



Red Hat OpenStack Platform 11

ネットワーク機能仮想化 (NFV) のプランニング および前提条件ガイド

Red Hat OpenStack Platform での NFV のプランニング

Red Hat OpenStack Platform 11 ネットワーク機能仮想化 (NFV) のプランニングおよび前提条件ガイド

Red Hat OpenStack Platform での NFV のプランニング

OpenStack Team
rhos-docs@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドには、NFV 対応の Red Hat OpenStack Platform 環境を設定/インストールする前に検討すべき重要なプランニング情報を記載します。

目次

第1章 はじめに	3
第2章 ソフトウェア要件	4
2.1. NFV デプロイメントでサポートされている構成	4
2.2. サポートされているドライバー	4
2.3. サードパーティー製のソフトウェアとの互換性	4
第3章 ハードウェア要件	5
3.1. サポートされている NIC	5
3.2. NUMA ノードのトポロジについての理解	5
3.3. LSTOPO を使用した NUMA の詳細情報の取得	6
第4章 ネットワークの考慮事項	7
第5章 SR-IOV デプロイメントのプランニング	8
5.1. NFV SR-IOV デプロイメント向けのハードウェアの分割	8
5.2. NFV SR-IOV デプロイメントのトポロジ	8
5.2.1. HCI を使用しない NFV SR-IOV	9
5.2.2. HCI の NFV SR-IOV	10
第6章 OVS-DPDK デプロイメントのプランニング	12
6.1. OVS-DPDK での CPU 分割と NUMA トポロジの使用方法	12
6.2. OVS-DPDK のパラメーターについての理解	12
6.2.1. CPU パラメーター	13
6.2.2. メモリーパラメーター	14
6.2.3. ネットワークパラメーター	15
6.2.4. その他のパラメーター	15
6.3. 2 NUMA ノード構成の OVS-DPDK デプロイメントの例	16
6.4. NFV OVS-DPDK デプロイメントのトポロジ	17
第7章 パフォーマンス	20
第8章 その他の参考資料	21

第1章 はじめに

ネットワーク機能仮想化 (NFV: Network Functions Virtualization) とは、通信事業者 (CSP) が従来のプロプライエタリーハードウェアの範囲を超えて、効率性および敏捷性を向上させるのに役立つソフトウェアベースのソリューションです。

NFV の概念に関する俯瞰的な情報は、[『ネットワーク機能仮想化 \(NFV\) の製品ガイド』](#)を参照してください。

Red Hat OpenStack Platform 11 director での SR-IOV および OVS-DPDK 設定に関する情報は [『ネットワーク機能仮想化 \(NFV\) の設定ガイド』](#)を参照してください。

第2章 ソフトウェア要件

本項では、サポートされている設定とドライバー、および NFV に必要なサブスクリプションの詳細について説明します。

Red Hat OpenStack Platform 11 をインストールするには、OpenStack 環境にある全システムを Red Hat サブスクリプションマネージャーで登録して、必要なチャンネルをサブスクライブします。詳しくは、「[システムの登録](#)」を参照してください。

2.1. NFV デプロイメントでサポートされている構成

Red Hat OpenStack Platform 11 は、director を使用した SR-IOV および OVS-DPDK のインストール向けの NFV デプロイメントをサポートしています。Red Hat OpenStack Platform 11 director で利用可能なコンポーザブルロールを使用して、カスタムのデプロイメントロールを作成できます。今回のリリースではサポートが制限されていますが、ハイパーコンバージドインフラストラクチャー (HCI) が提供されており、分散型の NFV 向けにコンピューターードと Red Hat Ceph Storage ノードを同じ場所に配置することができます。HCI のパフォーマンスを向上するために、CPU ピニングを使用します。HCI モデルでは、NFV のユースケースにおいてより効率的な管理を行うことができます。また、今回のリリースでは、テクノロジープレビュー機能として OpenDaylight および Real-Time KVM が提供されています。OpenDaylight とは、Software-Defined Network (SDN) デプロイメントに向けた、オープンソースのモジュール型マルチプロトコルコントローラーです。テクノロジープレビューとして提供されている機能のサポート範囲に関する詳しい情報は、「[テクノロジープレビュー機能のサポート範囲](#)」を参照してください。

2.2. サポートされているドライバー

サポートされるドライバーの完全な一覧は「[Red Hat OpenStack Platform におけるコンポーネント、プラグイン、およびドライバーのサポート](#)」を参照してください。

Red Hat OpenStack の NFV デプロイメント向けに検証済みの NIC の一覧は、「[サポートされている NIC](#)」を参照してください。

2.3. サードパーティー製のソフトウェアとの互換性

Red Hat のテクノロジー (Red Hat OpenStack Platform) で機能することを検証、サポート、認定済みの製品およびサービスの完全な一覧は、[Red Hat OpenStack Platform と互換性のあるサードパーティー製のソフトウェア](#) の情報を参照してください。製品バージョンやソフトウェアカテゴリー別に一覧をフィルタリングすることができます。

Red Hat のテクノロジー (Red Hat Enterprise Linux) で機能することを検証、サポート、認定済みの製品およびサービスの完全な一覧は、[Red Hat Enterprise Linux と互換性のあるサードパーティー製のソフトウェア](#) の情報を参照してください。製品バージョンやソフトウェアカテゴリー別に一覧をフィルタリングすることができます。

第3章 ハードウェア要件

本項では、NFV に必要なハードウェアの詳細情報を記載します。

「[Red Hat Technologies Ecosystem](#)」を使用し、カテゴリーを選んでから製品バージョンを選択して、認定済みハードウェア、ソフトウェア、クラウドプロバイダー、コンポーネントの一覧を確認してください。

Red Hat OpenStack Platform の認定済みハードウェアの完全一覧については「[Red Hat OpenStack Platform certified hardware](#)」を参照してください。

3.1. サポートされている NIC

以下のハードウェアは、Red Hat OpenStack Platform 11 の NFV デプロイメントで動作することをテストおよび承認済みです。

SR-IOV

Red Hat は SR-IOV 10G Mellanox、Qlogic および Intel カードをサポートします。ネットワークアダプターに関する詳しい情報は「[Network Adapter Feature Support for Red Hat Enterprise Linux](#)」を参照してください。

OVS-DPDK

Red Hat は Intel 10G ポート上で OVS-DPDK をサポートします (NIC: Dual/4 for port Intel X520)。詳しい情報は「[Intel Network Adapter Technical Specifications](#)」を参照してください。

3.2. NUMA ノードのトポロジーについての理解

NFV デプロイメントを計画する際には、コンピュータノードの NUMA トポロジーを理解した上で CPU とメモリーのリソースを分割し、パフォーマンスを最適化する必要があります。NUMA の情報を特定するには、以下の手順に従ってください。

1. コンピュータノード上の NUMA ノード数を確認します。

```
# ls /sys/devices/system/node/node* -d
/sys/devices/system/node/node0 /sys/devices/system/node/node1
```

