How to Tune vMotion for Lower Migration Times?

VMware Application Acceleration

vmware by Broadcom © VMware LLC.

Table of contents

vmware by Broadcom © VMware LLC.

How to Tune vMotion for Lower Migration Times?

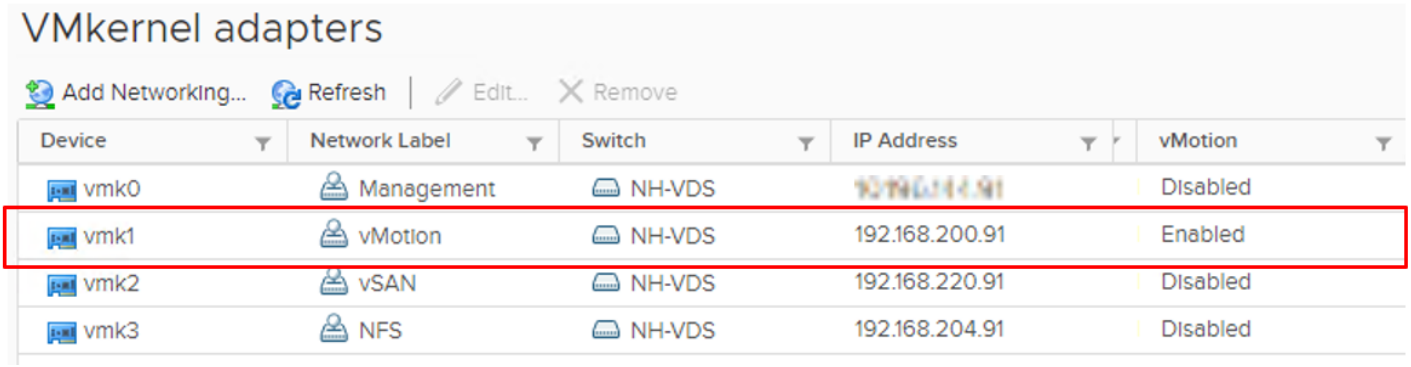
Overview

In another article we went over the vMotion process internals. Now that we understand how vMotion works, lets go over some of the options we have today to lower migration times even further. When enabled, by default, vMotion works beautifully. However, with high bandwidth networks quickly becoming mainstream, what can we do to fully take advantage of 25, 40 or even 100GbE NICs? This article goes into details on vMotion tunables and how they can help to optimize the process.

© VMware LLC.

