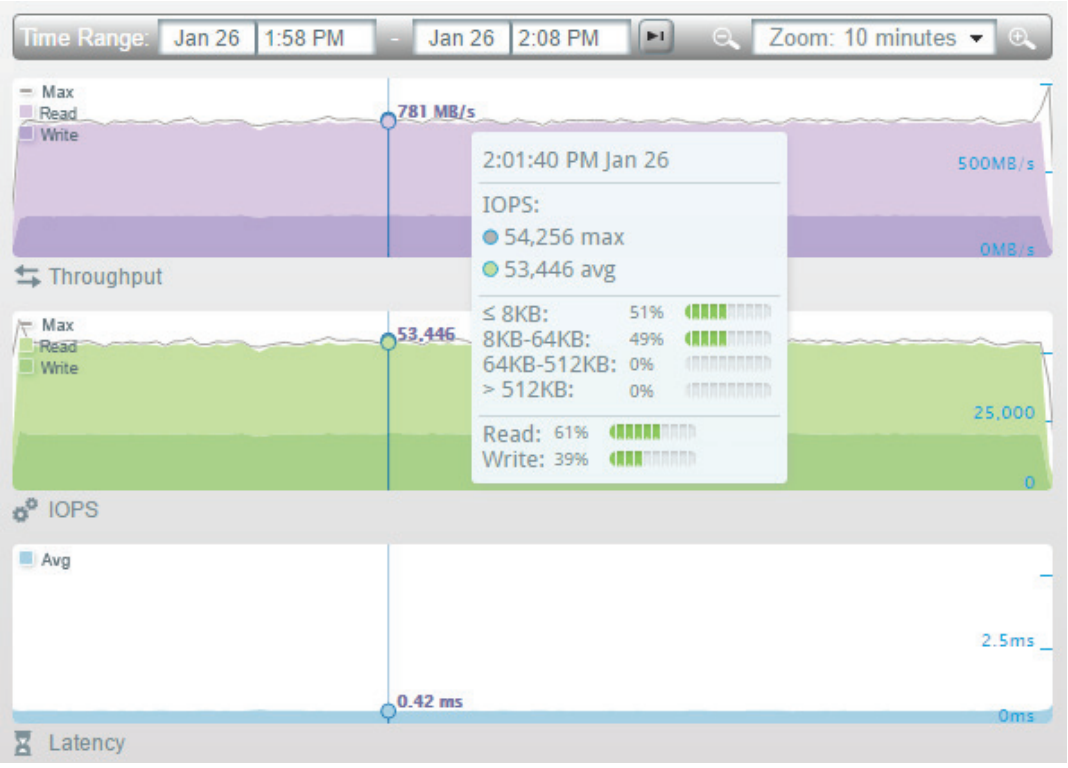


Figure 40: K2 Analysis - Default Settings

The K2 performance results are detailed in the following table. The latency specified in the table includes the estimated:

		Throughput	IOPS	K2 Latency
Default Settings	Average	785 MB/s	53,743	0.42ms
	Maximum	851 MB/s	61,760	0.46ms

As can be seen in the following screenshot of esxtop, only a single path is active due to the PSP set to Fixed. Also, the device queue length is set to 32 as we are running with the default setting of Number of Outstanding IOs With Competing Worlds. Important metrics to pay attention to are the DAVG/cmd and the KAVG/cmd. The first one, DAVG/cmd, is aligned with the K2 reported latency (including the SAN added latency) and is indicating a value of 0.63ms response time from the device (the K2 volume). On the other hand, the KAVG/cmd metric went up above 4ms, making the guest response time (GAVG/cmd) at least 6x slower and reach a value of 4.77ms.

DQLEN	WQLEN	ACTV	QUED	%USD	LOAD	CMDS/s	READS/s	WRITES/s	MBREAD/s	MBWRTN/s	DAVG/cmd	KAVG/cmd	GAVG/cmd
32	-	-	-	-	-	50484.36	31051.07	19433.29	510.50	229.02	0.63	4.14	4.77
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
32	-	-	-	-	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 41: esxtop - Default Settings

The next table summarizes the results seen in every one of the virtual machines running IOMeter. Considering GAVG/cmd of 4.77ms and the virtual guest's queues, we can see that the virtual machines latency was relatively high and went up to 9.7ms.

Workload Details					Results (Averages)		
Virtual Machine	Block Size	R/W	Outstanding IOs	Sequential/ Random	IOPS	Throughput	Latency
Mixed01	4KB	40/60	64	50/50	13,235	51.7 MB/s	9.6ms
Mixed02	8KB	75/25	64	80/20	13,176	102.94 MB/s	9.7ms
Mixed03	16KB	50/50	64	60/40	13,159	215.61MB/s	9.7ms
Mixed04	32KB	80/20	64	20/80	13,174	411.7 MB/s	9.7ms

An interesting observation is that IOPS are equally distributed between all four virtual machines, although workloads are different (Block size, R/W ratio and Sequential/Random ratio). The reason for this behavior is that the bottleneck is not at the K2, but at the initiator (the ESXi host). Since the ESXi has reached the performance limit, it tries to accomplish its default goal of fairness, so the workload is equally distributed between the virtual machines to gain fairness.

ESXi Host Optimizations

In this part of the comparison our test environment is configured as follows (changes in bold):

- **Round-Robin** Path Selection Policy (PSP)
  - Number of commands to switch path = **2**
- Disk.SchedQuantum = **64**
- Number of Outstanding IOs With Competing Worlds = **256**
- Qlogic HBA with default settings (QD=64, ZIO=1, IDT=100)

At this step we expect performance to increase drastically. The PSP change to Round-Robin has a great contribution to performance improvments as it makes the ESXi host better utilize all available paths and ports in the K2. Setting the Datastore with a queue length of 256 (using the Number of Outstanding IOs With Competing Worlds parameter) should improve performance as well, but with the HBA configured with the default queue length of 64 the full potential of performance improvement is limited. Another important change is the number of commands to switch paths on. Changing this parameter to a small number makes better utilization of the available K2 ports.

Looking again at the K2 Analysis screen, like before, we see performance is consistent in all metrics – Throughput, IOPS and Latnecy. This time, as expected, we get much better results – Throughput and IOPS got doubled and latency is less than what it used to be.