2. NUMA ノード 0 上の論理 CPU の一覧を取得します。

```
# cd /sys/devices/system/node/node0
# ls cpu* -d
cpu0  cpu10  cpu12  cpu14  cpu16  cpu18  cpu2   cpu21  cpu4   cpu45
cpu47  cpu49  cpu50  cpu52  cpu54  cpu56  cpu58  cpu6   cpu61  cpu63
cpu65  cpu8   cpu1   cpu11  cpu13  cpu15  cpu17  cpu19  cpu20  cpu3
cpu44  cpu46  cpu48  cpu5   cpu51  cpu53  cpu55  cpu57  cpu59  cpu60
cpu62  cpu64  cpu7   cpu9   cpumap cpulist
```

3. 論理 CPU 0 の物理コアの ID を取得します。

```
# cat cpu0/topology/core_id
0
```

4. 論理 CPU 0 の CPU スレッドシブリングを取得します。

```
# cat cpu0/topology/thread_siblings_list
0,44
```

5. DPDK インターフェイスとして使用されているインターフェイス (ens801f0) の NUMA ノード番号を取得します。

```
# cat /sys/class/net/ens801f0/device/numa_node
1
```

6. 2 番目の DPDK インターフェイス (enp12s0f0) の NUMA ノード番号を取得します。

```
# cat /sys/class/net/enp12s0f0/device/numa_node
0
```

3.3. **lstopo** を使用した **NUMA** の詳細情報の取得

lstopo を使用してコンピュータノードの NUMA メモリーノード、ソケット、共有キャッシュ、コア、同時マルチスレッディングをグラフィカル表示することも可能です。**lstopo** は、キャッシュやメモリーの情報、ネットワークインターフェイスのローカリティーなどのシステム属性も収集します。詳しい情報とユースケース例は、[「Portable Hardware Locality \(hwloc\)」](#) を参照してください。

グラフィカルモードのリモートサーバーで **lstopo** を実行してコアをチェックするには、以下のステップを実行します。

1. **hwloc** パッケージをインストールします。

```
# yum install hwloc hwloc-gui hwloc-devel hwloc-debuginfo xorg-x11-xauth
```

2. コンピュータサーバーに接続します。

```
# ssh -X
```

3. **qemu** のプロセス ID を確認します。

```
# ps aux | grep qemu
```

4. コアのグラフィカル表示を取得します。

```
# lstopo --pid qemu-pid --whole-system
```

第4章 ネットワークの考慮事項

アンダークラウドのホストには、最低でも以下のネットワークが必要です。

- プロビジョニングネットワーク: オーバークラウドで利用できるベアメタルシステムの検出に役立つ DHCP および PXE ブート機能を提供します。
- 外部ネットワーク: 全ノードへのリモート接続に使用する別個のネットワーク。このネットワークに接続するこのインターフェースには、静的または外部の DHCP サービス経由で動的に定義された、ルーティング可能な IP アドレスが必要です。

最小のオーバークラウドの構成は、以下のとおりです。

- シングル NIC 構成: ネイティブの VLAN および異なる種別のオーバークラウドネットワークのサブネットを使用するタグ付けされた VLAN 上にプロビジョニングネットワーク用の NIC を 1 つ。
- デュアル NIC 構成: プロビジョニングネットワーク用の NIC を 1 つと、外部ネットワーク用の NIC を 1 つ。
- デュアル NIC 構成: ネイティブの VLAN 上にプロビジョニングネットワーク用の NIC を 1 つと、異なる種別のオーバークラウドネットワークのサブネットを使用するタグ付けされた VLAN 用の NIC を 1 つ。
- 複数 NIC 構成: 各 NIC は、異なる種別のオーバークラウドネットワークのサブセットを使用します。

ネットワーク要件の詳しい情報は「[ネットワーク要件](#)」を参照してください。

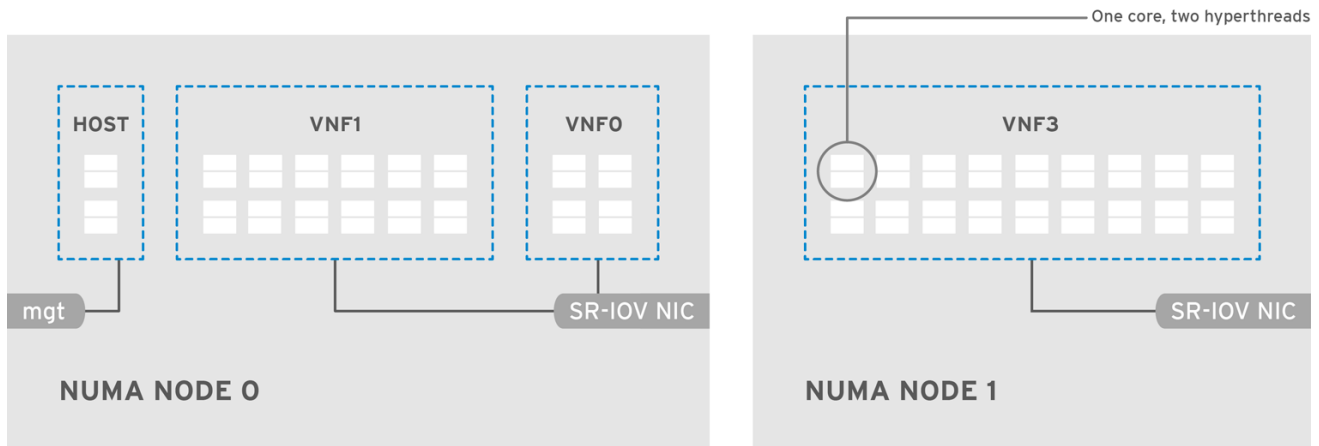
第5章 SR-IOV デプロイメントのプランニング

NFV 向けの SR-IOV デプロイメントを最適化するには、コンピュートノードのハードウェアに応じて、個別の OVS-DPDK パラメーターを設定する方法を理解しておく必要があります。

