KVM を使用した ASAv の導入

カーネルベースの仮想マシン（KVM）を実行できる任意のサーバークラスの x86 CPU デバイスに ASAv を導入できます。

重要 ASAv の最小メモリ要件は 2GB です。現在の ASAv が 2GB 未満のメモリで動作している場合、ASAv マシンのメモリを増やさないと、以前のバージョンから 9.13(1) 以降にアップグレードできません。また、最新バージョンを使用して新しい ASAv マシンを再導入できます。

- 注意事項と制約事項 （1 ページ）
- 概要 （4 ページ）
- 前提条件 （4 ページ）
- 第 0 日のコンフィギュレーション ファイルの準備 （5 ページ）
- 仮想ブリッジ XML ファイルの準備 （7 ページ）
- ASAv の導入 （9 ページ）
- ホットプラグ インターフェイス プロビジョニング （10 ページ）
- パフォーマンスの調整 （12 ページ）
- CPU 使用率とレポート （24 ページ）

注意事項と制約事項

ASAv の導入に使用される特定のハードウェアは、導入されるインスタンスの数や使用要件によって異なります。作成する各仮想アプライアンスには、ホストマシン上での最小リソース割り当て（メモリ、CPU 数、およびディスク容量）が必要です。

重要 ASAv は、8GB のディスクストレージサイズで導入されます。ディスク容量のリソース割り当てを変更することはできません。

（注） ASAv バージョン 9.16.x 以降で、デバイス構成が 16 vCPU および 32GB RAM の ASAv100 から ASAv10 にダウングレードする場合は、デバイス構成を 1 vCPU および 4GB RAM にする必要があります。

ASAv を導入する前に、次のガイドラインと制限事項を確認します。

KVM での ASAv のシステム要件

最適なパフォーマンスを確保するために、以下の仕様に準拠していることを確認してください。ASAv には、次の要件があります。

- ホスト CPU は、仮想化拡張機能を備えたサーバークラスの x86 ベースの Intel または AMD CPU である必要があります。

  たとえば、ASAv パフォーマンステストラボでは、2.6GHz で動作する Intel® Xeon® CPU E5-2690v4 プロセッサを搭載した Cisco Unified Computing System™（Cisco UCS®）C シリーズ M4 サーバーを最低限使用しています。

推奨される vNIC

最適なパフォーマンスを得るためには、次の vNIC を推奨します。

- PCI パススルーでの i40e：サーバーの物理 NIC を VM に関連付け、DMA（ダイレクトメモリアクセス）を介して NIC と VM の間でパケットデータを転送します。パケットの移動に CPU サイクルは必要ありません。
- i40evf/ixgbe-vf：実質的に上記と同じですが（NIC と VM 間の DMA パケット）、NIC を複数の VM 間で共有できます。SR-IOV は、導入の柔軟性が高いため、一般的に推奨されます。参照先
- virtio：10Gbps の動作をサポートしますが、CPU サイクルも必要な準仮想化ネットワークドライバです。

パフォーマンスの最適化

ASAv の最高のパフォーマンスを実現するために、VM とホストの両方を調整することができます。詳細については、パフォーマンスの調整 （12 ページ）を参照してください。

- **NUMA**：ゲスト VM の CPU リソースを単一の Non-Uniform Memory Access（NUMA）ノードに分離することで、ASAv のパフォーマンスを向上できます。詳細については、NUMA のガイドライン （14 ページ）を参照してください。
- **Receive Side Scaling**：ASAv は Receive Side Scaling（RSS）をサポートしています。これは、ネットワークアダプタによって複数のプロセッサコアにネットワーク受信トラフィックを分散するために使用されるテクノロジーです。詳細については、「Receive Side Scaling（RSS）用の複数の RX キュー （16 ページ）」を参照してください。

- **VPN の最適化**：ASAv で VPN パフォーマンスを最適化するための追加の考慮事項については、VPN の最適化 （19 ページ）を参照してください。

CPU ピニング

KVM 環境で ASAv を機能させるには、CPU ピニングが必要です。CPU ピンニングの有効化（12 ページ）を参照してください。

ハイ アベイラビリティ ガイドラインのためのフェールオーバー

フェールオーバー配置の場合は、スタンバイ装置が同じライセンス権限付与を備えていることを確認してください（たとえば、両方の装置が 2Gbps の権限付与であることなど）。

重要 ASAv を使用して高可用性ペアを作成する場合は、データインターフェイスを各 ASAv に同じ順序で追加する必要があります。完全に同じインターフェイスが異なる順序で各 ASAv に追加されると、ASAv コンソールにエラーが表示されることがあります。また、フェールオーバー機能にも影響が出ることがあります。

Proxmox VE 上の ASAv

Proxmox Virtual Environment（VE）は、KVM 仮想マシンを管理できるオープンソースのサーバー仮想化プラットフォームです。Proxmox VE は、Web ベースの管理インターフェイスも提供します。

Proxmox VE に ASAv を導入する場合は、エミュレートされたシリアルポートを持つように VM を設定する必要があります。シリアルポートがないと、ブートアッププロセス中に ASAv がループ状態になります。すべての管理タスクは、Proxmox VE Web ベース管理インターフェイスを使用して実行できます。

（注） Unix シェルまたは Windows Powershell に慣れている上級ユーザー向けに、Proxmox VE は仮想環境のすべてのコンポーネントを管理するコマンド ライン インターフェイスを提供します。このコマンドラインインターフェイスには、インテリジェントなタブ補完機能と UNIX の man ページ形式の完全なドキュメントがあります。

ASAv を正しく起動するには、VM にシリアルデバイスを設定する必要があります。

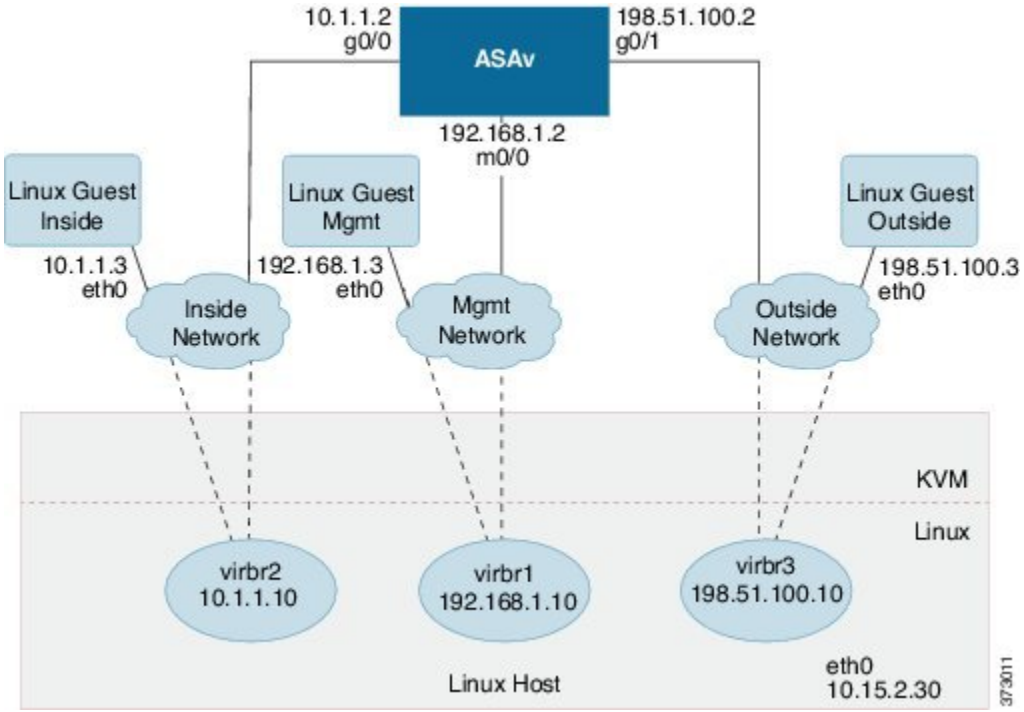
1. メイン Management Center の左側のナビゲーションツリーで ASAv マシンを選択します。
2. 仮想マシンの電源をオフにします。
3. **Hardware** > **Add** > **Network Device** を選択して、シリアルポートを追加します。
4. 仮想マシンの電源をオンにします。
5. Xterm.js を使用して ASAv マシンにアクセスします。

