# Red Hat Enterprise Linux 6

# 仮想化ホスト設定およびゲストインストールガイド

仮想環境のインストールと設定

Last Updated: 2023-03-02

# Red Hat Enterprise Linux 6 仮想化ホスト設定およびゲストインストールガイド

仮想環境のインストールと設定

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

## 法律上の通知



## 概要

本ガイドでは、KVM パッケージ、互換性、および制限について説明します。また、異なるタイプ、PCI デバイス設定、および SR-IOV のゲスト仮想マシンをインストールするためのホスト設定の詳細および手順も含まれます。

# 目次

# 第1章 はじめに

## 1.1. 本書の内容

本ガイドでは、仮想化ソフトウェアをインストールし、Red Hat Enterprise Linux 仮想化ホストでゲストマシンを設定する方法を説明します。

本ガイドの初期章では、Red Hat Enterprise Linux ホストマシンが仮想化をデプロイできるようにする前提条件の概要を説明します。システム要件、互換性のあるハードウェア、サポート、および製品の制限について詳細に説明します。

必須およびオプションの仮想化パッケージを含む基本的なホスト設定は、5章*仮想化パッケージのインストール* で説明されています。

ゲスト仮想マシンのインストールは、6章*ゲスト仮想マシンのインストールの概要* から詳細に説明されています。この操作では、virt-manager および virsh を使用して、完全に仮想化された Red Hat Enterprise Linux ゲストおよび Windows 準仮想化ゲストをインストールします。

ネットワーク、PCI デバイス設定、SR-IOV、KVM ゲストタイミング管理、および libvirt および SR-IOV のトラブルシューティングに関するヘルプの詳細情報については、後で説明します。

**注記**

本書では、仮想化ホスト設定およびゲストのインストールに関するガイダンスを提供します。より詳細なシステム設定情報については、『Red Hat Enterprise Linux - Virtualization Administration Guide』 を参照してください。

# 第2章 システム要件

本章では、Red Hat Enterprise Linux 6 の仮想マシンと呼ばれる、仮想マシンを正常に実行するためのシステム要件を説明します。仮想化は、Intel 64 および AMD64 アーキテクチャー上の Red Hat Enterprise Linux 6 で利用できます。

KVM ハイパーバイザーは、Red Hat Enterprise Linux 6 で提供されます。

仮想化パッケージのインストール方法は、5章*仮想化パッケージのインストール* を参照してください。

**最小システム要件**

- 6 GB の空きディスク容量
- 2 GB のメモリー

**推奨されるシステム要件**

- ゲスト仮想マシンの仮想化 CPU の最大数とホスト用に 1 つのプロセッサーコアまたはハイパースレッド。
- 2 GB のメモリーと仮想マシン用の追加の RAM。
- ホスト用に 6 GB のディスク容量と、各仮想マシンに必要なディスク領域。

  ほとんどのゲストオペレーティングシステムには、少なくとも 6 GB のディスク領域が必要ですが、各ゲストに必要な追加のストレージ領域はイメージ形式によって異なります。

  raw イメージを使用するゲスト仮想マシンの場合には、ゲスト **の必要な合計容量(raw 形式の合計)** は、ゲストの raw イメージファイル **（イメージ）、ホストオペレーティングシステム（ホスト）** に必要な 6GB 領域、およびゲストが必要とするスワップ領域 **(swap)** の合計以上になります。

  **式2.1 raw イメージを使用したゲスト仮想マシンに必要な領域の計算**

  total for raw format = images + host + swap

  qcow イメージでは、qcow イメージおよび qcow2 イメージが必要に応じて大きくなるため、ゲスト **(total for qcow format)** の予想される最大ストレージ要件も計算する必要があります。この拡張を可能にするには、最初にゲスト **(expected maximum guest storage)** のストレージ要件の上限に 1.01 を乗じ、これにホスト **(host)** に必要な領域と、必要なスワップ領域 **(swap)** を加算します。

  **式2.2 qcow イメージを使用したゲスト仮想マシンに必要な領域の計算**

  total for qcow format = (expected maximum guest storage * 1.01) + host + swap

ゲスト仮想マシンの要件は、第 6 章 『Red Hat Enterprise Linux 6 仮想化管理ガイド 』 で詳しく説明しています。KVM でのオーバーコミット。

**スワップ領域の計算**

スワップ領域を使用すると、利用可能な物理メモリーを超えてメモリーを追加できます。swap パーティションは、メモリーのパフォーマンスを高速化するために、使用されないメモリーをハードドライブにスワップするために使用されます。swap パーティションのデフォルトサイズは、ホストの物理 RAM から計算されます。

Red Hat ナレッジベースには、swap パーティションに適したサイズを安全かつ効率的に判断するための記事を参照してください。を参照し https://access.redhat.com/site/solutions/15244 てください。

### KVM 要件

KVM ハイパーバイザーには、以下が必要です。

- x86 ベースのシステム用の Intel VT-x および Intel 64 拡張機能を備えた Intel プロセッサー
- AMD-V および AMD64 拡張機能を備えた AMD プロセッサー。

プロセッサーに仮想化拡張機能があるかどうかは、『Red Hat Enterprise Linux 6 仮想化管理ガイド』を参照してください。

### ストレージのサポート

ゲスト仮想マシンのストレージ方法は次のとおりです。

- ファイル（ローカルストレージ上）
- 物理ディスクパーティション
- ローカルに接続された物理 LUN
- LVM パーティション
- NFS 共有ファイルシステム
- iSCSI,
- GFS2 クラスターファイルシステム
- ファイバーチャネルベースの LUN
- Fibre Channel over Ethernet (FCoE)

# 第3章 KVM ゲスト仮想マシンの互換性

プロセッサーが仮想化拡張機能に対応しているかどうか、およびそれらが無効な場合に仮想化拡張機能を有効にする方法は、『Red Hat Enterprise Linux Virtualization Administration Guide 』 を参照してください。

## 3.1. RED HAT ENTERPRISE LINUX 6 のサポート制限

Red Hat Enterprise Linux 6 サーバーには、特定のサポート制限があります。

以下の URL では、Red Hat Enterprise Linux のプロセッサーとメモリーの制限を説明します。

- ホストシステムの場合： http://www.redhat.com/resourcelibrary/articles/articles-red-hat-enterprise-linux-6-technology-capabilities-and-limits
- ハイパーバイザーの場合： http://www.redhat.com/resourcelibrary/articles/virtualization-limits-rhel-hypervisors

**注記**

Red Hat Enterprise Linux 6.5 は、KVM ゲストごとに 4TiB のメモリーに対応するようになりました。

以下の URL は、サポートされるオペレーティングシステムとホストとゲストの組み合わせを示す完全なリファレンスです。

- http://www.redhat.com/resourcelibrary/articles/enterprise-linux-virtualization-support

## 3.2. サポートされている CPU モデル

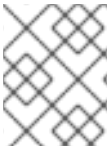
すべてのハイパーバイザーには、ゲストにデフォルトで表示される CPU 機能に関する独自のポリシーがあります。QEMU/KVM によりゲストに提示される CPU 機能のセットは、ゲスト仮想マシンの設定で選択された CPU モデルによって異なります。***qemu32 qemu64*** は基本的な CPU モデルですが、他のモデル（追加機能付き）も利用できます。

Red Hat Enterprise Linux 6 は、以下の QEMU CPU モデル定義の使用をサポートします。

```
<!-- This is only a partial file, only containing the CPU models. The XML file has more information
(including supported features per model) which you can see when you open the file yourself -->
<cpus>
  <arch name='x86'>
...

    <!-- Intel-based QEMU generic CPU models -->
    <model name='pentium'>
      <model name='486'/>
     </model>

    <model name='pentium2'>
      <model name='pentium'/>
    </model>

    <model name='pentium3'>
```

```
  <model name='pentium2'/>
</model>

<model name='pentiumpro'>
</model>

<model name='coreduo'>
  <model name='pentiumpro'/>
  <vendor name='Intel'/>
</model>

<model name='n270'>
  <model name='coreduo'/>
</model>

<model name='core2duo'>
  <model name='n270'/>
</model>

<!-- Generic QEMU CPU models -->
<model name='qemu32'>
  <model name='pentiumpro'/>
</model>

<model name='kvm32'>
  <model name='qemu32'/>
</model>

<model name='cpu64-rhel5'>
  <model name='kvm32'/>
</model>

<model name='cpu64-rhel6'>
  <model name='cpu64-rhel5'/>
</model>

<model name='kvm64'>
  <model name='cpu64-rhel5'/>
</model>

<model name='qemu64'>
  <model name='kvm64'/>
</model>

<!-- Intel CPU models -->
<model name='Conroe'>
  <model name='pentiumpro'/>
  <vendor name='Intel'/>
</model>

<model name='Penryn'>
  <model name='Conroe'/>
</model>

<model name='Nehalem'>
  <model name='Penryn'/>
```

```
    </model>

    <model name='Westmere'>
      <model name='Nehalem'/>
      <feature name='aes'/>
    </model>

    <model name='SandyBridge'>
      <model name='Westmere'/>
    </model>

    <model name='Haswell'>
      <model name='SandyBridge'/>
    </model>

    <!-- AMD CPUs -->
    <model name='athlon'>
      <model name='pentiumpro'/>
      <vendor name='AMD'/>
     </model>

    <model name='phenom'>
      <model name='cpu64-rhel5'/>
      <vendor name='AMD'/>
    </model>

    <model name='Opteron_G1'>
      <model name='cpu64-rhel5'/>
      <vendor name='AMD'/>
    </model>

    <model name='Opteron_G2'>
      <model name='Opteron_G1'/>
    </model>

    <model name='Opteron_G3'>
      <model name='Opteron_G2'/>
    </model>

    <model name='Opteron_G4'>
      <model name='Opteron_G2'/>
    </model>

    <model name='Opteron_G5'>
      <model name='Opteron_G4'/>
    </model>
  </arch>
</cpus>
```



**注記**

対応している CPU モデルと認識された CPUID フラグの全一覧も、**qemu-kvm -cpu ?** コマンドを使用して確認することができます。

# 第4章 仮想化の制限

本章では、Red Hat Enterprise Linux 6 における仮想化パッケージの追加サポートおよび製品の制限について説明します。

## 4.1. KVM の制限

KVM ハイパーバイザーには以下の制限が適用されます。

**ゲストごとの vCPU の最大数**

ゲストごとにサポートされる仮想 CPU の最大量は、ホストマシンとして使用している Red Hat Enterprise Linux 6 のマイナーバージョンによって異なります。6.0 のリリースでは最大 64 が導入されましたが、6.3 では最大 160 が導入されました。バージョン 6.7 の時点で、ゲストごとに最大 240 個の仮想 CPU がサポートされます。

**定数 TSC ビット**

Constant タイムスタンプカウンターのないシステムには、追加の設定が必要です。Constant タイムスタンプカウンターがあるかどうかの判断や、関連する問題を修正するための設定手順に関する詳細は、*14章KVM ゲストのタイミング管理* を参照してください。

**仮想化 SCSI デバイス**

SCSI エミュレーションは、Red Hat Enterprise Linux の KVM ではサポートされていません。

**仮想化 IDE デバイス**

KVM は、ゲスト仮想マシンごとに最大 4 つの仮想化（エミュレート）IDE デバイスに制限されます。

**移行の制限**

デバイスの割り当ては、仮想マシンに公開されている物理デバイスを、その仮想マシン専用で使用することを指します。デバイスの割り当ては、仮想マシンが実行される特定のホストのハードウェアを使用するため、デバイスの割り当てを使用中の際は、移行と保存/復元はサポートされません。ゲストオペレーティングシステムがホットプラグに対応している場合は、移行前に割り当てたデバイスを削除し、この機能を有効にするために保存/復元を行うことができます。

ライブマイグレーションは、同じ CPU タイプのホスト間でのみ可能です (つまり、Intel から Intel、または AMD から AMD のみ)。

ライブマイグレーションの場合、両方のホストで、NX (No eXecution) ビットが同じ値 (**on** または **off** のいずれか) に設定されている必要があります。

移行を機能させるには、書き込みモードで開いているすべてのブロックデバイスに **cache=none** を指定する必要があります。

**警告**

**cache=none** を指定しないと、ディスクが破損する可能性があります。

**ストレージの制限**

ゲスト仮想マシンに、ディスク全体またはブロックデバイス (**/dev/sdb** など) への書き込みアクセスを提供すると、それに関連するリスクが発生します。ゲスト仮想マシンがブロックデバイス全体にアクセスできる場合は、ホストマシンとボリュームラベルまたはパーティションテーブルを共有できます。ホストシステムのパーティション認識コードにバグが存在すると、セキュリティーリスクが発生する可能性があります。ゲスト仮想マシンに割り当てられたデバイスを無視するようにホストマシンを設定することで、このリスクを回避します。

**警告**

ストレージの制限に従わないと、セキュリティーのリスクが発生する可能性があります。

**コアダンプの制限**

コアダンプは移行と同じインフラストラクチャーを使用し、デバイス割り当てよりもより多くのデバイスナレッジと制御が必要になります。したがって、デバイスの割り当てが使用されている場合は、コアダンプはサポートされません。

## 4.2. アプリケーションの制限

仮想化には、特定タイプのアプリケーションには適さない側面があります。

I/O スループット要件が高いアプリケーションは、完全仮想化ゲストに準仮想化ドライバーを使用する必要があります。準仮想化ドライバーがないと、特定のアプリケーションは、大量の I/O 負荷では予測できない可能性があります。

I/O 要件が高いため、以下のアプリケーションは使用しないでください。

- **kdump**サーバー
- **netdump**サーバー

I/O を多用したり、リアルタイムパフォーマンスを必要としたりするアプリケーションやツールを慎重に評価する必要があります。I/O パフォーマンスを向上させるために、準仮想化ドライバーまたは PCI デバイスの割り当てを検討してください。完全仮想化ゲストの準仮想化ドライバーの詳細は、*10章KVM 準仮想化 (virtio) ドライバー* を参照してください。PCI デバイスの割り当ての詳細は、*12章PCI デバイスの割り当て* を参照してください。

仮想環境でアプリケーションを実行すると、パフォーマンスがわずかに低下します。新しいハードウェアおよび高速ハードウェアへの統合による仮想化のパフォーマンス上の利点は、仮想化の使用に伴う潜在的なアプリケーションパフォーマンスの問題に対して評価する必要があります。

## 4.3. その他の制限

仮想化に影響する他のすべての制限および問題の一覧については、『Red Hat Enterprise Linux 6 Release Notes』 を参照してください。『Red Hat Enterprise Linux 6 リリースノート 』 では、現在の新機能、既知の問題、および更新または発見された制限を説明します。

# 第5章 仮想化パッケージのインストール

仮想化を使用する前に、コンピューターに仮想化パッケージをインストールする必要があります。仮想化パッケージは、ホストのインストールシーケンス時またはホストのインストール後に Subscription Manager でインストールできます。

KVM ハイパーバイザーは、**kvm** カーネルモジュールとともにデフォルトの Red Hat Enterprise Linux カーネルを使用します。

## 5.1. 仮想化ホストインストールの設定

本セクションでは、Red Hat Enterprise Linux の新規インストールの一部として仮想化ツールと仮想化パッケージをインストールする方法を説明します。

**注記**

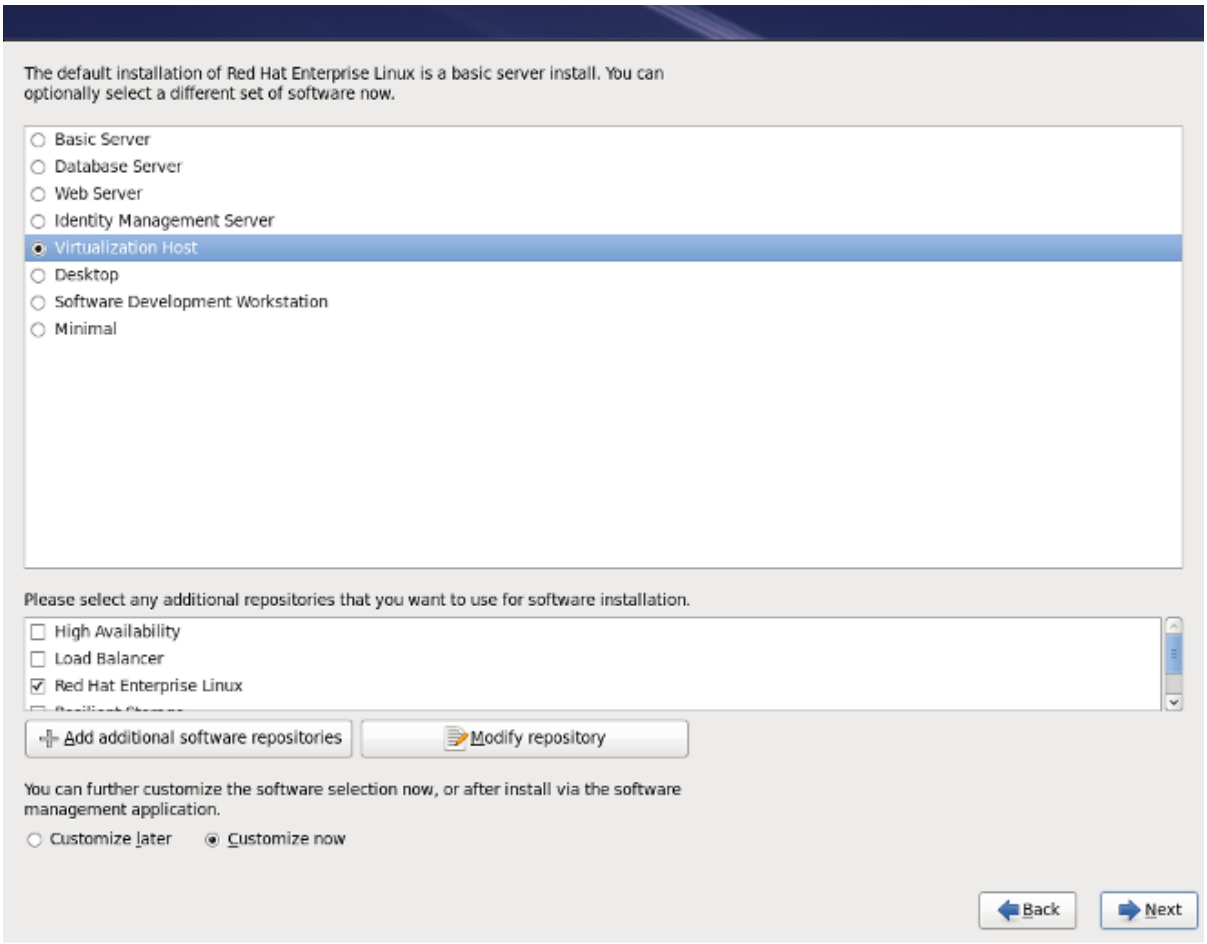
『Red Hat Enterprise Linux インストールガイド』では、Red Hat Enterprise Linux のインストールについて詳しく説明します。

**手順5.1 仮想化パッケージグループのインストール**

1. **Red Hat Enterprise Linux 6 インストールプログラムを起動します。**
   Red Hat Enterprise Linux インストール CD-ROM、DVD、または PXE から、インタラクティブな Red Hat Enterprise Linux 6 インストールを開始します。

2. **パッケージ選択までインストールを続行する**
   パッケージ選択のステップまで他の手順を完了します。

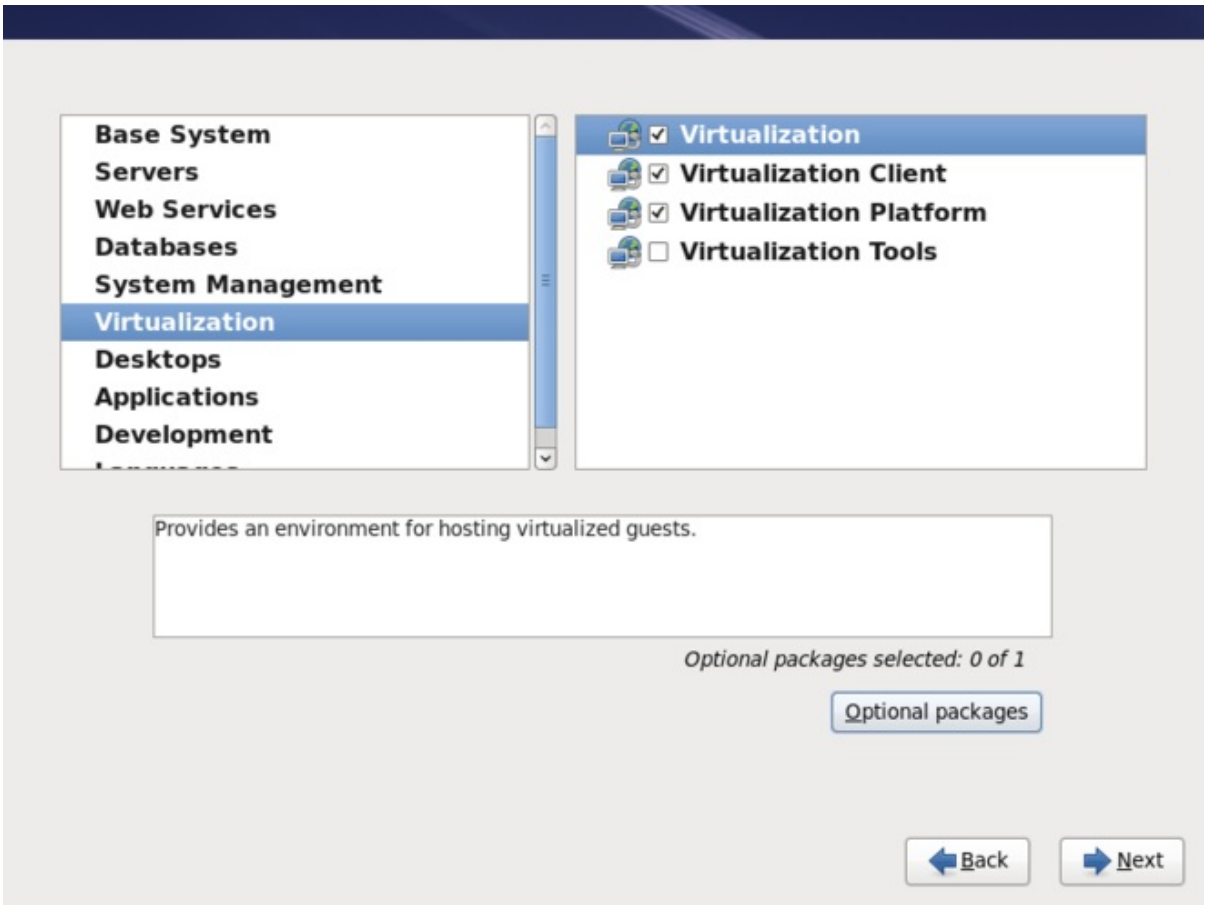
**図5.1 Red Hat Enterprise Linux パッケージ選択画面**




**Virtualization Host** server ロールを選択して、ゲスト仮想マシンのプラットフォームをインストールします。または、続行する前に **Customize Now** ラジオボタンが選択されていることを確認して、個々のパッケージを指定してください。

3. **仮想 化 パッケージグループの選択**
これにより、インストール用に qemu-kvm エミュレーター、**virt-manager**、**libvirt**、および **virt-viewer** を選択します。

図5.2 Red Hat Enterprise Linux パッケージ選択画面

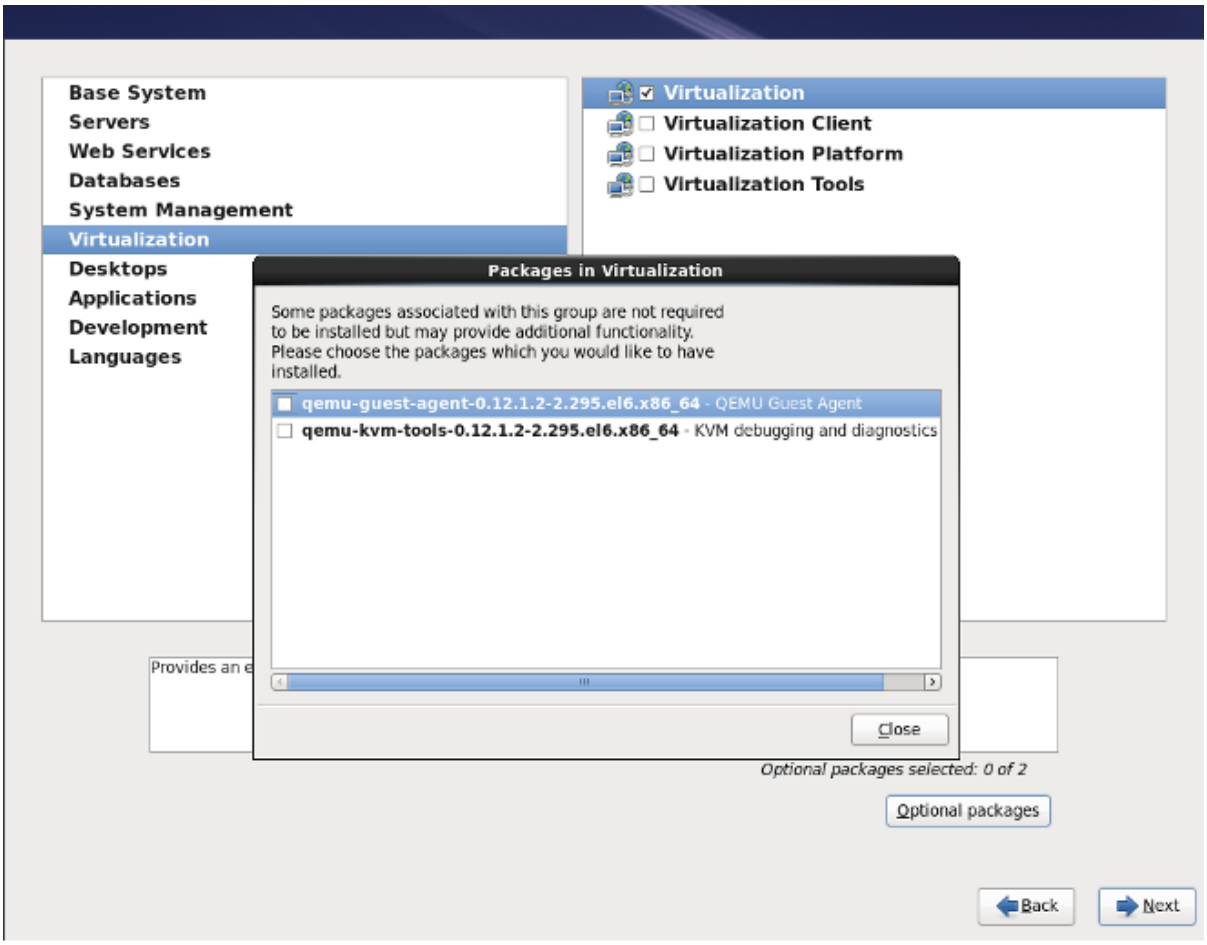



**注記**

後でグラフィカルユーザーインターフェイス(virt-manager)で仮想マシンを作成する場合は、**General Purpose Desktop** パッケージグループも選択する必要があります。

4. **パッケージのカスタマイズ（必要な場合）**
   他の仮想化パッケージが必要な場合は、仮想化グループをカスタマイズします。

図5.3 Red Hat Enterprise Linux パッケージ選択画面




**Close** ボタンをクリックし、**Next** ボタンをクリックしてインストールを続行します。

インストールが完了したら、システムを再起動します。

**重要**

仮想化パッケージの更新を受け取るには、有効な仮想化資格が必要です。

### キックスタートファイルを使用した KVM パッケージのインストール

キックスタートファイルを使用すると、ユーザーが各ホストシステムを手動でインストールすることなく、大規模な自動インストールが可能になります。本セクションでは、キックスタートファイルを作成して使用し、仮想化パッケージを使用して Red Hat Enterprise Linux をインストールする方法を説明します。

キックスタートファイルの **%packages** セクションで、以下のパッケージグループを追加します。

```
@virtualization
@virtualization-client
@virtualization-platform
@virtualization-tools
```

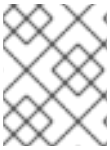
キックスタートファイルの詳細は、『Red Hat Enterprise Linux インストールガイド』を参照してください。

## 5.2. 既存の RED HAT ENTERPRISE LINUX システムへの仮想化パッケージのインストール

本セクションでは、稼働中の Red Hat Enterprise Linux 6 以降のシステムに KVM ハイパーバイザーをインストールする手順を説明します。

パッケージをインストールするには、マシンが登録されている必要があります。Red Hat Subscription Manager を使用して登録する場合は、**subscription-manager register** コマンドを実行し、プロンプトに従います。

有効な Red Hat サブスクリプションをお持ちでない場合は、Red Hat オンラインストア にアクセスして取得してください。



**注記**

Red Hat Network (RHN)は非推奨になりました。これで、登録タスクには Subscription Manager が使用されるはずです。

### yumを使用した仮想化パッケージのインストール

Red Hat Enterprise Linux で仮想化を使用するには、少なくとも **qemu-kvm** パッケージおよび **qemu-img** パッケージが必要です。これらのパッケージは、ホストの Red Hat Enterprise Linux システム上のユーザーレベルの KVM エミュレーターおよびディスクイメージマネージャーを提供します。

**qemu-kvm** パッケージおよび **qemu-img** パッケージをインストールするには、以下のコマンドを実行します。

```
# yum install qemu-kvm qemu-img
```

追加の仮想化管理パッケージも利用できます。

#### 推奨される仮想化パッケージ

**python-virtinst**

仮想マシンを作成するための **virt-install** コマンドを提供します。

**libvirt**

libvirt パッケージは、ハイパーバイザーおよびホストシステムと対話するためのサーバーおよびホストサイドライブラリーを提供します。libvirt パッケージは、ライブラリー呼び出しを処理し、仮想マシンを管理し、ハイパーバイザーを制御する **libvirtd** デーモンを提供します。

**libvirt-python**

libvirt-python パッケージには、Python プログラミング言語で書かれたアプリケーションが libvirt API が提供するインターフェイスを使用できるようにするモジュールが含まれています。

**virt-manager**

**virt-manager** ( Virtual Machine Manager とも呼ばれる)は、仮想マシンを管理するためのグラフィカルツールを提供します。libvirt-client ライブラリーを管理 API として使用します。

**libvirt-client**

libvirt-client パッケージは、libvirt サーバーにアクセスするためのクライアント側の API およびライブラリーを提供します。libvirt-client パッケージには、コマンドラインまたは特別な仮想化シェルか

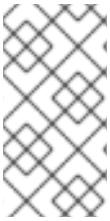
ら仮想マシンおよびハイパーバイザーを管理および制御する **virsh** コマンドラインツールが含まれます。

以下のコマンドを使用して、推奨される仮想化パッケージをすべてインストールします。

```
# yum install virt-manager libvirt libvirt-python python-virtinst libvirt-client
```

### 仮想化パッケージグループのインストール

仮想化パッケージは、パッケージグループからインストールすることもできます。次の表では、仮想化パッケージグループと、その提供内容を説明します。



**注記**

**qemu-img** パッケージは、システムにインストールされていない場合は、仮想 **化** パッケージグループの依存関係として にインストールされていることに注意してください。前述した **yum install qemu-img** コマンドを使用して手動でインストールすることもできます。

**表5.1 仮想化パッケージグループ**

| パッケージグループ | 説明 | 必須パッケージ | オプションパッケージ |
|---|---|---|---|
| 仮想化 | 仮想マシンをホストする環境を提供します。 | qemu-kvm | qemu-guest-agent, qemu-kvm-tools |
| Virtualization Client | 仮想化インスタンスのインストールおよび管理を行うクライアント | python-virtinst、virt-manager、virt-viewer | virt-top |
| Virtualization Platform | 仮想マシンおよびコンテナーへのアクセスと制御を行うインターフェイスを提供します。 | libvirt、libvirt-client、virt-who、virt-what | fence-virtd-libvirt、fence-virtd-multicast、fence-virtd-serial、libvirt-cim、libvirt-java、libvirt-qmf、libvirt-snmp、perl-Sys-Virt |
| Virtualization Tools | オフラインの仮想イメージ管理用ツール | libguestfs | libguestfs-java, libguestfs-tools, virt-v2v |

パッケージグループをインストールするには、**yum groupinstall <groupname>** コマンドを実行します。たとえば、仮想化ツール パッケージグループをインストールするには、**yum groupinstall " Virtualization Tools "** コマンドを実行します。

# 第6章 ゲスト仮想マシンのインストールの概要

ホストシステムに仮想化パッケージをインストールした後に、ゲストオペレーティングシステムを作成できます。本章では、仮想マシンにゲストオペレーティングシステムをインストールするための一般的なプロセスについて説明します。**virt-manager** の **New** ボタンを使用してゲスト仮想マシンを作成するか、またはコマンドラインインターフェイス **virt-install** を使用できます。この章では、両方の方法について説明します。

詳細なインストール手順は、Red Hat Enterprise Linux および Microsoft Windows の特定のバージョンについて以下の章を参照してください。

## 6.1. ゲスト仮想マシンの前提条件および考慮事項

ゲスト仮想マシンを作成する場合は、さまざまな要因を考慮してください。デプロイメントの前に仮想マシンのロールを考慮する必要があるだけでなく、変数要素（負荷、クライアントの量）に基づいて定期的な監視と評価を実行する必要があります。要因には、以下が含まれます。

**パフォーマンス**

ゲスト仮想マシンは、目的のタスクに基づいてデプロイおよび設定する必要があります。一部のゲストシステム (データベースサーバーを実行しているゲストなど) では、パフォーマンスに関する特別な考慮事項が必要になる場合があります。ゲストは、そのロールと、予測されるシステム負荷に基づいて、より多くの割り当てられた CPU またはメモリーを必要とする場合があります。

**入出力の要件と入出力の種類**

ゲスト仮想マシンによっては、I/O 要件が特に高い場合や、I/O のタイプ (通常のディスクブロックサイズのアクセスやクライアント数など) に基づいて、さらに検討または予測が必要になる場合があります。

**ストレージ**

ゲスト仮想マシンによっては、ストレージまたは高速なディスクタイプへのより高い優先度のアクセスが必要な場合や、ストレージの領域への排他的アクセスが必要な場合があります。ゲストが使用するストレージの量も定期的に監視され、ストレージのデプロイメントおよびメンテナーンスの際に考慮される必要があります。

**ネットワークとネットワークインフラストラクチャー**

使用環境によっては、ゲスト仮想マシンの中には、他のゲストよりも速いネットワークリンクが必要なものもあります。帯域幅または遅延は、特に要件や負荷の変更として、ゲストのデプロイメントおよびメンテナーンスを行う際の要因となることがよくあります。

**要件の要求**

virtio ドライブがディスク全体でサポートされ、ディスクデバイスパラメーターが **lun** に設定されている場合、virtio ドライブのゲスト仮想マシンにのみ SCSI 要求を発行できます。以下に例を示します。

```
<devices>
  <emulator>/usr/libexec/qemu-kvm</emulator>
  <disk type='block' device='lun'>
```

## 6.2. VIRT-INSTALL を使用したゲストの作成

**virt-install** コマンドを使用して、コマンドラインからゲスト仮想マシンを作成できます。**virt-install** は、対話式またはスクリプトの一部として使用して、仮想マシンの作成を自動化します。**virt-install** をキックスタートファイルで使用すると、仮想マシンの無人インストールが可能になります。

**virt-install** ツールは、コマンドラインで渡すことができるオプションを多数提供します。オプションの一覧を表示するには、以下のコマンドを実行します。

```
# virt-install --help
```

**virt-install** コマンドを正常に実行するには、root 権限が必要であることに注意してください。**virt-install** の man ページには、各コマンドオプションと重要な変数が記載されています。

**qemu-img** は、ストレージオプションの設定に **virt-install** の前に使用できる関連コマンドです。

重要なオプションは、***--graphics*** オプションで、仮想マシンのグラフィカルインストールを可能にします。

**例6.1 virt-install を使用した Red Hat Enterprise Linux 5 ゲスト仮想マシンのインストール**

この例では、Red Hat Enterprise Linux 5 ゲストを作成します。

```
virt-install \
  --name=guest1-rhel5-64 \
  --file=/var/lib/libvirt/images/guest1-rhel5-64.dsk \
  --file-size=8 \
  --nonsparse --graphics spice \
  --vcpus=2 --ram=2048 \
  --location=http://example1.com/installation_tree/RHEL5.6-Server-x86_64/os \
  --network bridge=br0 \
  --os-type=linux \
  --os-variant=rhel5.4
```

このコマンドを実行する際に、オペレーティングシステムの正しい ***os-type*** を選択してください。

その他の例は、**man virt-install** を参照してください。

**注記**

**virt-install** を使用して Windows ゲストをインストールする場合は、**--os-type=*windows*** オプションが推奨されます。このオプションは、インストール手順中に再起動すると、CD-ROM が切断されないようにします。**--os-variant** オプションは、特定のゲストオペレーティングシステムの設定をさらに最適化します。

インストールが完了したら、ゲストオペレーティングシステムに接続できます。詳細は、を参照してください。 「仮想マシンへの接続」

## 6.3. VIRT-MANAGER を使用したゲストの作成

**virt-manager** (Virtual Machine Manager とも呼ばれる)は、ゲスト仮想マシンを作成および管理するためのグラフィカルツールです。

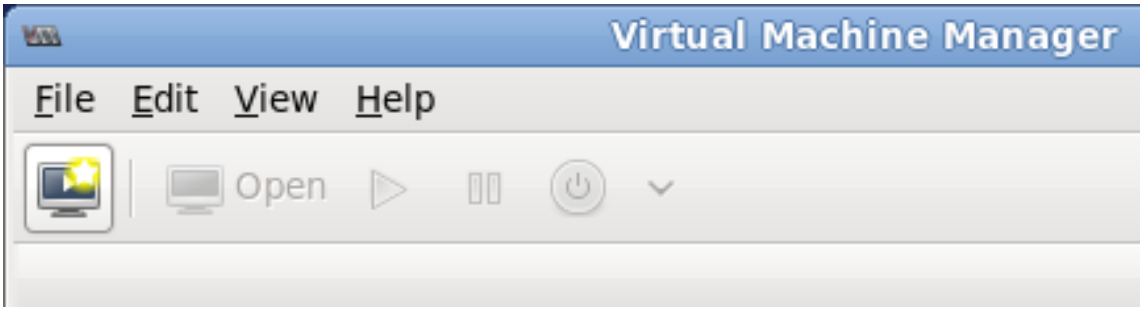
**手順6.1virt-managerを使用したゲスト仮想マシンの作成**

1. **virt-manager を開く**
   **virt-manager** を起動します。Applications メニューおよび System Tools サブメニューから Virtual Machine Manager アプリケーションを起動します。または、root で **virt-manager** を実行します。

2. **オプション：リモートハイパーバイザーを開く**
   ハイパーバイザーを選択し、**Connect** ボタンをクリックしてリモートハイパーバイザーに接続します。

3. **新しい仮想マシンを作成します。**
   virt-manager ウィンドウを使用すると、新しい仮想マシンを作成できます。**Create a new virtual machine** ボタン(図6.1「Virtual Machine Manager ウィンドウ」 )をクリックし、New VM ウィザードを開きます。

   **図6.1 Virtual Machine Manager ウィンドウ**

   New VM ウィザードでは、仮想マシンの作成プロセスが 5 つの手順に分けられます。

   1. ゲスト仮想マシンの命名およびインストールタイプの選択
   2. インストールメディアの場所と設定
   3. メモリーおよび CPU オプションの設定
   4. 仮想マシンのストレージの設定
   5. ネットワーク、アーキテクチャー、およびその他のハードウェア設定の設定

   続行する前に、**virt-manager** がインストールメディア (ローカルまたはネットワーク上) にアクセスできることを確認してください。

4. **名前およびインストールタイプを指定します。**
   ゲスト仮想マシンの作成プロセスは、名前とインストールタイプを選択して開始します。仮想マシン名には、アンダースコア(_)、ピリオド(.)、およびハイフン(-)を指定できます。

**図6.2 仮想マシンに名前を付け、インストール方法を選択します。**


仮想マシン名を入力して、インストールタイプを選択します。

**ローカルインストールメディア (ISO イメージまたは CDROM)**

この方法では、CD-ROM、DVD、またはインストールディスクのイメージ( **.iso**など)を使用します。

**ネットワークインストール (HTTP、FTP、または NFS)**

この方法では、ミラーリングした Red Hat Enterprise Linux または Fedora インストールツリーを使用してゲストをインストールします。インストールツリーには、HTTP、FTP、または NFS のいずれかを介してアクセスできる必要があります。

**ネットワーク起動 (PXE)**

この方法では、PXE (Preboot eXecution Environment) サーバーを使用して、ゲスト仮想マシンをインストールします。PXE サーバーの設定については、デプロイメントガイド で説明されています。『』ネットワークブートを使用してインストールするには、ゲストにルーティング可能な IP アドレスまたは共有ネットワークデバイスが必要です。PXE インストールに必要なネットワーク設定の詳細は、「PXE を使用したゲスト の作成」 を参照してください。

**既存のディスクイメージのインポート**

この方法を使用すると、新しいゲスト仮想マシンを作成し、(インストール済みの起動可能なオペレーティングシステムを含む) ディスクイメージを読み込むことができます。

**進む** をクリックして続けます。

5. **インストールの設定**
次に、インストールの **OS type** と **Version** を設定します。仮想マシンに適した OS タイプを選択してください。インストールの方法に応じて、インストール URL または既存のストレージパスを指定します。

**図6.3 リモートインストール URL**

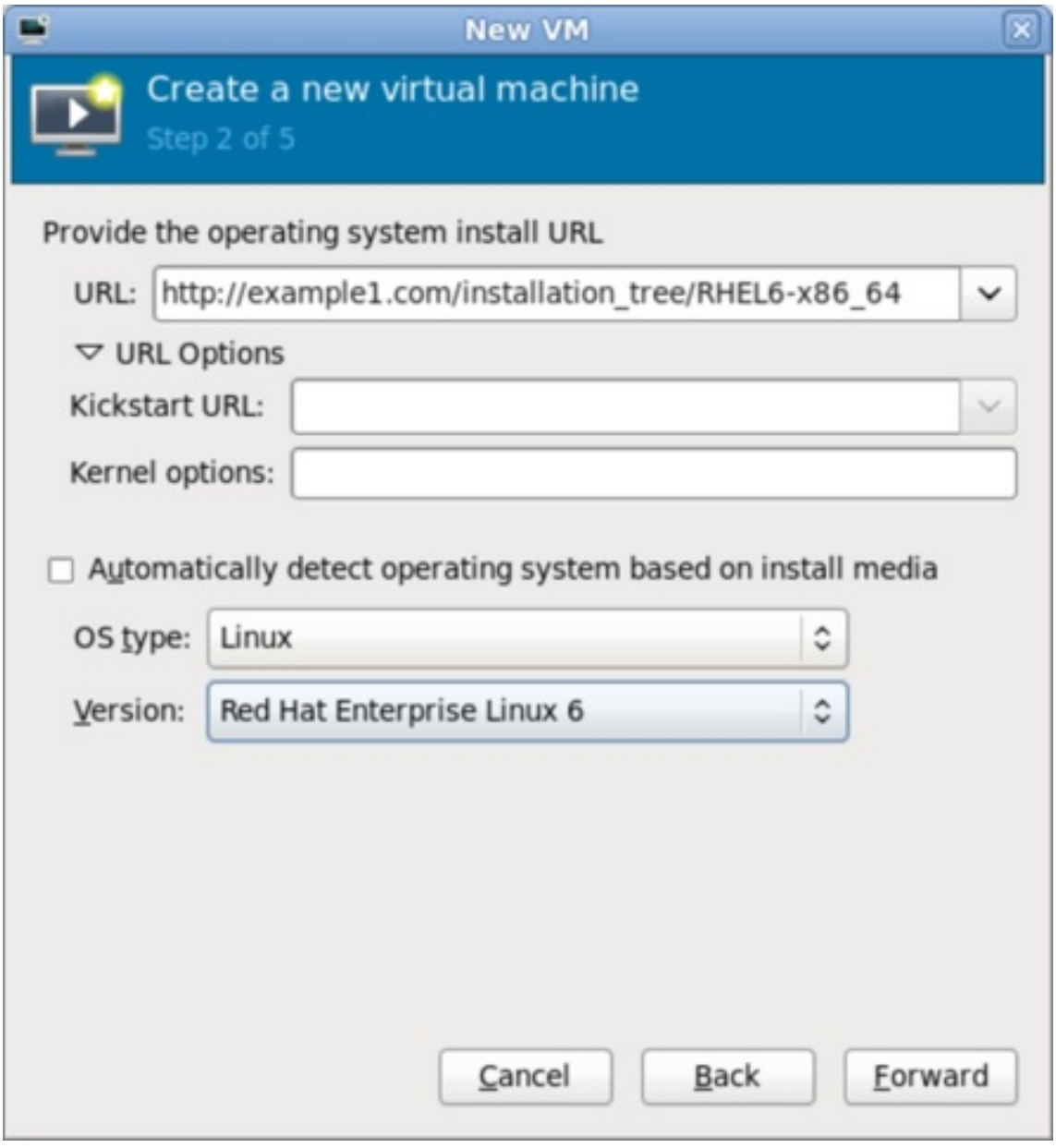
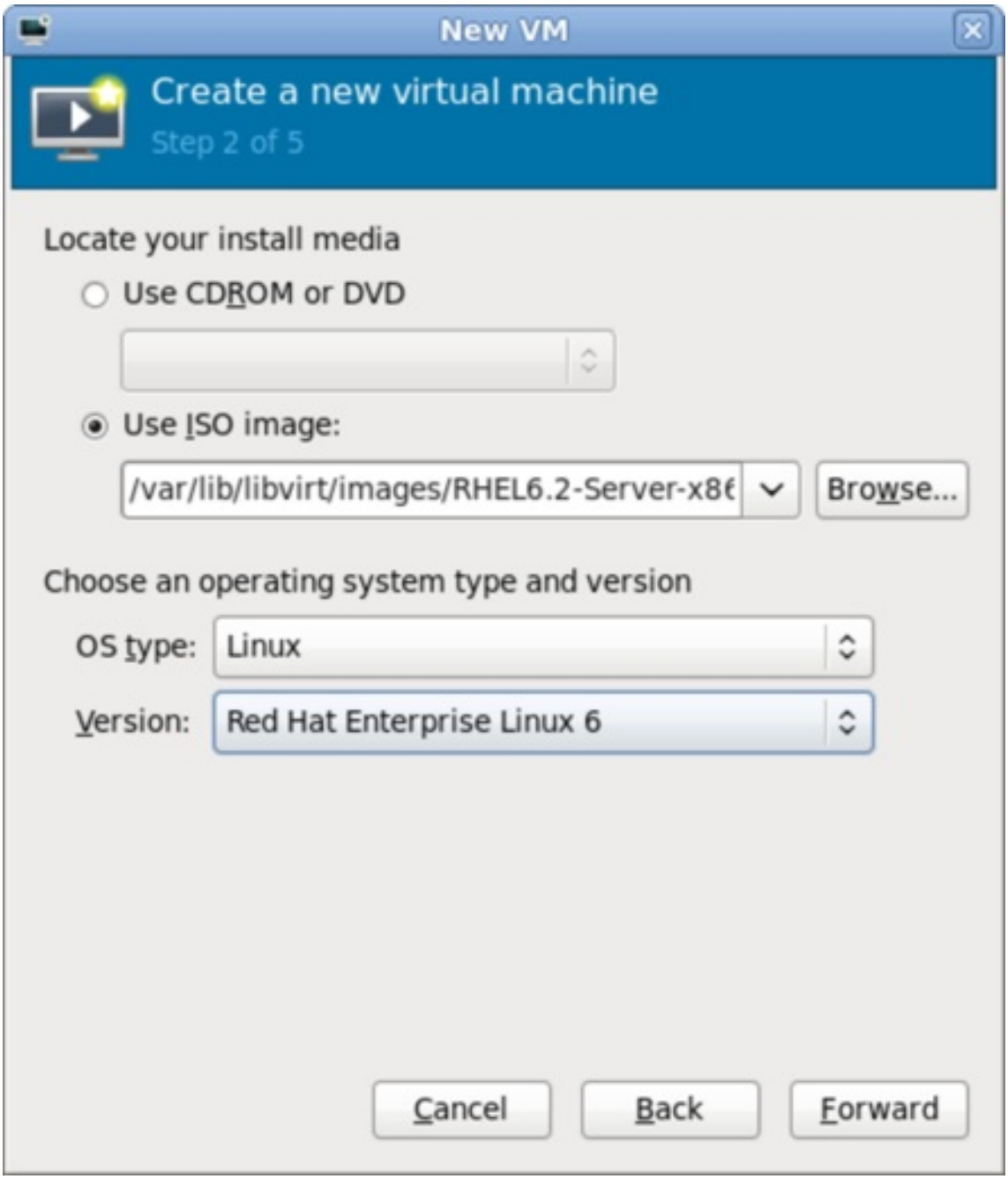
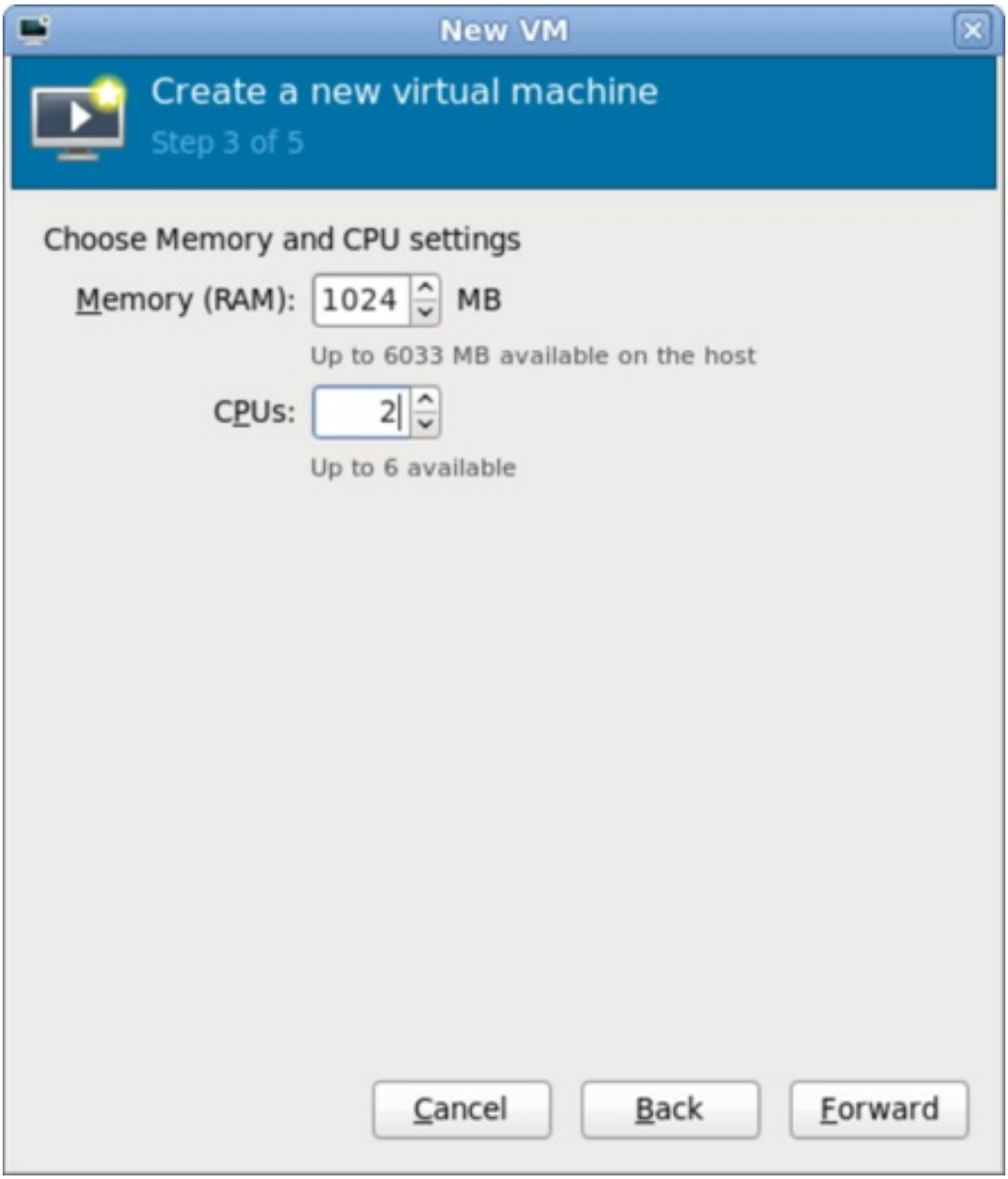


