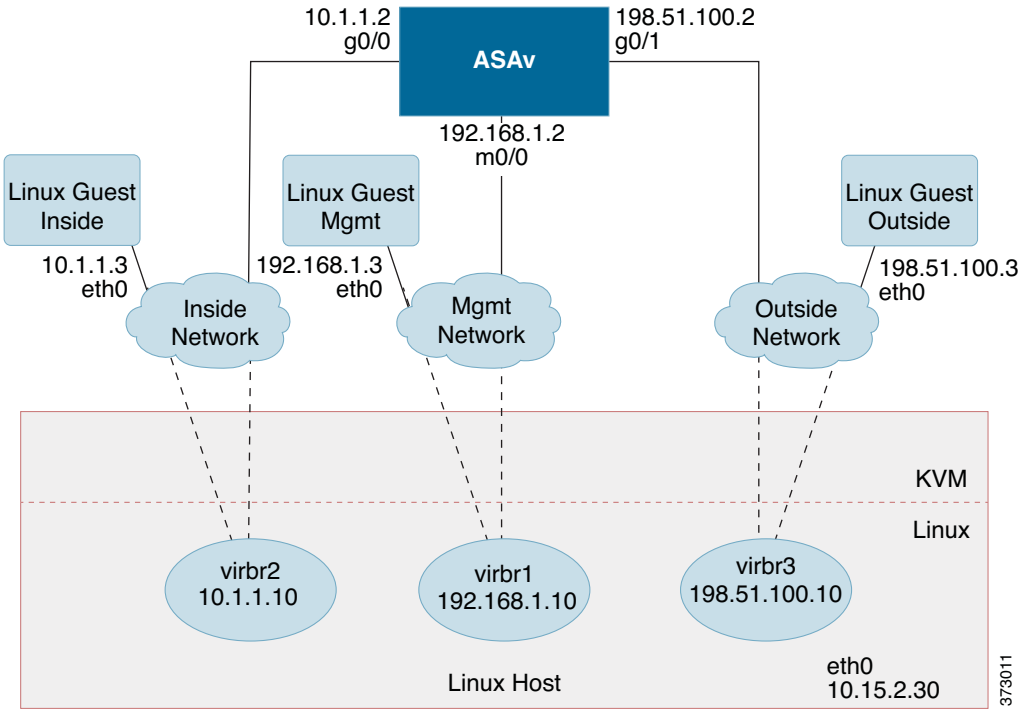
KVM を使用した ASAv の導入

カーネルベースの仮想マシン（KVM）を使用して ASAv を導入することができます。

KVM を使用した ASAv の導入について

図 1 頁 34 は、ASAv と KVM によるネットワーク トポロジの例を示しています。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。実際に必要な手順は、各自の要件に応じて異なります。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



ASAv と KVM の前提条件

- 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、ユーザが Ubuntu 14.04 LTS を使用していることを前提としています。Ubuntu 14.04 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、『Network Function Virtualization Packet Processing Performance of Virtualized Platforms with Linux and Intel Architecture』を参照してください。

- 以下の機能は Ubuntu 14.04 の最適化に役立ちます。
  - 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 14.04 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 Linux6 Virtualization Tuning and Optimization Guide』を参照してください。
- KVM のシステム要件については、「Cisco ASA Compatibility」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（Day 0）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 サーバをセットアップ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 またはデフォルトの day0.iso）は、最初の起動中に使用できなければなりません。

注：初期導入時に自動的に ASAv をライセンス許諾するには、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 日用コンフィギュレーション ファイルと同じディレクトリに保存します。

注：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 コンフィギュレーション ファイルを第 0 日用コンフィギュレーション ファイルとして使用します。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

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必要な部分をコピーすることです。day0-config 内の行の順序は重要で、既存の **show run** 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 9.5.1
!
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
```

2. (任意)Cisco Smart Software Manager により発行された 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ます。
3. (任意)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。
4. (任意)ASAv の初期導入時に自動的にライセンス許諾を行う場合は、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)Smart 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
   - 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
5. 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。

```
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 サーバに ASAv が自動的に登録されます。
6. 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コンフィギュレーション ファイルを作成します。

仮想ブリッジ XML ファイの準備

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 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、35 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

1. 次の 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。

   virbr1.xml:

```
<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>
```

   virbr2.xml:

```
<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>
```

   virbr3.xml:

```
<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>
```

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
```

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。スクリプトによって仮想ネットワークが確立されます。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。

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$
```