ゲスト/サーバーで端末をセットアップしてアクティブ化する方法については、Proxmox シリアル端末のページを参照してください。

概要

次の図は、ASAv と KVM のネットワークトポロジの例を示します。この章で説明している手順は、このトポロジの例に基づいています。ASAv は、内部ネットワークと外部ネットワークの間のファイアウォールとして動作します。また、別個の管理ネットワークが設定されます。

図 *1 : KVM* を使用した *ASAv* の導入例



前提条件

- Cisco.com から ASAv qcow2 ファイルをダウンロードし、Linux ホストに格納します。

  http://www.cisco.com/go/asa-software

（注） Cisco.com のログインおよびシスコ サービス契約が必要です。

- このマニュアルの導入例では、ユーザーが Ubuntu 18.04 LTS を使用していることを前提としています。Ubuntu 18.04 LTS ホストの最上部に次のパッケージをインストールします。
  - qemu-kvm

- libvirt bin
- bridge-utils
- Virt-Manager
- virtinst
- virsh tools
- genisoimage

- パフォーマンスはホストとその設定の影響を受けます。ホストを調整することで、KVM での ASAv のスループットを最大化できます。一般的なホスト調整の概念については、『NFV Delivers Packet Processing Performance with Intel』を参照してください。
- Ubuntu 18.04 の便利な最適化には、次のものが含まれます。
  - macvtap：高性能の Linux ブリッジ。Linux ブリッジの代わりに macvtap を使用できます。ただし、Linux ブリッジの代わりに macvtap を使用する場合は、特定の設定を行う必要があります。
  - Transparent Huge Pages：メモリページサイズを増加させます。Ubuntu 18.04 では、デフォルトでオンになっています。

    Hyperthread disabled：2 つの vCPUを 1 つのシングル コアに削減します。
  - txqueuelength：デフォルトの txqueuelength を 4000 パケットに増加させ、ドロップ レートを低減します。
  - pinning：qemu および vhost プロセスを特定のCPU コア にピン接続します。特定の条件下では、ピン接続によってパフォーマンスが大幅に向上します。
- RHEL ベースのディストリビューションの最適化については、『Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide』を参照してください。
- ASA ソフトウェアおよび ASAv ハイパーバイザの互換性については、Cisco ASA の互換性 [英語] を参照してください。

第 0 日のコンフィギュレーション ファイルの準備

ASAv を起動する前に、第 0 日用のコンフィギュレーション ファイルを準備できます。このファイルは、ASAv の起動時に適用される ASAv の設定を含むテキストファイルです。この初期設定は、「day0-config」というテキスト ファイルとして指定の作業ディレクトリに格納され、さらに day0.iso ファイルへと処理されます。この day0.iso ファイルが最初の起動時にマウントされて読み取られます。第 0 日用コンフィギュレーション ファイルには、少なくとも、管理インターフェイスをアクティブ化するコマンドと、公開キー認証用 SSH サーバーを設定するコマンドを含める必要がありますが、すべての ASA 設定を含めることもできます。

day0.iso ファイル（カスタム day0.iso またはデフォルト day0.iso）は、最初の起動中に使用できる必要があります。

- 初期導入時に自動的に ASAv にライセンスを付与するには、Cisco Smart Software Manager からダウンロードした Smart Licensing Identity（ID）トークンを「idtoken」というテキストファイルに格納し、第 0 日用構成ファイルと同じディレクトリに保存します。
- 仮想 VGA コンソールではなく、ハイパーバイザの**シリアルポート**から ASAv にアクセスし、設定する場合は、第 0 日用構成ファイルにコンソールシリアルの設定を追加して初回ブート時にシリアルポートを使用する必要があります。
- トランスペアレントモードで ASAv を導入する場合は、トランスペアレントモードで実行される既知の ASA 構成ファイルを、第 0 日用構成ファイルとして使用する必要があります。これは、ルーテッド ファイアウォールの第 0 日用コンフィギュレーション ファイルには該当しません。

（注） この例では Linux が使用されていますが、Windows の場合にも同様のユーティリティがあります。

手順

ステップ 1 「day0-config」というテキストファイルに ASAv の CLI 設定を記入します。3 つのインターフェイスの設定とその他の必要な設定を追加します。

最初の行は ASA のバージョンで始める必要があります。day0-config は、有効な ASA 構成である必要があります。day0-config を生成する最適な方法は、既存の ASA または ASAv から実行コンフィギュレーションの関連部分をコピーする方法です。day0-config 内の行の順序は重要で、既存の **show running-config** コマンド出力の順序と一致している必要があります。

例：

```
ASA Version 9.4.1
!
console serial
interface management0/0
nameif management
security-level 100
ip address 192.168.1.2 255.255.255.0
no shutdown
interface gigabitethernet0/0
nameif inside
security-level 100
ip address 10.1.1.2 255.255.255.0
no shutdown
interface gigabitethernet0/1
nameif outside
security-level 0
ip address 198.51.100.2 255.255.255.0
no shutdown
http server enable
http 192.168.1.0 255.255.255.0 management
```

```
crypto key generate rsa modulus 1024
username AdminUser password paSSw0rd
ssh 192.168.1.0 255.255.255.0 management
aaa authentication ssh console LOCAL
```

ステップ 2 （任意）ASAv の初期導入時に自動的にライセンスを許諾する場合は、day0-config ファイルに次の情報が含まれていることを確認してください。

- 管理インターフェイスの IP アドレス
- （任意）SSmart Licensing で使用する HTTP プロキシ
- HTTP プロキシ（指定した場合）または tools.cisco.com への接続を有効にする **route** コマンド
- tools.cisco.com を IP アドレスに解決する DNS サーバー
- 要求する ASAv ライセンスを指定するための Smart Licensing の設定
- （任意）CSSM での ASAv の検索を容易にするための一意のホスト名

ステップ 3 （任意）Cisco Smart Software Manager によって発行された Smart License ID トークンファイルをコンピュータにダウンロードし、ダウンロードファイルから ID トークンをコピーし、ID トークンのみを含む「idtoken」というテキストファイルを作成します。

ステップ 4 テキスト ファイルを ISO ファイルに変換して仮想CD-ROM を生成します。

例：

```
stack@user-ubuntu:-/KvmAsa$ sudo genisoimage -r -o day0.iso day0-config idtoken
I: input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 252
Total directory bytes: 0
Path table size (byptes): 10
Max brk space used 0
176 extents written (0 MB)
stack@user-ubuntu:-/KvmAsa$
```

