Openstack CVIMでのSRIOVネットワークを使用したDNS VNFの導入：Prime Network Registrar(DNS)の設定例

内容

はじめに

このドキュメントでは、SR-IOVとアクティブバックアップボンディングを使用して、OpenStack Cisco Virtualized Infrastructure Manager(CVIM)にCPNRを段階的に導入する方法について説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- OpenStackとシングルルート入出力(SR-IOV)の概念に精通していること
- Cisco Virtual Interface Manager(VIM)とCisco Elastic Services Controller、およびLinuxのコマンドとネットワーキングに関する実務知識

使用するコンポーネント

このドキュメントの内容は、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。稼働中のネットワークで作業を行う場合、コマンドの影響について十分に理解したうえで作業してください

背景説明

現在のネットワーキング環境では、仮想ネットワーク機能(VNF)は、俊敏でスケーラブルな効率的なネットワークサービスを実現するうえで重要な役割を果たします。高性能なネットワーク接続を必要とするVNFでは、SR-IOVがよく使用されるテクノロジーです。SR-IOVにより、VNFはハイパーバイザ仮想スイッチをバイパスし、物理ネットワークインターフェイスコントローラ(NIC)リソースに直接アクセスできるため、遅延が減少し、スループットが向上します。

コンフィギュレーション

導入を進める前に、次の前提条件が満たされていることを確認してください。

1. ハードウェア要件

- SR-IOV対応NIC:
  - BIOS/Unified Extensible Firmware Interface(UEFI)でSR-IOVが有効になっている、少なくとも2つのSR-IOV対応物理NIC。
  - 例：sriov0がNon-Uniform Memory Access(NUMA)ノード0にマッピングされ、sriov1がNUMAノード1にマッピングされる。

- NUMA対応ホスト：
  - コンピューティングノードはNUMAアーキテクチャをサポートする必要があります。
  - ホストBIOS/UEFIでNUMAサポートを有効にする必要があります。

2. Intel NICカードの識別

インテルXL710およびE810CQDA2 NICカードは、高パフォーマンスのSR-IOVネットワーキングに広く使用されています。ホストのNICカードモデルを確認するには、次の手順を参照してくだ

さい。

ステップ 1：lspciコマンドの使用

ネットワークコントローラに関連するPeripheral Component Interconnect(PCI)デバイスをリストするには、次のコマンドを実行します。

```
lspci | grep -i ethernet
```

出力例：

```
81:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
82:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 03)
```

ステップ 2：XL710の確認

NICがIntel XL710の場合は、出力にイーサネットコントローラXL710と表示されます。

ステップ 3：E810CQDA2の確認

NICがIntel E810CQDA2の場合は、出力に「Ethernet Controller E810-Cin」と表示されます。

ステップ 4：ドライバのサポートの確認

使用中のNICドライバを確認するには、次のコマンドを実行します。

```
ethtool -i
```

XL710の出力例：

```
driver: i40e
version: 2.13.10
```

E810CQDA2の出力例：

```
driver: ice
version: 1.7.12
```

ドライバのバージョンがOpenStackおよびLinuxディストリビューションの互換性マトリックスと一致していることを確認します。

3. BIOS/UEFI設定

- SR-IOVを有効にします。

サーバのBIOS/UEFIでSR-IOVが有効になっていることを確認します。

- Virtualization Technology for Directed I/O(VT-d)/AMD-Viを有効にします。

PCIパススルーおよびSR-IOV機能を使用するには、Intel VT-dまたはAMD-Viを有効にする必要があります。

4. OpenStackのセットアップ

- コアOpenStackサービス：

Nova、Neutron、Glance、KeystoneなどのOpenStackサービスがインストールされ、設定されていることを確認します。

- Neutron設定：

Neutronは、オーケストレーション/管理ネットワーク用のOpenvswitch(OVS)と、アプリケーション/サービスネットワーク用のSR-IOVの両方をサポートする必要があります。