**図6.4 ローカル ISO イメージのインストール**



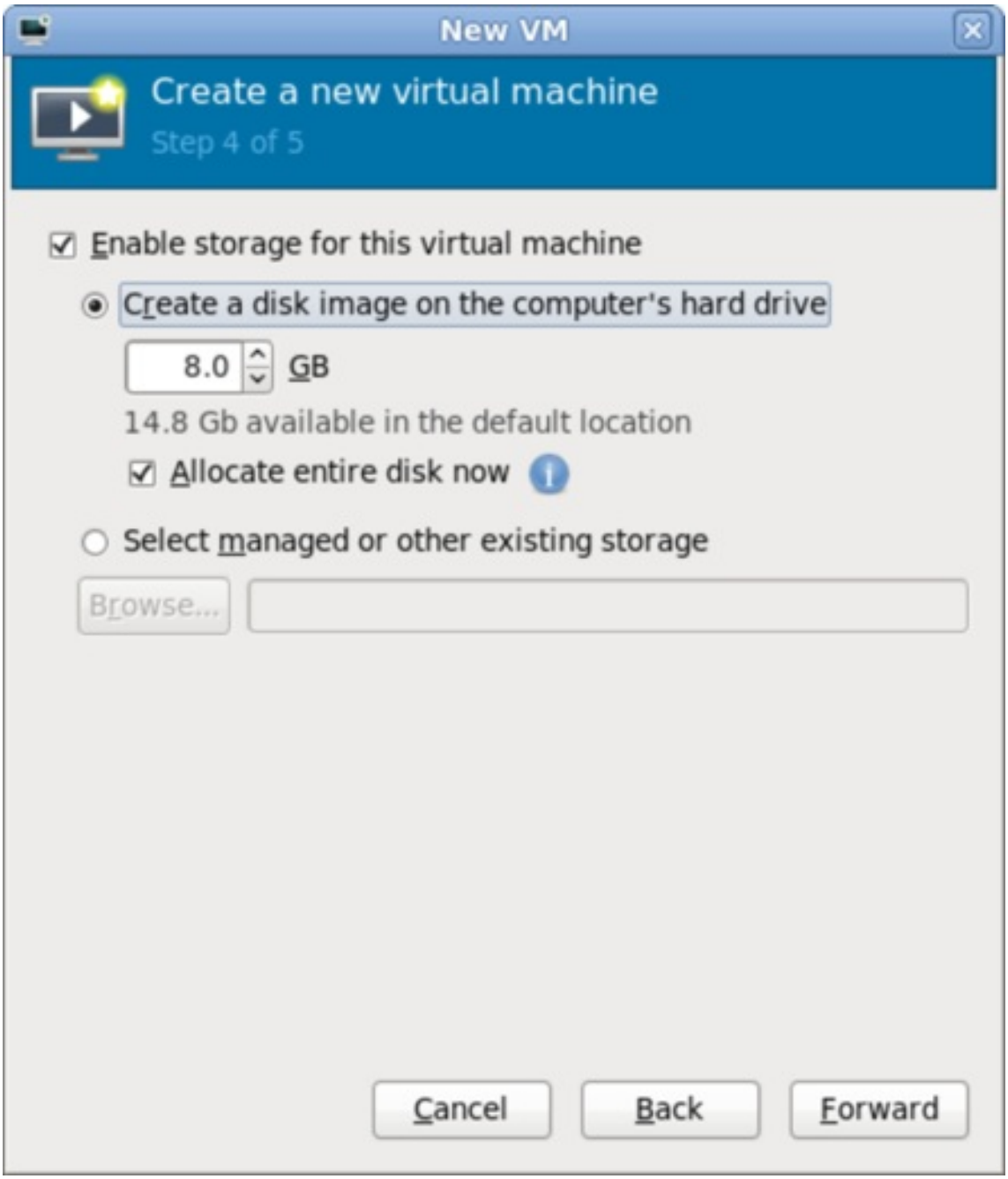

6. **CPU およびメモリーの設定**
   次の手順では、仮想マシンに割り当てる CPU の数およびメモリー量を設定する必要があります。ウィザードには、割り当てることのできる CPU の数とメモリー量が表示されます。これらの設定を設定し、**Forward** をクリックします。

**図6.5 CPU およびメモリーの設定**




7. **ストレージを設定します。**
   ゲスト仮想マシンにストレージを割り当てます。

**図6.6 仮想ストレージの設定**

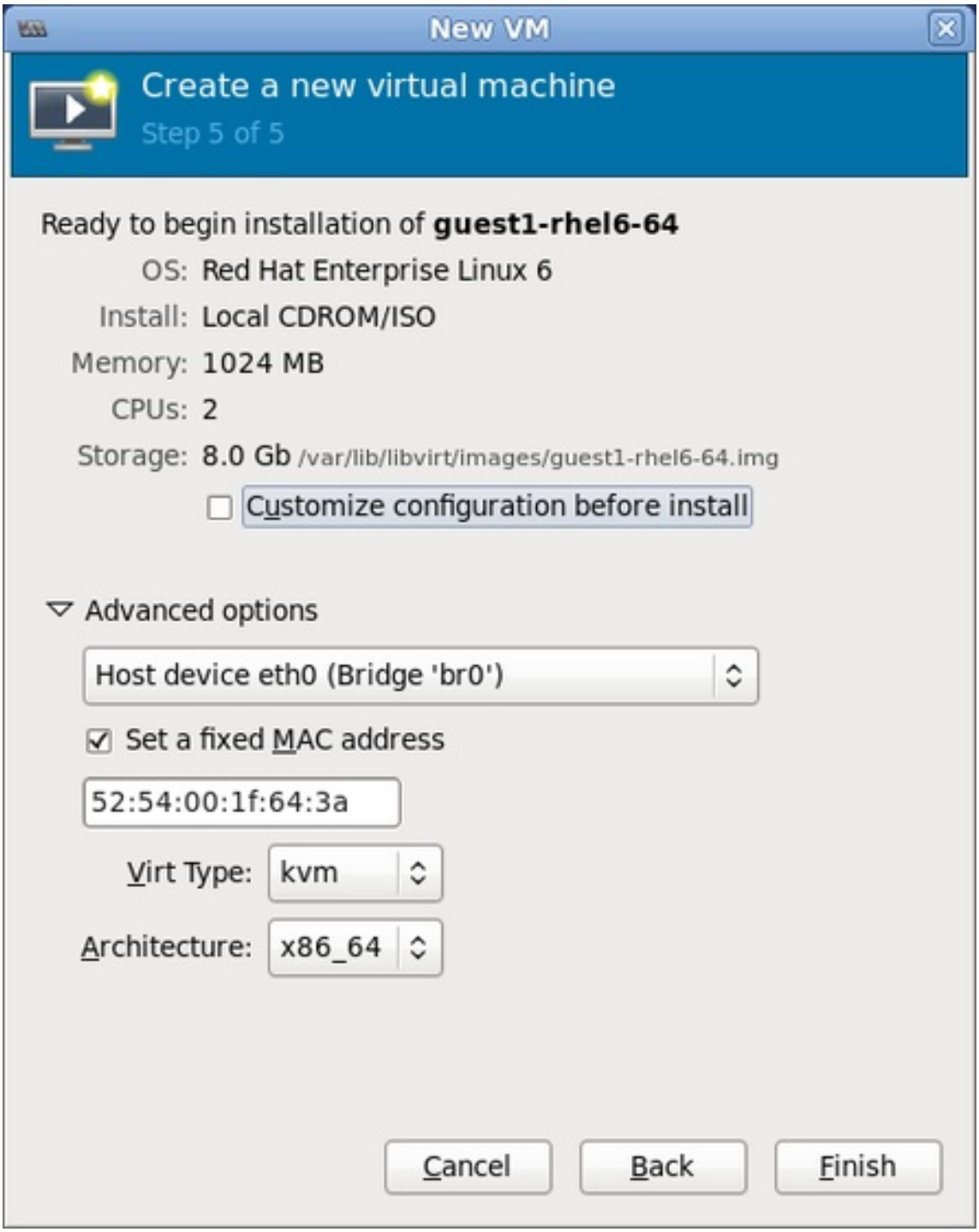



最初のステップで既存のディスクイメージのインポートを選択した場合、**virt-manager** はこの手順を省略します。

仮想マシンと必要なアプリケーションに十分な領域を割り当て、**Forward** をクリックして続行します。

8. **最終的な設定**
   仮想マシンの設定を確認し、問題がなければ **Finish** をクリックします。これにより、デフォルトのネットワーク設定、仮想化タイプ、およびアーキテクチャーで仮想マシンが作成されます。

**図6.7 設定の確認**




最初に仮想マシンのハードウェアをさらに設定する場合は、**Finish** をクリックする前に **Customize configuration before install** のボックスにチェックを入れます。これを実行すると、別のウィザードが開き、仮想マシンのハードウェア設定の追加、削除、および設定が可能になります。

仮想マシンのハードウェアを設定したら、**Apply** をクリックします。その後、**virt-manager** は、指定したハードウェア設定で仮想マシンを作成します。

9. インストールが完了したら、ゲストオペレーティングシステムに接続できます。詳細は、を参照してください。 「仮想マシンへの接続」

## 6.4. PXE を使用したゲスト の作成

このセクションでは、PXE でゲスト を作成する方法を説明します。

**要件**

PXE ゲストのインストールには、インストールするゲスト仮想マシンと同じサブネットで実行している PXE サーバーが必要です。これを実行する方法は、仮想マシンがネットワークに接続される方法によって異なります。PXE サーバーの設定が必要な場合は、サポートにお問い合わせください。

### virt-install を使用した PXE インストール

**virt-install** PXE インストールには、**--network=bridge:installation** パラメーターの両方が必要です。ここで、***installation*** はブリッジの名前および ***--pxe*** パラメーターです。

デフォルトでは、ネットワークが見つからない場合、ゲスト仮想マシンは別の起動可能なデバイスから起動しようとします。他の起動可能なデバイスが見つからない場合は、ゲストが一時停止します。**qemu-kvm** ブートパラメーター ***reboot-timeout*** を使用して、起動可能なデバイスが見つからなかった場合にゲストが起動を再試行できるようにします。以下に例を示します。

```
# qemu-kvm -boot reboot-timeout=1000
```

**例6.2 virt-install を使用した完全仮想化 PXE インストール**

```
# virt-install --hvm --connect qemu:///system \
--network=bridge:installation --pxe --graphics spice \
--name rhel6-machine --ram=756 --vcpus=4 \
--os-type=linux --os-variant=rhel6 \
--disk path=/var/lib/libvirt/images/rhel6-machine.img,size=10
```

上記のコマンドはテキストのみの環境で実行できないことに注意してください。完全仮想化(***--hvm***)ゲストは、***--graphics spice*** パラメーターの代わりに ***--location*** および ***--extra-args "console=console_type"*** が指定された場合にのみ、テキストのみの環境にインストールできます。

**手順6.2 virt-manager を使用した PXE インストール**

1. **PXE を選択します。**
   インストール方法として PXE を選択し、残りの手順に従って OS タイプ、メモリー、CPU、およびストレージ設定を行います。

**図6.8 インストール方法の選択**

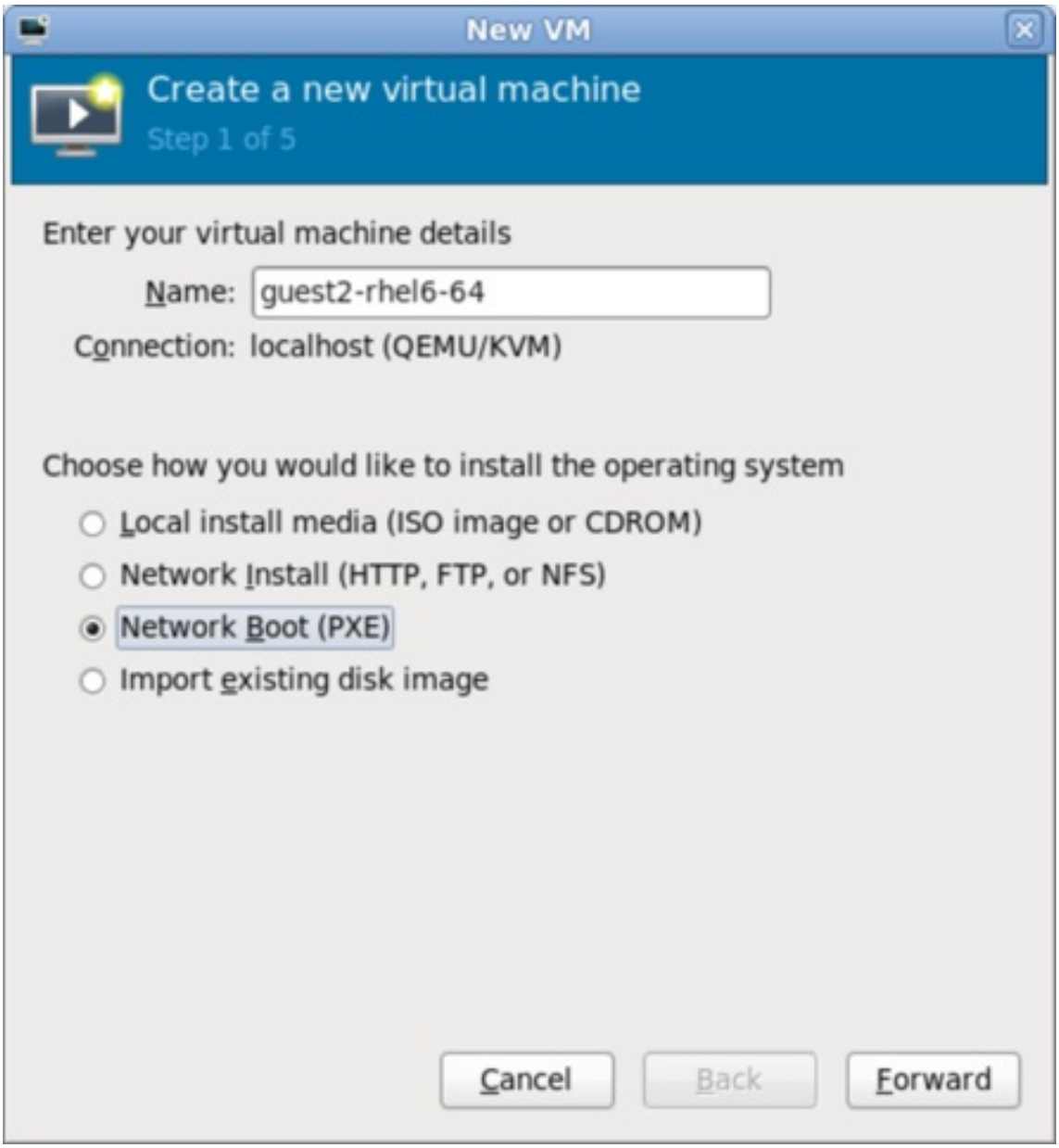


**図6.9 インストールタイプの選択**

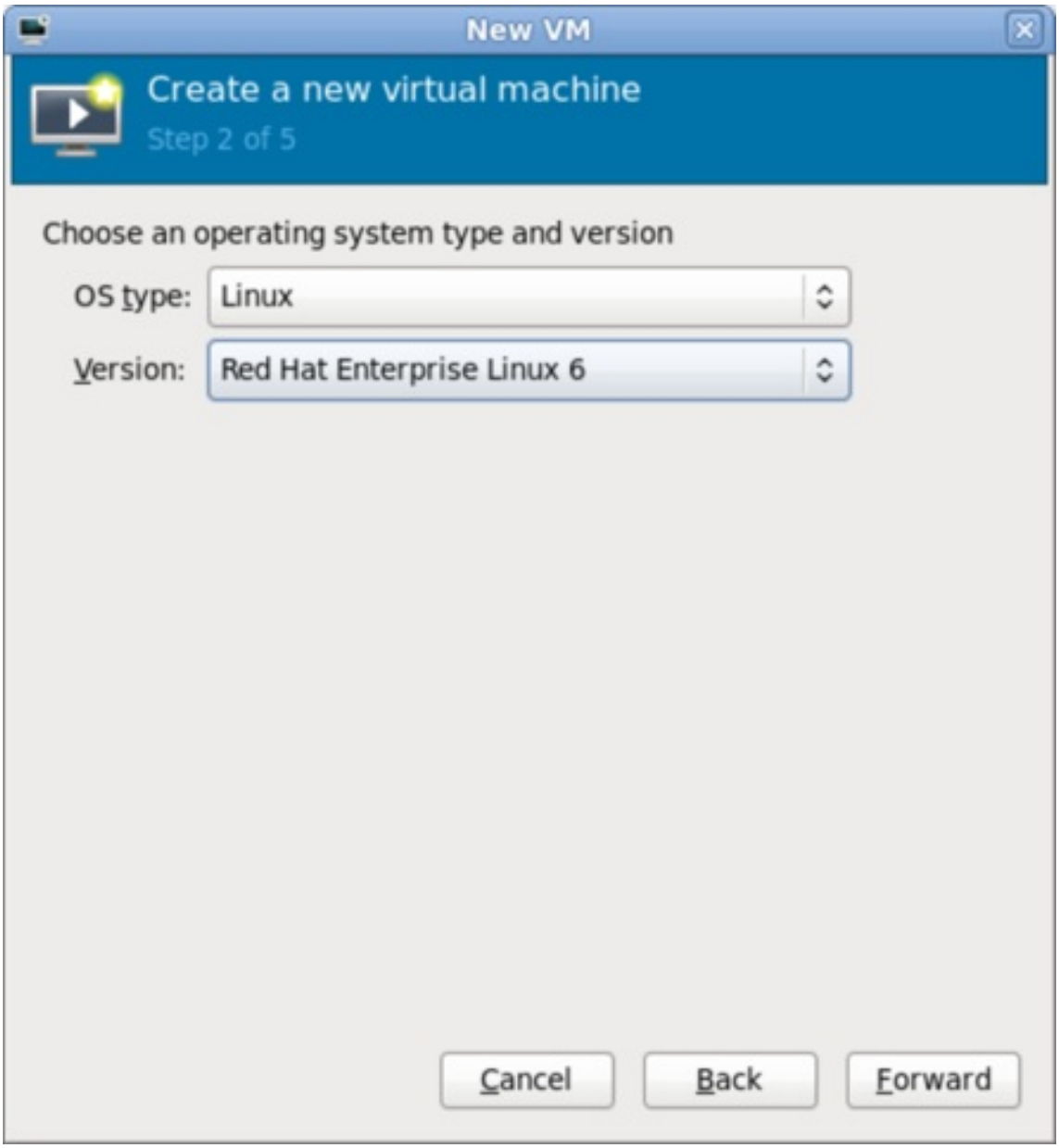
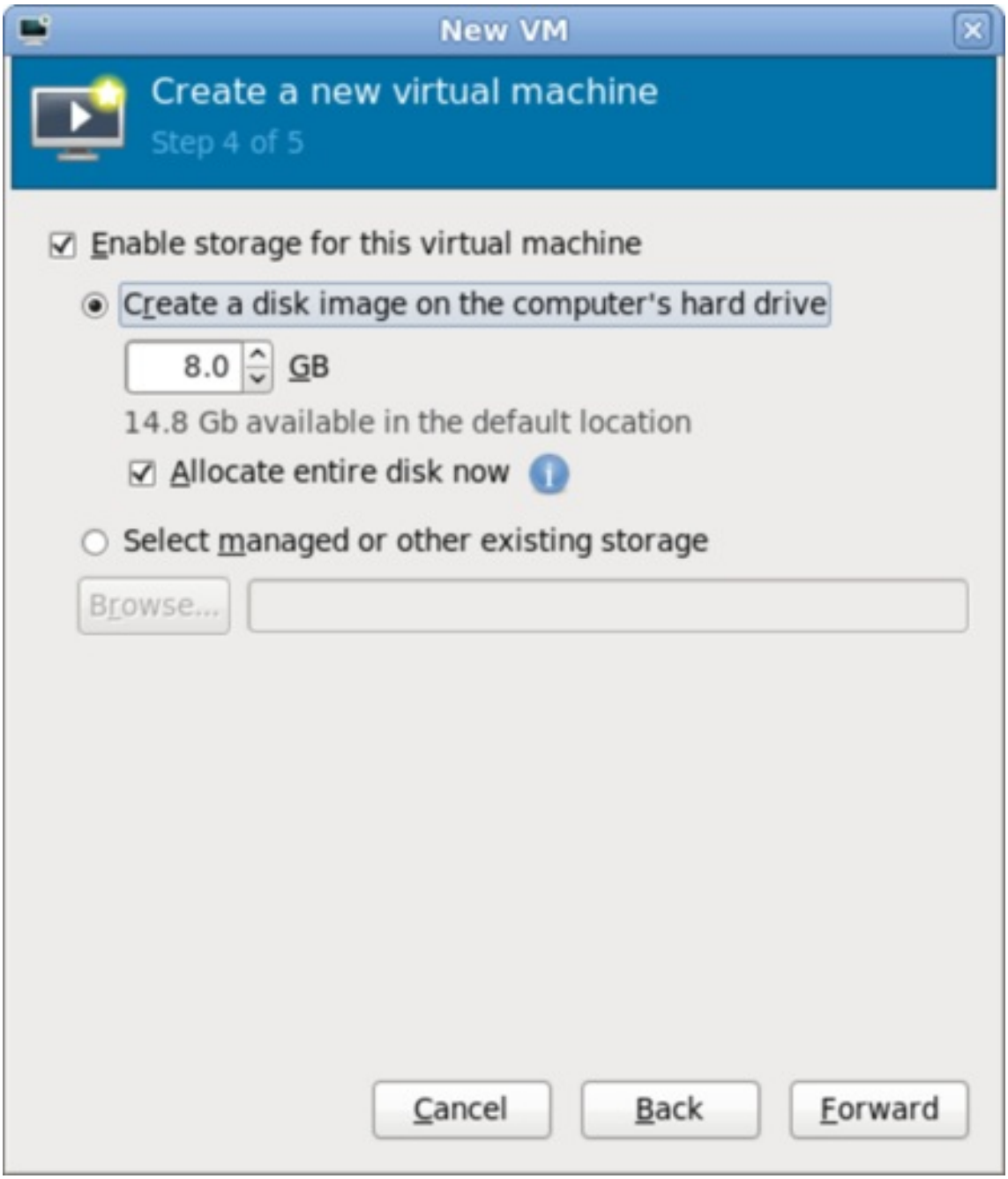


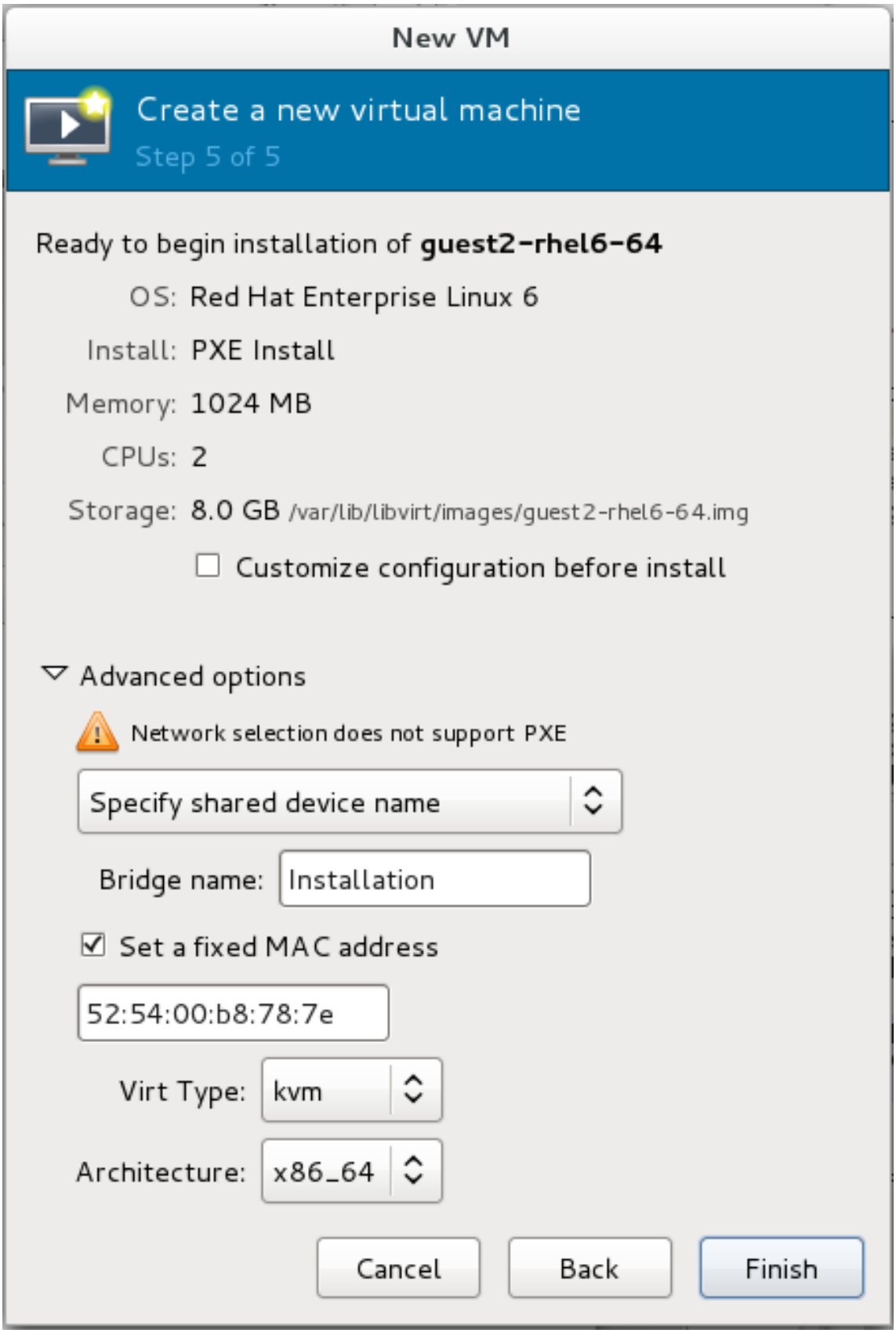
**図6.10 仮想化ハードウェアの詳細の指定**

**図6.11 ストレージの詳細の指定**




2. **インストールの開始**
   インストールを開始する準備が整いました。

**図6.12 仮想マシンの詳細の最終処理**




DHCP 要求が送信され、有効な PXE サーバーが見つかると、ゲスト仮想マシンのインストールプロセスが起動します。

インストールが完了したら、ゲストオペレーティングシステムに接続できます。詳細は、を参照してください。 「仮想マシンへの接続」

## 6.5. 仮想マシンへの接続

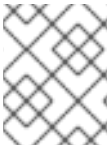
仮想マシンを作成したら、起動 したゲスト OS に接続できます。これを実行するには、以下を使用できます。

- **virt-viewer** または **remote-viewer** - 詳細は、 ゲスト仮想マシン管理用のグラフィカルユーザーインターフェイスツール を参照してください。
- **virt-manager** - 詳細は、 Managing guests with the Virtual Machine Manager を参照してください。
- ゲストのシリアルコンソール - 詳細は、ゲスト仮想マシン のシリアルコンソールの接続 を参照してください。

# 第7章 RED HAT ENTERPRISE LINUX 6 ホストへの RED HAT ENTERPRISE LINUX 6 ゲスト仮想マシンのインストール

本章では、Red Hat Enterprise Linux 6 ホストに Red Hat Enterprise Linux 6 ゲスト仮想マシン をインストールする方法を説明します。

この手順では、KVM ハイパーバイザーおよびその他の必要なパッケージがインストールされ、ホストが仮想化用に設定されていることを前提としています。



**注記**

仮想化パッケージのインストールに関する詳細は、5章 *仮想化パッケージのインストール* を参照してください。

## 7.1. ローカルインストールメディアを使用した RED HAT ENTERPRISE LINUX 6 ゲストの作成

この手順では、ローカルに保存されたインストール DVD または DVD イメージで Red Hat Enterprise Linux 6 ゲスト仮想マシンを作成する方法を説明します。DVD イメージは、Red Hat Enterprise Linux 6 では から http://access.redhat.com 利用できます。

**手順7.1 virt-manager を使用した Red Hat Enterprise Linux 6 ゲスト仮想マシンの作成**

1. **オプション：準備**
   仮想マシンのストレージ環境を準備します。ストレージの準備の詳細は、『Red Hat Enterprise Linux 6 Virtualization Administration Guide』 を参照してください。

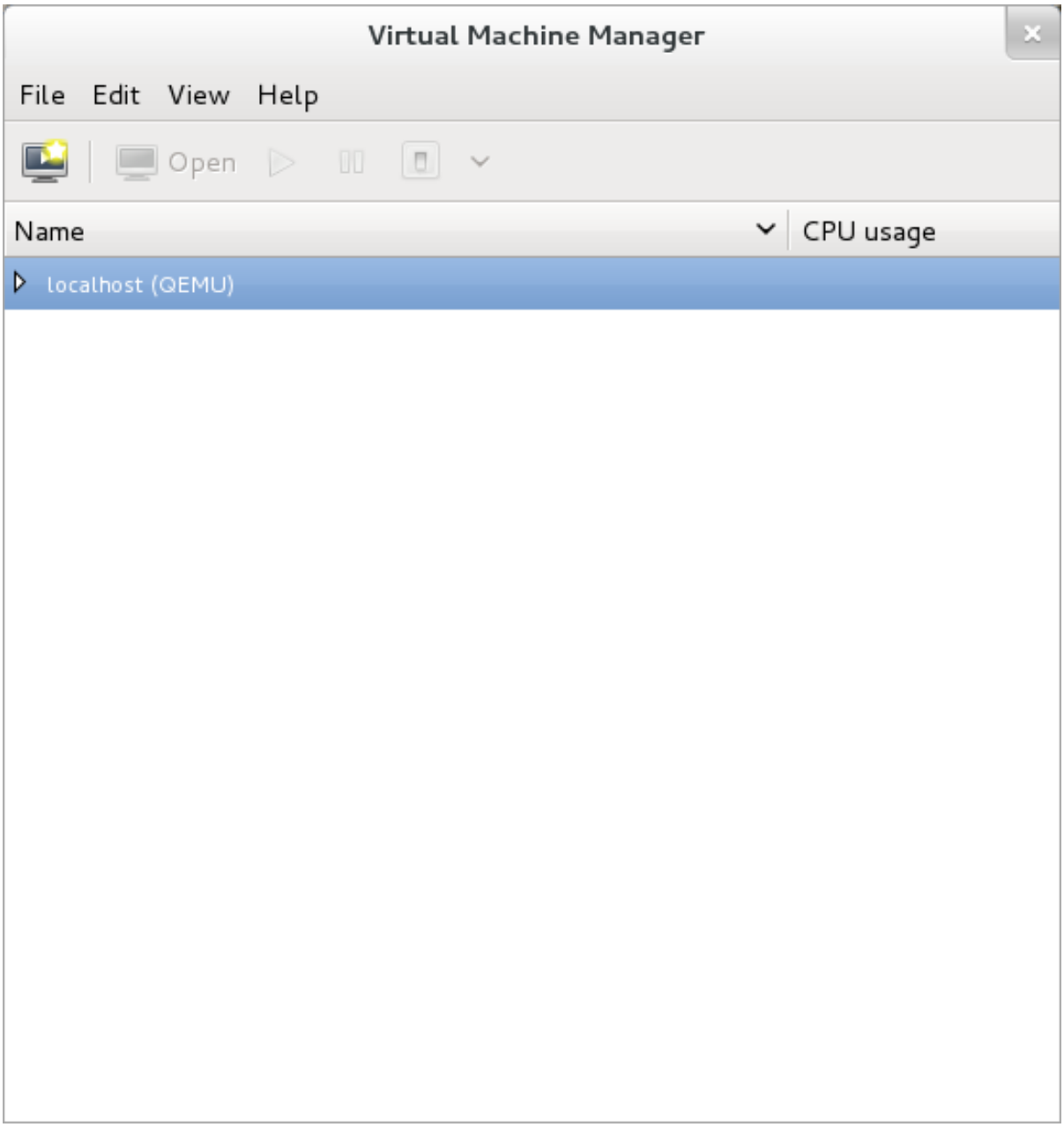
   **重要**

   ゲスト仮想マシンの保存には、さまざまなストレージタイプを使用できます。ただし、仮想マシンが移行機能を使用できるようにするには、ネットワークストレージに仮想マシンを作成する必要があります。

   Red Hat Enterprise Linux 6 には、1GB 以上のストレージ領域が必要です。ただし、Red Hat Enterprise Linux 6 のインストールと本ガイドの手順には、Red Hat では少なくとも 5GB のストレージ領域を推奨します。

2. **virt-manager を開き、ウィザードを起動します。**
   root で **virt-manager** コマンドを実行するか、**Applications → System Tools → Virtual Machine Manager** を開き、virt-manager を開きます。

**図7.1 Virtual Machine Manager ウィンドウ**




**Create a new virtual machine** ボタンをクリックして、新しい仮想化ゲストウィザードを開始します。

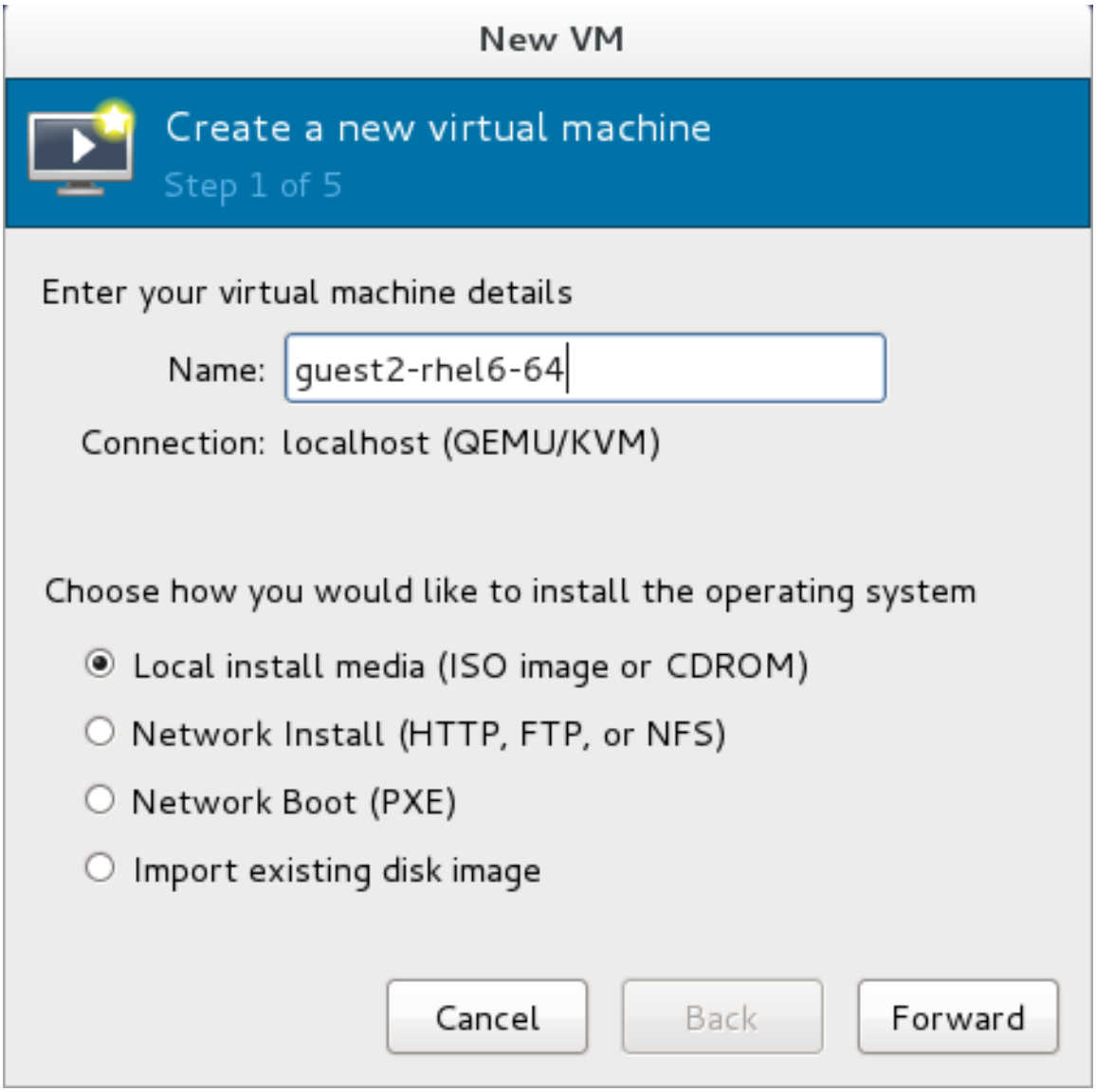
**図7.2 Create a new virtual machine ボタン**

**New VM** ウィンドウが開きます。

3. **仮想マシンに名前を付けます。**
仮想マシン名には、文字、数字、および文字('_'、'.'、'-'、および '-')を含めることができます。仮想マシンを移行するには、仮想マシンの名前は一意でなければならず、数字のみの名前は使用できません。

**Local install media (ISO image or CDROM)** ラジオボタンを選択します。

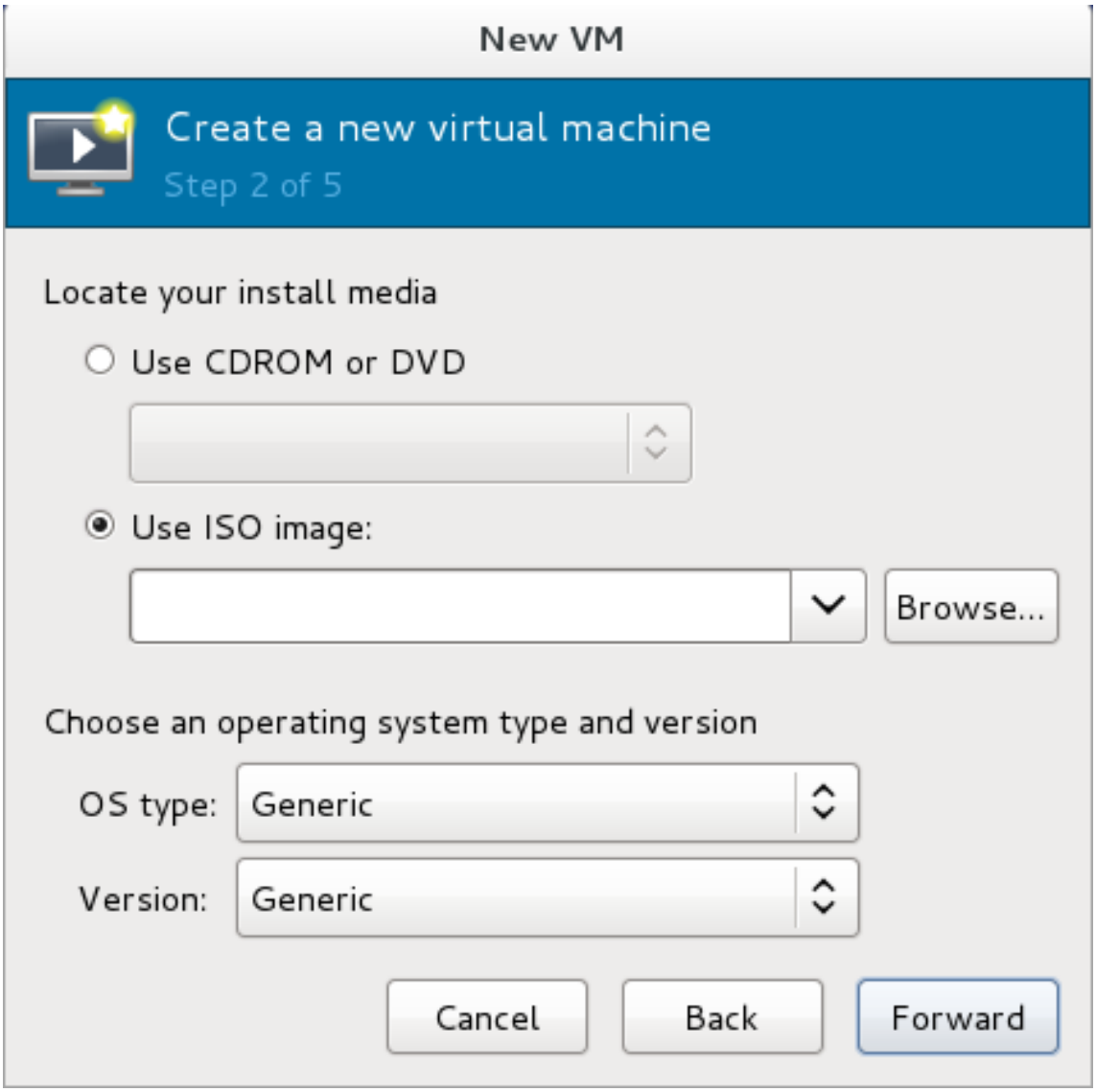
**図7.3 New VM ウィンドウ - ステップ 1**




**進む** をクリックして続けます。

4. **インストールメディアの選択**
   インストールメディアに適したラジオボタンを選択します。

**図7.4 インストールメディアの特定**




- CD-ROM または DVD からインストールする場合は、**Use CDROM or DVD** ラジオボタンを選択し、利用可能なドライブのドロップダウンリストから適切なディスクドライブを選択します。
- ISO イメージからインストールする場合は、**Use ISO image** を選択し、**Browse...** ボタンをクリックして **Locate media volume** ウィンドウを開きます。

  使用するインストールイメージを選択し、**Choose Volume** をクリックします。

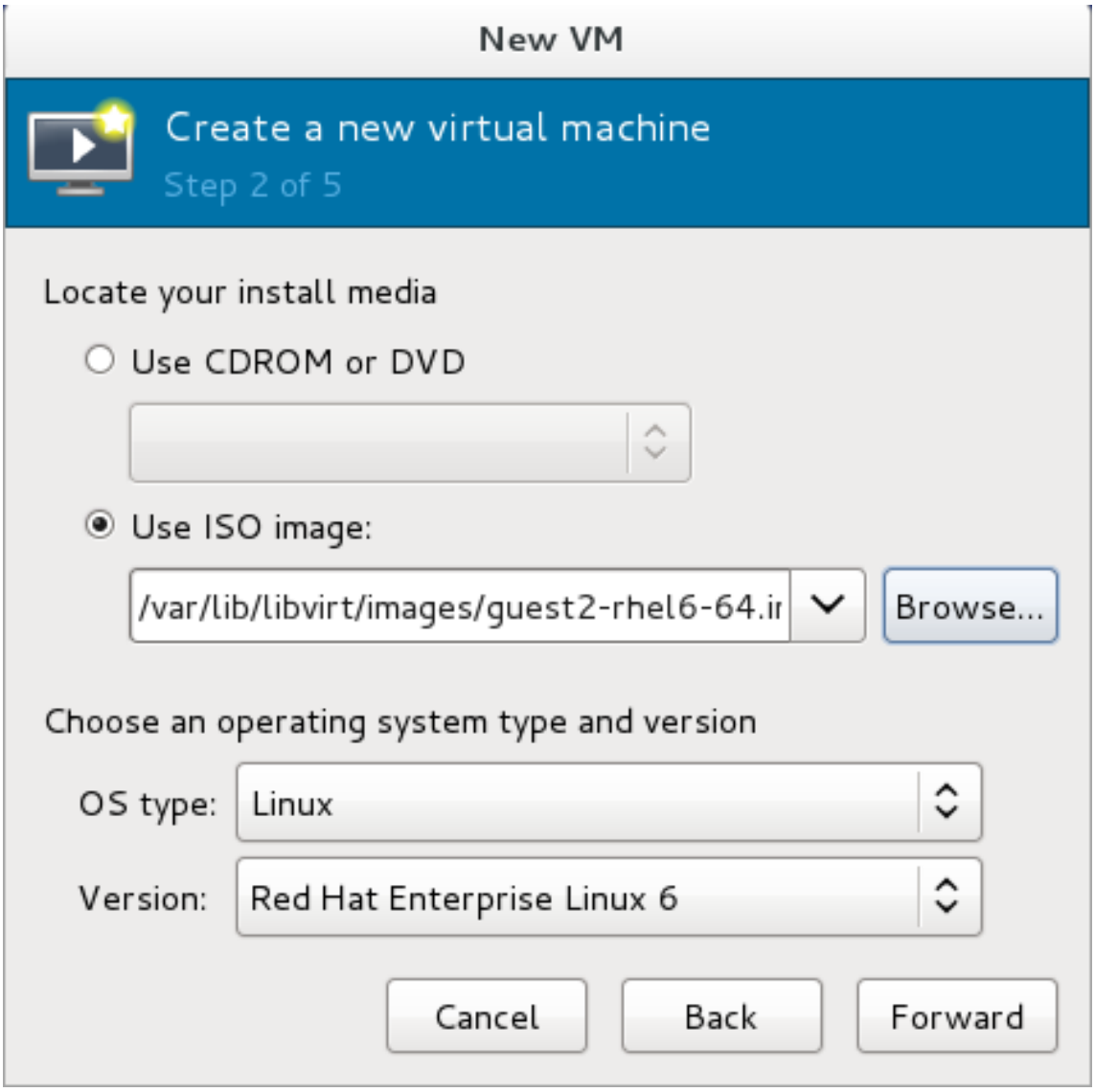
  **Locate media volume** ウィンドウにイメージが表示されない場合は、**Browse Local** ボタンをクリックしてインストールイメージ用のホストマシンまたはインストールディスクを含む DVD ドライブを参照します。インストールディスクが含まれるインストールイメージまたは DVD ドライブを選択し、**Open** をクリックします。使用するボリュームが選択され、**Create a new virtual machine** ウィザードに戻ります。