この ID トークンによって、Smart Licensing サーバーに ASAv が自動的に登録されます。

ステップ 5 ステップ 1 から 5 を繰り返し、導入する ASAv ごとに、適切な IP アドレスを含むデフォルトの構成ファイルを作成します。

仮想ブリッジ XML ファイルの準備

ASAv ゲストを KVM ホストに接続し、ゲストを相互接続する仮想ネットワークを設定する必要があります。

（注） この手順では、KVM ホストから外部への接続は確立されません。

KVM ホスト上に仮想ブリッジ XML ファイルを準備します。第 0 日のコンフィギュレーション ファイルの準備 （5 ページ） に記載されている仮想ネットワーク トポロジの例では、3 つの仮想ブリッジ ファイル（virbr1.xml、virbr2.xml、virbr3.xml）が必要です（これらの 3 つのファイル名を使用する必要があります。たとえば、virbr0 はすでに存在しているため使用できません）。各ファイルには、仮想ブリッジの設定に必要な情報が含まれています。仮想ブリッジに対して名前と一意の MAC アドレスを指定する必要があります。IP アドレスの指定は任意です。

手順

ステップ 1 3 つの仮想ネットワーク ブリッジ XML ファイルを作成します。次の例では、virbr1.xml、virbr2.xml、および virbr3.xml です。

例：

```
<network>
<name>virbr1</name>
<bridge name='virbr1' stp='on' delay='0' />
<mac address='52:54:00:05:6e:00' />
<ip address='192.168.1.10' netmask='255.255.255.0' />
</network>
```

例：

```
<network>
<name>virbr2</name>
<bridge name='virbr2' stp='on' delay='0' />
<mac address='52:54:00:05:6e:01' />
<ip address='10.1.1.10' netmask='255.255.255.0' />
</network>
```

例：

```
<network>
<name>virbr3</name>
<bridge name='virbr3' stp='on' delay='0' />
<mac address='52:54:00:05:6e:02' />
<ip address='198.51.100.10' netmask='255.255.255.0' />
</network>
```

ステップ 2 以下を含むスクリプトを作成します（この例では、スクリプトに virt_network_setup.sh という名前を付けます）。

```
virsh net-create virbr1.xml
virsh net-create virbr2.xml
virsh net-create virbr3.xml
```

ステップ 3 このスクリプトを実行して、仮想ネットワークを設定します。このスクリプトは、仮想ネットワークを稼働状態にします。ネットワークは、KVM ホストが動作している限り稼動します。

```
stack@user-ubuntu:-/KvmAsa$ virt_network_setup.sh
```

（注）

Linux ホストをリロードする場合は、virt_network_setup.sh スクリプトを再実行する必要があります。スクリプトはリブート後に継続されません。

ステップ 4 仮想ネットワークが作成されたことを確認します。

```
stack@user-ubuntu:-/KvmAsa$ brctl show
bridge name bridge id STP enabled Interfaces
virbr0 8000.0000000000000 yes
virbr1 8000.5254000056eed yes virb1-nic
virbr2 8000.5254000056eee yes virb2-nic
virbr3 8000.5254000056eec yes virb3-nic
stack@user-ubuntu:-/KvmAsa$
```

ステップ 5 virbr1 ブリッジに割り当てられている IP アドレスを表示します。これは、XML ファイルで割り当てた IP アドレスです。

```
stack@user-ubuntu:-/KvmAsa$ ip address show virbr1
S: virbr1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
link/ether 52:54:00:05:6e:00 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.10/24 brd 192.168.1.255 scope global virbr1
valid_lft forever preferred_lft forever
```

ASAv の導入

virt-install ベースの導入スクリプトを使用して ASAv を起動できます。

手順

ステップ 1 「virt_install_asav.sh」という virt-install スクリプトを作成します。

ASAv マシンの名前は、この KVM ホスト上の他の全 VM で一意である必要があります。

ASAv では最大 10 のネットワークがサポートされます。この例では 3 つのネットワークが使用されています。ネットワーク ブリッジの句の順序は重要です。リストの最初の句は常に ASAv の管理インターフェイス（Management 0/0）、2 番目の句は ASAv の GigabitEthernet 0/0、3 番目の句は ASAv の GigabitEthernet 0/1 に該当し、GigabitEthernet 0/8 まで同様に続きます。仮想 NIC は Virtio でなければなりません。

例：

```
virt-install \
--connect=qemu:///system \
--network network=default,model=virtio \
--network network=default,model=virtio \
--network network=default,model=virtio \
--name=asav \
--cpu host \
--arch=x86_64 \
--machine=pc-1.0 \
--vcpus=1 \
--ram=2048 \
```

```
--os-type=linux \
--virt-type=kvm \
--import \
--disk path=/home/kvmperf/Images/desmo.qcow2,format=qcow2,device=disk,bus=virtio,cache=none \
--disk path=/home/kvmperf/asav_day0.iso,format=iso,device=cdrom \
--console pty,target_type=virtio \
--serial tcp,host=127.0.0.1:4554,mode=bind,protocol=telnet
```

ステップ 2 virt_install スクリプトを実行します。

例：

```
stack@user-ubuntu:-/KvmAsa$ ./virt_install_asav.sh

Starting install...
Creating domain...
```

ウィンドウが開き、VMのコンソールが表示されます。VMが起動中であることを確認できます。VMが起動するまでに数分かかります。VM が起動したら、コンソール画面から CLI コマンドを実行できます。

ホットプラグ インターフェイス プロビジョニング

ASAv を停止して再起動することなく、インターフェイスを動的に追加および削除できます。ASAv マシンに新しいインターフェイスを追加する場合、ASAv はそのインターフェイスを通常のインターフェイスとして検出してプロビジョニングできる必要があります。同様に、ホットプラグ プロビジョニングによって既存のインターフェイスを削除する場合、ASAv はそのインターフェイスを削除して、関連付けられたすべてのリソースを解放する必要があります。

注意事項と制約事項

インターフェイスのマッピングと番号付け

- ホットプラグインターフェイスを追加する場合、そのインターフェイス番号は、現在の最後のインターフェイス番号に 1 を加えた数になります。
- ホットプラグインターフェイスを削除すると、それが最後の番号のインターフェイスである場合を除き、インターフェイス番号にギャップが生じます。
- インターフェイス番号にギャップがあると、次にホットプラグプロビジョニングされるインターフェイスはそのギャップを埋める番号を使用します。

フェールオーバー

- ホットプラグ インターフェイスをフェールオーバーリンクとして使用する場合、リンクは、ASAv のフェールオーバーペアとして指定されている両方のユニットでプロビジョニングする必要があります。

- まずハイパーバイザのアクティブASAvにホットプラグインターフェイスを追加してから、ハイパーバイザのスタンバイASAvにホットプラグインターフェイスを追加します。
- アクティブ ASAv に新たに追加したフェールオーバー インターフェイスを設定します。設定はスタンバイ装置に同期されます。
- プライマリ ユニットのフェールオーバーを有効にします。

- フェールオーバーリンクを削除する場合、最初にアクティブなASAvでフェールオーバー設定を削除します。
  - ハイパーバイザのアクティブなASAvからフェールオーバーインターフェイスを削除します。
  - 次に、ハイパーバイザのスタンバイ ASAv から対応するインターフェイスをすぐに削除します。

制限事項と制約事項

- ホットプラグ インターフェイス プロビジョニングは Virtio 仮想 NIC に限定されます。
- サポートされるインターフェイスの最大数は 10 です。10 を超える数のインターフェイスを追加しようとすると、エラーメッセージが表示されます。
- インターフェイス カード（media_ethernet/port/id/10）を開くことはできません。
- ホットプラグ インターフェイス プロビジョニングでは ACPI が必要です。virt-install スクリプトには `--noacpi` フラグを含めないでください。