Streams and Threads

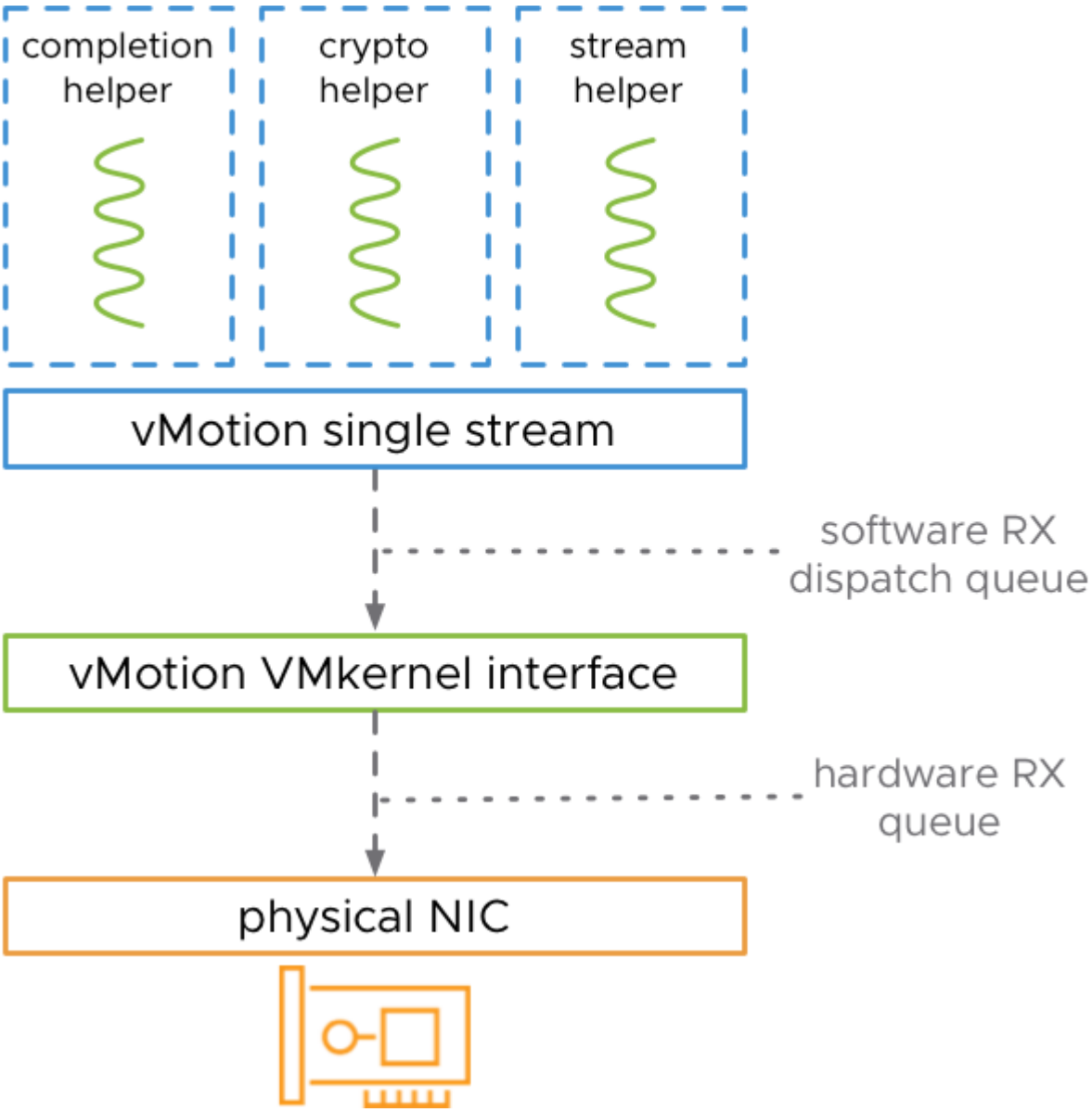
To understand how we can tune vMotion performance and thereby lower live-migration times, we first need to understand the concept of vMotion streams. The streaming architecture in vMotion was first introduced in vSphere 4.1 and has been developed and improved ever since. One of the prerequisites for performing live-migrations is to have a vMotion network configured. As part of the enabling vMotion, you need at least one VMkernel interface that is enabled for vMotion traffic on your applicable ESXi hosts.



When you have a VMkernel interface that is enabled for vMotion, a single vMotion stream is instantiated. One vMotion stream contains three helpers:

- **Completion helper**; Supports the vMotion process.
- **Crypto helper**; Dedicated for encrypting and decrypting traffic when Encrypted vMotion is used.
- **Stream helper**; Responsible for putting the data on the wire (network).