SR-IOV パラメーターに対するハードウェアの影響を評価するには、[「NUMA ノードトポロジーについての理解」](#) を参照してください。

5.1. NFV SR-IOV デプロイメント向けのハードウェアの分割

SR-IOV の場合は、高パフォーマンスを実現するためにホストとゲストの間でリソースを分割する必要があります。

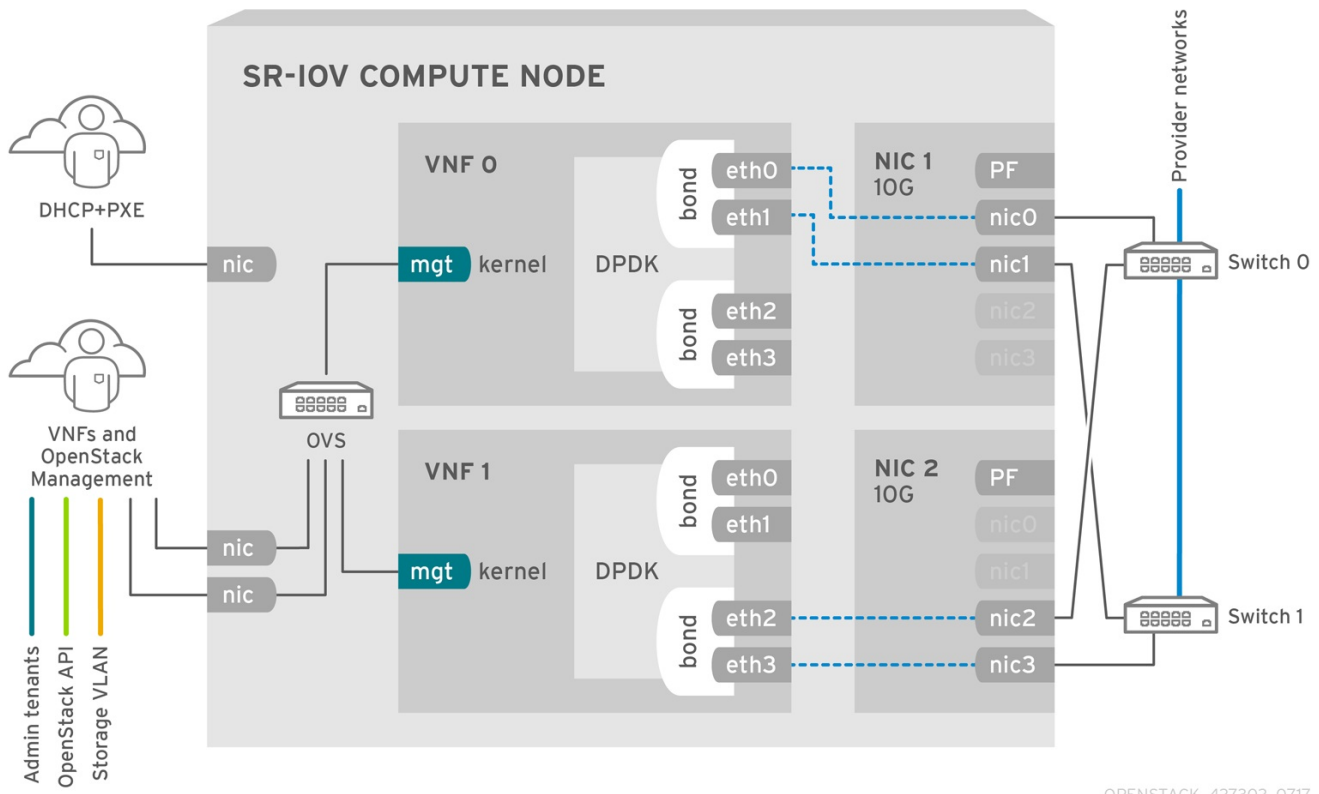


OPENSTACK_436587_0217

標準的なトポロジーでは、デュアルコアソケットのコンピュートノード上の NUMA ノードにはそれぞれ 18 のコアが実装されます。HT (ハイパースレッド) および非 HT のコアがサポートされています。各コアには 2 つのシプリングスレッドがあります。2 つのコアは、Open vSwitch を管理するためのホスト専用です。VNF は SR-IOV インターフェースのボンディングを処理します。すべての割り込み要求 (IRQ) はホストのコア上でルーティングされます。VNF コアは VNF 専用です。これらのコアは、他の VNF からの分離と、ホストからの分離を提供します。各 VNF は単一の NUMA ノード上のリソースを使用する必要があります。VNF によって使用される SR-IOV NIC はその同じ NUMA ノードに関連付ける必要もあります。このトポロジーでは、仮想化のオーバーヘッドはありません。ホスト、OpenStack Networking (neutron)、および Compute (nova) の設定パラメーターは単一のファイルで公開されるので、管理が簡単で、整合性を保つことができます。また、プリエンプションやパケット損失の原因となり、分離を適切に行うにあたって致命的となる一貫性の欠如を回避します。ホストと仮想マシンの分離は、**tuned** プロファイルに依存します。このプロファイルは、分離する CPU の一覧に基づいて、ブートパラメーターや OpenStack の変更を管理します。

5.2. NFV SR-IOV デプロイメントのトポロジー

以下の図には、2 つの VNF が示されています。各 VNF には、それぞれに **mgt** で示された管理インターフェースがあります。管理インターフェースは **ssh** アクセスなどを管理します。データプレーンインターフェースは VNF を DPDK にボンディングして、高可用性を確保します (VNF は DPDK ライブラリーを使用してデータプレーンインターフェースをボンディングします)。この図には、2 つの重複したプロバイダーネットワークも示されています。コンピュートノードには 2 つの標準 NIC がボンディングされ、VNF 管理と Red Hat OpenStack Platform API 管理の間で共有されています。