ネットワーク インターフェイスのホットプラグ

KVMハイパーバイザのインターフェイスを追加および削除するには、virshコマンドラインを使用します。

手順

ステップ 1 virsh コマンド ラインのセッションを開きます。

例：

```
[root@asav-kvmterm ~]# virsh
Welcome to virsh, the virtualization interactive terminal.
Type: 'help' for help with commands
'quit' to quit
```

ステップ 2 インターフェイスを追加するには、**attach-interface** コマンドを使用します。

attach-interface{ **--domain** *domain* **--type** *type* **--source** *source* **--model** *model* **--mac** *mac* **--live**}

--domain には、短整数、名前、または完全 UUID を指定できます。--type パラメータは、物理的なネットワーク デバイスを示す *network*、またはデバイスへのブリッジを示す *bridge* のどちらかを指定できます。--source パラメータは、接続のタイプを示します。--model パラメータは、仮想 NIC のタイプを示します。--mac パラメータは、ネットワーク インターフェイスの MAC アドレスを指定します。--live パラメータは、コマンドが実行しているドメインに影響を与えることを示します。

（注）
使用可能なオプションの詳細については、virsh の公式ドキュメントを参照してください。

例：

```
virsh # attach-interface --domain asav-network --type bridge --source br_hpi --model virtio --mac
52:55:04:4b:59:2f --live
```

（注）
ASAv でインターフェイス コンフィギュレーション モードを使用して、トラフィックの送受信インターフェイスを設定して有効化します。詳細については、『Cisco ASA シリーズ CLI コンフィギュレーション ガイド（一般的な操作）』の「*Basic Interface Configuration*」の章を参照してください。

ステップ 3 インターフェイスを削除するには、**detach-interface** コマンドを使用します。

detach-interface{ **--domain** *domain* **--type** *type* **--mac** *mac* **--live**}

（注）
使用可能なオプションの詳細については、*virsh* の公式ドキュメントを参照してください。

例：

```
virsh # detach-interface --domain asav-network --type bridge --mac 52:55:04:4b:59:2f --live
```

パフォーマンスの調整

KVM 構成でのパフォーマンスの向上

KVM ホストの設定を変更することによって、KVM 環境内の ASAv のパフォーマンスを向上させることができます。これらの設定は、ホスト サーバー上の構成時の設定とは無関係です。このオプションは、Red Hat Enterprise Linux 7.0 KVM で使用できます。

CPU ピニングを有効にすると、KVM 構成でのパフォーマンスを向上できます。

CPU ピンニングの有効化

ASAv では、KVM 環境での ASAv のパフォーマンスを向上させるために KVM CPU アフィニティオプションを使用する必要があります。プロセッサアフィニティ（CPU ピニング）により、プロセスまたスレッドと中央処理装置（CPU）や幅広い CPU 間のバインドとバインド解除が可能になり、任意の CPU ではなく、指定された CPU でのみプロセスまたはスレッドが実行されるようになります。

ピン接続されていないインスタンスでピン接続されているインスタンスのリソース要件が使用されないようにするために、CPU ピニングを使用しないインスタンスとは別のホストに CPU ピニングを使用するインスタンスを展開するようにホスト集約を設定します。

注目 NUMA トポロジを持たないインスタンスと同じホストに NUMA トポロジを持つインスタンスを展開しないでください。

このオプションを使用する場合は、KVM ホストで CPU ピンニングを構成します。

手順

ステップ 1 KVM ホスト環境で、ピンニングに使用できる vCPU の数を調べるために、ホストのトポロジを確認します。

例：

```
virsh nodeinfo
```

ステップ 2 使用可能な vCPU の数を確認します。

例：

```
virsh capabilities
```

ステップ 3 vCPU をプロセッサ コアのセットにピンニングします。

例：

```
virsh vcpupin <vm-name> <vcpu-number> <host-core-number>
```

virsh vcpupin コマンドは、ASAv 上の vCPU ごとに実行する必要があります。次の例は、vCPU が 4 個の ASAv 構成を使用し、ホストに 8 個のコアが搭載されている場合に必要になる KVM コマンドを示しています。

```
virsh vcpupin asav 0 2
virsh vcpupin asav 1 3
virsh vcpupin asav 2 4
virsh vcpupin asav 3 5
```

ホストのコア番号は、0 ～ 7 のどの番号でもかまいません。詳細については、KVM のドキュメンテーションを参照してください。

（注）
CPU ピンニングを構成する場合は、ホスト サーバーの CPU トポロジを慎重に検討してください。複数のコアで構成されたサーバーを使用している場合は、複数のソケットにまたがる CPU ピンニングを設定しないでください。

KVM 構成でのパフォーマンスの向上には、専用のシステム リソースが必要になるという短所もあります。

NUMA のガイドライン

Non-uniform Memory Access（NUMA）は、マルチプロセッサシステムのプロセッサに対するメインメモリモジュールの配置について記述する共有メモリアーキテクチャです。プロセッサが自身のノード（リモートメモリ）内に存在しないメモリにアクセスする場合は、ローカルメモリにアクセスする場合よりも低速の速度で、NUMA 接続を介してデータを転送する必要があります。

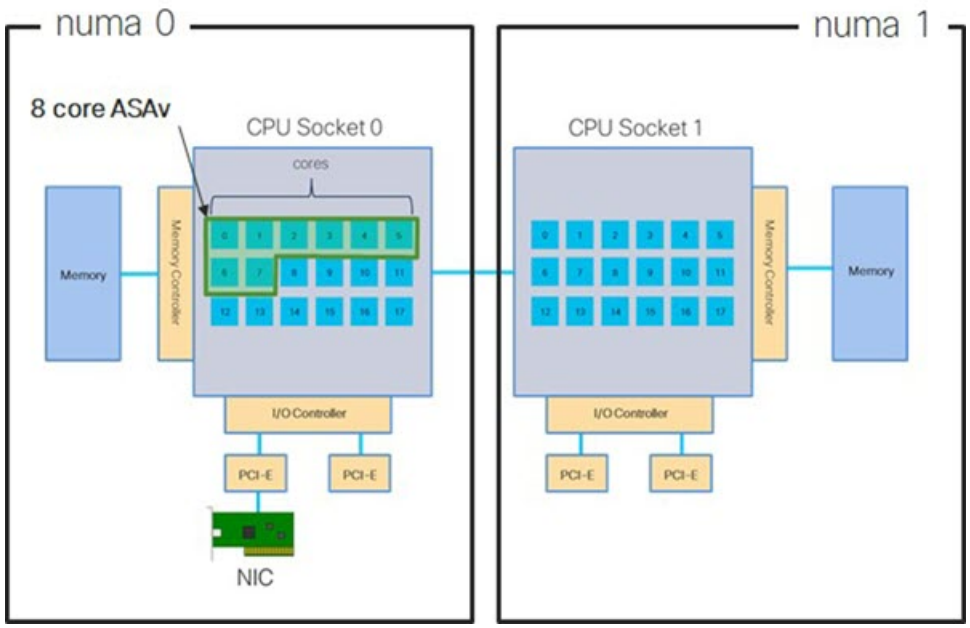
X86 サーバーアーキテクチャは、複数のソケットおよびソケット内の複数のコアで構成されています。各 CPU ソケットとそのメモリおよび I/O が、NUMA ノードと呼ばれます。メモリからパケットを効率的に読み取るには、ゲストアプリケーションおよび関連付けられている周辺機器（NIC など）が同じノード内に存在する必要があります。

