OpenStack を使用した、通信およびサービス事業者向けのクラウド型 NFV ソリューション商用化に向けた共同検証

40 インスタンス最繁時における 40Gpbs(1 インスタンスあたり 1Gbps)の通信性能を達成し、企業ネットワークに設置されている顧客構内設備（CPE）を、通信事業者側に仮想的に集約したクラウド WAN サービスへの実現の取り組みを加速。

目次

1. はじめに
2. 共同実証実験の目的と概要
3. クラウド型 vCPE (Cloud vCPE)
4. Red Hat の Openstack と NFV への取組み
5. Red Hat Enterprise Linux OpenStack Platform
6. NUMA トポロジ考慮と CPU ピンニング
7. Intel DPDK
8. SR-IOV とパケット処理性能
9. 検証
10. リソースの割当てとネットワーク設定
11. 検証結果
12. ネットワークの障害検知
13. 高密度メインストリームラックサーバ/Dell PowerEdge R630
14. Brocade vRouter 5600
15. 検証結果と今後の展望

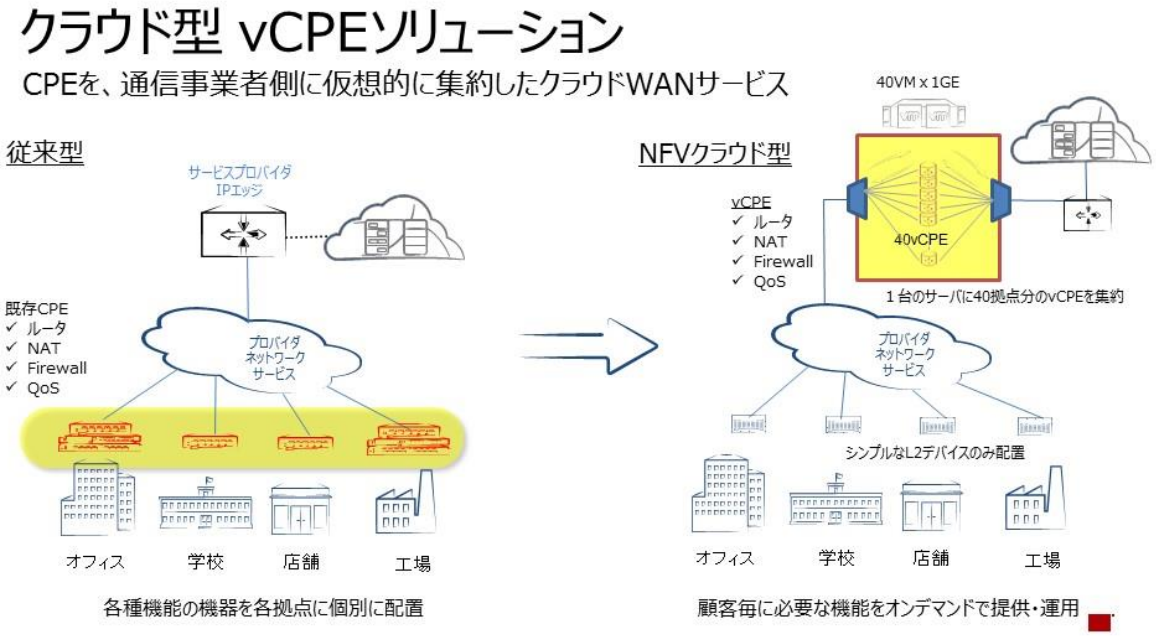
1. はじめに

この度、ブロケード、およびレッドハット、デル、インテルの 4 社は、OpenStack を使用した通信およびサービス事業者向けクラウド型 NFV ソリューションの共同検証実験を実施しました。今回の検証では、1 台のサーバに 40 インスタンスの仮想ルータ（vCPE）を集約して、40GB インタフェース最繁時における各インスタンスの通信性能を 1Gbps 均等に維持し、計 40Gbps の通信性能を達成しており、これにより、通信・サービス事業者がコストを抑えながら、優れたサービス品質と競争力のある柔軟なサービス開発を可能にする、新たな仮想クラウド WAN サービスの提供に向けた取り組みを加速させることになります。

2. 共同実証実験の目的と概要

現在、ビジネスを支える IT プラットフォームの世界では、“モバイル”、“ソーシャル”、“ビッグデータ”、“クラウド”に代表されるいわゆる「第 3 のプラットフォーム」が市場を牽引しています。そして、こうした要求に対応する新しいネットワークの形として求められているのが、「New IP」と呼ばれるアーキテクチャーです。NFV（ネットワーク機能仮想化）は New IP を実現するための大きな技術コンポーネントのひとつで、従来の高価な専用ハードウェア機器の代わりにインテル® アーキテクチャー(IA)ベースの汎用サーバを用いてネットワークの機能を仮想的に実現するもので、多くの場合オープンソース技術を取り入れ、ソフトウェア・ベースのコントローラやオーケストレーション・ツールと連携することでシンプルさと俊敏性を最大化します。

従来、企業では各拠点に顧客構内設備（CPE）と呼ばれるさまざまネットワーク機器を設置して、通信・サービス事業者のネットワークを介して、WAN を構成する必要があり、企業は CPE の管理およびコストの面で非常に大きな負担を強いられていました。



こうした状況の中、ブロケード、およびレッドハット、デル、インテルの 4 社は、デルが設立したオープン標準を利用した柔軟なクラウド環境を推進するコンソーシアム「Open Standard Cloud Association（略称：OSCA）を通じて、サービス事業者、およびサービスを利用する企業の双方にメリットのある仮想 WAN サービスの実現に向けた検証を実施しました。これは、CPE などのネットワーク・リソースを、通信・サービス事業者側のクラウドに vCPE として仮想的に集約して管理・運用することで、サービス提供を受ける企業の管理およびコスト負担を軽減しながら、事業者は運用の簡素化とコスト削減というメリットを享受することができるクラウド型 vCPE（Cloud vCPE）を使用した NFV ソリューションの商用化に向けた取り組みとなります。

ブロケード社内に設置されている「NEW IP ラボ」 において 2015 年 7 月に行われたこの共同実証実験は、デルの Dell PowerEdge R630 サーバ 1 台に、OpenStack を構成するレッドハットの Red Hat Enterprise Linux Open Stack Platform 7、ブロケードの Brocade vRouter 5600 仮想ルータ、およびインテルの DPDK などの製品技術を組み込んだ環境で行われました。

【共同実証実験の結果概要】
サーバ 1 台の上に 40 インスタンスの vCPE を稼働させた今回の検証実験では、40Gbps の性能を達成したことが確認されました。今回の検証実験により、「クラウド型 vCPE」を利用した NFV ソリューションとしては、業界に先駆けて高い水準の通信性能が実証されたことになり、この結果をレファレンスとしたサービス事業者における商用化に向けて大きな弾みとなります。今回の検証結果を受けて、レッドハット、デル、インテル、ブロケードの 4 社は、2016 年春頃の商用展開を目指して、4 社間で今後の協力関係や共通のチャネル構築について、協議を進めていく意向です。

なお、このたびの共同検証実験で使用した各社の主な製品技術は、以下の通りです。

【レッドハット】
・Red Hat Enterprise Linux OpenStack Platform 7
OpenStack Kilo をベースにした最新 OpenStack ディストリビューション。単純化された導入と管理、OpenStack のワークロードの高可用性、セキュリティコントロールの強化、ネットワークの柔軟性、差分バックアップなどの特長を持つ。
製品詳細情報：
https://www.redhat.com/ja/technologies/linux-platforms/openstack-platform

【デル】
・Dell PowerEdge R630 サーバ
CPU/メモリ/ストレージを高密度に搭載が可能な 2 ソケット/1U ラックサーバ。前面に最大 24 個の 1.8 インチ SSD を搭載可能。即応性、柔軟性の高いデータセンターを実現し、データベース、仮想化環境、Web 環境および HPC 用途に最適。
製品詳細情報：
http://ja.community.dell.com/dell-blogs/direct2dell/b/direct2dell/archive/2014/09/09/13-dell-poweredge

【インテル】
・DPDK
パケット処理のパフォーマンスとスループットを大幅に高めて、データプレーン・アプリケーションに多くの時間を確保するデータプレーン開発キット。DPDK の使用によりパケット処理パフォーマンスが最大 10 倍向上。その結果、通信およびネットワーク機器メーカー（TEM と NEM）は、開発コストの削減、使用するツールとサポートするチームの削減、市場投入までの期間の短縮が可能。
製品詳細情報：
http://www.intel.co.jp/content/www/jp/ja/communications/data-plane-development-kit.html?wapkw=dpdk&_ga=1.153745720.1293539213.1442062112

【ブロケード】
・Brocade Vyatta vRouter 5600
NFV の要求に対応し、ハードウェア・ネットワーク機器が持つ信頼性や性能はそのままに、先進のルーティング、ファイアウォール、および VPN 機能をソフトウェアで実現する初の仮想ルータ。高度なルーティングに加えて、ステートフル・ファイアウォール、VPN などの高性能なネットワーク機能を提供。スペインの大手通信事業者の Telefónica（テレフォニカ）社と 2014 年に実施した、Vyatta vRouter 5600 を利用した NFV ベンチマーク・テストにおいて、市販の商用 IA サーバ上で 80Gbs の性能を達成。

・2014 年 8 月 25 日発表報道資料 「Telefónica とブロケード、共同のリファレンス・ラボ・テストにおいて NFV 導入および性能面の常識を覆す優れた結果を達成」
http://www.brocadejapan.com/news/20140825
・製品詳細情報：
http://www.brocadejapan.com/products/network-functions-virtualization/5600-vrouter/overview

3. クラウド型 vCPE (Cloud vCPE)

NFV は、各種ネットワーク機器の機能をサーバに機能集約して提供することにより、これまで物理的に構築していたルータ、ロードバランサ、WAN 高速化装置やファイアウォールなどのネットワーク機器の仮想化を可能にします。また、カスタマーポータル画面などを通じてユーザー自身が必要に応じてネットワークや各種機器の設定変更ができ、運用コストの削減、納期の短縮ができます。

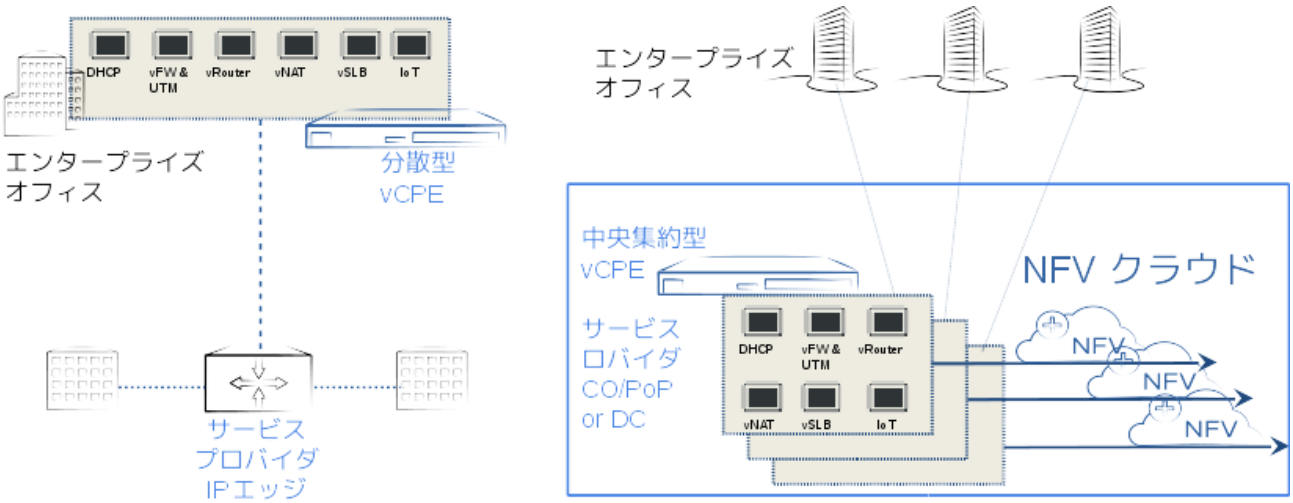
ユーザー企業が自前で構築、保有していたネットワーク機器を、ストレージやサーバなどのクラウド化とあわせてプロバイダが提供するクラウド上で利用することで、大幅な運用コスト削減、数カ月から数分への納期短縮など、柔軟で利便性の高い高信頼のネットワークが利用可能になります。

【Cloud vCPE の導入メリット】

- CPE 仮想化による初期投資・運用コストの低減
  - ➢ インフラの汎用サーバ化により、機器調達のコストを低減することができます
  - ➢ CPE の機能をネットワーク側に移すことで、インフラはシンプルとなり運用が楽になります。
  - ➢ ソフトウェアアップデートなどの作業をプロバイダ側で効率よく一括で行なうことができます

- 迅速な新規サービスの提供が可能
  - ➢ 現地の CPE を交換することなく、プロバイダ側での機能追加により、柔軟かつ迅速に新サービスを導入することができます。
  - ➢ 従来必要だったオンサイト作業に要する時間と人材育成のための時間を削減できます。