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 の起動

ASAv を起動するには、virt-install ベースの導入スクリプトを使用します。

手順

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

   ASAv VM の名前は、KVM ホスト上の他の仮想マシン（VM）全体において一意である必要があります。ASAv は最大 10 のネットワークをサポートできます。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（管理 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、GigabitEthernet0/8 まで同様に続きます。仮想 NIC は Virtio でなければなりません。

   注：*watchdog* 要素は、KVM ゲストの仮想ハードウェア ウォッチドッグ デバイスです。ASAv が何らかの理由で応答しなくなると、ウォッチドッグは KVM ゲストの再起動を開始できます。

   注：各 KVM ゲスト ディスク インターフェイスで、いずれかのキャッシュ モード（*writethrough*、*writeback*、*none*、*directsync*、または *unsafe*）を指定することができます。*cache=writethrough* は、ホストで突然の停電が発生した場合の KVM ゲスト マシン上のファイル破損を低減できます。ただし、*cache=writethrough* は、*cache=none* よりディスク I/O 書き込みが多いため、ディスク パフォーマンスに影響する可能性もあります。

```
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 \
   --os-variant=generic26 \
   --noacpi \
   --virt-type=kvm \
   --import \
    --watchdog i6300esb,action=reset
    --disk path=/home/kvmperf/Images/asav.qcow2,format=qcow2,device=disk,bus=virtio,
      cache=writethrough
    --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
```

2. virt_install スクリプトを実行します。

```
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 を停止して再起動しなくても、インターフェイスを動的に追加および削除できます。ASAv 仮想マシンに新しいインターフェイスを追加したときに、ASAv はそれ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と、ASAv は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。

ホットプラグ インターフェイス プロビジョニングのためのガイドライン

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、それ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 に新しく追加され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 ユニットに同期されます。
  - 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 リンクを解除するには、次の手順に従います。
  - 最初にアクティブ ASAv の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 ASAv からフェールオーバー インターフェイスを削除し、その後すぐ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。

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

1. virsh コマンド ラインのセッションを開きます。

```
[root@asav-kvmterm ~]# virsh
Welcome to virsh, the virtualization interactive terminal.

Type:    'help' for help with commands
         'quit' to quit
```

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。

```
virsh # attach-interface domain type source model mac live
```

例：

```
virsh # attach-interface --domain asav-network --type bridge --source br_hpi --model virtio --mac
52:55:04:4b:59:2f --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: トラフィックを送受信するためのインターフェイスを設定して有効にするには、ASAv のインターフェイス コンフィギュレーション モードを使用します。詳細については、「Navigating the Cisco ASA Series Documentation」を参照してください。

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。

```
virsh # detach-interface domain type mac live
```

例:

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

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 仮想マシンに最大 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、8 ページで説明します。

SR-IOV インターフェイスのプロビジョニングに関するガイドライン

SR-IOV をサポートする物理 NIC があれば、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 サーバには SR-IOV 対応物理 NIC が必要です。SR-IOV に対してサポートされている NIC、5 ページを参照してください。
- ホスト サーバの 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、ベンダーのマニュアルを参照してください。

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。

はじめる前に

- 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。ASAv と KVM の前提条件、34 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

1. "root" ユーザ 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。
2. Intel VT-d が有効になっていることを確認します。

   次に例を示します。

   ```
   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。
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。

   次に例を示します。

   ```
   # 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 パラメータに付加する必要があります。
4. iommu の変更を有効にするためにサーバをリブートします。

   次に例を示します。

   ```
   > shutdown -r now
   ```
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
   ```

   サーバの電源を入れ直すたびに必要な数の VF が作成されることを保証するには、***/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。

   ```
   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
   ```
6. サーバをリブートします。

   次に例を示します。

   ```
   > shutdown -r now
   ```

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

   次に例を示します。