- SR-IOV設定：

コンピューティングノードは、SR-IOVをサポートし、NIC上に仮想機能(VF)を作成するように設定する必要があります。

5. Cisco Prime Network Registrar(CPNR)VNFイメージ

- VNFイメージ互換性：

CPNR VNFイメージはSR-IOVインターフェイスをサポートし、必要なドライバを含む必要があります。

- 概要へのアップロード：

CPNR VNFイメージがOpenStack Glanceで使用できることを確認します。

6. 管理アクセス

- OpenStack CLIの場合：

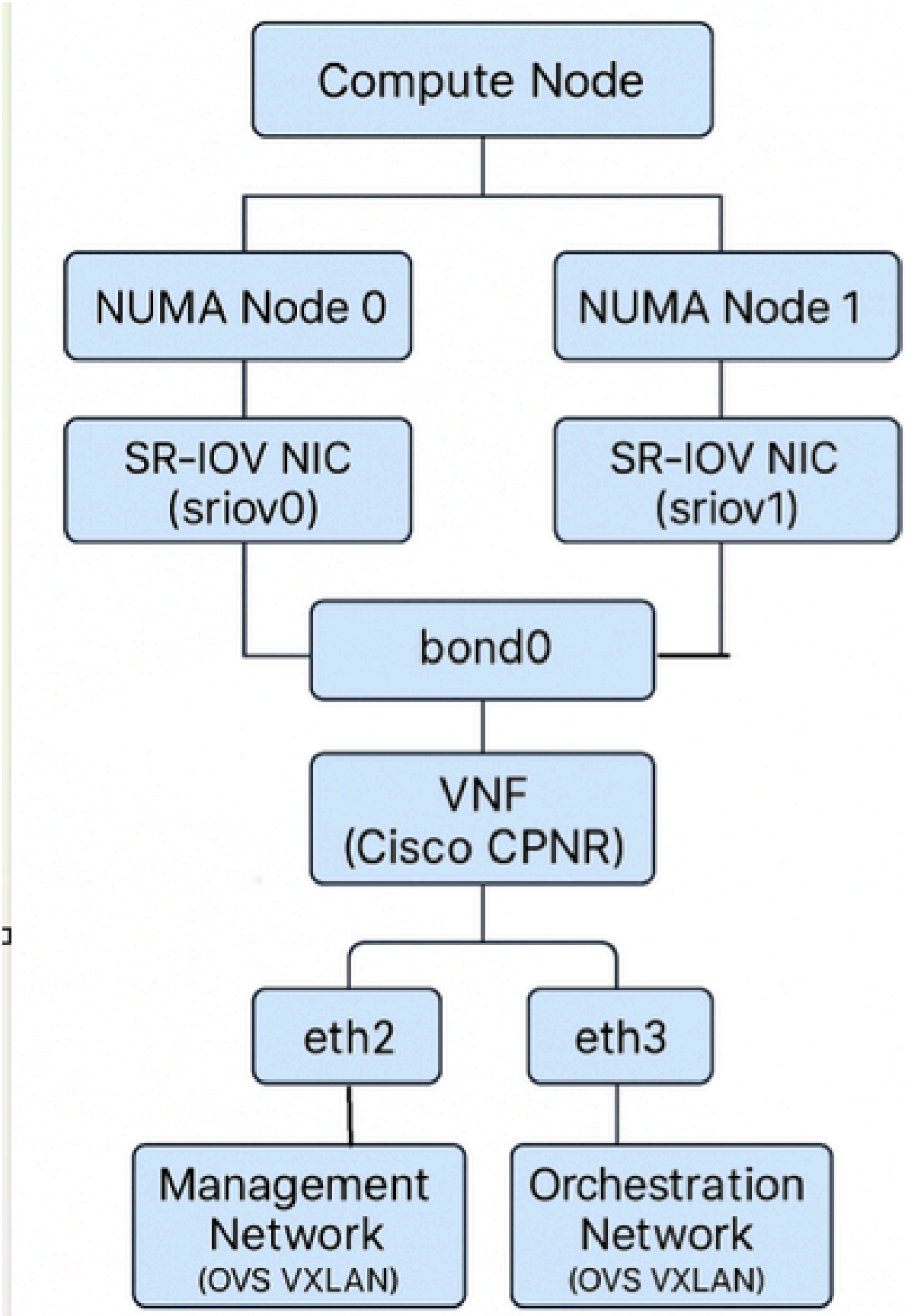
ネットワークの作成、フレーバー、およびVNFの起動を行うために、OpenStack CLIにアクセスできることを確認します。