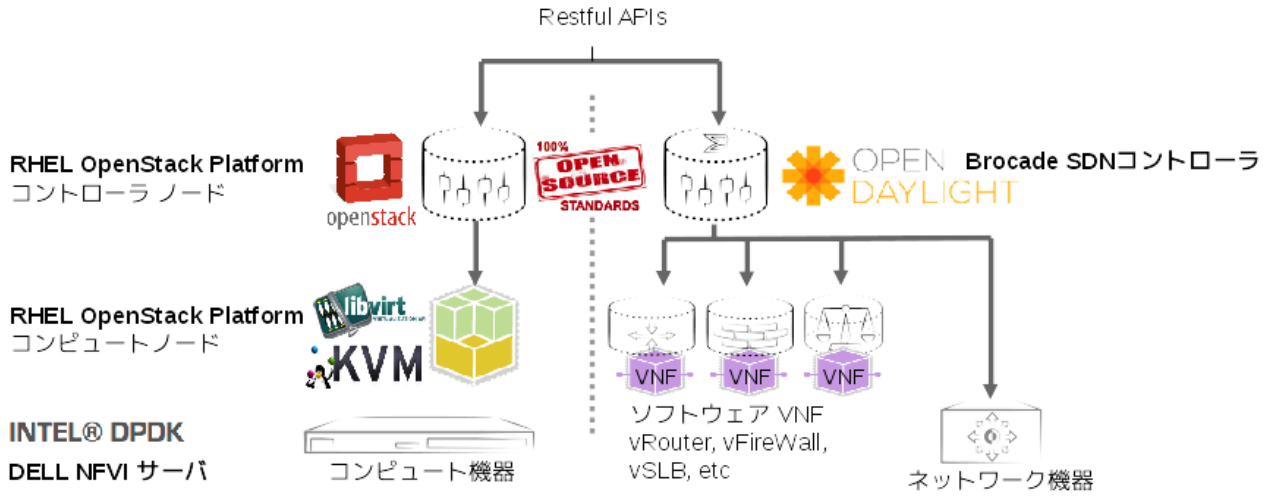
- プロバイダ側での障害対応の簡略化
  - ➢ 各種ネットワーク機能をプロバイダ側に移行したことで、接続された他の通信機器の制御やセキュリティの保証が可能になります。
  - ➢ ハードウェアの通信機器数を減らすことにより、故障頻度を減らすことにも繋がります



4. Red Hat の Openstack と NFV への取組み

Infonetics Research 社によって実施された 2014 SDN and NFV Strategies: Global Service Provider Survey によると、キャリア各社が通信ネットワークの構築/運用方法のモダナイゼーションを進める中で、調査対象の通信事業者の 93%は NFV の配備を計画しています。OpenStack が NFV ワークロード用の有力プラットフォームとしての地位を確立していることを受け、Red Hat は各社とグローバルパートナーシップを締結し、協力して CSP 向けに新たなキャリアグレードなクラウドソリューションを開発します。

このイニシアティブの一環として、RedHat は、各社の世界的水準の専門知識および通信企業との広範でグローバルな経験と、OpenStack とオープンソースに関する Red Hat のトップクラスの専門知識を組み合わせて、CSP の顧客である企業向けミッションクリティカル SLA が適用できる OpenStack ソリューションを使用したクラウドコンピューティングを CSP が導入する手助けになることを目指します。Red Hat は、Red Hat Enterprise Linux OpenStack Platform と各社のソリューションを統合し、オープンで柔軟性の高い、実運用に対応する統一されたクラウドソリューションを提供して、通信キャリア各社による NFV の進展をサポートします。また、各社との協力により、コミュニティへの貢献、エンジニアリング、製品、および市場進出作業の整合を図り、CSP による NFV 実装のための OpenStack 採用を促進します。



5. Red Hat Enterprise Linux OpenStack Platform

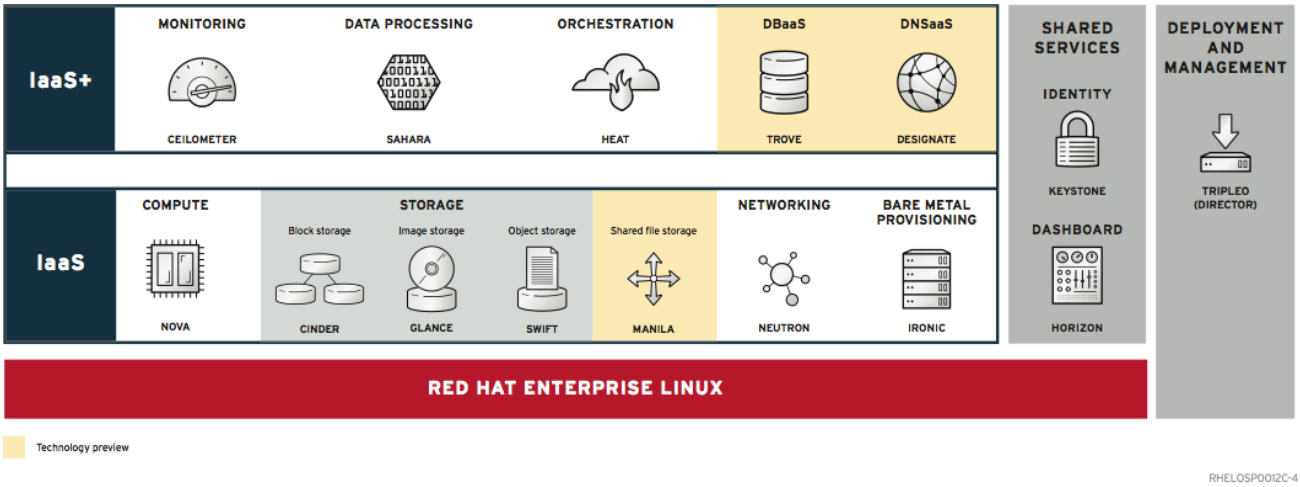
IT 部門は日々、増大するユーザーニーズへの迅速な対応を迫られています。そのため、Infrastructure-as-a-Service（IaaS）プライベートクラウドへの移行を推進することで IT インフラストラクチャのデプロイおよび拡張を高速化し、エンドユーザーの要求に応えようとする傾向が強まっています。しかし、どの OpenStack® クラウドでも、本稼働規模の環境のニーズや、パフォーマンス、スケーラビリティ、セキュリティの基準を満たせるというわけではありません。少なくとも、現時点ではそう言わざるを得ません。持続的イノベーションが必要だ。そこでお勧めしたいのが、Red Hat® Enterprise Linux® OpenStack Platform です。このプラットフォームなら、競争力を獲得でき、”Upstream First”を基本とした持続的イノベーションが期待できます。

OpenStack は、急速に発展しているオープンソースクラウドコミュニティプロジェクトであり、ベンダーロックインのないオープンなクラウド基盤を実現するものです。コンピュート、ネットワーク、スト

レージを OpenStack によって管理することにより、柔軟性が高く、拡張性に優れたクラウド基盤を構築で きます。開発者、ユーザー、テクノロジ企業を結び付け、あらゆるタイプのパブリッククラウドとプライベートクラウド向けの標準化されたクラウドコンピューティングプラットフォームを作り上げ、相互使用可能な複数のプロジェク トで構成されています。クラウドコンピューティングのサービス基盤を提供するとともに、拡張性が非常に高いサービス基盤を構築できます。

OpenStack は、Linux OS を利用して、ハードウェアアクセス、パフォーマンス、拡張性、セキュリティを実現します。また、実行されているゲストアプリケーションが OS を使用し ます。
OpenStack の導入を検討されている場合は、OpenStack ディストリビューションの選択だけでなく、Linux 製品の選択も重要です。Red Hat® Enterprise Linux® OpenStack® Platform は、Red Hat Enterprise Linux と Red Hat OpenStack クラウドプラットフォームを組み合わせた製品であり、拡張性に優れ、セキュアなプライベート/ パブリッククラウド基盤として利用できます。

Red Hat Enterprise Linux OpenStack Platform 7



OpenStack kilo(RHELOSP7)・リリースでは、juno(RHELOSP6)サイクル中に始まった取り組みが広まり、ゲスト・パフォーマンスを向上させることを目的とするいくつかの鍵となる機能拡張(CPU ピンニングと NUMA トポロジの考慮等) を含むことになりました。これらの機能拡張は OpenStack Nova コンピュートにコンピュート・ホスト・レイアウト情報とインスタンス起動時にスマートなスケジューリングと配置決定機能をもたせることができ、管理者が NFV (Network Function Virtualization) や HPC (High Performance Computing) を含む特別なワークロードを対象としたフレーバのパフォーマンスをカスタマイズ出来るようになりました。

6. NUMA トポロジーの考慮と CPU ピンニング

NUMA（**Non-Uniform Memory Access**）

従来の x86 システムでは、 メモリーはすべて均等に全 CPU からアクセスできるようになっていました。どの CPU が動作を実行するかに関わらず、 UMA (Uniform Memory Access) と呼ばれるアクセスタイムは同じになります。

最新のマルチ・ソケット x86 システムでは、この動作が異なってきています。システム・メモリ（セルまたはノードと呼ばれる）はゾーンに区分けされ、 特定の CPU やソケットに割り当てられます。このように CPU とメモリの対が複数存在し、それらをインターコネクトで接続したものを NUMA（Non-Uniform Memory Access）と呼び、複数プロセッサが共有するメインメモリへのアクセスコストが、メモリゾーンとプロセッサに依存して均一でないアーキテクチャとなっています。CPU に対してローカルとなるメモリーへのアクセスが、そのシステムのリモート CPU に接続しているメモリーに比べて速くなります。この種の区分は、CPU クロック・スピードを上げることより CPU ソケット、コアまた利用可能なスレッドを多く加えることに焦点をシフトさせた現代のシステムパフォーマンス向上のための鍵となる機能です。
また、すべての CPU がまだすべてのメモリにアクセスできるように、ノード間の接続をインターコネクトバスで提供しています。より新しいマザーボード・チップセットでは、CPU 間で PCIe_I/O レーンの NUMA スタイル区分を提供することによって、この概念を拡大しています。

RHEL7 での `numactl --hardware` コマンドで以下のように NUMA レイアウトを確認することができます。

NUMA レイアウト 例：

```
# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3
node 0 size: 8191 MB
node 0 free: 6435 MB
node 1 cpus: 4 5 6 7
node 1 size: 8192 MB
node 1 free: 6634 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

この”numactl --hardware”の出力例では、2 NUMA ノード： ノード 0、ノード 1 、各ノード： 4 CPU コア、8G メモリ表示に加え、NUMA ノード間相対的な距離 ”`node distances` を示しています。RHEL7 等のオペレーティング システムは、システム リソースの使用状況を監視し、動的にベスト パフォーマンスのためのプロセスと、関連するメモリが最適に置かれることを確認するための調整をするために numad のような付加サービスを提供することによって、システムの NUMA ポロジーを考慮するようしています。

NUMA（**Non-Uniform Memory Access**）トポロジーの考慮

NUMA トポロジーは、ホストの物理ハードウェアとインスタンスの仮想ハードウェアの両方に存在できます。ゲスト オペレーティング システムを実行する仮想ハードウェアが実際に 2 つの NUMA トポロジーを考慮する際、ホストの物理ハードウェア情報と仮想ハードウェア情報は、ゲスト オペレーティングシステムに認識させるようにします。 ホスト オペレーティング システムおよび関連するユーティリティはホストの NUMA トポロジーを認識しておりそれに応じて最適化されますが、ゲスト仮想マシンの場合は物理ハードウェアで実行されているのと同じようにゲストオペレーティングシステムに NUMA トポロジーを認識させることによって最適化できます。
OpenStack Compute は libvirt を使用して、NUMA トポロジーの利点を得るためにゲスト インスタンスをチューニングします。libvirt ドライバーの起動プロセスは、起動するゲスト インスタンスとホストの両方の NUMA トポロジーの項目を参照して、適切な設定を生成するためにその情報を使用します。
Libvirt は、NUMA トポロジの利点を活用するゲストをチューニングするため、広範なオプションを提供します（例えば、仮想 CPU を物理 CPU にピンニング、ゲストに関連付けられたエミュレーターのスレッドを物理 CPU にピンニング、通常のメモリ”4k ページ”とラージページ”2MB または 1G ページ”の両方の割り当てポリシーのチューニング等）。 以下の Virsh コマンド例で示す通り、ゲスト仮想マシンインスタンス最適配置のため、OpenStack Compute がインスタンス構築及びスケジューリングする際に NUMA トポロジーの考慮は重要なオプションとなります。前の例で使用されている同じホストをもとに表示すると、ホストの機能が表示され実行するさまざまな情報が得られますが、特に<topology>のセクションを以下に例示します。NUMA ノードはそれぞれ、ノードの中で利用できる CPU をリストする<cell>エントリ、ノードの中で利用できるメモリ情報（ページサイズ、およびノードとの距離）によって表現されます。

```
# virsh capabilities
...
      <topology>
          <cells num='2'>
            <cell id='0'>
              <memory unit='KiB'>4193872</memory>
              <pages unit='KiB' size='4'>1048468</pages>
              <pages unit='KiB' size='2048'>0</pages>
              <distances>
                <sibling id='0' value='10'/>
                <sibling id='1' value='20'/>
              </distances>
              <cpus num='4'>
                <cpu id='0' socket_id='0' core_id='0' siblings='0'/>
                <cpu id='1' socket_id='0' core_id='1' siblings='1'/>
                <cpu id='2' socket_id='0' core_id='2' siblings='2'/>
                <cpu id='3' socket_id='0' core_id='3' siblings='3'/>
              </cpus>
            </cell>
            <cell id='1'>
              <memory unit='KiB'>4194304</memory>
              <pages unit='KiB' size='4'>1048576</pages>
              <pages unit='KiB' size='2048'>0</pages>
              <distances>
```