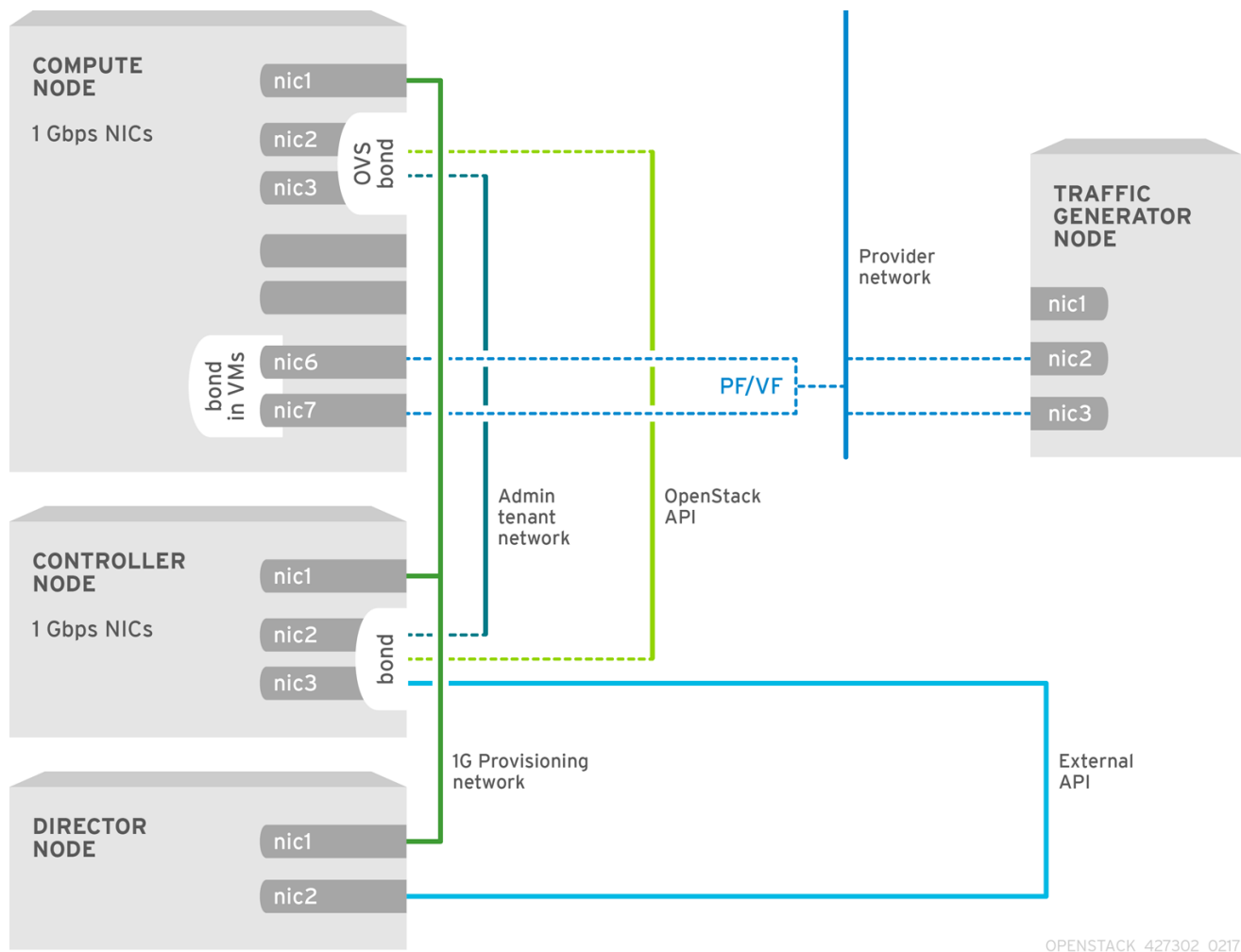


OPENSTACK_427302_0717

この図では、アプリケーションレベルで DPDK を活用し、SR-IOV VF/PF へのアクセスが可能な VNF を示しています。これらの両方を実装することにより、可用性またはパフォーマンスが向上します (ファブリックの設定に依存)。DPDK はパフォーマンスを向上させる一方、VF/PF DPDK のボンディングはフェイルオーバーに対応します (可用性)。VNF ベンダーは、DPDK PMD ドライバーが VF/PF として公開される SR-IOV カードを必ずサポートするようする必要があります。また、管理ネットワークは OVS を使用するので、VNF は標準の VirtIO ドライバーを使用する「mgmt」ネットワークデバイスを認識します。オペレーターは、VNF への初回の接続にそのデバイスを使用して、DPDK アプリケーションが 2 つの VF/PF を適切にボンディングすることができます。

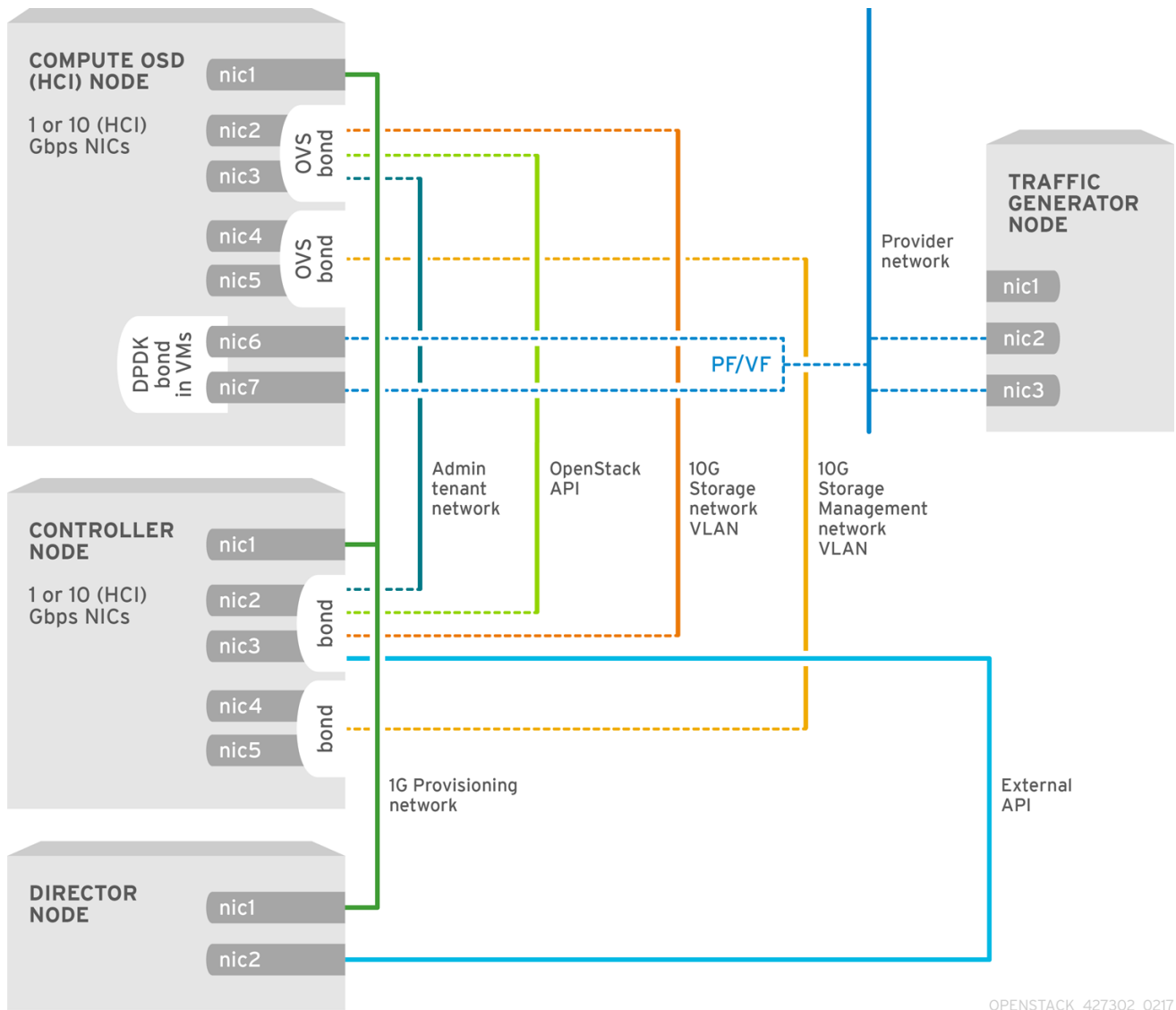
5.2.1. HCI を使用しない NFV SR-IOV

以下の図には、NFV ユースケース向けの非 HCI の SR-IOV のトポロジーを示しています。この環境は、1 Gbps の NIC を搭載したコンピュータノードおよびコントローラーノードと、director ノードで構成されます。



5.2.2. HCI の NFV SR-IOV

以下の図には、NFV ユースケース向けの HCI の SR-IOV のトポロジーを示しています。この環境は、1 Gbps または 10 Gbps の NIC を搭載した、HCI 対応の Compute OSD ノードおよびコントローラーノードと、director ノードで構成されます。