- rootまたはadmin権限：

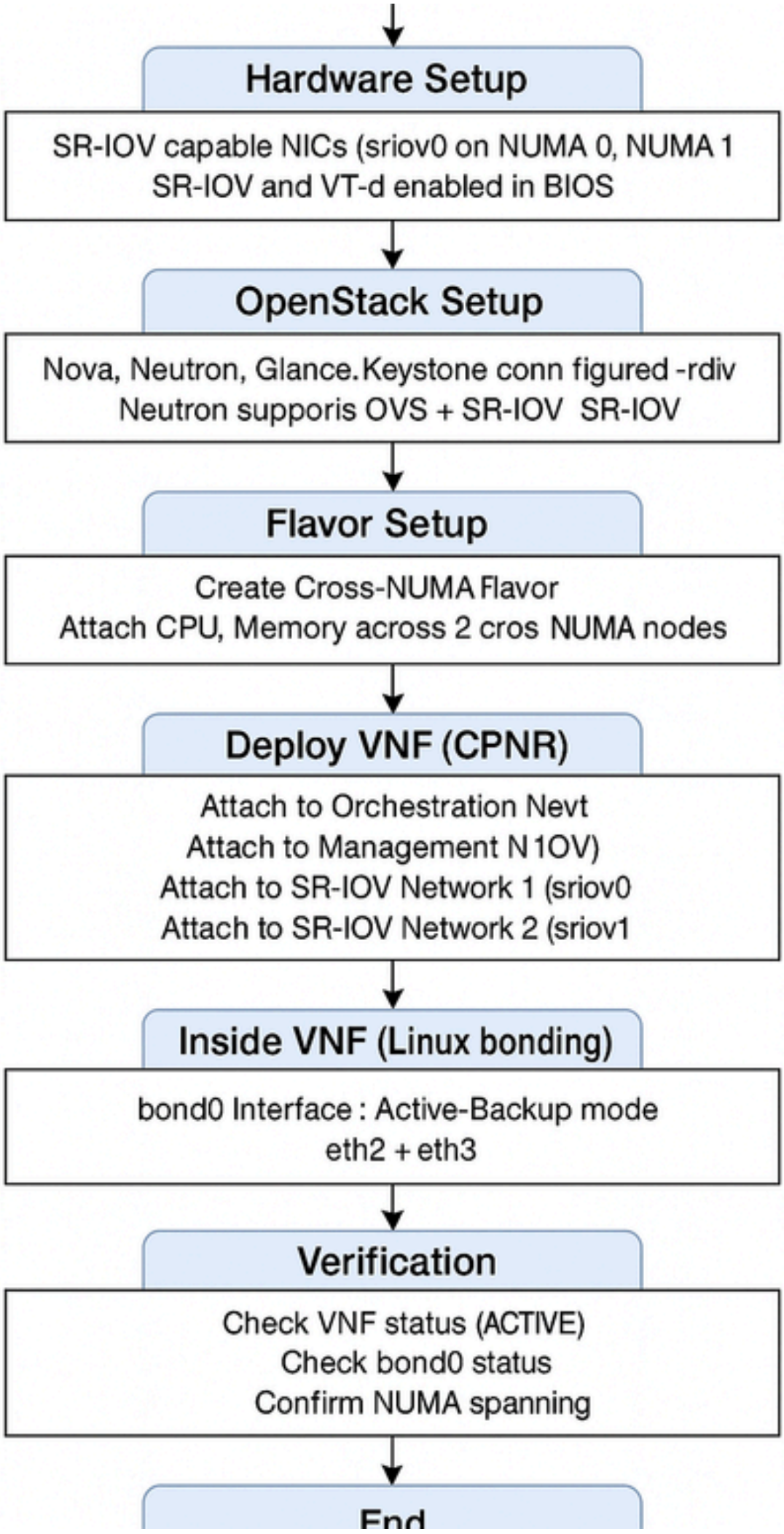
Linuxホスト上およびVNF内でネットワークを設定するためのルートまたは管理アクセス。

アーキテクチャ概要

VNFネットワークインターフェイスの接続図


Compute Node
NUMA Node 0
NUMA Node 1
SR-IOV NIC
(sriov0)
SR-IOV NIC
(sriov1)
bond0
VNF
(Cisco CPNR)
eth2
eth3
Management
Network
(OVS VXLAN)
Orchestration
Network
(OVS VXLAN)

フローチャート


Hardware Setup
SR-IOV capable NICs (sriov0 on NUMA 0, NUMA 1
SR-IOV and VT-d enabled in BIOS
OpenStack Setup
Nova, Neutron, Glance.Keystone conn figured -rdiv
Neutron supporis OVS + SR-IOV SR-IOV
Flavor Setup
Create Cross-NUMA Flavor
Attach CPU, Memory across 2 cros NUMA nodes
Deploy VNF (CPNR)
Attach to Orchestration Nevt
Attach to Management N1OV)
Attach to SR-IOV Network 1 (sriov0
Attach to SR-IOV Network 2 (sriov1
Inside VNF (Linux bonding)
bond0 Interface : Active-Backup mode
eth2 + eth3
Verification
Check VNF status (ACTIVE)
Check bond0 status
Confirm NUMA spanning
End

test-tenant

true

false

sriov-vm-deployment

sriov-vm-1-group

vim1

default

sriov-image

custom-flavor

300

30

REBOOT_ONLY

0

mgmt-net

192.168.10.101

1

direct

sriov-net-1

0

sriov-subnet-1

10.10.10.10

2

direct

sriov-net-2

0

sriov-subnet-2

10.10.20.10

1

1

false

```
--user-data



file://tmp/init/sriov-vm-1.cfg
```

要点

- <interface>の下のtype>direct</type>を使用して、そのNICのSR-IOV（PCIパススルー）をイネーブルにします。
- 各SR-IOVインターフェイスには、独自のネットワークとサブネットがあります。
- 必要に応じて、IPv4/IPv6を<addresses>で関連付けることができます。