```
                <sibling id='0' value='20'/>
                <sibling id='1' value='10'/>
              </distances>
              <cpus num='4'>
                <cpu id='4' socket_id='1' core_id='0' siblings='4'/>
                <cpu id='5' socket_id='1' core_id='1' siblings='5'/>
                <cpu id='6' socket_id='1' core_id='2' siblings='6'/>
                <cpu id='7' socket_id='1' core_id='3' siblings='7'/>
              </cpus>
            </cell>
          </cells>
        </topology>
...
```

捕捉：
OpenStack Compute は libvirt を使用して、ゲスト インスタンスをチューニングしますが、関連設定情報を以下にまとめておきます。

OpenStack Compute 設定

単一 OpenStack controller node の小規模な PoC 環境では、OpenStack controller node が OpenStack API サービス、データベース、メッセージ キュー、およびスケジューラをホストし、各 OpenStack compute node が Compute エージェント、Libvirt、および KVM 仮想マシンを実際に起動に必要なその他のコンポーネントを実行します。HA 機能等を含む大規模なシステム構築では Red Hat OpenStack ダイレクターの使用を推奨しますが、このように PoC 等小規模システム構築では Packstack が使えます。この設定例では、ホストに 8 つの CPU コアを備え、0-7、2 つの NUMA ノード中で広がり NUMA ノード 0 に CPU コア 0-3 を含み、NUMA ノード 1 に 4-7 の CPU コアが含まれ、 各 NUMA ノード上のホスト プロセス コア 0,1,4,5 の 2 つのコアを予約しています。残り 4 つの CPU コア-2,3,6,7-は、一般的なカーネル・バランシングとスケジューリングのプロセスを配置するアルゴリズムによって使用され、ゲスト仮想マシンのインスタンスを配置するとき使用するために分離したプールから削除されます。isolcpus カーネル引数を使用しています。

	Node 0		Node 1	
Host Processes	Core 0	Core 1	Core 4	Core 5
Guests	Core 2	Core 3	Core 6	Core 7

各 Compute ノードに許可される仮想マシンのピンニングのために/etc/nova/nova.conf ファイルで、vcpu_pin_set や reserved_host_memory_mb 等の値が変更されます。
Compute ノード設定変更した場合は、Compute ノードのリスタートが必要になります。

systemctl restart openstack-nova-compute.service

また、既に XML でいくつかを変更している場合、vcpu_pin_set に記載されているコアにゲスト vCPU(s) をピンニングします。

<vcpu placement='static' cpuset='2-3,6-7'>1</vcpu>

RHEL7 では、以下のよう grubby を使って、isolcpus カーネル引数を設定することができます。

```
# grubby --update-kernel=ALL --args="isolcpus=2,3,6,7"
```

＊この場合、grub2-install <device>でブートレコードを更新します。

スケジューラ設定

OpenStack Compute スケジューラ (openstack-nova-scheduler) を実行する各ノードは /etc/nova/nova.conf を編集し、scheduler_default_filters のリストに AggregateInstanceExtraSpecFilter と NUMATopologyFilter の値を追記します。
以下のフィルターが使用できないことから CPU ピンニング使用できる OpenStack Compute を分離し、インスタンスを起動するときに、NUMA 対応スケジューリング・ルールを適用します。

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,ComputeFilter,Compu
teCapabilitiesFilter,ImagePropertiesFilter,CoreFilter,
```

＊NUMATopologyFilter,AggregateInstanceExtraSpecsFilter 追記後、openstack-nova-scheduler service のリスタートが必要です。

```
# systemctl restart openstack-nova-scheduler.service
```

ホストアグリゲート設定

ホストアグリゲートにより、管理ユーザーがキーと値のペアをマシンのグループに割り当てることができるようになります。 専用の Compute リソースをマークし、物理リソースにそれに応じてピンニング仮想マシン インスタンスを起動することができるようにするため、OpenStack Compute で nova aggregate コマンドと nova flavor コマンドを実行して、設定を完成させます。

```
$ nova aggregate-create performance
+----+-------------+-------------------+-------+----------+
| Id | Name        | Availability Zone | Hosts | Metadata |
+----+-------------+-------------------+-------+----------+
| 1  | performance | -                 |       |          |
+----+-------------+-------------------+-------+----------+
$ nova aggregate-set-metadata 1 pinned=true
+----+-------------+-------------------+-------+---------------+
| Id | Name        | Availability Zone | Hosts | Metadata      |
+----+-------------+-------------------+-------+---------------+
| 1  | performance | -                 |       | 'pinned=true' |
+----+-------------+-------------------+-------+---------------+
$ nova aggregate-create normal
+----+--------+-------------------+-------+----------+
| Id | Name   | Availability Zone | Hosts | Metadata |
+----+--------+-------------------+-------+----------+
| 2  | normal | -                 |       |          |
+----+--------+-------------------+-------+----------+
$ nova aggregate-set-metadata 2 pinned=false
+----+--------+-------------------+-------+----------------+
| Id | Name   | Availability Zone | Hosts | Metadata       |
+----+--------+-------------------+-------+----------------+
| 2  | normal | -                 |       | 'pinned=false' |
+----+--------+-------------------+-------+----------------+
```

パフォーマンス指向インスタンスの新しいフレーバーを作成する前に、Compute ホストで既存のフレーバーすべてを以下のように更新する必要があります。

```
$ for FLAVOR in `nova flavor-list | cut -f 2 -d ' ' | grep \
 -o [0-9]*`; \
       do nova flavor-key ${FLAVOR} set \
        "aggregate_instance_extra_specs:pinned"="false"; \
 done
```

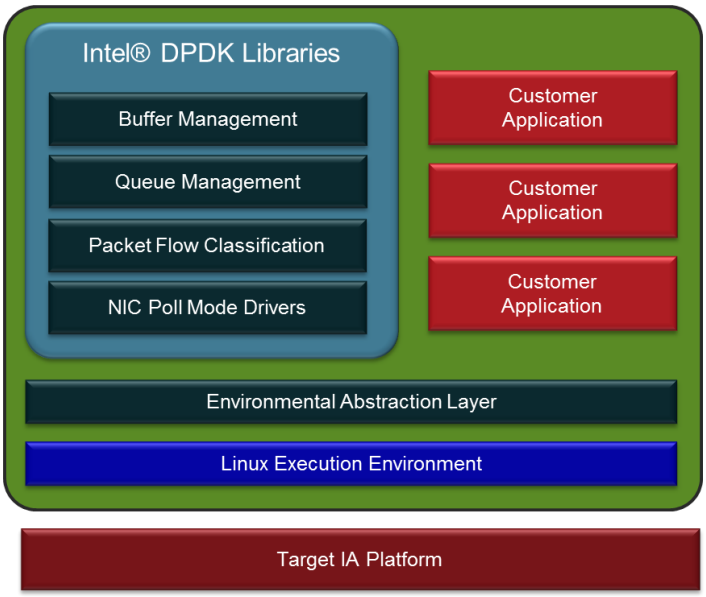
仮想ハードウェアのテンプレートは、OpenStack において「フレーバー」と呼ばれます。これは、RAM、ディスク、コア数などを定義します。

```
$ nova flavor-create m1.small.performance 6 4096 20 2
+--+---------------------+-----------+------+-----------+---+------+
|ID| Name                | Memory_MB | Disk |Ephemeral |Swp| VCPUs|
+--+---------------------+-----------+------+-----------+---+------+
|6 |m1.small.performance| 4096      | 20   | 0         |   | 2    |
+--+---------------------+-----------+------+-----------+---+------+

$ nova flavor-key 6 set hw:cpu_policy=dedicated

$ nova flavor-key 6 set aggregate_instance_extra_specs:pinned=true


$ nova aggregate-add-host 1 compute1.nova
+--+-----------+------------------+---------------+--------------+
|Id| Name      |Availability Zone | Hosts         | Metadata     |
+--+-----------+------------------+---------------+--------------+
|1|performance | -                |'compute1.nova'| 'pinned=true' |
+--+-----------+------------------+---------------+--------------+

$ nova aggregate-add-host 2 compute2.nova
+--+---------+------------------+---------------+---------------+
|Id|Name     |Availability Zone | Hosts         | Metadata      |
+--+---------+------------------+---------------+---------------+
|2| normal  | -                | 'compute2.nova'| 'pinned=false'|
+--+---------+------------------+---------------+---------------+
```

7. Intel DPDK

汎用プロセッサーでのパケット処理パフォーマンスを向上させるノウハウを実装した、オープンソースのソフトウェア・ライブラリーです。DPDK　の使用によりデータプレーン・アプリケーションのパケット処理パフォーマンス、スループットが向上。さらに、通信機器メーカー　は、使用するツールや最適化のための工数の削減により、開発コストの低減と市場投入までの期間の短縮が可能になります。

製品詳細情報：
http://dpdk.org



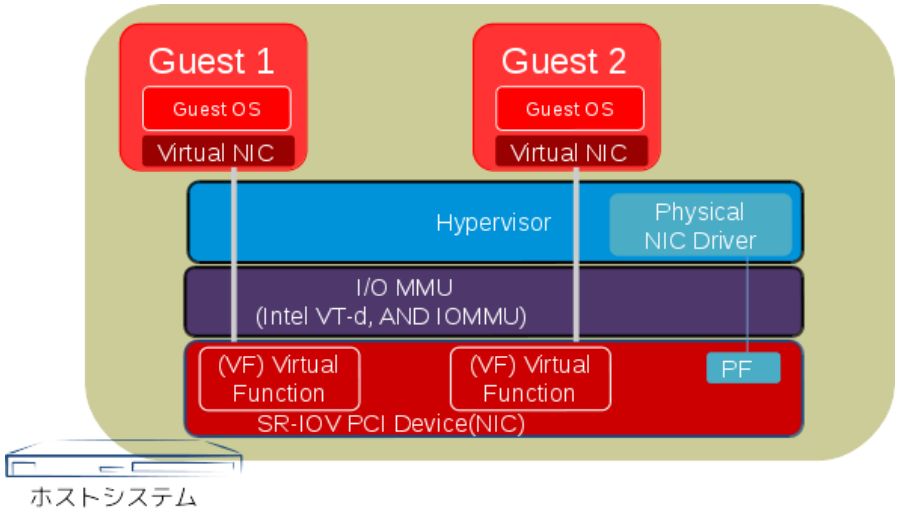
8. SR-IOV とパケット処理性能

RHEL-OSP6 (Red Hat Enterprise Linux OpenStack Platfiorm viesion 6)リリース以降、Single Root I/O virtualization (SR-IOV) ネットワークのサポートが導入されています。これは、Compute サービス (Nova) の PCI サポートの必要な拡張機能と同様に、OpenStack ネットワーキング (Neutron) Modular Layer 2 (ML2)で新しい SR-IOV メカニズム ドライバーを介してプラグインしています。

PCI パススルーは、ゲスト オペレーティング システム に PCI デバイスを直接割り当てることができます。これを行うための 1 つの前提条件は Intel vt-d または AMD IOMMU 拡張ハイパーバイザがサポートしていなければなりません。標準的なパススルー PCI デバイスへの排他アクセスを仮想マシン (VM) と PCI デバイスが表示され、あたかも物理的にゲスト オペレーティング システムに接続されています。全体のネットワーク デバイス (ネットワーク アダプター上の物理ポートなど) を占有する PCI パススルーを VM 内で実行中のゲスト オペレーティング システム で利用することが可能です。

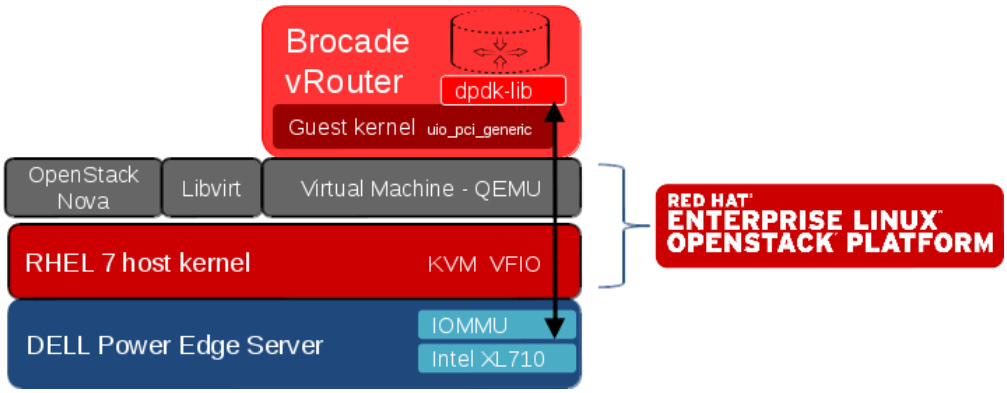
PCI-SIG (PCI Special Interest Group) により仕様が策定された Single Root I/O Virtualization (SR-IOV) は PCI デバイスと PCI ハードウェアの様々な機能のなかで、リソースへのアクセスを分離することができる仕様（PF:PhysicalFunction と 1 とつ又は複数の VF:VirtualFunction）となっています。PF は全 PCIe 機能を持つが、VF は構成リソースを持たないライトウェイトな機能です。VF の構成と管理は、データ移動だけで、PF を介して行われます。PF が利用できる全体的な帯域幅がそれと関連したすべての VF で分配される点に注意することが重要です。SR-IOV の仕様には、さまざまなゲストオペレーティングシステム間で単一の PCIExpress デバイスを共有する方法が詳述されています。ネットワーク インターフェイス カード（NIC）の各物理ポートが物理ファンクション（PF）として表される事と、各 PF は仮想ファンクシ