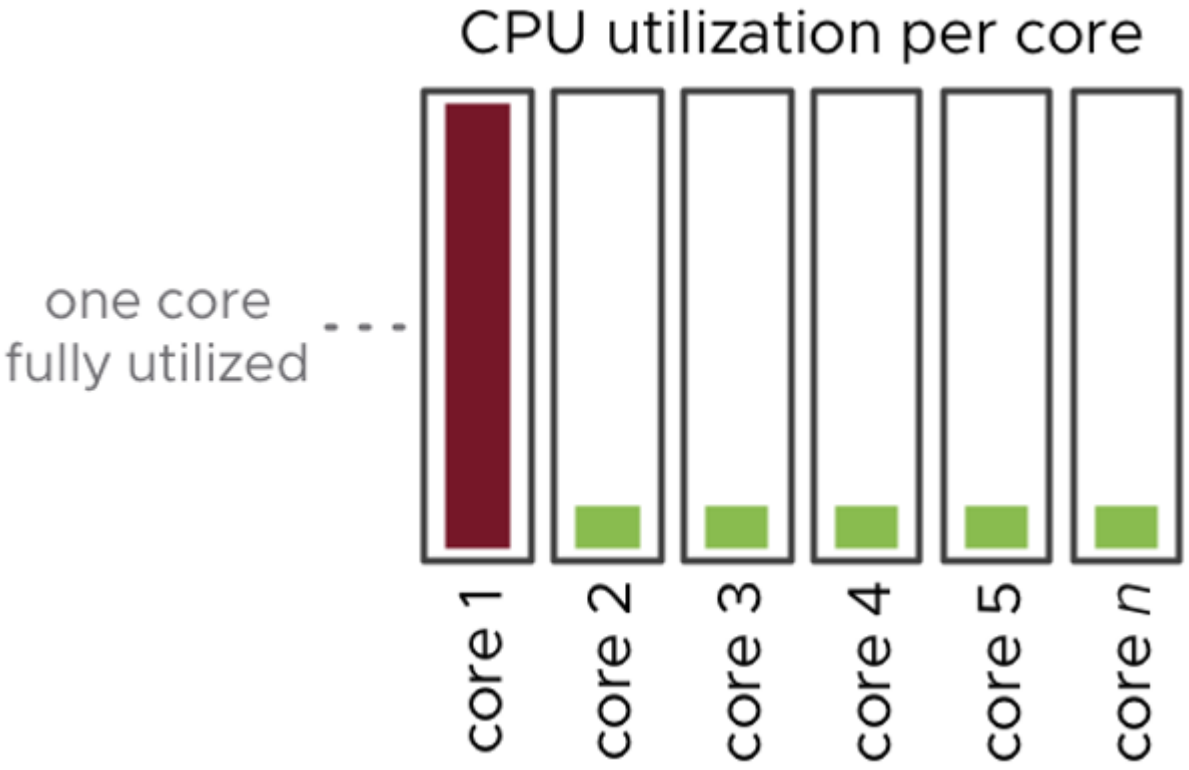
© VMware LLC.



The term 'helper' is used internally, but it is really a thread. One thread is able to consume one CPU core.

The Challenge

Looking at live-migration times, the duration of a migration is largely depended on the bandwidth that is consumed by the migration module to transmit the memory from the source to the destination ESXi host. However, we are constrained by only one stream helper, thus one thread that equals one CPU core. While this typically don't pose a problem when using modern CPU packages and 10GbE networks, it does become a challenge when using 25GbE or higher bandwidth NICs and networks. A 25GbE or higher NIC will not be fully utilized by one vMotion stream.


CPU utilization per core
one core
fully utilized
core 1
core 2
core 3
core 4
core 5
core n

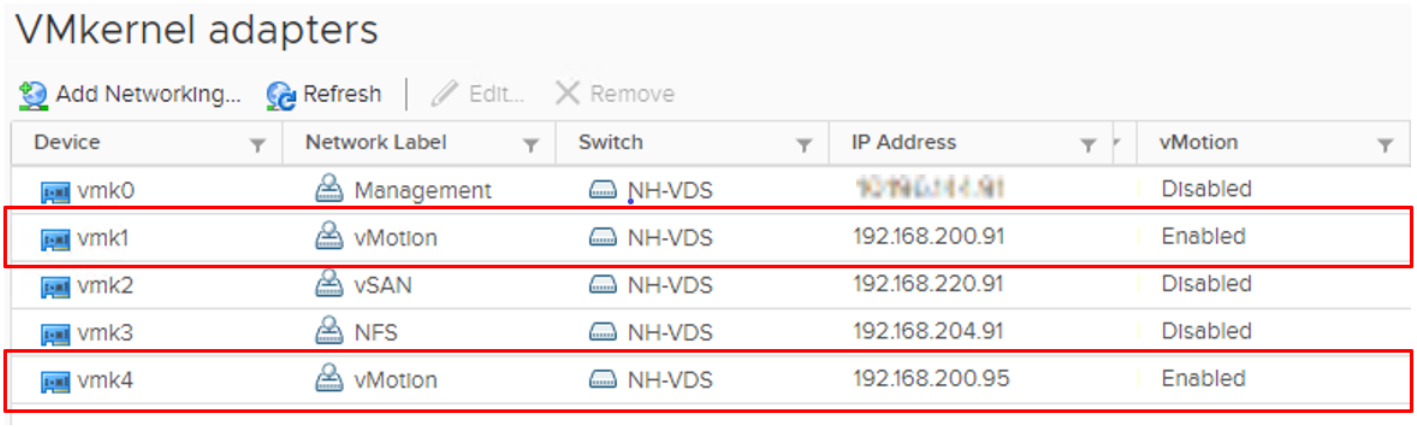
© VMware LLC.