Cisco ESC XMLで渡すDay0ファイルの例：

```
Content-Type: multipart/mixed; boundary="==========267839505026098033O=="
MIME-Version: 1.0`
```

```
--===============2678395050260980330==
MIME-Version: 1.0
Content-Type: text/cloud-boothook; charset="us-ascii"

#cloud-boothook
#!/bin/bash
if [ ! -f /etc/cloud/cloud.cfg.orig ]; then
cp /etc/cloud/cloud.cfg /etc/cloud/cloud.cfg.orig
cp /etc/cloud/cloud.cfg.norootpasswd /etc/cloud/cloud.cfg
fi

--===============2678395050260980330==
MIME-Version: 1.0
Content-Type: text/cloud-config; charset="us-ascii"

#cloud-config
hostname: cpnr
ssh_authorized_keys:
 - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC7pf8gvOWH/Zv8iAlTv6LWEiPGA3B6t96G6LwTHF6iXOqQxyIUkg8IkqZ6wNwx
 - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDAmkQGCZUrYqkZOC0J9t7mF9La9zYOqfzzFkk1wWtPga+aANOaFgjqbjj+VlBd
runcmd:
 - /usr/sbin/useradd FMLVL1 -d /home/FMLVL1 -s /bin/bash -g users; (/bin/echo changeme; /bin/echo chang
 - nmcli con add type ethernet con-name eth0 ifname eth0 ip4 10.xx.xx.xx/24
 - nmcli con add type ethernet con-name eth1 ifname eth1 ip4 172.xx.xx.xx/23
 - nmcli connection add type bond con-name bond0 ifname bond0 bond.options "mode=active-backup,miimon=1
 - nmcli connection add type ethernet ifname ens6 master bond0
 - nmcli connection add type ethernet ifname ens7 master bond0
 - nmcli con up eth0
 - nmcli con up eth1
 - nmcli con up bond-slave-ens6
 - nmcli con up bond-slave-ens7
 - nmcli con down bond0
 - nmcli con up bond0
 - nmcli connection reload
 - hostnamectl set-hostname CPNRDNS01CO

--===============2678395050260980330==
```

OpenStackでのSR-IOVポートとアクティブバックアップボンドインターフェイスを使用したCPNR VNFの導入

CPNRは、企業およびサービスプロバイダーのネットワークにIPアドレス管理(IPAM)、DHCP、およびドメインネームサーバ(DNS)サービスを提供する重要な仮想ネットワーク機能(VNF)です。OpenStackでVNFとしてCPNRを展開する場合、特に冗長性とパフォーマンスを確保するためにSR-IOVポート、クロスNUMA構成、およびアクティブバックアップボンドインターフェイスを活用する場合、慎重な計画が必要です。

この記事では、OpenStackにCPNR VNFを導入する手順を説明します。内容は以下を含みます。

- 複数のNUMAノードからSR-IOV NICにアクセスする場合に重要となる、NUMA間モードの設定。
- スイッチ側の設定を必要とせずにハイアベイラビリティを確保するアクティブバックアップボンディングの設定。
- OpenStackネットワーク、フレーバー、および概要の設定

- IPアドレスの計画、ifcfgファイルを使用したLinuxネットワークの設定、Cisco ESCを使用したVNFの導入

導入の主な機能

1. NUMA間の認識：

    - CPNR VNFはNUMAノードにまたがり、SR-IOV NIC(NUMA 0ではsriov0、NUMA 1ではsriov1)にアクセスします。
    - シングルNUMAモードでは、OpenStackではVNFが起動したのと同じNUMAノードに物理的に配置されているNICにのみVNFを接続できるため、クロスNUMAモードが必要です。NUMA間モードを有効にすることで、VNFは両方のNUMAノードからのNICとリソースを利用できます。

2. アクティブ/バックアップボンディング：

    - bond0インターフェイスはSR-IOV NICを使用して作成されます(sriov0からのeth2とsriov1からのeth3)。
    - アクティブバックアップモードでは、スイッチ側の設定を必要とせずに、冗長性と耐障害性を確保できます。