ョン(VFs) の設定可能な番号を関連付ける事ができます。

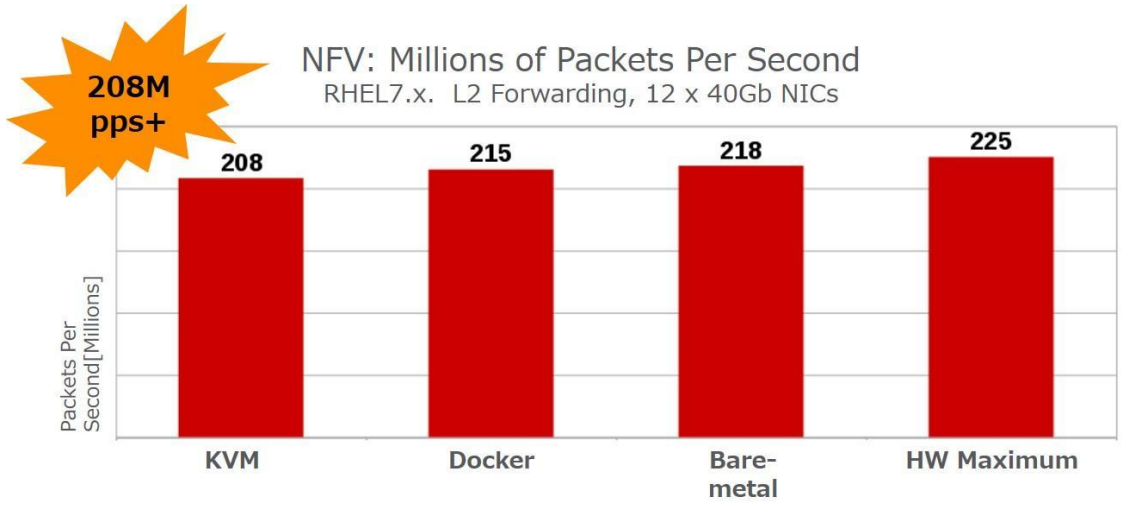


VF を仮想マシン・インスタンスに割り当てることは、ネットワーク・トラフィックがハイパーバイザーのソフトウェア層を回避して、VF とバーチャル・マシンの間で直接流れる事を可能にします。このように、I/O 操作のためのロジックはネットワーク・アダプタ自体にあります。そして、仮想マシンはそれらが複数の別々のネットワーク装置と相互作用します。これにより、個々の仮想マシンごとに個別の物理 NIC を占有する必要性がなく、ラインレートに近い通信性能を実現できるようになります。標準的な PCI Passthrough を SR-IOV と比較して、SR-IOV は、より多くの柔軟性を提供します。
ネットワーク・トラフィックが一般的に仮想化環境で使われるソフトウェア・スイッチを含むハイパーバイザーのソフトウェア層を完全に回避するので、物理ネットワーク・アダプタはトラフィックフロー（適当な分離とブリッジを含む）を管理する必要があります。これは、ネットワーク・アダプタが SR-IOV 対応を提供しなければならなく、何らかのハードウェア・ベースの Virtual Ethernet Bridge”VEB”を実行しなければならないことを意味します。

SR-IOV ネットワークを使用する主なモチベーションは、特定のネットワークと仮想マシンのパフォーマンスの強化特性(例えば、スループット、遅延)を提供するところです。RHEL-OSP Compute ノード (NFVI) の上で仮想ネットワーク機能 (VNFs) を実装しようとしている NFV ベンダーや CSP が SR-IOV を使用しており、さらに DPDK を活用した高速な通信性能を実現しています。 今回のテストでは以下の構成で SR-IOV が活用されております。



捕捉:
本テストとは別な目的で、RHEL7.1(RHEL7.1OSP7)と DPDK2.0 を用いて Compute ノードあたり-6x40Gbps,6 つの VM-を配備し、それぞれが基本的な VNF として DPDK の試験用 pkggen-dpdk を稼働させレイヤ 2 のフォワーディング性能テストを実施しました。
これは、Red Hat インターナルなテストで、同時にパケット処理するすべての 6 VM でこのテストを行い、パケット処理のオーバーヘッドの最悪の状況をシミュレートするために途中で帯域幅の限界に達することがないように 64 バイトのパケットサイズを使用し、213Mpps のパケット処理能力を実証しております。Openstack_KVM と DPDK の組み合わせだけでなく、Docker と DPDK の組み合わせ、ベアメタルと DPDK の組み合わせでも同じテストをしており、ベアメタルでは 218Mpps のパケット処理能力を確認しています。Openstack KVM と DPDK の組み合わせた SR-IOV パケット処理能力はベアメタルの 97.7%まで向上していることを確認しています。



詳しくはブログを参照ください。
http://redhatstackblog.redhat.com/2015/08/19/scaling-nfv-to-213-million-packets-per-second-with-red-hat-enterprise-linux-openstack-and-dpdk/

9. 検証

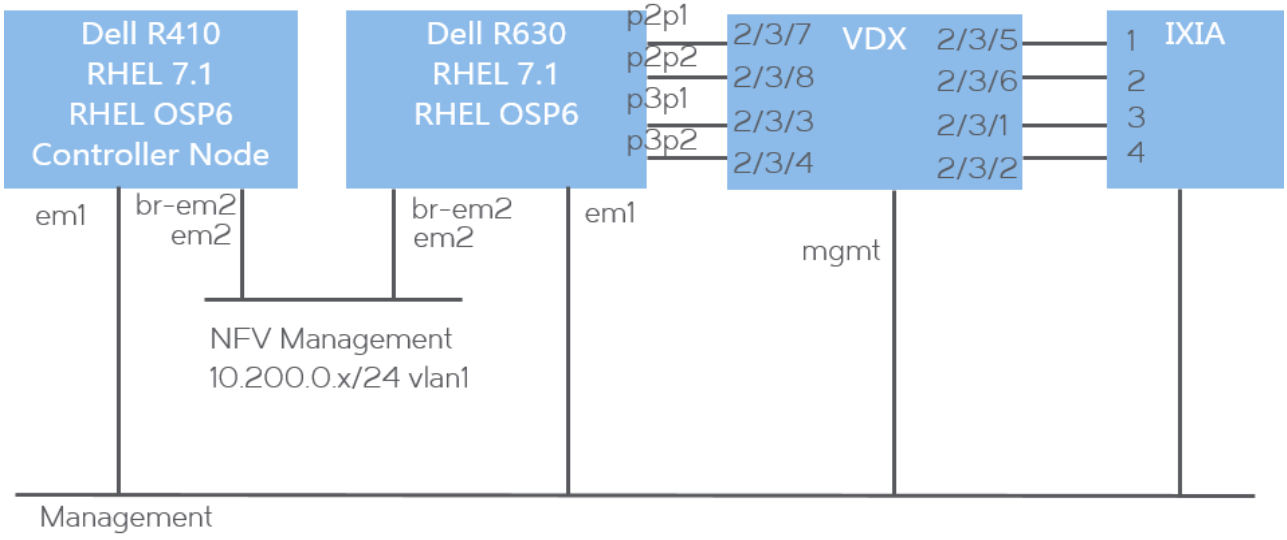
【検証項目】

VNF 管理	✓ Redhat Enterprise Linux OpenStack Platform 6 の利用 ✓ SRIOV NIC 追加/削除 ✓ SRIOV NIC と CPU ピンニング 設定がされた VM の起動
vCPE 集約度	1 サーバあたりの収容 vRouter 数（vRouter あたりのリソース） ✓ 2 vCPUs (1 vCPU for control plane /1 vCPU for dataplane) ✓ 2GB メモリ ✓ 3 NIC (2 SRIOV NIC and 1 virtio NIC for management)
パフォーマンス	IMIX フレームサイズ下の双方向 40Gbps* スループット * IMIX フレーズサイズ下の 40Gbps のワイヤーレートパフォーマンスは 38.68Gbps.
障害検知	BFD for BGP の利用

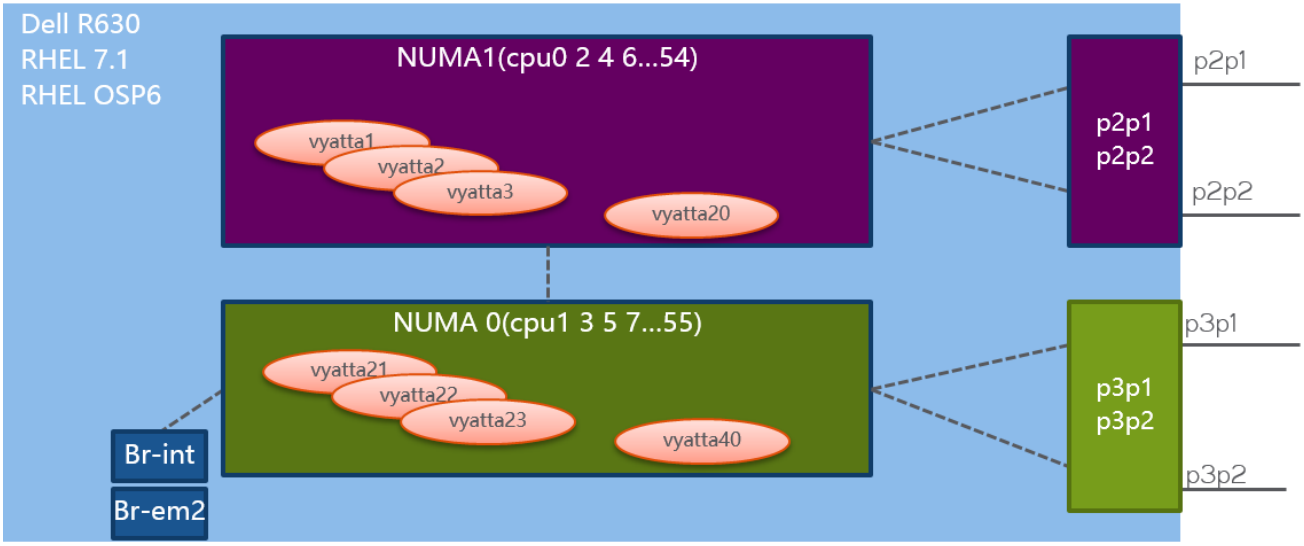
【検証で使用した部材】

Hardware: Dell Poweredge R630 for NFVI ➢ E5 2695V3 2.3GHz x2 ➢ 28 core (56 threads) ➢ 192 GB memory ➢ 4 onboard GE ➢ 4 x 10GE (2x 2port-10GE Intel 520) ➢ 4 x 10GE SFP+ SR Module Dell R410 for OpenStack Controller ➢ 8GB memory IXIA for traffic generator ➢ 10GE x 4(SFP+) VDX 8770(NOS6.0.1) for BFD router	**Software:** RHEL7OSP6A3 1) 28 vRouters per host with CPU pinning 2) 40 vRouters per host without CPU pinning * 2 vCPU and 4GB memory per vRouter instance

【物理接続構成】



Numa 構成：
各 vRouter インスタンスは、NUMA ノードを跨がないように CPU Pinning を行いました。



SRIOV/PCI パススルー設定：

- BIOS で SRIOV/PCI パススルーを Enable
- Grub 設定に IOMMU を追加
  - intel_iommu=on
- 10GE ドライバにて、SRIOV を有効化
  - # modprobe –r ixgbe
  - # modprobe ixgbe max_vfs=21

ネットワーク（Neutron/ML2）設定：

```
[ml2]
type_drivers = vlan
mechanism_drivers = openvswitch, sriovnicswitch
[ml2_type_vlan]
network_vlan_ranges =physnet1:1:100,sriovnet1:101:120,sriovnet2:121:140,sriovnet3:141:160,sriovnet4:161:180
[ml2_sriov]
supported_pci_vendor_devs = 8086:10ed
agent_required = False
```

Nfv management networks:

```
neuron net-create mgmt --provider:physical_network=physnet1 --provider:network_type=vlan --provider:segmentation_id=501
neuron subnet-create mgmt 10.200.0.0/24 –name mgmt.
```

Data networks:

```
neutron net-create vnet141 --provider:physical_network=sriovnet3 --provider:network_type=vlan --provider:segmentation_id=141
...
```

Data ports:

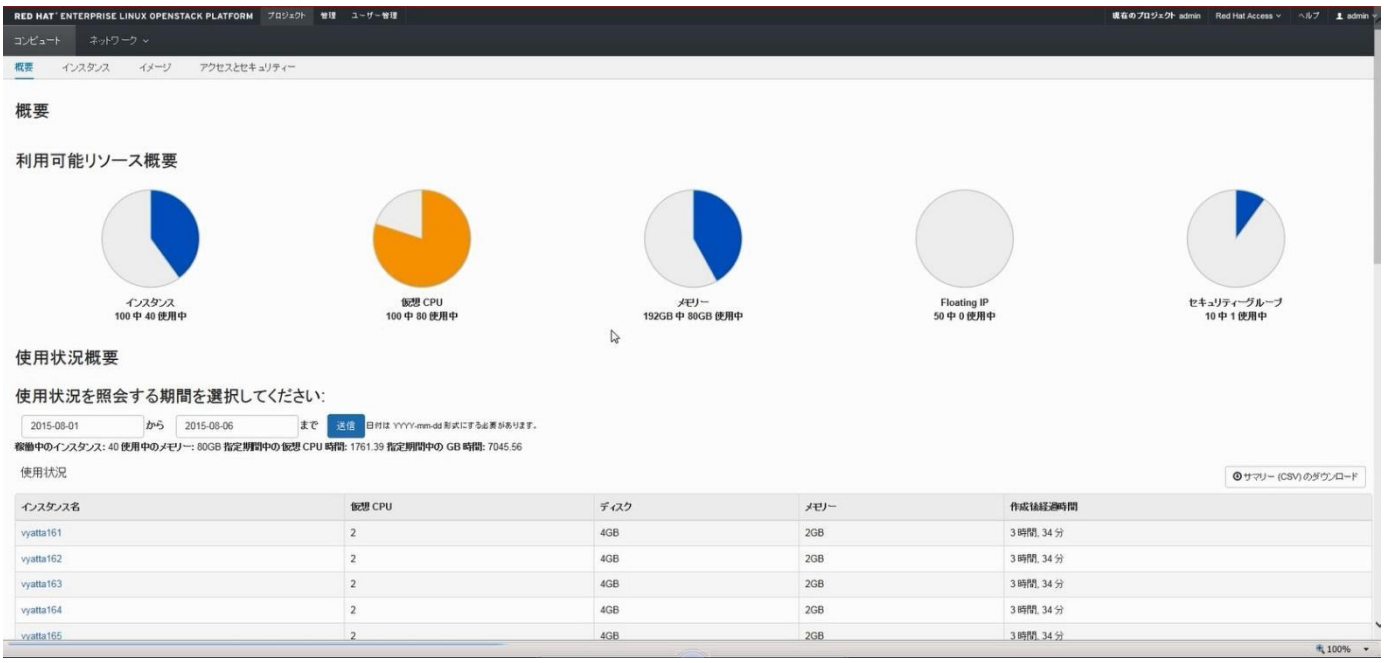
```
neutron port-create vnet141 --binding:vnic-type direct --name port141
…
```

