Enhanced vMotion Compatibility (EVC) Explained

VMware Application Acceleration

Table of contents

Enhanced vMotion Compatibility (EVC) Explained

Overview

vSphere Enhanced vMotion Compatibility (EVC) ensures that workloads can be live migrated, using vMotion, between ESXi hosts in a cluster that are running different CPU generations. The general recommendation is to have EVC enabled as it will help you in the future where you'll be scaling your clusters with new hosts that might contain new CPU models. To enable EVC in a brownfield scenario can be challenging. That's why we stress to have it enabled from the get-go. This blog post will go into details about EVC and the per-VM EVC feature.

How does EVC work?

The way EVC allows for uniform vMotion compatibility, is by enforcing a CPUID (instruction) baseline for the virtual machines running on the ESXi hosts. That means EVC will allow and expose CPU instruction-sets to the virtual machines depending on the chosen and supported compatibility level. If you would add a newer host to the cluster, containing newer CPU packages, EVC would potentially hide the new CPU instructions to the virtual machines. By doing so, EVC ensures that all virtual machines in the cluster are running on the same CPU instructions allowing for virtual machines to be live migrated (vMotion) between the ESXi hosts.

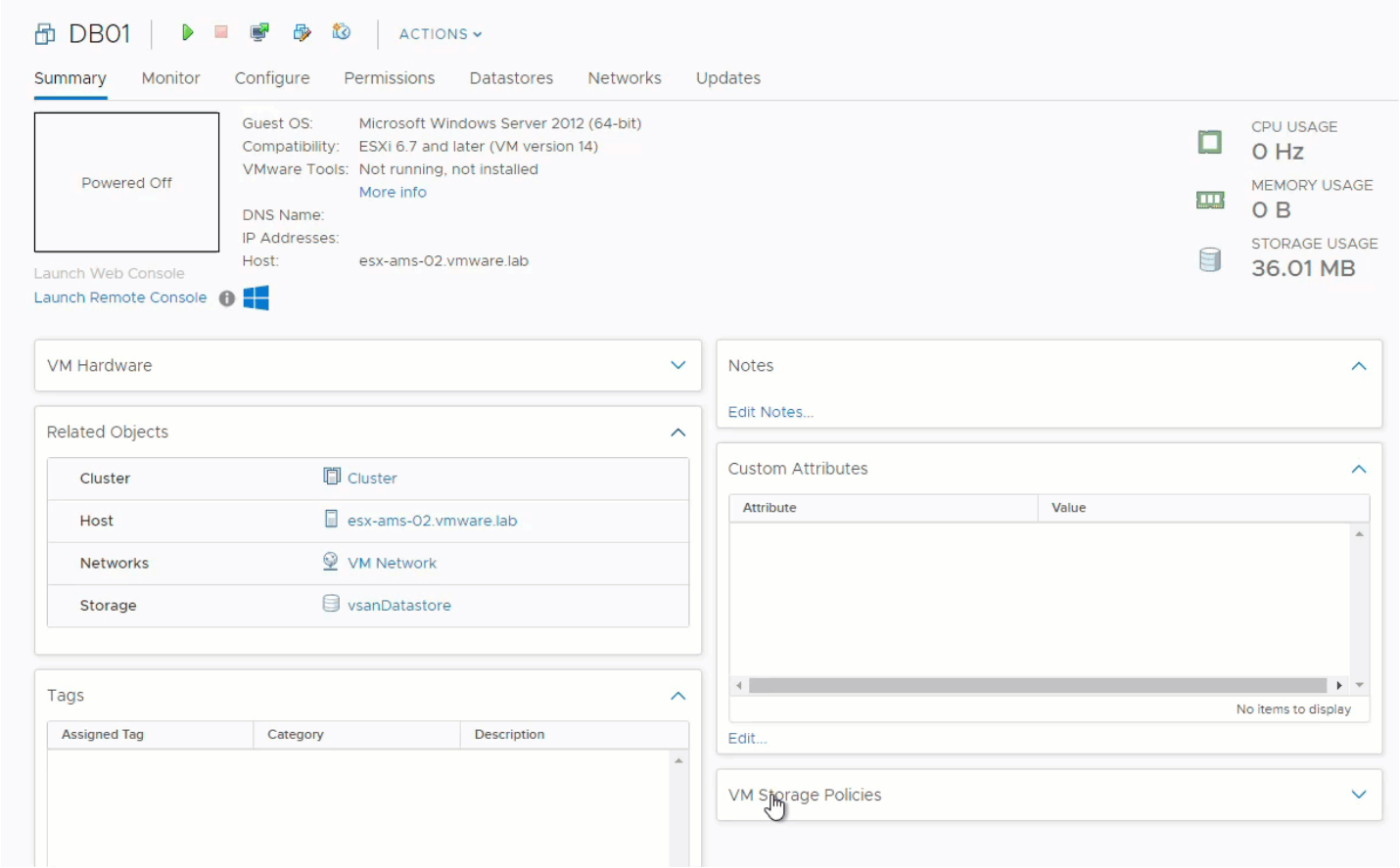
EVC determines what instructions to mask from the guest OS by using the CPUID. Basically, you can look at the CPUID being an API for the CPU. It allows EVC to learn what instruction-sets the CPU is capable of doing, and what instructions needs to be masked depending on the configured EVC baseline. When EVC is enabled on the cluster, all ESXi hosts in the cluster must adhere to that setting.