3. OpenStackネットワーキング：

    - オーケストレーションおよび管理ネットワーク：制御および管理トラフィック用のOpenVswitchベース。
    - アプリケーション/サービスネットワーク：高性能トラフィック用のSR-IOVベース。

クロスNUMAモードが必要な理由

1. OpenStackでのNUMA対応ネットワーキング

NUMAは、各CPU（およびそのローカルメモリとデバイス）がNUMAノードにグループ化されるメモリアーキテクチャです。OpenStackでは、NUMAに対応した配置により、遅延を最小限に抑え、パフォーマンスを最大化するために、VNFが同じNUMAノード上のリソースに最適に割り当てられます。

- SR-IOV NICはNUMAローカルです。
    - 各物理NICは特定のNUMAノードに関連付けられています。例：
        - sriov0はNUMAノード0に接続されています。
        - sriov1はNUMAノード1に接続されます。
- シングルNUMAモードの制限：
    - VNFがシングルNUMAモードで起動されると、OpenStackでは、VNFはVNFが起動さ

れたNUMAノードのローカルにあるNICにのみ接続できます。この場合、次を意味します。
    - VNFがNUMA 0で起動される場合、sriov0のNICにのみ接続できます。
    - VNFがNUMA 1で起動される場合、sriov1のNICにのみ接続できます。

2. クロスNUMAモードが必要な理由

CPNR VNFでは、次の機能にアクセスする必要があります。

- オーケストレーションネットワーク（Openvswitch、NUMA非依存）
- 管理ネットワーク（Openvswitch、NUMA非依存）
- SR-IOVネットワーク1：接続されたtosriov0（NUMAノード0）
- SR-IOVネットワーク2：接続されたtosriov1（NUMAノード1）。

この導入では、冗長性とハイアベイラビリティを提供するために、CPNR VNFはNUMA 0(sriov0)およびNUMA 1(sriov1)の両方からSR-IOV NICにアクセスする必要があります。これを実現するには、次の手順を実行します。

- VNFはクロスNUMAモードで起動する必要があります。これにより、OpenStackは複数のNUMAノードからCPU、メモリ、およびNICを割り当てることができます。
- これにより、VNFがsriov0およびsriov1上のNICに接続できるようになり、アクティブ/バックアップボンド設定で両方のSR-IOVポートを使用できるようになります。

OVSポートのConntrackサイズ制限

Conntrackとは

Conntrackは、特にネットワークアドレス変換(NAT)とファイアウォールルールのネットワーク接続を追跡するために使用されるLinuxカーネル機能です。OpenStackのOVSベースのポートでは、接続状態の管理とセキュリティグループルールの適用にconntrackが使用されます。

ConntrackがOVSポートに与える影響

1. Conntrackテーブル：

    - アクティブな各接続は、conntrackテーブルのエントリを1つ消費します。
    - conntrackテーブルのサイズはf_conntrack_maxparameterで制限されます。

2. デフォルトの制限：

    - デフォルトでは、conntrackテーブルのサイズは65536エントリです。接続レートが高いワークロード（たとえば、多数の同時フローがあるVNF）の場合、この制限はすぐに使い果たされ、パケットがドロップされる可能性があります。

3. OVSポートへの影響：

- conntrackテーブルがいっぱいになると、新しい接続がドロップされ、VNFのパフォーマンスに重大な影響を与える可能性があります。
- これは、OVSポートを使用するOrchestrationandManagementネットワークに特に関連します。

Conntrackの制限を緩和する方法

1. Conntrackテーブルのサイズを増やす：

   - 現在の制限を表示します。

     ```
     sysctl net.netfilter.nf_conntrack_max
     ```

   - 上限を上げる：

     ```
     sysctl -w net.netfilter.nf_conntrack_max=262144
     ```

   - 変更を永続的なものにします。

     ```
     echo "net.netfilter.nf_conntrack_max=262144" >> /etc/sysctl.conf
     ```

2. Monitor Conntrack Usage（コントローラの使用状況の監視）:

   conntrackの統計情報を確認します。

   ```
   cat /proc/sys/net/netfilter/nf_conntrack_count
   ```

3. セキュリティグループルールの最適化：

   OVSポートに適用されるルールの数を減らして、conntrackのオーバーヘッドを最小限に抑える。

SR-IOVによるconntrack問題の解決方法

1. Conntrackの依存関係の排除

SR-IOVポートは、OVSデータパスおよびconntrackなどのLinuxカーネル機能をバイパスします。

これにより、接続トラッキングのオーバーヘッドが完全に排除されます。

2. 高い拡張性

conntrackテーブルサイズ(nf_conntrack_max)に制限されるOVSポートとは異なり、SR-IOVポートは実質的に無制限の数の接続を処理できます。