```
 > 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)
```

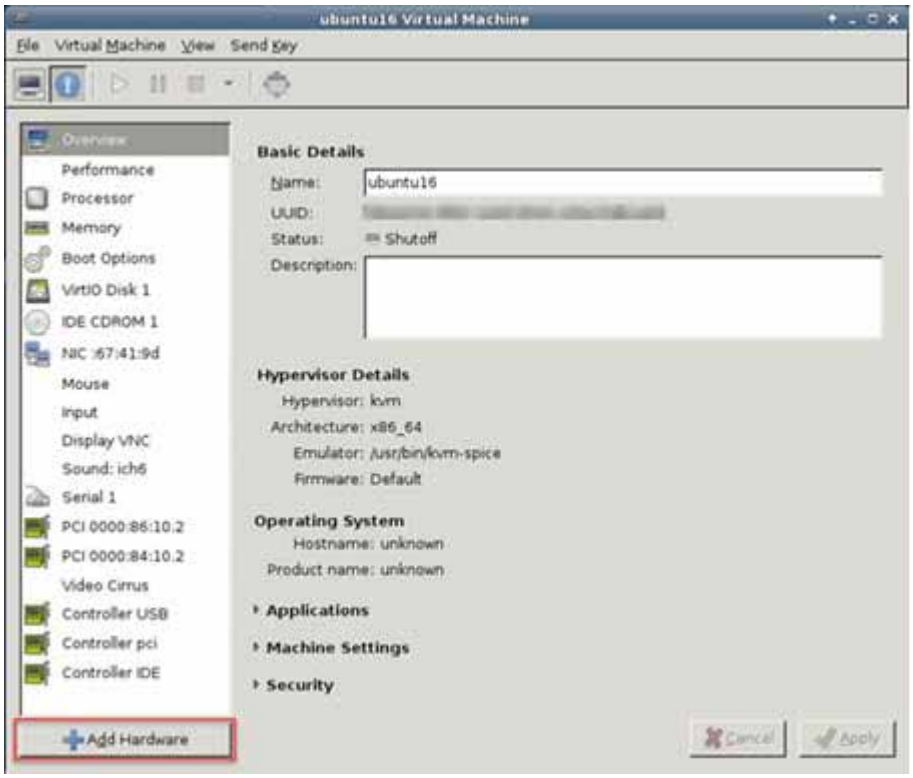
   注:ifconfig コマンドを使用して、新しいインターフェイスを表示します。

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。

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 2　ハードウェアの追加



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

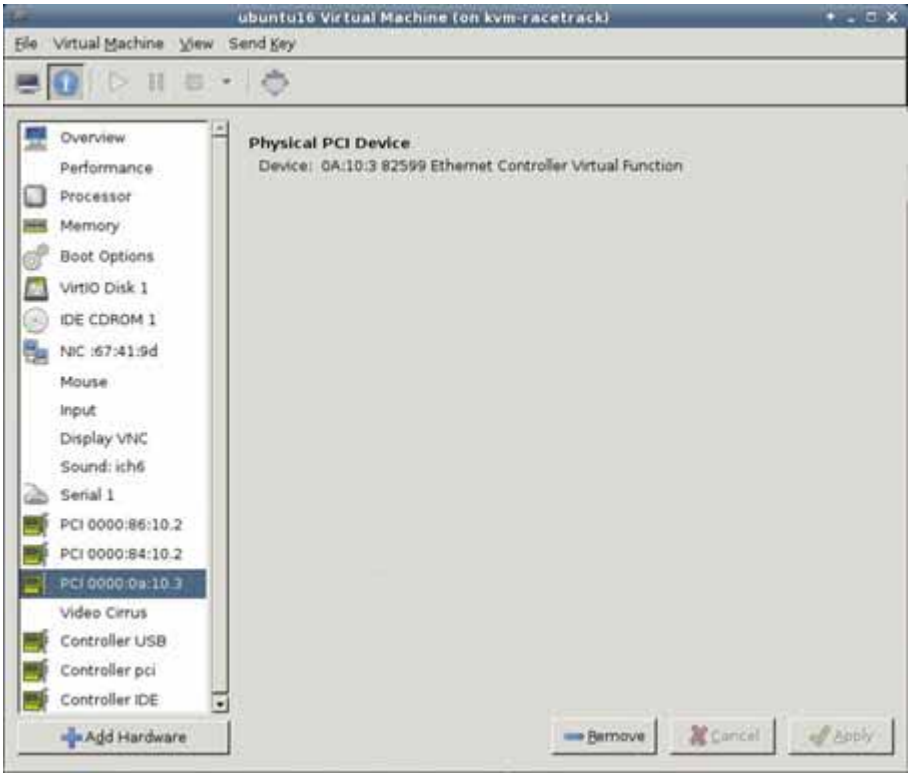
   VF を含む PCI デバイスのリストが中央ペインに表示されます。

図 3　仮想機能のリスト

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 4　追加された仮想機能



次の作業

- 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。
- トラフィックを送受信するためのインターフェイスを設定して有効にするには、ASAv のインターフェイス コンフィギュレーション モードを使用します。詳細については、『Navigating the Cisco ASA Series Documentation』を参照してください。

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 ピニングの有効化

KVM 環境内の 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 オプションを使用して特定のプロセッサに仮想マシンを割り当てることができます。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

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。

   ```
   virsh nodeinfo
   ```

2. 使用可能な vCPU の数を確認します。

   ```
   virsh capabilities
   ```

3. vCPU をプロセッサ コアのセットにピンニングします。

```
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 ピンニングを設定する場合は、ホスト サーバの 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。

KVM 構成でのパフォーマンスの向上