OPENSTACK_427302_0217

第6章 OVS-DPDK デプロイメントのプランニング

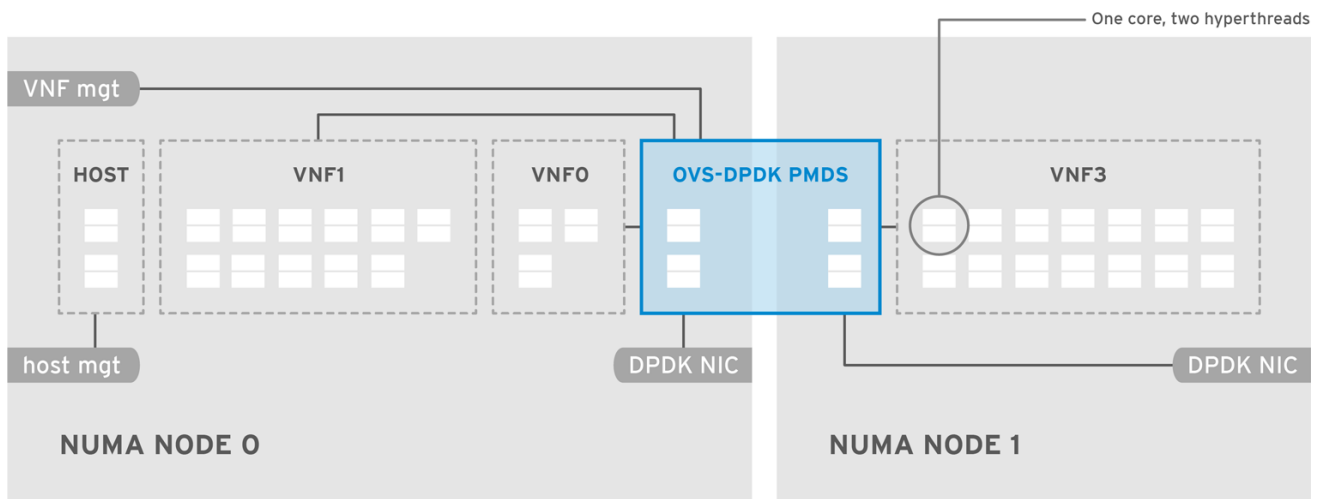
NFV 向けの OVS-DPDK デプロイメントを最適化するには、OVS-DPDK がコンピューターノードのハードウェア (CPU、NUMA ノード、メモリー、NIC) をどのように使用するかと、コンピューターノードに応じた OVS-DPDK の各パラメーターを決定するにあたっての考慮事項を理解しておくべきです。

CPU と NUMA トポロジーの概要は [「NFV Performance Considerations」](#) を参照してください。

6.1. OVS-DPDK での CPU 分割と NUMA トポロジーの使用方法

OVS-DPDK はホスト、ゲスト、および OVS-DPDK 自体用にハードウェアリソースを分割します。OVS-DPDK Poll Mode Driver (PMD) は、専用のコアを必要とする DPDK アクティブループを実行します。これは、CPU 一覧とヒュージページが OVS-DPDK で専用であることを意味します。

サンプルの分割では、デュアルソケットのコンピューターノード上の 1 NUMA ノードにつき 18 コアが含まれます。ホストと OVS-DPDK では NIC を共有できないので、このトラフィックには追加の NIC が必要です。



OPENSTACK_436587_0217

OVS-DPDK のパフォーマンスは、NUMA ノードにローカルなメモリーブロックの確保にも左右されます。メモリーと CPU ピニングに使用する同じ NUMA ノードに関連付けられた NIC を使用してください。また、ボンディングを構成する両方のインターフェースには、同じ NUMA ノード上の NIC を必ず使用してください。

6.2. OVS-DPDK のパラメーターについての理解

本項では、OVS-DPDK が director の `network_environment.yaml` HEAT テンプレート内のパラメーターを使用して CPU とメモリーを設定し、パフォーマンスを最適化する方法について説明します。この情報は、コンピューターノード上のハードウェアサポートと、そのハードウェアを分割して OVS-DPDK デプロイメントを最適化する最も有効な方法を評価するのに使用してください。



注記

CPU コアを割り当てる際には必ず、同じ物理コア上の CPU シブリングスレッド (論理 CPU) をペアにしてください。

コンピューターノード上の CPU と NUMA ノードを特定するには、[「NUMA ノードのトポロジーについて」](#)を参照してください。この情報を使用して、CPU と他のパラメーターをマッピングして、ホスト、ゲストインスタンス、OVS-DPDK プロセスのニーズに対応します。

6.2.1. CPU パラメーター

OVS-DPDK は以下の CPU 分割パラメーターを使用します。

NeutronDpdkCoreList

DPDK Poll Mode Driver (PMD) に使用する CPU コアを提供します。DPDK インターフェースのローカル NUMA ノードに関連付けられた CPU コアを選択します。**NeutronDpdkCoreList** は、Open vSwitch の **pmd-cpu-mask** の値に使用されます。

- シブリングスレッドをペアにします。
- **HostCpusList** からすべてのコアを除外します。
- 両方の NUMA ノード上の 1 番目の物理コアの論理 CPU が割り当てられないようにしてください。これらは、**HostCpusList** パラメーターに使用する必要があります。
- パフォーマンスは、この PMD コアリストに割り当てられている物理コアの数によって異なります。DPDK 用の NIC に関連付けられている NUMA ノードで、必要なコアを割り当てます。
- DPDK 用の NIC が 1 つある NUMA ノードの場合:
 - パフォーマンス要件に基づいて、必要な物理コア数を決定し、各物理コアに全シブリングスレッド (論理 CPU) を追加します。
- DPDK 用の NIC がない NUMA ノードの場合:
 - 1 つの物理コアのシブリングスレッド (論理 CPU) を割り当てます (NUMA ノードの 1 番目の物理コアを除く)。DPDK 用の NIC がない場合でも、ゲストインスタンス作成が失敗するのを回避するために、NUMA ノード上に最小限の DPDK Poll Mode Driver が必要です。

NovaVcpuPinSet

CPU ピニング用のコアを設定します。コンピュータノードは、ゲストインスタンスにこれらのコアを使用します。**NovaVcpuPinSet** は **nova.conf** ファイルの **vcpu_pin_set** 値として使用されます。

- **NeutronDpdkCoreList** および **HostCpusList** からすべてのコアを除外します。
- 残りのコアをすべて追加します。
- シブリングスレッドをペアにします。

HostIsolatedCpuList

ホストのプロセスから分離される CPU コアのセット。このパラメーターは、**tuned-profiles-cpu-partitioning** コンポーネント用の **cpu-partitioning-variable.conf** ファイルの **isolated_cores** 値として使用されます。

- **NeutronDpdkCoreList** と **NovaVcpuPinSet** のコア一覧が一致するようにします。
- シブリングスレッドをペアにします。

HostCpusList

handler および revalidator スレッドなどの、データパス以外の OVS-DPDK プロセス用の CPU コアを提供します。このパラメーターは、マルチ NUMA ノードハードウェア上でのデータパスの全体的

なパフォーマンスには影響は及ぼしません。このパラメーターは Open vSwitch の **dpdk-lcore-mask** 値に使用されます。

- 各 NUMA ノードから、1 番目のコア (およびスプリングスレッド) を割り当てます (NUMA に関連付けられている DPDK 用の NIC がいない場合も)。
- これらのコアは、**NeutronDpdkCoreList** および **NovaVcpuPinSet** のコアの一覧と相互に排他的である必要があります。

6.2.2. メモリーパラメーター

OVS-DPDK は、以下のメモリーパラメーターを使用します。

NeutronDpdkMemoryChannels

NUMA ノードごとに、CPU 内のメモリーチャネルをマッピングします。**NeutronDpdkMemoryChannels** パラメーターは Open vSwitch により **other_config:dpdk-extra="-n <value>"** 値として使用されます。