3. 遅延の削減

パケット処理をNICハードウェアにオフロードすることで、SR-IOVポートはソフトウェアベースのconntrack処理による遅延を解消します。

CPNR VMのSR-IOVポートにアクティブバックアップモードが選択される理由

アクティブバックアップボンディングモードは、その単純さ、耐障害性、およびSR-IOVインターフェイスとの互換性により、この導入に特に適しています。その理由は次のとおりです。

1. 複雑さを伴わない冗長性

- アクティブバックアップモード：常に1つのインターフェイス(アクティブインターフェイス)だけがトラフィックを送受信します。他のインターフェイスはスタンバイモードのままです。
- リンク障害やハードウェアの問題などによってアクティブインターフェイスに障害が発生すると、bondは自動的にスタンバイインターフェイスに切り替わります。これにより、手動による介入を必要とせずに、継続的なネットワーク接続が保証されます。

2. リンクアグリゲーショングループ(LAG)は不要

- 他のボンディングモード(802.3adorbalance-albなど)とは異なり、アクティブバックアップモードではLink Aggregation Control Protocol(LACP)やスイッチ側の設定は必要ありません。
- SR-IOV VFは通常LACPまたはLAG設定をサポートしないため、これはSR-IOVポートにとって特に重要です。

3. シームレスなフェールオーバー

- フェールオーバーはほぼ瞬時に実行され、トラフィックの中断は最小限に抑えられます。
- アクティブインターフェイスに障害が発生すると、bondは即座にスタンバイインターフェイスをアクティブステータスに昇格させます。

4. ハードウェアの独立性

アクティブ – バックアップモードは、基盤となる物理スイッチやハードウェアとは独立して動作します。フェールオーバーロジックは完全にLinuxカーネル内に存在し、移植性と汎用性が高くなっています。

5. SR-IOV向けに最適化

SR-IOV VFは特定の物理NICとNUMAノードに関連付けられます。アクティブバックアップモードを使用すると、異なるNUMAノードからのVFを単一の論理ボンドインターフェイス(bond0)に結合できます。 これにより、NUMAリソースを効率的に使用しながら、高可用性を確保できます。

Active-Backupモードは、Linuxボンディングで最も簡単で広く使用されているモードの1つです。これは、結合インターフェイスの1つに障害が発生しても、トラフィックがシームレスに流れ続けることを保証することによって、ハイアベイラビリティを提供するように設計されています。 ここでは、アクティブ – バックアップモードの仕組み、主な特性、および利点について詳しく説明します。

Linux Bond Interfaceとは何ですか？

Linuxのbond interfaceは2つ以上のネットワークインターフェイスを組み合わせて1つの論理インターフェイスにします。この論理インターフェイスはbond(bond0など)と呼ばれ、次の機能を提供するために使用されます。

- 冗長性：ネットワーク接続の高可用性を確保します。
- パフォーマンスの向上：他のモード(balance-rror802.3adなど)でも、帯域幅を集約できます。

アクティブ/バックアップモードの仕組み

アクティブバックアップモードでは、トラフィックの送受信に常に1つのインターフェイス(アクティブインターフェイスと呼ばれる)だけが使用されます。他のインターフェイスはスタンバイモードのままです。アクティブインターフェイスに障害が発生すると、スタンバイインターフェイスの1つがactiveステータスに格上げされ、トラフィックは自動的に新しいアクティブインターフェイスに再ルーティングされます。

アクティブ/バックアップモードの主な機能

1. 単一のアクティブインターフェイス：

    - 任意の時点で、bond内の1つの物理インターフェイスのみがトラフィックの送受信にアクティブです。
    - スタンバイインターフェイスは、フェールオーバーが発生しない限り、完全にパッシブです。

2. 自動フェールオーバー：

    - アクティブインターフェイスに障害が発生した場合（ハードウェアの問題、ケーブルの切断、リンク障害など）、bondは自動的にスタンバイインターフェイスに切り替わります。

- フェールオーバーはシームレスであり、手動による介入は必要ありません。

3. フェールバックサポート：

   障害が発生したインターフェイスが復旧すると、自動的に再びアクティブになるか（そのように設定されている場合）、またはボンディングの設定に応じてスタンバイモードのままになります。

4. スイッチ側の要件なし：

   - 他のボンディングモード(802.3adorbalance-rrなど)とは異なり、アクティブバックアップモードでは物理スイッチ（LAGやLACP）に特別な設定は必要ありません。
   - これは、スイッチ側の設定が不可能なシナリオや、LAGをサポートしないSR-IOV仮想機能をボンディングする場合に最適です。

5. モニタリング:

   - bondは`semimon`(Media Independent Interface Monitor)パラメータを使って、すべてのメンバーのインターフェースの状態を継続的に監視します。
   - リンク障害が検出されると、ボンドは即座に健全なスタンバイインターフェイスに切り替わります。

アクティブ/バックアップモードでのトラフィックフロー

正常な動作

1. アクティブなインターフェイス：

   - トラフィックはアクティブインターフェイスのみを通過します(たとえば、bond ofeth2およびdeth3のeth2)。
   - スタンバイインターフェイス(eth3)はアイドル状態のままであり、トラフィックの送受信は行われません。

2. モニタリング:

   - bondは、すべてのメンバーインターフェイスのステータスを定期的に監視します。これを行うには、次のコマンドを使用します。
     - miimon：設定可能な間隔（たとえば100ミリ秒ごと）で、各インターフェイスのリンクステータスをチェックします。
     - Address Resolution Protocol(ARP)モニタリング（オプション）:ARP要求を送信して、アクティブなインターフェイスが到達可能であることを確認します。

フェールオーバーシナリオ

1. アクティブインターフェイスのリンク障害：

アクティブなインターフェイス(eth2)に障害が発生した場合（たとえば、ケーブルが抜けた、NICハードウェアに障害が発生した、またはリンクダウンした場合）、bondはmiimonorARPモニタリングを使用して障害を即座に検出します。

2. 自動フェールオーバー：

   - bondがスタンバイインターフェイス(eth3)に切り替わり、これが新しいアクティブインターフェイスになります。
   - 手動による介入を必要とせずに、新しいアクティブインターフェイス経由でトラフィックが再ルーティングされる

3. フェールオーバーの適時性：

   フェールオーバープロセスはほぼ瞬時に実行されます(時間間隔によっては、通常は数ミリ秒以内)。

フェールバックシナリオ

1. 障害が発生したインターフェイスの復元：

   - 以前に障害が発生していたインターフェイス(eth2)が回復した場合、次のことが可能です。
     - アクティブなロールを自動的に再要求します（再要求するように設定されている場合）。
     - スタンバイモードのままにする（デフォルトの動作）。

2. トラフィック継続性：

   フェールバックはシームレスで、継続的なトラフィックフローの中断を回避します。

使用例：SR-IOVポートとのアクティブバックアップボンディング

アクティブバックアップモードは、次の理由から、SR-IOVインターフェイスに特に適しています。

- SR-IOV VFは通常、LACPのようなリンクアグリゲーションプロトコルをサポートしません。
- アクティブバックアップモードのbondは、スイッチ側の設定なしで冗長性を提供できます。

例：

- eth2はSR-IOV VF onsriov0（NUMAノード0）にマッピングされます。
- eth3はSR-IOV VF onsriov1（NUMAノード1）にマッピングされます。
- bond(bond0)は、これらのインターフェイスを組み合わせて、SR-IOV VF間のシームレスな

フェールオーバーを提供します。

ステップ 1：OpenStackネットワーキング

CPNR VNFには、次の4つのネットワークが必要です。

1. オーケストレーションネットワーク：制御およびオーケストレーショントラフィック（Openvswitchベース）。
2. 管理ネットワーク：管理アクセス用（Openvswitchベース）。
3. SR-IOVネットワーク1:sriov0のアプリケーション/サービストラフィック。
4. SR-IOVネットワーク2:sriov1のアプリケーション/サービストラフィック。

ステップバイステップの導入：

ステップ 1.1：Openvswitchネットワークの作成

- オーケストレーションネットワーク：

```
openstack network create --provider-network-type vxlan orchestration-network
```

- 管理ネットワーク：

```
openstack network create --provider-network-type vxlan management-network
```

ステップ 1.2：Openvswitchネットワークのサブネットの作成

- オーケストレーションサブネット：

```
openstack subnet create --network orchestration-network \
--subnet-range 192.168.100.0/24 orchestration-subnet
```

- 管理サブネット：

```
openstack subnet create --network management-network \
```

```
--subnet-range 10.10.10.0/24 management-subnet
```

ステップ 1.3：SR-IOVネットワークの作成

- SR-IOVネットワーク1:

```
openstack network create --provider-network-type vlan \
--provider-physical-network sriov0 --provider-segment 101 sriov-network-1
```

- SR-IOVネットワーク2:

```
openstack network create --provider-network-type vlan \
--provider-physical-network sriov1 --provider-segment 102 sriov-network-2
```

ステップ 2：OpenStackの種類

ステップ 2.1：クロスNUMAフレーバーの作成

VNFが両方のNUMAノードからSR-IOV NICにアクセスできるようにするには、クロスNUMAをサポートするフレーバーを作成します。

```
openstack flavor create --ram 8192 --vcpus 4 --disk 40 cross-numa-flavor
```