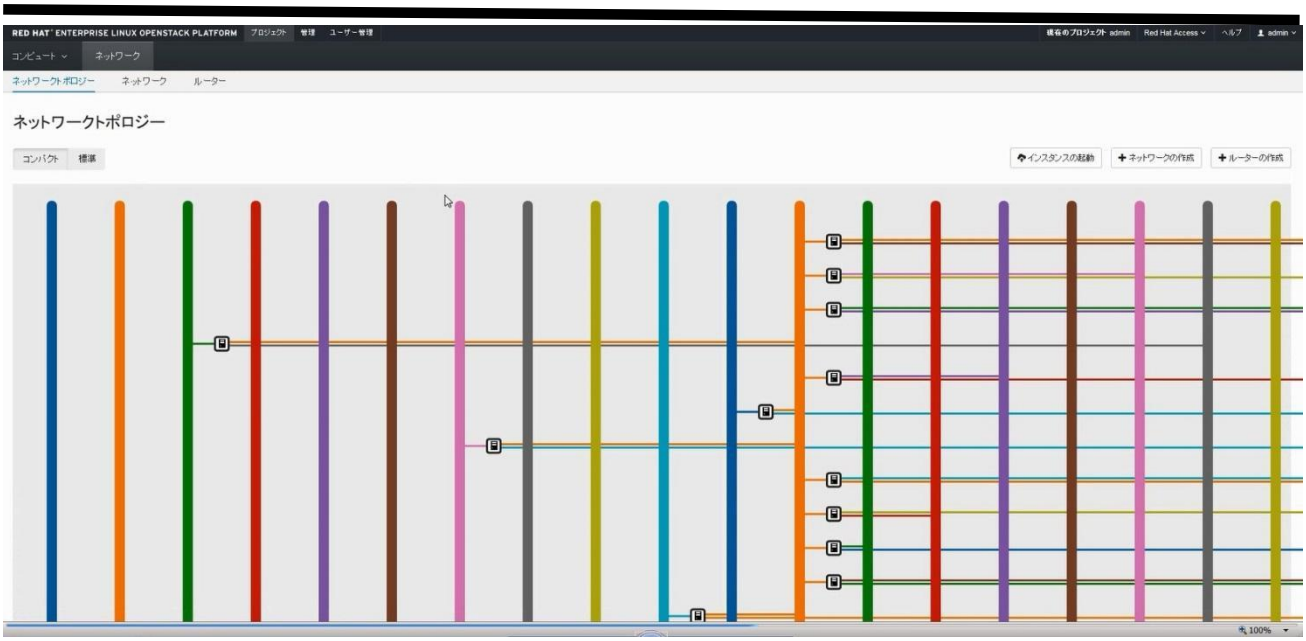
インスタンス（Nova）設定：
Compute node
#vi nova.conf
[DEFAULT]
….
pci_passthrough_whitelist = {"address":"04:14.6","physical_network":"sriovnet3"}
pci_passthrough_whitelist = {"address":"04:14.4","physical_network":"sriovnet3"}
pci_passthrough_whitelist = {"address":"04:14.2","physical_network":"sriovnet3"}

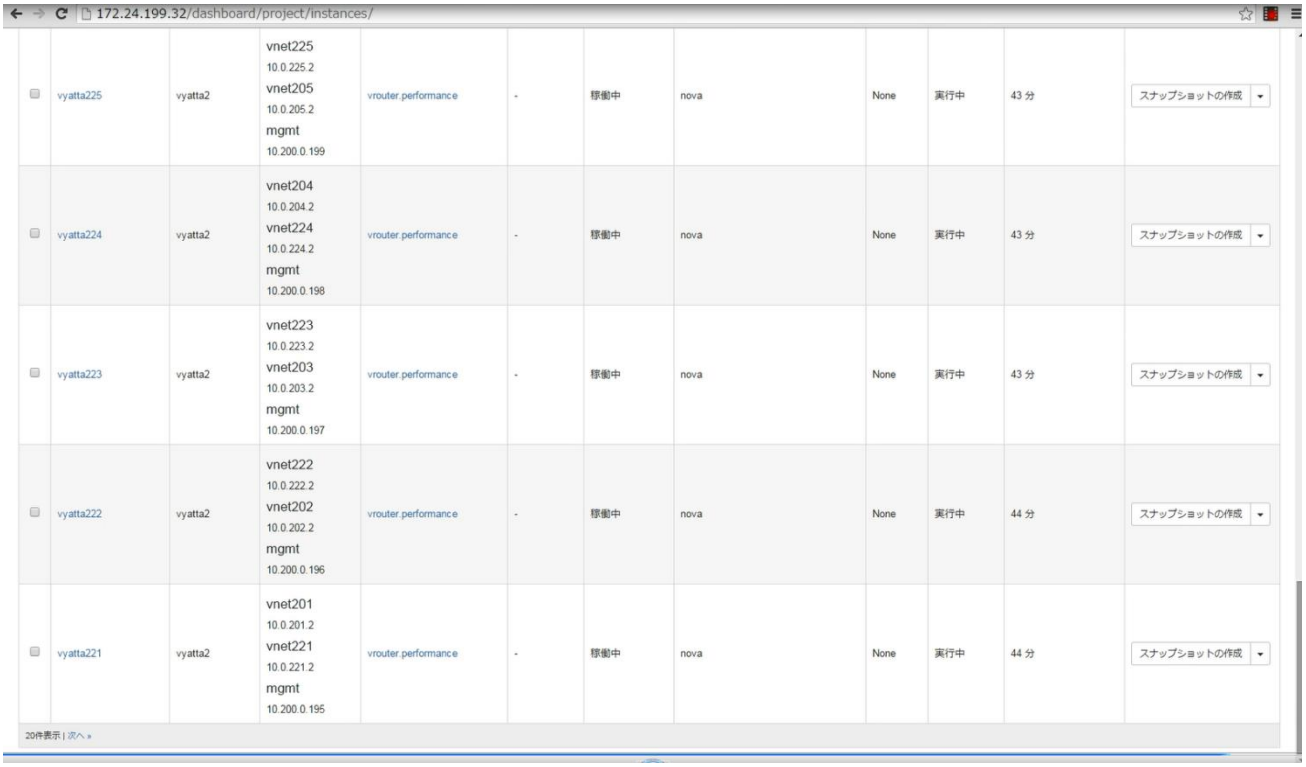
nova aggregate-create vrouter
nova aggregate-set-metadata 1 pinned=true
nova flavor-create vrouter 6 2048 4 2
nova flavor-key 6 set hw:cpu_policy=oversub
nova flavor-key 6 set aggregate_instance_extra_specs:pinned=true

nova boot --flavor vrouter --nic port-id= [dp0s6] --nic port-id=b6aa845e-c2bc-4fa3-85fb-b9c8907b7c9e[dp0s5] --nic net-id=1985b35a-b59a-4d71-9264-ccb200ab537a[dp0s3] --image vyatta vyatta141

10. リソースの割当てとネットワーク設定

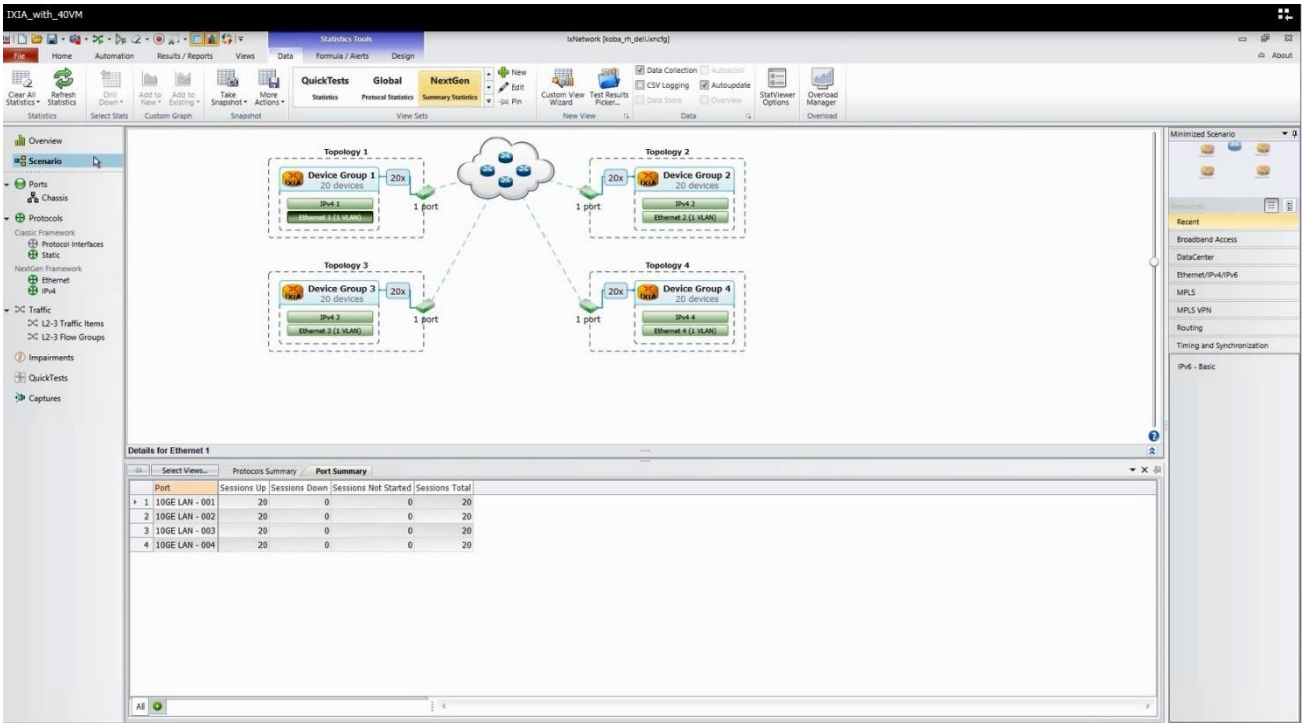


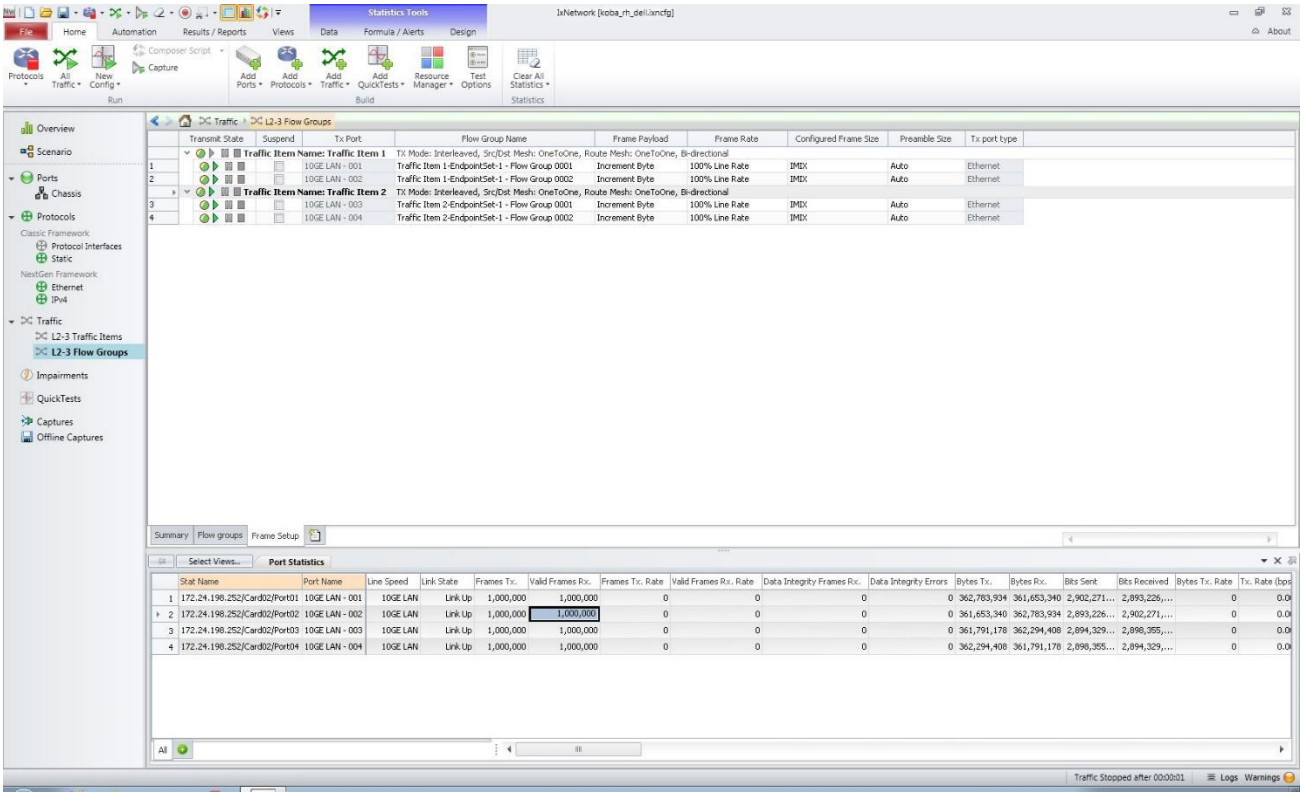

RED HAT ENTERPRISE LINUX OPENSTACK PLATFORM
プロジェクト
管理
ユーザー管理
admin
コンピュート
ネットワーク
ネットワークトポロジー
ネットワーク
ルーター
ネットワークトポロジー
コンパクト
標準
インスタンスの起動
ネットワークの作成
ルーターの作成
100%