**重要**

ISO イメージファイルおよびゲストストレージイメージの場合、推奨される使用場所は **/var/lib/libvirt/images/** です。それ以外の場所では、SELinux による追加の設定が必要になる場合があります。SELinux の設定に関する詳細は、Red Hat Enterprise Linux 6 仮想 『化管理ガイド』 を参照してください。

選択したインストールメディアに一致するオペレーティングシステムの種類とバージョンを選択します。

**図7.5 New VM ウィンドウ - ステップ 2**




**進む** をクリックして続けます。

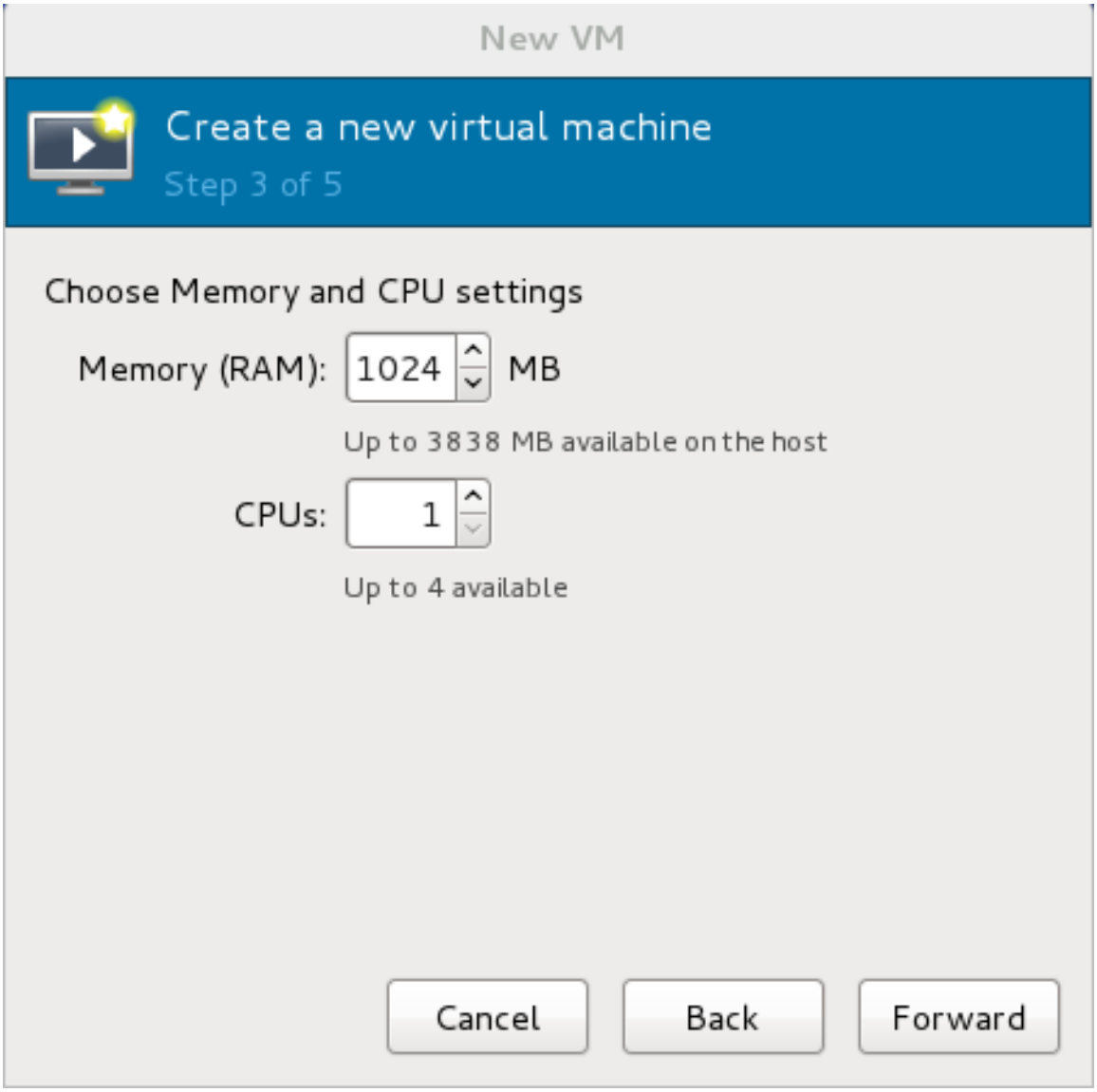
5. **RAM および仮想 CPU の設定**
仮想 CPU および RAM の割り当てに適切な値を選択します。これらの値は、ホストおよびゲストのパフォーマンスに影響します。メモリーおよび仮想 CPU はオーバーコミットできます。オーバーコミットの詳細は、『Red Hat Enterprise Linux 6 仮想化管理ガイド』を参照してください。

仮想マシンを効率的かつ効果的に実行するには、十分な物理メモリー (RAM) が必要です。Red Hat は、仮想マシンに 512MB 以上の RAM をサポートします。Red Hat では、各論理コアに 1024MB 以上の RAM を使用することを推奨しています。

仮想マシンに十分な仮想 CPU を割り当てます。仮想マシンがマルチスレッドアプリケーションを実行している場合は、ゲスト仮想マシンが効率的に実行するために必要な仮想 CPU の数を割り当てます。

ホストシステムで利用可能な物理プロセッサー (またはハイパースレッド) よりも多くの仮想 CPU を割り当てることはできません。利用可能な仮想 CPU の数は、**Up to *X* available** フィールドに記載されています。

**図7.6 新しい VM ウィンドウ - ステップ 3**




**進む** をクリックして続けます。

6. **ストレージを有効にして割り当てる**
Red Hat Enterprise Linux 6 ゲスト仮想マシンにストレージを有効にして割り当てます。デスクトップインストールの場合は 5GB 以上、最小インストールの場合は 1GB 以上を割り当てます。

**注記**

ライブおよびオフラインの移行では、仮想マシンを共有ネットワークストレージにインストールする必要があります。仮想マシンの共有ストレージを設定する方法は、『Red Hat Enterprise Linux Virtualization Administration Guide 』 を参照してください。

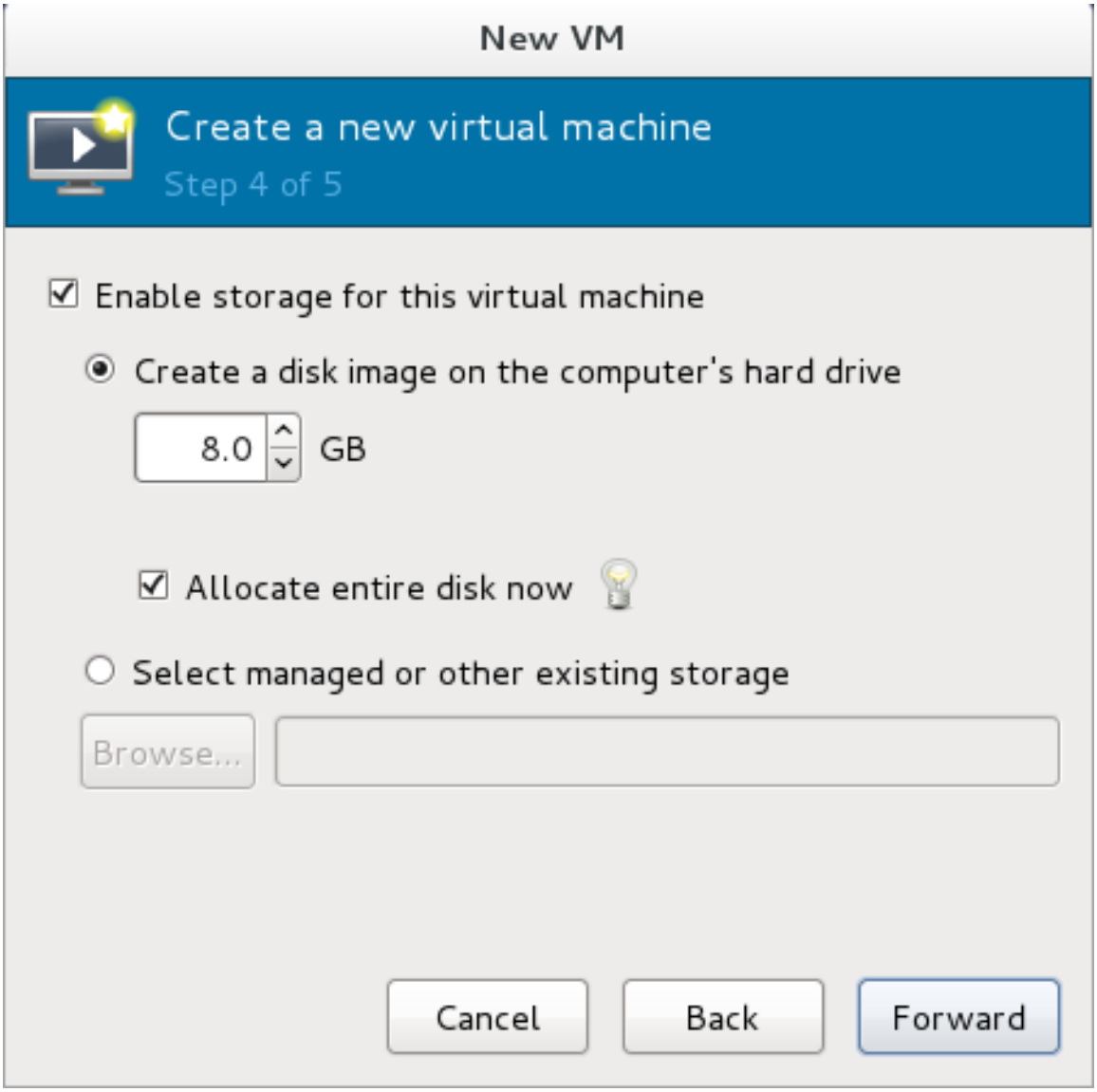
a. **デフォルトのローカルストレージの場合**
**Create a disk image on the computer's hard drive** のラジオボタンを選択して、デフォルトストレージプールの **/var/lib/libvirt/images/** ディレクトリーにファイルベースのイメージを作成します。作成するディスクイメージのサイズを入力します。**Allocate entire disk now** チェックボックスを選択すると、指定したサイズのディスクイメージが即座に作成されます。そうでない場合、ディスクイメージはいっぱいになると大きくなります。

**注記**

ストレージプールは仮想コンテナーですが、次の 2 つの要因で制限されます。つまり、qemu-kvm が許可する最大サイズと、ホストの物理マシンのディスクのサイズです。ストレージプールは、ホストの物理マシン上のディスクのサイズを超えてはなりません。最大サイズは、以下のとおりです。

- virtio-blk = 2^63 バイトまたは 8 エクサバイト (raw ファイルまたはディスクを使用)
- Ext4 = ~ 16TB (4KB のブロックサイズを使用)
- XFS = ~8 エクサバイト
- qcow2 およびホストのファイルシステムでは、非常に大きなイメージサイズを試行する際に、独自のメタデータとスケーラビリティーを評価/調整する必要があります。raw ディスクを使用すると、スケーラビリティーまたは最大サイズに影響を与えるレイヤーが減ります。

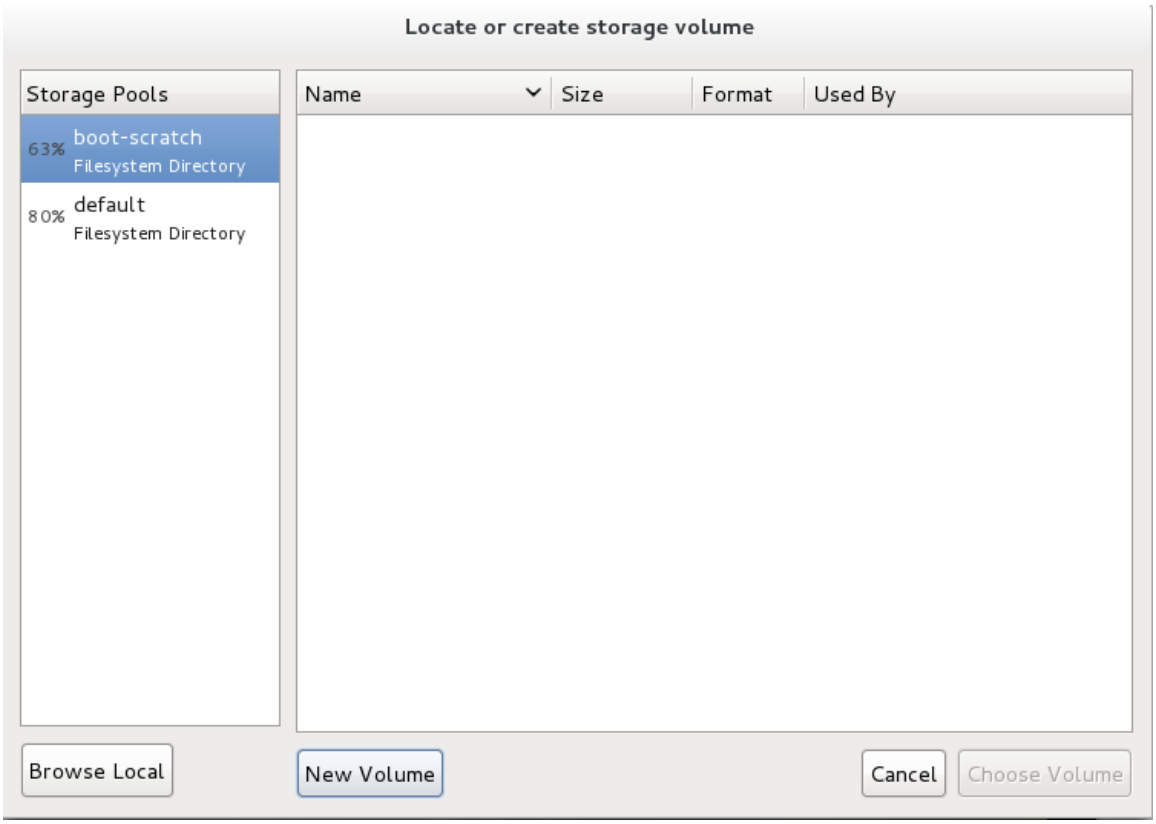
**図7.7 New VM ウィンドウ - ステップ 4**




**Forward** を選択して、ローカルハードドライブにディスクイメージを作成します。または、**Select managed or other existing storage** を選択し、**Browse** を選択して管理ストレージを設定します。

b. **ストレージプールを使用する場合**
前の手順で **Select managed or other existing storage** を選択してストレージプールを使用し、**Browse** をクリックすると、**Locate or create storage volume** ウインドウが表示されます。

**図7.8 Locate or create storage volume ウィンドウ**

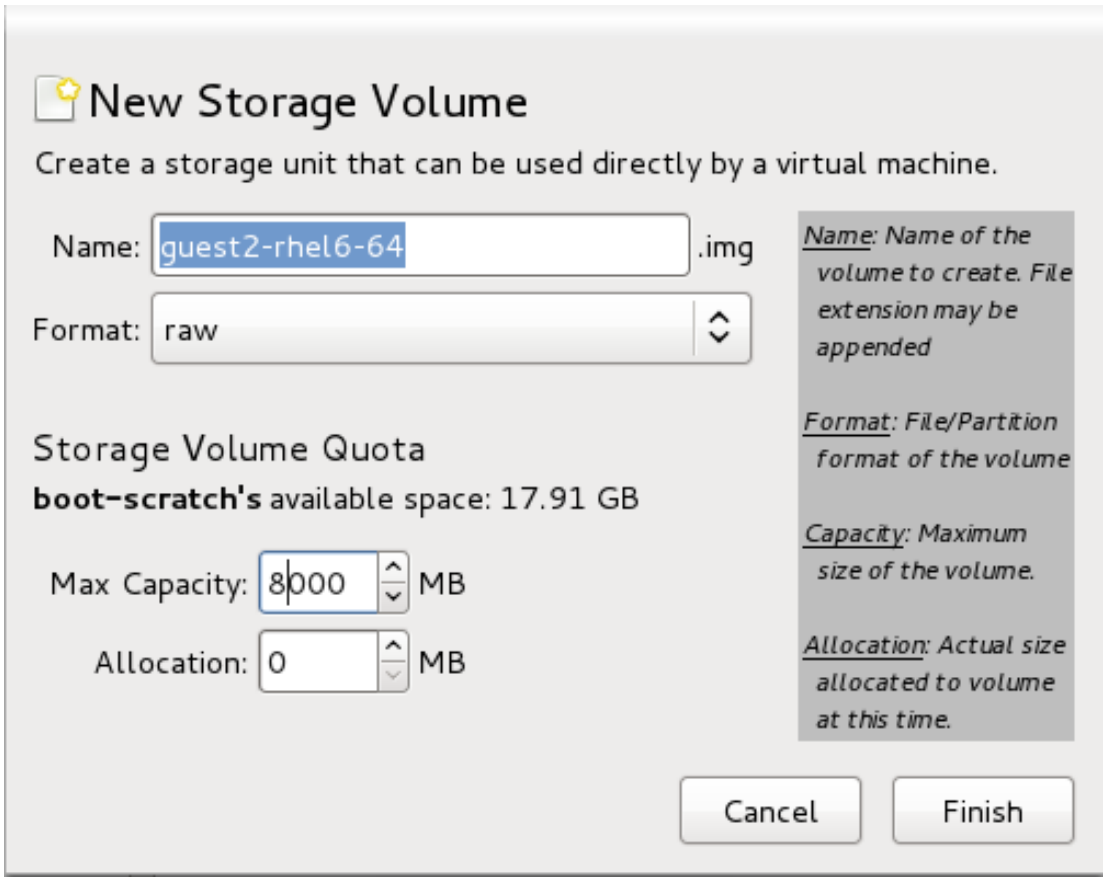



i. **Storage Pools** 一覧からストレージプールを選択します。

ii. 必要に応じて、**New Volume** ボタンをクリックして、新しいストレージボリュームを作成します。**Add a Storage Volume** のスクリーンが表示されます。新しいストレージボリュームの名前を入力します。

**Format** ドロップダウンメニューから形式オプションを選択します。形式のオプションには、raw、cow、qcow、qcow2、qed、vmdk、および vpc が含まれます。必要に応じて他のフィールドを調整します。

図7.9 Add a Storage Volume ウィンドウ



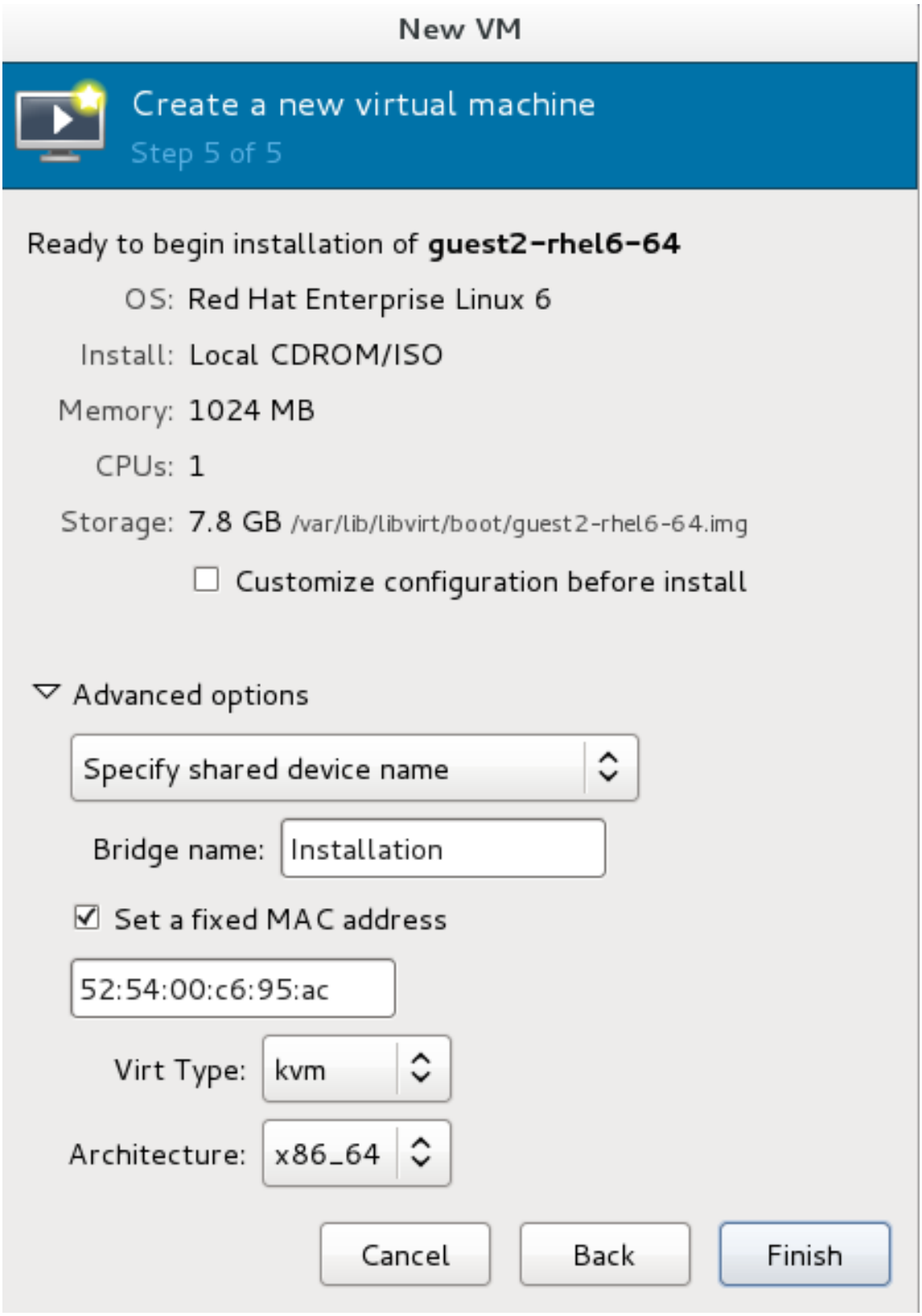

**Finish** をクリックして続行します。

7. **確認および終了**
ウィザード中にエラーが発生しておらず、すべてが想定通りに表示されることを確認します。

**Customize configuration before install** チェックボックスを選択してゲストのストレージまたはネットワークデバイスを変更するか、準仮想化ドライバーを使用するか、デバイスを追加します。

**Advanced options** 下矢印をクリックして、高度なオプションを調べて変更します。Red Hat Enterprise Linux 6 の標準的なインストールでは、これらのオプションを変更する必要はありません。

**図7.10 New VM ウィンドウ - ローカルストレージ**




**Finish** をクリックして、Red Hat Enterprise Linux インストールシーケンスに進みます。Red Hat Enterprise Linux 6 のインストールの詳細は、『Red Hat Enterprise Linux 6 インストールガイド』 を参照してください。

これで、ISO インストールディスクイメージから Red Hat Enterprise Linux 6 ゲスト仮想マシンが作成されます。インストールが完了したら、ゲストオペレーティングシステムに接続できます。詳細は、を参照してください。 「仮想マシンへの接続」

## 7.2. ネットワークインストールツリーを使用した RED HAT ENTERPRISE LINUX 6 ゲストの作成

**手順7.2 virt-manager を使用した Red Hat Enterprise Linux 6 ゲストの作成**

1. **オプション：準備**
   ゲスト仮想マシンのストレージ環境を準備します。ストレージの準備の詳細は、『Red Hat Enterprise Linux 6 Virtualization Administration Guide』 を参照してください。

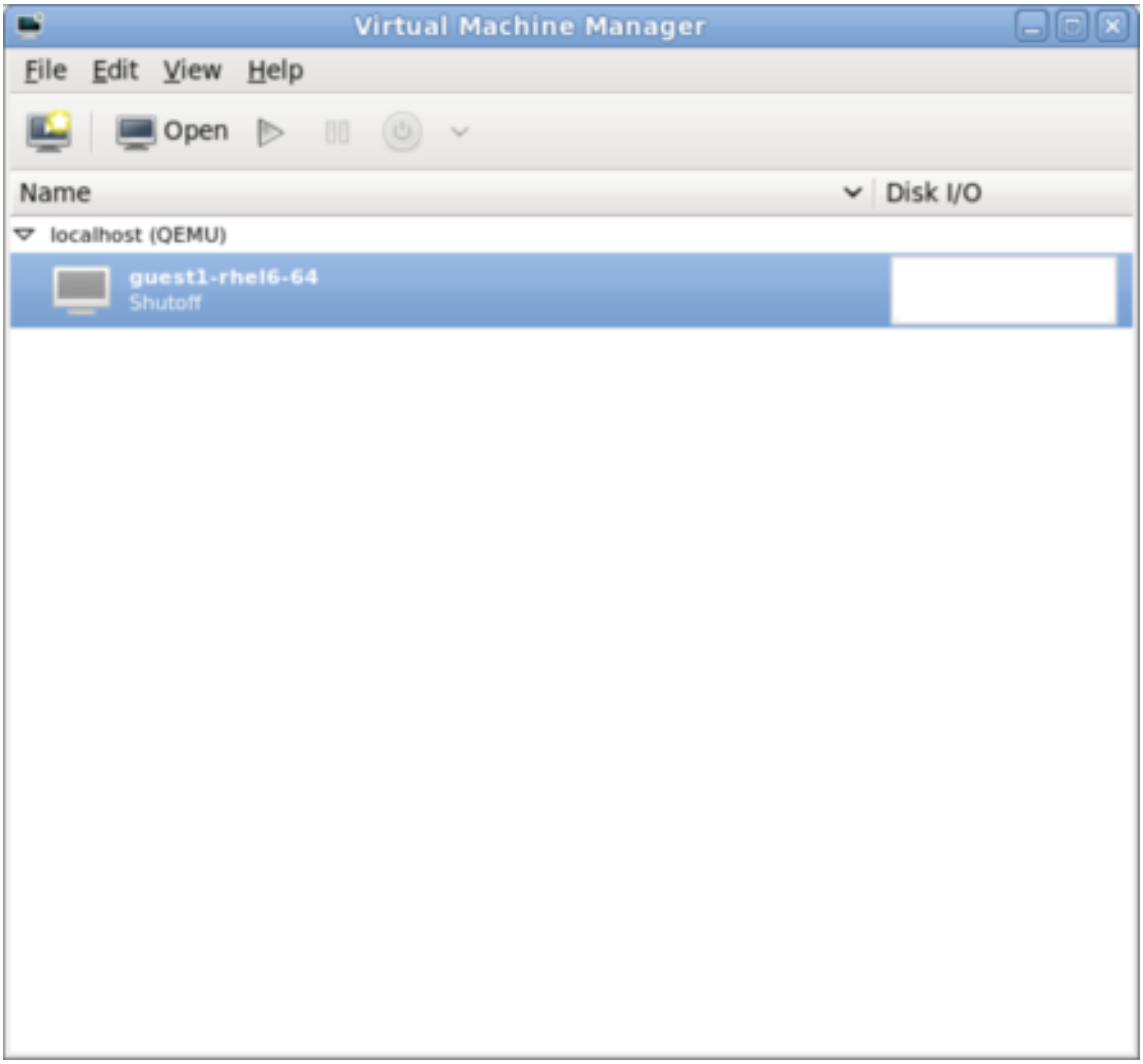
   **重要**

   ゲスト仮想マシンの保存には、さまざまなストレージタイプを使用できます。ただし、仮想マシンが移行機能を使用できるようにするには、ネットワークストレージに仮想マシンを作成する必要があります。

   Red Hat Enterprise Linux 6 には、1GB 以上のストレージ領域が必要です。ただし、Red Hat Enterprise Linux 6 のインストールと本ガイドの手順には、Red Hat では少なくとも 5GB のストレージ領域を推奨します。

2. **virt-manager を開き、ウィザードを起動します。**
   root で **virt-manager** コマンドを実行するか、**Applications → System Tools → Virtual Machine Manager** を開き、virt-manager を開きます。

   **図7.11 メインの virt-manager ウィンドウ**

**Create a new virtual machine** ボタンをクリックして、新しい仮想マシンウィザードを起動します。

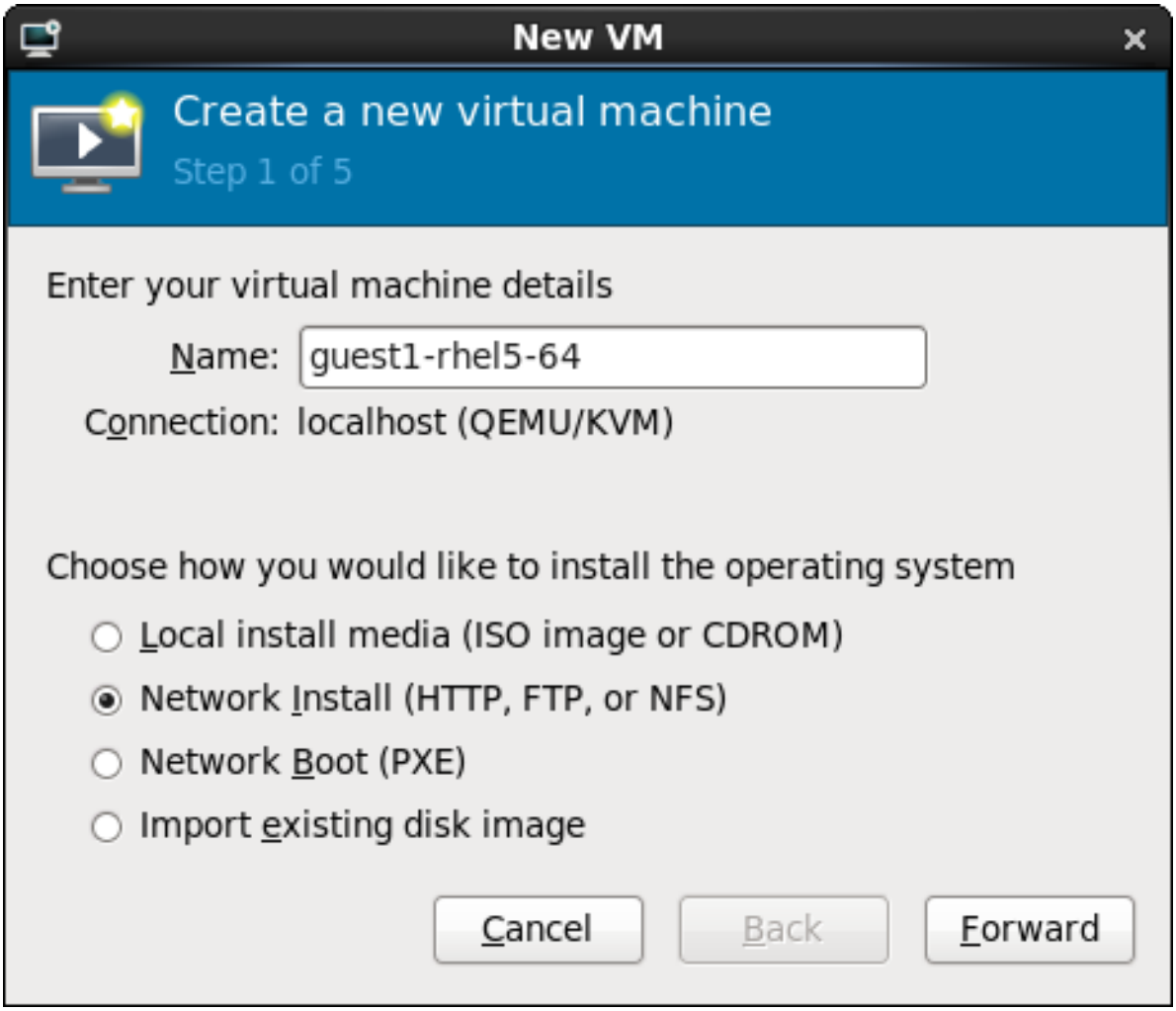
**図7.12 Create a new virtual machine ボタン**

**Create a new virtual machine** ウィンドウが開きます。

3. **仮想マシンに名前を付けます。**
仮想マシン名には、文字、数字、および文字('_'、'.'、'-'、および '-')を含めることができます。仮想マシンを移行するには、仮想マシンの名前は一意でなければならず、数字のみの名前は使用できません。

ラジオボタンの一覧からインストール方法を選択します。

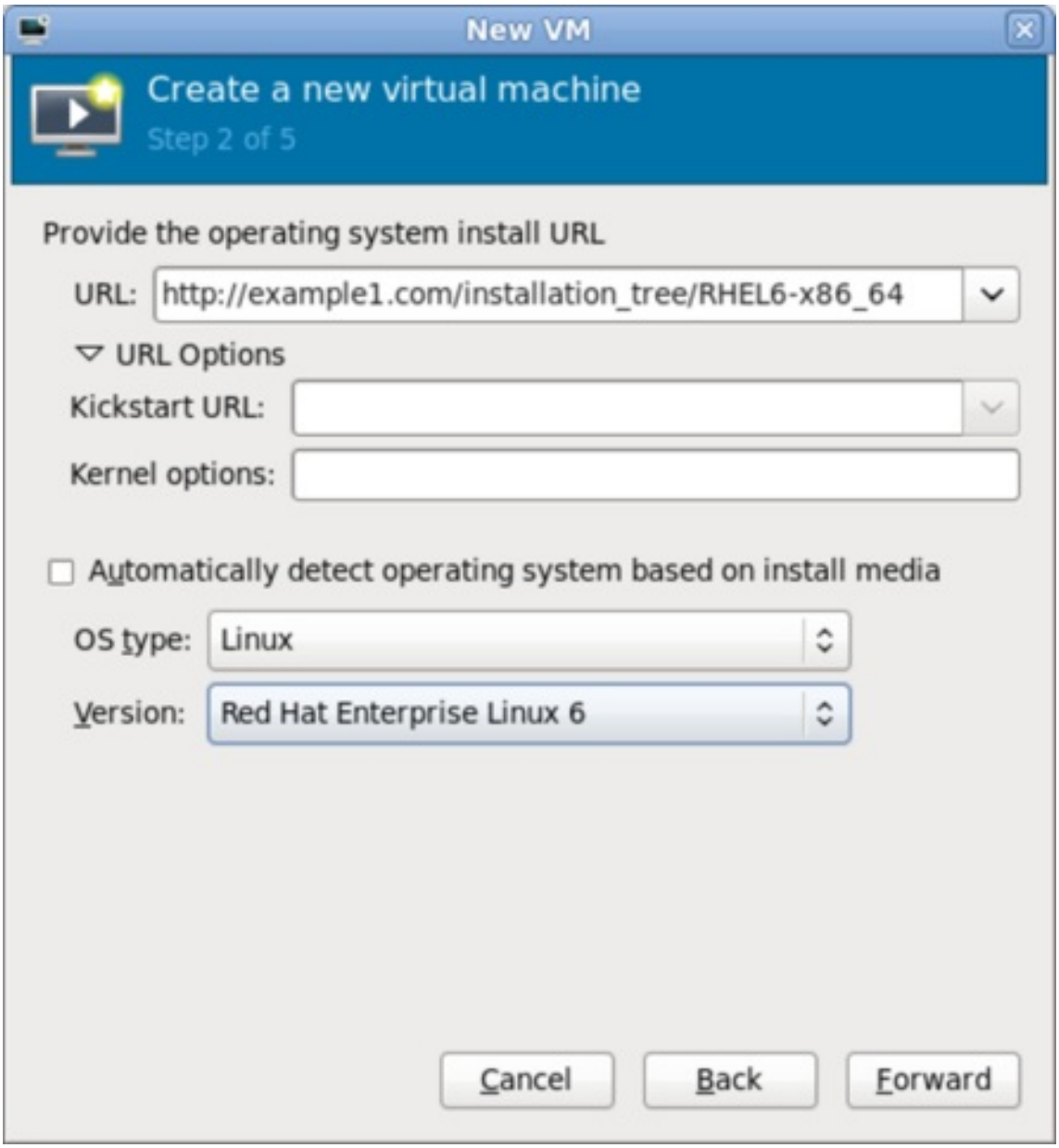
**図7.13 New VM ウィンドウ - ステップ 1**




**進む** をクリックして続けます。

4. 必要に応じて、インストール URL とキックスタート URL とカーネルオプションを指定します。

**図7.14 New VM ウィンドウ - ステップ 2**



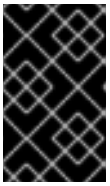

**進む** をクリックして続けます。

5. 残りの手順は ISO インストール手順と同じです。ISO インストール手順の ステップ 5 に進みます。

## 7.3. PXE を使用した RED HAT ENTERPRISE LINUX 6 ゲストの作成

**手順7.3 virt-manager を使用した Red Hat Enterprise Linux 6 ゲストの作成**

1. **オプション：準備**
仮想マシンのストレージ環境を準備します。ストレージの準備の詳細は、『Red Hat Enterprise Linux 6 Virtualization Administration Guide』 を参照してください。



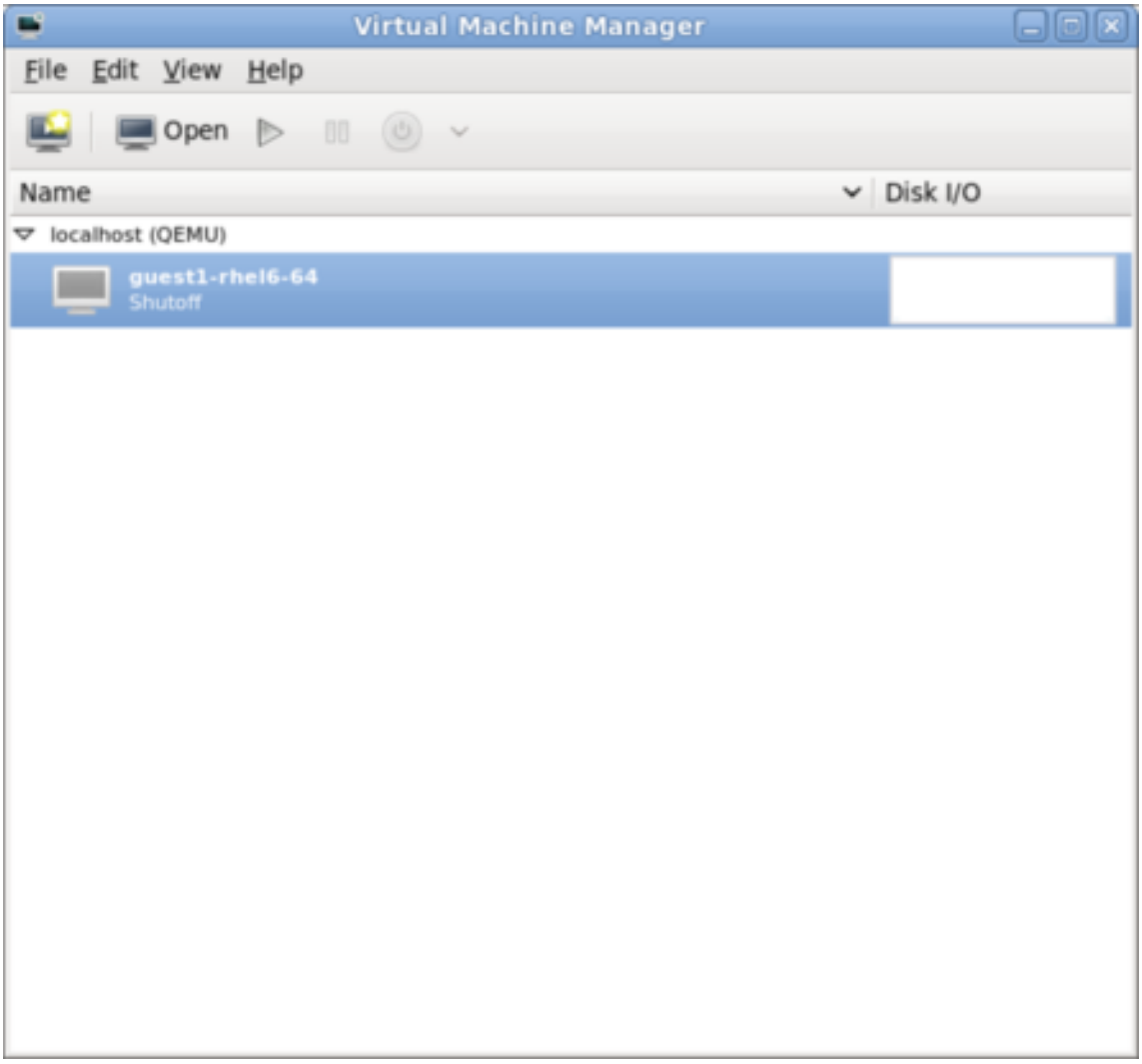
**重要**

ゲスト仮想マシンの保存には、さまざまなストレージタイプを使用できます。ただし、仮想マシンが移行機能を使用できるようにするには、ネットワークストレージに仮想マシンを作成する必要があります。

Red Hat Enterprise Linux 6 には、1GB 以上のストレージ領域が必要です。ただし、Red Hat Enterprise Linux 6 のインストールと本ガイドの手順には、Red Hat では少なくとも 5GB のストレージ領域を推奨します。

2. **virt-manager を開き、ウィザードを起動します。**
root で **virt-manager** コマンドを実行するか、**Applications → System Tools → Virtual Machine Manager** を開き、virt-manager を開きます。

**図7.15 メインの virt-manager ウィンドウ**




**Create new virtualization guest** ボタンをクリックして、新しい仮想化ゲストウィザードを開始します。

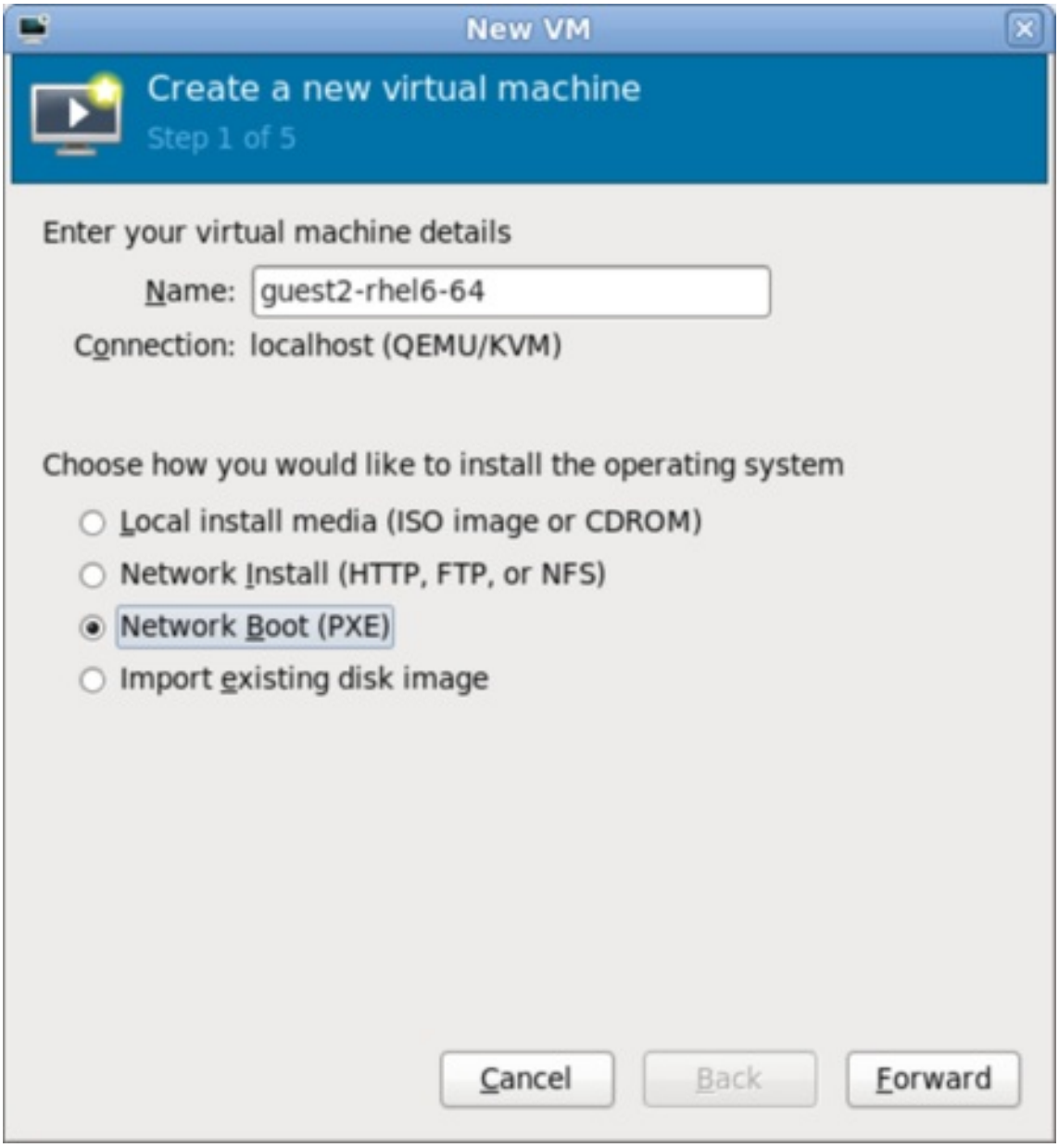
**図7.16 Create new virtualization guest ボタン**

**New VM** ウィンドウが開きます。

3. **仮想マシンに名前を付けます。**
仮想マシン名には、文字、数字、および文字('_'、'.'、'-'、および '-')を含めることができます。仮想マシンを移行するには、仮想マシンの名前は一意でなければならず、数字のみの名前は使用できません。

ラジオボタンの一覧からインストール方法を選択します。

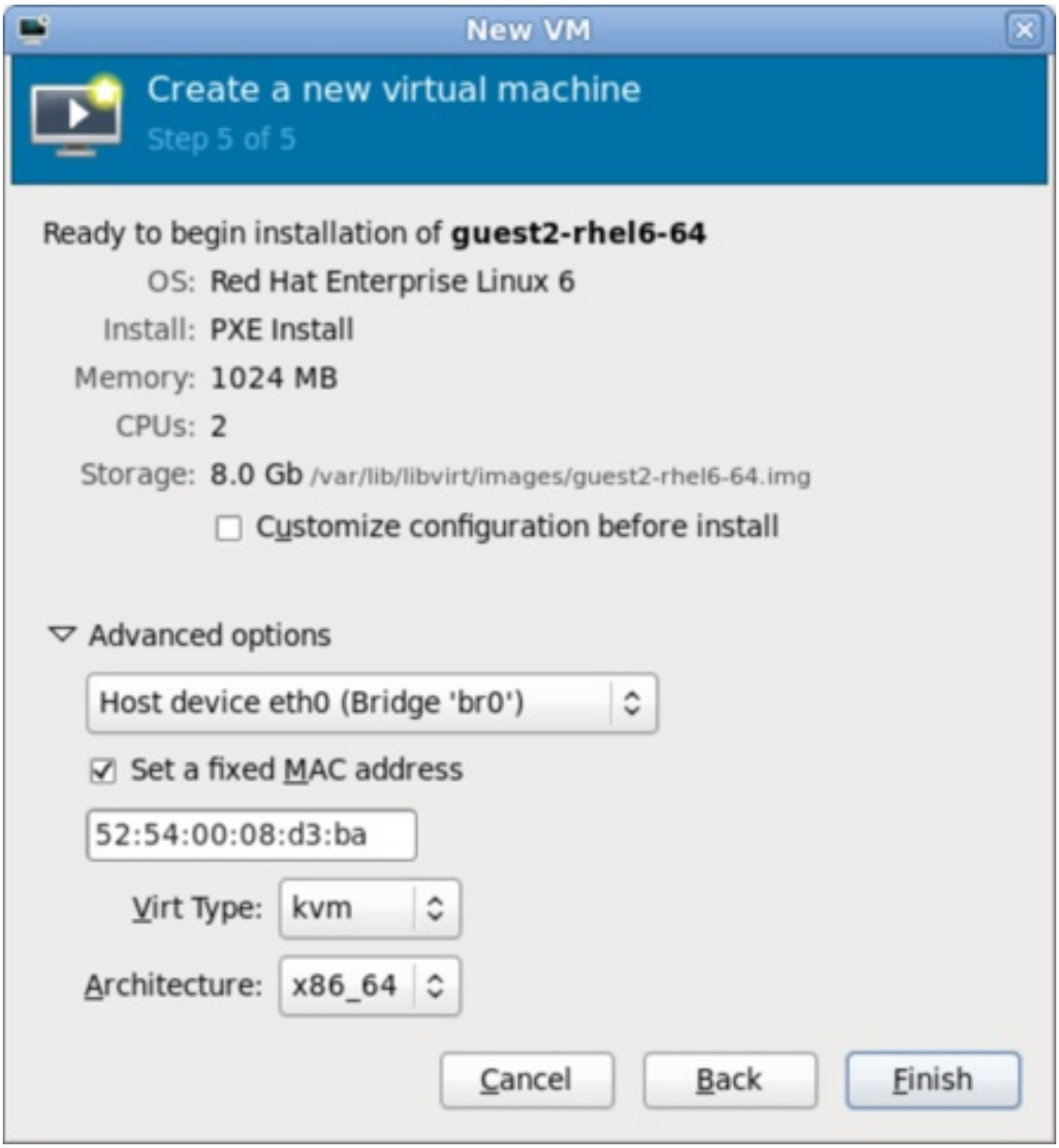
**図7.17 New VM ウィンドウ - ステップ 1**




**進む** をクリックして続けます。

4. 残りの手順は ISO インストール手順と同じです。ISO インストール手順の ステップ 5 に進みます。この時点から、この PXE この手順の唯一の違いは、**Install: PXE Install** フィールドを示す最後の **New VM** 画面にあります。