Scale vMotion Performance

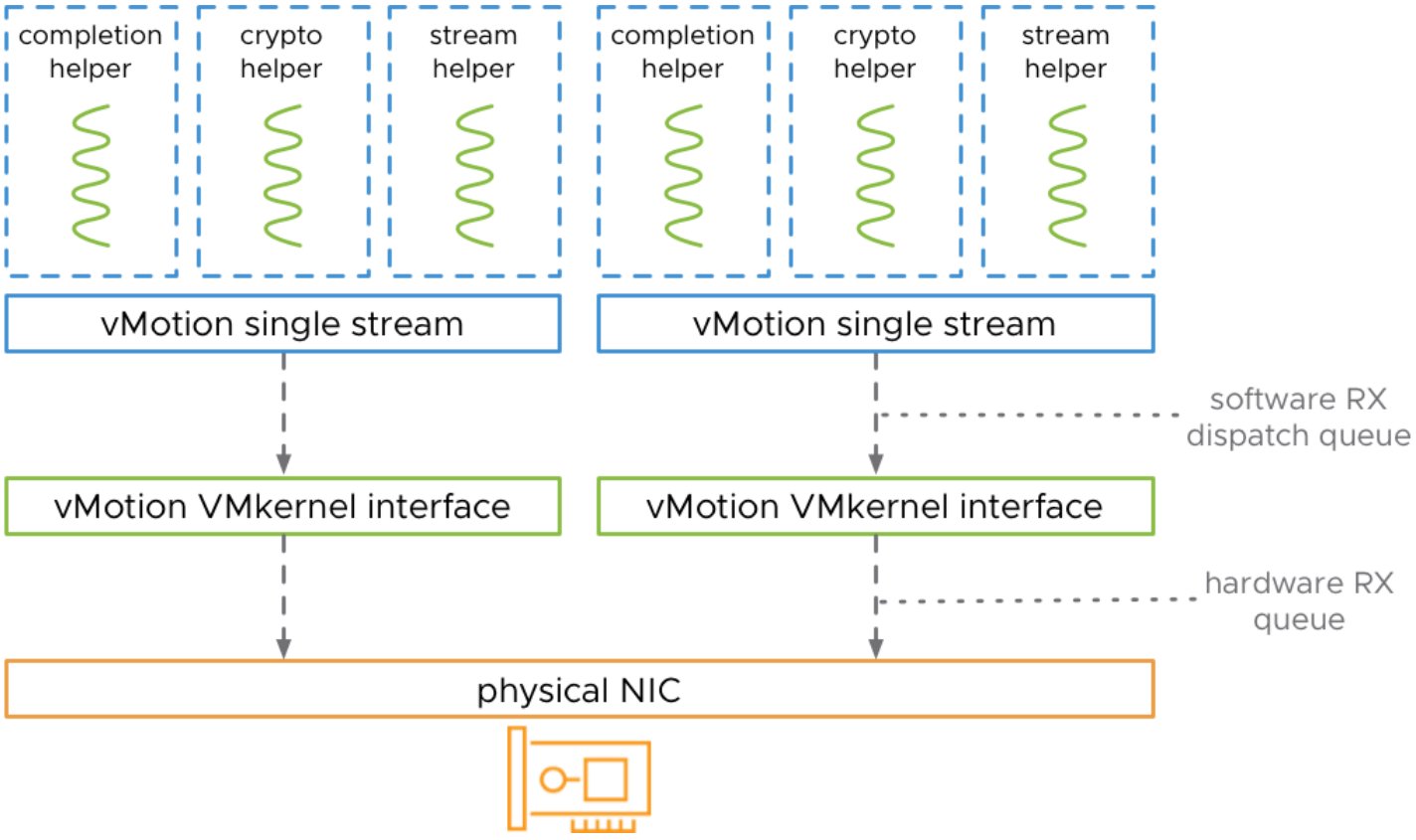
We do have ways to mitigate the risk of being constrained by one CPU core. The most straightforward solution is to instantiate more vMotion streams that contain stream helpers. To do that, we have two options.