- **dmidecode -t memory** のコマンドで利用可能なメモリーチャネルの数を確認します。
- **ls /sys/devices/system/node/node* -d** のコマンドで NUMA ノードの数を確認します。
- 利用可能なメモリーチャネル数を NUMA ノード数で除算します。

NeutronDpdkSocketMemory

NUMA ノードごとにヒュージページプールから事前に割り当てるメモリーの容量を MB 単位で指定します。この値は、Open vSwitch により **other_config:dpdk-socket-mem** 値として使用されます。

- コンマ区切りリストで指定します。たとえば、DPDK PMD (**NeutronDpdkCoreList**) が 2 つの NUMA ノードを使用する場合には、**NeutronDpdkSocketMemory** の設定は **1024,1024** となります。
- DPDK 用の NIC が 1 つある NUMA ノードの場合:
 - この値は、**compute.yaml** HEAT テンプレートで定義されている DPDK インターフェースの MTU 値によって異なります。
 - NUMA ノードの値 = $(\text{PER_MTU} + 800) * (4096 * 64)$ 。1 GB 未満を切り上げます。
 - 800 はオーバーヘッドの値です。
 - $4096 * 64$ は mempool 内のパケット数です。
- DPDK 用の NIC がいない NUMA ノードの場合:
 - 静的な推奨値 1024 MB (1 GB) を使用します。

以下に例を示します。

- NUMA ノード 1 上に DPDK 用の NIC が 1 つあり MTU が 9000 の場合:

```
OvsDpdkSocketMemory: "1024,4096"
```

- NUMA ノード 0 上に DPDK 用の NIC が 1 つあり MTU が 9000 で、NUMA ノード 1 上に DPDK 用の NIC が 1 つあり MTU が 9000 の場合:

```
OvsDpdkSocketMemory: "4096,4096"
```

NovaReservedHostMemory

ホスト上のタスク用にメモリーを MB 単位で確保します。この値は、コンピュートノードにより `nova.conf` の `reserved_host_memory_mb` 値として使用されます。

- 静的な推奨値 4096 MB を使用します。

6.2.3. ネットワークパラメーター

NeutronDpdkDriverType

DPDK によって使用されるドライバーの種別を設定します。`vfio-pci` のデフォルト値を使用してください。

NeutronDatapathType

OVS ブリッジ用のデータパスの種別を設定します。DPDK は `netdev` のデフォルト値を使用してください。

NeutronVhostuserSocketDir

OVS 向けに vhost-user ソケットディレクトリーを設定します。vhost クライアントモード用の `/var/lib/vhost_sockets` を使用してください。

6.2.4. その他のパラメーター

NovaSchedulerDefaultFilters

要求されたゲストインスタンスに対してコンピュートノードが使用するフィルターの順序付きリストを指定します。

ComputeKernelArgs

コンピュートノードのブート時用に、複数のカーネル引数を `/etc/default/grub` に指定します。設定に応じて、以下のパラメーターを追加します。

- **hugepagesz**: CPU 上のヒュージページのサイズを設定します。この値は、CPU のハードウェアによって異なります。OVS-DPDK デプロイメントには 1G に指定します (`default_hugepagesz=1GB hugepagesz=1G`)。 `pdpe1gb` CPU フラグが出力されるかどうかをチェックして、CPU が 1G をサポートしていることを確認してください。

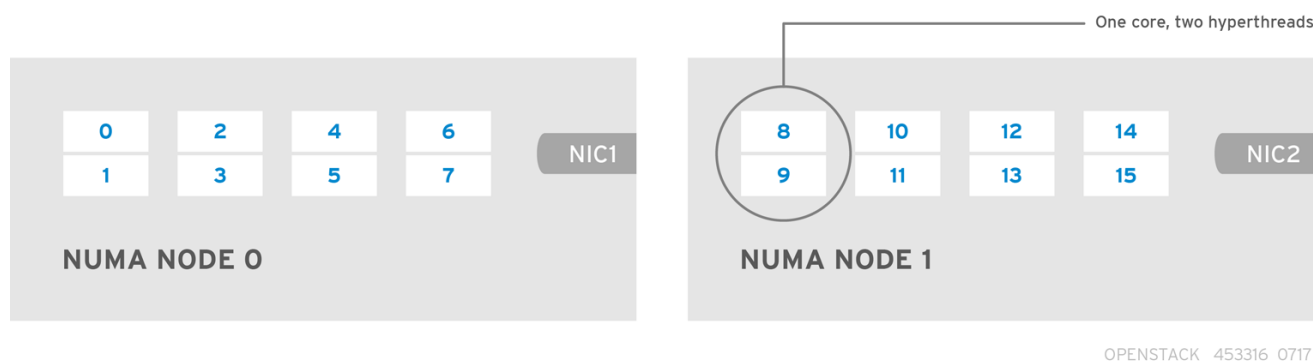
```
lshw -class processor | grep pdpe1gb
```

- **hugepages count**: ヒュージページの数を設定します。この値は、ホストの使用可能なメモリーの量によって異なります。利用可能なメモリーの大半を使用してください (`NovaReservedHostMemory` を除く)。ヒュージページ数の値は、お使いのコンピュートノードに関連付けられている OpenStack フレーバーの範囲内で設定する必要があります。
- **iommu**: Intel CPU の場合は、`"intel_iommu=on iommu=pt"` を追加します。
- **isolcpus**: チューニングされる CPU コアを設定します。この値は `HostIsolatedCpuList` と一致します。

6.3.2 NUMA ノード構成の OVS-DPDK デプロイメントの例

本項に例示するコンピュータノードは、以下のような 2 つの NUMA ノードで構成されます。

- NUMA 0 にはコア 0-7 がある (コア 1、3、5、7 はシブリングスレッド)。
- NUMA 1 にはコア 8-15 がある (コア 9、11、13、15 はシブリングスレッド)。
- 各 NUMA ノードが物理 NIC (NUMA 0 上の NIC1 と NUMA 1 上の NIC2) に接続されている。



注記

各 NUMA ノード上の 1 番目の物理コア (0、1 および 8、9) は、データパス以外の DPDK プロセス (`HostCpusList`) 用に確保します。

この例では、MTU が 1500 に設定されており、全ユースケースで `OvsDpdkSocketMemory` が同じであることも前提です。

`OvsDpdkSocketMemory`: "1024, 1024"

NIC 1 は DPDK 用で、1 つの物理コアは PMD 用です。

このユースケースでは、PMD 用の NUMA 0 に物理コアを 1 つ 割り当てます。NUMA 1 ノードでは NIC に DPDK が有効化されていませんが、NUMA 1 にも物理コアを 1 つ割り当てる必要があります。残りのコア (`HostCpusList` 用に確保されていないコア) はゲストインスタンスに割り当てられます。その結果、パラメーターの設定は以下のようになります。

`NeutronDpdkCoreList`: "2, 3, 10, 11"
`NovaVcpuPinSet`: "4, 5, 6, 7, 12, 13, 14, 15"

NIC 1 は DPDK 用で、2 つの物理コアは PMD 用

このユースケースでは、PMD 用の NUMA 0 に物理コアを 2 つ 割り当てます。NUMA 1 ノードでは NIC に DPDK が有効化されていませんが、NUMA 1 にも物理コアを 1 つ割り当てる必要があります。残りのコア (`HostCpusList` 用に確保されていないコア) はゲストインスタンスに割り当てられます。その結果、パラメーターの設定は以下のようになります。