This VMware KB article goes into more detail about all current EVC baselines and what CPU instructions they expose to the virtual machines.

Per-VM EVC

EVC is a cluster level setting that supports virtual machine mobility within a cluster. When a virtual machine is moved to another cluster, either on-prem or in a hybrid cloud environment, it loses its EVC configuration depending on the destination environment. Next to that, it is challenging to change a cluster EVC baseline in a environment with live workloads.

By implementing per-VM EVC, the EVC mode becomes an attribute of the virtual machine rather than the specific processor generation it happens to be booted on in the cluster. This helps to be more flexible with EVC enablement and baselines. We introduced the per-VM EVC feature in vSphere 6.7. Virtual machine hardware version 14 or up is required to enable per-VM EVC. When a virtual machine is powered off, you can change the per-VM EVC baseline.



The per-EVC configuration is saved in the vmx file. The vmx file is the file used as a value dictionary that provides the configuration of the virtual machine. If the virtual machine is migrated to another cluster, the per-EVC configuration is moving along with the virtual machine itself. The vmx file will contain *featMask.vm.cpuid* lines like the following when per-VM EVC is enabled:

featMask.vm.cpuid.Intel = "Val:1"
featMask.vm.cpuid.FAMILY = "Val:6"
featMask.vm.cpuid.MODEL = "Val:0x4f"
featMask.vm.cpuid.STEPPING = "Val:0"
featMask.vm.cpuid.NUMLEVELS = "Val:0xd"

Customer Feedback

A recent Twitter poll showed some interesting results and feedback. It looks like 80% of our customers are in fact using EVC. However, taking a look at our telemetry data, the number of EVC enabled clusters or virtual machines showed a slightly different picture. It's good to see that a large proportion of our customer-base already benefits from EVC by having it enabled on their clusters and/or virtual machines.

Going through the comments, the general consensus around having EVC enabled by default differs. We see a lot of customers that understand enabling EVC in a brownfield environment is challenging, so they opt to enable EVC from the start. On the other hand, we see customers who didn't enable EVC because they have a uniform clusters and don't see the value of having it enabled. It is important to understand that the EVC feature itself has zero overhead on your virtual infrastructure. However, it can save you from the burden of enabling cluster EVC later on when you might want to scale your cluster with additional hosts that might contain newer CPU versions.

Another concern of customer is the impact on performance. What about workloads that cannot use the latest and greatest CPU instructions because of the configured EVC baseline? It does depend on the workloads, but overall we don't see significant impact on performance because of all new CPU instructions not being exposed to the application running inside the guest OS. We did release a paper that goes into detail on this topic.