Option 1: Multiple VMkernel Interfaces

Perhaps the easiest way to achieve this, is to configure multiple VMkernel interfaces using the same NIC and network. Do not confuse this with Multi-NIC vMotion. That is all about making sure that you spread vMotion traffic over multiple 1 or 10GbE NICs because these speeds are easily saturated. We are talking how to up the bandwidth utilization for vMotion on a single NIC. By doing so, we are lowering live-migration times.



With each created VMkernel interface that is enabled for vMotion, a new vMotion stream is spun up. Each stream containing the helpers as discussed. The multiple stream helpers assist in putting more data on the vMotion network and by doing so, utilization more bandwidth to copy the memory pages across the vMotion network, reducing the time it takes to get to memory convergence between the source and destination ESXi host. That results in vMotion operations being completed faster.



A single vMotion stream has an average bandwidth utilization capability of 15 GbE. When we look at various NIC speeds, that lead to the following:

- 25 GbE : 1 stream = ~15 GbE
- 40 GbE : 2 streams = ~30 GbE
- 50 GbE : 3 streams = ~45 GbE
- 100 GbE : 6 streams = ~90 GbE

That means that you would need six vMotion VMkernel interfaces on a single ESXi host to be able to use the available bandwidth, using a 100GbE NIC, efficiently.

The downside of creating additional VMkernel interfaces is its operational overhead. All the VMkernel interfaces need to be created on all ESXi hosts and require IP addresses. Doing this on scale can have a significant impact on vSphere management tasks. This task can be fairly easy automated using PowerCLI, but the need for additional IP addresses remain.

Option 2: Tune Single vMotion VMkernel Performance

We do have another option. Using this option will avoid a VMkernel interface sprawl and the IP addresses required like with the first option. However, it is a really advanced option that requires manual configuration on each ESXi host. We have the ability to enforce settings for both the vMotion stream helpers and the default number of Rx queues per VMkernel interfaces that is enabled for vMotion traffic.

Note: In the first iteration of this article, I mentioned the need for the vsi shell to change the settings we're about to discuss. Luckily, we do have other methods. Customers should refrain from using the vsi shell. This is a tool used by VMware support.

To realize more vMotion streams per VMkernel interface and the Rx queues, we start by investigating the following settings:

```
/> get /config/Migrate/intOpts/VMotionStreamHelpers
Vmkernel Config Option {
   Default value:0
   Min value:0
   Max value:32
   Current value:0
   hidden config option:0
   Description:Number of helpers to allocate for VMotion streams, 0 to dynamically allocate one per stream IP
}
/>
/> get /net/tcpip/defaultNumRxQueue
1
/>
```