`NeutronDpdkCoreList`: "2, 3, 4, 5, 10, 11"
`NovaVcpuPinSet`: "6, 7, 12, 13, 14, 15"

NIC 2 は DPDK 用で、1 つの物理コアは PMD 用です。

このユースケースでは、PMD 用の NUMA 1 に物理コアを 1 つ 割り当てます。NUMA 0 ノードでは NIC

に DPDK が有効化されていませんが、NUMA 0 にも物理コアを 1 つ割り当てる必要があります。残りのコア (**HostCpusList** 用に確保されていないコア) はゲストインスタンスに割り当てられます。その結果、パラメーターの設定は以下のようになります。

```
NeutronDpdkCoreList: "2, 3, 10, 11"  
NovaVcpuPinSet: "4, 5, 6, 7, 12, 13, 14, 15"
```

NIC 2 は DPDK 用で、2 つの物理コアは PMD 用

このユースケースでは、PMD 用の NUMA 1 に物理コアを 2 つ 割り当てます。NUMA 0 ノードでは NIC に DPDK が有効化されていませんが、NUMA 0 にも物理コアを 1 つ割り当てる必要があります。残りのコア (**HostCpusList** 用に確保されていないコア) はゲストインスタンスに割り当てられます。その結果、パラメーターの設定は以下のようになります。

```
NeutronDpdkCoreList: "2, 3, 10, 11, 12, 13"  
NovaVcpuPinSet: "4, 5, 6, 7, 14, 15"
```

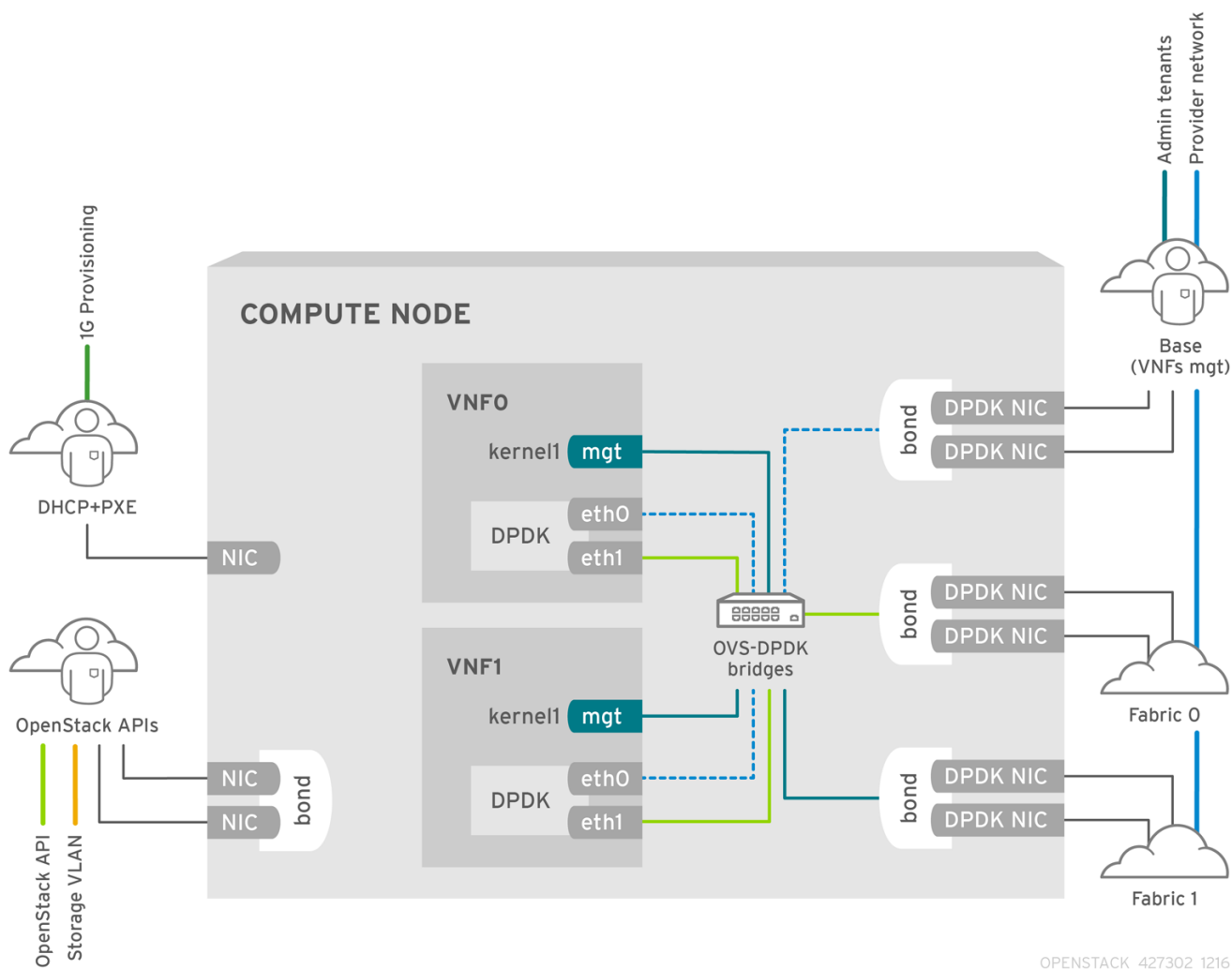
NIC 1 と NIC2 は DPDK 用で、2 つの物理コアは PMD 用

このユースケースでは、PMD 用の各 NUMA ノードに物理コアを 2 つ 割り当てます。残りのコア (**HostCpusList** 用に確保されていないコア) はゲストインスタンスに割り当てられます。その結果、パラメーターの設定は以下のようになります。

```
NeutronDpdkCoreList: "2, 3, 4, 5, 10, 11, 12, 13"  
NovaVcpuPinSet: "6, 7, 14, 15"
```