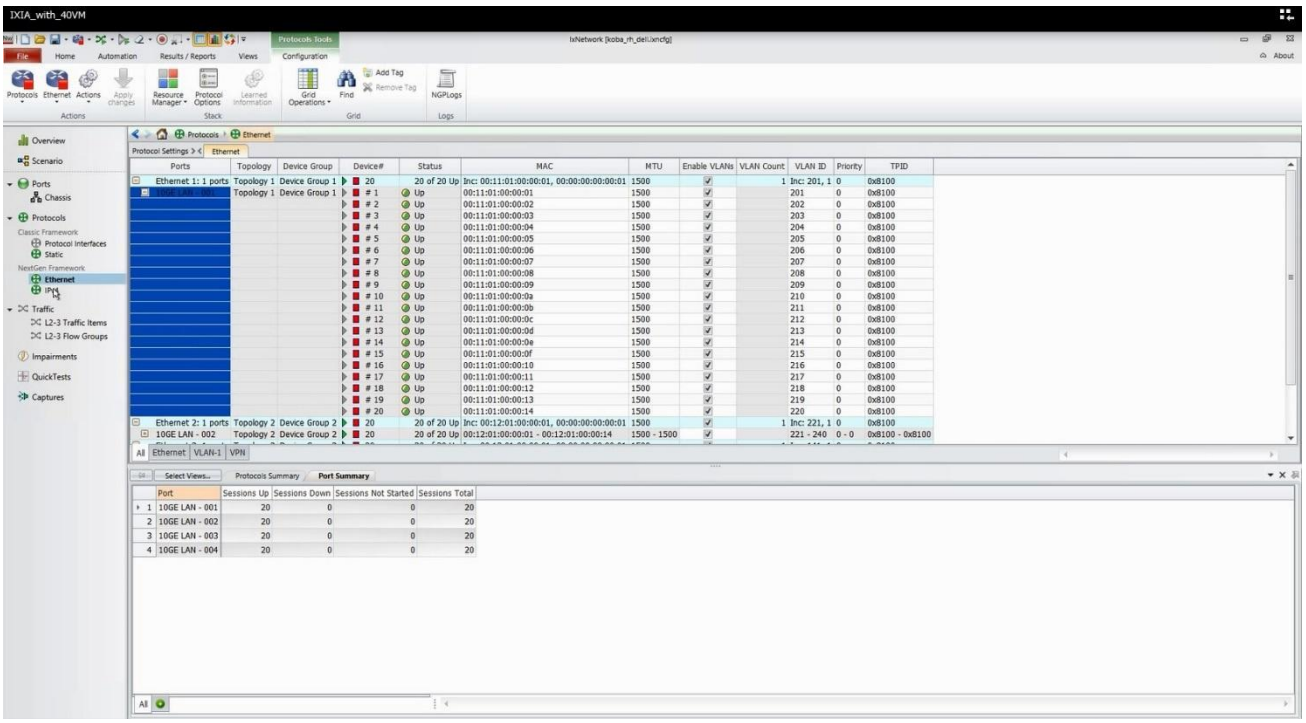

172.24.199.32/dashboard/project/instances/
vyatta225 vyatta2 vnet225 10.0.225.2 vnet205 10.0.205.2 mgmt 10.200.0.199 vrouter.performance - 稼働中 nova None 実行中 43 分 スナップショットの作成
vyatta224 vyatta2 vnet204 10.0.204.2 vnet224 10.0.224.2 mgmt 10.200.0.198 vrouter.performance - 稼働中 nova None 実行中 43 分 スナップショットの作成
vyatta223 vyatta2 vnet223 10.0.223.2 vnet203 10.0.203.2 mgmt 10.200.0.197 vrouter.performance - 稼働中 nova None 実行中 43 分 スナップショットの作成
vyatta222 vyatta2 vnet222 10.0.222.2 vnet202 10.0.202.2 mgmt 10.200.0.196 vrouter.performance - 稼働中 nova None 実行中 44 分 スナップショットの作成
vyatta221 vyatta2 vnet201 10.0.201.2 vnet221 10.0.221.2 mgmt 10.200.0.195 vrouter.performance - 稼働中 nova None 実行中 44 分 スナップショットの作成
20件表示 | 次へ »

11. トラフィック設定

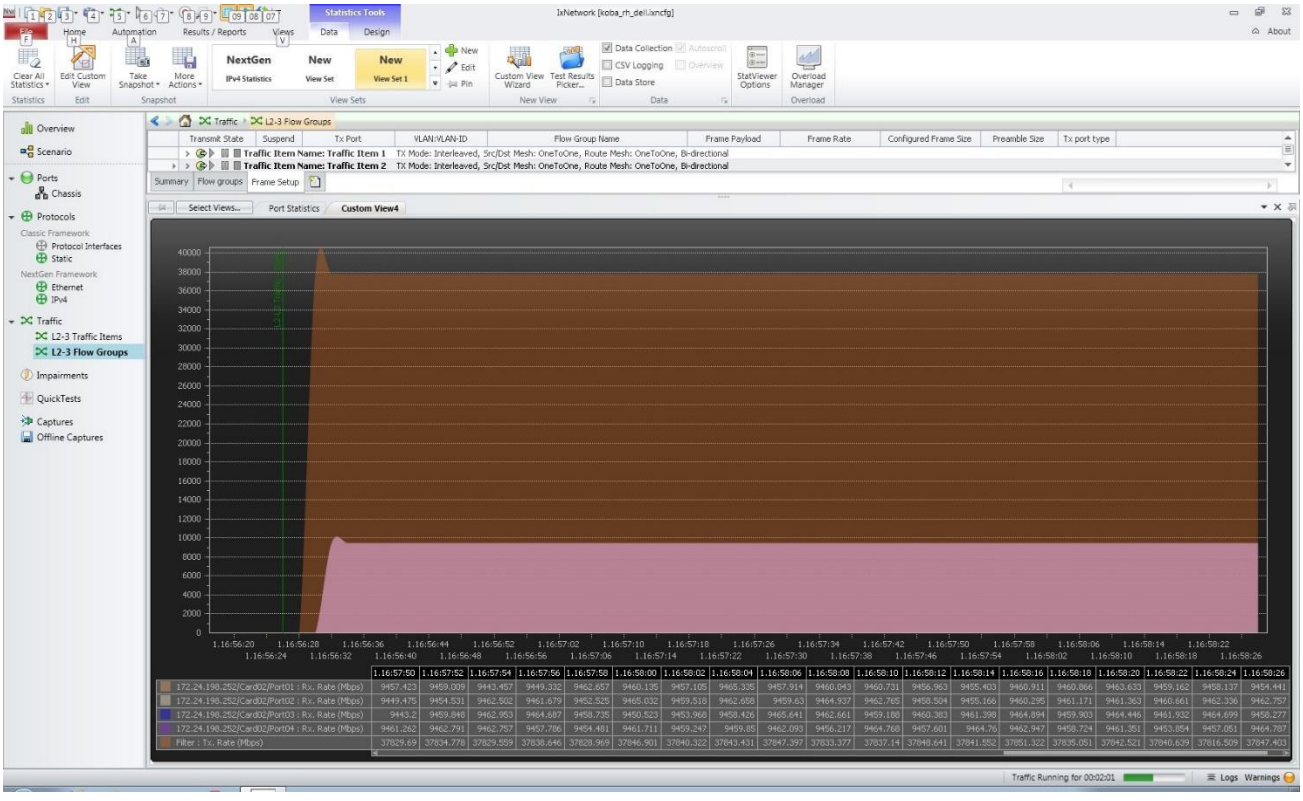
IXIA にて、10GE を 4 ポート利用し、各ポートから IMIX サイズのフレームを 100%ワイヤレートで送信し、各 vRouter に対して 1Gbps のトラフィックが流れるようにしました。







【性能試験結果】
10GE x 4 ポートで 100%のスループットを確認しました。（IMIX におけるフレームサイズで、40G のワイヤレートは 38.68Gbps）



12. ネットワークの障害検知

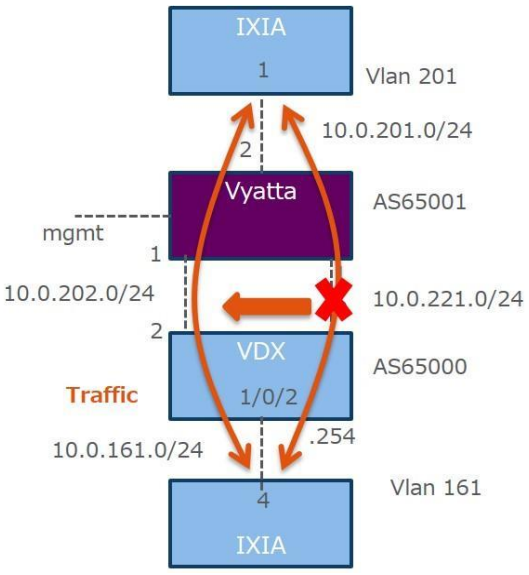
【BFD とは？（Bidirectional Forwarding Detection）】

- ➢ 統合された高速な障害検知メカニズム
  - ✓ リンク障害の検知
  - ✓ IP レベルでの接続性モニタリング
  - ✓ 各種ルーティングプロトコルに対応（BGP/OSPF/Static）
- ➢ ルータ間で BFD パケットを定期的に送信
  - ✓ 一定時間内に BFD パケットの受信がされない場合、リンク障害とみなす
- ➢ 障害時のネットワーク収束時間を大幅に削減

【検証内容】

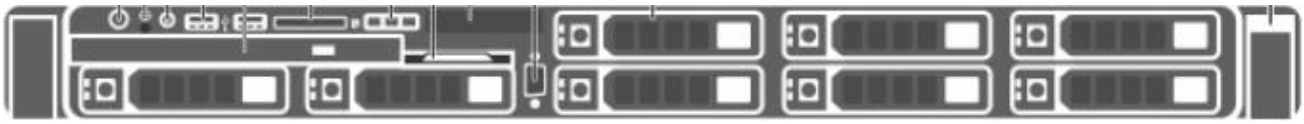
- ➢ Vyatta と VDX（VE インタフェース）間で BFD for BGP セッションを 2 本
  - ✓ BFD の設定：Interval 300ms、Multipler 3
  - ✓ BFD for BGP hold timer(VDX)：1 秒
- ➢ テスター（IXIA）で 100 パケット/秒のトラフィックを流し、リンク切断時のダウンタイムを計測
- ➢ BFD 有りの場合と無しの場合を計測

BFD	切り替わり時間
BFD for BGP 有り	3 秒
BFD for BGP 無し	90　秒



13. 高密度メインストリームラックサーバ/Dell PowerEdge R630

- 高密度、高パフォーマンスを実現する 1u 2 ソケットメインストリームラックサーバ
- 最新 Intel Xeon E5-2600v3 を 2 ソケット、最大 36 コアを搭載
- DDR4 メモリをサポート、最大 1.5TB のメモリを搭載可能
- 高パフォーマンスを実現するために最適化されたエアフローと電源ユニット
- PCIe バス直結の NVMe SSD をはじめとする豊富な SSD ラインナップ
- 高速 10Gb、40Gb をサポートする多彩なネットワークオプション
- NFV を実現するための必要な機能をサポート (CPU Pinning, DPDK Packet forwarding)
- iDRAC8 による洗練された管理機能



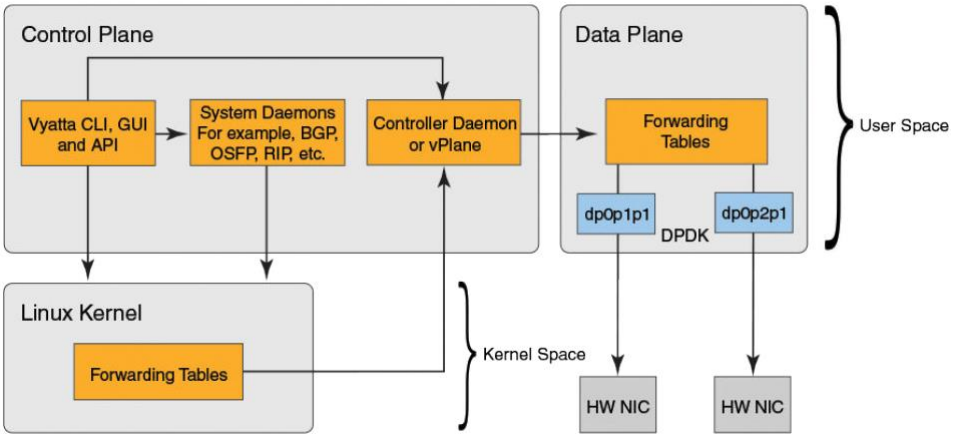
14. Brocade vRouter 5600

Brocade 5600 vRouter は、Network Functions Virtualization（NFV）の要求に対応し、ハードウェア・ネットワーク機器が持つ信頼性や性能はそのままに、先進のルーティング機能をソフトウェアで実現する初の仮想ルータです。高度なルーティングに加えて、ステートフル・ファイアウォール、VPN などの高性能なネットワーク機能を提供します。