図7.18 New VM ウィンドウ - ステップ 5 - PXE インストール



# 第8章 他のプラットフォームでの RED HAT ENTERPRISE LINUX の仮想化

本章では、他の仮想化ホストで Red Hat Enterprise Linux 6 を仮想化オペレーティングシステムとして実行しているお客様の役立つ参考資料を紹介します。

## 8.1. VMWARE ESX の場合

Red Hat Enterprise Linux 6.0 以降は、VMware ホストで Red Hat Enterprise Linux を実行する際に使用される準仮想化メモリーバルーンドライバーである **vmw_balloon** ドライバーが提供されます。このドライバーの詳細は、を参照して http://kb.VMware.com/selfservice/microsites/search.do?cmd=displayKC&docType=kc&externalId=1002586 ください。

Red Hat Enterprise Linux 6.3 以降では、VMware ホストで Red Hat Enterprise Linux を実行する際に使用される準仮想化マウスドライバーである **vmmouse_drv** ドライバーが提供されます。このドライバーの詳細は、を参照して http://kb.VMware.com/selfservice/microsites/search.do?cmd=displayKC&docType=kc&externalId=5739104 ください。

Red Hat Enterprise Linux 6.3 以降では、VMware ホストで Red Hat Enterprise Linux を実行する際に使用される準仮想化ビデオドライバー **vmware_drv** ドライバーが提供されます。このドライバーの詳細は、を参照して http://kb.VMware.com/selfservice/microsites/search.do?cmd=displayKC&docType=kc&externalId=1033557 ください。

Red Hat Enterprise Linux 6.3 以降は、VMware ホストで Red Hat Enterprise Linux を実行する際に使用される準仮想化ネットワークアダプターである **vmxnet3** ドライバーを提供します。このドライバーの詳細は、を参照して http://kb.VMware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1001805 ください。

Red Hat Enterprise Linux 6.4 以降は、VMware ホストで Red Hat Enterprise Linux を実行する際に使用される準仮想化 SCSI アダプターである **vmw_pvscsi** ドライバーを提供します。このドライバーの詳細は、を参照して http://kb.VMware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1010398 ください。

## 8.2. HYPER-V の場合

Red Hat Enterprise Linux 6.4 以降には、Microsoft の Linux 統合サービス（サポートされている仮想化オペレーティングシステムで合成デバイスのサポートを有効にするドライバー）が含まれます。提供されるドライバーの詳細は、を参照して http://technet.microsoft.com/en-us/library/dn531030.aspx ください。

**重要**

Hyper-V 用のビルトインの Red Hat Enterprise Linux Integration Service ドライバーでは、Red Hat Enterprise Linux ゲストが Hyper-V ホストの高性能合成デバイスを使用して実行するには十分です。これらの組み込みドライバーは、この使用のために Red Hat によって認定され、認定済みの設定は Red Hat カスタマーポータル で確認できます。

したがって、Linux Integration Services (LIS)パッケージをダウンロードおよびインストールする必要はなく、関連するナレッジベースの記事 で説明されているように、Red Hat サポートが制限される可能性があります。

Red Hat Enterprise Linux ゲスト仮想マシンのデプロイと管理を容易にするために、以下の機能拡張が追加されました。

- VMBUS プロトコルのアップグレード - VMBUS プロトコルが Windows 8 レベルにアップグレードされました。この作業の一環として、ゲストで利用可能な全仮想 CPU で VMBUS 割り込みを処理できるようになりました。さらに、Red Hat Enterprise Linux ゲスト仮想マシンと Windows ホストの物理マシン間のシグナルプロトコルが最適化されています。

- 合成フレームバッファードライバー - Red Hat Enterprise Linux デスクトップユーザー向けのグラフィックパフォーマンスと優れた解決策を提供します。

- ライブ仮想マシンのバックアップサポート: ライブの Red Hat Enterprise Linux ゲスト仮想マシンの中断のないバックアップサポートをプロビジョニングします。

- 固定サイズの Linux VHDX を動的に拡張 - ライブマウントの固定サイズ Red Hat Enterprise Linux VHDX の拡張を可能にします。

- UEFI を使用したブート：Hyper-V 2012 R2 ホストの Unified Extensible Firmware Interface (UEFI)を使用して仮想マシンを起動できるようにします。

詳細および機能については、『Red Hat Enterprise Linux - Virtualization Administration Guide 』 および Enabling Linux Support on Windows Server 2012 R2 Hyper-Vを参照してください。

# 第9章 完全仮想化 WINDOWS ゲストのインストール

本章では、コマンドライン(**virt-install**)を使用して完全仮想化 Windows ゲストを作成し、ゲスト内でオペレーティングシステムのインストーラーを起動し、**virt-viewer** を介してインストーラーにアクセスする方法を説明します。

ゲストに Windows オペレーティングシステムをインストールするには、**virt-viewer** ツールを使用します。このツールを使用すると、(SPICE または VNC プロトコルを使用して)仮想マシンのグラフィカルコンソールを表示できます。これにより、**virt-viewer** では、完全に仮想化されたゲストのオペレーティングシステムを、そのオペレーティングシステムのインストーラー( )でインストールできます。

Windows オペレーティングシステムをインストールするには、以下の 2 つの主要な手順を実行します。

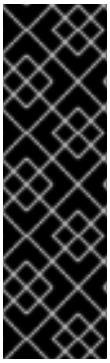
1. **virt-install** または **virt-manager** のいずれかを使用して、ゲスト仮想マシンを作成します。
2. **virt-viewer** を使用して、ゲスト仮想マシンに Windows オペレーティングシステムをインストールします。

**virt-install** または **virt-manager** を使用したゲスト仮想マシンの作成に関する詳細は、*6章 ゲスト仮想マシンのインストールの概要* を参照してください。

本章では、Windows オペレーティングシステムを完全仮想化ゲストにインストールする方法については説明していないことに注意してください。代わりに、ゲストを作成し、ゲスト内でインストーラーを起動する方法のみを説明します。Windows オペレーティングシステムのインストール方法は、関連する Microsoft インストールドキュメントを参照してください。

## 9.1. VIRT-INSTALL を使用したゲストの作成

**virt-install** コマンドを使用すると、GUI を必要とせずに、ターミナルから完全仮想化ゲストを作成できます。



**重要**

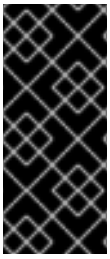
ゲストを作成する前に、ゲストが KVM Windows 準仮想化ドライバーを使用する必要があるかどうかを考慮してください。その場合は、ゲストへの Windows オペレーティングシステムのインストール **中** またはインストール **後** に行うことができることに注意してください。準仮想化ドライバーの詳細は、*10章KVM 準仮想化 (virtio) ドライバー* を参照してください。

KVM 準仮想化ドライバーのインストール方法は、「KVM Windows virtio ドライバーのインストール」 を参照してください。

1 つのコマンドのみを使用して、完全に仮想化されたゲストを作成できます。これを行うには、以下のプログラムを実行します（値を適宜置き換え）。

```
# virt-install \
   --name=guest-name \
   --os-type=windows \
   --network network=default \
   --disk path=path-to-disk,size=disk-size \
   --cdrom=path-to-install-disk \
   --graphics spice --ram=1024
```

***path-to-disk*** は、デバイス（例： **/dev/sda3**）またはイメージファイル(**/var/lib/libvirt/images/*名*.img**) **である必要があります。**また、***disk-size*** をサポートするのに十分な空き容量も必要です。



**重要**

すべてのイメージファイルは、デフォルトで **/var/lib/libvirt/images/** に保存されます。ファイルベースのイメージの他のディレクトリーの場所は可能ですが、SELinux 設定が必要になる場合があります。SELinux を Enforcing モードで実行する場合は、『Red Hat Enterprise Linux 6 仮想化管理ガイド』 で SELinux に関する詳細情報を参照してください。

**virt-install** を対話的に実行することもできます。これを行うには、以下のように **--prompt** コマンドを使用します。

```
# virt-install --prompt
```

完全に仮想化されたゲストが作成されると、**virt-viewer** はゲストを起動し、オペレーティングシステムのインストーラーを実行します。オペレーティングシステムのインストール方法については、関連する Microsoft インストールドキュメントを参照してください。

# 第10章 KVM 準仮想化 (VIRTIO) ドライバー

準仮想化ドライバーはゲストのパフォーマンスを向上し、ゲスト I/O レイテンシーを下げ、ベアメタルレベルまでスループットを増加させます。I/O の高いタスクとアプリケーションを実行する完全に仮想化されたゲストには、準仮想化ドライバーを使用することが推奨されます。

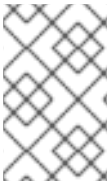
VirtIO ドライバーは、KVM ホストで実行している Windows ゲスト仮想マシンで利用可能な、KVM の準仮想化デバイスドライバーです。これらのドライバーは virtio パッケージに含まれています。virtio パッケージは、ブロック (ストレージ) デバイスおよびネットワークインターフェイスコントローラーに対応しています。

KVM virtio ドライバーが自動的に読み込まれ、インストールされます。

- Red Hat Enterprise Linux 4.8 以降
- Red Hat Enterprise Linux 5.3 以降
- Red Hat Enterprise Linux 6 以降
- Red Hat Enterprise Linux 7 以降
- 2.6.27 以降のカーネルバージョンをベースとする Linux のバージョンの一部。

上記の一覧の Red Hat Enterprise Linux のバージョンがドライバーを検出してインストールする場合、追加のインストール手順は必要ありません。

Red Hat Enterprise Linux 3 (3.9 以降)では、手動インストールが必要です。



**注記**

PCI デバイスは、仮想システムアーキテクチャーにより制限されます。割り当てられたデバイスを使用する場合の追加の制限については、「KVM の制限」 を参照してください。

KVM virtio ドライバーを使用すると、以下の Microsoft Windows バージョンがベアメタルベースのシステムと同様に実行されることが予想されます。

- Windows Server 2003 (32 ビットおよび 64 ビットバージョン)
- Windows Server 2008 (32 ビットおよび 64 ビットバージョン)
- Windows Server 2008 R2 (64 ビットのみ)
- Windows 7 (32 ビットおよび 64 ビットバージョン)
- Windows Server 2012 (64 ビットのみ)
- Windows Server 2012 R2 (64 ビットのみ)
- Windows 8 (32 ビットおよび 64 ビットバージョン)
- Windows 8.1 (32 ビットおよび 64 ビットバージョン)

## 10.1. KVM WINDOWS VIRTIO ドライバーのインストール

本セクションでは、KVM Windows virtio ドライバーのインストールプロセスを説明します。KVM virtio ドライバーは、Windows のインストール時に読み込むか、ゲストをインストールした後にインストールできます。

次のいずれかの方法を使用して、ゲスト仮想マシンに virtio ドライバーをインストールできます。

- 仮想マシンがアクセスできるネットワーク上のインストールファイルをホストする
- ドライバーインストールディスク .iso ファイルの仮想化 CD-ROM デバイスの使用
- USB ドライブを使用して、CD-ROM に使用するものと同じ（提供された）.ISO ファイルをマウントします。
- 仮想化されたフロッピーデバイスを使用した起動時にドライバーのインストール（必須および XP/2003 でのみ推奨されます）

本ガイドでは、準仮想化インストーラーディスクから仮想化 CD-ROM デバイスとしてインストールについて説明します。

1. **ドライバーをダウンロードします。**
   virtio-win パッケージには、対応しているすべての Windows ゲスト仮想マシンの virtio ブロックおよびネットワークドライバーが含まれます。

   **yum** コマンドを使用して、virtio-win パッケージをホストにダウンロードしてインストールします。

   ```
   # yum install virtio-win
   ```

   Windows オペレーティングシステムおよび現在の認定パッケージバージョンでサポートされる virtio-win パッケージの一覧は、windowsservercatalog.com の URL を参照してください。

   Red Hat Virtualization Hypervisor と Red Hat Enterprise Linux が同じコードベース上に作成され、同じバージョンのドライバー（例：Red Hat Virtualization Hypervisor 3.3 および Red Hat Enterprise Linux 6.5）が両方の環境でサポートされることに注意してください。

   virtio-win パッケージは、**/usr/share/virtio-win/** ディレクトリーに CD-ROM イメージ **virtio-win.iso** をインストールします。

2. **virtio ドライバーをインストールします。**
   virtio-win デバイスを使用する Windows ゲストを起動する場合は、関連する virtio-win デバイスドライバーがこのゲストにすでにインストールされている必要があります。virtio-win ドライバーは、Microsoft Windows インストールキットの inbox ドライバーとして提供されないため、virtio-win ストレージデバイス(viostor/virtio-scsi)に Windows ゲストをインストールするには、インストール **時** に **virtio-win.iso** から直接、または提供された Virtual Floppy イメージ **virtio-win <*version*> .vfd** から適切なドライバーを提供する必要があります。

## 10.2. インストールされた WINDOWS ゲスト仮想マシンへのドライバーのインストール

この手順では、Windows のインストール後に仮想化 CD-ROM で virtio ドライバーをインストールする方法を説明します。

以下の手順に従って、**virt-manager** を使用して CD-ROM イメージを追加し、ドライバーをインストールします。

**手順10.1 virt-manager を使用したドライバー CD-ROM イメージからのインストール**

1. **virt-manager およびゲスト仮想マシンを開きます。**
   **virt-manager** を開き、ゲスト名をダブルクリックしてリストからゲスト仮想マシンを開きます。

2. **ハードウェアウィンドウを開きます。**
   ウィンドウ上部のツールバーのライトバイザーアイコンをクリックして、仮想ハードウェアの詳細を表示します。

   **図10.1 仮想ハードウェアの詳細ボタン**

   次に、表示される新規ビューの下部にある **Add Hardware** ボタンをクリックします。これにより、新規デバイスを追加するためのウィザードが開きます。

3. **select the device type - for Red Hat Enterprise Linux 6 versions before to 6.2**
   Red Hat Enterprise Linux 6.2 以降を使用している場合は、この手順を省略します。

   バージョン 6.2 よりも前の Red Hat Enterprise Linux 6 バージョンでは、追加するデバイスのタイプを選択する必要があります。この場合は、ドロップダウンメニューから **Storage** を選択します。

   **図10.2 Red Hat Enterprise Linux 6.1 における Add new virtual hardware ウィザード**

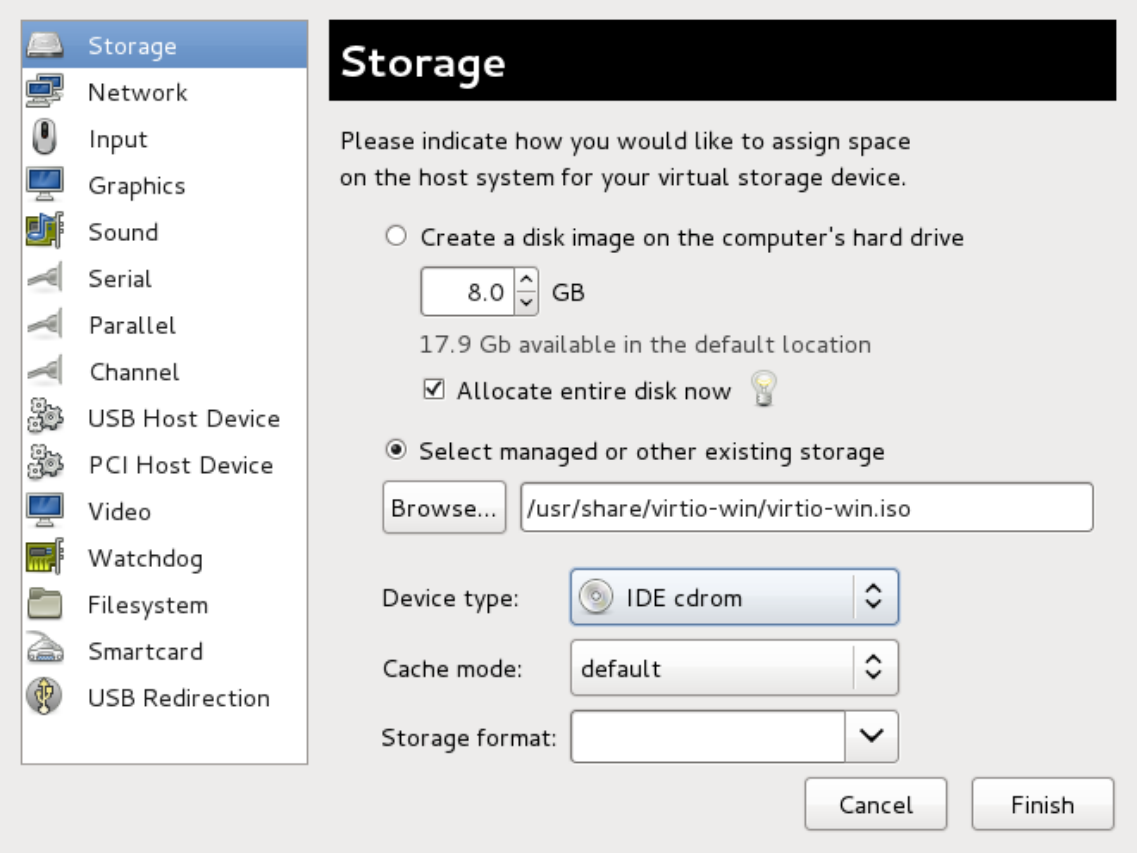
   
   

   **Finish** ボタンをクリックして続行します。

4. **ISO ファイルの選択**
   **Select managed or other existing storage** ラジオボタンが選択されていることを確認し、virtio ドライバーの **.iso** イメージファイルを参照します。ドライバーの最新版のデフォルトの場所は **/usr/share/virtio-win/virtio-win.iso** です。

   **デバイスタイプ** を **IDE cdrom** に変更し、**Forward** ボタンをクリックして続行します。

   **図10.3 Add new virtual hardware ウィザード**




5. **仮想ハードウェアの追加を完了する - 6.2 より前の Red Hat Enterprise Linux 6 バージョン**
   Red Hat Enterprise Linux 6.2 以降を使用している場合は、この手順を省略します。

   バージョン 6.2 よりも前の Red Hat Enterprise Linux 6 バージョンでは、**Finish** ボタンをクリックして仮想ハードウェアの追加を終了し、ウィザードを閉じます。

図10.4 Red Hat Enterprise Linux 6.1 における Add new virtual hardware ウィザード


6. **再起動**
   仮想マシンを再起動または起動して、ドライバーディスクの使用を開始します。仮想化 IDE デバイスは、仮想マシンが新しいデバイスを認識するために再起動する必要があります。

ドライバーを含む CD-ROM がアタッチされ、仮想マシンが起動したら、手順10.2「Windows 7 仮想マシンへの Windows インストール」 に進みます。

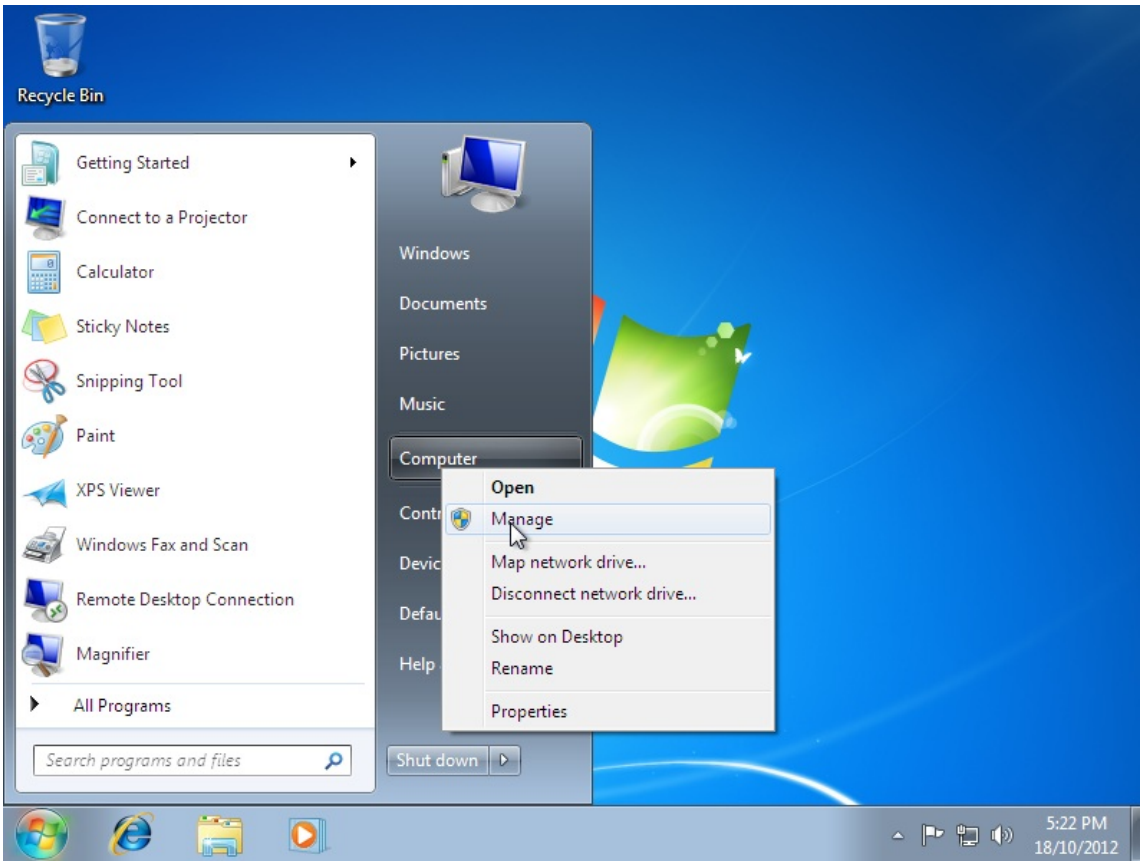
**手順10.2 Windows 7 仮想マシンへの Windows インストール**

この手順では、例として Windows 7 仮想マシンにドライバーをインストールします。Windows のインストール手順をゲストの Windows バージョンに適合させます。

1. **Computer Management ウィンドウを開きます。**
   Windows 仮想マシンのデスクトップで、画面下部の **Windows** アイコンをクリックして、Start メニューを開きます。

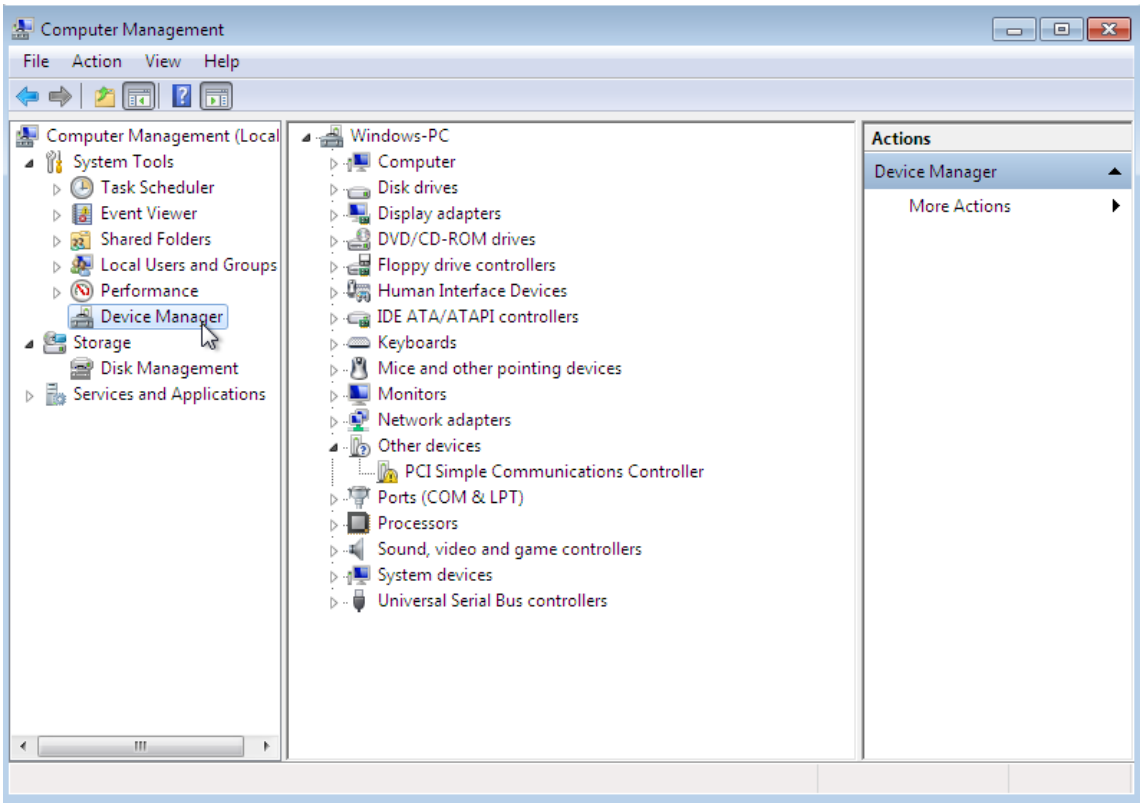
   **Computer** を右クリックし、ポップアップメニューから **Manage** を選択します。

図10.5 Computer Management ウィンドウ




2. **デバイスマネージャーを開きます。**
左側のペインから **Device Manager** を選択します。これは、**Computer Management** > **System Tools** にあります。

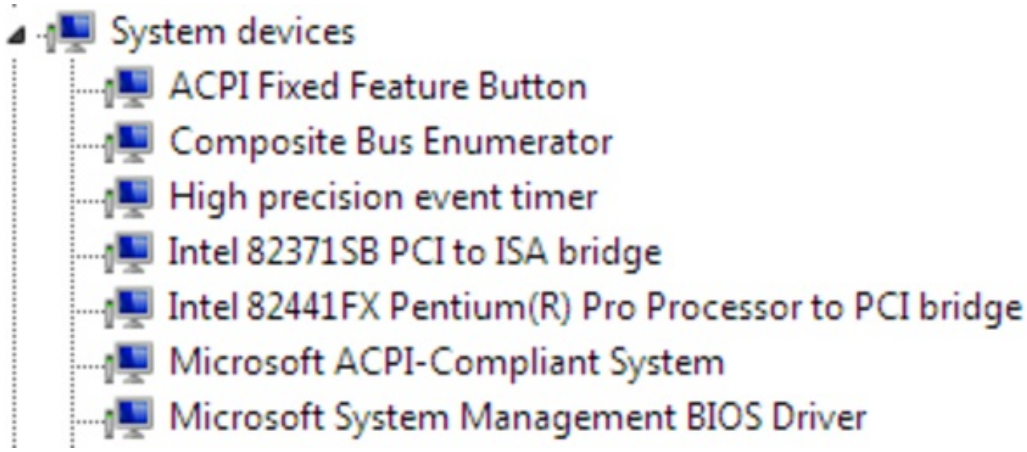
図10.6 Computer Management ウィンドウ




3. **ドライバー更新ウィザードの開始**

a. **利用可能なシステムデバイスの表示**
左側の矢印をクリックして、**システムデバイス** を展開します。

**図10.7 Computer Management ウィンドウで利用可能なシステムデバイスの表示**




b. **適切なデバイスを見つけます。**
バルーンドライバー、シリアルドライバー、ネットワークドライバー、ブロックドライバーの４つのドライバーが利用可能です。

- バルーンドライバーのバルーンは、**System devices** グループの **PCI 標準 RAM Controller** に影響します。

- **vioserial** （シリアルドライバー）は、**System devices** グループの **PCI Simple Communication Controller** に影響します。

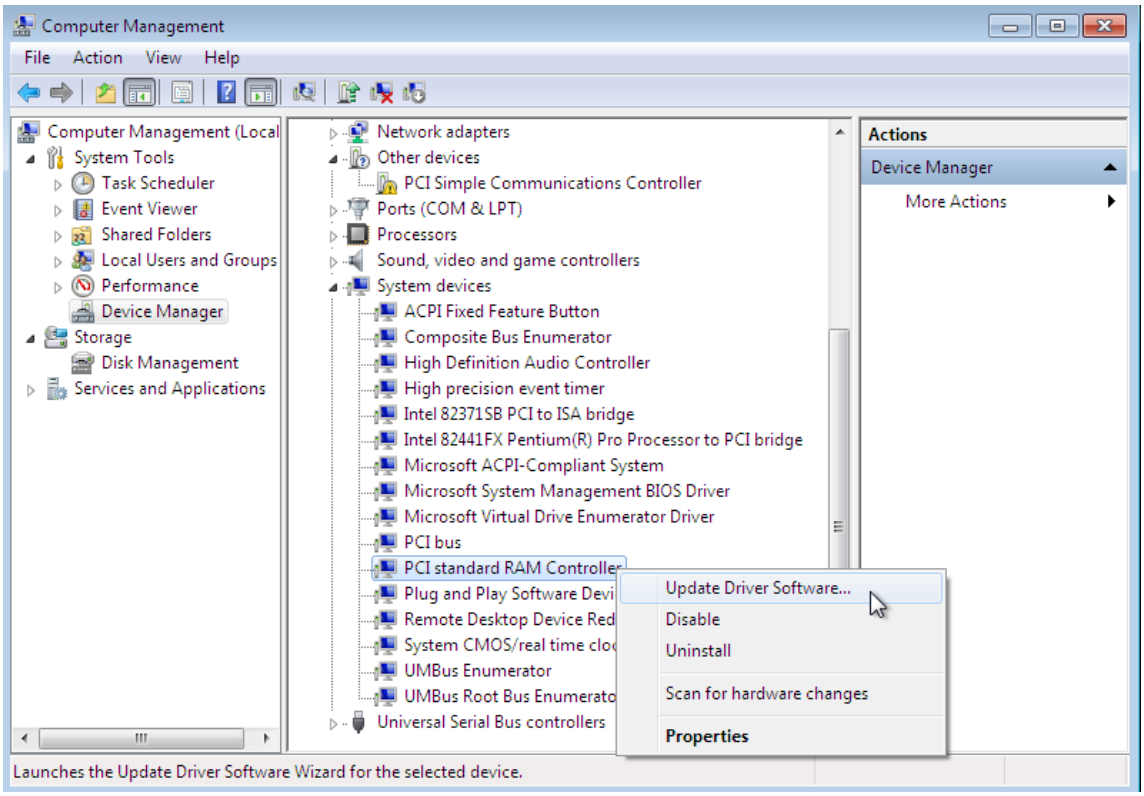
- **NetKVM** （ネットワークドライバー）は、**Network アダプター** グループに影響します。このドライバーは、**virtio NIC** が設定されている場合にのみ利用できます。このドライバーの設定可能なパラメーターは、付録A *NetKVM ドライバーパラメーター* に記載されています。

- **viostor** （ブロックドライバー）は **ディスクドライブ** グループに影響します。このドライバーは、**virtio** ディスクが設定されている場合にのみ利用できます。

**更新するドライバーを右クリックし、ポップアップメニューから Update Driver... を選択します。**

**この例では、バルーンドライバーをインストールするため、PCI 標準 RAM コントローラー を右クリックします。**

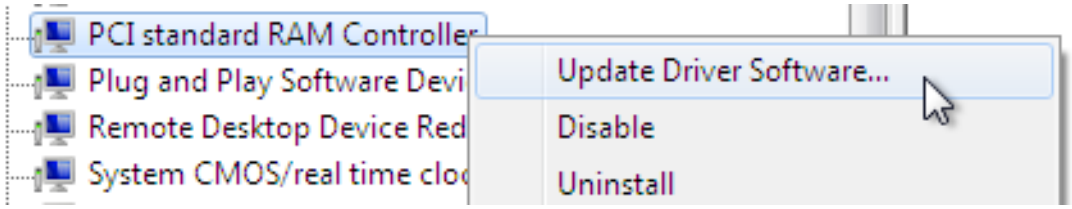
**図10.8 Computer Management ウィンドウ**




c. **ドライバー更新ウィザードを開く**

**ドロップダウンメニューから Update Driver Software... を選択し、ドライバー更新ウィザードにアクセスします。**

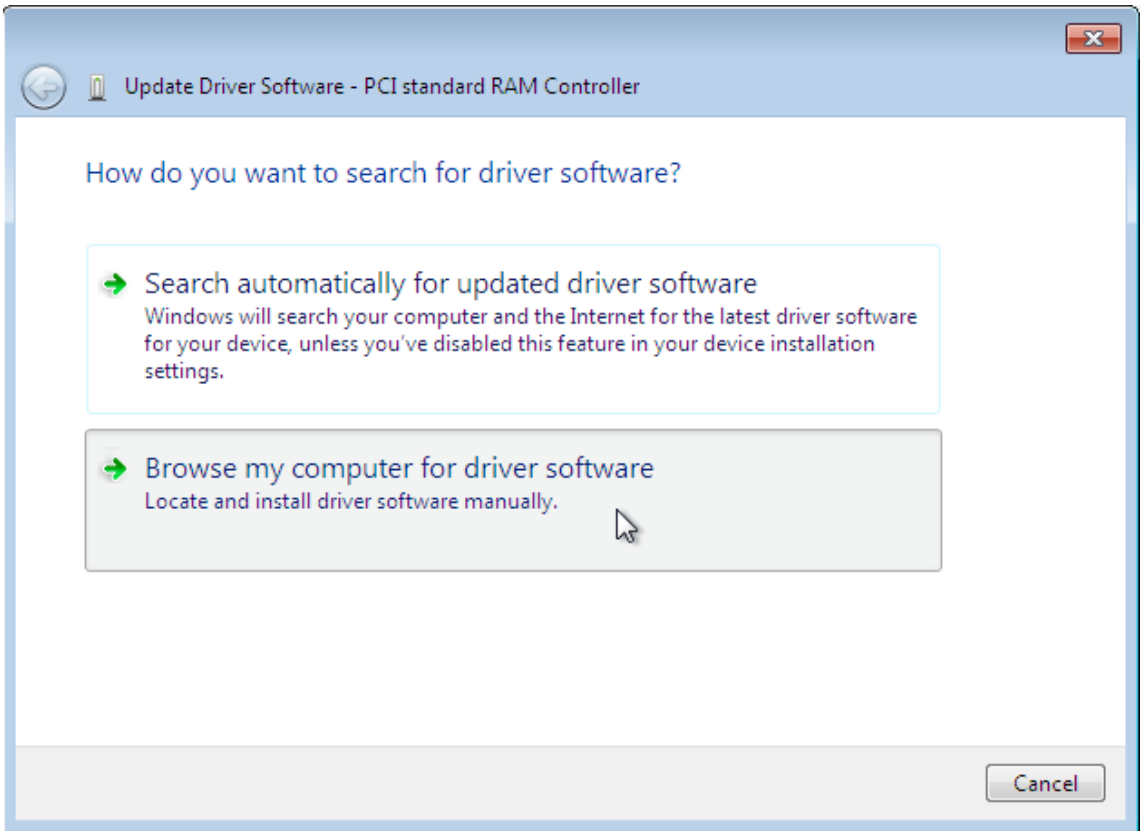
**図10.9 ドライバー更新ウィザードを開く**




4. **ドライバーの検索方法を指定します。**

**ドライバー更新ウィザードの最初のページでは、ドライバーソフトウェアを検索する方法を求めます。2 番目のオプション( Browse my computer for driver software )をクリックします。**

**図10.10 ドライバー更新ウィザード**

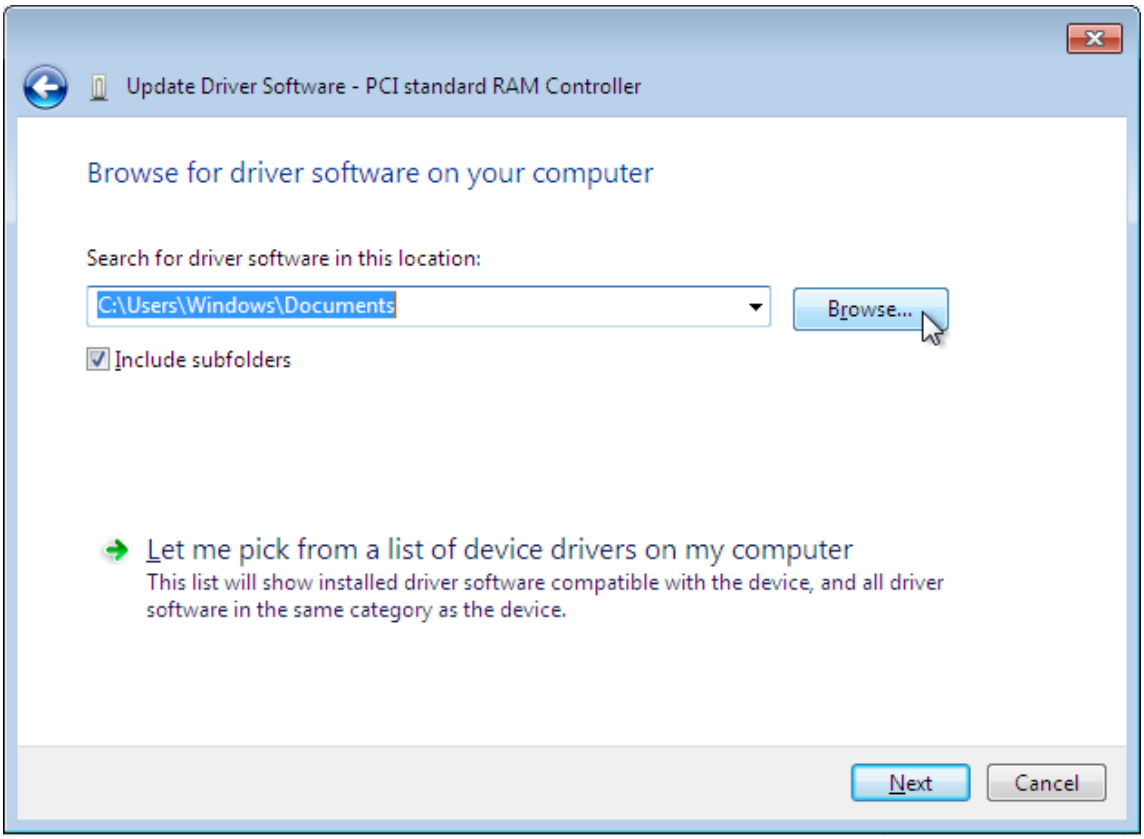



5. **インストールするドライバーを選択します。**

   a. **ファイルブラウザーを開きます。**

      **Browse...をクリックします。**

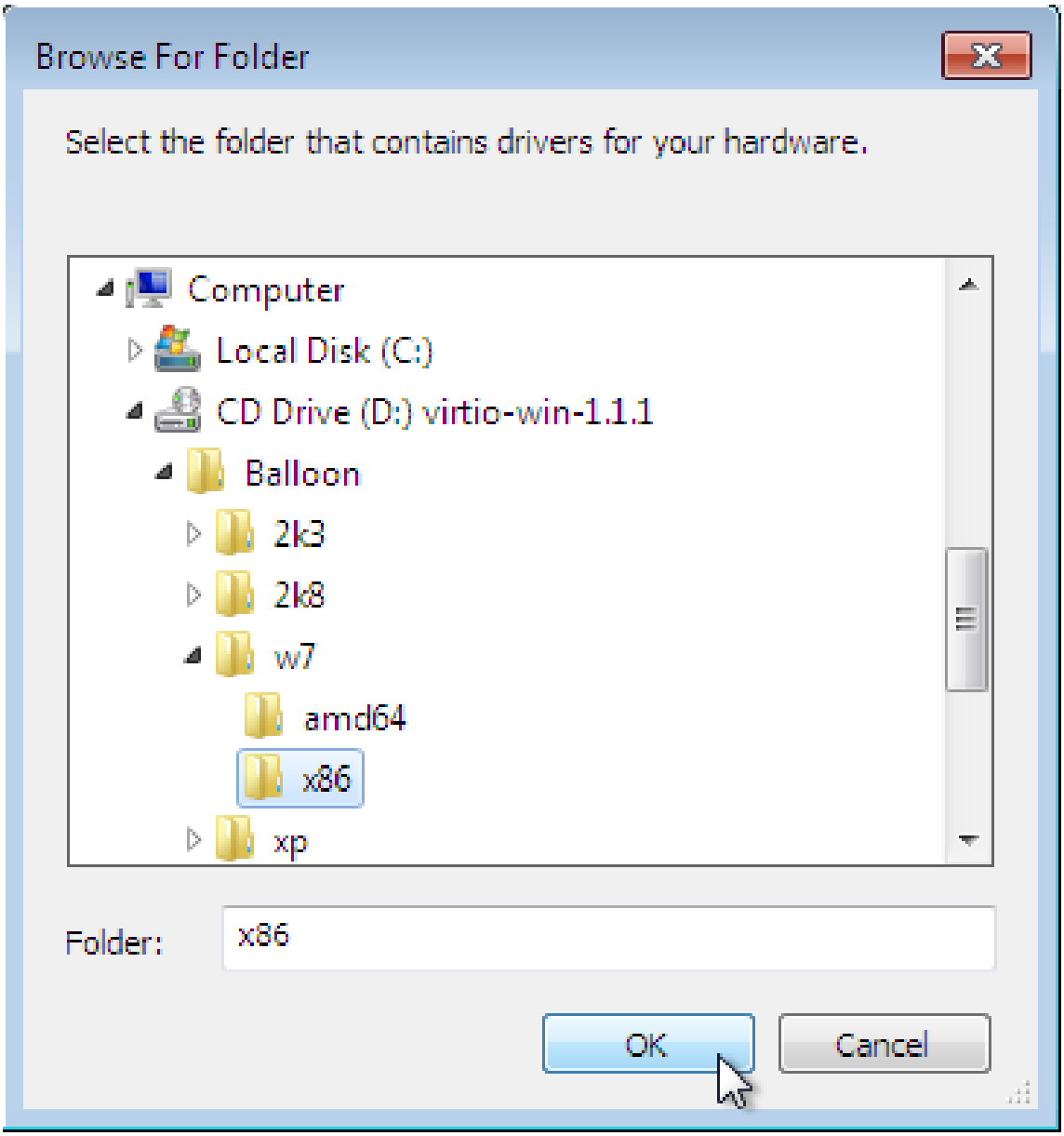
**図10.11 ドライバー更新ウィザード**




b. **ドライバーの場所を参照します。**

**オペレーティングシステムとアーキテクチャーの組み合わせごとに、個別のドライバーが提供されます。ドライバーは、ドライバーのタイプ、オペレーティングシステム、およびインストールされるアーキテクチャー( *driver_type*/*os*/*arch*/ )に従って階層的に編成されます。たとえば、x86 (32 ビット)アーキテクチャーを備えた Windows 7 オペレーティングシステムの Balloon ドライバーは Balloon/w7/x86 ディレクトリーにあります。**

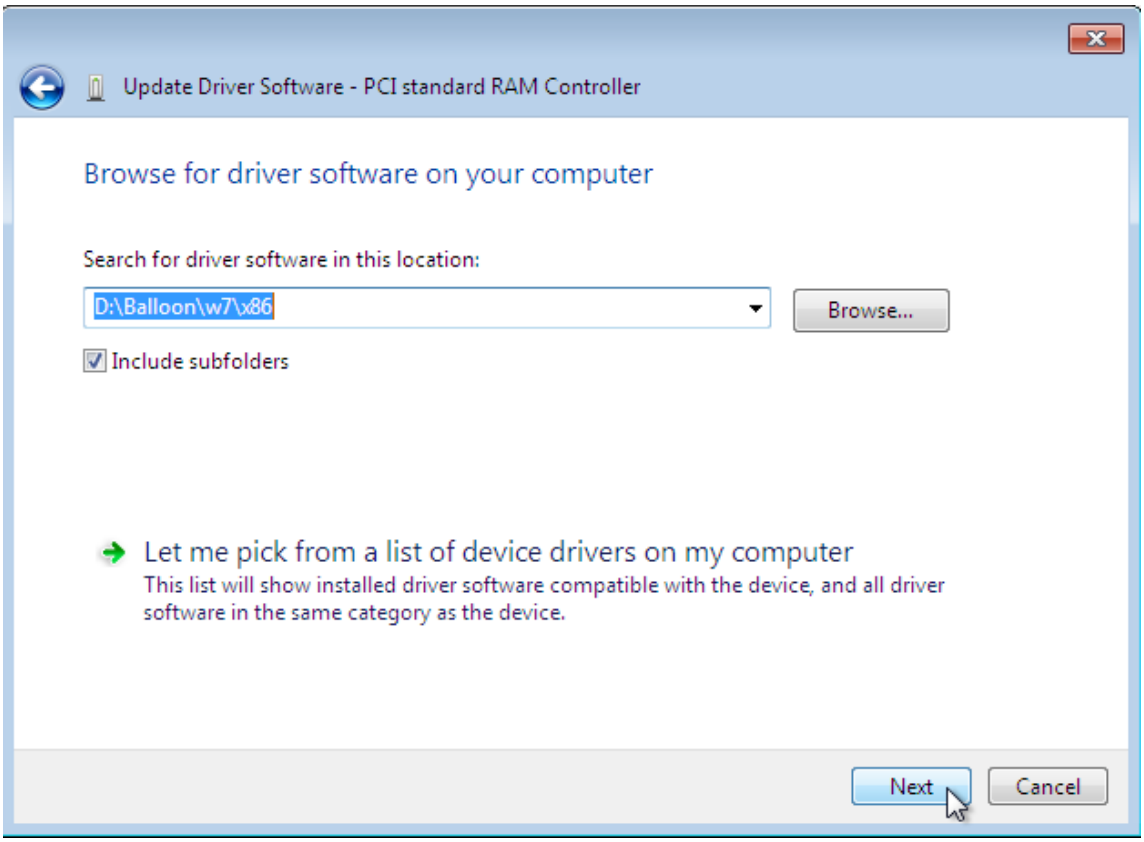
**図10.12 Browse for driver software ポップアップウィンドウ**




正しい場所に移動したら、**OK** をクリックします。

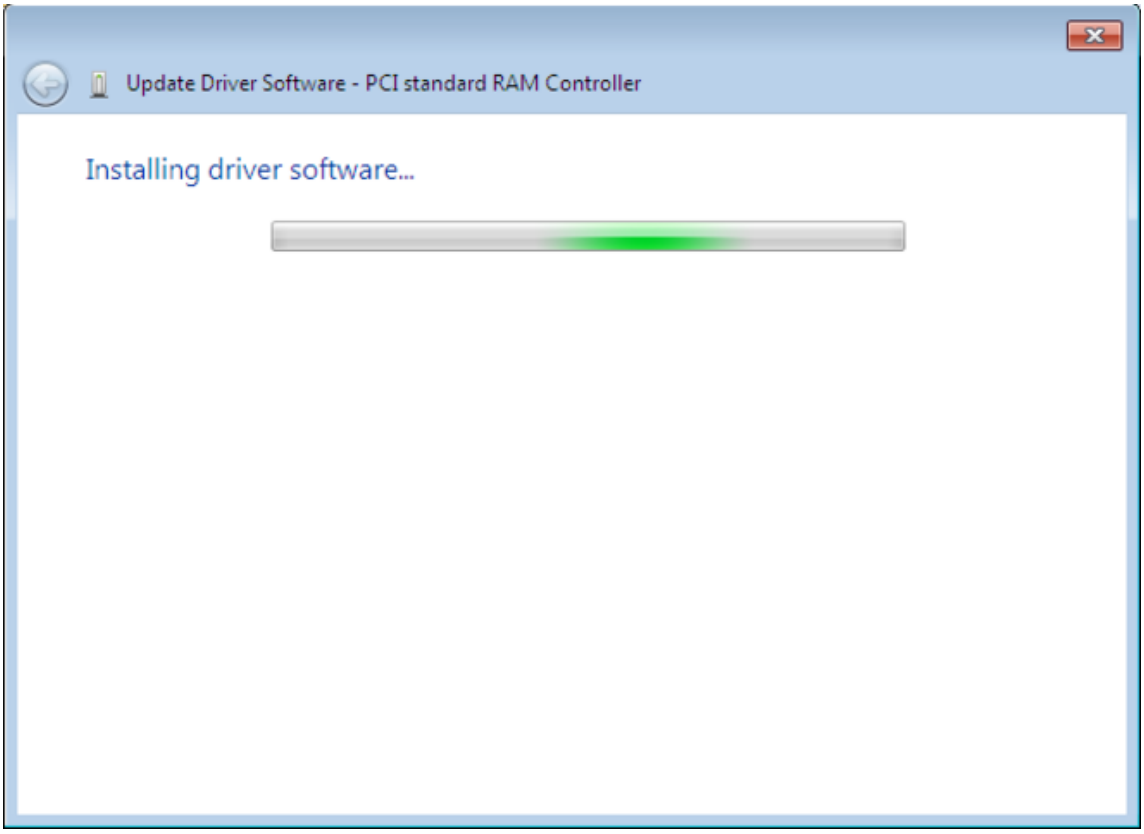
c. **Next** をクリックして続行します。

**図10.13 Update Driver Software ウィザード**




ドライバーのインストール中に、以下の画面が表示されます。

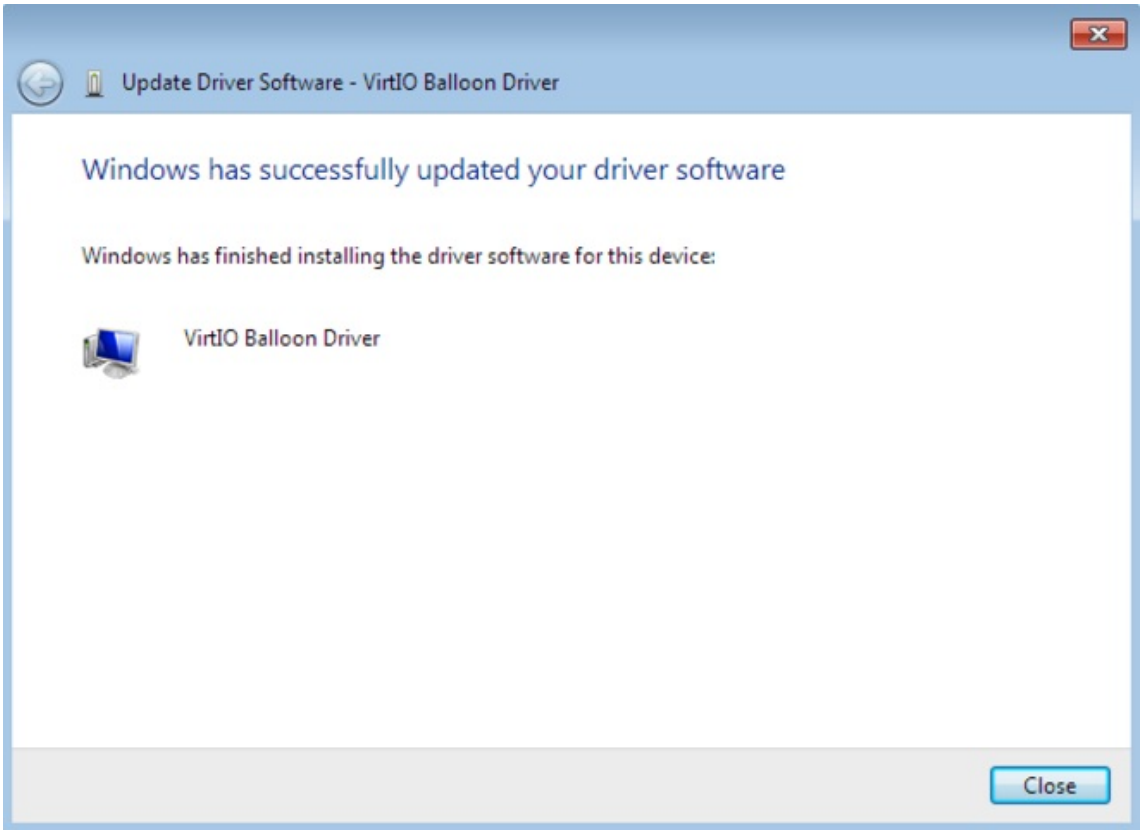
**図10.14 Update Driver Software ウィザード**




6. インストーラーを閉じます。

インストールが完了すると、以下の画面が表示されます。

**図10.15 Update Driver Software ウィザード**




Close をクリックしてインストーラーを閉じます。

7. 再起動

   仮想マシンを再起動してドライバーのインストールを完了します。

## 10.3. WINDOWS インストール時のドライバーのインストール

この手順では、Windows のインストール時に virtio ドライバーをインストールする方法を説明します。

この方法では、Windows ゲスト仮想マシンがデフォルトのストレージデバイスに virtio ドライバーを使用できます。

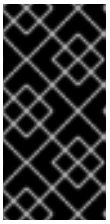
**手順10.3 Windows インストール時の virtio ドライバーのインストール**

1. virtio-win パッケージのインストール

以下のコマンドを使用して、virtio-win パッケージをインストールします。

```
# yum install virtio-win
```

2. ゲスト仮想マシンの作成



重要

仮想マシンを起動せずに、通常通り仮想マシンを作成します。以下の手順のいずれかに従ってください。

以下の いずれか のゲスト作成方法を選択し、手順に従います。

a. virsh を使用したゲスト仮想マシンの作成

この方法では、インストール 前 に virtio ドライバーフロッピーディスクを Windows ゲストに割り当てます。

virsh を使用して XML 定義ファイルから仮想マシンを作成する場合は、 virsh create コマンドではなく、virsh define コマンドを使用します。

i.