最適な ASAv パフォーマンスを実現するには：

- ASAv マシンは、1つの NUMA ノード上で実行する必要があります。1つの ASAv が 2 つのソケットで実行されるように導入されている場合、パフォーマンスは大幅に低下します。
- 8 コア ASAv（図 2 : 8 コア ASAv NUMA アーキテクチャの例 （15 ページ））では、ホスト CPU の各ソケットが、それぞれ 8 個以上のコアを備えている必要があります。サーバー上で実行されている他の VM についても考慮する必要があります。
- 16 コア ASAv（図 3 : 16 コア ASAv NUMA アーキテクチャの例 （15 ページ））では、ホスト CPU 上の各ソケットが、それぞれ 16 個以上のコアを備えている必要があります。サーバー上で実行されている他の VM についても考慮する必要があります。
- NIC は、ASAv マシンと同じ NUMA ノード上にある必要があります。

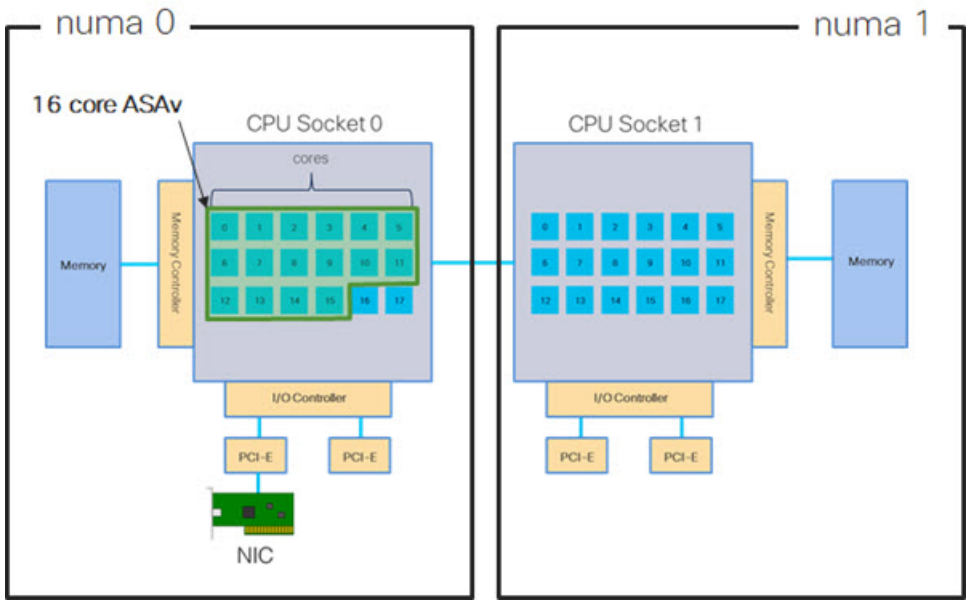
次の図は、2 つの CPU ソケットがあり、各 CPU に 18 個のコアが搭載されているサーバーを示しています。8 コア ASAv では、ホスト CPU の各ソケットに最低 8 個のコアが必要です。

図 *2 : 8* コア *ASAv NUMA* アーキテクチャの例



次の図は、2 つの CPU ソケットがあり、各 CPU に 18 個のコアが搭載されているサーバーを示しています。16 コア ASAv では、ホスト CPU の各ソケットに最低 16 個のコアが必要です。

図 *3 : 16* コア *ASAv NUMA* アーキテクチャの例



NUMA の最適化

理想的には、ASAv マシンは、NIC が動作しているノードと同じ NUMA ノード上で実行する必要があります。手順は次のとおりです。

1. 「lstopo」を使用して NIC がオンになっているノードを判別し、ノードの図を表示します。NIC を見つけて、どのノードが接続されているかをメモします。
2. KVM ホストで、`virsh list` を使用して ASAv を検出します。
3. `virsh edit <VM Number>` を使用して VM を編集します。
4. 選択したノードに ASAv を配置します。次の例では、18 コアノードを想定しています。

   ノード 0 への配置：

   ```
   <vcpu placement='static' cpuset='0-17'>16</vcpu>
   <numatune>
       <memory mode='strict' nodeset='0'/>
   </numatune>
   ```

   ノード 1 への配置：

   ```
   <vcpu placement='static' cpuset='18-35'>16</vcpu>
   <numatune>
       <memory mode='strict' nodeset='1'/>
   </numatune>
   ```

5. .xml の変更を保存し、ASAv マシンの電源を再投入します。
6. VM が目的のノードで実行されていることを確認するには、`ps aux | grep <name of your ASAv VM>` を実行して、プロセス ID を取得します。
7. `sudo numastat -c <ASAv VM Process ID>` を実行して、ASAv マシンが適切に配置されているか確認します。

KVM での NUMA 調整の使用に関する詳細については、RedHat のドキュメント『9.3. libvirt NUMA Tuning』を参照してください。

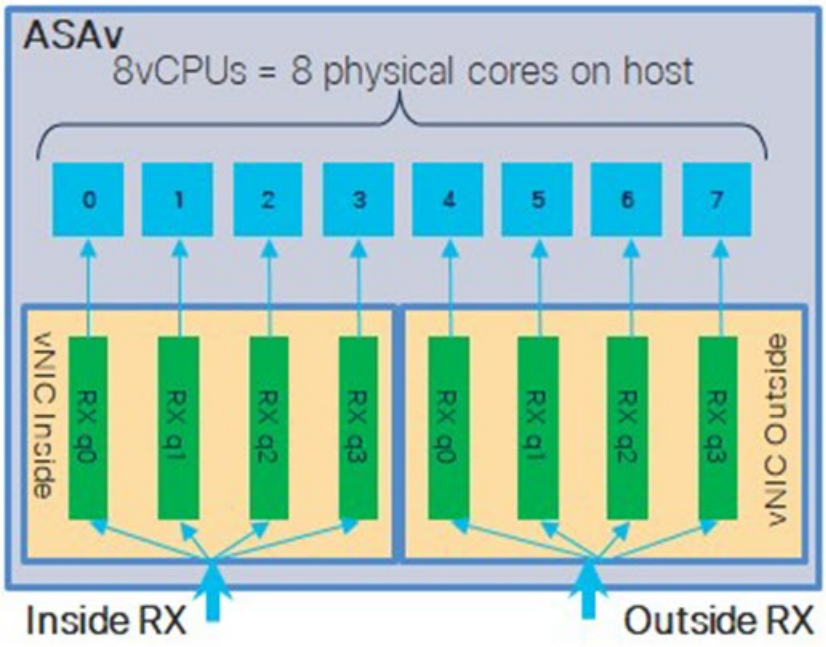
Receive Side Scaling（RSS）用の複数の RX キュー

ASAvは、複数のプロセッサコアにネットワーク受信トラフィックを分散するためにネットワークアダプタによって使用されるテクノロジーである Receive Side Scaling（RSS）をサポートしています。最大スループットを実現するには、各 vCPU（コア）に独自の NIC RX キューが設定されている必要があります。一般的な RA VPN 展開では、1つの内部/外部ペアのインターフェイスを使用する場合があることに注意してください。

重要 複数の RX キューを使用するには、ASAv バージョン 9.13(1) 以降が必要です。KVM の場合、*libvirt* のバージョンは 1.0.6 以降である必要があります。

