PREPARING A CLUSTER OF MIXED SERVER PROCESSORS FOR LIVE KVM MIGRATIONS

AT A GLANCE

Qualified workloads running in Virtual Machines (VMs) on AMD EPYC™ and Intel® Xeon® servers can be moved in real time, "live-migrated," between the servers without shutting the servers down if the Linux® operating system on each is configured properly. This brief describes the process to determine and apply the common x86-64 instruction set to be used across the servers to enable VM migration.

INTRODUCTION

Linux and the native Kernel-based Virtual Machine feature (KVM) along with other open-source tools enable a cluster of servers of different microarchitectures and x86-64 instruction support to "live"- or "hot"-migrate KVM processes between the servers. This capability presents a cluster administrator with flexibilities to leverage legacy with new servers, to load-balance workloads, and to build in resiliency, to cite a few examples. Implementing migration requires some straightforward planning and cluster configuration up front.

EXAMPLE USE CASES

- Warm up on the datacenter integration of the first systems of a new generation
- Rolling datacenter refresh
- Dual supplier policy
- Repurpose legacy Kubernetes compute nodes as control nodes
- Maintenance flexibility
- High Availability architecture

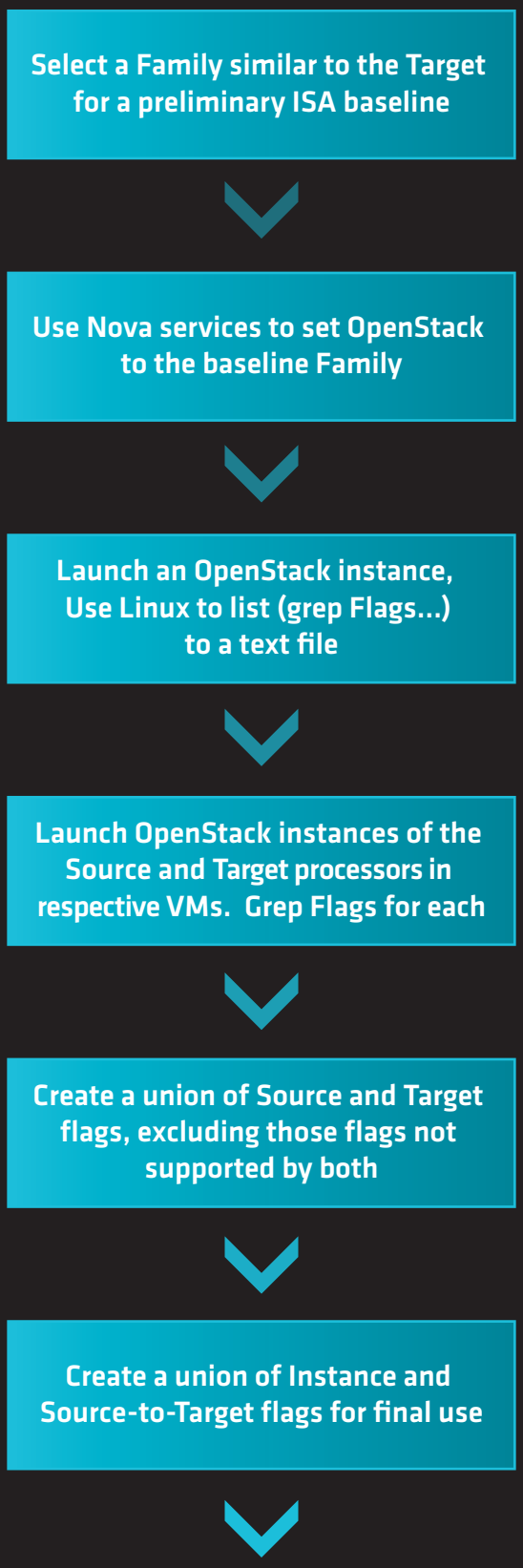
TERMS

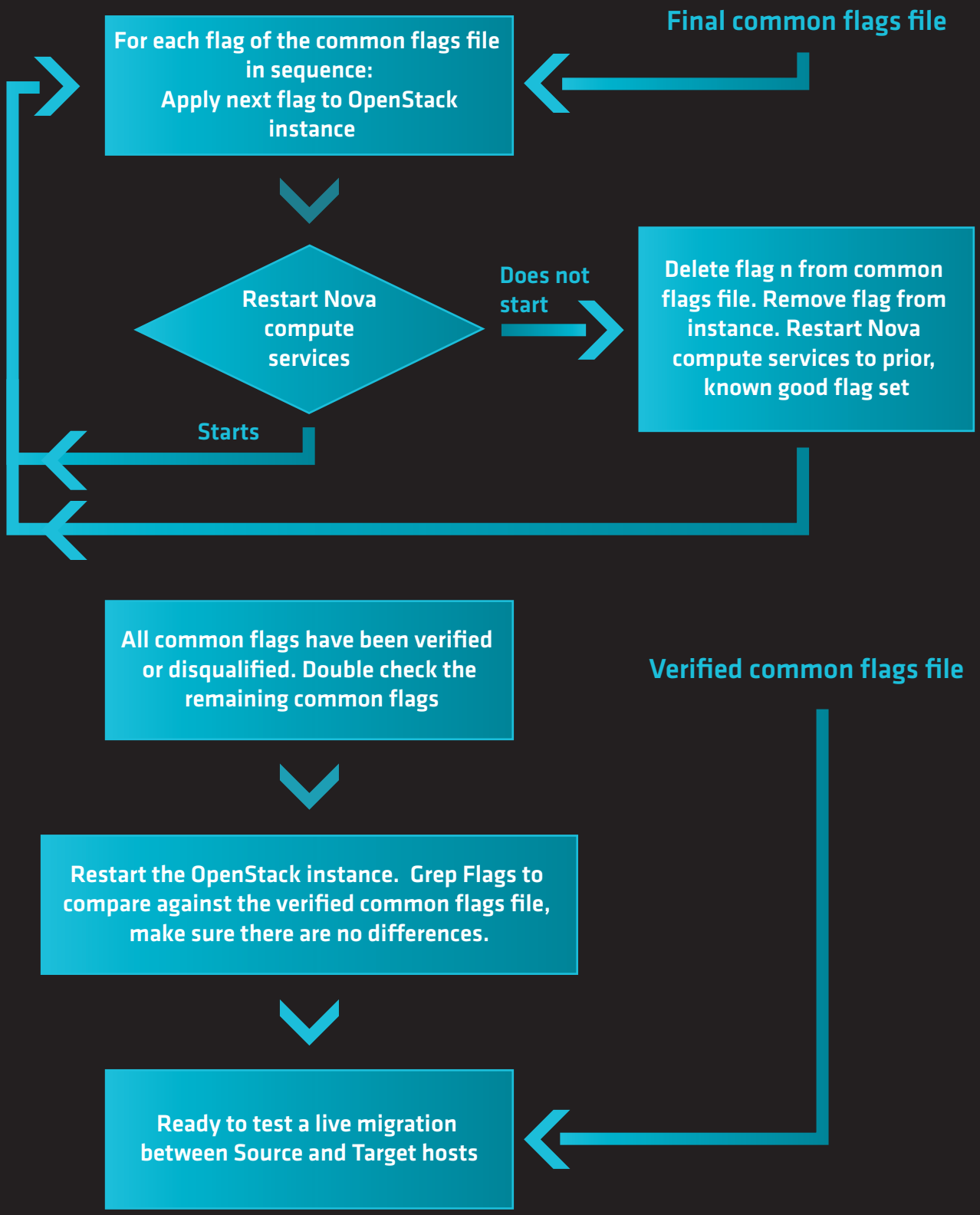
- ISA: x86-64 Instruction Set Architecture and extensions
- Flags: Character string descriptors that identify what instructions the host's processor supports, or
  descriptors that direct the operating system as to what ISA instructions are to be used at the exclusion of others not identified
- Source: the host server or CPU on which an application is running, to be migrated to a Target.
- Target: the host server or CPU to which an application will be migrated from a Source
- Family: A set of a vendor's x86 CPU models that support the same instruction set
- QEMU: A generic and open-source machine (CPU) emulator and virtualizer
- Virsh: open-source program that serves as the main interface for managing virsh guest domains.
- Nova: OpenStack project for provisioning virtual serves as compute instances.

PROCEDURE

The fundamental task is to establish the set of instruction flags that are commonly supported across the various processors to be migrated between. That set will be used to confine each server in the cluster when applications are run so that each node is running the applications on the same instruction set, allowing the application processes to be migrated seamlessly across different processor Families. The following steps outline the process of determining what the common instruct set and flags should be given the specific processor models to be migrated between.

The sequence of steps we'll go through in detail is best introduced with the following flowchart, followed by the step-by-step instructions:


Select a Family similar to the Target for a preliminary ISA baseline
Use Nova services to set OpenStack to the baseline Family
Launch an OpenStack instance, Use Linux to list (grep Flags...) to a text file
Launch OpenStack instances of the Source and Target processors in respective VMs. Grep Flags for each
Create a union of Source and Target flags, excluding those flags not supported by both
Create a union of Instance and Source-to-Target flags for final use


Final common flags file
For each flag of the common flags file in sequence:
Apply next flag to OpenStack instance
Restart Nova compute services
Does not start
Delete flag n from common flags file. Remove flag from instance. Restart Nova compute services to prior, known good flag set
Starts
All common flags have been verified or disqualified. Double check the remaining common flags
Verified common flags file
Restart the OpenStack instance. Grep Flags to compare against the verified common flags file, make sure there are no differences.
Ready to test a live migration between Source and Target hosts

STEP BY STEP

1.

Review the upstream QEMU documentation (for more information see QEMU / KVM CPU model configuration) and select a baseline x86 CPU family that is closest to your target processors considering ABI compatibility.

Note that depending on the version of QEMU, Virsh or Nova, you may need to choose a virtual CPU model that is older or newer for the target ISA set.

The cpu-model selection will not have a performance impact because it is a temporary representation of available physical CPU flags and features and will be extended by adding missing flags in the next steps.

Example Selections:

- Intel Broadwell-noTSX-IBRS
- AMD EPYC-IBPB

2.

In your OpenStack deployment, Nova services need to be configured for this Baseline CPU configuration (nova.conf):

```
-- cpu-mode: custom
-- cpu-model: Broadwell-noTSX-IBRS
-- cpu-model-extra-flags: NONE
```

For more details, see CPU models.

3.

To capture the Baseline flags, launch an OpenStack instance of the Baseline CPU model using any type of Linux operating system, SSH to it and capture the available CPU flags under the instance:

```
$ ssh <instance-name-or-ip> lscpu | grep Flags | awk -F"Flags:" '{print $2}' | sed s/" "/"\n"/g | sort -u | sed '/./!d' > instance.flags-lscpu.sorted
```

4.

Here we assume live migrations are intended to span two distinct CPU models and generations: Source and Target. SSH to an OpenStack Hypervisor running each of these CPU generations and capture the CPU flags:

```
$ ssh <hypervisor-running-source> lscpu | grep Flags | awk -F"Flags:" '{print $2}' | sed s/" "/"\n"/g | sort -u | sed '/./!d' > source.flags-lscpu.sorted

$ ssh <hypervisor-running-target> lscpu | grep Flags | awk -F"Flags:" '{print $2}' | sed s/" "/"\n"/g | sort -u | sed '/./!d' > target.flags-lscpu.sorted
```

5.

Generate a list of common flags between Source and Target:

```
$ diff -urN source.flags-lscpu.sorted target.flags-lscpu.sorted | egrep -v '\-|\+' > source-to-target.common.flags.list
```

6.

Then generate the final common list:

```
$ diff -urN instance.flags-lscpu.sorted source-to-target.common.list | egrep '^\+[^+]' | sed 's/+//g' > final-common.flags.list
```

7.

Using the common flags list, for each flag, one at a time in sequence:

1. Apply configuration flag.
   a. Example: avx2
2. Restart Nova compute services
   a. Depending on the version of QEMU, Virsh or Nova, there might be overlapping or unsupported flags that would cause nova-compute services not to start.
   b. Some flags might be simply too low level to be supported on Nova.
   c. If the nova-compute services fail to start after adding a given flag, remove that flag and restart the service to bring it back online before moving forward with adding another flag.

8.

Review and double check the verified common flags to make sure nothing was missed throughout the process.

9.

Turn off the test OpenStack instance and turn it back on. Once it is online again, execute the lscpu command again comparing the results:

```
$ ssh <instance-name-or-ip> lscpu | grep Flags | awk -F"Flags:" '{print $2}' | sed s/" "/"\n"/g | sort -u | sed '/./!d' > instance.flags-lscpu.sorted
```

10.

At this point we assume that all the flags have been applied and nova-compute services have been restarted on all source and target hypervisors.

To make sure everything works, attempt to live-migrate the test

instance between source and target hypervisors and perform validations to make sure the migrations were successful. For more information, see Live-migrate instances. At your discretion, add variations to your tests by running appropriate workloads inside the test instance while the live migrations are being executed.

A third or more Source processors can be added to the set of processors to be migrated between by repeating the steps using the verified common flags file as the Source and the third processor as the Target resulting in a refined common flags file.

SUMMARY

Linux KVM enables the live migration of application processes between different processor generations and families by limiting application use to a common set of x86-64 instructions (flags). The process for determining that common flag set between Source and Target servers is presented. More than two different processor Families can be migrated between if the common flag set is built cumulatively for each Target added.