To enable EVC on a live environment with virtual machines powered on, you would need to power down the virtual machines in order to change the EVC configuration. This is an area where per-VM EVC helps. Check out this extensive post by Kyle Ruddy on how you can enable per-VM EVC in an automated way.

Check EVC Configurations

To gain insights of your environment and what EVC configurations are used, scripting can be utilized. The following snippets allows for creating an overview that includes the virtual machines and the virtual machine EVC level next to the cluster EVC level. Because it is tabular data, it is easily exported to a CSV file by adding | *ConvertTo-CSV.*

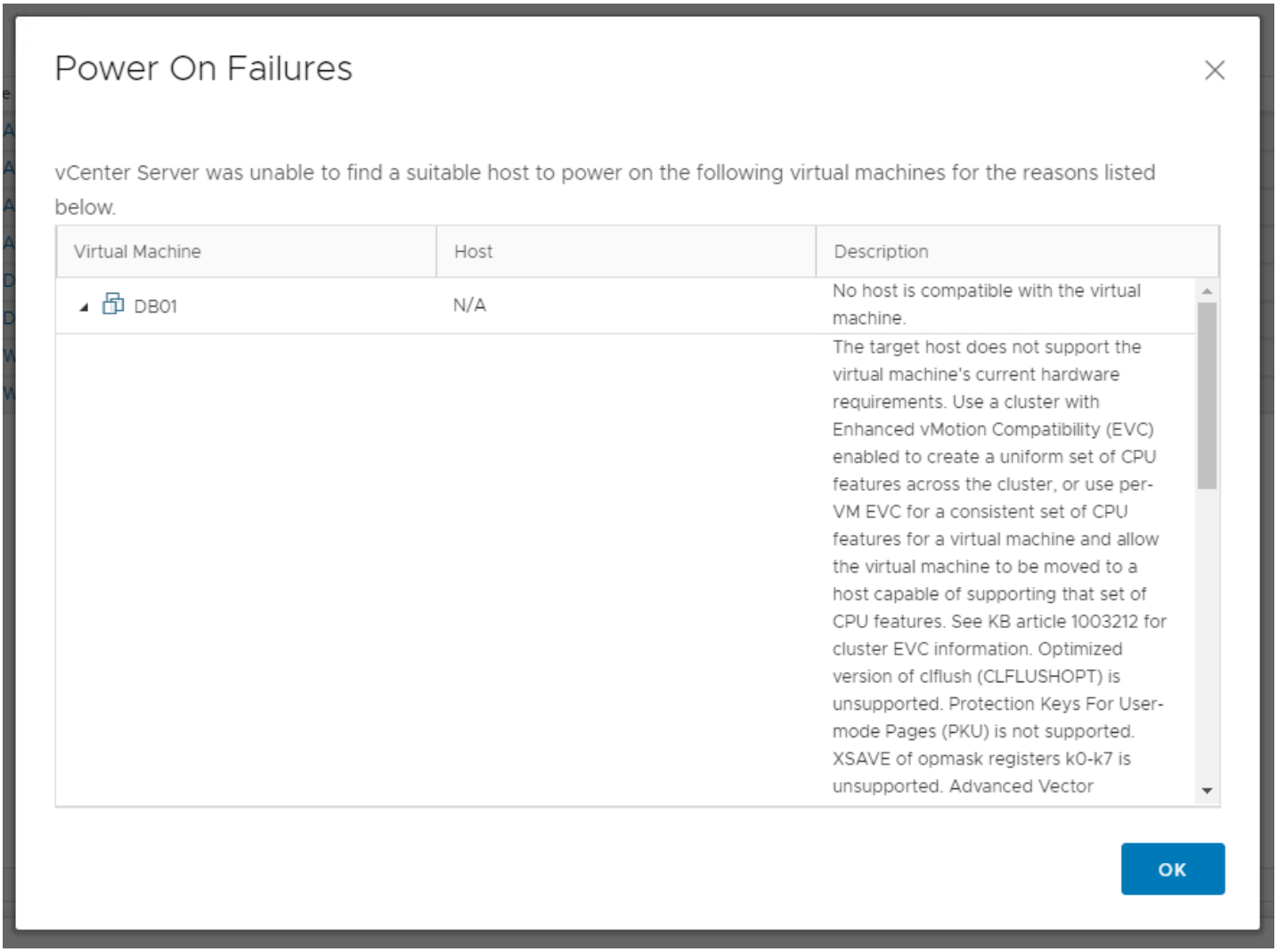
```
Get-VM | Select Name,HardwareVersion,
 @{Name='VM_EVC_Mode';Expression={$_.ExtensionData.Runtime.MinRequiredEVCModeKey}},
 @{Name='Cluster_Name';Expression={$_.VMHost.Parent}},
 @{Name='Cluster_EVC_Mode';Expression={$_.VMHost.Parent.EVCMode}} | ft
```