仮想マシンを作成しますが、起動しません。virsh コマンドで仮想マシンを作成する方法は、『Red Hat Enterprise Linux Virtualization Administration Guide』 を参照してください。

ii.

virsh コマンドを使用して、ドライバーディスクを仮想化フロッピーディスクとして追加します。この例では、ゲスト仮想マシンに仮想化されたフロッピーデバイスがない場合はコピーおよび使用できます。*vm_name* は仮想マシンの名前に置き換える必要があることに注意してください。

```
# virsh attach-disk vm_name /usr/share/virtio-win/virtio-win.vfd fda --type floppy
```

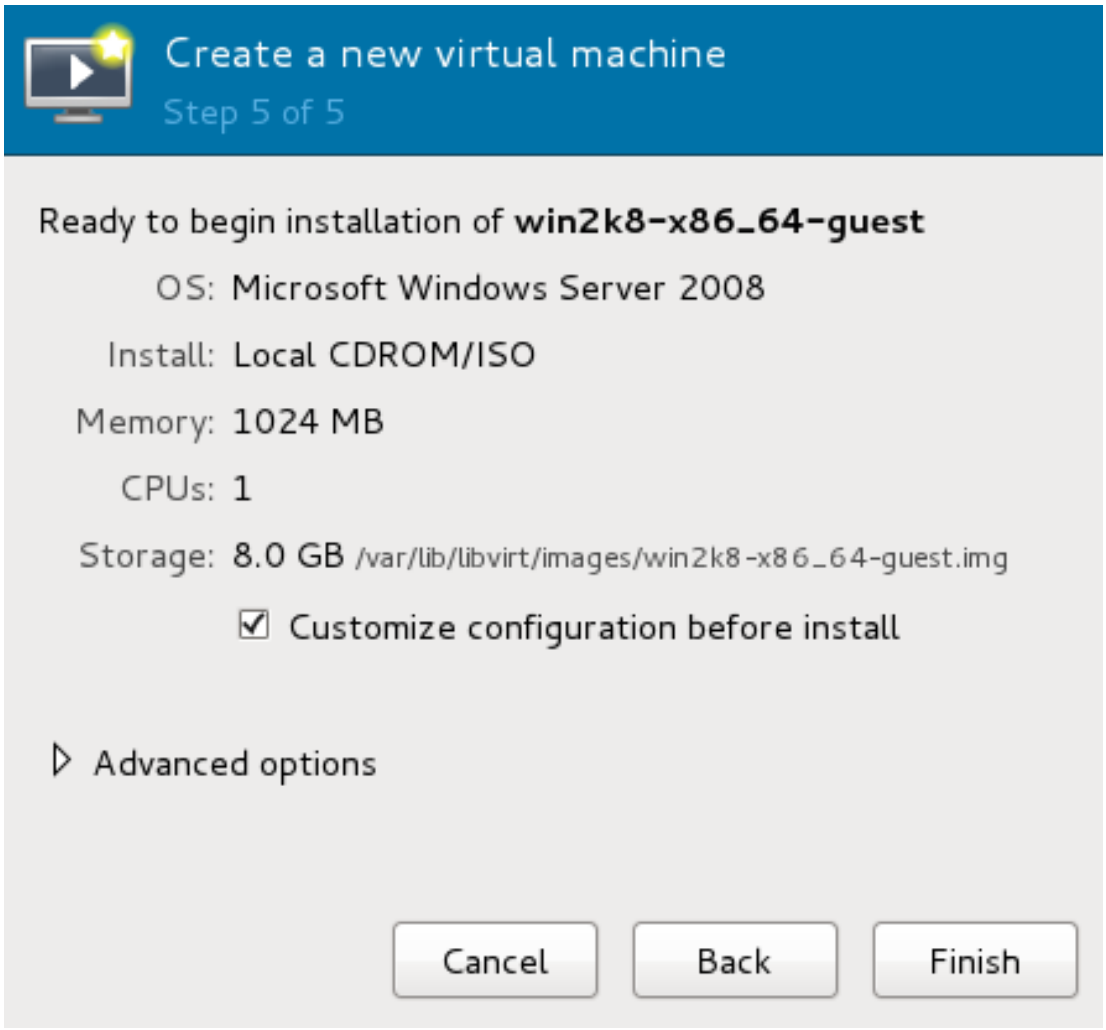
ステップ 3 に進むことができます。

b. virt-manager を使用してゲスト仮想マシンを作成し、ディスクタイプを変更します。

i.

virt-manager ゲスト作成ウィザードの最後のステップで、Customize configuration before install チェックボックスをオンにします。

**図10.16 virt-manager ゲスト作成ウィザード**




**Finish ボタンをクリックして続行します。**

ii. **Add Hardware ウィザードを開きます。**

**新しいパネルの左下にある Add Hardware ボタンをクリックします。**

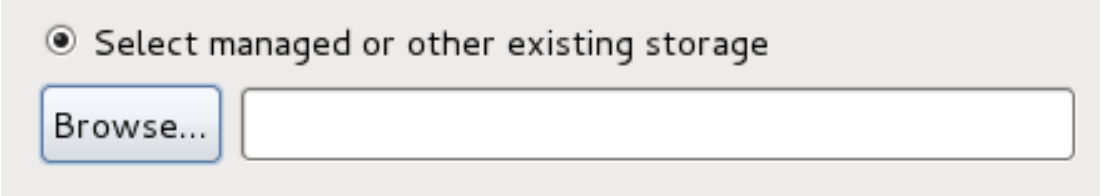
iii. **ストレージデバイスの選択**

**ストレージ は、ハードウェアタイプ 一覧でのデフォルト選択です。**

**図10.17 Add new virtual hardware ウィザード**


**Select managed or other existing storage** ラジオボタンが選択されていることを確認します。**Browse...** をクリックします。

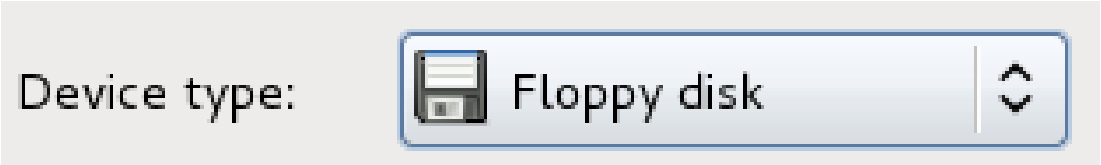
**図10.18 管理ストレージまたは既存のストレージを選択**




表示された新しいウィンドウで、**Browse Local** をクリックします。**/usr/share/virtio-win/virtio-win.vfd** に移動し、**選択** をクリックして確定します。

**Device type** を **Floppy disk** に変更し、**Finish** をクリックして続行します。

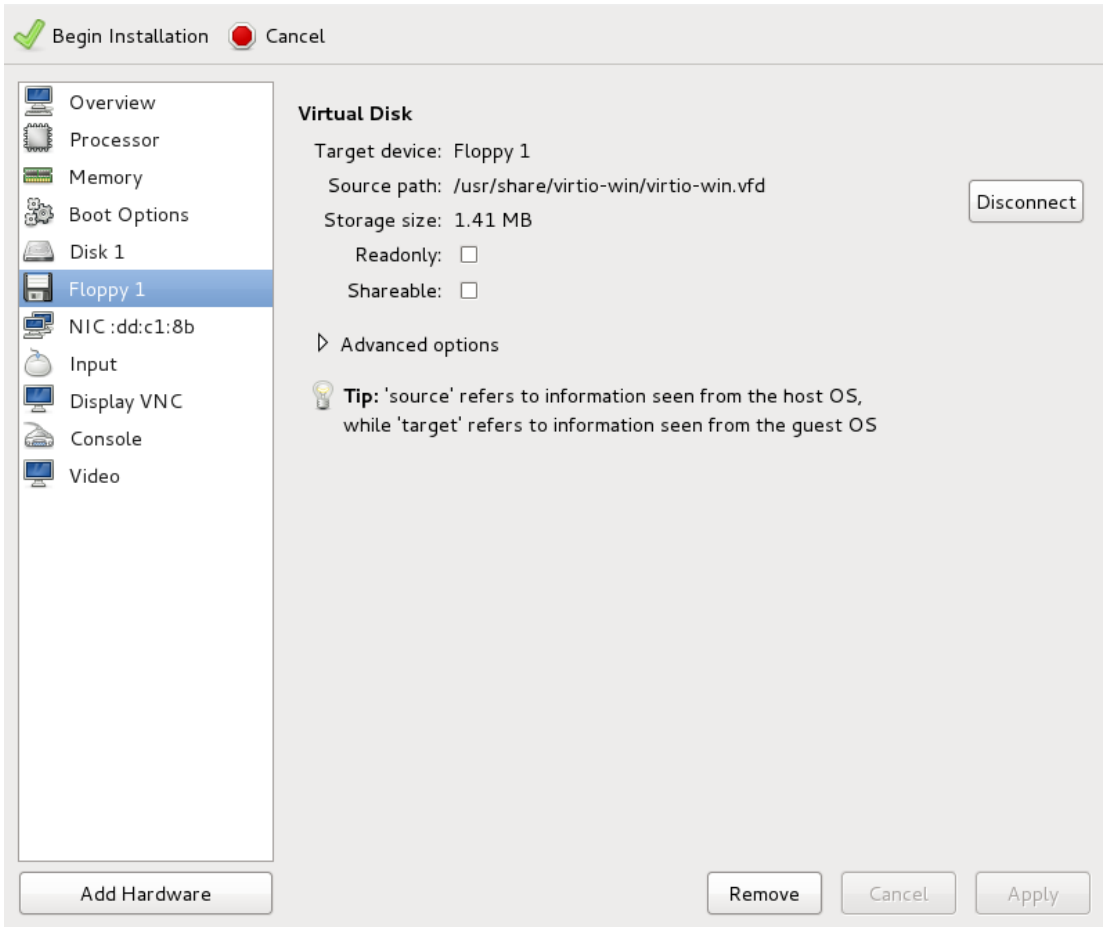
**図10.19 デバイスタイプの変更**




iv. **設定の確認**

デバイス設定を確認します。

**図10.20 仮想マシンのハードウェア情報ウィンドウ**

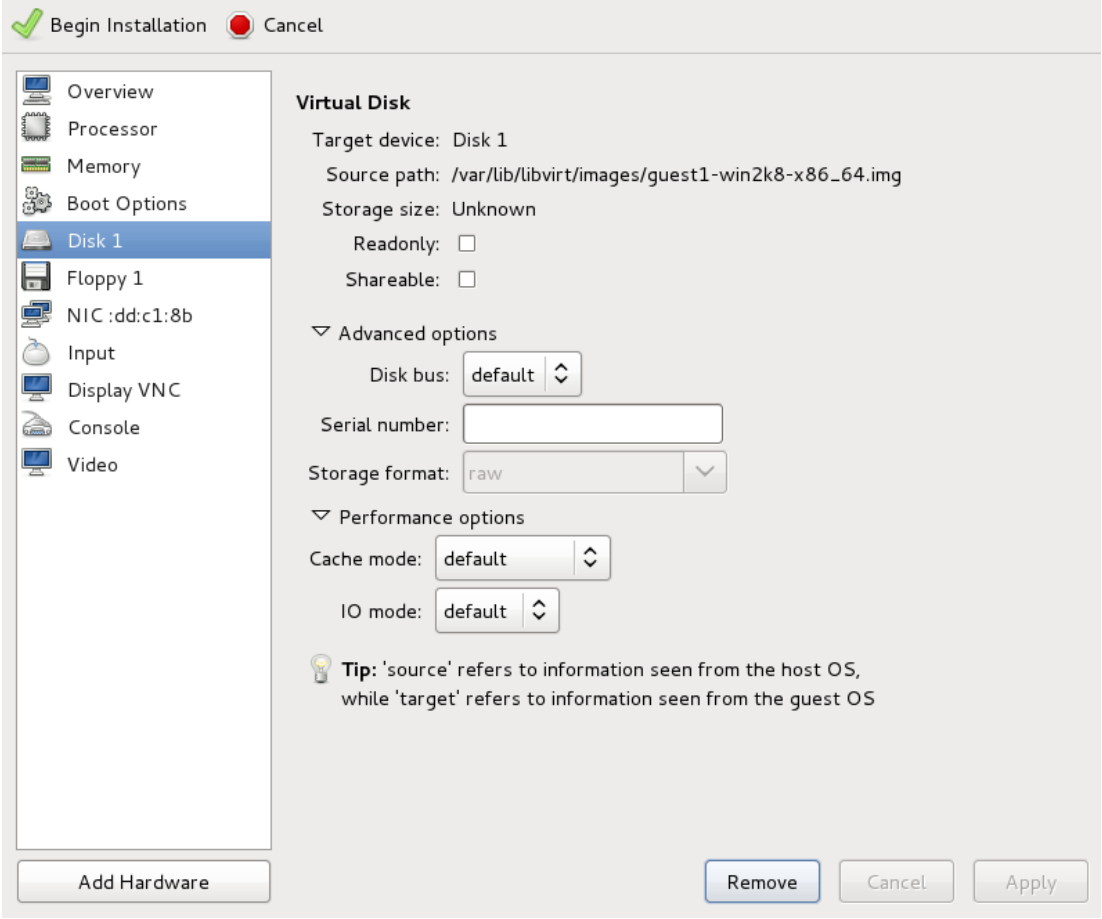



これで、仮想マシンがアクセスできるリムーバブルデバイスが作成されました。

v. ハードディスクの種類を変更する

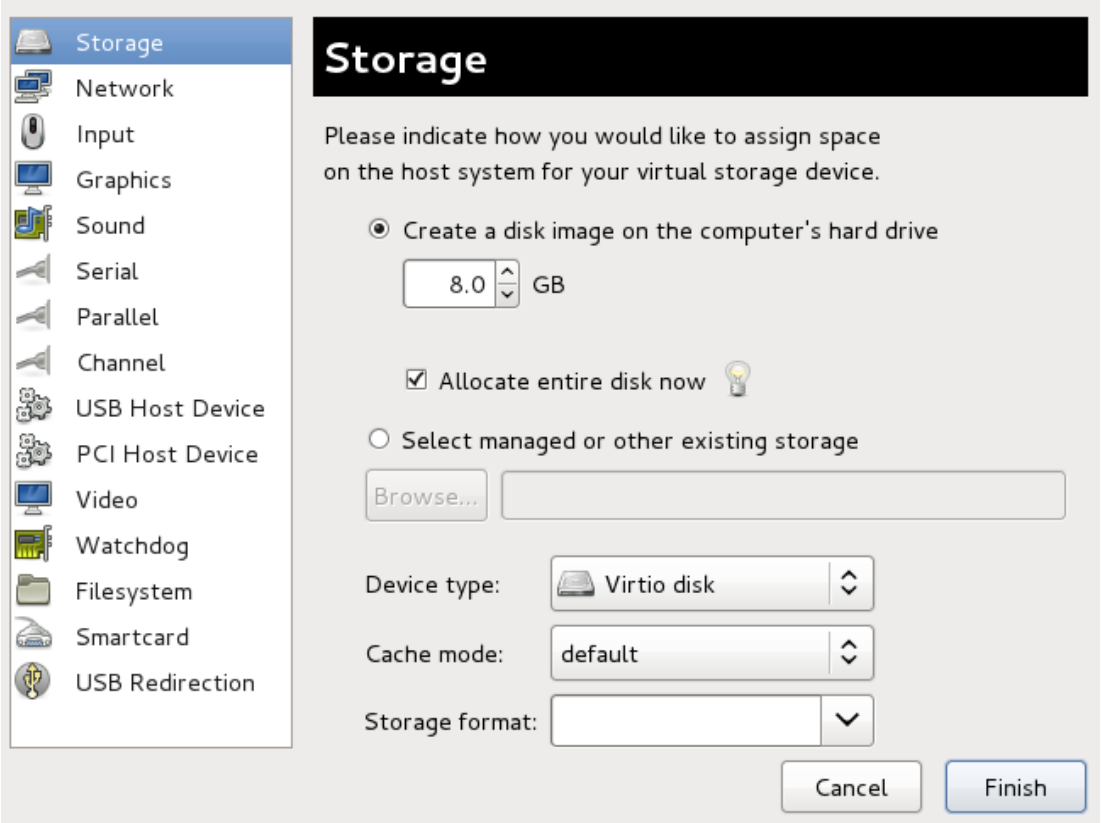
ハードディスクタイプを *IDE ディスクから Virtio Disk* に変更するには、最初に既存のハードディスクである **Disk 1** を削除する必要があります。ディスクを選択し、削除 ボタンをクリックします。

**図10.21 仮想マシンのハードウェア情報ウィンドウ**




**Add Hardware をクリックして、新しい仮想ストレージデバイスを追加します。次に、Device type を *IDE ディスク* から *Virtio Disk* に変更します。Finish をクリックして操作を確認します。**

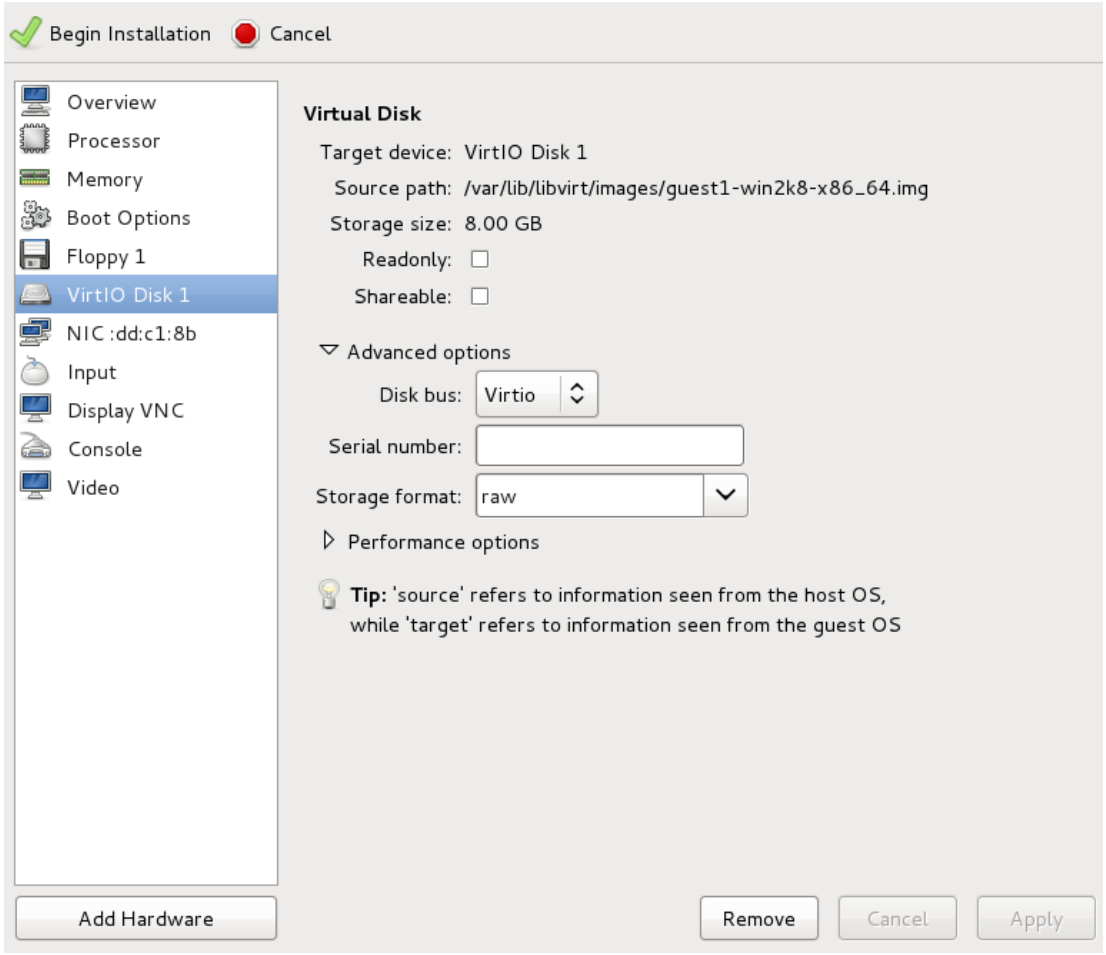
**図10.22 仮想マシンのハードウェア情報ウィンドウ**




vi. **設定が正しいことを確認します。**

***VirtIO Disk 1* の設定を確認します。**

**図10.23 仮想マシンのハードウェア情報ウィンドウ**




設定の詳細が適切であれば **インストールの開始** ボタンをクリック します。

ステップ 3 に進むことができます。

c. **virt-install を使用したゲスト仮想マシンの作成**

**virt-install** コマンドを使用してドライバーディスクをインストールに追加するには、以下のパラメーターを以下のように完全に追加します。

```
--disk path=/usr/share/virtio-win/virtio-win.vfd,device=floppy
```

**重要**

追加するデバイスが ディスク ( フロッピー または **cdrom**ではなく)の場合は、以下のように ***--disk*** パラメーターの最後に **bus=virtio** オプションを追加する必要があります。

```
--disk path=/usr/share/virtio-win/virtio-win.vfd,device=disk,bus=virtio
```

インストールする Windows のバージョンに応じて、以下のいずれかのオプションを virt-install コマンドに追加します。

```
--os-variant win2k3
```

```
--os-variant win7
```

ステップ 3 に進むことができます。

3. ドライバーインストールの追加手順

インストール時に、Windows ゲストのタイプに応じて、ドライバーをインストールするために追加の手順が必要です。

a. Windows Server 2003

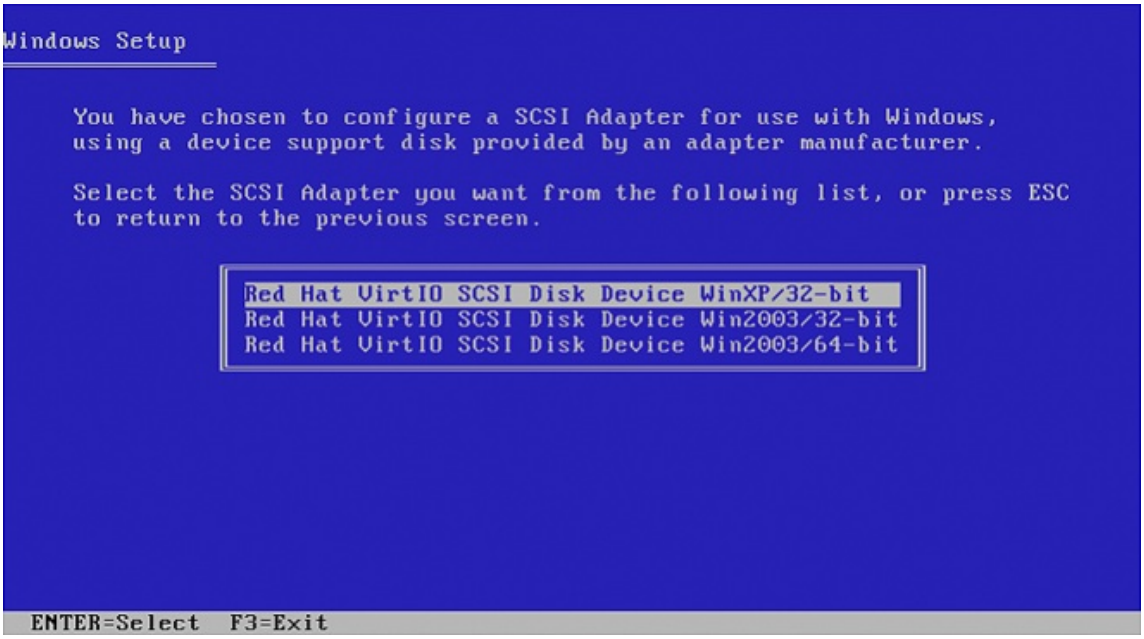
インストール Blue 画面がサードパーティーのドライバー用に F6 を繰り返し押す前。

図10.24 Windows 設定画面


S を押して、追加のデバイスドライバーをインストールします。

**図10.25 Windows 設定画面**


**図10.26 Windows 設定画面**




**Enter を押して、インストールを続行します。**

b. **Windows Server 2008**

**Windows Server 2003 でも同じ手順に従いますが、インストーラーがドライバーを要求されたら、Load Driver をクリックし、インストーラーに A のドライブをポイントして、ゲストオペレーティングシステムとアーキテクチャーに適したドライバーを選択します。**

## 10.4. 既存のデバイス用の KVM VIRTIO ドライバーの使用

ゲストに接続されている既存のハードディスクデバイスを変更して、仮想 IDE ドライバーの代わりに virtio ドライバーを使用できます。このセクションの例では、libvirt 設定ファイルを編集します。これらの手順を実行するためにゲスト仮想マシンをシャットダウンする必要はありませんが、ゲストが完全にシャットダウンされて再起動されるまで、変更は適用されないことに注意してください。

**手順10.4 既存のデバイスへの KVM virtio ドライバーの使用**

1. この手順を続行するには、「KVM Windows virtio ドライバーのインストール」 で説明されているように、適切なドライバー(viostor)がインストールされていることを確認してください。

2. root で virsh edit <guestname> コマンドを実行し、デバイスの XML 設定ファイルを編集します。たとえば、virsh edit guest1 です。設定ファイルは /etc/libvirt/qemu にあります。

3. 以下は、仮想化 IDE ドライバーを使用したファイルベースのブロックデバイスです。これは、virtio ドライバーを使用しない仮想マシンの一般的なエントリーです。

```
<disk type='file' device='disk'>
  <source file='/var/lib/libvirt/images/disk1.img'/>
  <target dev='hda' bus='ide'/>
</disk>
```

4. bus= エントリーを virtio に変更して、virtio デバイスを使用するエントリーを変更します。ディスクが以前 IDE であった場合は、hda、hdb、hdc などのターゲットがあります。bus=virtio に変更する場合は、それに応じてターゲットを vda、vdb、または vdc に変更する必要があります。

```
<disk type='file' device='disk'>
  <source file='/var/lib/libvirt/images/disk1.img'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

5. ディスク タグ内の アドレス タグを削除します。これは、この手順が機能するために必ず行う必要があります。libvirt は、仮想マシンが次に起動したときに、アドレス タグを適切に再生成します。

あるいは、virt-manager、virsh attach-disk、または virsh attach-interface は、virtio ドライバーを使用して新しいデバイスを追加できます。

Virtio の使用に関する詳細は、libvirt の Web サイトを参照してください。 http://www.linux-kvm.org/page/Virtio

## 10.5. 新しいデバイス用の KVM VIRTIO ドライバーの使用

この手順では、virt-manager で KVM virtio ドライバーを使用して新しいデバイスを作成する方法を説明します。

あるいは、virsh attach-disk コマンドまたは virsh attach-interface コマンドを使用して、virtio ドライバーを使用してデバイスを接続できます。

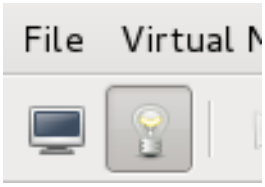
**重要**

新しいデバイスのインストールを続行する前に、ドライバーが Windows ゲストにインストールされていることを確認してください。ドライバーが利用できない場合は、デバイスが認識されず、動作しません。

**手順10.5 virtio ストレージドライバーを使用したストレージデバイスの追加**

1. virt-manager でゲストの名前をダブルクリックして、ゲスト仮想マシンを開きます。
2. lightbulb ボタンをクリックして、Show virtual hardware details タブを開きます。

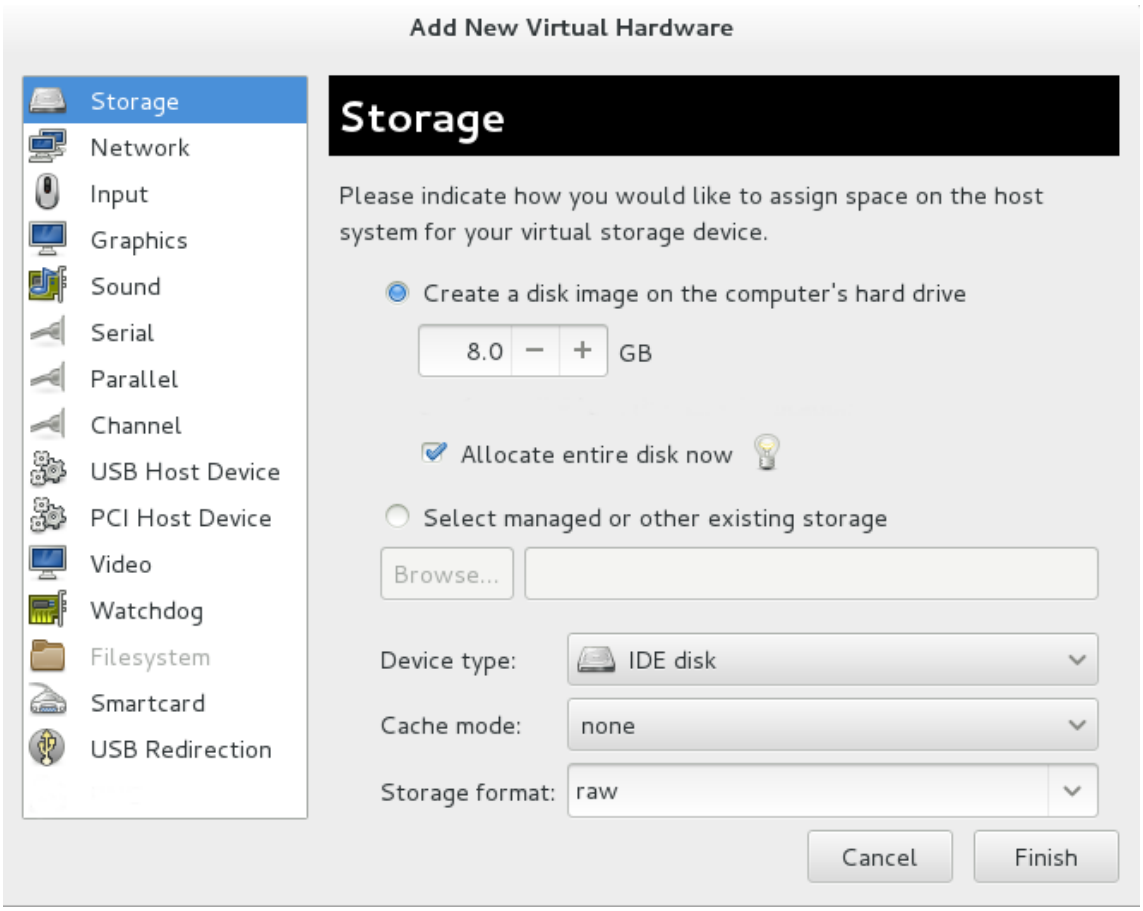
**図10.27 Show virtual hardware details タブ**




3. Show virtual hardware details タブで、Add Hardware ボタンをクリックします。
4. **ハードウェアタイプの選択**

   Hardware typeとして Storage を選択します。

**図10.28 Add new virtual hardware ウィザード**

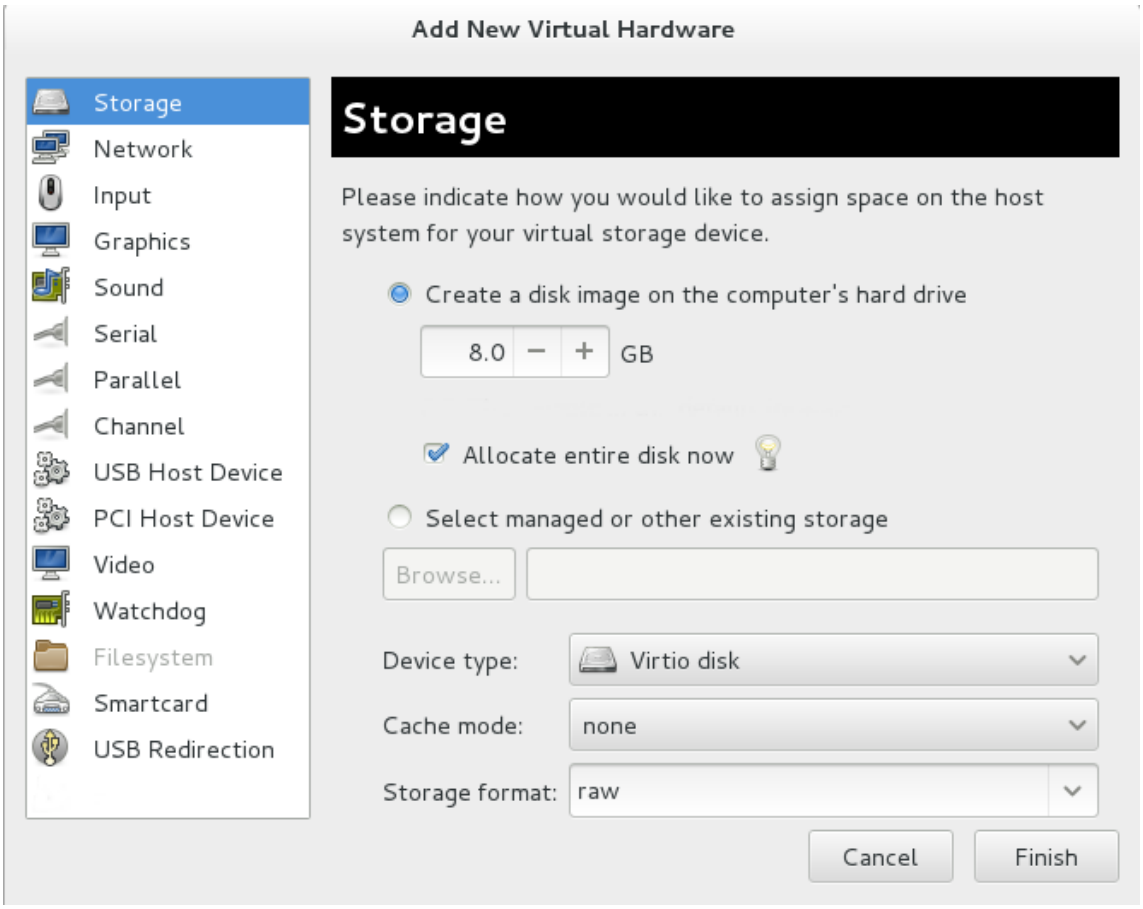



5. **ストレージデバイスとドライバーを選択します。**

   **新しいディスクイメージを作成するか、ストレージプールボリュームを選択します。**

   **Device type を Virtio disk に設定して、virtio ドライバーを使用します。**

**図10.29 Add new virtual hardware ウィザード**



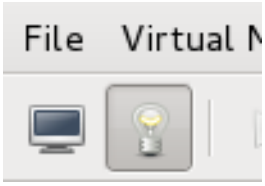

**Finish をクリックして手順を完了します。**

**手順10.6 virtio ネットワークドライバーを使用したネットワークデバイスの追加**

1. **virt-manager でゲストの名前をダブルクリックして、ゲスト仮想マシンを開きます。**

2. **lightbulb ボタンをクリックして、Show virtual hardware details タブを開きます。**

**図10.30 Show virtual hardware details タブ**

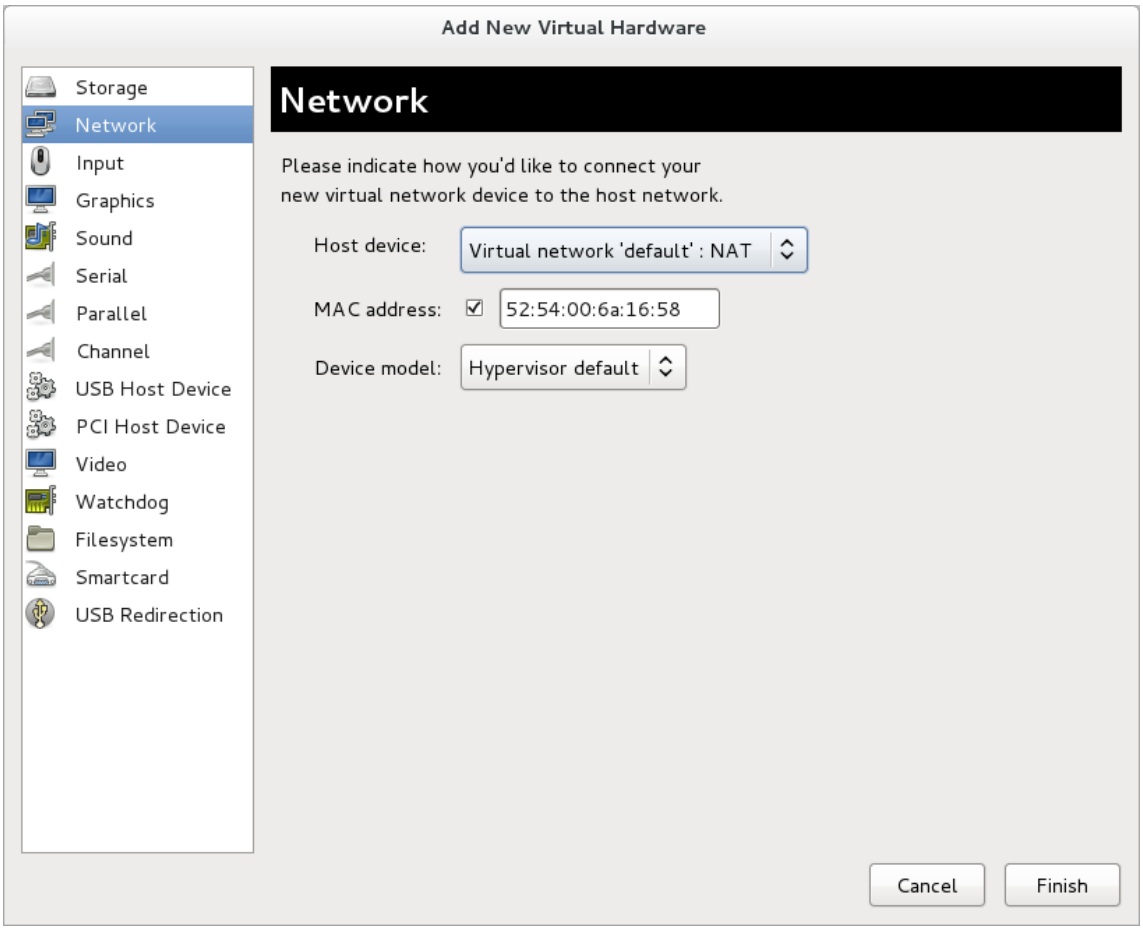



3. **Show virtual hardware details タブで、Add Hardware ボタンをクリックします。**

4. **ハードウェアタイプの選択**

**Network に Hardware type を選択します。**

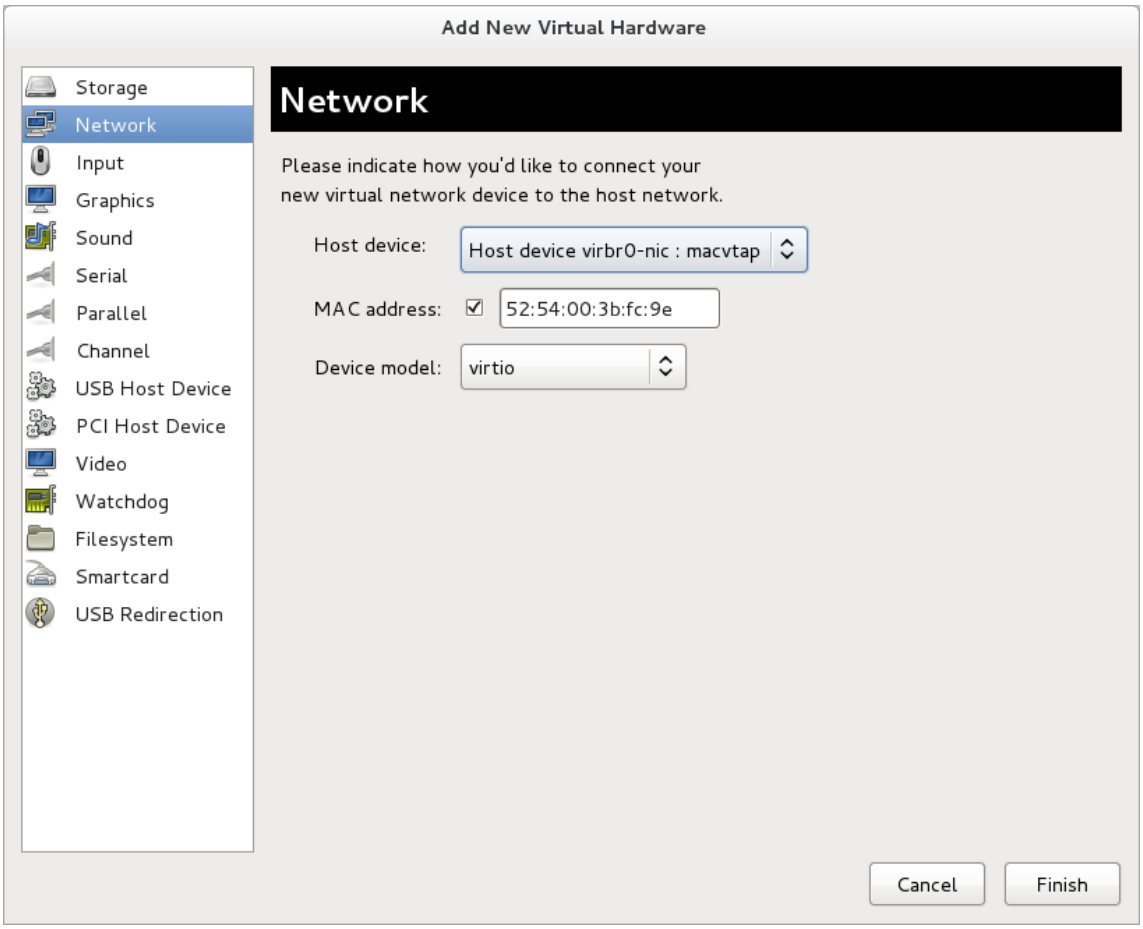
**図10.31 Add new virtual hardware ウィザード**




5. **ネットワークデバイスとドライバーを選択します。**

   **Device model を virtio に設定して、virtio ドライバーを使用します。該当する Host device を選択します。**

**図10.32 Add new virtual hardware ウィザード**




**Finish をクリックして手順を完了します。**

**すべての新規デバイスが追加されたら、仮想マシンを再起動します。Windows 仮想マシンは、ゲストを再起動するまでデバイスを認識しない場合があります。**

# 第11章 NETWORK CONFIGURATION

本章では、libvirt ベースのゲスト仮想マシンが使用する一般的なネットワーク設定を紹介します。詳細は、libvirt ネットワークアーキテクチャーのドキュメントを参照し http://libvirt.org/intro.html てください。

Red Hat Enterprise Linux 6 は、仮想化用に以下のネットワーク設定をサポートします。

- ネットワークアドレス変換 (*NAT*) を使用した仮想ネットワーク
- PCI デバイスの割り当てを使用して直接割り当てた物理デバイス
- PCIe SR-IOV を使用して直接割り当てられた仮想機能
- ブリッジネットワーク

外部ホストがゲスト仮想マシンのネットワークサービスにアクセスできるようにするには、NAT を有効にするか、ネットワークブリッジングを有効にするか、PCI デバイスを直接割り当てる必要があります。

## 11.1. LIBVIRT を使用した NAT (ネットワークアドレス変換)

ネットワーク接続を共有する最も一般的な方法の 1 つは、ネットワークアドレス変換 (NAT) 転送 (仮想ネットワークとも呼ばれます) を使用することです。

### ホストの設定

すべての標準的なlibvirt インストールでは、デフォルトの仮想ネットワークとして、仮想マシンに NAT ベースの接続が提供されます。virsh net-list --all コマンドで使用できることを確認します。

```
# virsh net-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
```

見つからない場合は、サンプルの XML 設定ファイルをリロードしてアクティベートできます。

```
# virsh net-define /usr/share/libvirt/networks/default.xml
```

デフォルトのネットワークは、/usr/share/libvirt/networks/default.xmlから定義されます。

デフォルトのネットワークを自動的に起動するように設定します。

```
# virsh net-autostart default
Network default marked as autostarted
```

デフォルトのネットワークを起動します。

```
# virsh net-start default
Network default started
```

libvirt のデフォルトネットワークが実行すると、分離されたブリッジデバイスが表示されます。このデバイスには、物理インターフェイスが追加されていません。新しいデバイスは、NAT および IP 転送を使用して物理ネットワークに接続します。新規インターフェイスを追加しないでください。

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
```