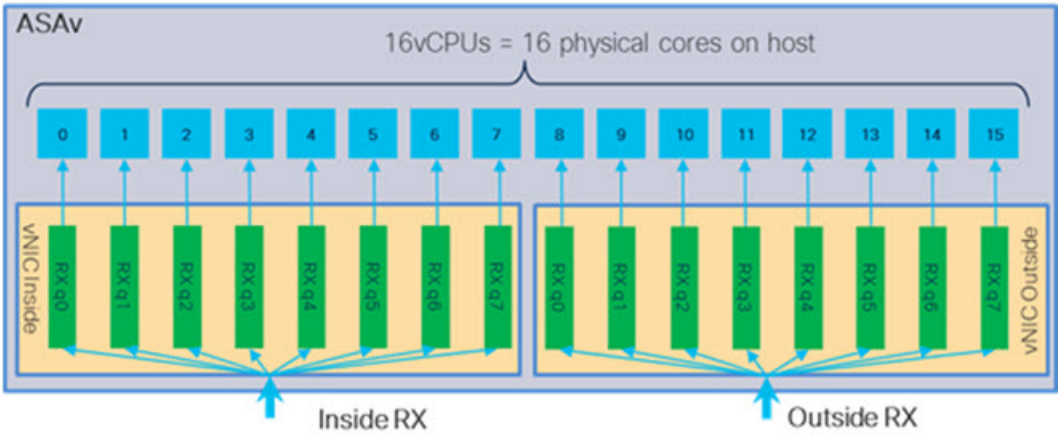
内部/外部ペアのインターフェイスを持つ 8 コア VM の場合、図 4 : 8 コア ASAv RSS RX キュー（17 ページ）に示すように、各インターフェイスには 4 つの RX キューがあります。

図 *4 : 8* コア *ASAv RSS RX* キュー



内部/外部ペアのインターフェイスを持つ 16 コア VM の場合、図 5 : 16 コア ASAv RSS RX キュー （17 ページ）に示すように、各インターフェイスには 8 つの RX キューがあり ます。

図 *5 : 16* コア *ASAv RSS RX* キュー



次の表に、KVM 用の ASAv の vNIC およびサポートされている RX キューの数を示します。サポートされている vNIC の説明については、推奨される vNIC （2 ページ）を参照してください。

表 *1 : KVM* で推奨される *NIC/vNIC*

NIC カード	vNIC ドライバ	ドライバテクノロジー	RX キューの数	パフォーマンス
x710	i40e	PCI パススルー	8（最大）	x710 の PCI パススルーおよび SR-IOV モードは、最適なパフォーマンスを提供します。通常、仮想展開では、複数の VM 間で NIC を共有できるため、SR-IOV が推奨されます。
	i40evf	SR-IOV	8	
x520	ixgbe	PCI パススルー	6	x520 NIC は、x710 よりも 10 ～ 30% パフォーマンスが低くなります。X520 の PCI パススルーおよび SR-IOV モードは、同様のパフォーマンスを提供します。通常、仮想展開では、複数の VM 間で NIC を共有できるため、SR-IOV が推奨されます。
	ixgbe-vf	SR-IOV	2	
該当なし	virtio	準仮想化	8（最大）	ASAv100 には推奨されません。 その他の展開については、KVM での Virtio のマルチキューサポートの有効化 （18 ページ） を参照してください。

KVM での Virtio のマルチキューサポートの有効化

次の例は、libvirt xml を編集するために、Virtio NIC RX キューの数を 4 に設定する方法を示しています。

```
<interface type='bridge'>
 <mac address='52:54:00:43:6e:3f'/>
 <source bridge='clients'/>
 <model type='virtio'/>
 <driver name='vhost' queues='4'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</interface>
```

重要 複数の RX キューをサポートするには、*libvirt* のバージョンが 1.0.6 以降である必要があります。

VPN の最適化

ASAv で VPN パフォーマンスを最適化するための追加の考慮事項は、次のとおりです。

- IPSec のスループットは DTLS よりも高くなります。
- GCM 暗号には、CBC の約 2 倍のスループットがあります。

SR-IOV インターフェイスのプロビジョニング

SR-IOV を使用すれば、複数の VM でホスト内部の 1 台の PCIe ネットワーク アダプタを共有することができます。SR-IOV は次の機能を定義しています。

- 物理機能（PF）：PF は、SR-IOV 機能を含むフル PCIe 機能です。これらは、ホスト サーバー上の通常のスタティック NIC として表示されます。
- 仮想機能（VF）：VF は、データ転送を支援する軽量 PCIe 機能です。VF は、PF から抽出され、PF を介して管理されます。

VF は、仮想化されたオペレーティング システム フレームワーク内の ASAv マシンに最大 10 Gbps の接続を提供できます。このセクションでは、KVM 環境で VF を設定する方法について説明します。ASAv 上の SR-IOV サポートについては、ASAv と SR-IOV インターフェイスのプロビジョニングを参照してください。

ASAv5 および ASAv10 で最適なパフォーマンスを得るため、VMXNET3 ドライバを強く推奨します。さらに、SR-IOV インターフェイスを組み合わせて（インターフェイスが混在した状態で）使用すると、特により多くの CPU コアとリソースを割り当てることで、ASAv とのネットワークパフォーマンスが向上します。

SR-IOV インターフェイスのプロビジョニングに関する要件

SR-IOV をサポートする物理 NIC がある場合、SR-IOV 対応 VF または仮想 NIC（vNIC）を ASAv インスタンスにアタッチできます。SR-IOV は、BIOS だけでなく、ハードウェア上で実行しているオペレーティング システム インスタンスまたはハイパーバイザでのサポートも必要です。KVM 環境で実行中の ASAv 用の SR-IOV インターフェイスのプロビジョニングに関する一般的なガイドラインのリストを以下に示します。

- ホスト サーバーには SR-IOV 対応物理 NIC が必要です。SR-IOV インターフェイスに関するガイドラインと制限事項を参照してください。
- ホスト サーバーの BIOS で仮想化が有効になっている必要があります。詳細については、ベンダーのマニュアルを参照してください。
- ホスト サーバーの BIOS で IOMMU グローバル サポートが SR-IOV に対して有効になっている必要があります。詳細については、ハードウェア ベンダーのマニュアルを参照してください。

- SR-IOV インターフェイスを使用する KVM 上の ASAv では、インターフェイスタイプの混在がサポートされています。管理インターフェイスには SR-IOV または VMXNET3 を使用し、データインターフェイスには SR-IOV を使用することができます。

KVM ホスト BIOS とホスト OS の変更

このセクションでは、KVM システム上の SR-IOV インターフェイスのプロビジョニングに関するさまざまなセットアップ手順と設定手順を示します。このセクション内の情報は、Intel Ethernet Server Adapter X520 - DA2 を使用した Cisco UCS C シリーズ サーバー上の Ubuntu 14.04 を使用して、特定のラボ環境内のデバイスから作成されたものです。

始める前に

- SR-IOV 互換ネットワーク インターフェイス カード（NIC）が取り付けられていることを確認します。
- Intel 仮想化テクノロジー（VT-x）機能と VT-d 機能が有効になっていることを確認します。

（注） システム メーカーによっては、これらの拡張機能がデフォルトで無効になっている場合があります。システムごとに BIOS 設定にアクセスして変更する方法が異なるため、ベンダーのマニュアルでプロセスを確認することをお勧めします。

- オペレーティング システムのインストール中に、Linux KVM モジュール、ライブラリ、ユーザ ツール、およびユーティリティのすべてがインストールされていることを確認します。前提条件 （4 ページ）を参照してください。
- 物理インターフェイスが稼働状態であることを確認します。ifconfig <*ethname*> を使用して確認します。

手順

ステップ 1 "root" ユーザー アカウントとパスワードを使用してシステムにログインします。

ステップ 2 Intel VT-d が有効になっていることを確認します。

例：

```
kvmuser@kvm-host:/$ dmesg | grep -e DMAR -e IOMMU
[    0.000000] ACPI: DMAR 0x000000006F9A4C68 000140 (v01 Cisco0 CiscoUCS 00000001 INTL 20091013)
[    0.000000] DMAR: IOMMU enabled
```