Running this PowerCLI command will give you insights on what EVC baselines are configured when EVC is enabled. When the cluster EVC mode is enabled, the script will show the used baseline. It also shows the VM hardware version because per-EVC is only available for VM hardware version 14 and up. If a virtual machine is configured with per-VM EVC, the baseline could differ from the cluster EVC configuration. In the following explamary output you'll notice the virtual machine 'DB01' with a different per-VM EVC baseline as opposed to the cluster setting.

```
PS C:\> Get-VM | Select Name,HardwareVersion,
>> @{Name='VM-EVC-Mode';Expression={$_.ExtensionData.Runtime.MinRequiredEVCModeKey}},
>> @{Name='Cluster Name';Expression={$_.VMHost.Parent}},
>> @{Name='Cluster-EVC-Mode';Expression={$_.VMHost.Parent.EVCMode}} | ft

Name  HardwareVersion VM-EVC-Mode     Cluster Name Cluster-EVC-Mode
----  --------------- -----------     ------------ ----------------
APP02 vmx-13          intel-broadwell Cluster      intel-broadwell
APP01 vmx-14          intel-broadwell Cluster      intel-broadwell
DB02  vmx-14          intel-broadwell Cluster      intel-broadwell
DB01  vmx-14          intel-skylake   Cluster      intel-broadwell
AD02  vmx-14          intel-broadwell Cluster      intel-broadwell
WEB01 vmx-13          intel-broadwell Cluster      intel-broadwell
WEB02 vmx-14          intel-broadwell Cluster      intel-broadwell
AD01  vmx-13          intel-broadwell Cluster      intel-broadwell
```

This is a supported situation. However, if a virtual machine per-VM EVC baseline is higher than supported by the ESXi hosts in the cluster, the virtual machines will not power on because there is no host compatible with its per-VM EVC baseline.



You should always verify if your ESXi hosts support the configured EVC baselines to ensure it can accommodate the virtual machines running a per-VM EVC configuration. If you need information about your ESXi hosts and what the maximum supported EVC level is, you could issue the following PowerCLI command: **Get-VMHost | Select-Object Name,ProcessorType,MaxEVCMode**

The output shows you the ESXi hosts, what CPU packages they are running, and the maximum supported EVC baseline.

```
PS C:\> Get-VMHost | Select-Object Name,ProcessorType,MaxEVCMode

Name                 ProcessorType                          MaxEVCMode
----                 -------------                          ----------
esx-ams-01.vmware.lab Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz intel-broadwell
esx-ams-02.vmware.lab Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz intel-broadwell
esx-ams-03.vmware.lab Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz intel-broadwell
esx-ams-04.vmware.lab Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz intel-broadwell
```

As stated before, the key takeaway is that the general recommendation is to have EVC enabled. For a more granular approach and hybrid cloud support, the per-VM EVC feature is a good starting point when implementing EVC in your virtual infrastructure. Having the EVC feature enabled will allow you to benefit from workload mobility to the fullest!