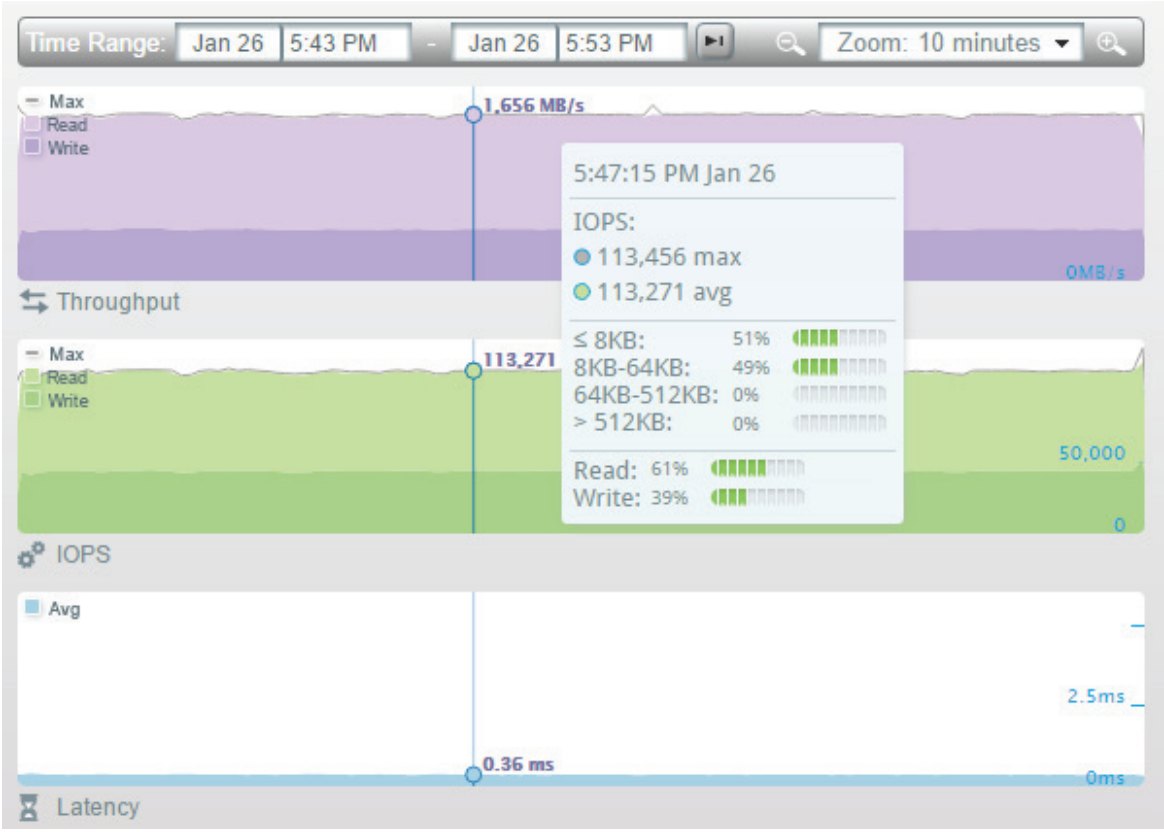


Figure 42: K2 Analysis – ESXi Host Optimization

The K2 performance results are detailed for the two configurations in the following table :

		Throughput	IOPS	K2 Latency
Default Settings	Average	785 MB/s	53,743	0.42ms
	Maximum	851 MB/s	61,760	0.46ms
ESXi Host Optimization	Average	1,654 MB/s	113,121	0.36ms
	Maximum	1,753 MB/s	128,788	0.4ms

This time, when we look at the output of esxtop, we see all paths are active due to the PSP set to Round-Robin, and that the load is equally distributed between all paths. Although the Number of Outstanding IOs With Competing Worlds parameter was configured to 256, the HBA default queue depth of 64 is limiting the device queue to 64.

Looking at the key metrics DAVG/cmd and KAVG/cmd, we can see that the K2 response time is improved and consistent between all paths. Even more interesting is the improvement for the KAVG/cmd - from above 4ms the KAVG/cmd was reduced to about 1.5ms, a 63% improvement yet not near zero as expected and should be. This improvement in the KAVG/cmd metric is a result of releasing some of the bottleneck from the ESXi VMKernel by enlarging the device queue to 64 and changing the PSP to Round-Robin.

DQLEN	WQLEN	ACTV	QUED	%USD	LOAD	CMDS/s	READS/s	WRITES/s	MBREAD/s	MBWRTN/s	DAVG/cmd	KAVG/cmd	GAVG/cmd
64	-	-	-	-	-	14247.01	8767.94	5479.08	145.06	66.00	0.56	1.49	2.05
64	-	-	-	-	-	14396.08	8698.68	5697.40	145.00	67.21	0.55	1.49	2.04
64	-	-	-	-	-	14421.90	8809.02	5611.71	144.61	66.56	0.55	1.49	2.04
64	-	-	-	-	-	14149.59	8715.12	5434.48	144.54	64.25	0.56	1.49	2.05
64	-	-	-	-	-	14164.85	8645.86	5518.99	143.51	64.90	0.56	1.49	2.05
64	-	-	-	-	-	14362.04	8825.45	5536.59	143.35	66.13	0.56	1.49	2.05
64	-	-	-	-	-	14281.05	8730.37	5550.68	142.85	65.19	0.56	1.49	2.05
64	-	-	-	-	-	14384.34	8732.72	5651.62	140.68	66.61	0.55	1.49	2.05

Figure 43: esxtop - ESXi Host Optimization

The IOMeter results from this step reveals a similar behavior to what we have seen before in the Default Settings run. IOPS are equally distributed between the virtual machines although each virtual machine performs a different workload. Again, such a behavior can suggest that the ESXi has reached the performance limit and distributes workload equally to reach the goal of fairness. Note that the latency as reported by IOMeter was reduced dramatically.

Workload Details					Results (Averages)		
Virtual Machine	Block Size	R/W	Outstanding IOs	Sequential/ Random	IOPS	Throughput	Latency
Mixed01	4KB	40/60	64	50/50	28,240	110.3 MB/s	4.5ms
Mixed02	8KB	75/25	64	80/20	28,145	219 MB/s	4.5ms
Mixed03	16KB	50/50	64	60/40	28,124	439 MB/s	4.5ms
Mixed04	32KB	80/20	64	20/80	28,082	877 MB/s	4.5ms

ESXi Host Optimization and HBA Recommended Settings

In this part of the comparison our test environment was configured as follows (new changes in bold, previous changes which are still applied in underline):

- **Round-Robin** Path Selection Policy (PSP)
  - Number of commands to switch path = 2
- Disk.SchedQuantum = 64
- Number of Outstanding IOs With Competing Worlds = 256
- Qlogic HBA with default settings (QD=400, ZIO=6, IDT=1)

We expect to see even better results, especially due to releasing the VMkernel bottleneck (KAVG/cmd) by enlarging the HBA queue depth. As a result of a larger queue in the HBA, we can expect the DAVG/cmd to increase, as commands spends more time at the HBA queue. Nevertheless, the virtual machine response time as reported in IOMeter should be improved since the K2 latency is still sub-1ms and there are less delays along the data path.