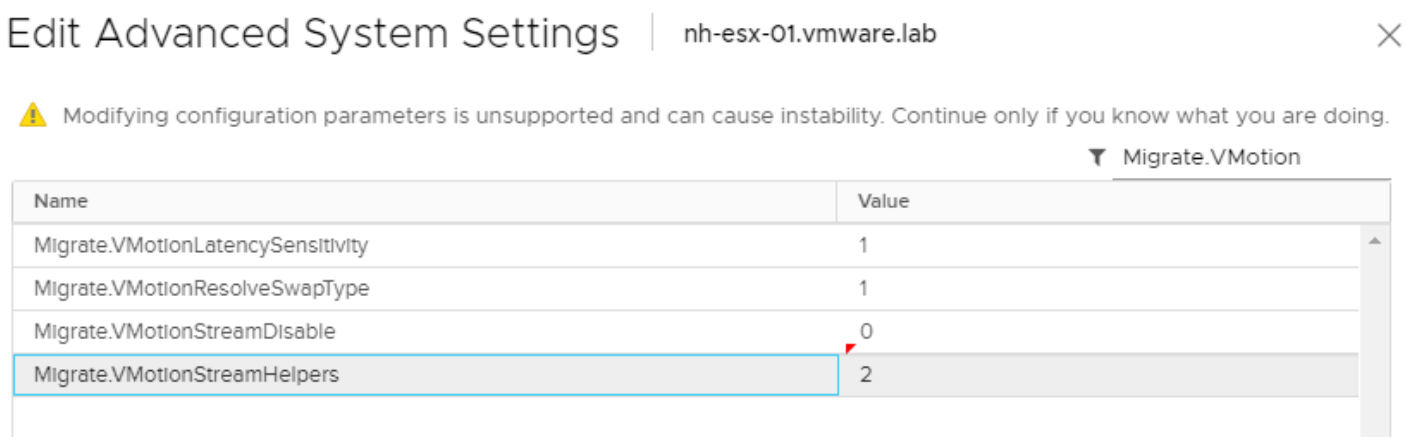
The screenshot above shows the default values for both configurables in the vsi shell. Because vsi shell changes are not persistent and recommended, let's find the ESXi host advanced settings that correlate with the vsi shell equivalents.

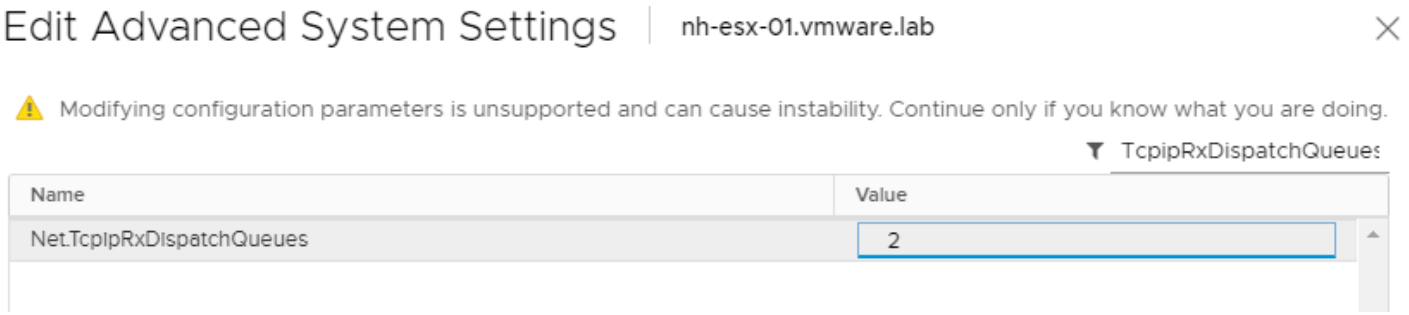
- **/net/tcpip/defaultNumRxQueue** translates to advanced setting **Net.TcpipRxDispatchQueues**
- **/config/Migrate/intOpts/VMotionStreamHelpers** translates to advanced setting **Migrate.VMotionStreamHelpers**

It is important to notice that by default, the setting VMotionStreamHelpers is set to '0', which is the setting to dynamically allocate one stream per IP. As this might change in future releases, be mindful when you adjust this setting to another value. When we configure the VMotionStreamHelpers setting to another value, the TcpipRxDispatchQueues setting should change accordingly. Please note that increasing TcpipRxDispatchQueues requires it to consume more host memory.

In order to change these setting, we can simply configure the ESXi host advanced settings in the UI. In these examples, we configure them to have two Rx queues and two stream helpers per VMkernel interface.

© VMware LLC.