ブロケードの 5600 vRouter は、ブロケード最新の vPlane テクノロジを採用し、ルータのコントロール・プレーンをデータ・プレーンから分離し、ハードウェア機器に匹敵するルーティング性能をソフトウェア・ベースの製品で実現しています。Brocade 5600 vRouter は、キャリア・クラスの性能と信頼性をソフトウェアで提供し、ネットワークインフラ構築の経済性を高め、NFV のための性能や拡張性、および耐障害性にも対処します。

【利用例】
Brocade 5600 vRouter は、高性能な拡張が必要とされる NFV のために設計され、次のような様々な用途に使用することができます。
➢ 仮想ファイアウォール
➢ 仮想 VPN/NAT ゲートウェイ
➢ 仮想 CPE, 仮想 CE
➢ 仮想ルート・リフレクタ
➢ 高パフォーマンスの BGP ルーティング

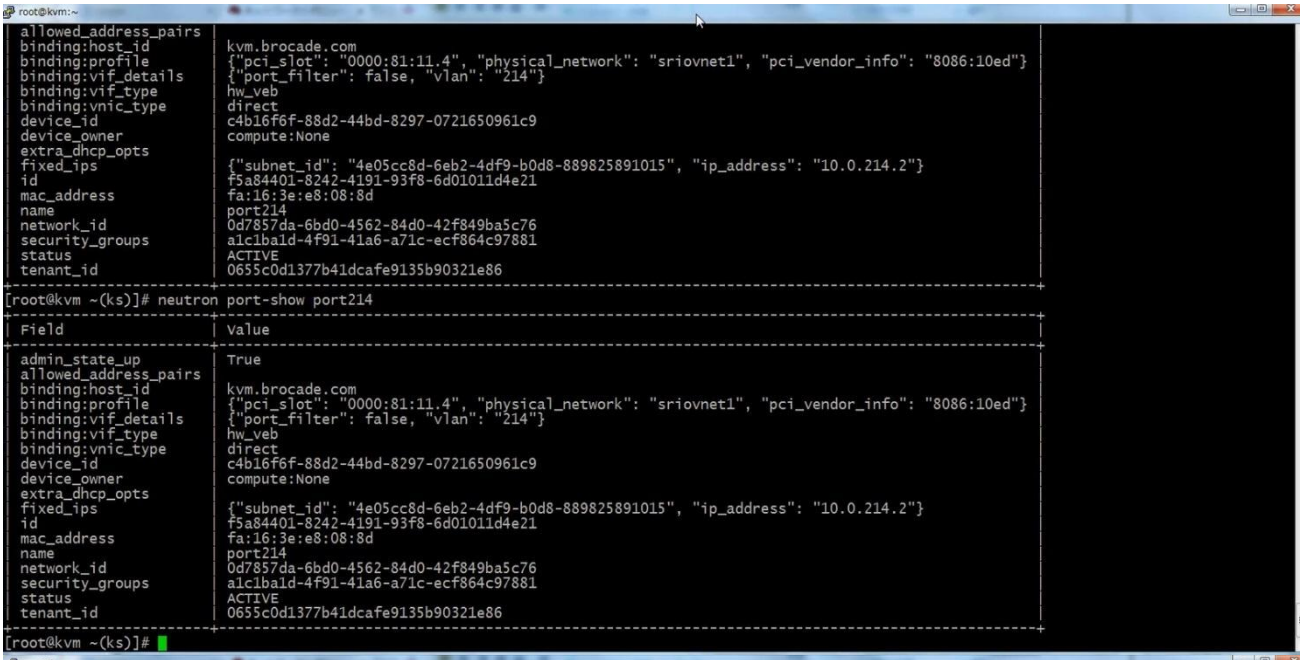
【概要】
➢ Routing/FW/NAT/VPN 機能
➢ DPDK-based vPlane Architecture
  ➢ コントロールプレーンと DPDK ベースのデータプレーンに分離
  ➢ vCPU2 個以上、メモリ 2GB 以上で動作
    ✧ コントロールプレーン: 1 vCPU
    ✧ データプレーン: 1 vCPU 以上
➢ データプレーンに割り当てる CPU 数に応じてリニアに性能が向上

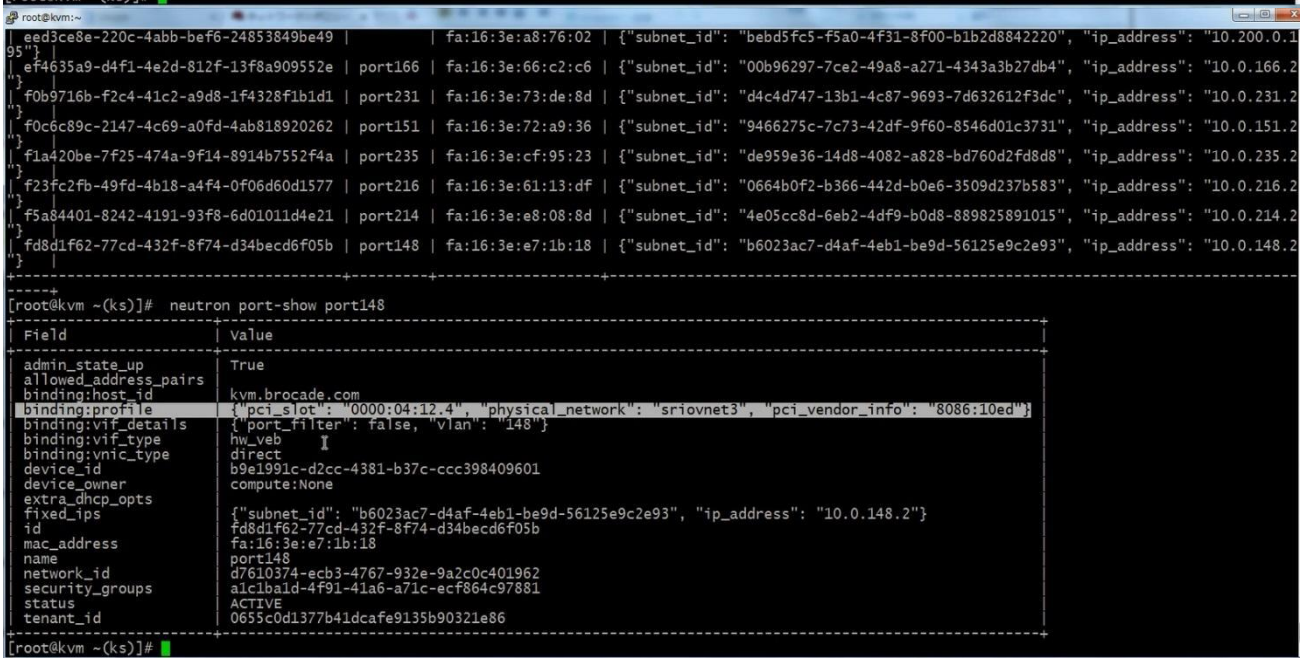


15. 検証結果と今後の展望

サーバ 1 台の上に 40 インスタンスの vCPE を稼働させた今回の検証実験では、40Gbps の性能を達成したことが確認されました。今回の検証実験により、「クラウド型 vCPE」を利用した NFV ソリューションとしては、業界に先駆けて高い水準の通信性能が実証されたことになり、この結果をレファレンスとしたサービス事業者における商用化に向けて大きな弾みとなります。今回の検証結果を受けて、レッドハット、デル、インテル、ブロケードの 4 社は、2016 年春頃の商用展開を目指して、4 社間で今後の協力関係や共通のチャネル構築について、協議を進めていく意向です。

【Appendix】
Neutron 設定詳細

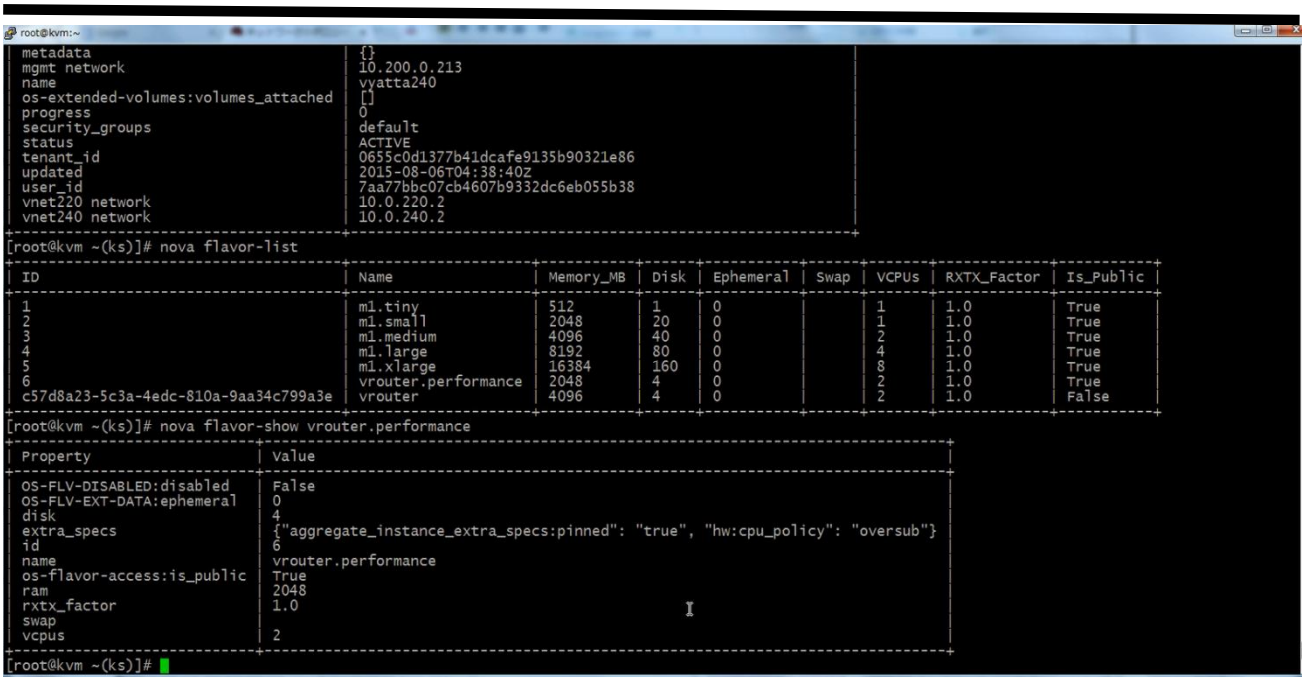




Nova 設定詳細

```
root@kvm:~
| 6595a8b6-991a-4395-99e3-f052f22501ab | vyatta237 | ACTIVE | -          | Running     | vnet237=10.0.237.2; vnet217=10.0.217.2; mgmt=10.200.0.210 |
| 2b2d7272-1ce5-4b5c-bfdc-99517f319aa9 | vyatta238 | ACTIVE | -          | Running     | vnet218=10.0.218.2; vnet238=10.0.238.2; mgmt=10.200.0.211 |
| 52c19ed9-92aa-45bc-b05d-13810a2f3d46 | vyatta239 | ACTIVE | -          | Running     | vnet219=10.0.219.2; vnet239=10.0.239.2; mgmt=10.200.0.212 |
| 0b6409b7-36a8-4353-b1d7-a8a20b57e03a | vyatta240 | ACTIVE | -          | Running     | vnet220=10.0.220.2; vnet240=10.0.240.2; mgmt=10.200.0.213 |
+--------------------------------------+-----------+--------+------------+-------------+-----------------------------------------------------------+
[root@kvm ~(ks)]# nova show vyatta240 | more
+-------------------------------------+----------------------------------------------------------+
| Property                            | Value                                                    |
+-------------------------------------+----------------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                                                   |
| OS-EXT-AZ:availability_zone         | nova                                                     |
| OS-EXT-SRV-ATTR:host                | kvm.brocade.com                                          |
| OS-EXT-SRV-ATTR:hypervisor_hostname | kvm.brocade.com                                          |
| OS-EXT-SRV-ATTR:instance_name       | instance-000001a2                                        |
| OS-EXT-STS:power_state              | 1                                                        |
| OS-EXT-STS:task_state               | -                                                        |
| OS-EXT-STS:vm_state                 | active                                                   |
| OS-SRV-USG:launched_at              | 2015-08-06T04:38:39.000000                               |
| OS-SRV-USG:terminated_at            | -                                                        |
| accessIPv4                          |                                                          |
| accessIPv6                          |                                                          |
| config_drive                        |                                                          |
| created                             | 2015-08-06T04:38:30Z                                     |
| flavor                              | vrouter.performance (6)                                  |
| hostId                              | 196bdf6a0e2b51d445b2338abfc211741ec008154862513ecc587583 |
| id                                  | 0b6409b7-36a8-4353-b1d7-a8a20b57e03a                     |
| image                               | vyatta2 (eb845f6d-02ae-4839-b75a-bcfea3f44b97)           |
| key_name                            | -                                                        |
| metadata                            | {}                                                       |
| mgmt network                        | 10.200.0.213                                             |
| name                                | vyatta240                                                |
| os-extended-volumes:volumes_attached | []                                                      |
| progress                            | 0                                                        |
| security_groups                     | default                                                  |
| status                              | ACTIVE                                                   |
| tenant_id                           | 0655c0d1377b41dcafe9135b90321e86                         |
| updated                             | 2015-08-06T04:38:40Z                                     |
| user_id                             | 7aa77bbc07cb4607b9332dc6eb055b38                         |
| vnet220 network                     | 10.0.220.2                                               |
| vnet240 network                     | 10.0.240.2                                               |
+-------------------------------------+----------------------------------------------------------+
[root@kvm ~(ks)]# nova f
```

```
root@kvm:~
| 42e69c7f-80c9-42b0-bf6c-5a2a7a74ff02 | vyatta161 | ACTIVE | - | Running | vnet161=10.0.161.2; vnet141=10.0.141.2; mgmt=10.200.0.175 |
| f8b4805b-e7ea-4605-9d9a-da0c4e7e7560 | vyatta162 | ACTIVE | - | Running | vnet162=10.0.162.2; vnet142=10.0.142.2; mgmt=10.200.0.176 |
| dfcadbc4-8260-42b1-a761-7e31bf7ba323 | vyatta163 | ACTIVE | - | Running | vnet163=10.0.163.2; vnet143=10.0.143.2; mgmt=10.200.0.177 |
| c2382a85-941e-4891-b5b7-76fd522ec68d | vyatta164 | ACTIVE | - | Running | vnet144=10.0.144.2; vnet164=10.0.164.2; mgmt=10.200.0.178 |
| 51d8640c-bba8-4ca8-ac89-758789f9b279 | vyatta165 | ACTIVE | - | Running | vnet145=10.0.145.2; vnet165=10.0.165.2; mgmt=10.200.0.179 |
| 972eb724-d619-49b8-b21d-82061a4f2ea5 | vyatta166 | ACTIVE | - | Running | vnet146=10.0.146.2; mgmt=10.200.0.180; vnet166=10.0.166.2 |
| 9bfb9c20-5ae6-4055-a146-9c189d5a89b6 | vyatta167 | ACTIVE | - | Running | vnet147=10.0.147.2; vnet167=10.0.167.2; mgmt=10.200.0.181 |
| b9e1991c-d2cc-4381-b37c-ccc398409601 | vyatta168 | ACTIVE | - | Running | vnet168=10.0.168.2; mgmt=10.200.0.182; vnet148=10.0.148.2 |
| 5de9ed98-9e08-49d6-9e8d-c6a0fe9e6333 | vyatta169 | ACTIVE | - | Running | vnet169=10.0.169.2; vnet149=10.0.149.2; mgmt=10.200.0.183 |
| 2992e216-05d2-4d23-af2c-3d7dcd5c2b70 | vyatta170 | ACTIVE | - | Running | vnet150=10.0.150.2; vnet170=10.0.170.2; mgmt=10.200.0.184 |
| 085e09c1-42bb-45a0-99ce-16fde8a0e1c0 | vyatta171 | ACTIVE | - | Running | mgmt=10.200.0.185; vnet151=10.0.151.2; vnet171=10.0.171.2 |
| b664f5d3-8278-471f-a938-53511eac3fd6 | vyatta172 | ACTIVE | - | Running | vnet172=10.0.172.2; vnet152=10.0.152.2; mgmt=10.200.0.186 |
| 20dad32b-e0fb-43fd-ac8f-b11b28556a8e | vyatta173 | ACTIVE | - | Running | mgmt=10.200.0.187; vnet173=10.0.173.2; vnet153=10.0.153.2 |
| 6a1b732d-571d-46c0-befe-8e6f746ab404 | vyatta174 | ACTIVE | - | Running | vnet154=10.0.154.2; vnet174=10.0.174.2; mgmt=10.200.0.188 |
| 3f4c68f7-f807-48f9-88bd-48765aff0ef6 | vyatta175 | ACTIVE | - | Running | vnet155=10.0.155.2; vnet175=10.0.175.2; mgmt=10.200.0.189 |
| c03cd131-111a-457e-afda-8d9de7a869c1 | vyatta176 | ACTIVE | - | Running | vnet176=10.0.176.2; vnet156=10.0.156.2; mgmt=10.200.0.190 |
| d4e6c91b-812a-466e-a705-3375a143d062 | vyatta177 | ACTIVE | - | Running | vnet177=10.0.177.2; vnet157=10.0.157.2; mgmt=10.200.0.191 |
| 0b4c0adb-7f07-45a9-bb78-9941dac5d674 | vyatta178 | ACTIVE | - | Running | vnet158=10.0.158.2; vnet178=10.0.178.2; mgmt=10.200.0.192 |
| e0332e1b-acb5-494c-b7a9-cc557752b964 | vyatta179 | ACTIVE | - | Running | mgmt=10.200.0.193; vnet159=10.0.159.2; vnet179=10.0.179.2 |
| 3c99ca0c-cdd3-4a4c-90c7-aa3db21ea4bd | vyatta180 | ACTIVE | - | Running | vnet160=10.0.160.2; vnet180=10.0.180.2; mgmt=10.200.0.194 |
| f5942197-6faa-4ffe-bb5f-1f4930182706 | vyatta221 | ACTIVE | - | Running | vnet201=10.0.201.2; vnet221=10.0.221.2; mgmt=10.200.0.195 |
| bc8213ff-dd32-4040-9c38-a024cd37d9e4 | vyatta222 | ACTIVE | - | Running | vnet222=10.0.222.2; vnet202=10.0.202.2; mgmt=10.200.0.196 |
| 2072bd32-79ef-4103-b8f5-316909d71d75 | vyatta223 | ACTIVE | - | Running | vnet223=10.0.223.2; vnet203=10.0.203.2; mgmt=10.200.0.197 |
| cca901f9-750b-4119-b1c2-3cf782dcf592 | vyatta224 | ACTIVE | - | Running | vnet204=10.0.204.2; vnet224=10.0.224.2; mgmt=10.200.0.198 |
| 1d72e23a-50ad-47eb-8589-aa68b163f4a3 | vyatta225 | ACTIVE | - | Running | mgmt=10.200.0.199; vnet205=10.0.205.2; vnet225=10.0.225.2 |
| 6adf2c51-2c26-475c-8c70-75943cd92cf4 | vyatta226 | ACTIVE | - | Running | vnet226=10.0.226.2; vnet206=10.0.206.2; mgmt=10.200.0.2 |
| 19a1cb18-b464-445a-8fb9-31f7ae688f66 | vyatta227 | ACTIVE | - | Running | mgmt=10.200.0.200; vnet227=10.0.227.2; vnet207=10.0.207.2 |
| cc1ec0c0-a034-4263-a708-52eee6d8b23e | vyatta228 | ACTIVE | - | Running | vnet208=10.0.208.2; vnet228=10.0.228.2; mgmt=10.200.0.201 |
| f9f3240b-f221-4fca-b17f-056fd84d15d2 | vyatta229 | ACTIVE | - | Running | vnet209=10.0.209.2; vnet229=10.0.229.2; mgmt=10.200.0.202 |
| 8ee05d73-7eca-4a58-864d-a1b29659baaa | vyatta230 | ACTIVE | - | Running | vnet210=10.0.210.2; vnet230=10.0.230.2; mgmt=10.200.0.203 |
| a0d06c31-2665-4f28-a663-a0ca677a56db | vyatta231 | ACTIVE | - | Running | vnet211=10.0.211.2; vnet231=10.0.231.2; mgmt=10.200.0.204 |
| 4e7cecba-c9ba-4e58-95fc-9a81fd2eec21 | vyatta232 | ACTIVE | - | Running | vnet212=10.0.212.2; mgmt=10.200.0.205; vnet232=10.0.232.2 |
| 4a8b8226-d518-41f6-90d7-86d5e928234c | vyatta233 | ACTIVE | - | Running | vnet213=10.0.213.2; vnet233=10.0.233.2; mgmt=10.200.0.206 |
| c4b16f6f-88d2-44bd-8297-0721650961c9 | vyatta234 | ACTIVE | - | Running | vnet234=10.0.234.2; mgmt=10.200.0.207; vnet214=10.0.214.2 |
| 8fe21e5f-7c11-463c-a1c2-ae82966c5bbb | vyatta235 | ACTIVE | - | Running | vnet235=10.0.235.2; vnet215=10.0.215.2; mgmt=10.200.0.208 |
| 5436a831-9417-4af1-b0df-81dcb693d0c4 | vyatta236 | ACTIVE | - | Running | vnet236=10.0.236.2; vnet216=10.0.216.2; mgmt=10.200.0.209 |
| 6595a8b6-991a-4395-99e3-f052f22501ab | vyatta237 | ACTIVE | - | Running | vnet237=10.0.237.2; vnet217=10.0.217.2; mgmt=10.200.0.210 |
| 2b2d7272-1ce5-4b5c-bfdc-99517f319aa9 | vyatta238 | ACTIVE | - | Running | vnet218=10.0.218.2; vnet238=10.0.238.2; mgmt=10.200.0.211 |
| 52c19ed9-92aa-45bc-b05d-13810a2f3d46 | vyatta239 | ACTIVE | - | Running | vnet219=10.0.219.2; vnet239=10.0.239.2; mgmt=10.200.0.212 |
| 0b6409b7-36a8-4353-b1d7-a8a20b57e03a | vyatta240 | ACTIVE | - | Running | vnet220=10.0.220.2; vnet240=10.0.240.2; mgmt=10.200.0.213 |
+--------------------------------------+-----------+--------+---+---------+-----------------------------------------------------------+
[root@kvm ~(ks)]# nova
```



```
| metadata                            | {}
| mgmt network                        | 10.200.0.213
| name                                | vyatta240
| os-extended-volumes:volumes_attached | []
| progress                            | 0
| security_groups                     | default
| status                              | ACTIVE
| tenant_id                           | 0655c0d1377b41dcafe9135b90321e86
| updated                             | 2015-08-06T04:38:40Z
| user_id                             | 7aa77bbc07cb4607b9332dc6eb055b38
| vnet220 network                     | 10.0.220.2
| vnet240 network                     | 10.0.240.2
+-------------------------------------+----------------------------------------------------------+
[root@kvm ~(ks)]# nova flavor-list
+--------------------------------------+---------------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID                                   | Name                | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+--------------------------------------+---------------------+-----------+------+-----------+------+-------+-------------+-----------+
| 1                                    | m1.tiny             | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2                                    | m1.small            | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3                                    | m1.medium           | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4                                    | m1.large            | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5                                    | m1.xlarge           | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| 6                                    | vrouter.performance | 2048      | 4    | 0         |      | 2     | 1.0         | True      |
| c57d8a23-5c3a-4edc-810a-9aa34c799a3e | vrouter             | 4096      | 4    | 0         |      | 2     | 1.0         | False     |
+--------------------------------------+---------------------+-----------+------+-----------+------+-------+-------------+-----------+
[root@kvm ~(ks)]# nova flavor-show vrouter.performance
+----------------------------+-------------------------------------------------------------------------------------+
| Property                   | Value                                                                               |
+----------------------------+-------------------------------------------------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                                                               |
| OS-FLV-EXT-DATA:ephemeral  | 0                                                                                   |
| disk                       | 4                                                                                   |
| extra_specs                | {"aggregate_instance_extra_specs:pinned": "true", "hw:cpu_policy": "oversub"}       |
| id                         | 6                                                                                   |
| name                       | vrouter.performance                                                                 |
| os-flavor-access:is_public | True                                                                                |
| ram                        | 2048                                                                                |
| rxtx_factor                | 1.0                                                                                 |
| swap                       |                                                                                     |
| vcpus                      | 2                                                                                   |
+----------------------------+-------------------------------------------------------------------------------------+
[root@kvm ~(ks)]#
```

BFD 設定（vRouter）

```
vyatta@vyatta# sh pro
 protocols {
     bfd {
         destination 10.0.202.254 {
             source 10.0.202.2 {
                 template test
             }
         }
         destination 10.0.221.254 {
             source 10.0.221.2 {
                 template test
             }
         }
         template test {
             minimum-rx 300
             minimum-tx 300
             multiplier 3
         }
     }
     bgp 65001 {
         address-family {
             ipv4-unicast {
                 network 10.0.201.0/24
             }
         }
         neighbor 10.0.202.254 {
             address-family {
                 ipv4-unicast

             }
             fall-over {
                 bfd
             }
             remote-as 65000
         }
         neighbor 10.0.221.254 {
             address-family {
                 ipv4-unicast
             }
             fall-over {
                 bfd
             }
             remote-as 65000
             route-map {
                 import test
             }
```

BFD 設定（VDX）

```
sw0(config-bgp-router)# do sh ru rb 2 router bgp
rbridge-id 2
 router bgp
  local-as 65000
  neighbor 10.0.202.2 remote-as 65001
  neighbor 10.0.202.2 bfd
  neighbor 10.0.202.2 bfd holdover-interval 1
  neighbor 10.0.202.2 bfd interval 300 min-rx 300 multiplier 3
  neighbor 10.0.221.2 remote-as 65001
  neighbor 10.0.221.2 bfd
  neighbor 10.0.221.2 bfd holdover-interval 1
  neighbor 10.0.221.2 bfd interval 300 min-rx 300 multiplier 3
  address-family ipv4 unicast
   network 10.0.161.0/24
   neighbor 10.0.221.2 activate
   neighbor 10.0.221.2 route-map in test
   neighbor 10.0.202.2 activate
  !
  address-family ipv6 unicast
  !
 !
```