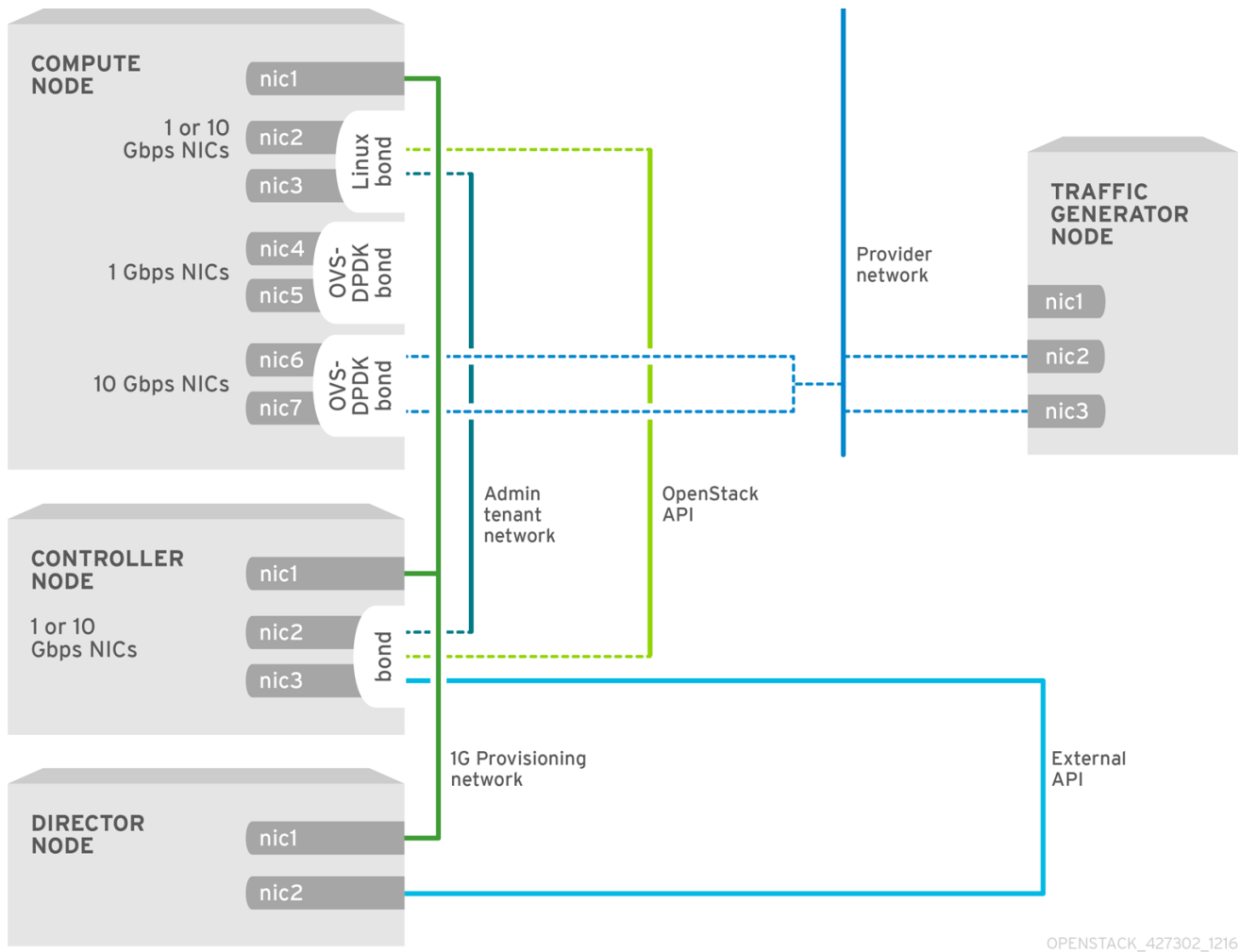
6.4. NFV OVS-DPDK デプロイメントのトポロジー

この例の OVS-DPDK デプロイメントは 2 つの VNF で構成され、それぞれに 2 つのインターフェース (**mgt** で示されている管理インターフェースとデータプレーンインターフェース) があります。OVS-DPDK デプロイメントでは、VNF は、物理インターフェースをサポートする組み込みの DPDK で稼働します。OVS-DPDK は、vSwitch レベルでボンディングを管理します。OVS-DPDK デプロイメントでは、カーネルと OVS-DPDK の NIC を **混在させない** ことを推奨します。混在させた場合には、パフォーマンスが低下する可能性があります。仮想マシン向けのベースプロバイダーネットワークに接続された管理 (**mgt**) ネットワークを分離するには、追加の NIC があることを確認する必要があります。コンピュータノードは、OpenStack API 管理向けの標準の NIC 2 つで構成されます。これは、Ceph API で再利用できますが、OpenStack テナントとは一切共有できません。



NFV OVS-DPDK のトポロジー

以下の図には、NFV のユースケース向けの OVS_DPDK のトポロジーを示しています。この環境は、1 Gbps または 10 の 1 Gbps の NIC を搭載したコンピュータードおよびコントローラーノードと、director ノードで構成されます。



第7章 パフォーマンス

Red Hat OpenStack Platform 11 director は、コンピュータノードがリソースの分割および微調整を有効にしてゲスト VNF のラインレートパフォーマンスを実現できるように設定します。NFV ユースケースの中での主要なパフォーマンス要素は、スループット、レイテンシー、ジッターです。

DPDK-accelerated OVS は、物理 NIC と仮想マシンの間で高性能のパケット切り替えを有効にします。OVS 2.6 は、DPDK 16.04 に対応しており、**vhost-user** のマルチキューをサポートしているので、スケーラブルなパフォーマンスを実現できます。OVS-DPDK は、ゲスト VNF のラインレートパフォーマンスを提供します。

SR-IOV ネットワークでは、固有なネットワークや仮想マシンのスループットの向上など、強化されたパフォーマンス特性が提供されます。

パフォーマンスチューニングの他の重要な機能には、ヒュージページ、NUMA 調整、ホストの分離、CPU ピニングなどが挙げられます。VNF フレーバーには、パフォーマンス向上のためにヒュージページが必要です。ホストの分離や CPU ピニングにより、NFV パフォーマンスが向上され、擬似パケットロスが回避されます。

CPU と NUMA トポロジーの概要は [「NFV Performance Considerations」](#) を参照してください。

第8章 その他の参考資料

以下の表には、その他の参考となる Red Hat のドキュメントの一覧を記載しています。

Red Hat OpenStack Platform のドキュメントスイートは [Red Hat OpenStack Platform 11 の製品ドキュメント](#) から参照してください。

表8.1 参考資料一覧

コンポーネント	参考情報
Red Hat Enterprise Linux	Red Hat OpenStack Platform は Red Hat Enterprise Linux 7.3 でサポートされています。Red Hat Enterprise Linux のインストールに関する情報は、 Red Hat Enterprise Linux の対応するインストールガイドを参照してください。
Red Hat OpenStack Platform	<p>OpenStack のコンポーネントとそれらの依存関係をインストールするには、Red Hat OpenStack Platform director を使用します。director は、基本的な OpenStack 環境を アンダークラウド として使用して、最終的な オーバークラウド 内の OpenStack ノードをインストール、設定、管理します。デプロイしたオーバークラウドに必要な環境に加えて、アンダークラウドをインストールするための追加のホストマシンが 1 台必要である点に注意してください。詳しい手順は、『Red Hat OpenStack Platform director のインストールと使用方法』 を参照してください。</p> <p>Red Hat OpenStack Platform director を使用して、ネットワークの分離、ストレージ設定、SSL 通信、一般的な設定方法など Red Hat OpenStack Platform のエンタープライズ環境の高度な機能設定に関する情報は 『オーバークラウドの高度なカスタマイズ』 を参照してください。</p> <p>Red Hat OpenStack Platform コンポーネントを手動でインストールすることもできます。方法については、『手動インストール手順』 を参照してください。</p>
NFV のドキュメント	<p>NFV の概念に関する俯瞰的な情報は、『ネットワーク機能仮想化 (NFV) の製品ガイド』 を参照してください。</p> <p>Red Hat OpenStack Platform 11 director での SR-IOV および OVS-DPDK 設定に関する情報は 『ネットワーク機能仮想化 (NFV) の設定ガイド』 を参照してください。</p>