最後の行は、VT-d が有効になっていることを示しています。

ステップ 3 */etc/default/grub* 設定ファイル内の GRUB_CMDLINE_LINUX エントリに *intel_iommu=on* パラメータを付加することによって、カーネル内の Intel VT-d をアクティブにします。

例：

```
# vi /etc/default/grub
...
GRUB_CMDLINE_LINUX="nofb splash=quiet console=tty0 ... intel_iommu=on"
...
```

（注）
AMD プロセッサを使用している場合は、代わりに、*amd_iommu=on* をブート パラメータに付加します。

ステップ 4 iommu の変更を有効にするためにサーバーをリブートします。

例：

```
> shutdown -r now
```

ステップ 5 次の形式を使用して *sysfs* インターフェイス経由で *sriov_numvfs* パラメータに適切な値を書き込むことによって、VF を作成します。

```
#echo n > /sys/class/net/device name/device/sriov_numvfs
```

サーバーの電源を入れ直すたびに必要な数の VF が作成されるようにするには、*/etc/rc.d/* ディレクトリに配置されている *rc.local* ファイルに上記コマンドを付加します。Linux OS は、ブート プロセスの最後で *rc.local* スクリプトを実行します。

たとえば、ポートあたり 1 つの VF を作成するケースを以下に示します。お使いのセットアップではインターフェイスが異なる可能性があります。

例：

```
echo '1' > /sys/class/net/eth4/device/sriov_numvfs
echo '1' > /sys/class/net/eth5/device/sriov_numvfs
echo '1' > /sys/class/net/eth6/device/sriov_numvfs
echo '1' > /sys/class/net/eth7/device/sriov_numvfs
```

ステップ 6 サーバーをリブートします。

例：

```
> shutdown -r now
```

ステップ 7 *lspci* を使用して、VF が作成されたことを確認します。

例：

```
 > lspci | grep -i "Virtual Function"
kvmuser@kvm-racetrack:~$ lspci | grep -i "Virtual Function"
0a:10.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
0a:10.1 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
0a:10.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
0a:10.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
```

（注）
ifconfig コマンドを使用して、新しいインターフェイスを表示します。

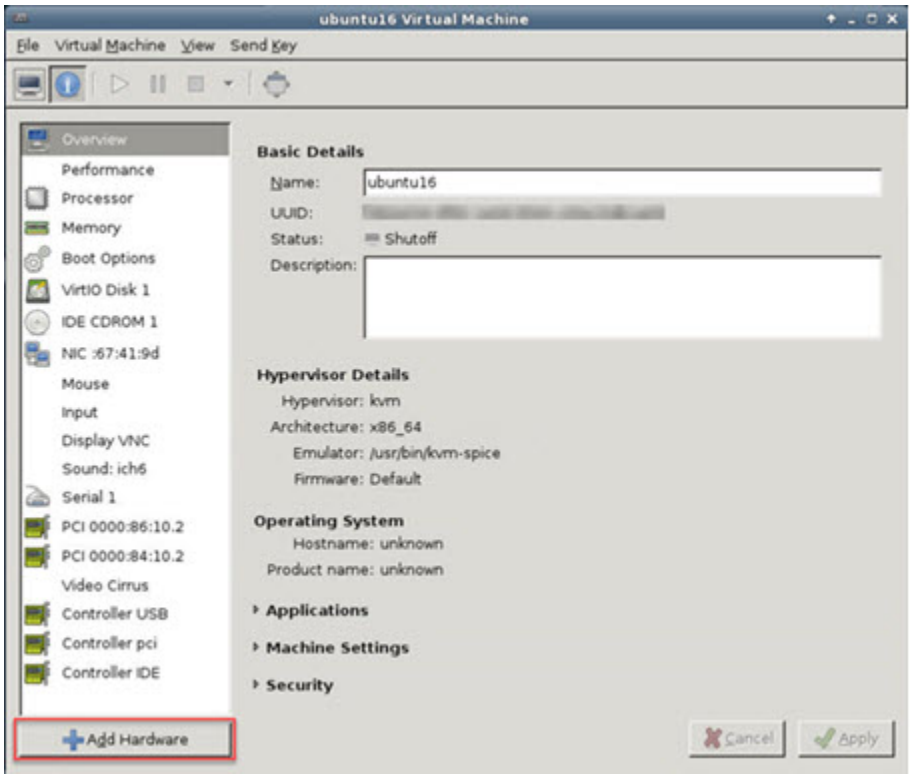
ASAv への PCI デバイスの割り当て

VF を作成したら、PCI デバイスを追加するのと同様に、VF を ASAv に追加できます。次の例では、グラフィカル **virt-manager** ツールを使用して、イーサネット VF コントローラを ASAv に追加する方法について説明します。