ステップ 2.2：NUMAプロパティの構成

NUMA固有のプロパティを設定する：

```
openstack flavor set cross-numa-flavor \
--property hw:numa_nodes=2 \
--property hw:cpu_policy=dedicated \
--property hw:mem_page_size=large
```

ステップ 3：アクティブバックアップモードでのボンディングの設定

VNFを起動した後、VNF上のSR-IOVポート(eth2およびeth3)のbondインターフェイスを設定します。

ステップ 3.1：Bondインターフェイスの設定

アクティブバックアップモードでbondインターフェイス(bond0)を作成します。

```
vi /etc/sysconfig/network-scripts/ifcfg-bond0
```

```
DEVICE=bond0
BOOTPROTO=static
ONBOOT=yes
BONDING_OPTS="mode=active-backup miimon=100"
IPADDR=172.16.1.10
NETMASK=255.255.255.0
GATEWAY=172.16.1.1
```

ステップ 3.2：スレーブインターフェイスの設定

- eth2:

  ```
  vi /etc/sysconfig/network-scripts/ifcfg-eth2
  ```

  ```
  DEVICE=eth2
  ONBOOT=yes
  MASTER=bond0
  SLAVE=yes
  ```

- eth3:

  ```
  vi /etc/sysconfig/network-scripts/ifcfg-eth3
  ```

```
DEVICE=eth3
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

ステップ 3.3：設定の適用

設定を適用するには、ネットワークサービスを再起動します。

```
systemctl restart network
```

確認

VNFを導入した後、次の手順を使用してその機能を確認します。

1. VNFステータスの確認

VNFインスタンスがアクティブであることを確認します。

```
openstack server show cpnr-instance
```

ステータスがACTIVEであることを確認します。

2. ネットワーク接続の確認

- pingテスト：VNFがすべてのネットワークで通信できることを確認します。

```
ping
```

```
ping
```

- 結合インタフェース：
    - bond0がアクティブであることを確認します。

```
cat /proc/net/bonding/bond0
```

チェック ポイント：

- 現在アクティブなスレーブ：アクティブなインターフェイスを示します。
- スレーブインターフェイス：botheth2およびth3が結合の一部であることを確認します。

3. NUMA配置の確認

VNFが両方のNUMAノードのリソースを使用していることを確認します。

```
nova show
        --human | grep numa
```

ベスト プラクティス

- モニタリングとトラブルシューティング：SR-IOVインターフェイスを監視するには、ツールliketcpdumpandethtooltoolを使用します。
- セキュリティ：物理ネットワークへのアクセスを慎重に管理し、テナント間の厳密な分離を実施します。
- スケーリング：使用可能なVFの数はNICハードウェアによって制限されるため、SR-IOV導入をスケーリングする場合は物理NIC容量を計画します。

トラブルシュート

展開が期待どおりに動作しない場合は、次のトラブルシューティング手順を参照してください。

1. SR-IOV設定の確認

- SR-IOVがBIOSで有効になっているかどうかを確認します。

```
dmesg | grep -i "SR-IOV"
```

- NIC上でVFが作成されたことを確認します。

```
lspci | grep Ethernet
```

2. NUMA配置の確認

VNFが両方のNICにアクセスできない場合は、NUMA間モードが有効になっていることを確認します。

- フレーバーのNUMAプロパティを確認してください：

```
openstack flavor show cross-numa-flavor
```

3. Bondインターフェイスの問題

- 債券ステータスをチェックします。

```
cat /proc/net/bonding/bond0
```

- ボンドが機能していない場合：
  - スレーブインターフェイス(eth2およびeth3)がbondの一部として正しく設定されていることを確認します。
  - ネットワークサービスを再起動します。

```
systemctl restart network
```

4. ネットワーク接続の問題

- OpenStackポートバインディングを確認します。

```
openstack port list --server cpnr-instance
```

- VNF内で正しいIP設定を確認します。

```
ip addr show
```

結論

SR-IOVポートを使用してOpenStackにCPNR VNFを導入する場合、VNFが両方のNUMAノードからNICに接続できるように、NUMAをまたぐモードにする必要があります。OpenStackでは、VNFが起動されるNUMAノード内のリソース（NIC、CPU、メモリ）にのみアクセスするようにシングルNUMAモードでVNFが制限されるため、これが不可欠です。NUMA間モードとアクティブバックアップボンディングを組み合わせることで、高可用性、耐障害性、および効率的なリソース使用率が保証され、この導入の耐障害性とパフォーマンスが向上します。

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。