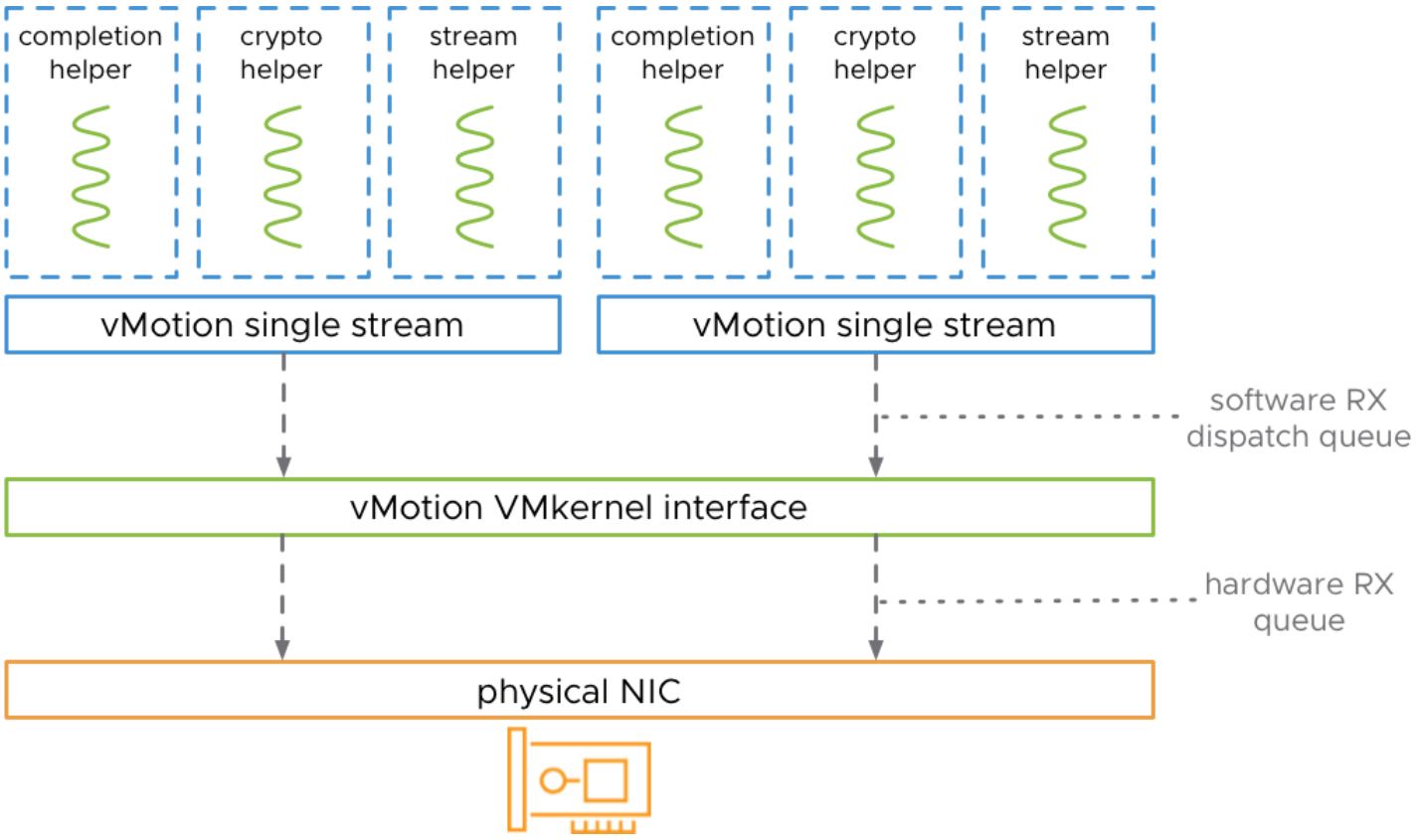




Next to the UI, you can also use PowerCLI to perform the same task:

Get-AdvancedSetting -Entity <esxi host> -Name Net.TcpipRxDispatchQueues | Set-AdvancedSetting -Value '2'
Get-AdvancedSetting -Entity <esxi host> -Name Migrate.VMotionStreamHelpers | Set-AdvancedSetting -Value '2'

After tuning both settings, make sure the ESXi host is rebooted for them to take effect. Using this option we have one VMkernel interface, but in the background, multiple helpers are spun up.