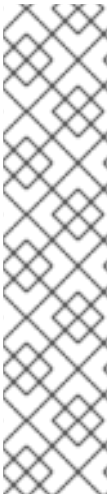
libvirt は、INPUT、FORWARD、OUTPUT、および POSTROUTING チェーンの virbr0 デバイスに接続されているゲスト仮想マシンとの間のトラフィックを許可する iptables ルールを追加します。次に、libvirt は ip_forward パラメーターの有効化を試みます。他のアプリケーションではip_forward が無効になる場合があるため、最善の選択肢は、以下を /etc/sysctl.conf に追加することです。

```
 net.ipv4.ip_forward = 1
```

ゲスト仮想マシンの設定

ホストの設定が完了したら、ゲスト仮想マシンは、その名前を基にして仮想ネットワークに接続できます。ゲストをデフォルトの仮想ネットワークに接続するには、ゲストの XML 設定ファイル( /etc/libvirtd/qemu/myguest.xmlなど)で以下を使用できます。

```
<interface type='network'>
  <source network='default'/>
</interface>
```



注記

MAC アドレスの定義はオプションです。定義しない場合は、MAC アドレスが自動的に生成され、ネットワークが使用するブリッジデバイスの MAC アドレスとして使用されます。MAC アドレスを手動で設定すると、環境全体で一貫性や簡単な参照を維持したり、競合の可能性が非常に低い場合に役立つ場合があります。

```
<interface type='network'>
  <source network='default'/>
  <mac address='00:16:3e:1a:b3:4a'/>
</interface>
```

## 11.2. VHOST-NET の無効化

vhost-net モジュールは virtio ネットワーク用のカーネルレベルのバックエンドで、ユーザー空間 (qemu プロセス)およびカーネル(vhost-net ドライバー)から virtio パケット処理タスクをカーネル( vhost-net ドライバー)に移動することで、仮想化のオーバーヘッドを軽減します。vhost-net は virtio ネットワークインターフェイスでのみ利用できます。vhost-net カーネルモジュールがロードされている場合は、すべての virtio インターフェイスに対してデフォルトで有効になっていますが、vhost-net の使用時に特定のワークロードでパフォーマンスが低下する場合は、インターフェイス設定で無効にできます。

具体的には、UDP トラフィックがホストマシンからそのホスト上のゲスト仮想マシンに送信される場合、ゲスト仮想マシンが受信データをホストマシンが送信する速度よりも遅い速度で処理すると、パフォーマンスが低下する可能性があります。この場合は、vhost-net を有効にすると、UDP ソケットの受信バッファーがより速くオーバーフローし、パケットロスが大きくなります。そのため、このような状況では vhost-net を無効にしてトラフィックを遅くし、全体的なパフォーマンスを向上させることが推奨されます。

vhost-net を無効にするには、ゲスト仮想マシンの XML 設定ファイルの <interface> サブ要素を変更し、次のようにネットワークを定義します。

```
<interface type="network">
  ...
  <model type="virtio"/>
  <driver name="qemu"/>
  ...
</interface>
```

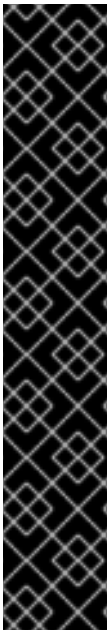
ドライバー名を qemu に設定すると、パケット処理が qemu ユーザー空間に強制され、そのインターフェイスの vhost-net を効果的に無効にします。

## 11.3. LIBVIRT を使用したブリッジネットワーク

ブリッジネットワーク（物理デバイス共有とも呼ばれる）は、物理デバイスを仮想マシンに割り当てるために使用されます。ブリッジングは多くの場合、より高度なセットアップや、複数のネットワークインターフェイスを持つサーバーで使用されます。

**eth0** インターフェイスに基づいてブリッジ(**br0**)を作成するには、ホストで以下のコマンドを実行します。

```
# virsh iface-bridge eth0 br0
```



**重要**

**NetworkManager** はブリッジに対応していません。ネットワークスクリプト( **/etc/sysconfig/network-scripts/** ディレクトリーにある)でネットワークを使用するには、**NetworkManager** を無効にする必要があります。

```
# chkconfig NetworkManager off
# chkconfig network on
# service NetworkManager stop
# service network start
```

**NetworkManager** を完全に無効にしない場合は、ブリッジに使用される **ifcfg-*** ネットワークスクリプトに "***NM_CONTROLLED=no***" を追加します。

# 第12章 PCI デバイスの割り当て

Red Hat Enterprise Linux 6 は、仮想マシンに 3 つのクラスのデバイスを公開します。

- *Emulated devices* は、実際のハードウェアを模倣する純粋な仮想デバイスであり、未変更のゲストオペレーティングシステムは標準のインボックスドライバーを使用して動作できるようにします。

- *VirtIO devices* は、仮想マシンで最適に動作するように設計された仮想デバイスです。VirtIO デバイスはエミュレートされたデバイスと似ていますが、Linux 以外の仮想マシンには、デフォルトで必要となるドライバーは含まれません。Virtual Machine Manager (virt-manager)や Red Hat Enterprise Virtualization Hypervisor などの仮想化管理ソフトウェアは、サポートされている Linux 以外のゲストオペレーティングシステム用にこれらのドライバーを自動的にインストールします。

- *割り当てられたデバイス* は、仮想マシンに公開される物理デバイスです。このメソッドは passthrough としても知られています。デバイスの割り当てにより、仮想マシンはさまざまなタスクで PCI デバイスに排他的にアクセスできるようになり、PCI デバイスがゲストオペレーティングシステムに物理的に接続されているかのように見え、動作します。

  デバイスの割り当ては、グラフィックカードを除く PCI Express デバイスでサポートされています。パラレル PCI デバイスは、割り当てられたデバイスとしてサポートされる場合がありますが、セキュリティーとシステム設定の競合のために厳しい制限があります。

Red Hat Enterprise Linux 6 は、仮想マシンごとに 32 個の PCI デバイススロットをサポートし、デバイススロットごとに 8 つの PCI 機能をサポートします。これにより、ゲストごとに理論上最大 256 個の設定可能な PCI 機能が提供されます。

ただし、この理論上の最大値は、以下の制限の対象となります。

- 各仮想マシンは、割り当てられた最大 8 個のデバイス機能をサポートします。

- PCI デバイススロットは、デフォルトで 5 つのエミュレートされたデバイスで設定されます (2 つのデバイスはスロット 1 にあります)。ただし、ゲストオペレーティングシステムが操作に必要ない場合、ユーザーはデフォルトで設定されるエミュレートされたデバイスの 2 つを明示的に削除できます（スロット 2 のビデオアダプターデバイス、および利用可能な最小スロット（通常はスロット 3）のメモリーバルーンドライバーデバイス）。これにより、サポートされる機能の最大値は、仮想マシンごとに 30 個の PCI デバイススロットになります。

Red Hat Enterprise Linux 6.0 以降では、割り当てられた PCI デバイスを仮想マシンにホットプラグできるようになりました。ただし、PCI デバイスのホットプラグはスロットレベルで動作するため、多機能 PCI デバイスはサポートされません。静的デバイス設定にのみ、多機能の PCI デバイスが推奨されます。

注記

Red Hat Enterprise Linux 6.0 では、デバイスの標準および拡張設定領域へのゲストオペレーティングシステムドライバーへのアクセスが制限されています。Red Hat Enterprise Linux 6.0 に存在する制限は、Red Hat Enterprise Linux 6.1 では大幅に削減され、はるかに大きな PCI Express デバイスセットを KVM ゲストに正常に割り当てることができました。

安全なデバイス割り当てには、割り込みの再マッピングのサポートも必要です。プラットフォームが割り込みの再マッピングをサポートしていない場合、デバイスの割り当ては失敗します。開発環境で割り込みの再マッピングのサポートなしでデバイスの割り当てを使用するには、*allow_unsafe_assigned_interrupts* KVM モジュールパラメーターを 1 に設定します。

PCI デバイスの割り当ては、Intel VT-d または AMD IOMMU のいずれかに対応するハードウェアプラットフォームでのみ利用できます。PCI デバイスの割り当てを機能させるには、この Intel VT-d または AMD IOMMU の仕様が BIOS で有効になっている必要があります。

**手順12.1 PCI デバイス割り当てのための Intel システムの準備**

1. **Intel VT-d 仕様を有効にする**

   Intel VT-d 仕様では、物理デバイスを仮想マシンに直接割り当てるハードウェアサポートが提供されます。この仕様は、Red Hat Enterprise Linux で PCI デバイスの割り当てを使用するために必要です。

   Intel VT-d 仕様は、BIOS で有効にする必要があります。システムの製造元によっては、この仕様をデフォルトで無効にしている場合があります。これらの仕様を表示するのに使用される用語はメーカーにより異なります。適切な用語は、システムの製造元のドキュメントを参照してください。

2. **カーネルで Intel VT-d をアクティブにします。**

   /boot/grub/grub.conf ファイルのカーネル行に *intel_iommu=on* パラメーターを追加して、カーネルで Intel VT-d をアクティブにします。

   以下の例は、Intel VT-d がアクティブ化された変更された grub.conf ファイルです。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-330.x86_645)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-330.x86_64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
intel_iommu=on
        initrd /initrd-2.6.32-330.x86_64.img
```

3. **使用準備完了**

   **システムを再起動して、変更を有効にします。これで、システムで PCI デバイスの割り当てが可能になります。**

**手順12.2 PCI デバイス割り当て用の AMD システムの準備**

1. **AMD IOMMU 仕様を有効にする**

   **Red Hat Enterprise Linux で PCI デバイスの割り当てを使用するには、AMD IOMMU の仕様が必要です。この仕様は、BIOS で有効にする必要があります。システムの製造元によっては、この仕様をデフォルトで無効にしている場合があります。**

2. **IOMMU カーネルサポートの有効化**

   **システムの起動時に AMD IOMMU 仕様が有効になるように、*amd_iommu=on* を /boot/grub/grub.conf のカーネルコマンドラインに追加します。**

3. **使用準備完了**

   **システムを再起動して、変更を有効にします。これで、システムで PCI デバイスの割り当てが可能になります。**

## 12.1. VIRSH を使用した PCI デバイスの割り当て

**この手順では、KVM ハイパーバイザーの仮想マシンに PCI デバイスを割り当てる方法を説明します。**

**この例では、PCI 識別子コード、pci_0000_01_00_0、および完全に仮想化されたゲストマシン *guest1-rhel6-64* を持つ PCIe ネットワークコントローラーを使用します。**

**手順12.3 virsh を使用した PCI デバイスのゲスト仮想マシンへの割り当て**

1. **デバイスの識別**

   **まず、仮想マシンへのデバイス割り当てに指定されている PCI デバイスを特定します。使**

用可能な PCI デバイスの一覧を表示する場合は、lspci コマンドを実行します。grep を使用して、lspci の出力を絞り込むことができます。

この例では、以下の出力で強調表示されているイーサネットコントローラーを使用します。

```
# lspci | grep Ethernet
00:19.0 Ethernet controller: Intel Corporation 82567LM-2 Gigabit Network Connection
01:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
01:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
```

このイーサネットコントローラーは、短い識別子 00:19.0 で表示されます。この PCI デバイスを仮想マシンに割り当てるには、virsh が使用する完全な ID を確認する必要があります。

これを行うには、virsh nodedev-list コマンドを grep コマンドと組み合わせて、ホストマシンに接続されている特定タイプ(pci)のデバイスをすべて一覧表示します。次に、使用するデバイスの短い識別子にマップする文字列の出力を調べます。

この例では、短い識別子 00:19.0 を使用して、イーサネットコントローラーにマップする文字列を示しています。: 文字および . 文字は、完全識別子の下線に置き換えられることに注意してください。

```
# virsh nodedev-list --cap pci
pci_0000_00_00_0
pci_0000_00_01_0
pci_0000_00_03_0
pci_0000_00_07_0
pci_0000_00_10_0
pci_0000_00_10_1
pci_0000_00_14_0
pci_0000_00_14_1
pci_0000_00_14_2
pci_0000_00_14_3
pci_0000_00_19_0
pci_0000_00_1a_0
pci_0000_00_1a_1
pci_0000_00_1a_2
pci_0000_00_1a_7
pci_0000_00_1b_0
pci_0000_00_1c_0
pci_0000_00_1c_1
pci_0000_00_1c_4
pci_0000_00_1d_0
pci_0000_00_1d_1
pci_0000_00_1d_2
pci_0000_00_1d_7
pci_0000_00_1e_0
pci_0000_00_1f_0
```

```
pci_0000_00_1f_2
pci_0000_00_1f_3
pci_0000_01_00_0
pci_0000_01_00_1
pci_0000_02_00_0
pci_0000_02_00_1
pci_0000_06_00_0
pci_0000_07_02_0
pci_0000_07_03_0
```

**使用するデバイスにマップする PCI デバイス番号を記録します。これは別の手順で必要になります。**

2. **デバイス情報の確認**

**ドメイン、バス、および機能の情報は、virsh nodedev-dumpxml コマンドの出力から取得できます。**

```
virsh nodedev-dumpxml pci_0000_00_19_0
<device>
  <name>pci_0000_00_19_0</name>
  <parent>computer</parent>
  <driver>
    <name>e1000e</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>0</bus>
    <slot>25</slot>
    <function>0</function>
    <product id='0x1502'>82579LM Gigabit Network Connection</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
    <capability type='virt_functions'>
    </capability>
  </capability>
</device>
```

3. **必要な設定の詳細を決定する**

**設定ファイルに必要な値は、virsh nodedev-dumpxml pci_0000_00_19_0 コマンドの出力を参照してください。**

**必要に応じて、slot および function の値を 16 進数から(10 進数から)変換し、PCI バスアドレスを取得します。出力の先頭に 0x を追加して、値が 16 進数であることをコンピューターに指示します。**

**この例のデバイスは、bus = 0、slot = 25、および function = 0 の値を持ちます。10 進数の設定では、この 3 つの値が使用されます。**

```
bus='0'
slot='25'
function='0'
```

**16 進数の値に変換したい場合は、以下の例のように printf ユーティリティーを使用して 10 進数の値から変換できます。**

```
$ printf %x 0
0
$ printf %x 25
19
$ printf %x 0
0
```

**このサンプルデバイスは、設定ファイルで以下の 16 進数値を使用します。**

```
bus='0x0'
slot='0x19'
function='0x0'
```

4. **設定の詳細の追加**

**virsh edit を実行し、仮想マシン名を指定し、<source> セクションにデバイスエントリーを追加して、PCI デバイスをゲスト仮想マシンに割り当てます。**

```
# virsh edit guest1-rhel6-64
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0' bus='0x0' slot='0x19' function='0x0'/>
  </source>
</hostdev>
```

**または、virsh attach-device を実行して、仮想マシンの名前とゲストの XML ファイルを指定します。**

```
virsh attach-device guest1-rhel6-64 file.xml
```

5. **デバイス管理を許可する**

**仮想マシンから PCI デバイスを管理できるようにするには、SELinux ブール値を設定します。**

```
# setsebool -P virt_use_sysfs 1
```

6. **仮想マシンの起動**

```
# virsh start guest1-rhel6-64
```

これにより、PCI デバイスが仮想マシンに正常に割り当てられ、ゲストオペレーティングシステムにアクセスできるようになります。

## 12.2. VIRT-MANAGER を使用した PCI デバイスの割り当て

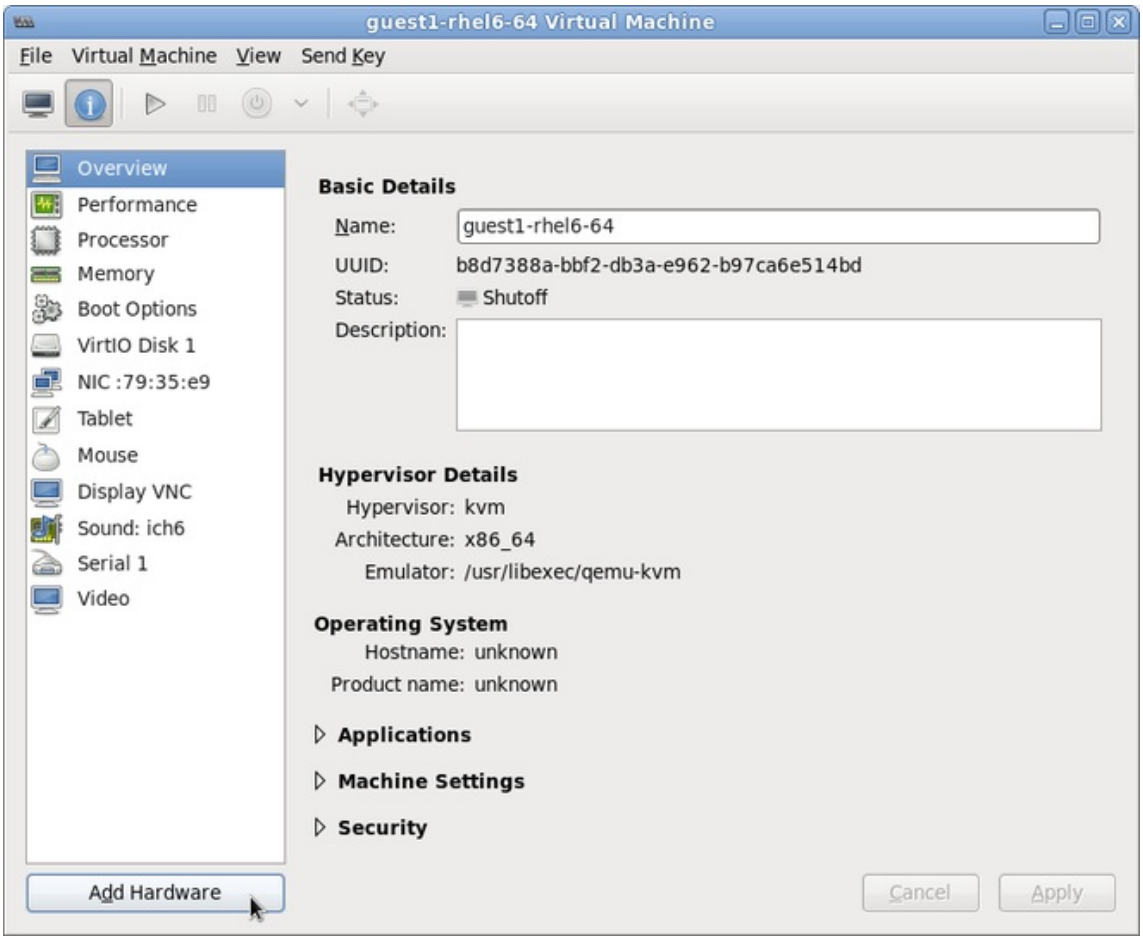
PCI デバイスは、グラフィカルvirt-managerツールを使用してゲスト仮想マシンに追加できます。次の手順では、ギガビットイーサネットコントローラーをゲスト仮想マシンに追加します。

**手順12.4 virt-manager を使用した PCI デバイスのゲスト仮想マシンへの割り当て**

1. **ハードウェア設定を開く**

   ゲスト仮想マシンを開き、Add Hardware をクリックして、仮想マシンに新しいデバイスを追加します。

   **図12.1 仮想マシンのハードウェア情報ウィンドウ**

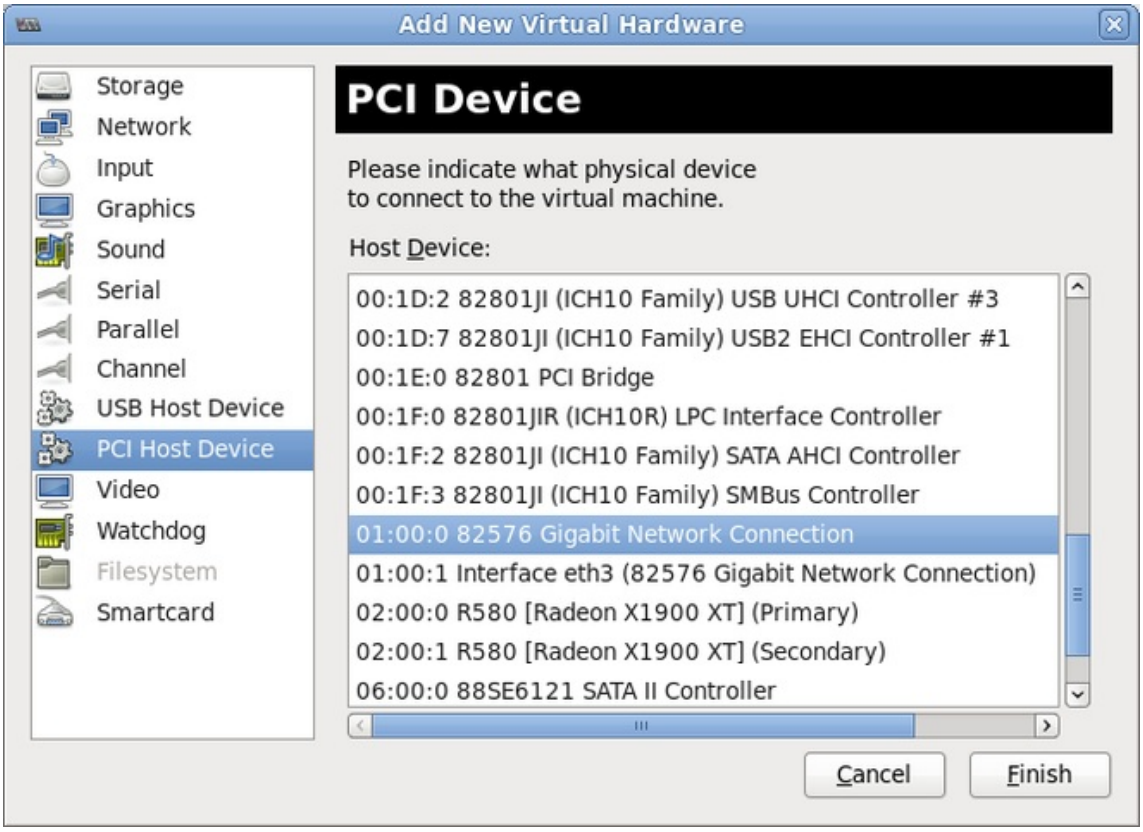



2. **PCI デバイスの選択**

   左側の Hardware 一覧から PCI Host Device を選択します。

未使用の PCI デバイスを選択します。現在、ホストで使用されている PCI デバイスを選択するとエラーが発生することに注意してください。この例では、予備の 82576 ネットワークデバイスが使用されています。Finish を選択して設定を完了します。

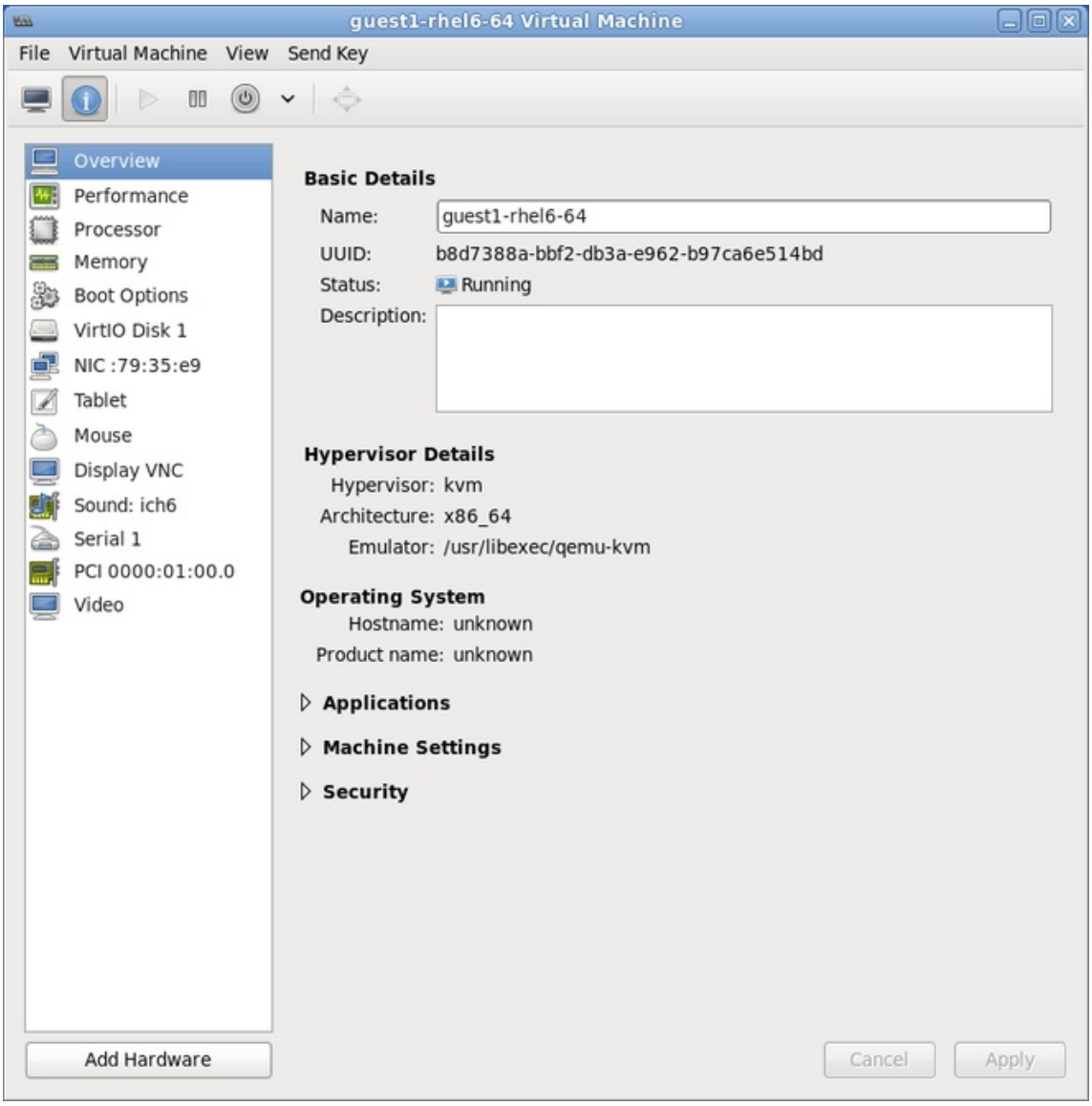
**図12.2 Add new virtual hardware ウィザード**




3. **新しいデバイスの追加**

セットアップが完了し、ゲスト仮想マシンが PCI デバイスに直接アクセスできるようになりました。

図12.3 仮想マシンのハードウェア情報ウィンドウ




## 12.3. VIRT-INSTALL を使用した PCI デバイスの割り当て

virt-install を使用して PCI デバイスを割り当てるには、*--host-device* パラメーターを使用します。

**手順12.5 virt-install を使用した、仮想マシンへの PCI デバイスの割り当て**

1. **デバイスの識別**

   ゲスト仮想マシンにデバイス割り当てに指定されている PCI デバイスを特定します。

```
# lspci | grep Ethernet
00:19.0 Ethernet controller: Intel Corporation 82567LM-2 Gigabit Network Connection
01:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
01:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
```

   virsh nodedev-list コマンドは、システムに接続されているすべてのデバイスの一覧を表示

し、各 PCI デバイスを文字列で識別します。出力を PCI デバイスのみに制限するには、次のコマンドを実行します。

```
# virsh nodedev-list --cap pci
pci_0000_00_00_0
pci_0000_00_01_0
pci_0000_00_03_0
pci_0000_00_07_0
pci_0000_00_10_0
pci_0000_00_10_1
pci_0000_00_14_0
pci_0000_00_14_1
pci_0000_00_14_2
pci_0000_00_14_3
pci_0000_00_19_0
pci_0000_00_1a_0
pci_0000_00_1a_1
pci_0000_00_1a_2
pci_0000_00_1a_7
pci_0000_00_1b_0
pci_0000_00_1c_0
pci_0000_00_1c_1
pci_0000_00_1c_4
pci_0000_00_1d_0
pci_0000_00_1d_1
pci_0000_00_1d_2
pci_0000_00_1d_7
pci_0000_00_1e_0
pci_0000_00_1f_0
pci_0000_00_1f_2
pci_0000_00_1f_3
pci_0000_01_00_0
pci_0000_01_00_1
pci_0000_02_00_0
pci_0000_02_00_1
pci_0000_06_00_0
pci_0000_07_02_0
pci_0000_07_03_0
```

PCI デバイス番号を記録します。この番号は他の手順で確認する必要があります。

ドメイン、バス、および機能の情報は、virsh nodedev-dumpxml コマンドの出力から取得できます。

```
# virsh nodedev-dumpxml pci_0000_01_00_0
<device>
  <name>pci_0000_01_00_0</name>
  <parent>pci_0000_00_01_0</parent>
  <driver>
    <name>igb</name>
```

```
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>1</bus>
    <slot>0</slot>
    <function>0</function>
    <product id='0x10c9'>82576 Gigabit Network Connection</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
    <capability type='virt_functions'>
    </capability>
  </capability>
</device>
```

2. **デバイスの追加**

   **virshnodedev コマンドから出力された PCI 識別子を *--host-device* パラメーターの値として使用します。**

```
virt-install \
--name=guest1-rhel6-64 \
--disk path=/var/lib/libvirt/images/guest1-rhel6-64.img,size=8 \
--nonsparse --graphics spice \
--vcpus=2 --ram=2048 \
--location=http://example1.com/installation_tree/RHEL6.1-Server-x86_64/os \
--nonetworks \
--os-type=linux \
--os-variant=rhel6 \
--host-device=pci_0000_01_00_0
```

3. **インストールを完了する**

   **ゲストのインストールを完了します。PCI デバイスはゲストに接続する必要があります。**

## 12.4. 割り当てられた PCI デバイスの取り外し

**ホストの PCI デバイスがゲストマシンに割り当てられると、ホストがそのデバイスを使用できなくなります。このセクションを読んで、virsh または virt-manager を使用してデバイスをゲストからデタッチし、ホストで使用できるようにする方法を学習してください。**

**手順12.6 virsh を使用したゲストからの PCI デバイスの切り離し**

1. **デバイスの取り外し**

   **次のコマンドを使用して、ゲストの XML ファイルから PCI デバイスを削除し、ゲストから PCI デバイスを切り離します。**

```
# virsh detach-device name_of_guest file.xml
```

2. **デバイスをホストに再接続します (オプション)。**

デバイスが *managed* モードにある場合は、この手順を省略します。デバイスは自動的にホストに戻ります。

デバイスが *managed* モードを使用していない場合は、以下のコマンドを使用して PCI デバイスをホストマシンに再接続します。

```
# virsh nodedev-reattach device
```

たとえば、pci_0000_01_00_0 デバイスーをホストコンピューターに再接続するには、次のコマンドを実行します。

```
virsh nodedev-reattach pci_0000_01_00_0
```

これで、デバイスがホストで使用できるようになります。

**手順12.7 virt-manager を使用したゲストからの PCI デバイスの切り離し**

1. **仮想ハードウェアの詳細画面を開きます。**

   virt-manager で、デバイスを含む仮想マシンをダブルクリックします。Show virtual hardware details ボタンを選択すると、仮想ハードウェアの一覧が表示されます。

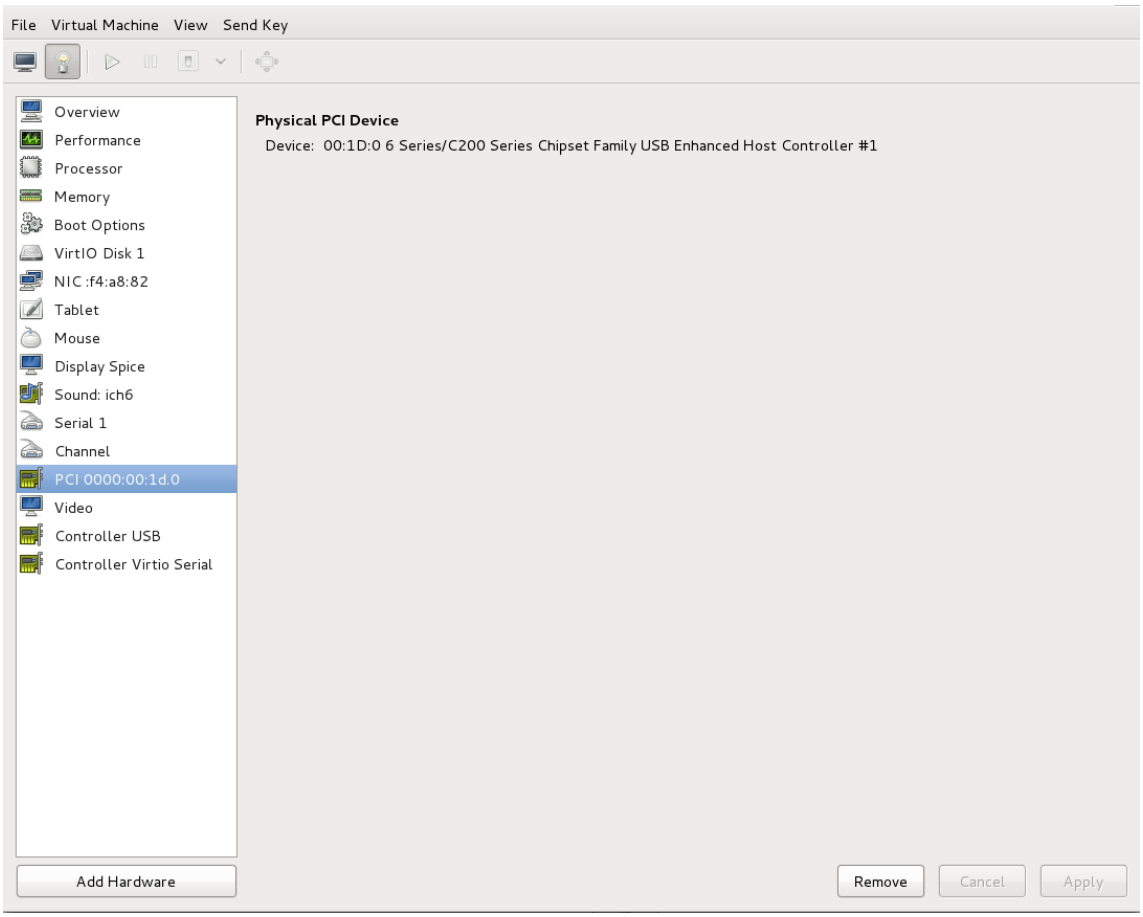
   **図12.4 仮想ハードウェアの詳細ボタン**

2. **デバイスを選択して削除する**

   左側のパネルにある仮想デバイスの一覧から、取り外す PCI デバイスを選択します。

図12.5 取り外す PCI デバイスの選択




Remove ボタンをクリックして確定します。これで、デバイスがホストで使用できるようになります。

## 12.5. PCI デバイスの制限

Red Hat Enterprise Linux 6 は、仮想マシンごとに 32 個の PCI デバイススロットをサポートし、デバイススロットごとに 8 つの PCI 機能をサポートします。これにより、多機能の機能が有効な場合に、理論上最大 256 個の PCI 機能が提供されます。

ただし、この理論上の最大値は、以下の制限の対象となります。

- 各仮想マシンは、割り当てられた最大 8 個のデバイス機能をサポートします。
- PCI デバイススロットは、デフォルトで 5 つのエミュレートされたデバイスで設定されます (2 つのデバイスはスロット 1 にあります)。ただし、ゲストオペレーティングシステムが操作に必要ない場合、ユーザーはデフォルトで設定されるエミュレートされたデバイスの 2 つを明示的に削除できます（スロット 2 のビデオアダプターデバイス、および利用可能な最小スロット

（通常はスロット 3）のメモリーバルーンドライバーデバイス）。これにより、サポートされる機能の最大値は、仮想マシンごとに 30 個の PCI デバイススロットになります。

以下の制限は、PCI デバイスの割り当てにも適用されます。

- PCI デバイスの割り当て (仮想マシンへの PCI デバイスの割り当て) では、ホストシステムが AMD IOMMU または Intel VT-d をサポートして、PCIe デバイスのデバイス割り当てを有効にする必要があります。
- 並行/レガシー PCI の場合、PCI ブリッジの背後にある単一デバイスのみがサポートされます。
- 非ルート PCIe スイッチを介して接続された複数の PCIe エンドポイントには、PCIe スイッチの PCIe ブリッジで ACS サポートが必要です。この制限を無効にするには、/etc/libvirt/qemu.conf ファイルを編集し、以下の行を追加します。

```
relaxed_acs_check=1
```

- Red Hat Enterprise Linux 6 では、ゲストデバイスドライバーによる PCI 設定領域のアクセスが制限されています。この制限により、PCI 設定領域に依存するドライバーの設定が失敗する可能性がありました。
- Red Hat Enterprise Linux 6.2 では、PCI デバイスの割り当ての要件として割り込みの再マッピングが導入されました。プラットフォームが割り込みの再マッピングに対応しない場合は、コマンドラインプロンプトで以下のコマンドを使用し、KVM チェックを回避してください。

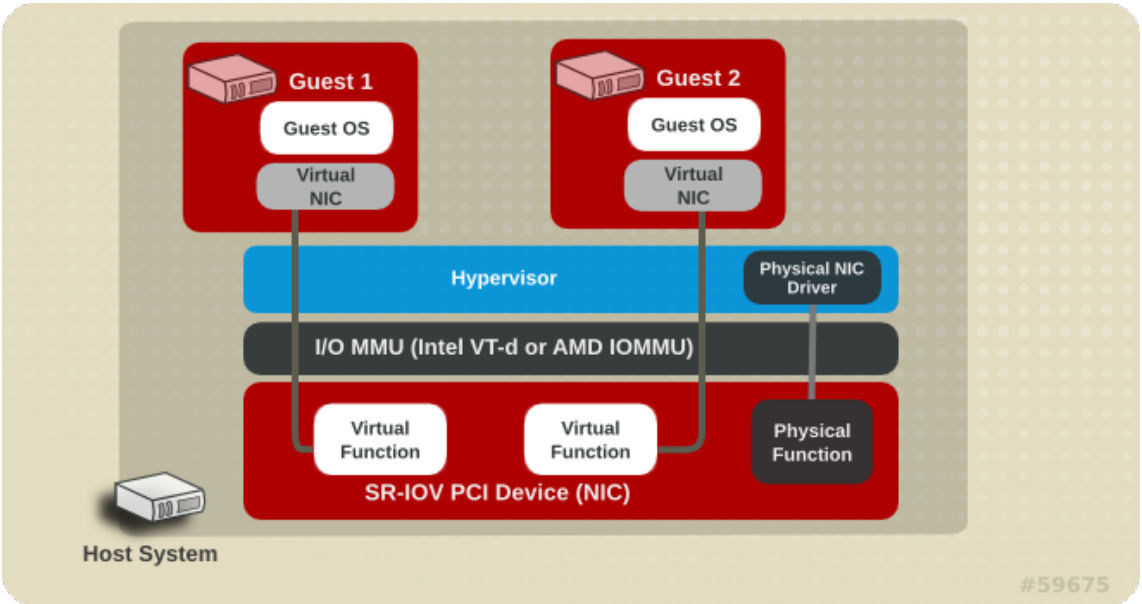
```
# echo 1 > /sys/module/kvm/parameters/allow_unsafe_assigned_interrupts
```

# 第13章 SR-IOV

## 13.1. はじめに

PCI-SIG (PCI Special Interest Group) が開発した SR-IOV (Single Root I/O Virtualization) 仕様は、単一デバイスを複数の仮想マシンに共有できる PCI デバイス割り当てタイプの標準です。SR-IOV は、仮想マシンのデバイスパフォーマンスを改善します。

図13.1 SR-IOV の仕組み




SR-IOV を使用すると、単一の root 機能 (たとえば、単一のイーサネットポート) を複数の個別の物理デバイスとして表示できます。SR-IOV 機能を持つ物理デバイスは、PCI 設定領域に複数の機能として表示されるように設定できます。各デバイスには、ベースアドレスレジスタ (BAR) を含む独自の設定領域があります。

SR-IOV は、次の 2 つの PCI 機能を使用します。

- Physical Function (PF) は、SR-IOV 機能を含む完全な PCIe デバイスです。物理機能は、検出、管理、および通常の PCI デバイスとして設定されます。物理機能は、仮想機能を割り当てることにより、SR-IOV 機能を設定および管理します。

- 仮想機能 (VF) は、I/O のみを処理する単純な PCIe 機能です。各仮想機能は、物理機能から派生しています。デバイスに含まれる仮想機能の数は、デバイスのハードウェアにより制限されます。単一のイーサネットポートである物理デバイスは、仮想マシンで共有できる多くの仮想機能にマップできます。

ハイパーバイザーは、1 つ以上の Virtual Function を仮想マシンにマッピングできます。その後、仮

想機能の設定領域は、ゲストに提示される設定領域にマッピングされます。

仮想機能には実際のハードウェアリソースが必要なため、各仮想機能は一度に 1 つのゲストにのみマッピングできます。仮想マシンには、複数の仮想機能を設定できます。仮想機能は、通常のネットワークカードがオペレーティングシステムに表示されるのと同じ方法で、ネットワークカードとして表示されます。

SR-IOV ドライバーはカーネルで実装されます。コア実装は PCI サブシステムに含まれていますが、物理機能 (PF) デバイスと仮想機能 (VF) デバイスの両方のドライバーサポートも必要です。SR-IOV 対応デバイスは、PF から VF を割り当てることができます。VF は、キューやレジスタセットなどのリソースにより、物理 PCI デバイスにバッキングされている PCI デバイスとして表示されます。

**SR-IOV の利点**

SR-IOV デバイスは、単一の物理ポートを複数の仮想マシンと共有できます。

仮想機能は、準仮想化ドライバーおよびエミュレートされたアクセスよりもネイティブに近いパフォーマンスを発揮し、パフォーマンスを向上させます。仮想機能は、データがハードウェアによって管理および制御されるのと同じ物理サーバー上の仮想マシン間のデータ保護を提供します。

この機能により、データセンター内のホストで仮想マシンの密度を上げることができます。

SR-IOV は、複数のゲストがあるデバイスの帯域幅をより有効に利用できます。

## 13.2. SR-IOV の使用

本セクションでは、PCI パススルーを使用して、SR-IOV 対応マルチポートネットワークカードの Virtual Function をネットワークデバイスとして仮想マシンに割り当てる方法について説明します。

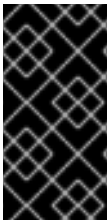
SR-IOV Virtual Function (VF)は、virsh edit コマンドまたは virsh attach-device コマンドを使用して *<hostdev>* にデバイスエントリーを追加することで、仮想マシンに割り当てることができます。ただし、これは通常のネットワークデバイスとは異なり、SR-IOV VF ネットワークデバイスには永続的な一意の MAC アドレスがなく、ホストを再起動するたびに新しい MAC アドレスが割り当てられるため、問題になることがあります。このため、システムの再起動後にゲストに同じ VF が割り当てられた場合でも、ホストの再起動時に、ゲストのネットワークアダプターが新しい MAC アドレスを持つと判断します。その結果、ゲストは毎回新しいハードウェアが接続されていると考え、通常はゲストのネットワーク設定を再設定する必要があります。

libvirt-0.9.10 後半には *<interface type='hostdev'>* インターフェイスデバイスが含まれています。このインターフェイスデバイスを使用すると、libvirt は最初に、示されたネットワーク固有のハードウェア/スイッチの初期化(MAC アドレス、VLAN タグ、802.1Qbh 仮想ポートパラメーターの設定など)を実行し、次にゲストへの PCI デバイス割り当てを実行します。

*<interface type='hostdev'>* インターフェイスデバイスを使用するには、以下が必要です。

- SR-IOV 対応のネットワークカード、
- Intel VT-d または AMD IOMMU 拡張機能のいずれかをサポートするホストハードウェア
- 割り当てる VF の PCI アドレス。

SR-IOV 対応のネットワークインターフェイスカード(NIC)の一覧は、を参照してください https://access.redhat.com/articles/1390483。



重要

SR-IOV デバイスを仮想マシンに割り当てるには、ホストハードウェアが Intel VT-d または AMD IOMMU 仕様をサポートしている必要があります。

Intel または AMD システムに SR-IOV ネットワークデバイスを接続する場合は、以下の手順を行います。

手順13.1 Intel または AMD システムでの SR-IOV ネットワークデバイスの接続

1. BIOS およびカーネルで、Intel VT-d または AMD IOMMU の仕様を有効にします。

   Intel システムでは、BIOS で Intel VT-d が有効になっていない場合は有効にします。BIOS およびカーネルで Intel VT-d を有効にする方法については、手順12.1「PCI デバイス割り当てのための Intel システムの準備」 を参照してください。

   Intel VT-d が既に有効で機能している場合は、この手順をスキップしてください。

AMD システムでは、BIOS で AMD IOMMU 仕様が有効になっていない場合は有効にします。BIOS で IOMMU を有効にする方法については、手順12.2「PCI デバイス割り当て用の AMD システムの準備」 を参照してください。

2. サポートの確認

SR-IOV 機能のある PCI デバイスが検出されたかどうかを確認します。この例では、SR-IOV に対応する Intel 82576 ネットワークインターフェイスカードを一覧表示します。lspci コマンドを使用して、デバイスが検出されたかどうかを確認します。

```
# lspci
03:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
03:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
```

この出力は、その他のデバイスをすべて削除するように変更されていることに注意してください。

3. SR-IOV カーネルモジュールの起動

デバイスがサポートされている場合は、ドライバーカーネルモジュールをカーネルが自動的に読み込む必要があります。オプションのパラメーターは、modprobe コマンドを使用してモジュールに渡すことができます。Intel 82576 ネットワークインターフェイスカードは、igb ドライバーカーネルモジュールを使用します。

```
# modprobe igb [<option>=<VAL1>,<VAL2>,]
# lsmod |grep igb
igb    87592  0
dca    6708    1 igb
```

4. 仮想機能のアクティブ化

igb モジュールの *max_vfs* パラメーターは、仮想機能の最大数を割り当てます。*max_vfs* パラメーターにより、ドライバーは Virtual Functions の パラメーターの値まで生成されます。この特定のカードでは、有効な範囲は *0* から *7* です。

モジュールを削除して変数を変更します。

```
# modprobe -r igb
```

*max_vfs* に設定してモジュールを再起動するか、デバイスでサポートされる最大数の Virtual Function を設定します。*7*

```
# modprobe igb max_vfs=7
```

5. **仮想機能を永続化します。**

   **igb max_vfs=7 の行オプション を /etc/modprobe.d のファイルに追加して、仮想機能を永続的にします。以下に例を示します。**

```
# echo "options igb max_vfs=7" >>/etc/modprobe.d/igb.conf
```

6. **新しい仮想機能の検査**

   **lspci コマンドを使用して、Intel 82576 ネットワークデバイスに追加した仮想機能の一覧を表示します。(もしくは、grep を使用して Virtual Function を検索し、仮想機能に対応するデバイスを検索します。)**

```
# lspci | grep 82576
0b:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
0b:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection(rev 01)
0b:10.0 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.1 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.2 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.3 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.4 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.5 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.6 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.7 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.0 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.1 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.2 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.3 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.4 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.5 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
```

   **PCI デバイスの識別子は、lspci コマンドの *-n* パラメーターで確認できます。Physical Function は、0b:00.0 および 0b:00.1 に対応しています。すべての Virtual Function には、説明に Virtual Function が含まれます。**

7. **デバイスが virsh で存在することを確認します。**

   **libvirt サービスは、デバイスを仮想マシンに追加する前にデバイスを認識する必要があります。libvirt は、lspci 出力と同様の表記を使用します。lspci 出力の句読点文字（、、および）はすべてアンダースコア(_)に変更され ます。**

   **virsh nodedev-list コマンドと grep コマンドを使用して、利用可能なホストデバイスの一覧から Intel 82576 ネットワークデバイスをフィルターにかけます。*0b* は、この例では Intel 82576 ネットワークデバイスのフィルターです。これはシステムにより異なるため、デバイスが追加される可能性があります。**

```
# virsh nodedev-list | grep 0b
pci_0000_0b_00_0
```

```
pci_0000_0b_00_1
pci_0000_0b_10_0
pci_0000_0b_10_1
pci_0000_0b_10_2
pci_0000_0b_10_3
pci_0000_0b_10_4
pci_0000_0b_10_5
pci_0000_0b_10_6
pci_0000_0b_11_7
pci_0000_0b_11_1
pci_0000_0b_11_2
pci_0000_0b_11_3
pci_0000_0b_11_4
pci_0000_0b_11_5
```

**仮想機能と物理機能のシリアル番号は、一覧に含まれている必要があります。**

8. **virsh でデバイスの詳細を取得する**

**pci_0000_0b_00_0 は物理機能の 1 つで、pci_0000_0b_10_0 は、その物理機能で最初に対応する仮想機能です。virsh nodedev-dumpxml コマンドを使用して、両方のデバイスの高度な出力を取得します。**

```
# virsh nodedev-dumpxml pci_0000_0b_00_0
<device>
  <name>pci_0000_0b_00_0</name>
  <parent>pci_0000_00_01_0</parent>
  <driver>
    <name>igb</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>11</bus>
    <slot>0</slot>
    <function>0</function>
    <product id='0x10c9'>82576 Gigabit Network Connection</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
  </capability>
</device>
```

```
# virsh nodedev-dumpxml pci_0000_0b_10_0
<device>
  <name>pci_0000_0b_10_0</name>
  <parent>pci_0000_00_01_0</parent>
  <driver>
    <name>igbvf</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>11</bus>
    <slot>16</slot>
    <function>0</function>
```

```
    <product id='0x10ca'>82576 Virtual Function</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
  </capability>
</device>
```

**この例では、仮想機能 pci_0000_0b_ 10_0 を ステップ 9 の仮想マシンに追加します。仮想機能の bus、slot および function のパラメーターに注意してください。これらはデバイスの追加に必要です。**

**このパラメーターは、/tmp/new-interface.xml などの一時 XML ファイルにコピーします。**

```
<interface type='hostdev' managed='yes'>
  <source>
    <address type='pci' domain='0' bus='11' slot='16' function='0'/>
  </source>
</interface>
```

**注記**

**MAC アドレスを指定しないと、MAC アドレスが自動的に生成されます。*<virtualport>* 要素は、802.11Qbh ハードウェアスイッチに接続する場合にのみ使用されます。*<vlan>* 要素は Red Hat Enterprise Linux 6.4 の新機能で、ゲストのデバイスを VLAN タグ付き *42* に透過的に配置します。**

**仮想マシンが起動すると、物理アダプターが提供するタイプのネットワークデバイスと、設定された MAC アドレスが表示されます。この MAC アドレスは、ホストおよびゲストの再起動後も変更されません。**

**以下の *<interface>* の例は、オプションの *<mac address>*、*<virtualport>*、および *<vlan>* 要素の構文を示しています。実際には、例に示すように、*<vlan>* 要素または *<virtualport>* 要素のいずれかを同時に使用しないでください。**

```
...
 <devices>
  ...
  <interface type='hostdev' managed='yes'>
   <source>
     <address type='pci' domain='0' bus='11' slot='16' function='0'/>
   </source>
   <mac address='52:54:00:6d:90:02'>
   <vlan>
      <tag id='42'/>
   </vlan>
   <virtualport type='802.1Qbh'>
     <parameters profileid='finance'/>
   </virtualport>
  </interface>
  ...
 </devices>
```

9. **仮想マシンへの仮想機能の追加**

**上の手順で作成した一時ファイルを使用し、次のコマンドを実行して仮想マシンに仮想機能を追加します。これにより、新しいデバイスがすぐに接続され、以降のゲストの再起動のために保存されます。**

```
virsh attach-device MyGuest /tmp/new-interface.xml  --config
```

**--config オプションを使用すると、今後ゲストを再起動しても新しいデバイスを使用できるようになります。**

仮想マシンが新しいネットワークインターフェイスカードを検出します。この新しいカードは、SR-IOV デバイスの仮想機能です。

## 13.3. SR-IOV のトラブルシューティング

本セクションは、SR-IOV に影響を与える可能性がある問題の解決策を記載します。

### ゲストの起動エラー

設定した仮想マシンを起動すると、次のようなエラーが発生します。

```
# virsh start test
error: Failed to start domain test
error: internal error unable to start guest: char device redirected to
/dev/pts/2
get_real_device: /sys/bus/pci/devices/0000:03:10.0/config: Permission denied
init_assigned_device: Error: Couldn't get real device (03:10.0)!
Failed to initialize assigned device host=03:10.0
```

このエラーは、多くの場合、別のゲストまたはホスト自体に割り当てられているデバイスが原因で発生します。

### ゲストの移行、保存、またはダンプのエラー

仮想マシンの移行およびダンプを試行すると、以下のようなエラーが発生します。

```
# virsh dump --crash 5 /tmp/vmcore
error: Failed to core dump domain 5 to /tmp/vmcore
error: internal error unable to execute QEMU command 'migrate': An undefined
error has occurred
```

デバイスの割り当ては、仮想マシンが起動した特定ホストのハードウェアを使用するため、デバイスの割り当てが使用中の場合は、ゲストの移行と保存に対応しません。現在では、ゲストのコアダンプにも同じ制限が適用されます。これは将来変更される可能性があります。

## 13.4. SR-IOV の制限

SR-IOV は、以下のデバイスでの徹底したテストのみが行われています。

- **Intel® 82576NS ギガビットイーサネットコントローラー (igb ドライバー)**
- **Intel® 82576EB ギガビットイーサネットコントローラー (igb ドライバー)**
- **Intel® 82599ES 10 ギガビットイーサネットコントローラー (ixgbe ドライバー)**
- **Intel® 82599EB 10 ギガビットイーサネットコントローラー (ixgbe ドライバー)**

**その他の SR-IOV デバイスは動作するかもしれませんが、リリース時にはテストされていません。**

# 第14章 KVM ゲストのタイミング管理

仮想化には、ゲスト仮想マシンでの時間管理に関するいくつかの課題が含まれます。

- 割り込みは、すべてのゲスト仮想マシンに常に同時に瞬時に配信されるとは限りません。これは、仮想マシンの割り込みは実際の割り込みではなく、ホストマシンによってゲスト仮想マシンに挿入されるためです。

- ホストは、別のゲスト仮想マシンを実行している場合や、別のプロセスの場合があります。ただし、割り込みで通常必要とされる正確なタイミングは、常に可能とは限りません。

正確な時刻管理が行われていないゲスト仮想マシンでは、セッションの有効性、移行、およびその他のネットワークアクティビティーがタイムスタンプに依存して正しい状態を維持するため、ネットワークアプリケーションとプロセスで問題が発生する可能性があります。

KVM は、ゲスト仮想マシンに準仮想化クロック (kvm-clock) を提供することで、この問題を回避します。ただし、ゲストの移行など、時間管理の不正確さによって影響を受ける可能性のあるアクティビティーを試行する前に、タイミングをテストすることは依然として重要です。

重要

上記の問題を回避するには、ホストとゲスト仮想マシンで、ネットワークタイムプロトコル (NTP) を設定する必要があります。Red Hat Enterprise Linux 6 以前を使用するゲストでは、NTP が ntpd サービスにより実装されます。詳細は、Red Hat Enterprise 6 Deployment Guide を参照してください。

## 14.1. CONSTANT タイムスタンプカウンター (TSC)

最新の Intel および AMD の CPU では、Constant TSC (Time Stamp Counter) が提供されます。省電力ポリシーに準拠するために、CPU コア自体の周波数が変更した場合、定数 TSC のカウント頻度は変わりません。TSC を KVM ゲストのクロックソースとして使用するには、Constant TSC 周波数の CPU が必要です。

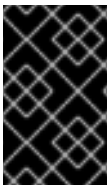
constant_tsc フラグが存在する場合、CPU には一定のタイムスタンプカウンターがあります。CPU に constant_tsc フラグがあるかどうかを確認するには、以下のコマンドを実行します。

```
$ cat /proc/cpuinfo | grep constant_tsc
```

出力があれば、CPU には constant_tsc ビットがあります。出力が表示されない場合は、以下の手順に従ってください。

#### 14.1.1. Constant タイムスタンプカウンターを使用しないホストの設定

TSC の周波数が一定でないシステムは、仮想マシンのクロックソースとして TSC を使用できないため、追加の設定が必要になります。電源管理機能は正確な時間管理を妨げるため、ゲスト仮想マシンが KVM で時間を正確に保持するには、無効にする必要があります。



重要

この手順は、AMD リビジョン F の CPU のみを対象としています。

CPU に constant_tsc ビットがない場合は、電源管理機能をすべて無効にします(BZ#513138)。各システムには、時間を維持するために使用するいくつかのタイマーがあります。TSC はホストで安定していません。これは、cpufreqの変更、ディープ C ステート、またはより高速な TSC を使用したホストへの移行が原因である場合があります。ディープ C のスリープ状態は、TSC を停止する可能性があります。ディープ C 状態を使用するカーネルを防ぐには、ホストの grub.conf ファイルのカーネル起動オプションに processor.max_cstate=1 を追加します。

```
title Red Hat Enterprise Linux (2.6.32-330.x86_64)
        root (hd0,0)
 kernel /vmlinuz-2.6.32-330.x86_64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet \
   processor.max_cstate=1
```

/etc/sysconfig/cpuspeed 設定ファイルを編集し、cpufreq を無効にし( constant_tscのないホストでのみ必要)、MIN_SPEED 変数および MAX_SPEED 変数を利用可能な最大頻度に変更します。有効な制限は、/sys/devices/system/cpu/cpu*/cpufreq/scaling_available_frequencies ファイルにあります。

### 14.2. RED HAT ENTERPRISE LINUX ゲストに必要なパラメーター

特定の Red Hat Enterprise Linux ゲスト仮想マシンでは、追加のカーネルパラメーターが必要です。このパラメーターは、ゲスト仮想マシンの /boot/grub/grub.conf ファイルの /kernel 行の末尾に追加することで設定できます。

以下の表は、Red Hat Enterprise Linux のバージョンと、指定したシステムで必要とされるパラメーターの一覧です。

**表14.1 カーネルパラメーターの要件**

| Red Hat Enterprise Linux のバージョン | ゲストカーネルの追加パラメーター |
|---|---|
| 7.0 以降 (AMD64 および Intel 64 システム、kvm-clock 使用) | 追加のパラメーターは必要ありません。 |
| 6.1 以降 (AMD64 および Intel 64 システム、kvm-clock 使用) | 追加のパラメーターは必要ありません。 |
| 6.0 (AMD64 および Intel 64 システム、kvm-clock 使用) | 追加のパラメーターは必要ありません。 |
| 6.0 (AMD64 および Intel 64 システム、kvm-clock 使用せず) | notsc lpj=*n* |
| 5.5 AMD64/Intel 64 (kvm-clock 使用) | 追加のパラメーターは必要ありません |
| kvm-clock を使用しない 5.5 AMD64 および Intel 64 システム | notsc lpj=*n* |
| 5.5 (32 ビット AMD および Intel システム、kvm-clock 使用) | 追加のパラメーターは必要ありません |
| kvm-clock のない 32 ビット AMD および Intel システムで 5.5 | clocksource=acpi_pm lpj=*n* |
| AMD64 および Intel 64 システムの場合：5.4 | notsc |
| 32 ビット AMD および Intel システムにおける 5.4 | clocksource=acpi_pm |
| AMD64 および Intel 64 システムの場合：5.3 | notsc |
| 32 ビット AMD および Intel システムの場合：5.3 | clocksource=acpi_pm |

**注記**

**_lpj_ パラメーターには、ゲスト仮想マシンが実行される特定の CPU の jiffy ごとのループ 値と同等の数値が必要です。この値が分からない場合は、_lpj_ パラメーターを設定しないでください。**

警告

以前は、*divider* カーネルパラメーターは、応答要件が高くない Red Hat Enterprise Linux 4 および 5 ゲスト仮想マシンに推奨されていました。また、ゲスト密度が高いシステムに存在していました。Red Hat Enterprise Linux 4 を実行しているゲストまたはバージョン 5.8 より前の Red Hat Enterprise Linux 5 バージョンでの使用は推奨されなくなりました。

*divider* タイマー割り込みの頻度を減らすことで、5.8 以上のバージョンの Red Hat Enterprise Linux 5 バージョンのスループットを向上させることができます。たとえば、HZ=1000、および *divider* が 10 ( divider=10)に設定されている場合、期間ごとのタイマー割り込みの数はデフォルト値(1000)から 100 に変更されます（デフォルト値の 1000 は divider 値 10 で除算されます）。

BZ#698842 *divider* パラメーターが割り込みおよびティックの記録と対話する方法にバグが詳述されています。このバグは、Red Hat Enterprise Linux 5.8 で修正されています。ただし、*divider* パラメーターは、バージョン 5.8 より前の Red Hat Enterprise Linux 4 または Red Hat Enterprise Linux 5 バージョンを使用するゲストでカーネルパニックを引き起こす可能性があります。

このパラメーターは Red Hat Enterprise Linux 3 では実装されていないため、このバグは Red Hat Enterprise Linux 3 ゲストには影響しません。

Red Hat Enterprise Linux 6 には固定頻度のクロック割り込みがありません。ティックレスモード で動作し、必要に応じてタイマーを動的に使用します。したがって、*divider* パラメーターは Red Hat Enterprise Linux 6 では役に立ち、Red Hat Enterprise Linux 6 ゲストはこのバグの影響を受けません。

## 14.3. WINDOWS SERVER 2008、WINDOWS SERVER 2008 R2、および WINDOWS 7 ゲストでの REAL-TIME クロックの使用

Windows は、RTC (Real-Time Clock)と Time Stamp Counter (TSC)の両方を使用します。Windows ゲスト仮想マシンでは、ゲストのタイミング問題を解決するすべてのタイムソースに TSC の代わりに Real-Time クロックを使用できます。

boot.ini ファイルは、Windows Server 2008 以降で使用されなくなりました。Windows Server 2008、Windows Server 2008 R2、および Windows 7 は、hypervisor-present ビットが設定されている場合、タイムソースとして TSC を使用しません。Red Hat Enterprise Linux 6 KVM ハイパーバイ

ザーは、デフォルトでこの CPUID ビットを有効にするため、Windows ブートパラメーターを変更するために Boot Configuration Data Editor (bcdedit.exe)を使用する必要がなくなりました。

**手順14.1 Windows Server 2008 R2 および Windows 7 ゲストでの Real-Time クロックの使用**

1. Windows ゲスト仮想マシンを開きます。

2. 起動 メニューの Accessories メニューを開きます。Command Prompt application を右クリックし、Run as Administrator を選択します。

3. プロンプトが表示されたら、セキュリティー例外を確認します。

4. プラットフォームクロックを使用するようにブートマネージャーを設定します。これにより、Windows がプライマリークロックソースに PM タイマーを使用するように指示されます。システムの UUID がデフォルトのブートデバイスと異なる場合は、システムの UUID （以下の例では*{default}* ）を変更する必要があります。

```
C:\Windows\system32>bcdedit /set {default} USEPLATFORMCLOCK on
The operation completed successfully
```

今回の修正により、Windows Server 2008 R2 および Windows 7 ゲストの時間の維持が改善されます。Windows 2008 (R2 以外)は *USEPLATFORMCLOCK* パラメーターをサポートしませんが、デフォルトで Real-Time クロックをすでに使用しています。

## 14.4. スチールタイムアカウンティング

スチールタイムは、ホストが提供していないゲスト仮想マシンが必要とする CPU 時間の量です。スチールタイムは、ホストがこれらのリソースを別の場所 (たとえば、別のゲスト) に割り当てるときに発生します。

スチールタイムは、/proc/stat の CPU 時間フィールドに *st* として報告されます。これは、top や vmstat などのユーティリティーにより自動的に報告され、オフにすることはできません。

スチールタイムが長くなると CPU 競合が発生するため、ゲストのパフォーマンスが低下する可能性があります。CPU の競合を軽減するには、ゲストの CPU 優先度または CPU クォータを上げるか、ホストで実行するゲストを減らします。

# 第15章 LIBVIRT でのネットワークブート

ゲストの仮想マシンは、PXE を有効にして起動できます。PXE を使用すると、ゲスト仮想マシンを起動して、ネットワーク自体から設定を読み込むことができます。本セクションでは、libvirt で PXE ゲストを設定する基本的な設定手順を説明します。

本セクションでは、ブートイメージまたは PXE サーバーの作成は説明しません。これは、プライベートネットワークまたはブリッジネットワークで libvirt を設定し、PXE ブートを有効にしてゲスト仮想マシンを起動する方法を説明するために使用されます。

**警告**

以下の手順は、例としてのみ提供されます。続行する前に、十分なバックアップがあることを確認してください。

## 15.1. ブートサーバーの準備

本章の手順を実行するには、以下が必要になります。

- PXE サーバー (DHCP および TFTP) - libvirt の内部サーバー、手動で設定した dhcpd および tftpd、dnsmasq、Cobbler が設定したサーバー、またはその他のサーバーを指します。
- ブートイメージ - たとえば、手動または Cobbler で設定した PXELINUX。

### 15.1.1. プライベートの libvirt ネットワークに PXE ブートサーバーを設定する

この例では、デフォルト のネットワークを使用します。以下の手順を実行します。

**手順15.1 PXE ブートサーバーの設定**

1. PXE ブートイメージおよび設定を /var/lib/tftp に配置します。

2. 以下のコマンドを実行します。

```
# virsh net-destroy default
# virsh net-edit default
```

3. デフォルト ネットワークの設定ファイルの <ip> 要素を編集して、適切なアドレス、ネットワークマスク、DHCP アドレス範囲、および起動ファイルを追加します。*BOOT_FILENAME* は、ゲスト仮想マシンの起動に使用するファイル名を表します。

```
<ip address='192.168.122.1' netmask='255.255.255.0'>
  <tftp root='/var/lib/tftp' />
  <dhcp>
    <range start='192.168.122.2' end='192.168.122.254' />
    <bootp file='BOOT_FILENAME' />
  </dhcp>
</ip>
```

4. PXE を使用してゲストを起動します (「PXE を使用したゲストの起動」 を参照)。

## 15.2. PXE を使用したゲストの起動

本セクションでは、PXE でゲスト仮想マシンを起動する方法を説明します。

### 15.2.1. ブリッジネットワークの使用

**手順15.2 PXE およびブリッジネットワークを使用したゲストの起動**

1. ネットワークで PXE ブートサーバーが使用できるように、ブリッジが有効になっていることを確認します。

2. PXE 起動が有効になっているゲスト仮想マシンを起動します。以下のコマンド例に示すように、virt-install コマンドを使用して、PXE ブートが有効になっている新しい仮想マシンを作成できます。

```
virt-install --pxe --network bridge=breth0 --prompt
```

または、以下の例のように、ゲストネットワークがブリッジネットワークを使用するように設定されており、XML ゲスト設定ファイルの <os> 要素内に <boot dev='network'/> 要素があることを確認します。

```
<os>
  <type arch='x86_64' machine='rhel6.2.0'>hvm</type>
  <boot dev='network'/>
  <boot dev='hd'/>
</os>
<interface type='bridge'>
  <mac address='52:54:00:5a:ad:cb'/>
  <source bridge='breth0'/>
  <target dev='vnet0'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

### 15.2.2. プライベートの libvirt ネットワークの使用

**手順15.3 プライベートの libvirt ネットワークの使用**

1. 「プライベートの libvirt ネットワークに PXE ブートサーバーを設定する」 に示すように、libvirt で PXE ブートを設定します。

2. PXE 起動を有効にして libvirt を使用してゲスト仮想マシンを起動します。virt-install コマンドを使用して、PXE を使用して新しい仮想マシンを作成/インストールできます。

```
virt-install --pxe --network network=default --prompt
```

または、以下の例のように、ゲストネットワークがブリッジネットワークを使用するように設定されており、XML ゲスト設定ファイルの <os> 要素内に <boot dev='network'/> 要素があることを確認します。

```
<os>
  <type arch='x86_64' machine='rhel6.2.0'>hvm</type>
  <boot dev='network'/>
  <boot dev='hd'/>
</os>
```

また、ゲスト仮想マシンがプライベートネットワークに接続されていることを確認します。

```
<interface type='network'>
  <mac address='52:54:00:66:79:14'/>
  <source network='default'/>
  <target dev='vnet0'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

# 第16章 ハイパーバイザーおよび仮想マシンの登録

Red Hat Enterprise Linux 6 および 7 では、すべてのゲスト仮想マシンが特定のハイパーバイザーにマッピングされ、すべてのゲストに同じレベルのサブスクリプションサービスが割り当てられるようにする必要があります。これを行うには、インストールされて登録されている各 KVM ハイパーバイザーで、すべてのゲスト仮想マシンを自動的に検出するサブスクリプションエージェントをインストールする必要があります。これにより、ホストにあるマッピングファイルが作成されます。このマッピングファイルにより、すべてのゲスト仮想マシンに次の利点があります。

- 仮想システムに固有のサブスクリプションは簡単に利用でき、関連するすべてのゲスト仮想マシンに適用できます。
- ハイパーバイザーから継承できるすべてのサブスクリプションのメリットは、すぐに利用でき、関連するすべてのゲスト仮想マシンに適用できます。

**注記**

本章で説明する情報は、Red Hat Enterprise Linux のサブスクリプションのみを対象としています。Red Hat Virtualization サブスクリプションまたは Red Hat Satellite サブスクリプションもある場合は、それらのサブスクリプションで提供される virt-who 情報も参照してください。Red Hat Subscription Management の詳細は、カスタマーポータルの Red Hat Subscription Management ガイドを参照してください。

## 16.1. ホストの物理マシンに VIRT-WHO をインストールする

1. **KVM ハイパーバイザーの登録**

   ホスト物理マシンの root ユーザーで、ターミナルで subscription-manager register [options] コマンドを実行して、KVM Hypervisor を登録します。# subscription-manager register --help メニューを使用すると、より多くのオプションを使用できます。ユーザー名とパスワードを使用している場合は、サブスクリプションマネージャーが認識する認証情報を使用します。初めてサブスクライブする際に、ユーザーアカウントがない場合は、カスタマーサポートにご連絡ください。たとえば、仮想マシンを 'admin' として登録 (パスワードは 'secret') するには、次のコマンドを送信します。

   ```
   [root@rhel-server ~]# subscription-manager register --username=admin --password=secret --auto-attach --type=hypervisor
   ```

2. **virt-who パッケージをインストールします。**

   ホストの物理マシンの root として以下のコマンドを実行して、virt-who パッケージをインストールします。

   ```
   [root@rhel-server ~]# yum install virt-who
   ```

3. **virt-who 設定ファイルの作成**

   **/etc/virt-who.d/ ディレクトリーに設定ファイルを追加します。ファイル名は何であるかは重要ではありませんが、意味のある名前を指定する必要があります。ファイルは /etc/virt-who.d/ ディレクトリーに置く必要があります。このファイル内に以下のスニペットを追加し、ファイルを閉じる前に必ず保存してください。**

```
[libvirt]
type=libvirt
```

4. **virt-who サービスを再起動します。**

   **ホストの物理マシンの root として以下のコマンドを実行して、virt-who サービスを起動します。**

```
[root@virt-who ~]# service virt-who start
[root@virt-who ~]# chkconfig virt-who on
```

5. **virt-who サービスがゲスト情報を受信していることを確認する**

   **この時点で、virt-who サービスは、ホストからドメインの一覧の収集を開始します。ホストの物理マシンの /var/log/rhsm/rhsm.log ファイルを確認して、ゲスト仮想マシンの一覧がファイルに含まれていることを確認します。たとえば、以下のようになります。**

```
2015-05-28 12:33:31,424 DEBUG: Libvirt domains found: [{'guestId': '58d59128-cfbb-4f2c-93de-230307db2ce0', 'attributes': {'active': 0, 'virtWhoType': 'libvirt', 'hypervisorType': 'QEMU'}, 'state': 5}]
```

**手順16.1 カスタマーポータルでのサブスクリプションの管理**

1. **ハイパーバイザーのサブスクライブ**

   **仮想マシンはハイパーバイザーと同じサブスクリプションのメリットを享受するため、ハイパーバイザーに有効なサブスクリプションがあり、仮想マシンがサブスクリプションを使用できることが重要です。**

   a. **カスタマーポータルにログインします。**

      **Red Hat カスタマーポータル https://access.redhat.com/ にログインし、ページ上部の サブスクリプション ボタンをクリックします。**

   b. **Systems リンクをクリックします。**

      **Subscriber Inventory セクションで（ページの下部に移動する）、Systems リンクをクリックします。**

   c. **ハイパーバイザーを選択します。**

Systems ページには、サブスクライブしているすべてのシステムの表があります。ハイパーバイザーの名前をクリックします（例：localhost.localdomain）。表示された詳細ページで Attach a subscription をクリックし、一覧表示されたすべてのサブスクリプションを選択します。Attach Selected をクリックします。これにより、ホストの物理サブスクリプションがハイパーバイザーに割り当てられ、ゲストがサブスクリプションを利用できるようになります。

2. ゲスト仮想マシンのサブスクライブ - 初めての使用

この手順は、新しいサブスクリプションを持っていて、これまでゲスト仮想マシンをサブスクライブしたことがない人を対象としています。仮想マシンを追加する場合は、この手順をスキップします。virt-who サービスを実行しているマシンで、ハイパーバイザープロファイルに割り当てられたサブスクリプションを使用するには、ターミナルで root としてゲスト仮想マシンで以下のコマンドを実行し、自動サブスクライブします。

```
[root@virt-who ~]# subscription-manager attach --auto
```

3. 追加のゲスト仮想マシンのサブスクライブ

を初めてサブスクライブする場合は、この手順を省略します。仮想マシンを追加する場合は、このコマンドを実行しても、必ずしも同じサブスクリプションをゲスト仮想マシンに再アタッチするわけではないことに注意してください。これは、すべてのサブスクリプションを削除してから自動割り当てを有効にして、特定のゲスト仮想マシンに必要なものを解決できるようにすると、以前とは異なるサブスクリプションが使用される可能性があるためです。これはシステムには影響を及ぼさない可能性がありますが、注意してください。以下で説明されていない手動の接続手順を使用して仮想マシンを接続した場合は、自動接続が機能しないため、これらの仮想マシンを手動で再接続する必要があります。ターミナルの root で以下のコマンドを使用して、最初に古いゲストのサブスクリプションを削除してから、自動アタッチを使用してすべてのゲストにサブスクリプションを割り当てます。ゲスト仮想マシンで次のコマンドを実行します。

```
[root@virt-who ~]# subscription-manager remove --all
[root@virt-who ~]# subscription-manager attach --auto
```

4. サブスクリプションが割り当てられていることを確認する

ゲスト仮想マシンのターミナルで root として以下のコマンドを実行して、サブスクリプションがハイパーバイザーに割り当てられていることを確認します。

```
[root@virt-who ~]# subscription-manager list --consumed
```

以下のような出力が表示されます。サブスクリプションの詳細に注意してください。'Subscription is current' と表示されるはずです。

```
[root@virt-who ~]#  subscription-manager list --consumed
+-------------------------------------------+
   Consumed Subscriptions
```

```
+-------------------------------------------+
Subscription Name: Awesome OS with unlimited virtual guests
Provides:   Awesome OS Server Bits
SKU:    awesomeos-virt-unlimited
Contract:   0
Account:   ######### Your account number #####
Serial:   ######### Your serial number ######
Pool ID:   XYZ123
Provides Management:  No
Active:   True
Quantity Used:   1
Service Level:
Service Type:
Status Details:  Subscription is current
Subscription Type:
Starts:   01/01/2015
Ends:    12/31/2015
System Type:   Virtual
```

**???**

**The ID for the subscription to attach to the system is displayed here. You will need this ID if you need to attach the subscription manually.**

**???**

**Indicates if your subscription is current. If your subscription is not current, an error message appears. One example is Guest has not been reported on any host and is using a temporary unmapped guest subscription. In this case the guest needs to be subscribed. In other cases, use the information as indicated in 「サブスクリプションステータスエラーが表示された場合の対応」.**

5. **追加のゲストを登録**

**ハイパーバイザーに新しいゲスト仮想マシンをインストールする場合は、ゲスト仮想マシンを登録し、ハイパーバイザーに割り当てられたサブスクリプションを使用する必要があります。ターミナルで以下のコマンドを実行して、ゲスト仮想マシンの root として実行します。**

```
[root@server1 ~]# subscription-manager register
[root@server1 ~]# subscription-manager attach --auto
[root@server1 ~]# subscription-manager list --consumed
```

### 16.2. 新しいゲスト仮想マシンの登録

**登録済みで実行中のホストで、新しいゲスト仮想マシンを作成する場合は、virt-who サービスも実行している必要があります。これにより、virt-who サービスがゲストをハイパーバイザーにマップできる**

ため、システムが仮想システムとして適切に登録されます。仮想マシンを登録するには、ターミナルで root で以下のコマンドを実行します。

```
[root@virt-server ~]# subscription-manager register --username=admin --password=secret --auto-attach
```

### 16.3. ゲスト仮想マシンエントリーの削除

ゲスト仮想マシンが実行されている場合は、ターミナルウィンドウでゲストの root として次のコマンドを実行して、システムの登録を解除します。

```
[root@virt-guest ~]# subscription-manager unregister
```

ただし、システムが削除された場合、仮想サービスは、サービスが削除されたか、一時停止したかを認識できません。この場合は、以下の手順に従って、サーバー側からシステムを手動で削除する必要があります。

1. サブスクリプションマネージャーへのログイン

   サブスクリプションマネージャーは、Red Hat カスタマーポータルにあります。画面最上部にあるログインアイコンをクリックし、ユーザー名とパスワードを使用してカスタマーポータルにログインします。

2. Subscriptions タブをクリックします。

   Subscriptions タブをクリックします。

3. Systems リンクをクリックします。

   ページを下にスクロールして、Systems をクリックします。

4. システムを削除します。

   システムプロファイルを削除するには、テーブルで指定したシステムのプロファイルを検索し、その名前の横にあるチェックボックスを選択して、Delete をクリックします。

### 16.4. 手動での VIRT-WHO のインストール

本セクションでは、ハイパーバイザーが提供するサブスクリプションを手動で割り当てる方法を説明します。

**手順16.2 サブスクリプションを手動で割り当てる方法**

1. **サブスクリプション情報の一覧を表示し、プール ID を見つけます。**

   **最初に、仮想タイプで利用可能なサブスクリプションを一覧表示する必要があります。root としてターミナルで次のコマンドを実行します。**

```
[root@server1 ~]# subscription-manager list --avail --match-installed | grep 'Virtual' -B12
Subscription Name: Red Hat Enterprise Linux ES (Basic for Virtualization)
Provides:          Red Hat Beta
                   Oracle Java (for RHEL Server)
                   Red Hat Enterprise Linux Server
SKU:               -------
Pool ID:           XYZ123
Available:         40
Suggested:         1
Service Level:     Basic
Service Type:      L1-L3
Multi-Entitlement: No
Ends:              01/02/2017
System Type:       Virtual
```

   **表示されたプール ID をメモします。この ID は、次の手順で必要に応じてコピーします。**

2. **プール ID を使用したサブスクリプションの割り当て**

   **前の手順でコピーしたプール ID を使用して、attach コマンドを実行します。プール ID XYZ123 を、取得したプール ID に置き換えます。root としてターミナルで次のコマンドを実行します。**

```
[root@server1 ~]# subscription-manager attach --pool=XYZ123

Successfully attached a subscription for: Red Hat Enterprise Linux ES (Basic for
Virtualization)
```

### 16.5. VIRT-WHO のトラブルシューティング

**本セクションでは、virt-who のトラブルシューティングに関する情報を提供します。**

#### 16.5.1. ハイパーバイザーのステータスが赤である理由

**シナリオ：サーバー側で、サブスクリプションを持たないハイパーバイザーにゲストをデプロイします。24 時間後、ハイパーバイザーはステータスを赤で表示します。この状況を改善するには、そのハイパーバイザーのサブスクリプションを取得する必要があります。または、サブスクリプションを使用してゲストをハイパーバイザーに永続的に移行します。**

#### 16.5.2. サブスクリプションステータスエラーが表示された場合の対応

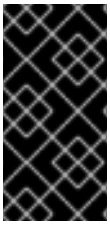
シナリオ：以下のエラーメッセージのいずれかが表示されます。

- System not properly subscribed
- Status unknown
- Late binding of a guest to a hypervisor through virt-who (host/guest mapping)

エラーの理由を調べるには、/var/log/rhsm/ ディレクトリーにある rhsm.log という名前の virt-who ログファイルを開きます。

# 付録A NETKVM ドライバーパラメーター

NetKVM ドライバーをインストールした後は、ご使用の環境に応じて設定を行うことができます。このセクションに記載するパラメーターは、Windows デバイスマネージャー (devmgmt.msc)で設定できます。



重要

ドライバーのパラメーターを変更すると、Windows はそのドライバーを再ロードします。これにより、既存のネットワークアクティビティーが中断します。

**手順A.1 NetKVM パラメーターの設定**

1. オープン デバイスマネージャー

   Start ボタンをクリックします。右側のペインで、Computer を右クリックし、Manage をクリックします。プロンプトが表示されたら、User Account Control ウィンドウで Continue をクリックします。これにより、Computer Management ウィンドウが開きます。

   Computer Management ウィンドウの左側のペインで、Device Manager をクリックします。

2. 正しいデバイスの特定

   Computer Management ウィンドウの中央ペインで、Network アダプター の横にある + 記号をクリックします。

   Red Hat VirtIO Ethernet Adapter デバイスのリストの下で、NetKVM をダブルクリックします。これにより、そのデバイスの Properties ウィンドウが開きます。

3. デバイスパラメーターの表示

   Properties ウィンドウで、Advanced タブをクリックします。

4. デバイスパラメーターの変更

   変更するパラメーターをクリックして、そのパラメーターのオプションを表示します。

   必要に応じてオプションを変更し、OK をクリックして変更を保存します。

## A.1. NETKVM の設定可能なパラメーター

### ロギングパラメーター

#### logging.Enable

ロギングが有効であるかどうかを決定するブール値。デフォルト値は 1 （有効）です。

#### logging.Level

ロギングレベルを定義する整数。この整数を高くすると、ログの詳細度が上がります。デフォルト値は 0 （エラーのみ）です。1-2 は設定メッセージを追加します。3-4 は、パケットフロー情報を追加します。5-6 は割り込みおよび DPC レベルのトレース情報を追加します。

重要

ロギングレベルが高くなると、ゲスト仮想マシンの速度が低下します。

#### Logging.Statistics(sec)

ログ統計を出力するかどうか、および各定期的な統計の出力の間隔（秒単位）を定義する整数。デフォルト値は 0 （ロギング統計なし）です。

### 初期パラメーター

#### Assign MAC

準仮想化 NIC のローカル管理 MAC アドレスを定義する文字列。これはデフォルトでは設定されません。

#### Init.ConnectionRate(Mb)

接続レートをメガバイト単位で表す整数。Windows 2008 以降のデフォルト値は 10000 です。

#### Init.Do802.1PQ

Priority/VLAN タグポピュレーションと削除サポートを有効にするブール値。デフォルト値は 1 （有効）です。

#### Init.UseMergedBuffers

マージ可能な RX バッファーを有効にするブール値。デフォルト値は 1 （有効）です。

**Init.UsePublishEvents**

公開されたイベントの使用を有効にするブール値。デフォルト値は 1 （有効）です。

**Init.MTUSize**

最大伝送単位 (MTU) を定義する整数。デフォルト値は 1500 です。500 から 65500 までの値はすべて受け入れ可能です。

**Init.IndirectTx**

間接リング記述子が使用されているかどうかを制御します。デフォルト値は Disable で、間接リング記述子の使用を無効にします。他の有効な値は Enable で、間接リング記述子の使用を有効にします。また、間接リング記述子の条件付き使用を可能にする Enable* です。

**Init.MaxTxBuffers**

割り当てられる TX リング記述子の量を表す整数。デフォルト値は 1024 です。有効な値は、16、32、64、128、256、512、または 1024 です。

**Init.MaxRxBuffers**

割り当てられる RX リング記述子の量を表す整数。デフォルト値は 256 です。有効な値は、16、32、64、128、256、512、または 1024 です。

**Offload.Tx.Checksum**

TX チェックサムオフロードモードを指定します。

Red Hat Enterprise Linux 6.4 以降では、このパラメーターの有効な値は All （デフォルト）です。このパラメーターは、IPv4 と IPv6 の両方で IP、TCP、および UDP チェックサムオフロードを有効にする All （デフォルト）です。TCP/UDP (v4,v 6)は、IPv4 と IPv6 の両方で TCP および UDP チェックサムオフロードを有効にする TCP/UDP (v 4)です。 IPv4 でのみ TCP および UDP チェックサムオフロードを有効にし、IPv4 でのみ TCP チェックサムオフロードのみを有効にする TCP (v4) を有効にします。

Red Hat Enterprise Linux 6.3 以前では、このパラメーターの有効な値は TCP/UDP （デフォルト値）で、TCP および UDP チェックサムオフロードを有効にします。TCP チェックサムオフロードのみを有効にする TCP、または TX チェックサムオフロードを無効にする TCP です。

**Offload.Tx.LSO**

TX TCP Large Segment Offload (LSO)を有効にするブール値。デフォルト値は 1 （有効）です。

**Offload.Rx.Checksum**

RX チェックサムオフロードモードを指定します。

Red Hat Enterprise Linux 6.4 以降では、このパラメーターの有効な値は All （デフォルト）です。このパラメーターは、IPv4 と IPv6 の両方で IP、TCP、および UDP チェックサムオフロードを有効にする All （デフォルト）です。TCP/UDP (v4,v 6)は、IPv4 と IPv6 の両方で TCP および UDP チェックサムオフロードを有効にする TCP/UDP (v 4)です。 IPv4 でのみ TCP および UDP チェックサムオフロードを有効にし、IPv4 でのみ TCP チェックサムオフロードのみを有効にする TCP (v4) を有効にします。

Red Hat Enterprise Linux 6.3 以前では、有効な値は Disable （デフォルト）で、RX チェックサムオフロードが無効になります。TCP、UDP、および IP チェックサムオフロードを有効にする すべて。TCP および UDP チェックサムオフロードを有効にする TCP/UDP、TCP チェックサムオフロードのみを有効にする TCP/UDP です。

テストおよびデバッグパラメーター

重要

test および debug パラメーターは、テストまたはデバッグにのみ使用してください。これらは実稼働環境では使用しないでください。

**TestOnly.DelayConnect(ms)**

起動時に接続を遅延させる期間（ミリ秒単位）。デフォルト値は 0 です。

**TestOnly.DPCChecking**

DPC チェックモードを設定します。0 （デフォルト）は DPC チェックを無効にします。1 は DPC チェックを有効にします。各ハングテストは DPC アクティビティーを検証し、DPC が生成されたかのように動作します。2 はデバイスの割り込みステータスをクリアし、それ以外の場合は 1 と同じです。

**TestOnly.Scatter-Gather**

scatter-gather 機能を有効にするかどうかを決定するブール値。デフォルト値は 1 （有効）です。この値を 0 に設定すると、scatter-gather 機能およびすべての依存機能が無効になります。

**TestOnly.InterruptRecovery**

割り込みリカバリーが有効になっているかどうかを決定するブール値。デフォルト値は 1 （有効）です。

**TestOnly.PacketFilter**

パケットフィルターリングが有効であるかどうかを決定するブール値。デフォルト値は 1 （有効）です。

**TestOnly.BatchReceive**

パケットがバッチで受信されるか、単数的に受信されるかどうかを決定するブール値。デフォルト値は 1 で、バッチ処理されたパケット受信を有効にします。

**TestOnly.Promiscuous**

プロミスキャスモードが有効になっているかどうかを決定するブール値。デフォルトの値は 0 (無効) です。

**TestOnly.AnalyzeIPPackets**

送信 IP パケットのチェックサムフィールドがテストされ、デバッグ目的で検証されるかどうかを決定するブール値。デフォルト値は 0 （チェックなし）です。

**TestOnly.RXThrottle**

単一の DPC で処理される受信パケットの数を決定する整数。デフォルト値は 1000 です。

**TestOnly.UseSwTxChecksum**

ハードウェアのチェックサムが有効かどうかを決定するブール値。デフォルトの値は 0 (無効) です。

# 付録B 一般的なLIBVIRTエラーとトラブルシューティング

この付録では、一般的な libvirt 関連の問題とエラー、およびそれらに対処するための手順について説明します。

トラブルシューティングの詳細は、以下の表でエラーを特定し、Solution の対応するリンクを参照してください。

**表B.1 一般的なlibvirtエラー**

| Error | 問題の説明 | ソリューション |
|---|---|---|
| **libvirtd Failed to Start** | libvirt デーモンの起動に失敗しました。ただし、**/var/log/messages** にはこのエラーに関する情報はありません。 | 「libvirtd が起動しない」 |
| **Cannot read CA certificate** | これは、URI がハイパーバイザーに接続できない場合に発生するいくつかのエラーのいずれかになります。 | 「URI のハイパーバイザー接続に失敗する」 |
| **Failed to connect socket ... : Permission denied** | これは、URI がハイパーバイザーに接続できない場合に発生するいくつかのエラーのいずれかになります。 | 「URI のハイパーバイザー接続に失敗する」 |
| その他の接続エラー | これは、URI がハイパーバイザーに接続できない場合に発生するその他のエラーです。 | 「URI のハイパーバイザー接続に失敗する」 |
| **Internal error guest CPU is not compatible with host CPU** | ホストとゲストプロセッサーが異なるため、ゲスト仮想マシンを開始できません。 | 「ゲスト仮想マシンを起動できません。 **internal error guest CPU is not compatible with host CPU**」 |
| **Failed to create domain from vm.xml error: monitor socket did not show up.: Connection refused** | ゲスト仮想マシン（またはドメイン）の起動に失敗し、このエラーまたは同様のエラーが返されます。 | 「ゲストの起動は、以下のエラーで失敗します。 **monitor socket did not show up**」 |
| **Internal error cannot find character device (null)** | このエラーは、ゲストのコンソールを接続しようとすると発生する可能性があります。ゲスト仮想マシンにシリアルコンソールが設定されていないことが報告されます。 | 「**Internal error cannot find character device (null)**」 |

| Error | 問題の説明 | ソリューション |
| --- | --- | --- |
| **No boot device** | 既存のディスクイメージからゲスト仮想マシンを構築した後、ゲストの起動が停止します。ただし、ゲストは **QEMU** コマンドを直接使用して正常に起動できます。 | 「ゲスト仮想マシンの停止をエラーで起動します。 **No boot device**」 |
| **The virtual network *"default"* has not been started** | *デフォルト* のネットワーク（またはその他のローカルで作成されたネットワーク）が起動できない場合、その接続にそのネットワークを使用するように設定された仮想マシンも起動に失敗します。 | 「仮想ネットワークの *デフォルト* は開始されていません」 |
| ゲスト上の PXE ブート (または DHCP ) が失敗 | ゲスト仮想マシンは正常に起動しますが、DHCP、PXE プロトコルを使用した起動、またはその両方から IP アドレスを取得することはできません。これは多くの場合、ブリッジの転送遅延時間が長く設定されているか、iptables パッケージとカーネルがチェックサムの難号化ルールをサポートしていない場合に発生します。 | 「ゲスト上の PXE ブート (または DHCP ) が失敗」 |
| ゲストは外部ネットワークにアクセスできるが、macvtap インターフェイスの使用時にはホストにアクセスできない | ゲストは他のゲストと通信できますが、macvtap （または ***type='direct'***）ネットワークインターフェイスを使用するように設定するとホストマシンに接続できません。<br><br>これは、実際にはエラーではなく、macvtap の定義済みの動作です。 | 「ゲストは外部ネットワークに到達できるが、macvtap インターフェイスの使用時にはホストにアクセスできない」 |
| **Could not add rule to fixup DHCP response checksums on network *'default'*** | この警告メッセージはほぼ常に無害ですが、間違って問題の証拠と見なされることがよくあります。 | 「Could not add rule to fixup DHCP response checksums on network *'default'*」 |
| **Unable to add bridge br0 port vnet0: No such device** | このエラーメッセージまたは同様の **Failed to add tap interface to bridge *'br0'*: No such device** は、ゲストの（またはドメインの） **<interface>** 定義で指定されたブリッジデバイスが存在しないことを示しています。 | 「Unable to add bridge br0 port vnet0: No such device」 |

| Error | 問題の説明 | ソリューション |
| --- | --- | --- |
| **Warning: could not open /dev/net/tun: no virtual network emulation qemu-kvm: -netdev tap,script=/etc/my-qemu-ifup,id=hostnet0: Device 'tap' could not be initialized** | ホストシステムに **type='ethernet'**（または generic ethernet）インターフェイスを設定した後は、ゲスト仮想マシンが起動しません。このエラーは、**libvirtd.log**、**/var/log/libvirt/qemu/*name_of_guest*.log**、またはその両方で表示されます。 | 「ゲストが Unable to Start with Error: **warning: could not open /dev/net/tun**」 |
| **Unable to resolve address *name_of_host* service '49155': Name or service not known** | QEMU ゲストの移行が失敗し、このエラーメッセージが見慣れないホスト名で表示されます。 | 「で移行に失敗する **Error: unable to resolve address**」 |
| **Unable to allow access for disk path /var/lib/libvirt/images/qemu.img: No such file or directory** | libvirt がディスクイメージにアクセスできないため、ゲスト仮想マシンを移行できません。 | 「**Unable to allow access for disk path: No such file or directory**で移行に失敗」 |
| libvirtd の開始時に存在するゲスト仮想マシンがない | libvirt デーモンは正常に起動しますが、**virsh list --all** の実行時にゲスト仮想マシンが存在しないように見えます。 | 「libvirtd の開始時に存在するゲスト仮想マシンがない」 |
| **Unable to connect to server at 'host:16509': Connection refused ... error: failed to connect to the hypervisor** | libvirtd は接続のために TCP ポートをリッスンする必要がありますが、ハイパーバイザーへの接続に失敗します。 | 「Unable to connect to server at 'host:16509': Connection refused ... error: failed to connect to the hypervisor」 |
| 一般的な XML エラー | **libvirt** は、XML ドキュメントを使用して構造化データを保存します。XML ドキュメントが、API を介して**libvirt** に渡されると、いくつかの一般的なエラーが発生します。このエントリーでは、ゲスト XML 定義の編集方法と、XML 構文および設定における一般的なエラーの詳細を説明します。 | 「一般的な XML エラー」 |

### B.1. LIBVIRTD が起動しない

**現象**

**libvirt デーモンが自動的に起動しない。libvirt デーモンの手動による起動も失敗。**

```
# /etc/init.d/libvirtd start
```

```
* Caching service dependencies ...                                    [ ok ]
* Starting libvirtd ...
/usr/sbin/libvirtd: error: Unable to initialize network sockets. Check /var/log/messages or run
without --daemon for more info.
* start-stop-daemon: failed to start `/usr/sbin/libvirtd'             [ !! ]
* ERROR: libvirtd failed to start
```

**また、/var/log/messages ではこのエラーに関する'more info'はありません。**

**調査**

**以下の行のコメントを解除して、/etc/libvirt/libvirtd.conf で libvirt のロギングを変更します。行のコメントを解除しるには、テキストエディターで /etc/libvirt/libvirtd.conf ファイルを開き、以下の行の先頭からハッシュ（または #）記号を削除して変更を保存します。**

```
log_outputs="3:syslog:libvirtd"
```

**注記**

**この行は、libvirt が過剰なログメッセージを作成しないように、デフォルトではコメントアウトされています。問題を診断したら、/etc/libvirt/libvirtd.conf ファイルーでこの行を再度コメント入力することが推奨されます。**

**libvirt を再起動し、問題が解決されたかどうかを確認します。**

**それでも libvirtd が正常に起動しない場合は、以下のようなエラーが /var/log/messages ファイルに表示されます。**

```
Feb  6 17:22:09 bart libvirtd: 17576: info : libvirt version: 0.9.9
Feb  6 17:22:09 bart libvirtd: 17576: error : virNetTLSContextCheckCertFile:92: Cannot read
CA certificate '/etc/pki/CA/cacert.pem': No such file or directory
Feb  6 17:22:09 bart /etc/init.d/libvirtd[17573]: start-stop-daemon: failed to start
`/usr/sbin/libvirtd'
Feb  6 17:22:09 bart /etc/init.d/libvirtd[17565]: ERROR: libvirtd failed to start
```

**libvirtd の man ページでは、libvirt を Listen for TCP/IP connections モードで実行すると、足りない cacert.pem ファイルが TLS 認証局として使用されることを示しています。これは、*--listen* パラメーターが渡されることを意味します。**

**ソリューション**

libvirt デーモンを以下のいずれかの方法で設定します。

- CA 証明書をインストールする。

  注記

  CA 証明書およびシステム認証の設定に関する詳細は、『Red Hat Enterprise Linux 6 デプロイメントガイド』 の 認証の設定 の章を参照してください。

- TLS は使用せずに TCP を使用してください。/etc/libvirt/libvirtd.conf で、listen_tls = 0 および listen_tcp = 1 を設定します。デフォルト値は listen_tls = 1 および listen_tcp = 0 です。

- *--listen* パラメーターは渡さないでください。/etc/sysconfig/libvirtd.conf で、*LIBVIRTD_ARGS* 変数を変更します。

## B.2. URI のハイパーバイザー接続に失敗する

サーバーへの接続時に、いくつかの異なるエラーが発生する可能性があります (virshを実行する場合など)。

### B.2.1. Cannot read CA certificate

**現象**

コマンドの実行中に、以下のエラー (または同様のエラー) が表示される。

```
$ virsh -c name_of_uri list
error: Cannot read CA certificate '/etc/pki/CA/cacert.pem': No such file or directory
error: failed to connect to the hypervisor
```

**調査**

このエラーメッセージは、実際の原因に関して誤解を招くものです。このエラーは、誤って指定された URI や未設定の接続など様々な要素によって起こり得ます。

**ソリューション**

誤って指定された URI

接続 URI として *qemu:///system* または *qemu:///session* を指定すると、virsh はそれぞれホスト名 *system* または *session* への接続を試みます。これは、virsh が、2 番目のスラッシュの後のテキストをホストとして認識するためです。

前方スラッシュを 3 つ使用してローカルホストに接続します。たとえば、*qemu:///system* を指定すると、virsh がローカルホストの libvirtd の *system* インスタンスに接続するように指示します。

ホスト名が指定されていると、QEMU トランスポートがデフォルトで TLS に設定されます。これは証明書になります。

接続が未設定

URI は適切ですが (例: *qemu[+tls]://server/system*)、証明書がマシンで適切に設定されていません。TLS の設定に関する詳細は、libvirt の Web サイトで利用可能な TLS の libvirt の設定 を参照してください。

### B.2.2. Failed to connect socket ... : Permission denied

現象

virsh コマンドを実行すると、以下のエラー（または同様のもの）が表示されます。

```
$ virsh -c qemu:///system list
error: Failed to connect socket to '/var/run/libvirt/libvirt-sock': Permission denied
error: failed to connect to the hypervisor
```

調査

ホスト名が指定されていない場合、QEMU への接続はデフォルトで UNIX ソケットを使用します。このコマンドを root として実行したエラーがない場合は、/etc/libvirt/libvirtd.conf の UNIX ソケットオプションが誤って設定されている可能性があります。

解決方法

UNIX ソケットを使用して非 root ユーザーとして接続するには、/etc/libvirt/libvirtd.conf で以下のオプションを設定します。

```
unix_sock_group = <group>
unix_sock_ro_perms = <perms>
unix_sock_rw_perms = <perms>
```

注記

virsh を実行しているユーザーは、unix_sock_group オプションで指定した グループ のメンバーである必要があります。

### B.2.3. その他の接続エラー

**Unable to connect to server at server:port: Connection refused**

デーモンはサーバー上で実行されていないか、設定オプション listen_tcp または listen_tls を使用してリッスンしないように設定されています。

**データの読み取り時のファイルの終了：nc: using stream socket: Input/output error**

ssh トランスポートを指定した場合、デーモンはサーバー上で実行されない可能性があります。このエラーを解決するには、デーモンがサーバーで実行していることを確認します。

## B.3. ゲスト仮想マシンを起動できません。 INTERNAL ERROR GUEST CPU IS NOT COMPATIBLE WITH HOST CPU

**現象**

Intel Core i7 プロセッサーで実行する( virt-manager は Nehalem、または Penrynと呼ばれる古い Core 2 Duo)は、virt-manager を使用して KVM ゲスト（またはドメイン）を作成します。インストール後に、ゲストのプロセッサーがホストの CPU に一致するように変更されます。その後、ゲストは起動できず、以下のエラーを報告します。

```
2012-02-06 17:49:15.985+0000: 20757: error : qemuBuildCpuArgStr:3565 : internal error guest CPU is not compatible with host CPU
```

さらに、virt-manager で Copy host CPU 設定 をクリックすると、Nehalem または Penryn ではなく Pentium が表示されます。

**調査**

/usr/share/libvirt/cpu_map.xml ファイルには、各 CPU モデルを定義するフラグが一覧表示されます。Nehalem および Penryn 定義には、以下が含まれます。

```
<feature name='nx'/>
```

その結果、NX （または No eXecute）フラグを指定して、CPU を Nehalem または Penryn として識別する必要があります。ただし、/proc/cpuinfo にはこのフラグがありません。

**解決方法**

新しい BIOS のほとんどは、No eXecute ビットを有効または無効にできます。ただし、無効にすると、一部の CPU はこのフラグを報告しないため、libvirt は別の CPU を検出します。この機能を有効にすると、libvirt に正しい CPU を報告するように指示します。この件に関する詳細は、ハードウェアのドキュメントを参照してください。

## B.4. ゲストの起動は、以下のエラーで失敗します。 MONITOR SOCKET DID NOT SHOW UP

**現象**

ゲスト仮想マシン（またはドメイン）の起動は、このエラー（または同様のもの）で失敗します。

```
# virsh -c qemu:///system create name_of_guest.xml error: Failed to create domain from
name_of_guest.xml error: monitor socket did not show up.: Connection refused
```

**調査**

このエラーメッセージには、以下が表示されます。

1. libvirt が動作している。
2. QEMU プロセスは起動に失敗し、
3. QEMU または QEMU エージェントモニターソケットへの接続を試みると、libvirt は終了します。

エラーの詳細を理解するには、ゲストログを確認します。

```
# cat /var/log/libvirt/qemu/name_of_guest.log
LC_ALL=C PATH=/sbin:/usr/sbin:/bin:/usr/bin QEMU_AUDIO_DRV=none /usr/bin/qemu-kvm -S -
M pc -enable-kvm -m 768 -smp 1,sockets=1,cores=1,threads=1 -name name_of_guest -uuid
ebfaadbe-e908-ba92-fdb8-3fa2db557a42 -nodefaults -chardev
```

```
socket,id=monitor,path=/var/lib/libvirt/qemu/name_of_guest.monitor,server,nowait -mon
chardev=monitor,mode=readline -no-reboot -boot c -kernel /var/lib/libvirt/boot/vmlinuz -initrd
/var/lib/libvirt/boot/initrd.img -append
method=http://www.example.com/pub/product/release/version/x86_64/os/ -drive
file=/var/lib/libvirt/images/name_of_guest.img,if=none,id=drive-ide0-0-0,boot=on -device ide-
drive,bus=ide.0,unit=0,drive=drive-ide0-0-0,id=ide0-0-0 -device virtio-net-
pci,vlan=0,id=net0,mac=52:40:00:f4:f1:0a,bus=pci.0,addr=0x4 -net
tap,fd=42,vlan=0,name=hostnet0 -chardev pty,id=serial0 -device isa-serial,chardev=serial0 -usb -
vnc 127.0.0.1:0 -k en-gb -vga cirrus -device virtio-balloon-pci,id=balloon0,bus=pci.0,
addr=0x3
char device redirected to /dev/pts/1
qemu: could not load kernel '/var/lib/libvirt/boot/vmlinuz':
Permission denied
```

**解決方法**

**ゲストログには、エラーを修正するために必要な詳細が含まれます。**

**ゲストが 0.9.5 よりも前の libvirt バージョンを実行している間にホストがシャットダウンした場合、libvirt-guest の init スクリプトはゲストの管理対象保存の実行を試みます。管理保存が不完全であった場合（たとえば、管理保存イメージがディスクにフラッシュされる前に電力が失われた場合など）、保存イメージが破損し、QEMU によって読み込まれません。古いバージョンの libvirt は破損を認識しないため、問題を許容します。この場合、ゲストログで -incoming を引数の 1 つとして使用しようとします。つまり、libvirt が保存された状態ファイルで移行して QEMU を起動しようとします。**

**この問題は、virsh managedsave-remove *name_of_guest* を実行して、破損した管理保存イメージを削除することで修正できます。新しいバージョンの libvirt では、最初の場所で破損を回避し、virsh start --force-boot *name_of_guest* を追加して管理保存イメージをバイパスします。**

## B.5. INTERNAL ERROR CANNOT FIND CHARACTER DEVICE (NULL)

**現象**

**ゲスト仮想マシンのコンソールへの接続を試みると、このエラーメッセージが表示されます。**

```
# virsh console test2 Connected to domain test2 Escape character is ^] error: internal error cannot
find character device (null)
```

**調査**

**このエラーメッセージは、ゲスト仮想マシンにシリアルコンソールが設定されていないことを示しています。**

**解決方法**

ゲストの XML ファイルでシリアルコンソールを設定します。

**手順B.1 ゲストの XML でのシリアルコンソールの設定**

1. virsh edit を使用して、次の XML をゲスト仮想マシンの XML に追加します。

```
<serial type='pty'>
  <target port='0'/>
</serial>
<console type='pty'>
  <target type='serial' port='0'/>
</console>
```

2. ゲストカーネルコマンドラインでコンソールを設定します。

これを行うには、ゲスト仮想マシンにログインして /boot/grub/grub.conf ファイルを直接編集するか、virt-edit コマンドラインツールを使用します。ゲストカーネルコマンドラインに以下を追加します。

```
console=ttyS0,115200
```

3. 以下のコマンドを実行します。

```
# virsh start vm && virsh console vm
```

## B.6. ゲスト仮想マシンの停止をエラーで起動します。 NO BOOT DEVICE

**現象**

既存のディスクイメージからゲスト仮想マシンを構築すると、ゲストの起動が停止し、エラーメッセージ No boot device が表示されます。ただし、ゲスト仮想マシンは QEMU コマンドを直接使用して正常に起動できます。

**調査**

ディスクのバスタイプは、既存のディスクイメージをインポートするために コマンドで指定されていません。

```
# virt-install \
```

```
--connect qemu:///system \
--ram 2048 -n rhel_64 \
--os-type=linux --os-variant=rhel5 \
--disk  path=/root/RHEL-Server-5.8-64-virtio.qcow2,device=disk,format=qcow2 \
--vcpus=2 --graphics spice --noautoconsole --import
```

**ただし、QEMU を使用してゲスト仮想マシンを起動するために使用されるコマンドラインは、バスタイプに virtio を使用することを示しています。**

```
# ps -ef | grep qemu
/usr/libexec/qemu-kvm -monitor stdio -drive file=/root/RHEL-Server-5.8-32-
virtio.qcow2,index=0,if=virtio,media=disk,cache=none,format=qcow2 -net
nic,vlan=0,model=rtl8139,macaddr=00:30:91:aa:04:74 -net tap,vlan=0,script=/etc/qemu-
ifup,downscript=no -m 2048 -smp 2,cores=1,threads=1,sockets=2 -cpu qemu64,+sse2 -soundhw
ac97 -rtc-td-hack -M rhel5.6.0 -usbdevice tablet -vnc :10 -boot c -no-kvm-pit-reinjection
```

**インポートしたゲスト用に libvirt によって生成されたゲストの XML の *bus=* をメモします。**

```
<domain type='qemu'>
 <name>rhel_64</name>
 <uuid>6cd34d52-59e3-5a42-29e4-1d173759f3e7</uuid>
 <memory>2097152</memory>
 <currentMemory>2097152</currentMemory>
 <vcpu>2</vcpu>
 <os>
   <type arch='x86_64' machine='rhel5.4.0'>hvm</type>
   <boot dev='hd'/>
 </os>
 <features>
   <acpi/>
   <apic/>
   <pae/>
 </features>
 <clock offset='utc'>
   <timer name='pit' tickpolicy='delay'/>
 </clock>
 <on_poweroff>destroy</on_poweroff>
 <on_reboot>restart</on_reboot>
 <on_crash>restart</on_crash>
 <devices>
   <emulator>/usr/libexec/qemu-kvm</emulator>
   <disk type='file' device='disk'>
     <driver name='qemu' type='qcow2' cache='none'/>
     <source file='/root/RHEL-Server-5.8-64-virtio.qcow2'/>
     <emphasis role="bold"><target dev='hda' bus='ide'/></emphasis>
     <address type='drive' controller='0' bus='0' unit='0'/>
   </disk>
   <controller type='ide' index='0'/>
   <interface type='bridge'>
     <mac address='54:52:00:08:3e:8c'/>
     <source bridge='br0'/>
```

```
    </interface>
    <serial type='pty'>
      <target port='0'/>
    </serial>
    <console type='pty'>
      <target port='0'/>
    </console>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' keymap='en-us'/>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
    </video>
  </devices>
  </domain>
```

ディスクのバスタイプは *ide* に設定されています。これは、libvirt によって設定されるデフォルト値です。これは間違ったバスタイプであり、インポートされたゲストで起動に失敗した原因となっていました。

**解決方法**

**手順B.2 ディスクバスタイプの修正**

1. インポートされたゲストの定義を解除してから、*bus=virtio* と以下のように再インポートします。

```
# virsh destroy rhel_64
# virsh undefine rhel_64
# virt-install \
--connect qemu:///system \
--ram 1024 -n rhel_64 -r 2048 \
--os-type=linux --os-variant=rhel5  \
--disk path=/root/RHEL-Server-5.8-64-virtio.qcow2,device=disk,bus=virtio,format=qcow2 \
--vcpus=2 --graphics spice --noautoconsole --import
```

2. virsh edit を使用してインポートされたゲストの XML を編集し、ディスクバスの種類を修正します。

## B.7. 仮想ネットワークの *デフォルト* は開始されていません

**現象**

通常、*default* という名前の仮想ネットワークの設定は、libvirt パッケージの一部としてインストールされ、libvirtd の起動時に自動起動を行うように設定されます。

*デフォルト* のネットワーク（またはその他のローカルで作成されたネットワーク）が起動できない場合、その接続にそのネットワークを使用するように設定された仮想マシンも起動に失敗し、以下のエラーメッセージが表示されます。

```
Virtual network default has not been started
```

**調査**

libvirt 仮想ネットワークの起動に失敗する最も一般的な理由の 1 つは、そのネットワーク上のクライアントからの DHCP および DNS 要求に対応するために必要な dnsmasq インスタンスが起動に失敗したことです。

原因を確認するには、root シェルから virsh net-start default を実行して、*デフォルト* の仮想ネットワークを起動します。

このアクションで仮想ネットワークが正常に起動しない場合は、/var/log/libvirt/libvirtd.log を開き、完全なエラーログメッセージを表示します。

以下のようなメッセージが表示されると、問題は、libvirt のブリッジですでにリッスンしているシステム全体の dnsmasq インスタンスであり、libvirt の独自の dnsmasq インスタンスがこれを実行できない可能性があります。エラーメッセージで注意すべき最も重要な部分は dnsmasq で、終了ステータスは 2 です。

```
Could not start virtual network default: internal error
Child process (/usr/sbin/dnsmasq --strict-order --bind-interfaces
--pid-file=/var/run/libvirt/network/default.pid --conf-file=
--except-interface lo --listen-address 192.168.122.1
--dhcp-range 192.168.122.2,192.168.122.254
--dhcp-leasefile=/var/lib/libvirt/dnsmasq/default.leases
--dhcp-lease-max=253 --dhcp-no-override) status unexpected: exit status 2
```

**解決方法**

マシンが dnsmasq を使用して物理ネットワークに DHCP を提供していない場合は、dnsmasq を完全に無効にします。

物理ネットワークに DHCP を提供するために dnsmasq を実行する必要がある場合は、/etc/dnsmasq.conf ファイルを編集します。最初の行と、その行に続く 2 行のいずれかを追加またはコメント解除します。3 行すべてを追加またはコメント解除しないでください。

```
bind-interfaces
interface=name_of_physical_interface
listen-address=chosen_IP_address
```

この変更を加えてファイルを保存したら、システム全体の dnsmasq サービスを再起動します。

次に、virsh net-start *default* コマンドでデフォルトのネットワークを起動します。

仮想マシンを起動します。

## B.8. ゲスト上の PXE ブート (または DHCP ) が失敗

現象

ゲスト仮想マシンは正常に起動するが、DHCP から IP アドレスを取得できないか、PXE プロトコルを使用してブートを実行できない、またはその両方。このエラーには、ブリッジの転送遅延時間が長く設定されている場合と iptables パッケージとカーネルがチェックサムの難号化 (mangle) 規則をサポートしない場合という 2 つの一般的な原因があります。

ブリッジの転送遅延時間が長い

調査

これは、このエラーの最も一般的な原因になります。ゲストのネットワークインターフェイスが STP (Spanning Tree Protocol) 対応のブリッジデバイスに接続しており、かつ長時間の転送遅延時間が設定されていると、ゲストがブリッジに接続してからその転送遅延時間が経過してからでなければゲスト仮想マシンからブリッジにネットワークパケットを転送しません。この遅延により、インターフェイスからのトラフィックの監視、背後での MAC アドレスの決定、ネットワークトポロジー内の転送ループ防止がブリッジ時間で可能になります。

転送遅延がゲストの PXE または DHCP クライアントのタイムアウトよりも長い場合、クライアントの操作に失敗し、ゲストは(PXE の場合)起動に失敗するか、または(DHCP の場合) IP アドレスの取得に失敗します。

解決方法

この場合は、ブリッジの転送遅延を 0 に変更するか、ブリッジの STP を無効にします。

**注記**

この解決法は、ブリッジが複数のネットワーク接続に使用されておらず、複数のエンドポイントを単一のネットワーク接続のみに使用されている場合にのみ適用されます (libvirt で使用される最も一般的なブリッジのユースケース)。

ゲストが libvirt が管理する仮想ネットワークに接続するインターフェイスを備えている場合、そのネットワークの定義を編集し、再起動します。たとえば、以下のコマンドでデフォルトのネットワークを編集します。

```
# virsh net-edit default
```

<bridge> 要素に以下の属性を追加します。

```
<name_of_bridge='virbr0' delay='0' stp='on'/>
```

**注記**

delay='0' および stp='on' は仮想ネットワークのデフォルト設定なので、このステップは設定がデフォルトから変更されている場合にのみ必要となります。

ゲストインターフェイスが libvirt 外で設定されているホストブリッジに接続されている場合は、遅延設定を変更します。

/etc/sysconfig/network-scripts/ifcfg-*name_of_bridge* ファイルで以下の行を追加または編集し、0 秒の遅延で STP を有効にします。

```
STP=on
DELAY=0
```

設定ファイルの変更後にブリッジデバイスを再起動します。

```
/sbin/ifdown name_of_bridge
/sbin/ifup name_of_bridge
```

注記

*name_of_bridge* がネットワークのルートブリッジではない場合、そのブリッジの遅延は最終的にルートブリッジに設定された遅延時間にリセットされます。この場合の唯一の解決策は、*name_of_bridge* で STP を完全に無効にすることです。

**iptables パッケージとカーネルがチェックサム難号化ルールをサポートしない場合**

**調査**

このメッセージは、以下の 4 つの条件すべてが該当する場合にのみ問題となります。

- ゲストが virtio ネットワークデバイスを使用している。

  その場合、設定ファイルに model type='virtio' が含まれています。

- ホストに vhost-net モジュールがロードされている。

  ls /dev/vhost-net を実行して空の結果が返されなければ、このモジュールはロードされています。

- ゲストがホスト上で直接実行されている DHCP サーバーから IP アドレスを取得しようとしている。

- ホストの iptables のバージョンが 1.4.10 よりも古い。

  iptables 1.4.10 は libxt_CHECKSUM 拡張を追加する最初のバージョンでした。libvirtd ログに以下のメッセージが表示される場合は、これに該当します。

```
warning: Could not add rule to fixup DHCP response checksums on network default
warning: May need to update iptables package and kernel to support CHECKSUM
rule.
```

重要

この一覧の最初の 3 つの条件すべてが該当していなければ上記の警告メッセージは問題を示すものではなく、無視することができます。

これらの条件が当てはまる場合、ホストからゲストへ送信される UDP パケットには未算出のチェックサムがあります。これにより、ホストの UDP パケットがゲストのネットワークスタックには無効のように表示されます。

ソリューション

この問題を解決するには、上記の 4 つの条件のいずれかを無効にします。最善の解決策は、ホスト iptables およびカーネルを可能な限り iptables-1.4.10 以降に更新することです。または、この特定のゲストの vhost-net ドライバーを無効にすることが最も具体的な修正方法になります。これを実行するには、以下のコマンドでゲストの設定を編集します。

```
virsh edit name_of_guest
```

<driver> の行を変更するか、<interface> セクションに追加します。

```
<interface type='network'>
  <model type='virtio'/>
  <driver name='qemu'/>
  ...
</interface>
```

変更を保存し、ゲストをシャットダウンしてから再起動します。

この問題が解決されない場合、firewalld とデフォルトの libvirt ネットワーク間に競合がある可能性があることが原因として考えられます。

これを修正するには、service firewalld stop コマンドで firewalld を停止し、service libvirtd restart コマンドで libvirt を再起動します。

## B.9. ゲストは外部ネットワークに到達できるが、MACVTAP インターフェイスの使用時にはホストにアクセスできない

**現象**

ゲストは他のゲストと通信できるが、macvtap (または *type='direct'*) ネットワークインターフェイスを使用するように設定された後にはホストマシンに接続できない。

**調査**

Virtual Ethernet Port Aggregator (VEPA) や VN-Link 対応スイッチに接続していない場合でも、macvtap インターフェイスは役に立ちます。このようなインターフェイスのモードを bridge に設定すると、従来のホストブリッジデバイスの使用に伴う設定の問題（または NetworkManager の非互換性）なしに、ゲストが非常に簡単に物理ネットワークに直接接続できます。

ただし、ゲスト仮想マシンが macvtap などの *type='direct'* ネットワークインターフェイスを使用するように設定されている場合、ネットワーク上の他のゲストや他の外部ホストと通信できるにも拘らず、ゲストは独自のホストと通信できません。

この状況は、実際にはエラーではありません。これは、macvtap の定義済み動作です。ホストの物理イーサネットが macvtap ブリッジに割り当てられている方法が原因で、物理インターフェイスに転送されるゲストからそのブリッジへのトラフィックは、ホストの IP スタックに送り返されることはありません。さらに、物理インターフェイスに送信されたホストの IP スタックからのトラフィックも macvtap ブリッジに送り返されず、ゲストに転送することができません。

**ソリューション**

libvirt を使って、分離されたネットワークと、このネットワークに接続する各ゲスト仮想マシンの 2 つ目のインターフェイスを作成します。これでホストとゲストはこの分離されたネットワーク上で直接通信でき、NetworkManager との互換性も維持できます。

**手順B.3 libvirt で分離されたネットワークを作成します。**

1. /tmp/isolated.xml ファイルに以下の XML を追加して保存します。自分のネットワーク上で 192.168.254.0/24 が既に使われている場合、別のネットワークを選びます。

```
<network>
  <name>isolated</name>
  <ip address='192.168.254.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.254.2' end='192.168.254.254' />
    </dhcp>
  </ip>
</network>
```

2. 次のコマンドでネットワークを作成します。virsh net-define /tmp/isolated.xml

3. **virsh net-autostart isolated** コマンドでネットワークを自動起動するように設定します。

4. **virsh net-start isolated** コマンドでネットワークを起動します。

5. **virsh edit *name_of_guest*** を使用して、ネットワーク接続に **macvtap** を使用する各ゲストの設定を編集し、***<devices>*** セクションに新しい ***<interface>*** を追加します( ***<model type='virtio'/>*** 行は任意です)。

```
<interface type='network'>
  <source network='isolated'/>
  <model type='virtio'/>
</interface>
```

6. 各ゲストをシャットダウンし、再起動します。

これでゲストは **192.168.254.1** のアドレスにあるホストにアクセスでき、ホストは各ゲストが **DHCP** から取得した **IP** アドレスでゲストにアクセスできます (または、ゲストに手動で **IP** アドレスを設定することもできます)。この新たなネットワークはこのホストとゲスト専用に分離されているので、ゲストからの他の通信はすべて **macvtap** インターフェイスを使用することになります。

## B.10. COULD NOT ADD RULE TO FIXUP DHCP RESPONSE CHECKSUMS ON NETWORK *'DEFAULT'*

**現象**

以下のメッセージが表示されます。

```
Could not add rule to fixup DHCP response checksums on network 'default'
```

**調査**

このメッセージはエラーに見えますが、ほとんどの場合は問題ありません。

**ソリューション**

ゲスト仮想マシンが **DHCP** から **IP** アドレスを取得できないという問題が発生していない限り、このメッセージは無視してかまいません。

この場合は、「ゲスト上の PXE ブート (または DHCP ) が失敗」 を参照してください。

## B.11. UNABLE TO ADD BRIDGE BR0 PORT VNET0: NO SUCH DEVICE

**現象**

以下のエラーメッセージが表示されます。

```
Unable to add bridge name_of_bridge port vnet0: No such device
```

たとえば、ブリッジ名が ***br0*** の場合、エラーメッセージは以下のようになります。

```
Unable to add bridge br0 port vnet0: No such device
```

**libvirt** のバージョン **0.9.6** 以前では、以下のようなメッセージになります。

```
Failed to add tap interface to bridge name_of_bridge: No such device
```

たとえば、ブリッジの名前が ***br0*** の場合、エラーは次のようになります。

```
Failed to add tap interface to bridge 'br0': No such device
```

**調査**

いずれのエラーメッセージも、ゲストの (またはドメインの) **<interface>** 定義で指定されたブリッジデバイスが存在しないことを示しています。

エラーメッセージに記載されているブリッジデバイスが存在しないことを確認するには、シェルコマンド **ip addr show*br0*** を使用します。

以下のようなメッセージが表示されると、その名前のブリッジが存在しないことが確認できます。

```
br0: error fetching interface information: Device not found
```

存在しない場合は、解決法に進みます。

ただし、メッセージが以下のようであれば、問題は別に存在します。

```
 3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
    link/ether 1b:c4:94:cf:fd:17 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global br0
```

**解決方法**

既存のブリッジを編集するか、virsh で新しいブリッジを作成します。

virsh を使って既存のブリッジもしくはネットワークの設定を編集するか、ブリッジデバイスをホストシステム設定に追加します。

virsh を使用して、既存のブリッジ設定を編集します。

virsh edit *name_of_guest* を使用して、すでに存在するブリッジまたはネットワークを使用するように <interface> 定義を変更します。

たとえば、type='bridge' を type='network' に変更し、<source bridge='br0'/> を <source network='default'/> に変更します。

virsh を使用したホストブリッジの作成

libvirt バージョン 0.9.8 以降の場合は、virsh iface-bridge コマンドでブリッジデバイスを作成できます。これにより、eth0 を備えたブリッジデバイス *br0* が作成されます。これは、ブリッジの一部として設定された物理ネットワークインターフェイスです。

```
virsh iface-bridge eth0 br0
```

必要に応じて、以下のコマンドを使用して、このブリッジを削除し、元の eth0 設定を復元します。

```
virsh iface-unbridge br0
```

ホストブリッジを手動で作成

libvirt の古いバージョンでは、ホスト上にブリッジデバイスを手動で作成することができます。手順は、「libvirt を使用したブリッジネットワーク」 を参照してください。

### B.12. ゲストが UNABLE TO START WITH ERROR: WARNING: COULD NOT OPEN /DEV/NET/TUN

**現象**

ホストシステムに type='ethernet' (generic ethernet とも呼ばれる)インターフェイスを設定した後は、ゲスト仮想マシンは起動しません。以下のメッセージと同様に、libvirtd.log、/var/log/libvirt/qemu/*name_of_guest*.log、またはその両方にエラーが表示されます。

```
warning: could not open /dev/net/tun: no virtual network emulation qemu-kvm: -netdev
tap,script=/etc/my-qemu-ifup,id=hostnet0: Device 'tap' could not be initialized
```

**調査**

QEMU とそのゲストの潜在的なセキュリティー上の欠陥に対してホスト保護のレベルを減らす必要があるため、汎用イーサネットインターフェイスタイプ(<interface type='ethernet'>)の使用は推奨されません。ただし、このタイプのインターフェイスを使用して、libvirt で直接サポートされていないその他の機能を利用する必要がある場合があります。たとえば、openvswitch は libvirt-0.9.11 まで libvirt ではサポートされていませんでした。そのため、古いバージョンの libvirt では、<interface type='ethernet'> がゲストを openvswitch ブリッジに接続する唯一の方法でした。

ただし、ホストシステムに他の変更を加えずに <interface type='ethernet'> インターフェイスを設定すると、ゲスト仮想マシンは正常に起動しません。

この障害の原因は、このタイプのインターフェイスでは、QEMU によって呼び出されるスクリプトが tap デバイスを操作する必要があることです。ただし、type='ethernet' が設定されている場合は、QEMU のロックダウンを試み、libvirt と SELinux がこれを防ぐためにいくつかのチェックが行われています。（通常は、libvirt はすべての tap デバイスの作成と操作を実行し、タップデバイスのオープンファイル記述子を QEMU に渡します。）

**解決方法**

一般的なイーサネットインターフェイスと互換性を持つようにホストシステムを再設定します。

**手順B.4 一般的なイーサネットインターフェイスを使用するようにホストシステムを再設定する**

1. /etc/selinux/config で SELINUX=permissive を設定して、SELinux を Permissive に設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2. **root シェルから、 setenforce permissive コマンドを実行します。**

3. **/etc/libvirt/qemu.conf で、以下の行を追加または編集します。**

```
clear_emulator_capabilities = 0
```

```
user = "root"
```

```
group = "root"
```

```
cgroup_device_acl = [
        "/dev/null", "/dev/full", "/dev/zero",
        "/dev/random", "/dev/urandom",
        "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
 "/dev/rtc", "/dev/hpet", "/dev/net/tun",
```

4. **libvirtd を再起動します。**

**重要**

**これらの手順はそれぞれ、QEMU ゲストドメインに対するホストのセキュリティー保護を大幅に減らすため、<interface type='ethernet'> を使用する代わりにこの設定を使用する必要があります。**

**注記**

**SELinux の詳細は、『Red Hat Enterprise Linux 6 Security-Enhanced Linux User Guide』 を参照してください。**

### B.13. で移行に失敗する ERROR: UNABLE TO RESOLVE ADDRESS

**現象**

**QEMU ゲストの移行が失敗し、以下のエラーメッセージが表示される。**

```
# virsh migrate qemu qemu+tcp://192.168.122.12/system
 error: Unable to resolve address name_of_host service '49155': Name or service not known
```

**たとえば、移行先のホスト名が newyork の場合、エラーメッセージは以下のようになります。**

```
# virsh migrate qemu qemu+tcp://192.168.122.12/system
error: Unable to resolve address 'newyork' service '49155': Name or service not known
```

**ただし、このエラーは newyork ホスト名をどこにも使用していないため、異常なように見えます。**

**調査**

**移行時に、移行先ホスト上で稼働している libvirtd は移行データの受信が予想されるアドレスおよびポートから URI を作成し、これを移行元ホスト上で稼働している libvirtd に送信します。**

**上記の例では、移行先ホスト (192.168.122.12) は名前を *'newyork'* に設定しています。何らかの理由で、そのホスト上で実行中の libvirtd は IP アドレスに対して名前を解決できず、送信されるべきものが有効なままとなっています。このため、移行元の *libvirtd* が名前を解決することを予期してホスト名 'newyork' が返されました。これは、DNS が適切に設定されていないか、/etc/hosts にローカルループバックアドレス (127.0.0.1) に関連付けられたホスト名がある場合に発生します。**

**移行データに使用するアドレスは、移行先libvirtdへの接続に使用するアドレス (qemu+tcp://192.168.122.12/system など) から自動的に決定されないことに注意してください。これは、移行先 libvirtd と通信するために、移行元 libvirtd が（別のマシンで実行する可能性がある）virsh が必要とするネットワークインフラストラクチャーを使用する必要があるためです。**

**解決方法**

**最善の解決法として、DNS を正しく設定し、移行に関連するすべてのホストがすべてのホスト名を解決できるようにすることができます。**

**DNS をこのように設定できない場合は、各ホスト上の /etc/hosts ファイルに、移行に使用するすべてのホストのリストを手動で追加することができます。しかし、ダイナミックな環境ではそのようなリストの一貫性の維持は困難です。**

いずれの方法でもホスト名を解決できない場合、**virsh migrate** は移行ホストの指定をサポートします。

```
# virsh migrate qemu qemu+tcp://192.168.122.12/system tcp://192.168.122.12
```

移行先の **libvirtd** は **tcp://192.168.122.12 URI** を取得し、自動生成されたポート番号を追加します。この番号が望ましくない場合は (たとえば、ファイアウォール設定との関連により適切でない場合)、ポート番号は以下のコマンドで指定できます。

```
# virsh migrate qemu qemu+tcp://192.168.122.12/system tcp://192.168.122.12:12345
```

別のオプションとして、トンネル化された移行を使用することもできます。トンネル化された移行は移行データ用に別の接続を作成しませんが、宛先 **libvirtd** との通信に使用される接続を介してデータをトンネルします（例： **qemu+tcp://192.168.122.12/system**）。

```
# virsh migrate qemu qemu+tcp://192.168.122.12/system --p2p --tunnelled
```

## B.14. UNABLE TO ALLOW ACCESS FOR DISK PATH: NO SUCH FILE OR DIRECTORYで移行に失敗

### 現象

**libvirt** がディスクイメージにアクセスできないため、ゲスト仮想マシン (またはドメイン) を移行できない。

```
# virsh migrate qemu qemu+tcp://name_of_host/system
error: Unable to allow access for disk path /var/lib/libvirt/images/qemu.img: No such file or
directory
```

たとえば、移行先のホスト名が **newyork** の場合、エラーメッセージは以下のようになります。

```
# virsh migrate qemu qemu+tcp://newyork/system
error: Unable to allow access for disk path /var/lib/libvirt/images/qemu.img: No such file or
directory
```

### 調査

デフォルトでは、移行で移動するのは実行中のゲストのメモリー内の状態のみです (メモリー–または **CPU** 状態など)。移行中にはディスクイメージは移動しませんが、両方のホストから同じパスでアクセスできる状態である必要があります。

**ソリューション**

両方のホストの同じ位置に共有ストレージをセットアップし、マウントします。最も簡単な方法として、**NFS** を使用することができます。

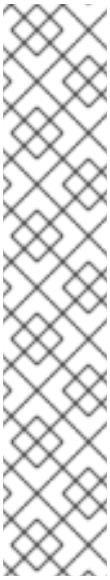
**手順B.5 共有ストレージのセットアップ**

1. ホスト上に共有ストレージとして機能する **NFS** サーバーをセットアップします。関連するすべてのホストが **NFS** で共有ストレージにアクセスしている限り、**NFS** サーバーには移行関連のいずれかのホストを使用できます。

```
# mkdir -p /exports/images
# cat >>/etc/exports <<EOF
/exports/images    192.168.122.0/24(rw,no_root_squash)
EOF
```

2. **libvirt** を実行しているすべてのホスト上の共通の場所にエクスポートされたディレクトリーをマウントします。たとえば、**NFS** サーバーの **IP** アドレスが **192.168.122.1** の場合は、以下のコマンドでディレクトリーをマウントします。

```
# cat >>/etc/fstab <<EOF
192.168.122.1:/exports/images  /var/lib/libvirt/images  nfs  auto  0 0
EOF
# mount /var/lib/libvirt/images
```



**注記**

**NFS** を使ってあるホストからローカルディレクトリーをエクスポートし、別のホストの同じパスにマウントすることはできません。ディスクイメージの保存に使われるディレクトリーは、両方のホストで共有ストレージからマウントされる必要があります。これが正確に設定されていない場合、ゲスト仮想マシンは移行時にそのディスクイメージへのアクセスを失う可能性があります。これは、ゲストを移行先に正常に移行した後に、移行元ホストの **libvirt** デーモンがディスクイメージ上の所有者、権限および **SELinux** ラベルを変更する可能性があるからです。

**libvirt** は、ディスクイメージが共有ストレージの場所からマウントされたことを検出すると、これらの変更を実施しません。

## B.15. LIBVIRTD の開始時に存在するゲスト仮想マシンがない

**現象**

**libvirt デーモンは正常に起動したが、ゲスト仮想マシンが見当たらない。**

```
# virsh list --all
 Id    Name                           State
----------------------------------------------------
#
```

**調査**

**この問題の原因としていくつもの原因が考えられます。以下のテストを実施して原因を特定します。**

**KVM カーネルモジュールの確認**

**KVM カーネルモジュールがカーネルに挿入されていることを確認する。**

```
# lsmod | grep kvm
kvm_intel             121346  0
kvm                   328927  1 kvm_intel
```

**AMD マシンを使用している場合は、root シェルの同様のコマンドlsmod | grep kvm_amdを使用して、kvm_amdのカーネルモジュールがカーネルに挿入されていることを確認します。**

**モジュールがない場合は、modprobe *<modulename>* を使用して挿入します。**

**注記**

**一般的ではありませんが、KVM 仮想化サポートがカーネルにコンパイルされている場合もあります。その場合は、モジュールは必要ありません。**

**仮想化拡張機能の確認**

**仮想化拡張機能がホスト上でサポートされ、有効にされていることを確認します。**

```
# egrep "(vmx|svm)" /proc/cpuinfo
flags  : fpu vme de pse tsc ... svm ... skinit wdt npt lbrv svm_lock nrip_save
flags  : fpu vme de pse tsc ... svm ... skinit wdt npt lbrv svm_lock nrip_save
```

**BIOS 設定内のファームウェア設定で仮想化拡張機能を有効にします。詳細は、ハードウェアのドキュメントを参照してください。**

**クライアント URI 設定の確認**

**クライアントの URI が必要に応じて設定されていることを確認します。**

```
# virsh uri
vbox:///system
```

**たとえば、このメッセージは URI が QEMUではなく VirtualBox ハイパーバイザーに接続されていることを示し、本来は QEMU ハイパーバイザーに接続するように設定されているはずの URI の設定エラーを示しています。URI が QEMU に正常に接続している場合は、メッセージは以下のようになります。**

```
# virsh uri
qemu:///system
```

**他のハイパーバイザーが存在し、libvirt がこのハイパーバイザーとデフォルトで通信する場合に、この状況が発生します。**

**ソリューション**

**これらのテスト実行後に、以下のコマンドでゲスト仮想マシンの一覧を表示します。**

```
# virsh list --all
```

## B.16. UNABLE TO CONNECT TO SERVER AT 'HOST:16509': CONNECTION REFUSED ... ERROR: FAILED TO CONNECT TO THE HYPERVISOR

**現象**

**libvirtd 接続で TCP ポートをリッスンしている間に、接続が失敗する。**

```
# virsh -c qemu+tcp://host/system
error: unable to connect to server at 'host:16509': Connection refused
error: failed to connect to the hypervisor
```

**/etc/libvirt/libvirtd.conf で設定を変更した後も、libvirt デーモンは TCP ポートをリッスンしません。**

```
# grep listen_ /etc/libvirt/libvirtd.conf
listen_tls = 1
listen_tcp = 1
listen_addr = "0.0.0.0"
```

しかし、**libvirt** の **TCP** ポートは設定変更後も開いたままです。

```
# netstat -lntp | grep libvirtd
#
```

**調査**

**libvirt** デーモンは、**--listen** オプションなしで起動されました。以下のコマンドを実行してこれを確認します。

```
# ps aux | grep libvirtd
root     27314  0.0  0.0 1000920 18304 ?       Sl   Feb16   1:19 libvirtd --daemon
```

出力に **--listen** オプションが含まれていません。

**解決方法**

**--listen** オプションでデーモンを起動します。

これを行うには、**/etc/sysconfig/libvirtd** ファイルを修正して、以下の行のコメントを解除します。

```
#LIBVIRTD_ARGS="--listen"
```

次に、以下のコマンドで **libvirtd** サービスを再起動します。

```
# /etc/init.d/libvirtd restart
```

## B.17. 一般的な XML エラー

**libvirt** では、**XML** ドキュメントを使用して構造化データを保存します。**XML** ドキュメントが **API** を介して **libvirt** に渡されると、さまざまな一般的なエラーが発生します。誤った形式の **XML**、不適切な値、欠落している要素など、一般的な **XML** エラーの一部を以下に示します。

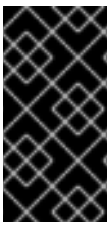
### B.17.1. ドメイン定義の編集

これは推奨されていませんが、ゲスト仮想マシン (またはドメインの XML ファイル) を手動で編集することが必要になる場合があります。ゲストの XML にアクセスして編集するには、次のコマンドを使用します。

```
# virsh edit name_of_guest.xml
```

このコマンドは、ゲスト仮想マシンの現在の定義を持つテキストエディターでファイルを開きます。編集を終了して変更を保存したら、libvirt により XML が再読み込みされ、解析されます。XML が正しい場合は、以下のメッセージが表示されます。

```
# virsh edit name_of_guest.xml

Domain name_of_guest.xml XML configuration edited.
```



**重要**

**virsh で edit コマンドを使用して XML ドキュメントを編集する場合は、すべての変更を保存してから、エディターを終了します。**

XML ファイルを保存したら、xmllint コマンドを使用して XML の形式が正しいことを確認するか、virt-xml-validate コマンドを使用して使用上の問題を確認します。

```
# xmllint --noout config.xml
```

```
# virt-xml-validate config.xml
```

エラーが返されない場合、XML 記述は正しく設定され、libvirt スキーマと一致します。スキーマがすべての制約を把握していない場合は、報告されたエラーを修正するとさらにトラブルシューティングが行われます。

**libvirt が保存する XML ドキュメント**

これらのドキュメントには、ゲストの状態と設定の定義が記載されています。これらのドキュメントは自動的に生成されるため、手動で編集しないでください。これらのドキュメントのエラーには、破損したドキュメントのファイル名が含まれています。ファイル名は、URI で定義されたホストマシンでのみ有効です。これは、コマンドが実行されたマシンを参照する場合があります。

libvirt が作成したファイルでエラーが発生することはまれです。ただし、このエラーの原因の 1 つとして、libvirt のダウングレードが考えられます。新しいバージョンの libvirt では、古いバー

ジョンで生成された XML を読み取ることができるのに対し、古いバージョンの libvirt では、新しいバージョンで追加された XML 要素により混同される可能性があります。

### B.17.2. XML 構文エラー

構文エラーは XML パーサーで検出されます。エラーメッセージには、問題を特定するための情報が記載されています。

XML パーサーのこのエラーメッセージの例は 3 行で設定されています。最初の行はエラーメッセージを表し、以降の 2 行にはエラーを含む XML コードのコンテキストと場所が含まれます。3 行目には、エラーがその上の行のどこにあるかを示すインジケーターが含まれています。

```
error: (name_of_guest.xml):6: StartTag: invalid element name
<vcpu>2</vcpu><
-----------------^
```

このメッセージに含まれる情報

**(*name_of_guest.xml*)**

エラーが含まれるドキュメントのファイル名です。括弧内のファイル名は、メモリーから解析される XML ドキュメントを説明するシンボリック名であり、ディスク上のファイルに直接対応しません。括弧内に含まれないファイル名は、接続のターゲットにあるローカルファイルです。

**6**

エラーが含まれる XML ファイルの行番号です。

**StartTag: 無効な要素名**

これは、特定の XML エラーを説明するlibxml2パーサーからのエラーメッセージです。

#### B.17.2.1. ドキュメントの迷子の <

現象

以下のエラーが発生します。

```
error: (name_of_guest.xml):6: StartTag: invalid element name
<vcpu>2</vcpu><
-----------------^
```

**調査**

**このエラーメッセージは、パーサーがゲストの XML ファイルの 6 行目の< 記号の後に新しい要素名を想定していることを示しています。**

**テキストエディターで行番号の表示が有効になっていることを確認します。XML ファイルを開き、6 行目のテキストを探します。**

```
<domain type='kvm'>
  <name>name_of_guest</name>
<memory>524288</memory>
<vcpu>2</vcpu><
```

**ゲストの XML ファイルのこのスニペットには、ドキュメントに余分な<が含まれています。**

**ソリューション**

**余分な<を削除するか、新しい要素を終了します。**

#### B.17.2.2. 終了していない属性

**現象**

**以下のエラーが発生します。**

```
error: (name_of_guest.xml):2: Unescaped '<' not allowed in attributes values
<name>name_of_guest</name>
--^
```

**調査**

**ゲストの XML ファイルのこのスニペットには、終了していない要素の属性値が含まれます。**

```
<domain type='kvm>
<name>name_of_guest</name>
```

**この例では、'kvm'に 2 番目の引用符が欠けています。XML の開始タグや終了タグと同様に、引用符やアポストロフィーなどの属性値の文字列は開いて閉じる必要があります。**

**解決方法**

すべての属性値文字列を正しく開いたり閉じたりします。

#### B.17.2.3. タグの開始と終了の不一致

**現象**

以下のエラーが発生します。

```
error: (name_of_guest.xml):61: Opening and ending tag mismatch: clock line 16 and domain
</domain>
----------^
```

**調査**

上記のエラーメッセージには、問題のあるタグを識別するためのヒントが 3 つ含まれています。

最後のコロンに続くメッセージ **clock line 16 and domain** は、**<clock>** にはドキュメントの 16 行目に一致しないタグが含まれていることを示しています。最後のヒントは、メッセージのコンテキスト部分のポインターで、2 番目の問題のあるタグを識別します。

ペアになっていないタグは、**/>** で閉じる必要があります。以下のスニペットはこのルールに従わず、上記のエラーメッセージが表示されます。

```
<domain type='kvm'>
  ...
    <clock offset='utc'>
```

このエラーは、ファイル内で **XML** タグが一致しないために発生します。すべての **XML** タグに、一致する開始タグと終了タグが必要です。

**一致しない XML タグのその他の例**

以下の例では、同様のエラーメッセージと、一致しない **XML** タグのバリエーションを示しています。

このスニペットには、< features > の非終了ペアタグが含まれています。

```
<domain type='kvm'>
 ...
 <features>
   <acpi/>
   <pae/>
 ...
 </domain>
```

このスニペットには、対応する開始タグのない終了タグ (</name>) が含まれます。

```
<domain type='kvm'>
  </name>
  ...
</domain>
```

**解決方法**

すべての XML タグが正しく開始および終了していることを確認します。

#### B.17.2.4. タグの書式エラー

**現象**

以下のエラーメッセージが表示されます。

```
error: (name_of_guest.xml):1: Specification mandate value for attribute ty
<domain ty pe='kvm'>
-----------^
```

**調査**

XML エラーは、簡単なタイプミスが原因で簡単に発生します。このエラーメッセージでは、XML エラー (この例では type という単語の真ん中に余計な空白があります) をポインターで強調表示します。

```
<domain ty pe='kvm'>
```

これらの XML の例は、特殊文字の欠落や追加の文字などの誤植が原因で正しく解析されません。

-

```
<domain type 'kvm'>
```

```
<dom#ain type='kvm'>
```

**解決方法**

問題のあるタグを特定するには、ファイルのコンテキストのエラーメッセージを読み、ポインターでエラーを特定します。**XML** を修正し、変更を保存します。

### B.17.3. ロジックおよび設定エラー

フォーマットが適切な **XML** ドキュメントには、構文は正しいものの、**libvirt** が解析できないエラーが含まれる場合があります。このようなエラーは多数存在します。以下では、最も一般的な **2** つのケースについて説明しています。

#### B.17.3.1. バニッシュ部分

**現象**

加えた変更の一部が表示されず、ドメインの編集または定義後も反映されません。**define** または **edit** コマンドは機能しますが、再度 **XML** をダンプすると変更が消えます。

**調査**

このエラーは、**libvirt** が解析しない設定または構文の破損が原因で発生します。**libvirt** ツールは通常、認識している設定のみを探しますが、その他のすべてを無視します。これにより、**libvirt** が入力を解析した後に **XML** の変更の一部が回避されます。

**解決方法**

**XML** 入力を検証してから、**edit** コマンドまたは **define** コマンドに渡します。 **libvirt** 開発者は、**libvirt** にバンドルされた **XML** スキーマのセットを維持します。これは、**libvirt** が使用する **XML** ドキュメントで許可されている設定の大半を定義します。

以下のコマンドを使用して、**libvirt XML** ファイルを検証します。

```
# virt-xml-validate libvirt.xml
```

このコマンドが成功すると、**libvirt** はおそらく **XML** からすべての設定を理解します。ただし、スキーマが特定のハイパーバイザーに対してのみ有効なオプションを検出できない場合を除き

ます。たとえば、**virsh dump** コマンドの結果として **libvirt** が生成した **XML** は、エラーなしで検証する必要があります。

#### B.17.3.2. ドライブのデバイスタイプが間違っている

**現象**

**CD-ROM** 仮想ドライブのソースイメージの定義は追加されていますが、ここには記載されていません。

```
# virsh dumpxml domain
<domain type='kvm'>
  ...
  <disk type='block' device='cdrom'>
    <driver name='qemu' type='raw'/>
    <target dev='hdc' bus='ide'/>
    <readonly/>
  </disk>
  ...
</domain>
```

**解決方法**

欠落している **<source>** パラメーターを次のように追加して、**XML** を修正します。

```
<disk type='block' device='cdrom'>
  <driver name='qemu' type='raw'/>
  <source file='/path/to/image.iso'/>
  <target dev='hdc' bus='ide'/>
  <readonly/>
</disk>
```

**type='block'** ディスクデバイスでは、ソースが物理デバイスであることが想定されます。イメージファイルでディスクを使用するには、代わりに **type='file'** を使用します。

# 付録C 更新履歴

| | | |
|---|---|---|
| **改訂 0.5-45**<br>6.9 GA リリースの更新 | Fri Mar 10 2017 | Jiri Herrmann |
| **改訂 0.5-43**<br>6.9 ベータ版リリースの更新 | Mon Jan 02 2017 | Jiri Herrmann |
| **改訂 0.5-42**<br>6.8 GA リリースの複数の更新 | Mon May 02 2016 | Jiri Herrmann |
| **改訂 0.5-41**<br>6.8 ベータリリースの複数の更新 | Tue Mar 01 2016 | Jiri Herrmann |
| **改訂 0.5-40**<br>改訂履歴の整理 | Thu Oct 08 2015 | Jiri Herrmann |
| **改訂 0.5-39**<br>6.7 Beta の再公開。 | Wed Apr 29 2015 | Dayle Parker |