The K2 Analysis screen reveals another major improvement in performance, and once again, we see performance is consistent in all metrics – Throughput, IOPS and Latnecy.

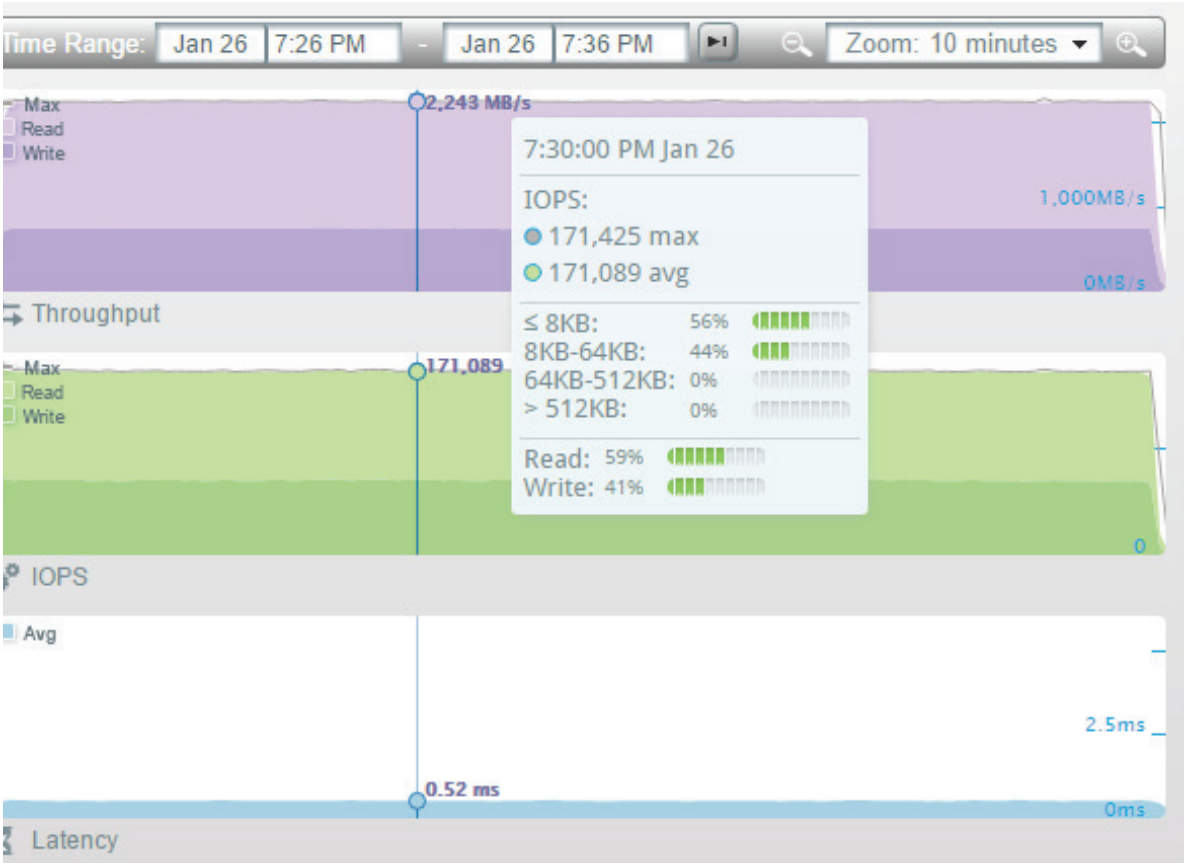


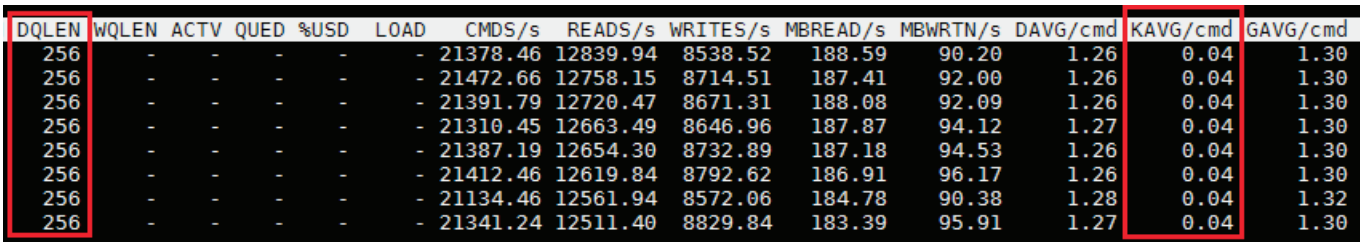
Figure 44: K2 Analysis – ESXi Host Optimization and HBA Recommended Settings

The K2 performance results are detailed for each configuration in the following table:

		Throughput	IOPS	K2 Latency
Default Settings	Average	785 MB/s	53,743	0.42ms
	Maximum	851 MB/s	61,760	0.46ms
ESXi Host Optimization	Average	1,654 MB/s	113,121	0.36ms
	Maximum	1,753 MB/s	128,788	0.4ms
With HBA Setting	Average	2,246 MB/s	171,780	0.52ms
	Maximum	2,277 MB/s	179,982	0.55ms

Since the only change from the previous run was the HBA settings, we can focus on the Number of Outstanding IOs With Competing Worlds parameter which is configured to 256. Now that the HBA queue depth is set to 400, it is not limiting the device queue to 64 as it did before, and now we are utilizing the whole device queue.

As can be seen in the following screenshot of esxtop, the key metrics DAVG/cmd and KAVG/cmd has drastically changed. The KAVG/cmd has dropped to nearly zero – as expected and as it should be. This drop indicates that the bottleneck of the ESXi VMKernel was removed. On the other hand, we see an increment in the DAVG/cmd. Although the K2 reports an average latency of 0.52ms, esxtop reports a latency of above 1ms for the DAVG/cmd metric since it includes the amount of time spent in the HBA queue and the round-trip time.



DQLEN	WQLEN	ACTV	QUED	%USD	LOAD	CMDS/s	READS/s	WRITES/s	MBREAD/s	MBWRTN/s	DAVG/cmd	KAVG/cmd	GAVG/cmd
256	-	-	-	-	-	21378.46	12839.94	8538.52	188.59	90.20	1.26	0.04	1.30
256	-	-	-	-	-	21472.66	12758.15	8714.51	187.41	92.00	1.26	0.04	1.30
256	-	-	-	-	-	21391.79	12720.47	8671.31	188.08	92.09	1.26	0.04	1.30
256	-	-	-	-	-	21310.45	12663.49	8646.96	187.87	94.12	1.27	0.04	1.30
256	-	-	-	-	-	21387.19	12654.30	8732.89	187.18	94.53	1.26	0.04	1.30
256	-	-	-	-	-	21412.46	12619.84	8792.62	186.91	96.17	1.26	0.04	1.30
256	-	-	-	-	-	21134.46	12561.94	8572.06	184.78	90.38	1.28	0.04	1.32
256	-	-	-	-	-	21341.24	12511.40	8829.84	183.39	95.91	1.27	0.04	1.30

Figure 45: esxtop - ESXi Host Optimization and HBA Recommended Settings

At this step, the IOMeter results are unlike what we have seen before. The IOPS are not distributed equally as they did before and latency is different for each workload. This change in performance results is an outcome of removing the bottleneck from the ESXi host, and by that, getting the performance the K2 can serve for each type of the four different workloads. The K2, and especially the Dual K-Block system used in this comparison, can produce better performance, but we should keep in mind that this performance are the results of a single initiator only.

Workload Details					Results (Averages)		
Virtual Machine	Block Size	R/W	Outstanding IOs	Sequential/ Random	IOPS	Throughput	Latency
Mixed01	4KB	40/60	64	50/50	51,000	199 MB/s	2.5ms
Mixed02	8KB	75/25	64	80/20	44,286	346 MB/s	2.8ms
Mixed03	16KB	50/50	64	60/40	42,985	671 MB/s	2.9ms
Mixed04	32KB	80/20	64	20/80	32,796	1025 MB/s	3.9ms

Performance Results Comparison

In this part, we examine the results from two points of view – the K2 and the guest virtual machine.

K2 Results

Starting with the K2 results, we see improvement in IOPS and Throughput through all steps of the comparison. We chose to show both average and maximum results so it is clear to see that the average results are very close to the maximum, meaning that performance variance is relatively low and performance is consistent.

The next two charts of IOPS and Throughput reveal that using the recommended settings both for the ESXi Host and for the HBA can improve performance by nearly 3x. Starting with the Default Settings where the maximum IOPS reached 61,760 and maximum throughput was 851MB/s, we reached 179,982 IOPS and 2,277MB/s of throughput accordingly.

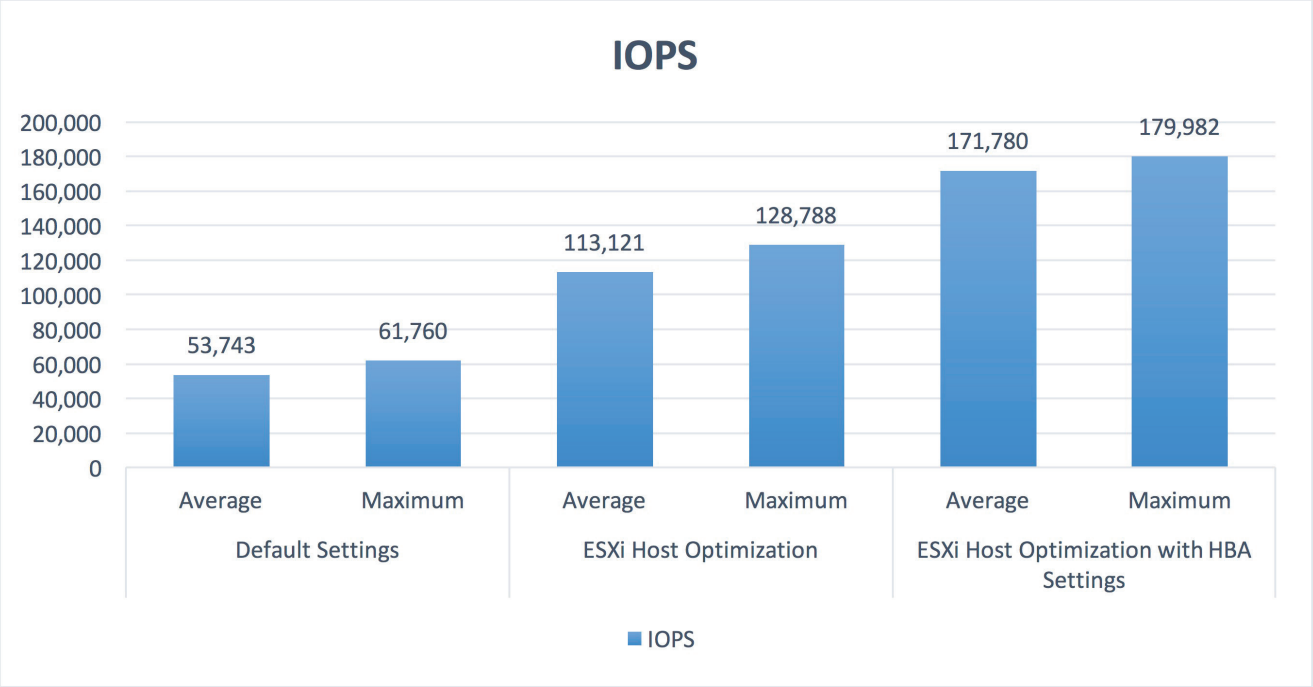


Figure 46: K2 IOPS Comparison

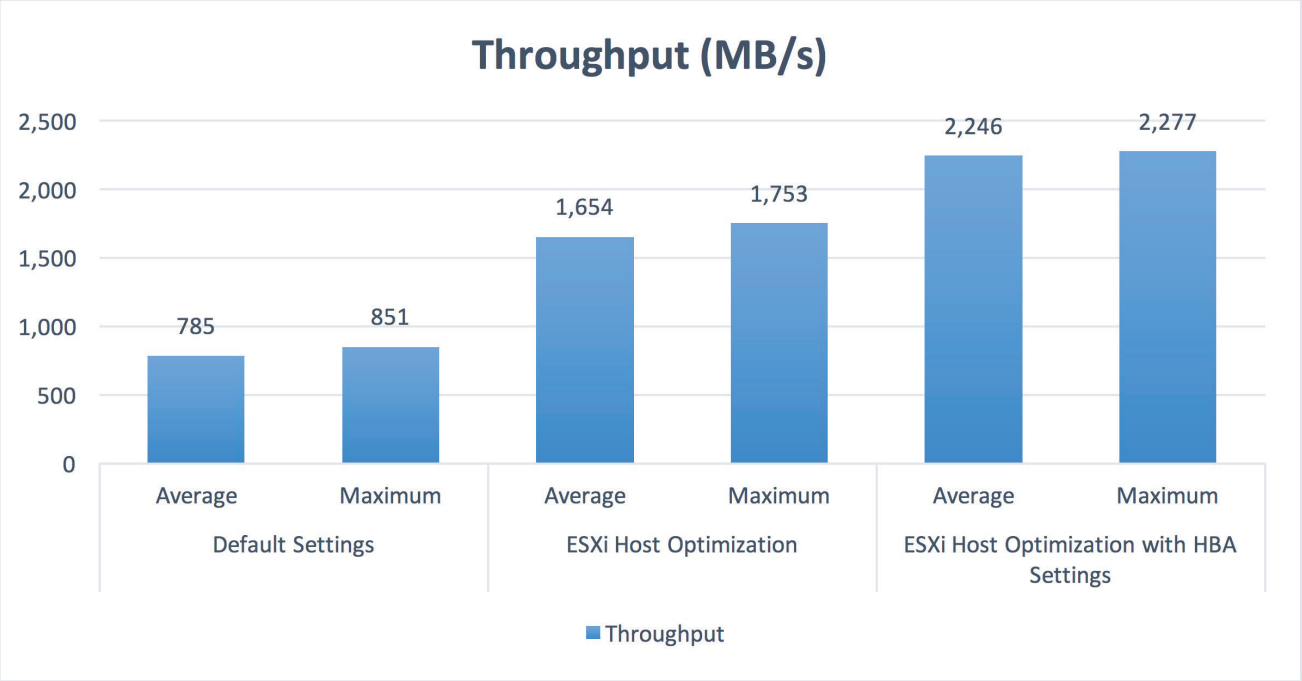


Figure 47: K2 Throughput Comparison

Relating to the latency results we can see that all results in all configurations were sub-1ms.

In the Default Settings, where we used a single path and no other optimizations, results still were sub-1ms as expected. Keep in mind that at this point, the bottleneck was at the ESXi host, so the K2 could easily satisfy the required performance with sub-1ms latency as it wasn't under full-load.

At the next step where we applied the ESXi Host Optimization settings, we see an improvement in latency as all paths are active and in use. However, we did mention that at this step there's still a bottleneck at the ESXi host as the KAVG/cmd was around 1.5ms and not near-zero as expected. So, like the Default Settings run, the K2 was not under full-load and could supply great latency in ease.

The last step is where results are getting interesting, we would expect that at this configuration latency would be the best. But keep in mind that with this configuration the bottleneck was removed from the ESXi, and the K2 was the one to set the limits, hence it is under the full-load that the single initiator can produce. Under such circumstances, the K2 serves each workload with the best results it can, considering all virtual machines are stored on the same single volume and are hosted in a single ESXi host. Yet, worth to mention, latency was still sub-1ms with an average of 0.62ms.

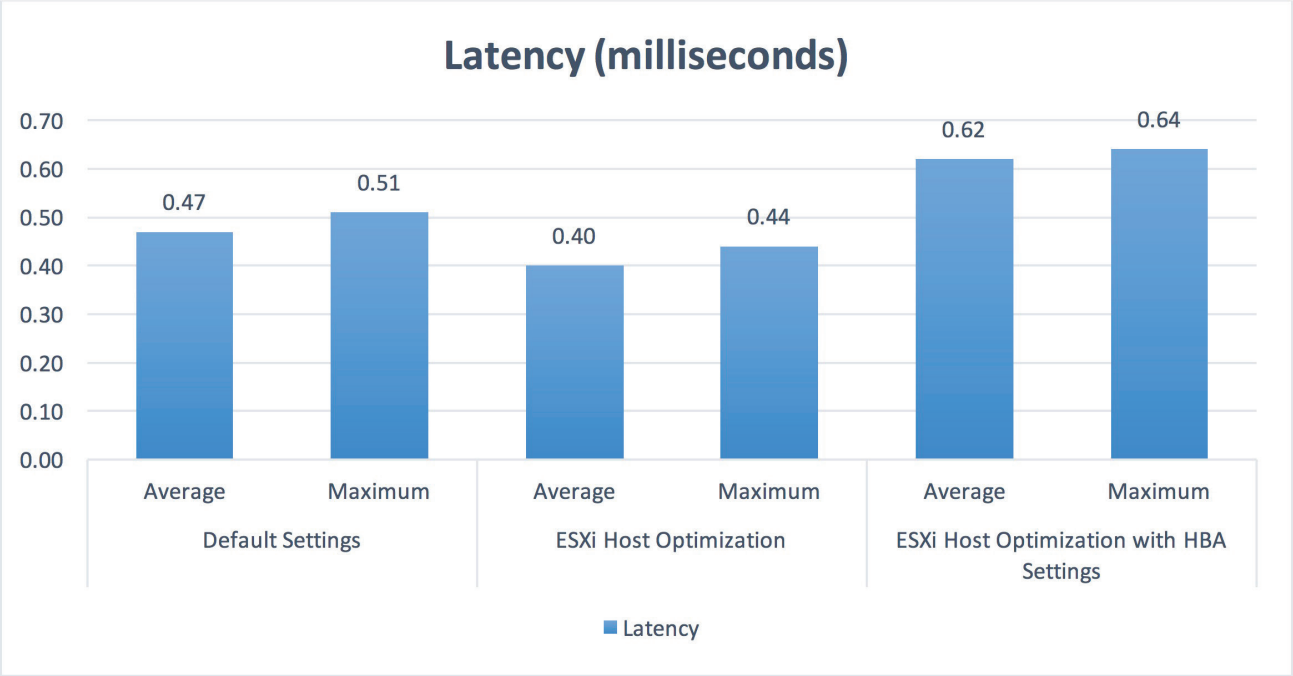


Figure 48: K2 Latency Comparison

Virtual Machines Results (IOMeter)

In this section results are displayed a little different. As each virtual machine performed a different workload in terms of block size, R/W ratio and sequential/random ratio, the following charts are grouped by the virtual machines and not by the different configurations as we did with the K2 results comparison before.

In the following three charts (IOPS, Throughput and Latency) we can see a similar behavior that we already covered before – the performance results for the Default Settings (Blue bars) and for the ESXi Host Optimizations (Orange bars) are equally distributed between the four virtual machines.

Starting with the IOPS chart, we can see that once we configured the HBA Settings each virtual machine performs per the workload it is configured with. For the first virtual machine, Mixed01, we see higher value for IOPS comparing to the rest of the virtual machines. The reason is that Mixed01 runs a workload of small block size (4KB) where the latency is lower and the IOPS rate tends to be higher. Following the same idea – we can see that the greater the block size, the less IOPS, yet the overall IOPS produced with the HBA Settings configured was the best within all configurations.

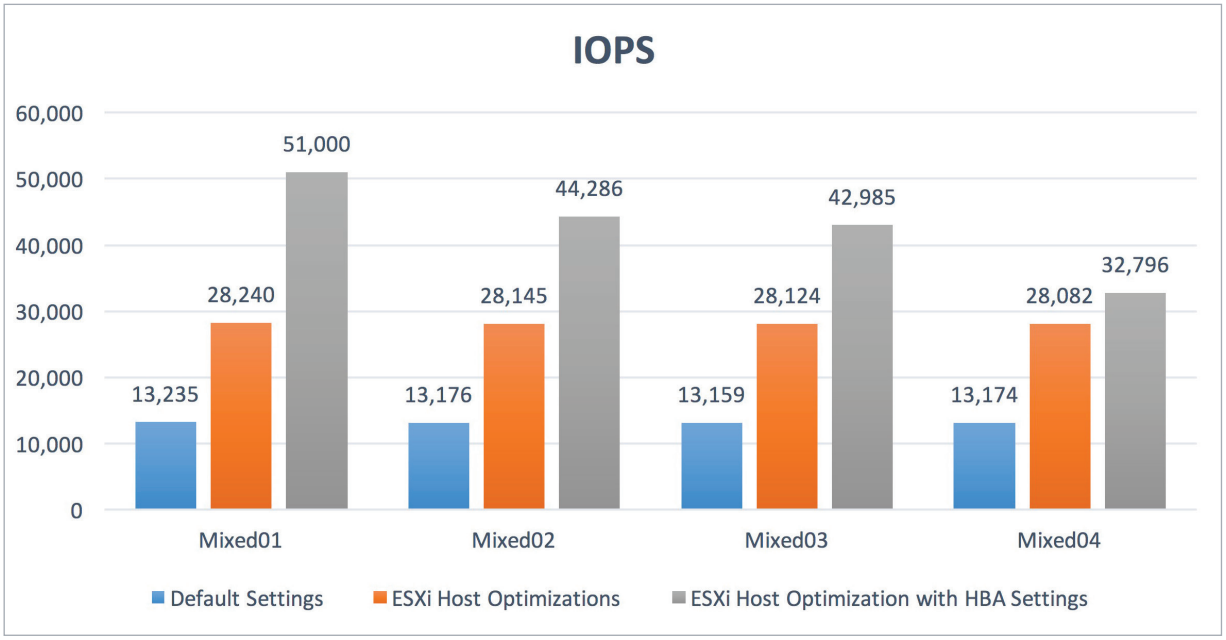


Figure 49: Virtual Machines IOPS Comparison

Moving on to Throughput, we can see the same behavior as we saw with the IOPS chart in relation to the workload distribution in the Default Settings (Blue bars) and the ESXi Host Optimizations (Orange bars). Although virtual machine Mixed04 has generated the lowest number of IOPS, the configured block size is relatively large (32KB) hence the total throughput for the virtual machine has reached 1GB/s.

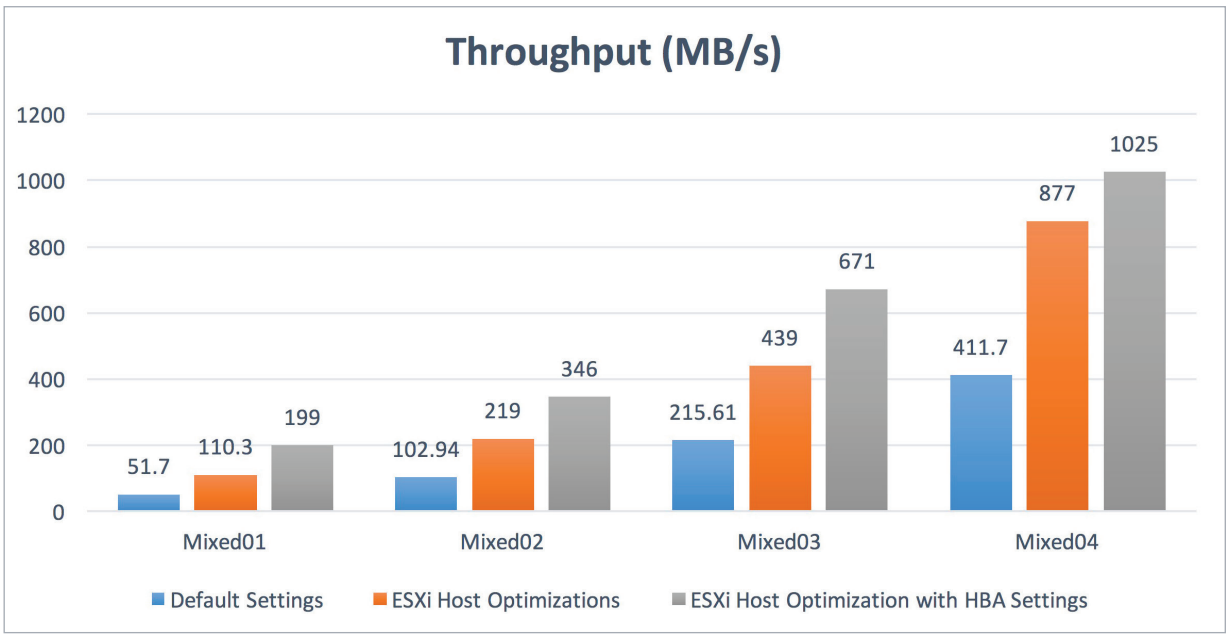


Figure 50: Virtual Machines Throughput Comparison

In the last chart of latency comparison, we see once again the same behavior – latency is consistent across all virtual machines for the Default Settings (Blue bars) and for the ESXi Host Optimizations (Orange bars) due to the same reasons. When HBA Settings are configured, we can see that the latency for each virtual machine is different and fits the workload in each case.

Keep in mind that values presented in the following chart are reported by IOMeter inside the guest virtual machines. Once a data block leaves the K2, it travels a long way until it reaches the IOMeter application. While going down the path, a data block can be queued and delayed in multiple points such as the HBA, the ESXi host and the virtual machine operating system. Those queues and delays along the way are accumulated and reflected in the IOMeter reported results.

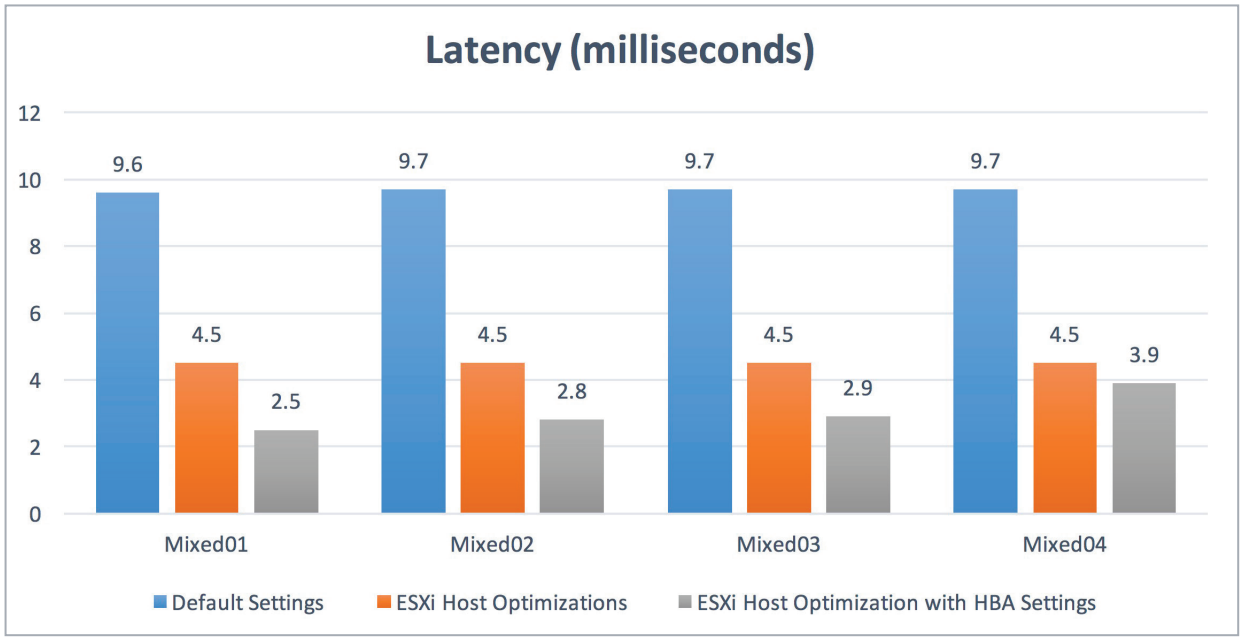


Figure 51: Virtual Machines Latency Comparison

Performance Comparison Conclusions

Following the three-steps comparison, we showed the amount of impact that each of the recommended settings has on the K2 and the virtual machines performance.

At the second step where we applied the ESXi Host Optimizations only we saw about 100% increment in performance, meaning we doubled the IOPS and Throughput performance while latency was reduced. When moved to the last part including the HBA settings, we saw an improvement of about 30% - 50% in performance, depending on the workload in question.

The best results were when all recommended settings were applied, both ESXi Host Optimizations and the HBA recommended settings. Yet, we are aware that changing HBA settings is not always an option as it may influence the SAN environment in ways that must are not beneficial to other parts of the datacenter.

Clearly, the ESXi Host Optimizations settings that are directed at K2 like the Path Selection Policy and the Number of Outstanding IOs With Competing Worlds should be applied in any case, as these settings has massive impact on performance and are targeted at the K2 only.

Kaminario Integrations with VMware

Kaminario's vCenter Plug-In (VCP)

The Kaminario Plug-in is a browser based tool that integrates with the vSphere vCenter Web Client, providing an alternative interface that allows you to monitor and manage the Kaminario K2 storage system.

The Kaminario Plug-in allows you to view the K2 storage system's status and enables you to create new Datastores. You can also monitor real-time error and alert messages for events that may occur in the K2 system.

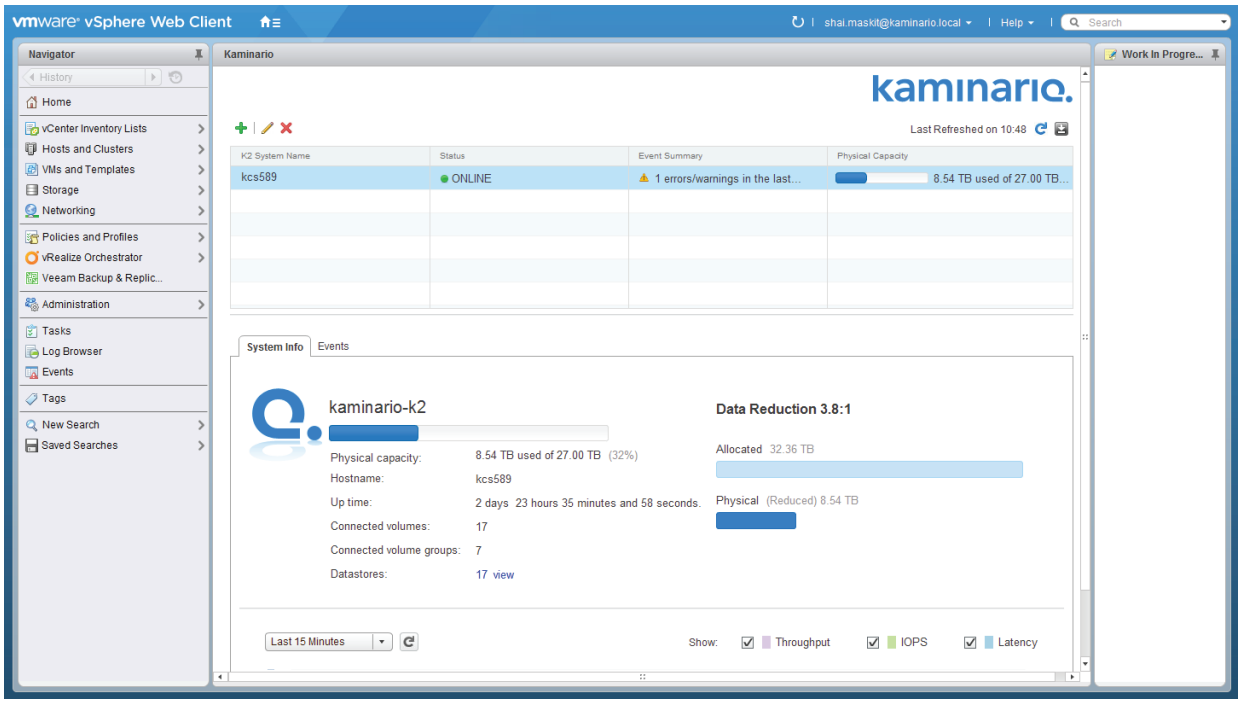


Figure 52: Kaminario vCenter Plug-in

For more information on the Plug-in and installation instructions refer to the "K2 Plug-in for VMware vCenter Server User Guide" document available on Kaminario's Customer Portal.

Kaminario's SRA for VMware SRM

Kaminario K2 Storage Replication Adapter (SRA) for VMware Site Recovery Manager (SRM) is an application that allows VMware SRM to communicate with the Kaminario K2 all-flash storage array. SRM, using the Kaminario K2 SRA, can discover replicated K2 Volume Groups and its corresponding Volume Groups in the remote K2. In addition to the discovery process, the SRA is responsible for suspending, reversing, failing over and restoring replication session in the Kaminario K2.

Using the Kaminario K2 SRA improves the redundancy and availability of the organization's virtual infrastructure and allows a relatively easy Disaster Recovery (DR) solution.

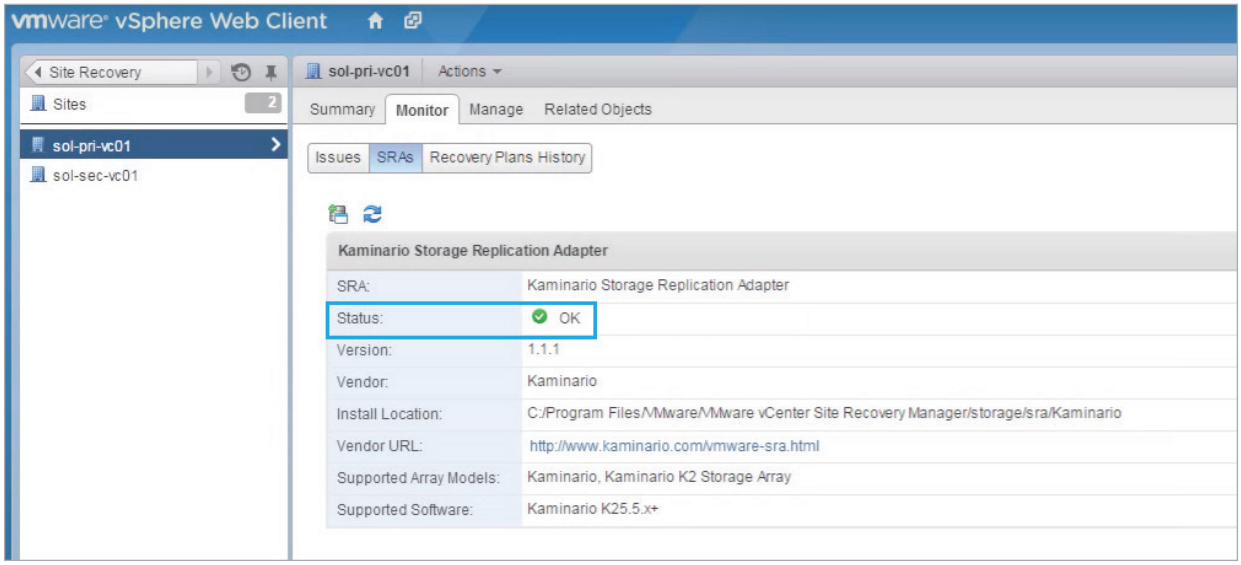


Figure 53: Kaminario SRM

For more information on the Kaminario K2 SRA for SRM including installation instructions refer to the "Kaminario K2 and VMware SRM Guide" document available on Kaminario's Customer Portal.

Kaminario's Content Pack for vRealize Log Insight

VMware vRealize Log Insight is a powerful tool for log management and analytics, supporting heterogeneous environments including physical, virtual and cloud environments. Log Insight is highly customizable and allows a user to create custom field extractors, filters, dashboards and more. Using Log Insight gives a user the power to analyze sophisticated environments with multiple log input streams in an easy and quick manner.

Kaminario's Content Pack is an extension to Log Insight which adds built-in filters, dashboard, queries and more. With Kaminario's Content Pack, a system administrator can monitor K2 arrays easily with no need to create custom rules or dashboards (although a user can always do so) but use the provided objects from Kaminario's Content Pack.

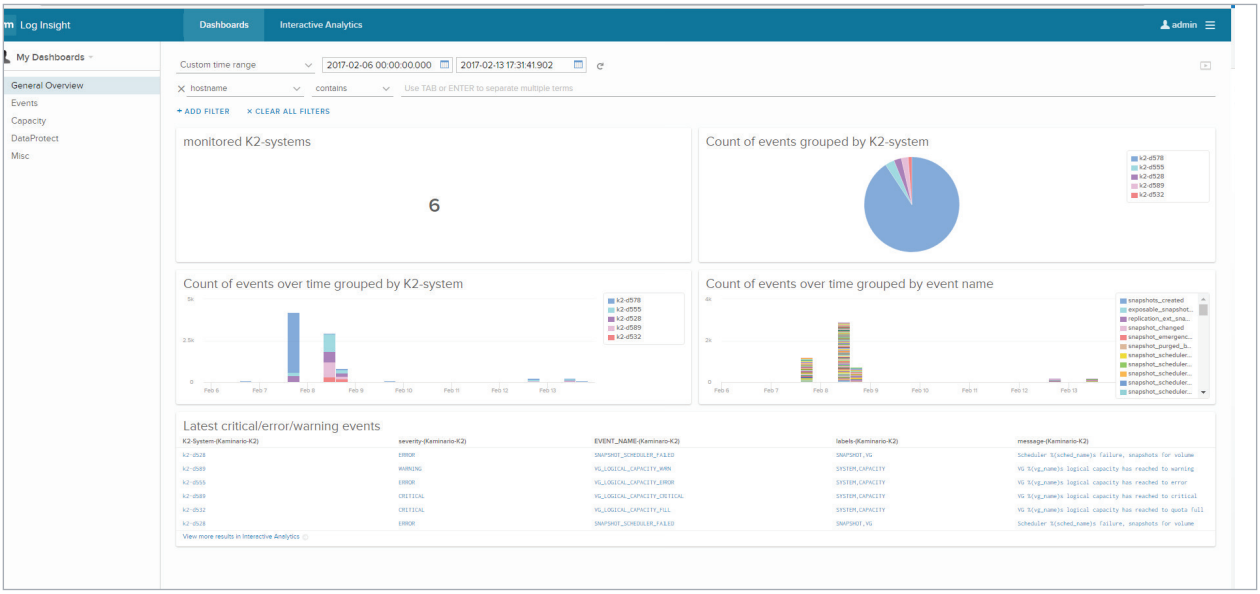


Figure 54: VMware Log Insight

For more information on the Kaminario's Content Pack for vRealize Log Insight including installation instructions refer to the "Kaminario's Content Pack for vRealize Log Insight" document available on **Kaminario's Customer Portal.**

kaminario.

Contact

Contact a business development representative to answer any questions you may have.

Schedule a Demo

Schedule a demo with an engineer and learn if Kaminario's solution works for you.

Request a Quote

Request a quote for your application from our business development team.

About Kaminario

Kaminario, the leading all-flash storage company, is redefining the future of modern data centers. Its unique solution enables organizations to succeed in today's on-demand world and prepares them to seamlessly handle tomorrow's innovations. Only Kaminario K2 delivers the agility, scalability, performance and economics a data center requires to deal with today's cloud-first, dynamic world and provide real-time data access -- anywhere, anytime. Hundreds of customers rely on the Kaminario K2 all-flash array to power their mission critical applications and safeguard their digital ecosystem. Headquartered in Needham, MA, Kaminario works with an extensive network of resellers and distributors, globally.

For more information, visit www.kaminario.com