手順

ステップ 1 ASAv を開いて、[Add Hardware] ボタンをクリックし、新しいデバイスを仮想マシンに追加します。

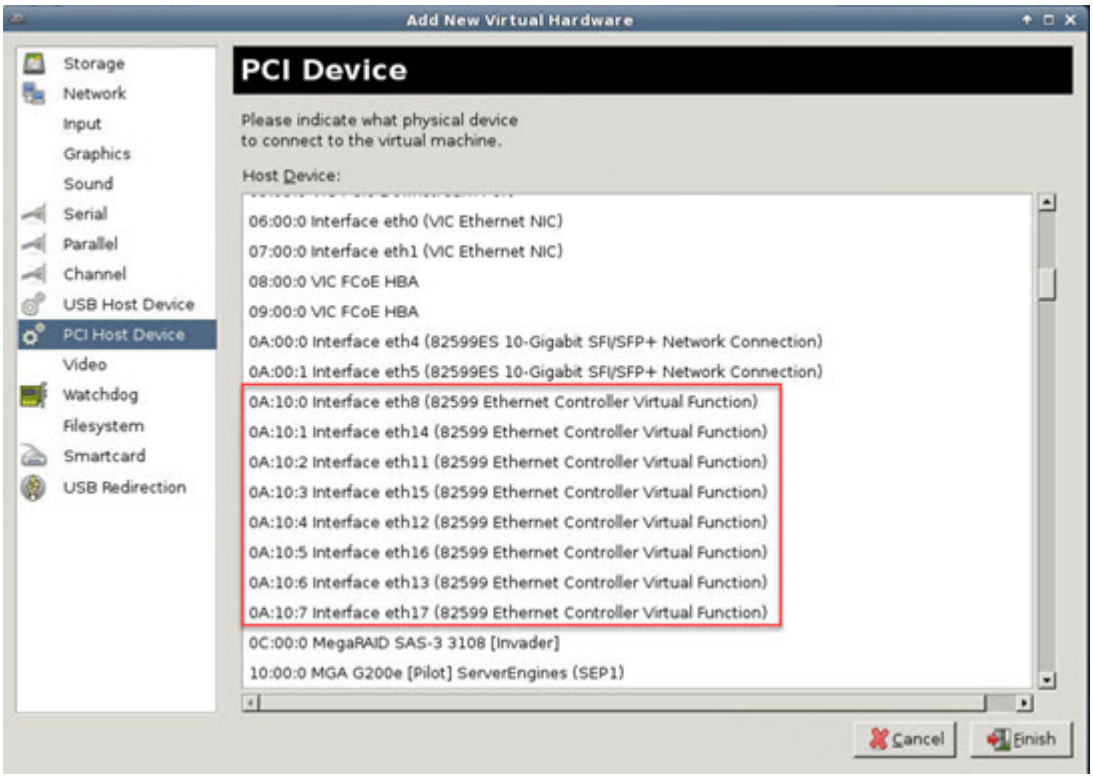
図 *6:* ハードウェアの追加



ステップ 2 左ペインの [Hardware] リストで [PCI Host Device] をクリックします。

VF を含む PCI デバイスのリストが中央ペインに表示されます。

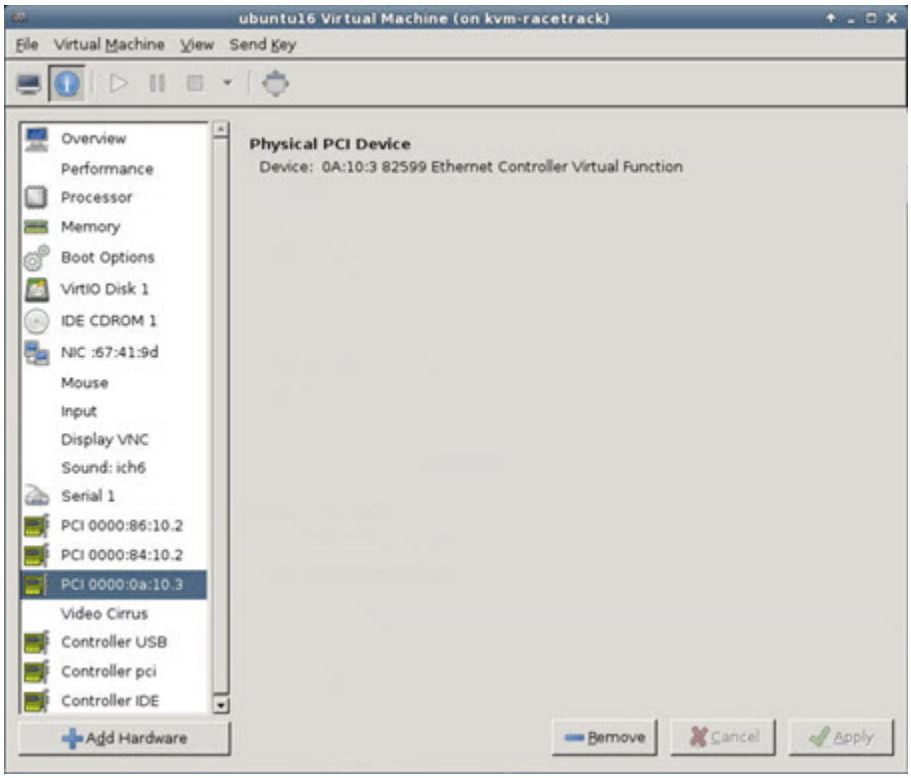
図 *7*:仮想機能のリスト



ステップ 3 使用可能な仮想機能のいずれかを選択して、[Finish] をクリックします。

PCI デバイスがハードウェア リストに表示されます。デバイスの記述が Ethernet Controller Virtual Function になっていることに注意してください。

図 *8 :* 追加された仮想機能



次のタスク

- ASAv コマンドラインから、**show interface** コマンドを使用して、新しく設定したインターフェイスを確認します。
- ASAv でインターフェイスコンフィギュレーションモードを使用して、トラフィックの送受信インターフェイスを設定して有効化します。詳細については、『Cisco ASA シリーズ CLI コンフィギュレーション ガイド（一般的な操作）』の「*Basic Interface Configuration*」の章を参照してください。

CPU 使用率とレポート

CPU使用率レポートには、指定された時間内に使用されたCPUの割合の要約が表示されます。通常、コアはピーク時以外には合計 CPU 容量の約 30 ～ 40% で動作し、ピーク時は約 60 ～ 70% の容量で動作します。

重要 9.13(1) 以降では、サポートされているすべての ASA Virtual vCPU/メモリ構成ですべての ASA Virtual ライセンスを使用できるようになり、ASA Virtual を使用しているお客様は、さまざまな VM リソースフットプリントで実行できます。

ASA Virtual の vCPU 使用率

ASA Virtual の vCPU 使用率には、データパス、制御ポイント、および外部プロセスで使用されている vCPU の量が表示されます。

vSphere で報告される vCPU の使用率には、ASA Virtual の使用率に加えて、次のものが含まれます。

- ASA Virtual アイドル時間
- ASA Virtual マシンに使用された %SYS オーバーヘッド
- vSwitch、vNIC および pNIC の間を移動するパケットのオーバーヘッド。このオーバーヘッドは非常に大きくなる場合があります。

CPU 使用率の例

CPU 使用率の統計情報を表示するには、**show cpu usage** コマンドを使用します。

例

```
Ciscoasa#show cpu usage

CPU □□□□5□□□ 1%□1 □□□ 2%□5 □□□ 1%
```

報告された vCPU の使用率が大幅に異なる例を次に示します。

- ASA Virtual レポート：40%
- DP：35%
- 外部プロセス：5%
- ASA（ASA Virtual レポート）：40%
- ASA アイドル ポーリング：10%
- オーバーヘッド：45%

オーバーヘッドは、ハイパーバイザ機能の実行、および vSwitch を使用した NIC と vNIC の間のパケット転送に使用されています。

KVM CPU 使用率レポート

値は、

```
virsh cpu-stats domain --total start count
```

コマンドを実行すると、指定されたゲスト仮想マシンのCPU統計情報が表示されます。デフォルトでは、すべてのCPUの統計と合計が表示されます。`--total`オプションを指定すると、合計統計のみ表示されます。`--count`オプションを指定すると、count 個の CPU の統計のみ表示されます。

OProfile、topなどのツールを実行すると、ハイパーバイザとVMの両方のCPU使用率を含む、特定の KVM VM の合計 CPU 使用率が表示されます。同様に、Xen VMM に固有の XenMon などのツールの場合、Xenハイパーバイザ、つまりDom 0の合計CPU使用率が表示されますが、VM ごとのハイパーバイザ使用率には分割されません。

これらのツールとは別に、OpenNebula などのクラウド コンピューティング フレームワークには、VM によって使用される仮想 CPU の割合の大まかな情報のみを提供する特定のツールが存在します。

ASA Virtual と KVM のグラフ

ASA Virtual と KVM の間には CPU % の数値に違いがあります。

- KVM グラフの数値は ASA Virtual の数値よりも常に大きくなります。
- KVMではこの値は「%CPU usage」と呼ばれ、ASA Virtualではこの値は「%CPU utilization」と呼ばれます。

用語「%CPU utilization」と「%CPU usage」は別のものを意味しています。

- CPU utilization は、物理 CPU の統計情報を提供します。
- CPU usage は CPU のハイパースレッディングに基づいた論理 CPU の統計情報を提供します。しかし、1 つの vCPU のみが使用されるため、ハイパースレッディングは動作しません。

KVM では「%CPU usage」は次のように計算されます。

アクティブに使用された仮想 CPU の量。使用可能な CPU の合計に対する割合として指定されます。

この計算は、ホストから見た CPU 使用率であり、ゲスト オペレーティング システムから見た CPU 使用率ではありません。また、これは仮想マシンで使用可能なすべての仮想 CPU の平均 CPU 使用率になります。

たとえば、1 個の仮想 CPU を搭載した 1 つの仮想マシンが、4 個の物理 CPU を搭載した 1 台のホストで実行されており、その CPU 使用率が 100% の場合、仮想マシンは、1 個の物理 CPU をすべて使用しています。仮想 CPU の使用率は、「MHz 単位の使用率 / 仮想 CPU の数 x コア周波数」として計算されます。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。