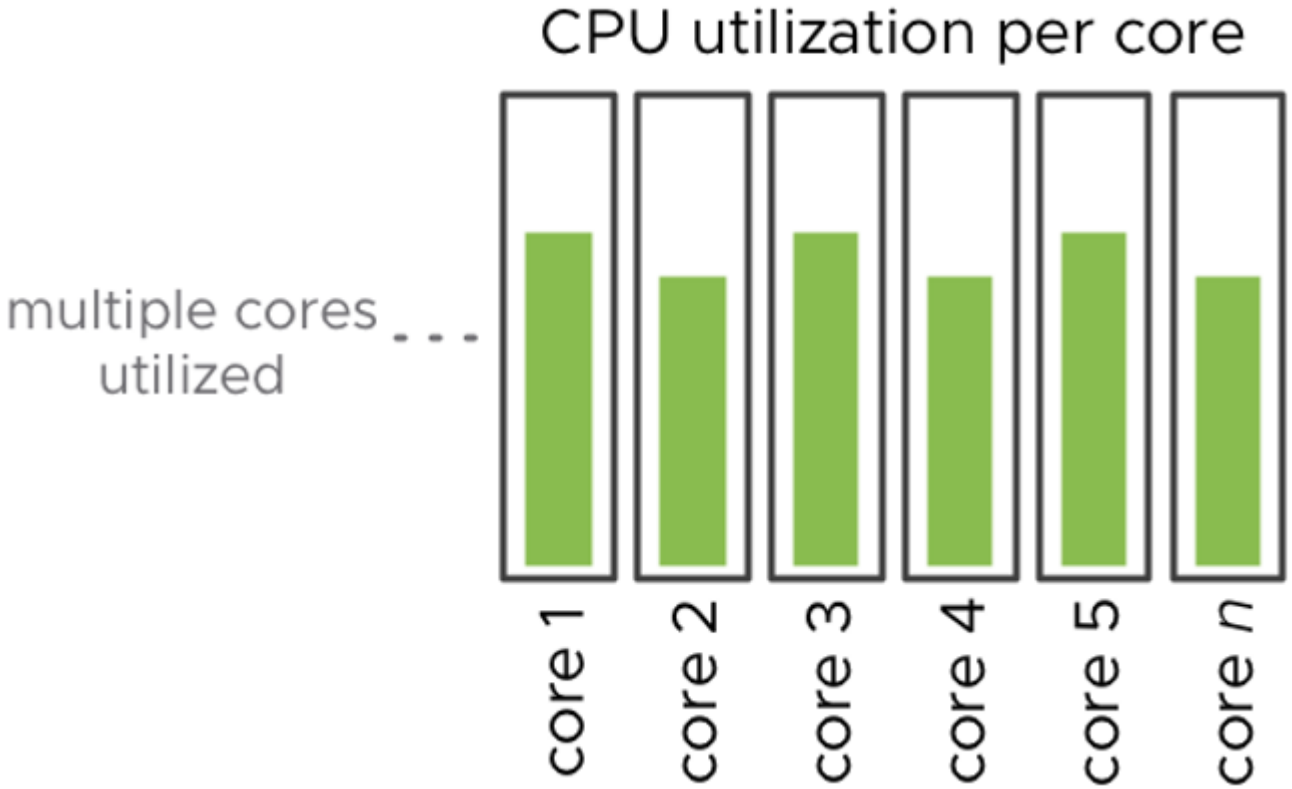
While this is a great solution, the realization of it requires manual configuration and ESXi host reboots.

© VMware LLC.

© VMware LLC.

Result

In the end, we would like to achieve a scenario in which we can efficiently utilize the available bandwidth for vMotion. That also means we have a more 'even' CPU utilization because now we use multiple threads, and so, multiple CPU cores. Especially with larger workloads, tuning vMotion can help to lower migration times significantly.



It's good to know that today, we have multiple options for high bandwidth networks to be fully utilized by vMotion. However, both options require manual configuration. And although you can automate the operations involved, we would rather keep it more simple for you. That is why we are thinking about doing the exact same thing in a more dynamic way. It would be great if a future vSphere release is able to detect the network and NIC bandwidth and specifics to adjust vMotion tunables accordingly. For now, you can benefit from these manual options.

© VMware LLC.

© VMware LLC.

Copyright © 2005-2024 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries.