Red Hat Enterprise Linux 5

仮想化ガイド

Red Hat Enterprise Linux の為の Red Hat Virtualization ガイド決定版
エディッション 4

Last Updated: 2017-10-16

Red Hat Enterprise Linux 5 仮想化ガイド

Red Hat Enterprise Linux の為の Red Hat Virtualization ガイド決定版
エディッション 4

Christopher Curran
Red Hat エンジニアリングコンテンツサービス
ccurran@redhat.com

Jan Mark Holzer
Red Hat 進出技術グループ

法律上の通知

Copyright © 2008,2009 Red Hat, Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

Red Hat Enterprise Linux 仮想化ガイドには、Red Hat Enterprise Linux で使用されている 仮想化技術のインストール、設定、管理、ヒント、及びトラブルシューティングについての 情報が含まれています。

目次

序文

このマニュアルは Red Hat Enterprise Linux 仮想化ガイドです。 このガイドには、 Red Hat Enterprise Linux に収納されている 仮想化製品の使用と 管理に関する全てが網羅されています。

1. このマニュアルについて

このマニュアルは 7 部門に分けてあります:

- システムの要件
- インストール
- 設定
- 管理
- 参照事項
- ヒントと裏技
- トラブルシューティング

2. CIO（主席インフォメーションオフィサー）の仮想化に関する観点

著者 Lee Congdon、主席インフォメーションオフィサー、Red Hat, Inc

急速に進展する仮想化技術にすでに大規模な投資をされているかも知れません。その場合は、 以下に示してある提案の一部を考慮してこの技術の更なる活用をしてみて下さい。まだそのような投資をされていない場合は、今が開始するのに最適な時期です。

仮想化は柔軟性の増大とコスト低減の為のツールセットです。これらはどの企業や 情報技術でも重要な項目です。仮想化ソリューションは日増しに利用量が増えており、その機能も豊富になってきています。

仮想化は複数の分野で企業に重要な利便性を与えることが出来るため、パイロット企画を作り、専門家を育成することにより、仮想化技術をすぐに実践できるようにする ことをお薦めします。

改革のための仮想化

基本的に、仮想化はシステムでサポートされているオペレーティングシステムとサービスとアプリケーションを、特定のハードウェアから分離することにより柔軟性を増大します。これにより、共有のハードウェアプラットフォーム上で複数の仮想環境を確立することができるようになります。

改革を検討している企業は、追加のハードウェアを設置することなく（そして必要でなくなった時にはそれらのシステムとサービスを素早く廃止できる）新規のシステムとサービスを構築できる能力が改革への重要な原動力となることがお判りに なるはずです。

実行可能なアプローチとして、カスタムソフトウェアの作成の為の開発システムの迅速な確立、テスト環境を素早くセットアップする能力、大幅なハードウェア投資が不要な 代替ソフトウェアソリューションの装備とそれらの機能比較、迅速なプロトタイプ化と開発環境へのサポート、オンデマンドで対応できる新規プロダクションサービスの迅速な確立能力などが挙げられます。

これらの環境は、企業内部で、又は Amazon の EC2 のように外注で構築できます。 新規の仮想環境を構築するコストはかなり低くなる可能性があり、既存のハードウェアを 活用できるため、改革は最小限の投資で実践できて加速することができます。

仮想化は、また訓練と学習用の仮想環境の使用を通じて改革をサポートする点に於いても優れています。これらのサービスは仮想化技術の理想的な応用となります。学習者は 既知の標準的なシステム環境でコース学習を始めることができます。クラスでの操作は 実稼働ネットワークから隔離することが可能です。このようにして学習者は独占的な ハードウェアリソースの使用を要求することなく、独特なソフトウェア環境を確立 することができます。

仮想化環境の能力は発達を続けるため、特定ユーザーのニーズに適合するような ポータブル環境を有効にできる仮想化活用の日を迎える可能性があります。これらの環境は、ユーザーの存在位置に関係なくアクセス可能な又はローカルの、プロセス環境を動的に移動できるものです。ユーザーの環境はネットワーク上に保存されるか、あるいはポータブルメモリーデバイスに保存されます。

これに関連した概念として、仮想環境内で稼働するように設計されている アプリケーションパッケージ指向のオペレーティングシステムである Appliance Operating System があります。パッケージアプローチは、より低い開発とサポートのコストを維持し、アプリケーションが既知の安全な環境で稼働することを確実にします。Appliance Operating System ソリューションは これらのアプリケーションの開発者と消費者の両方に利便性を提供します。

仮想化技術のこれらのアプリケーションが企業で応用できる手段は各種あります。この技術をすでに上記に示してある分野の複数箇所で使用している場合は、迅速な開発を必要とするソリューションへ追加の投資を考慮してみて下さい。仮想化をまだ始めていない場合は、訓練と学習の実施によって技能を開発し、それからアプリケーションの開発とテストに進んで下さい。仮想化における幅広い経験を有する企業では、ポータブル仮想環境やアプリケーションアプライアンスをの実施を考慮してみて下さい。

コスト削減のための仮想化

仮想化はコスト削減にも利用できます。複数サーバーを統合することにより、仮想化環境の 集合を稼働する小規模でより強力なハードウェアプラットフォームのセットにする点で、 明確な利便性を得ることができます。ハードウェアの規模を減少し、未使用設備を 削減することでコストを低減するだけでなく、より強力なハードウェア上で仮想ゲストを実行することでアプリケーションのパフォーマンスは実際に向上します。

更なる利便性として、稼働を妨害しない方法でハードウェア能力を追加すること、作業負荷を利用可能なリソースに動的に移行できることなどが含まれます。

企業のニーズに応じて、災害からの復元の為の仮想環境を構築することも可能です。 仮想化の導入は同一ハードウェア環境複製の必要性を確実に低減し、そして低コストで災害シナリオのテスト設定も可能にすることができます。

仮想化は、ピーク時の作業負荷やシーズン変化の作業負荷への対応に優秀な ソリューションとなります。組織内に補完の作業能力がある場合、現在最大の作業需要を受けているアプリケーションにリソースを動的に割り当てることができます。組織内で現在供給している作業能力がピーク状態にある場合、オンディマンド式に外部からその余裕能力を仕入れて、仮想化技術を使用してそれを効率良く実装することができます。

サーバー統合によるコスト削減は効率的です。この目的で仮想化を活用されていない 場合は、 今すぐ、この計画を始めるべきです。仮想化の経験が増えるに従って作業負荷の均一化と 仮想化した災害復元環境からの利便性を追求できるようになります。

標準ソリューションとしての仮想化

企業の特定ニーズに関係なく、この技術は間違いなく普及していくことになりますから、 仮想化をシステムとアプリケーションのポートフォリオとして研究してみるべきです。 今後はオペレーティングシステムのベンダーが仮想化を標準のコンポーネントとして含み、 ハードウェアのベンダーがそのプ

ラットフォームに仮想化機能を組込み、仮想化のベンダーがその提供物の範囲を拡張していくと予想されます。

ソリューションアーキテクチャの中にまだ仮想化を統合する計画をお持ちでない場合は、パイロットプロジェクトを立ち上げて、あまり使用していない一部のハードウェアプラットフォームを割り当てて、この柔軟なコスト削減の技術で専門知識を開発するのには今が最適な時期です。その後は仮想ソリューションを統合するターゲットアーキテクチャを拡張していきます。既存のサービスを仮想化するだけでもかなりの利便性を得ますが、統合化した仮想化戦略による新規のアプリケーション構築は管理と可用性の両方で更なる利便性を与えてくれます。

Red Hat の仮想化ソリューションについては、http://www.redhat.com/products/ でもっと知ることが出来ます。

パート I. RED HAT ENTERPRISE LINUX での仮想化に於ける要件と制限

システムの要件、サポートの制約及び制限

この章では、Red Hat Enterprise Linux 上の仮想化に於けるシステムの要件とサポートの制約、 及び制限の概要を提供します。

第1章 システム要件

この章では、Red Hat Enterprise Linux 上で仮想化を正しく稼働するためのシステム要件の 一覧を表示します。仮想化にはまず、仮想化パッケージを持つ Red Hat Enterprise Linux 5 Server を 実行しているシステムが必要です。仮想化パッケージのインストールに 関する情報には、*5章 仮想化パッケージをインストール* をご覧下さい。

最低限のシステム要件

- 6 GB の空きディスク領域
- 2GB の RAM.

推奨されるシステム要件

- 6GB と更にゲスト毎のゲストオペレーティングシステムに推奨されるディスク容量。 ほとんどのオペレーティングシステムには、6GB 以上のディスク容量が推奨されます。
- 各仮想化 CPU 用及び hypervisor 用にプロセッサコア、又はハイパースレッド 1 つずつ
- 2GB の RAM と更に仮想化ゲスト用の 追加の RAM

注記

KVM hypervisor は、仮想化ゲスト用に使用可能な RAM とプロセッサコアよりも多くを使用してゲストをサポートします。KVM での安全なリソースのオーバーコミットに関する 情報については 「KVM でオーバーコミット」 を参照して下さい。

Xen para-virtualization の要件

Para-virtualized ゲストは Red Hat Enterprise Linux 5 のインストールツリーを 必要とします。これは、**NFS** か、**FTP** か、 **HTTP** のプロトコルを使用して 入手できます。

Xen 完全仮想化の要件

Xen Hypervisor を持つ完全仮想化は以下を必要とします:

- Intel VT 拡張を持つ Intel プロセッサ 1 つ、又は
- AMD-V 拡張を持つ AMD プロセッサ 1 つ、又は
- Intel Itanium プロセッサ 1 つ

ご使用のプロセッサが仮想化拡張を持っているかどうかを判定するには、 「仮想化拡張を確認」 を参照して下さい。

KVM の要件

KVM hypervisor は以下を必要とします:

- Intel VT 及び Intel 64 拡張を持つ Intel プロセッサ 1 つ、又は
- AMD-V と AMD64 拡張を持つ AMD プロセッサ 1 つ

ご使用のプロセッサが仮想化拡張を持っているかどうかを判定するには、 「仮想化拡張を確認」 を参照して下さい。

ストレージのサポート

サポートのあるゲストストレージの方法は以下のようになります:

- ローカルストレージ上のファイル
- 物理ディスクのパーティション
- ローカルで接続している物理 LUN
- LVM パーティション
- iSCSI とファイバーチャンネルベースの LUN

重要

ファイルベースのゲストイメージは **/var/lib/xen/images/** フォルダーに格納されるべきです。別のディレクトリを使用する場合は、そのディレクトリを SELinux ポリシーに追加しなければなりません。 詳細については、「SELinux と仮想化」 をご覧下さい。

第2章 XEN の制約とサポート

Red Hat Enterprise Linux 5 はホストと仮想化ゲスト用に各種アーキテクチャの 組み合わせをサポートします。この一覧は、Red Hat Enterprise Linux 5 ホスト用に テストされた互換性のあるゲストを示します。他の組み合わせも可能かも知れませんが、 テストはされておらず、Red Hat によるサポートがありません。

x86 アーキテクチャ

x86 互換のシステム上の 32 bit Red Hat Enterprise Linux kernel-xen は 16 のプロセッサコアに制限されます。

サポートされている完全仮想化ゲスト

オペレーティングシステム	サポートレベル
Red Hat Enterprise Linux 3 x86	最適化
Red Hat Enterprise Linux 4 x86	最適化
Red Hat Enterprise Linux 5 x86	最適化
Windows Server 2000 32-Bit	サポートあり
Windows Server 2003 32-Bit	サポートあり
Windows XP 32-Bit	サポートあり
Windows Vista 32-Bit	サポートあり

注記、他の 32 bit オペレーティングシステムは機能する可能性がありますが、テストは されておらず、サポートはありません。

Red Hat Enterprise Linux 5 で完全仮想化を利用するには、プロセッサが Intel-VT か AMD-V 認識のプロセッサでなければなりません。

サポートのある Para-virtualized ゲスト

オペレーティングシステム	サポートレベル
Red Hat Enterprise Linux 4 x86 Update 5 及びそれ以降	最適化
Red Hat Enterprise Linux 5 x86	最適化

Red Hat Enterprise Linux 5 で para-virtualization を利用するには、プロセッサが PAE（Physical Address Extension）のインストラクションセットを持っていなければなりません。

AMD64 と Intel 64 のアーキテクチャ

AMD64 と Intel 64 のマシン上の kernel-xen パッケージは 以下のようなプロセッサ制限を持ちます:

- Red Hat Enterprise Linux 5.0 は 最大 32 の CPU プロセッサコアをサポートします。
- Red Hat Enterprise Linux 5.1 は最大 32 CPU のプロセッサコアをサポートします。
- Red Hat Enterprise Linux 5.2 は最大 64 CPU のプロセッサコアをサポートします。
- Red Hat Enterprise Linux 5.3 は最大 126 の CPU プロセッサコアをサポートします。
- Red Hat Enterprise Linux 5.4 は最大 256 の CPU プロセッサコアをサポートします。

サポートされている完全仮想化ゲスト

オペレーティングシステム	サポートレベル
Red Hat Enterprise Linux 3 x86-64	最適化
Red Hat Enterprise Linux 3 x86	最適化
Red Hat Enterprise Linux 4 AMD64/Intel 64	最適化
Red Hat Enterprise Linux 4 x86	最適化
Red Hat Enterprise Linux 5 AMD64/Intel 64	最適化
Red Hat Enterprise Linux 5 x86	最適化
Windows Server 2000 32-Bit	サポートあり
Windows Server 2003 32-Bit	サポートあり
Windows XP 32-Bit	サポートあり
Windows Vista 32-Bit	サポートあり
Windows Vista 64-Bit	サポートあり
Windows Server 2008 32-Bit	サポートあり
Windows Server 2008 64-Bit	サポートあり
Solaris 32 bit	サポートあり

他の x86 と AMD64、又は Intel 64 ベースのオペレーティングシステムも機能するかも 知れません。但し、この一覧内に存在しないオペレーティングシステムはテストが 終了しておらず、サポートされていません。

サポートのある Para-virtualized ゲスト

オペレーティングシステム	サポートレベル
Red Hat Enterprise Linux 4 AMD64/Intel 64 Update 5 及びそれ以降	最適化
Red Hat Enterprise Linux 4 x86 Update 5 及びそれ以降	5.2 と 5.3 内で技術プレビュー。5.4 とそれ以降で全面的サポート。
Red Hat Enterprise Linux 5 AMD64/Intel 64	最適化
Red Hat Enterprise Linux 5 x86	5.2 と 5.3 内で技術プレビュー。5.4 とそれ以降で全面的サポート。

Intel Itanium アーキテクチャ

Itanium システム上の kernel-xen パッケージは 32 の プロセッサコアまでに制限されています。

サポートされている完全仮想化ゲスト

オペレーティングシステム	サポートレベル
Red Hat Enterprise Linux 3 Itanium	サポートあり
Red Hat Enterprise Linux 4 Itanium	最適化
Red Hat Enterprise Linux 5 Itanium	最適化
Windows Server 2003、 Itanium ベースのシステム対応	サポートあり

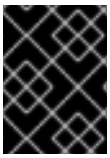
サポートのある Para-virtualized ゲスト

オペレーティングシステム	サポートレベル
Red Hat Enterprise Linux 5 Itanium	最適化

注記

Intel Itanium アーキテクチャ上で Xen hypervisor を使用した仮想化には、 ゲストファームウェアイメージパッケージが必要になります。詳細は **yum** を使用して Xen hypervisor をインストール を参照して下さい。

第3章 KVM の制約とサポート



重要

KVM は、Red Hat Enterprise Linux の AMD64 バージョンと Intel 64 バージョンのみをサポートします。他のアーキテクチャは現時点ではサポートがありません。

Red Hat Enterprise Linux kvm パッケージは 64 プロセッサコアに 制限されています。制限が 96 プロセッサコアとなる特定のプラットフォーム以外では、 この制限が適用されます。

サポートのある完全仮想化ゲスト

オペレーティングシステム	サポートのレベル
Red Hat Enterprise Linux 3 x86	para-virtualized ドライバーでの最適化
Red Hat Enterprise Linux 4 x86	para-virtualized ドライバーでの最適化
Red Hat Enterprise Linux 4 AMD 64 及び Intel 64	para-virtualized ドライバーでの最適化
Red Hat Enterprise Linux 5 x86	para-virtualized ドライバーでの最適化
Red Hat Enterprise Linux 5 AMD 64 及び Intel 64	para-virtualized ドライバーでの最適化
Windows Server 2003 R2 32-Bit	para-virtualized ドライバーでの最適化
Windows Server 2003 R2 64-Bit	para-virtualized ドライバーでの最適化
Windows Server 2003 Service Pack 2 32-Bit	para-virtualized ドライバーでの最適化
Windows Server 2003 Service Pack 2 64-Bit	para-virtualized ドライバーでの最適化
Windows XP 32-Bit	para-virtualized ドライバーでの最適化（ネットワークドライバーのみ）
Windows Vista 32-Bit	サポート有り
Windows Vista 64-Bit	サポート有り
Windows Server 2008 32-Bit	para-virtualized ドライバーでの最適化
Windows Server 2008 64-Bit	サポート有り

第4章 仮想化の制限

この章では、Red Hat Enterprise Linux の仮想化パッケージに於けるその他の 制限を説明しています。

4.1. 仮想化に於ける一般的制限

hypervisor の切り替え

現時点では、Xen ベースのゲストを KVM に切り替えたり、KVM ベースのゲストを Xen に 切り替えたりするアプリケーションは存在しません。ゲストはそれが作成された hypervisor のタイプ上でのみ使用できます。このタスクを自動化するようなアプリケーションの 開発がこのマニュアル作成の時点で開発中です。このアプリケーションは Red Hat Enterprise Linux の 将来のバージョンでリリースされるかも知れません。

他の制限

仮想化に影響を与える他の制限や問題の全ての一覧を見るには、ご使用のバージョンの 『Red Hat Enterprise Linux リリースノート』 をお読み下さい。 『リリースノート』は現在の新しい機能や既知の問題や制限などを その更新や発見の時点で提供します。

導入前のテスト

大負荷の I/O アプリケーションを導入する前に、予期される最大負荷と仮想化ネットワークのストレスをテストすべきです。増加した I/O を使用する仮想化の影響でパフォーマンスが低下するため、ストレステストは重要になります。

4.2. KVM の制限

以下の制限が KVM hypervisor に適用されます:

不変 TSC ビット

不変タイムスタンプカウンタ（Constant Time Stamp Counter）のないシステムは 追加の設定を必要とします。関連した問題を修復するためにこの不変タイムスタンプカウンタが あるかどうかを判定するための詳細には 17章*KVM ゲストのタイミング管理*を参照して下さい。

メモリーオーバーコミット

KVM はメモリーオーバーコミットをサポートし、ゲストメモリーをスワップ内に格納 できます。ゲストは何回もスワップされると実行が遅くなります。KSM (Kernel SamePage Merging) が使用されるときは、スワップ サイズがオーバーコミット比率のサイズであることを確認して下さい。

CPU オーバーコミット

物理プロセッサコア毎に10以上の仮想 CPU を持つことはサポートされていません。 そして物理プロセッサコアの数を越えたオーバーコミットの仮想 CPU の数は 特定の仮想化ゲストで問題になる可能性があります。

CPU のオーバーコミットはリスクを伴い、不安定の原因になります。CPU のオーバーコミットに関する ヒントと推薦については 「KVM でオーバーコミット」 を参照して下さい。

仮想化 SCSI デバイス

SCSI 模倣は現時点ではサポートされていません。仮想化 SCSI デバイスは KVM 内では 完全に無効になっています。

仮想化 IDE デバイス

KVM はゲスト毎に最大６つの仮想化（模倣）IDE デバイスに制限されています。

Para-virtualized デバイス

Para-virtualized デバイスは **virtio** ドライバーと PCI デバイスを使用します。現時点では、ゲストは最大 32 PCI デバイスまでに 制限されています。一部の PCI デバイスはゲストの実行に重要なものであり、これらの デバイスは削除できません。デフォルトで以下のようなデバイスが必須です:

- ホストブリッジ
- ISA ブリッジと usb ブリッジ（usb と isa のブリッジは同じデバイスです）
- グラフィックカード（Cirrus か qxl のドライバーを使用）
- メモリーバルーンデバイス

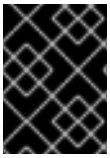
ゲスト用に最大利用可能な32 の PCI デバイスの内、４つは削除できません。このことは、 ゲスト毎に、28 PCI スロットのみが追加デバイス用に利用できると言うことです。それぞれの para-virtualized ネットワーク、又はブロックデバイスが１つのスロットを使用します。 そのため、各ゲストは para-virtualized ネットワークと para-virtualized ディスクデバイスと VT-d を使用している他の PCI デバイスの組合せから 28 までの追加デバイスを使用できることになります。

移行の制限

ライブ移行は同じベンダー（即ち、Intel と Intel、あるいは AMD と AMD）のみからの CPU 群で 可能になります。

ライブ移行には、No eXecution (NX) ビットは 両方の CPU 用にオンでもオフでもセット しなければなりません。

4.3. XEN の制限

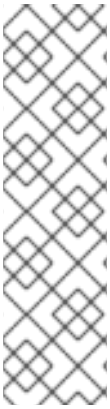


重要

この章内の全ての制限は、注記がある箇所以外はすべて Red Hat Enterprise Linux 5.4 の為の制限です。旧来のバージョンでは数的により少ない制限を持つ可能性が あります。

Xen ホスト (dom0) の制限

- ホスト毎に **tap:aio** ドライバーとファイルベースのデバイスを 使用する 100 のブロックデバイスまでの制限があります。para-virtualized ゲストに添付された ブロックデバイスの合計数はホスト毎に100デバイスを超過することは出来ません。



注記

para-virtualized デバイスの制限問題を回避するのには、２つの方法があります。１つは **phy** デバイスの使用で、もう１つはゲスト上で LVM の 使用です。

ホストが十分なリソースを持っていれば、それが所有できる **phy** デバイスの数には制限がありません。

LVM、又は同様な論理パーティション設定ツールは、ブロックデバイス上で使用して 単独の para-virtualized ブロックデバイスに追加の論理パーティションを作成する ことができます。

Xen Para-virtualization の制限

- 仮想化ゲスト毎に最大合計 256 デバイス
- 仮想化ゲスト毎に最大 15 のネットワークデバイス

Xen 完全仮想化の制限

- ゲスト毎に最大 4 つの仮想化（模倣）IDE デバイス

  完全仮想化ゲスト用に para-virtualized ドライバーを使用するデバイスは この制限を受けません。

- 仮想化した（模倣）IDE デバイスはシステムでサポートされるループバック デバイスの合計数により制限されます。Red Hat Enterprise Linux 5.4 で利用できる ループバックデバイスのデフォルトの数は 8 です。このことは、システム上の全ての 仮想化ゲストは合計で 8 つ以上の仮想化（模倣）IDE デバイスを持てないことになります。

  ループバックデバイスの詳細情報には、Red Hat KnowledgeBase を 参照して下さい。

注記

デフォルトでは、Red Hat Enterprise Linux は利用可能なループバックデバイスの 数を制限しています。この数はカーネルの制限を解除することにより増加することが できます。

/etc/modprobe.conf 内で、以下の行を追加します:

```
options loop max_loop=64
```

マシンをリブートするか、又は以下のコマンドでカーネルを更新して下さい:

```
# rmmod loop
# modprobe loop
```

- ホスト毎に 100 の para-virtualized ブロックデバイスの制限となります。para-virtualized ゲストに添付されたブロックデバイス（**tap:aio** ドライバーを 使用）の合計数は 100 デバイスを超過できません。
- para-virtualized ドライバーを使用して仮想化ゲスト毎に最大 256 ブロックデバイスです。
- 仮想化ゲスト毎に最大 15 のネットワークデバイス
- 仮想化ゲスト毎に最大 15 仮想化 SCSI デバイスです。

4.4. アプリケーションの制限

特定タイプのアプリケーションに対して仮想化が不適切になるような 仮想化の側面がいくつか存在します。

高い I/O スループットの要求を持つアプリケーションは、完全仮想化用の para-virtualized ドライバーを使用すべきです。para-virtualized ドライバーが無いと、 特定のアプリケーションは高い I/O 負荷では不安定になります。

以下のアプリケーションはそれらの高度な I/O 要求の為に回避すべきです:

- **kdump** サーバー
- **netdump** サーバー

仮想化ゲストでデータベースアプリケーションを使用する前に、そのアプリケーションを 注意して検証すべきです。データベースは一般的にネットワークとストレージ I/O デバイスを 集中的に使用します。これらのアプリケーションは完全仮想化環境には不適切かも知れません。 I/O パフォーマンスの増強のためには、 para-virtualization 又は para-virtualized ドライバーを 考慮してください。完全仮想化用の para-virtualized ドライバーについての詳細は 14章*Xen Para-virtualized ドライバー*でご覧下さい。

I/O を大幅に活用するか、又はリアルタイムパフォーマンスを要求する 他のアプリケーションやツールは慎重に査定すべきです。para-virtualized を 使用した完全仮想化の使用（14章*Xen Para-virtualized ドライバー* 参照）、あるいは、para-virtualization の 使用は、I/O 集中型アプリケーションでより良い結果を出します。アプリケーションは それでもまだ仮想化環境での実行に少々のパフォーマンスロスを被ります。完全仮想化のハードウェアの 使用に関連したアプリケーションのパフォーマンス問題の可能性に対して、より新しく、 より速いハードウェアへの統合を通じた仮想化の利便性を査定する必要があります。

パート II. インストール

仮想化インストールのトピック

これらの章では、**Red Hat Enterprise Linux** でのホストのセットアップと 仮想化ゲストのインストールを説明しています。これらの章を注意深く読むことで 仮想化ゲストオペレーティングシステムのインストールの達成を確実にすることが 推奨されます。

第5章 仮想化パッケージをインストール

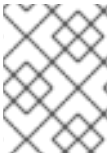
仮想化を使用できるようにするためには、仮想化パッケージが Red Hat Enterprise Linux 上に インストールされている必要があります。仮想化パッケージはインストールシーケンスの 途中か、又は **yum** コマンドと Red Hat Network (RHN) を 使用してインストールの後にインストールすることが出来ます。

単一のシステムに KVM hypervisor と Xen hypervisor の両方をインストールすることが できます。Xen hypervisor は kernel-xen パッケージを使用し、 KVM hypervisor は kvm カーネルモジュールを持つデフォルトの Red Hat Enterprise Linux カーネルを使用します。Xen と KVM はそれぞれ異なるカーネルを 使用し、任意の時点にどちらかの hypervisor の１つのみがアクティブになることができます。 Red Hat では、ユーザーが仮想化に使用する予定の hypervisor の一方だけをインストールする ことを推奨します。

hypervisor を Xen から KVM へ、又は KVM から Xen へ変更するには、「KVM と Xen hypervisor との間での切り替え」 を参照して下さい。

5.1. RED HAT ENTERPRISE LINUX の新規インストールで XEN を インストール

このセクションは、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 Xen パッケージのインストールを説明しています。



注記

『インストールガイド』（redhat.com から入手可能）は Red Hat Enterprise Linux のインストールの詳細を取り扱います。

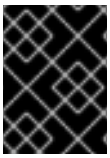
1. Red Hat Enterprise Linux のインストール CD-ROM、DVD、あるいは PXE から 対話式の Red Hat Enterprise Linux インストールを開始します。
2. プロンプトが出たら、有効なインストール番号を入力して、仮想化と他の高度なプラットフォーム パッケージへのアクセス権を受理するようにします。
3. パッケージ選択のステップまで他のステップを完了して下さい。

仮想化（Virtualization） パッケージグループを選択して、**今、カスタマイズする（Customize Now）** ラジオボタンを選択します。

4. **Virtualization** パッケージグループを選択します。**Virtualization** パッケージグループは Xen hypervisor、**virt-manager**、**libvirt**、及び **virt-viewer**、それにインストール用のすべての依存関係を選択します。

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

閉じる をクリックしてから、**次** を クリックしてインストールを継続します。



重要

仮想化パッケージに更新を受け取るには、有効な RHN 仮想化のエンタイトルメントが必要です。

キックスタートファイルで Xen パッケージをインストール

このセクションでは、Xen hypervisor パッケージを用いた Red Hat Enterprise Linux のインストールに於けるキックスタートファイルの使用法を 説明します。キックスタートファイルは各システムにユーザーの手動インストール無しで 大規模な自動化したインストールを可能にします。このセクション内のステップは、仮想化パッケージを持つ Red Hat Enterprise Linux のインストールの為にユーザーがキックスタートファイルの作成とその使用をするためのお手伝いをします。

使用する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記します:

```
%packages
@virtualization
```

注記

Itanium® アーキテクチャ上の 完全仮想化には、ゲストファームウェアイメージパッケージ（**xen-ia64-guest-firmware**）が 必要です。使用するキックスタートファイルにこのパッケージを追記します。

```
xen-ia64-guest-firmware
```

キックスタートファイルに関する詳細情報は Red Hat のウェブサイト：redhat.com にある 『インストールガイド』 をご覧下さい。

5.2. 既存の RED HAT ENTERPRISE LINUX システムに XEN パッケージをインストール

このセクションは、稼働可能な Red Hat Enterprise Linux システムに仮想化パッケージを インストールするのに必要なステップを説明しています。

Red Hat Network エンタイトルメントの一覧にパッケージを追加

このセクションでは、仮想化パッケージ用の Red Hat Network (RHN) エンタイトルメントを 有効にする方法を説明します。Red Hat Enterprise Linux 上に仮想化パッケージをインストールして それを更新するのにこれらのエンタイトルメントが必要になります。Red Hat Enterprise Linux に 仮想化パッケージをインストールするためには、有効な Red Hat Network アカウントが要求されます。

更に、ご使用のマシンが RHN で登録される必要があります。未登録の Red Hat Enterprise Linux インストールを 登録するには、**rhn_register** コマンドを実行して、プロンプトに従います。

有効な Red Hat サブスクリプション（購読）をお持ちでない場合は、 Red Hat オンラインストア をご覧ください。

手順5.1 RHN を使用して仮想化エンタイトルメントを追加

1. お持ちの RHN ユーザー名とパスワードを使用して RHN に ログインします。
2. 仮想化のインストール先となるシステムを選択します。
3. **システムのプロパティ** セクションでは、現在のシステム用の エンタイトルメントが**エンタイトルメント** のヘッディングの 横に一覧表示してあります。**このプロパティを編集** リンクを 使用するとエンタイトルメントを変更することが出来ます。
4. **Virtualization** チェックボックスを選択します。

ご使用のシステムはこれで、仮想化パッケージを受け取るエンタイトルメントを 持つようになりました。次のセクションでは、これらのパッケージのインストール法を説明しています。

yum を使用して Xen hypervisor をインストール

Red Hat Enterprise Linux 上で仮想化を使用するには、**xen** と **kernel-xen** のパッケージが必要になります。**xen** パッケージには 、 Xen hypervisor と 基本的仮想化の ツールが含まれています。**kernel-xen** パッケージには、 Xen hypervisor 上で仮想マシンゲストとして稼働する修正した linux カーネルが含まれています。

xen と **kernel-xen** のパッケージを インストールするには、以下を実行します:

```
# yum install xen kernel-xen
```

Itanium® アーキテクチャ上の 完全仮想化ゲストには、補足のインストール DVD からゲストファームウェアイメージ パッケージ (**xen-ia64-guest-firmware**) が必要になります。 このパッケージは **yum** コマンドを使用して RHN から インストールすることもできます:

```
# yum install xen-ia64-guest-firmware
```

管理と設定のために追加の仮想化パッケージをインストールすることが推奨されます。 推奨される仮想化パッケージ: は推奨パッケージを一覧表示しています。

推奨される仮想化パッケージ:

python-virtinst

仮想マシンの作成用に **virt-install** コマンドを提供します。

libvirt

libvirt は、hypervisor と通信する API ライブラリです。 **libvirt** は仮想マシンの 管理と制御の為に **xm** 仮想化 フレームワークと **virsh** コマンドラインツールを使用します。

libvirt-python

libvirt-python パッケージには、Python プログラミング言語内に書き込まれている アプリケーションが **libvirt** API で供給され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

仮想マシンマネージャ として知られる **virt-manager** は 仮想マシンの管理の為のグラフィカルなツールを提供します。これは、管理 API として **libvirt** ライブラリを使用します。

推奨されるその他の仮想化パッケージをインストールします:

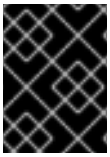
```
# yum install virt-manager libvirt libvirt-python python-virtinst
```

5.3. RED HAT ENTERPRISE LINUX の新規インストールで KVM をインストール

このセクションは 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 KVM のインストールを説明しています。

注記

『インストールガイド』（redhat.com から入手可能）は Red Hat Enterprise Linux のインストールの詳細を取り扱います。



重要

有効なインストール番号が無いとインストール中に仮想化パッケージを選択できません。

1. Red Hat Enterprise Linux のインストール CD-ROM、DVD、あるいは PXE から 対話式の Red Hat Enterprise Linux インストールを開始します。

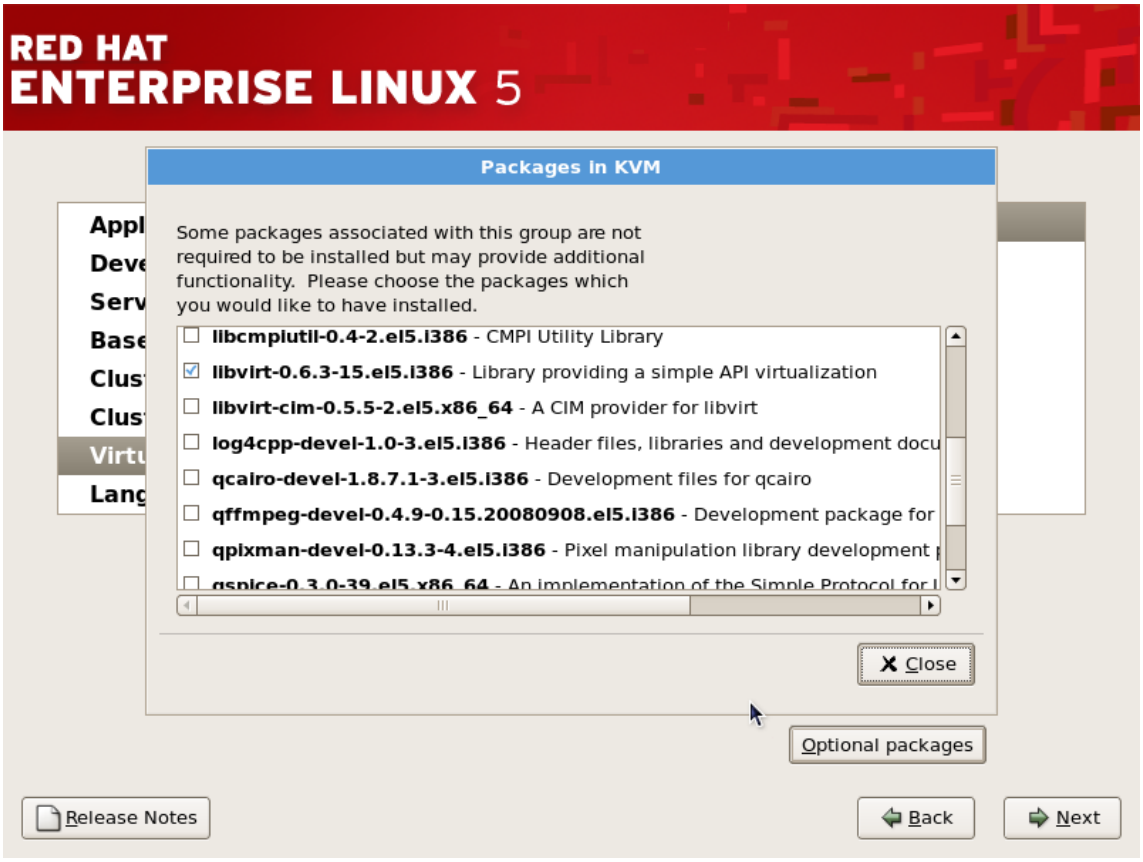
2. プロンプトが出たら、有効なインストール番号を入力して、仮想化と他の高度なプラットフォーム パッケージへのアクセス権を受理するようにします。

3. パッケージ選択のステップまで他のステップを完了して下さい。

仮想化（Virtualization） パッケージグループを選択して、**今、カスタマイズする（Customize Now）** ラジオボタンを選択します。

4. **KVM** パッケージグループを選択します。**Virtualization** パッケージグループを選択解除します。これでインストールのために KVM hypervisor、**virt-manager**、 **libvirt**、及び **virt-viewer** を選ぶことになります。

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



閉じる をクリックしてから、**次** を クリックしてインストールを継続します。

重要

仮想化パッケージに更新を受け取るには、有効な RHN 仮想化のエンタイトルメントが必要です。

キックスタートファイルで KVM パッケージをインストール

このセクションは KVM hypervisor パッケージを用いた Red Hat Enterprise Linux をインストールをするためのキックスタートファイルの使用法を説明します。 キックスタートファイルは、各個別のシステム用にユーザーの手動インストール無しで大規模な 自動化インストールを可能にします。このセクション内のステップは、仮想化パッケージを持つ Red Hat Enterprise Linux のインストールの為にユーザーが キックスタートファイルの作成とその使用をするためのお手伝いをします。

使用する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記します:

```
%packages
@kvm
```

キックスタートファイルに関する詳細情報は Red Hat のウェブサイト：redhat.com にある 『インストールガイド』 をご覧下さい。

5.4. 既存の RED HAT ENTERPRISE LINUX システムに KVM パッケージをインストール

このセクションでは、稼働可能な Red Hat Enterprise Linux 5.4 又はそれ以降のシステムに KVM hypervisor をインストールするためのステップを説明しています。

Red Hat Network エンタイトルメントの一覧にパッケージを追加

このセクションでは、仮想化パッケージ用の Red Hat Network (RHN) エンタイトルメントを 有効にする方法を説明します。Red Hat Enterprise Linux 上に仮想化パッケージをインストールして それを更新するのにこれらのエンタイトルメントが必要になります。Red Hat Enterprise Linux に 仮想化パッケージをインストールするためには、有効な Red Hat Network アカウントが要求されます。

更に、ご使用のマシンが RHN で登録される必要があります。未登録の Red Hat Enterprise Linux インストールを 登録するには、**rhn_register** コマンドを実行して、プロンプトに従います。

有効な Red Hat サブスクリプション（購読）をお持ちでない場合は、 Red Hat オンラインストア をご覧ください。

手順5.2 RHN を使用して仮想化エンタイトルメントを追加

1. お持ちの RHN ユーザー名とパスワードを使用して RHN に ログインします。
2. 仮想化のインストール先となるシステムを選択します。
3. **システムのプロパティ** セクションでは、現在のシステム用の エンタイトルメントが**エンタイトルメント** のヘッディングの 横に一覧表示してあります。**このプロパティを編集** リンクを 使用するとエンタイトルメントを変更することが出来ます。
4. **Virtualization** チェックボックスを選択します。

ご使用のシステムはこれで、仮想化パッケージを受け取るエンタイトルメントを 持つようになりました。次のセクションでは、これらのパッケージのインストール法を説明しています。

yum を使用して KVM hypervisor をインストール

Red Hat Enterprise Linux 上で仮想化を使用するには、**kvm** パッケージが必要になります。**kvm** パッケージには 、KVM カーネルモジュールが含まれており、 デフォルトの Red Hat Enterprise Linux カーネル上に KVM hypervisor を提供します。

kvm パッケージをインストールするには、以下を実行します:

```
# yum install kvm
```

ここで、追加の仮想化管理パッケージをインストールします。

推奨される仮想化パッケージ:

python-virtinst

仮想マシンの作成用に **virt-install** コマンドを提供します。

libvirt

libvirt は、hypervisor と通信する API ライブラリです。 **libvirt** は仮想マシンの 管理と制御の為に **xm** 仮想化 フレームワークと **virsh** コマンドラインツールを使用します。

libvirt-python

libvirt-python パッケージには、Python プログラミング言語内に書き込まれている アプリケーションが **libvirt** API で供給され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

仮想マシンマネージャ として知られる **virt-manager** は 仮想マシンの管理の為のグラフィカルなツールを提供します。これは、管理 API として **libvirt** ライブラリを使用します。

推奨されるその他の仮想化パッケージをインストールします:

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

第6章 仮想化ゲストインストールの概要

仮想化パッケージをホストシステム上にインストールした後は、ゲスト オペレーティングシステムの作成が可能になります。この章では、仮想マシン上に ゲストオペレーティングシステムをインストールする為の手順を説明します。 ゲストを作成するには、**virt-manager** 内の **新規**ボタンをクリックするか、又はコマンドライン インターフェイス **virt-install** を使用します。 これらの両方の方法はこの章で説明されています。

インストール案内の詳細は、Red Hat Enterprise Linux の各バージョン用、他の Linux ディストリビューション用、Solaris 用、そして Windows 用に取得できます。それぞれの 手順については *7章ゲストオペレーティングシステムのインストール手順* を参照して下さい。

6.1. VIRT-INSTALL を使用してゲストを作成

コマンドラインから **virt-install** コマンドを使用して、 仮想化ゲストを作成することができます。**virt-install** は 対話式か、あるいはスクリプトによる自動化で仮想マシンを作成することが できます。キックスタートファイルで **virt-install** を使用すると 仮想化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のできる 多くのオプションを提供します。それらのオプションの完全一覧を見るには、次を実行します:

```
$ virt-install --help
```

virt-install man ページはコマンドオプションと重要な変数を それぞれドキュメント化しています。

qemu-img は関連したコマンドであり、ストレージオプションを 設定する為に**virt-install** の前に使用することができます。

重要なオプションの１つは ***--vnc*** オプションです。これは ゲストインストールの為にグラフィカルウィンドウを開きます。

例6.1 KVM を使用した virt-install で Red Hat Enterprise Linux 3 のゲストを作成

この例では、仮想ネットワーキングと 5 GB のファイルベースブロックデバイスイメージを 使用して、CD-ROM から ***rhel3support*** と言う名前の Red Hat Enterprise Linux 3 ゲストを作成します。この例は KVM hypervisor を使用します。

```
# virt-install --accelerate --hvm --connect qemu:///system \
        --network network:default \
        --name rhel3support --ram=756\
        --file=/var/lib/libvirt/images/rhel3support.img \
        --file-size=6 --vnc --cdrom=/dev/sr0
```

例6.2 virt-install を使用して fedora 11 ゲストを作成

```
# virt-install --name fedora11 --ram 512 --
file=/var/lib/libvirt/images/fedora11.img \
        --file-size=3 --vnc --
cdrom=/var/lib/libvirt/images/fedora11.iso
```

-

6.2. VIRT-MANAGER を使用してゲストを作成

仮想マシンマネージャとして知られる **virt-manager** は 仮想化ゲストの作成と管理の為のグラフィカルツールです。

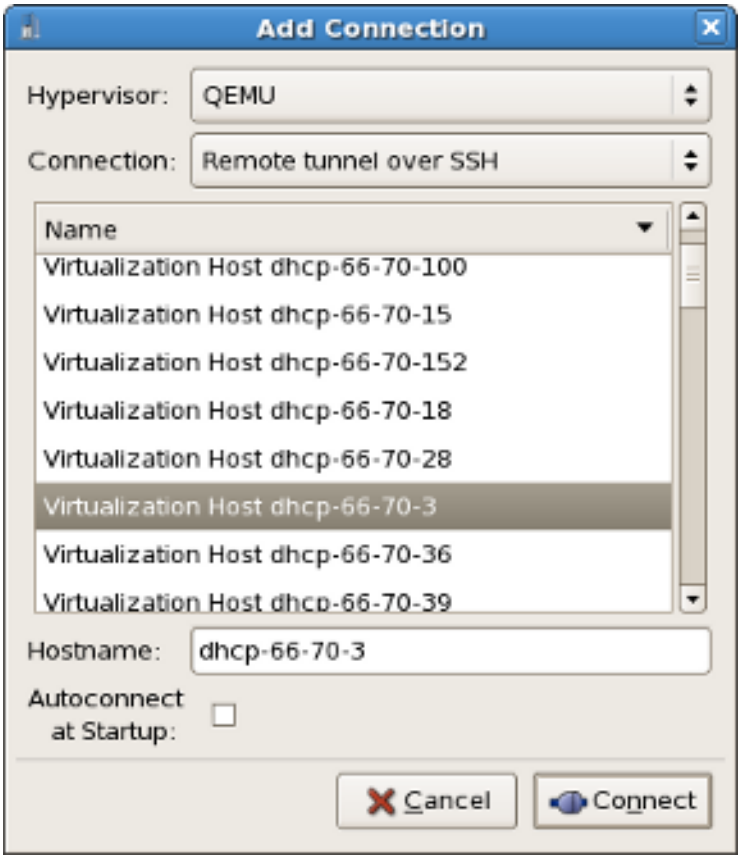
手順6.1 virt-manager を使用して仮想化ゲストを作成

1. **virt-manager** を開始するには、root として 以下のコマンドを実行します:

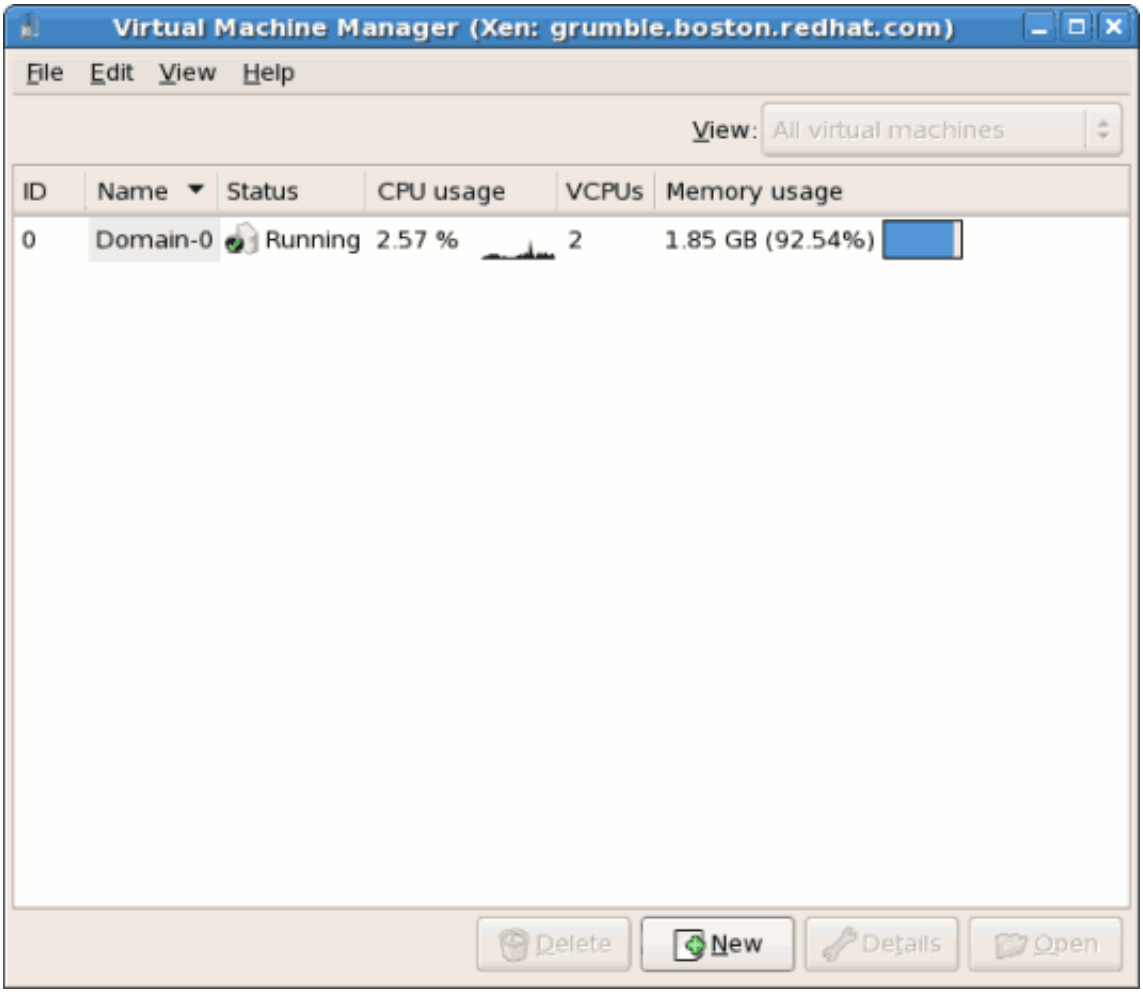
```
# virt-manager &
```

virt-manager コマンドはグラフィカルユーザーインターフェイスの ウィンドウを開きます。root 権限や **sudo** 設定を持たないユーザーは **新規** ボタンを含む各種機能が利用できず、新しい仮想化ゲストを 作成することが出来ません。

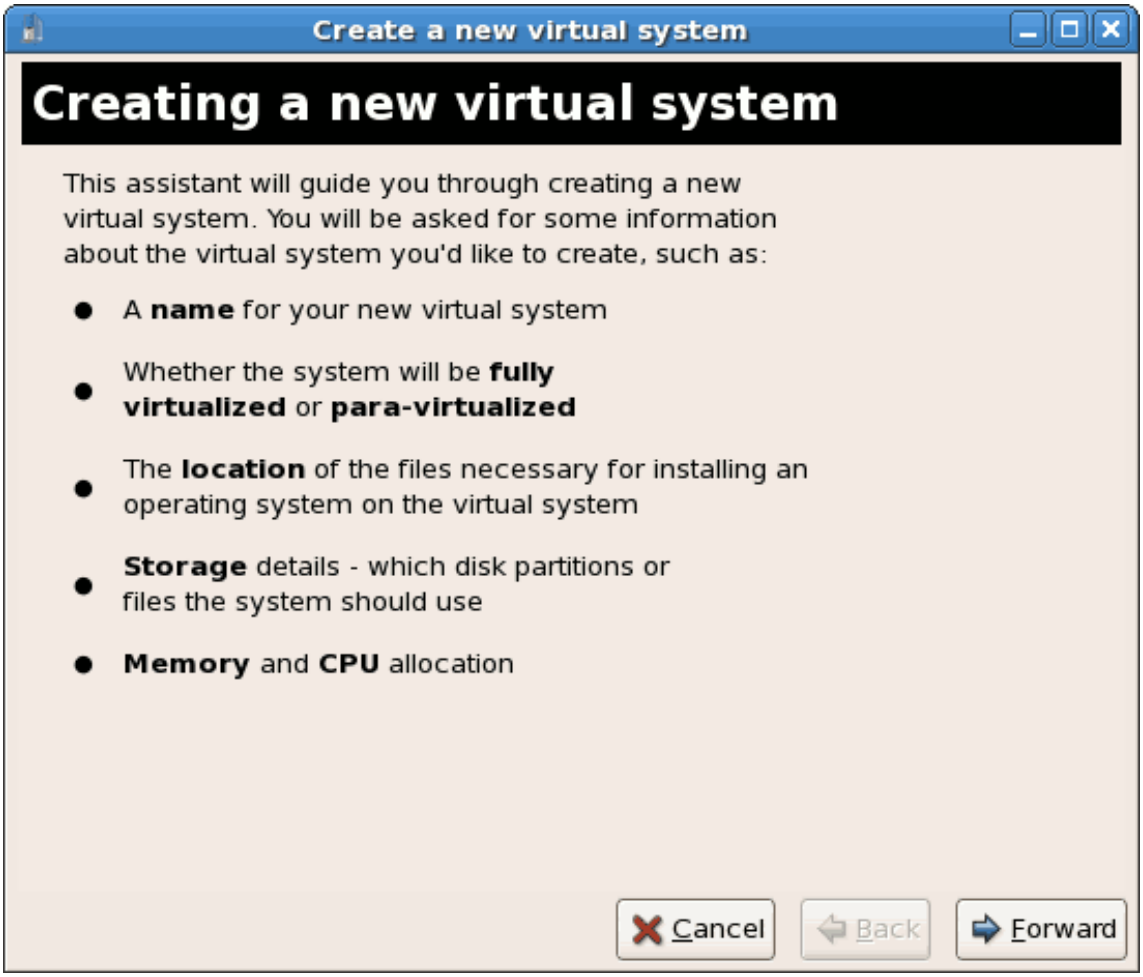
2. **ファイル -> 接続開始** を開きます。以下のダイアログ ボックスが開きます。hypervisor を選択して、**接続** ボタンをクリックします。



3. **virt-manager** ウィンドウでは、新しい 仮想マシンの作成ができます。**新規** ボタンを使用して新しいゲストを作成します。これによりスクリーンショットで示してあるウィザードが 開きます。



4. **新規仮想システムを作成** ウィンドウは、仮想マシンを 作成するためにユーザーが用意する必要のある情報の要約を提供します:

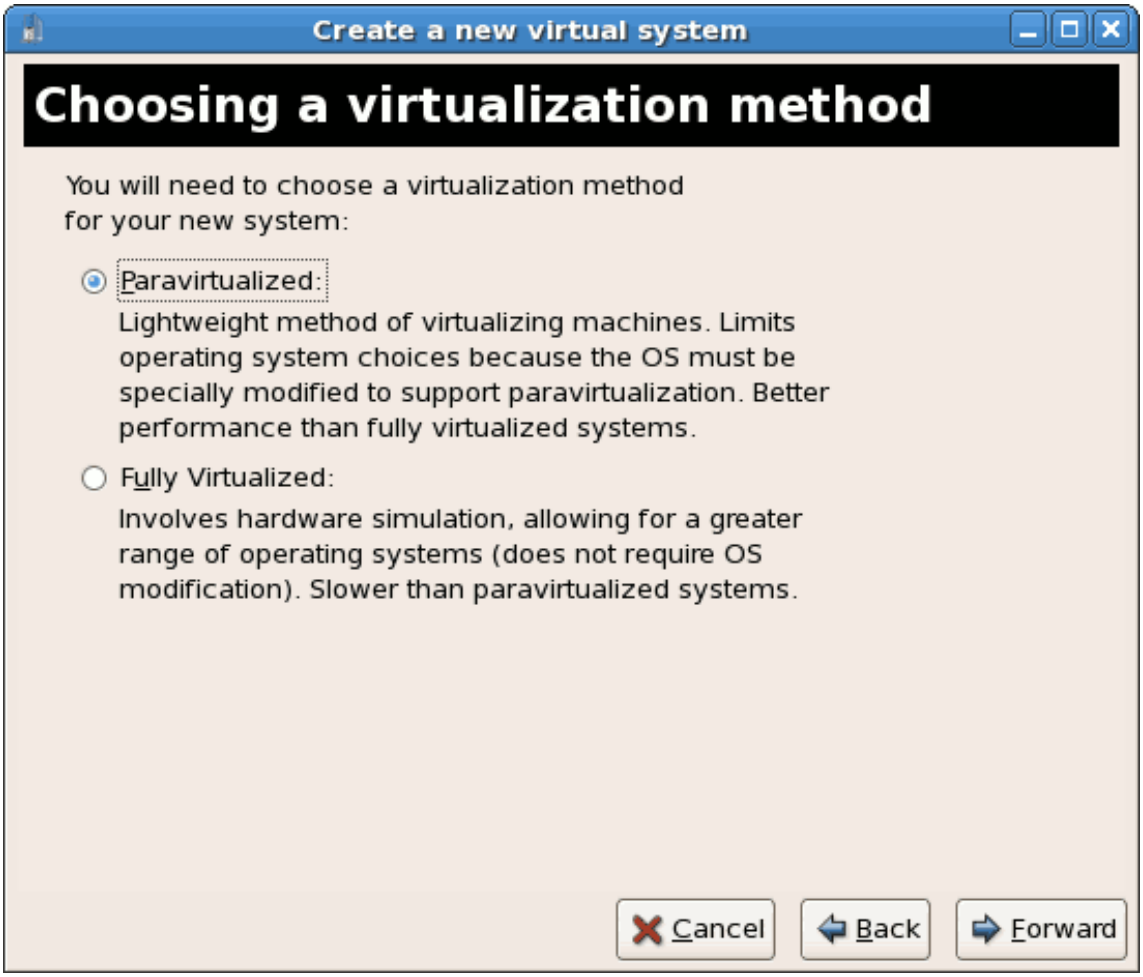


使用するインストールの情報を再確認して、**進む** ボタンを クリックします。

5. **仮想化の方法を選択** ウィンドウが出てきます。 **Para-virtualized** か **完全仮想化** の どちらかを選択して下さい。

   完全仮想化には、Intel® VT、又は AMD-V プロセッサが必要になります。仮想化の拡張が 存在しない場合は、**完全仮想化** ラジオボタン、あるいは**カーネル/ハードウェアアクセラレーションを有効にする** は選択 出来ません。 **Para-virtualized** オプションは、**kernel-xen** が その時点のカーネルとして稼働していない場合は、灰色になって使用不可になっています。

   KVM hypervisor に接続すると、完全仮想化のみが利用可能になります。



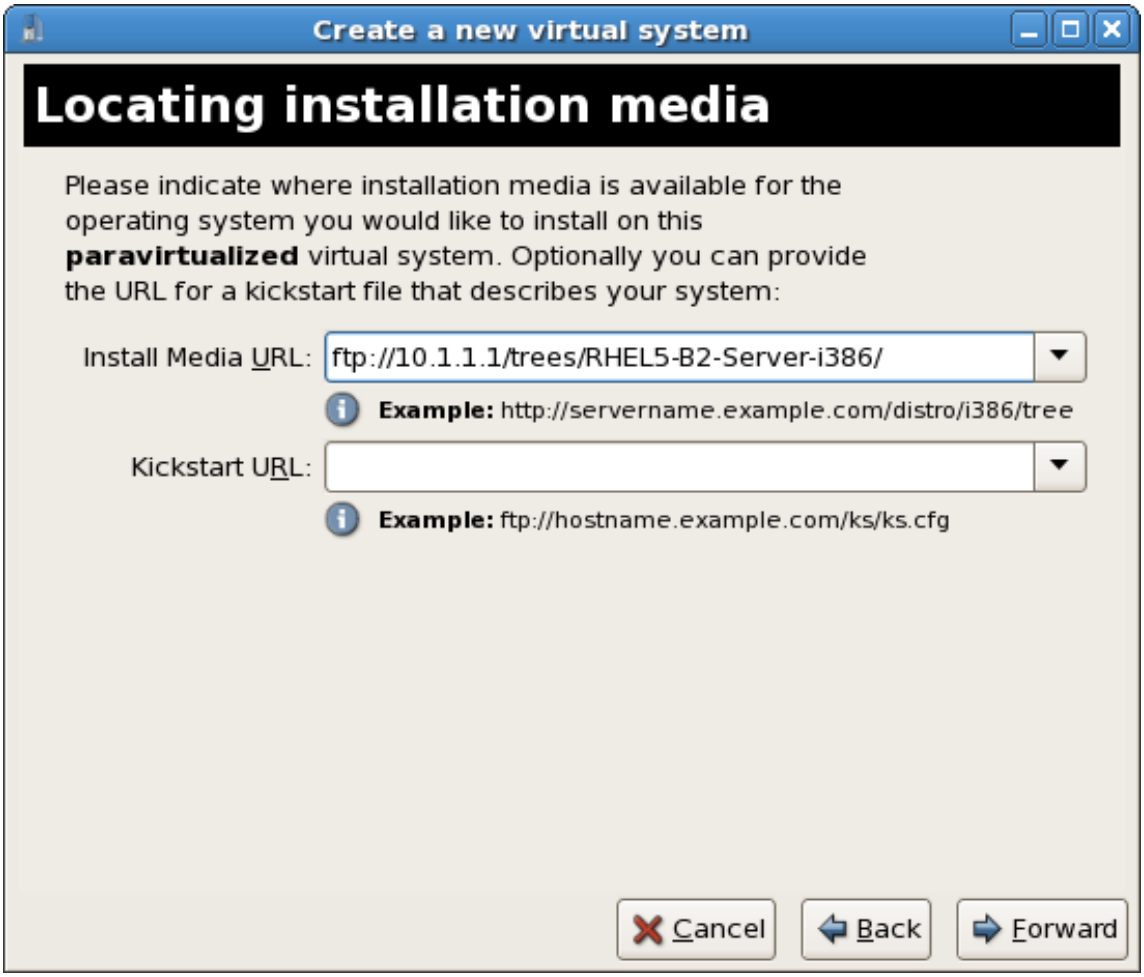
仮想化のタイプを選択したら、**次** ボタンをクリックします。

6. **インストールメディアの検出** プロンプトは選択されている インストールタイプ用のインストールメディアを求めてきます。この画面は 前のステップで選択したものに依存します。

   a. para-virtualized のインストールには、以下のネットワークプロトコルの１つを使用してインストールツリーへのアクセスが必要になります: **HTTP**、**FTP**、又は **NFS**。インストールメディアの URL は Red Hat Enterprise Linux installation ツリーを含んでいなければなりません。このツリーは **NFS**、**FTP**、 又は **HTTP** を使用してホストされています。ネットワーク サービスとファイルは、ホスト上、又は別のミラー上でネットワークサービスを使用してホストできます。

   CD-ROM か DVD イメージ（**.iso** ファイルとしてタグ付き）を使用。 CD-ROM イメージをマウントしてマウントしたファイルを前述のプロトコルの１つでホストします。

   別の方法として、Red Hat Enterprise Linux のミラーからインストールツリーをコピーします。



b. 完全仮想化インストールはブート可能なインストール DVD か、CD-ROM か、あるいはブート可能なインストール DVD / CD-ROM のイメージ（ファイルタイプ .iso 又は .img）をローカルで必要とします。Windows インストールは DVD か、 CD-ROM か、.iso ファイルを 使用します。Linux 及び UNIX-タイプのオペレーティングシステムの多くは .iso ファイルを 使用してベースシステムをインストールしてから、ネットワークベースのインストール ツリーでのインストールを終了します。

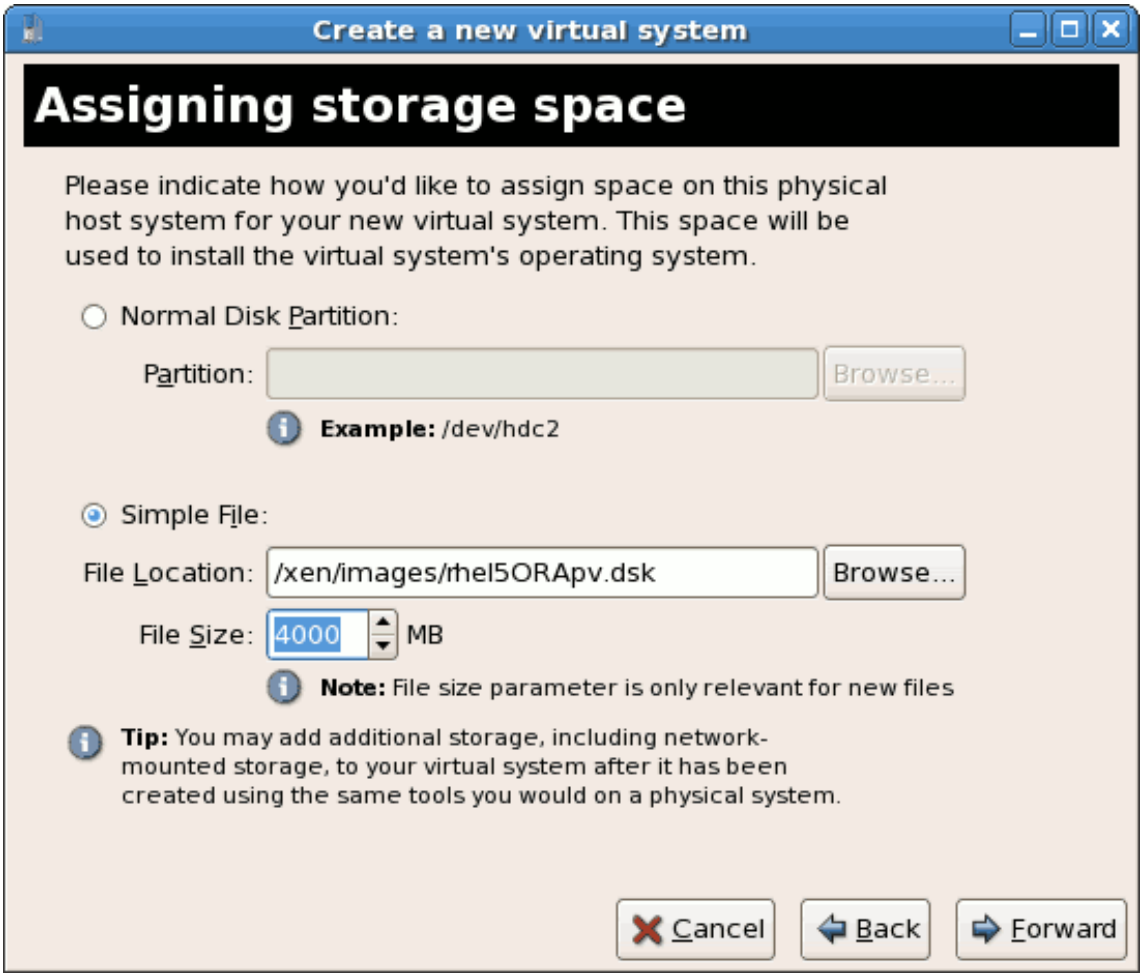
適切なインストールメディアを選択してから、**進む** ボタンをクリックします。

7. **ストレージスペースの割り当て** ウィンドウが表示されます。 ゲストストレージ用のディスクパーティション、LUN、又はファイルベースイメージの作成を 選択します。

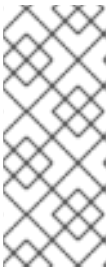
   全てのイメージファイルは **/var/lib/libvirt/images/** ディレクトリに格納しなければなりません。ファイルベース イメージ用としては他のディレクトリは SELinux によって禁止されています。 SELinux を強制モードで実行する場合は、「SELinux と仮想化」 でゲストインストールのための 詳細を確認して下さい。

   使用するゲストストレージイメージは、インストールのサイズ、追加のパッケージ、 及びアプリケーション、更にはゲストスワップファイルのサイズの合計よりも大きい ことが要求されます。インストールプロセスはゲストに割り当てられた RAM のサイズを 基にしてゲストスワップのサイズを選択します。

   アプリケーションや他のデータの為にゲストが追加のスペースを必要とする場合には、 余分のスペースを割り当てます。例えば、ウェブサービスはログファイルの為に追加の スペースを必要とします。



選択したストレージタイプでゲスト用に適切なサイズを選びます。それから **進む** ボタンをクリックします。



注記

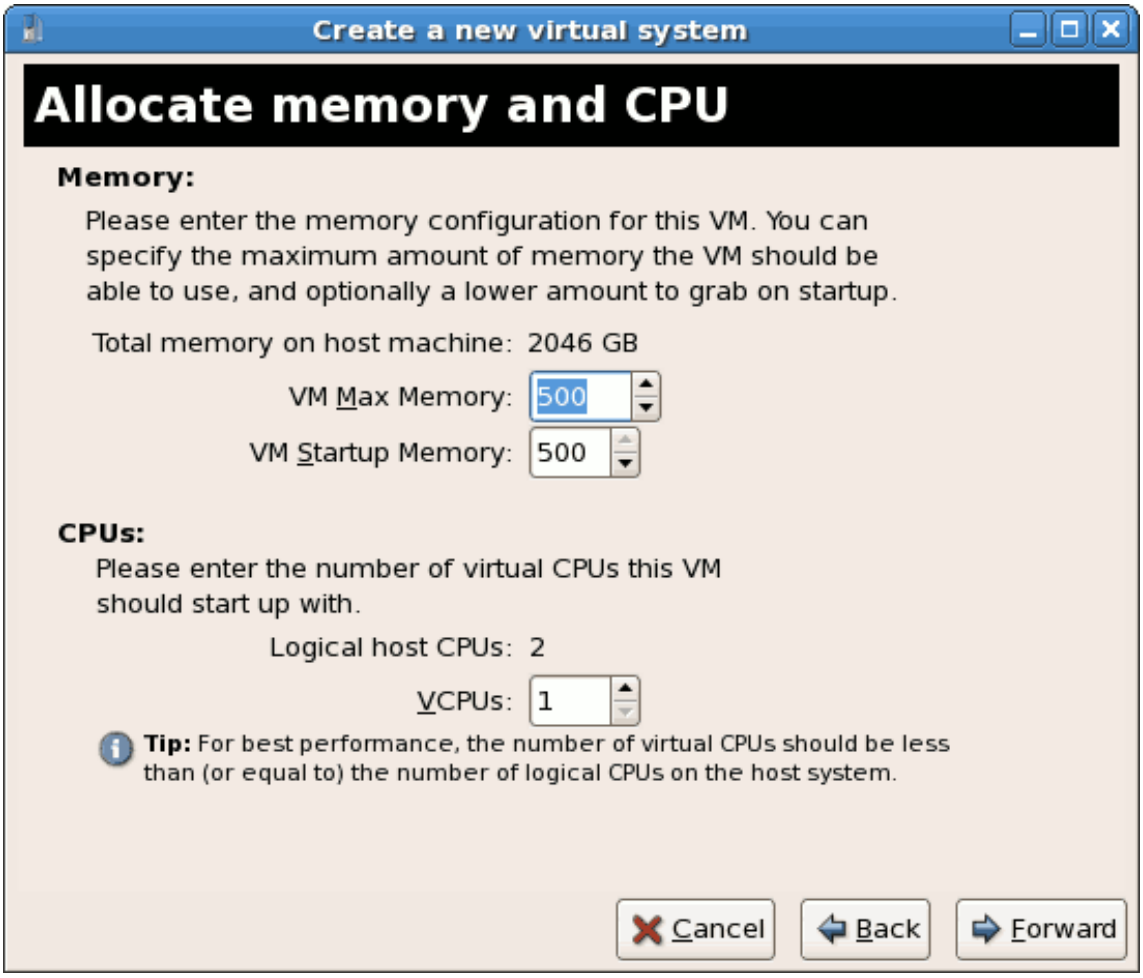
仮想マシンイメージ **/var/lib/xen/images/** 用には デフォルトのディレクトリを使用することが推奨されます。異なるディレクトリ （このサンプルでは例えば、**/xen/images/**）を 使用している場合は、SELinux ポリシーにそれを追加して、インストールを継続 する前にラベル変更することを確認して下さい（SELinux ポリシーの変更法はドキュメントの後半で 説明してあります）。

8. 「メモリーと CPU 割り当て」ウィンドウが表示されます。仮想化の CPU と RAM の 割り当てに適切な値を選択します。これらの値は、ホストとゲストのパフォーマンスに 影響します。

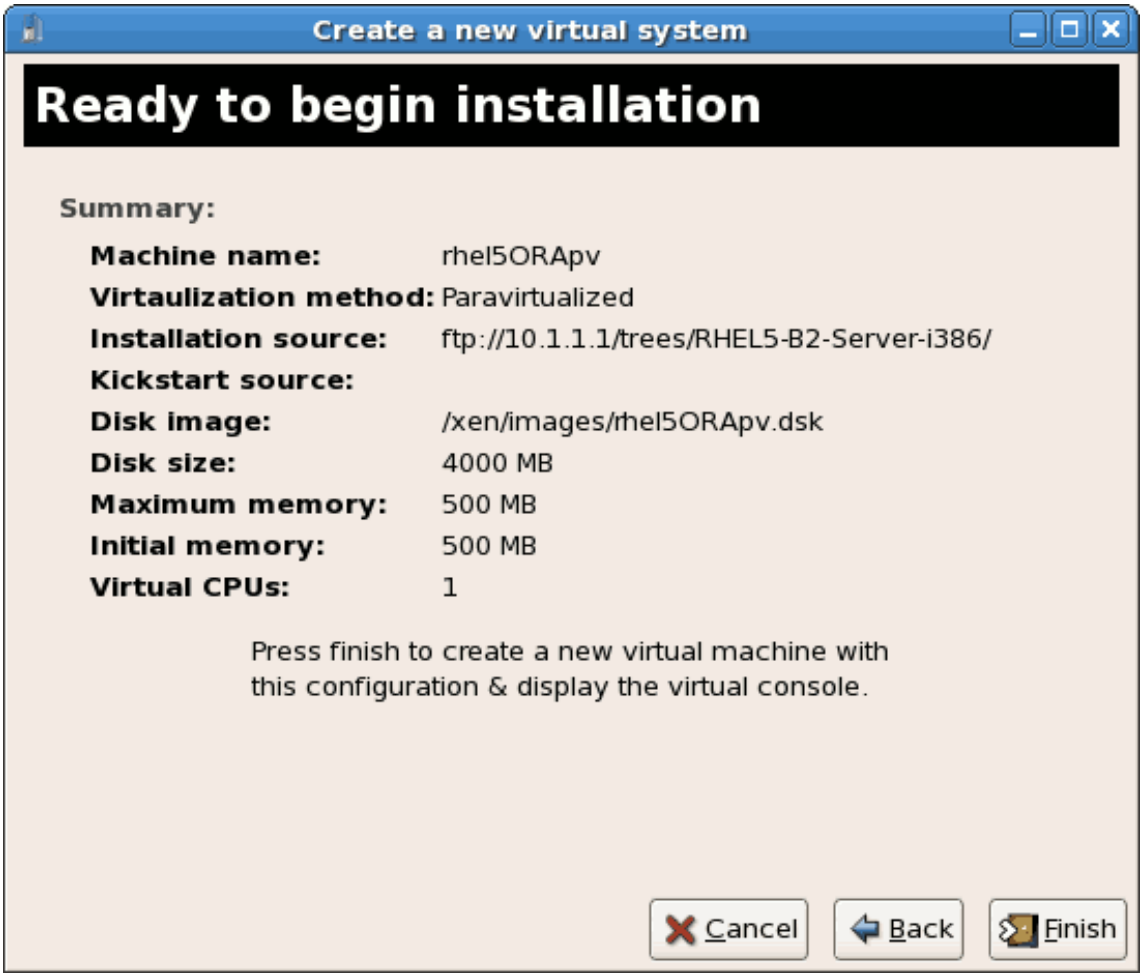
   ゲストは効率的にそして効果的に稼働するために十分な物理メモリー(RAM) を必要とします。ゲストオペレーティングシステムとアプリケーションの要件に適したメモリーの値を選択します。 ほとんどのオペレーティングシステムは反応良く機能するのに少なくとも 512MB の RAM を 必要とします。ゲストは物理 RAM を使用することを忘れないで下さい。稼働ゲストが多すぎたり、 ホストシステム用に十分なメモリーを残さなかったりすると仮想メモリーをかなり消費してしまいます。 仮想メモリーは比較的に遅いため、システムパフォーマンスと反応性に悪影響を与えます。全ての ゲストとホストが効率的に稼働できるように十分なメモリーを割り当てることを確認して下さい。

   十分な仮想 CPU を仮想ゲストに割り当てます。ゲストがマルチスレッドのアプリケーションを実行している場合は、ゲストが効率的に稼働できるのに必要な数の仮想化 CPU を割り当てます。 しかし、ホストシステム上で利用できる物理プロセッサ（又はハイパースレッド）以上の

仮想 CPU を割り当てないで下さい。仮想プロセッサを超過して割り当てることは可能ですが、超過割り当てでは、プロセッサコンテキストがオーバーヘッドを切り替えるため、ゲストとホストに 深刻な悪影響を与えます。



9. ユーザーが入力した全ての設定情報の要約を示すインストール準備完了画面が表示されます。表示された情報を再確認して、変更したい場合は **戻る** ボタンを 使用します。入力データに満足できる状態ならば、**完了** ボタンをクリックして、インストールプロセスを開始させます。



VNC ウィンドウが開いて、ゲストオペレーティングシステムのインストールプロセスの 開始を示します。

これで、**virt-manager** によるゲスト作成のための一般的な プロセスは終了です。*7章ゲストオペレーティングシステムのインストール手順* には各種の一般的なオペレーティング システムインストールに関するステップバイステップの案内が含まれています。

6.3. PXE を使用してゲストのインストール

このセクションでは、PXE を使用したゲストのインストールに必要な手順を説明します。 PXE のゲストインストールには、ネットワークブリッジとして知られる共有ネットワーク デバイスが必要になります。以下の手順はブリッジの作成法と PXE インストール用の ブリッジの活用に必要なステップを扱っています。

1. **新規ブリッジの作成**

   a. **/etc/sysconfig/network-scripts/** ディレクトリ内に新規の ネットワークスクリプトファイルを作成します。ここの例では、**ifcfg-installation** と 言う名前のファイルを作成し、それが ***installation*** と言う名前のブリッジを 作ります。

```
# cd /etc/sysconfig/network-scripts/
# vim ifcfg-installation
DEVICE=installation
TYPE=Bridge
BOOTPROTO=dhcp
ONBOOT=yes
```

警告

TYPE=Bridge の行は、大文字/小文字を区別します。 大文字の 'B' と小文字の 'ridge' でなければなりません。

b. 新規ブリッジの開始

```
# ifup installation
```

c. この時点では、まだブリッジにインターフェイスが追加されていません。**brctl show** コマンドを使用してシステム上のネットワークブリッジの詳細を見ることが できます。

```
# brctl show
bridge name     bridge id               STP enabled
interfaces
installation    8000.000000000000       no
virbr0          8000.000000000000       yes
```

virbr0 ブリッジとは、デフォルトのイーサネットデバイス上の NAT (Network Address Translation) のための **libvirt** で使用される デフォルトのブリッジです。

2. **新規ブリッジにインターフェイスを追加**
インターフェイス用の設定ファイルを編集します。先の手順で作成されたブリッジの 名前を持つ設定ファイルへ **BRIDGE** パラメータを追加 します。

```
# Intel Corporation Gigabit Network Connection
DEVICE=eth1
BRIDGE=installation
BOOTPROTO=dhcp
HWADDR=00:13:20:F7:6E:8E
ONBOOT=yes
```

設定ファイルの編集の後に、ネットワークを再スタートするか又はリブートします。

```
# service network restart
```

brctl show コマンドを使用してインターフェイスが添付 されていることを確認します:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
installation    8000.001320f76e8e       no              eth1
virbr0          8000.000000000000       yes
```

3. **セキュリティの設定**
iptables を設定して全てのトラフィックがブリッジまで 転送されるようにします。

```
# iptables -I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
# service iptables save
# service iptables restart
```

注記

別の方法として、ブリッジ化したトラフィックが **iptables** によって プロセスされるのを阻止します。**/etc/sysctl.conf** 内で以下の行を 追加します:

```
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0
```

設定されたカーネルパラメータを **sysctl** で再ロードします。

```
# sysctl -p /etc/sysctl.conf
```

4. **インストールの前に libvirt を再スタート**
   libvirt デーモンの再スタート

```
# service libvirtd reload
```

ブリッジは設定終了です。これでインストールを開始できます。

virt-install を使用した PXE インストール

virt-install に **--network=bridge:installation** インストールパラメータを追記します。ここで ***installation*** とは、該当ブリッジの名前です。PXE インストールには、***--pxe*** パラメータを使用します。

例6.3 virt-install を使用した PXE インストール

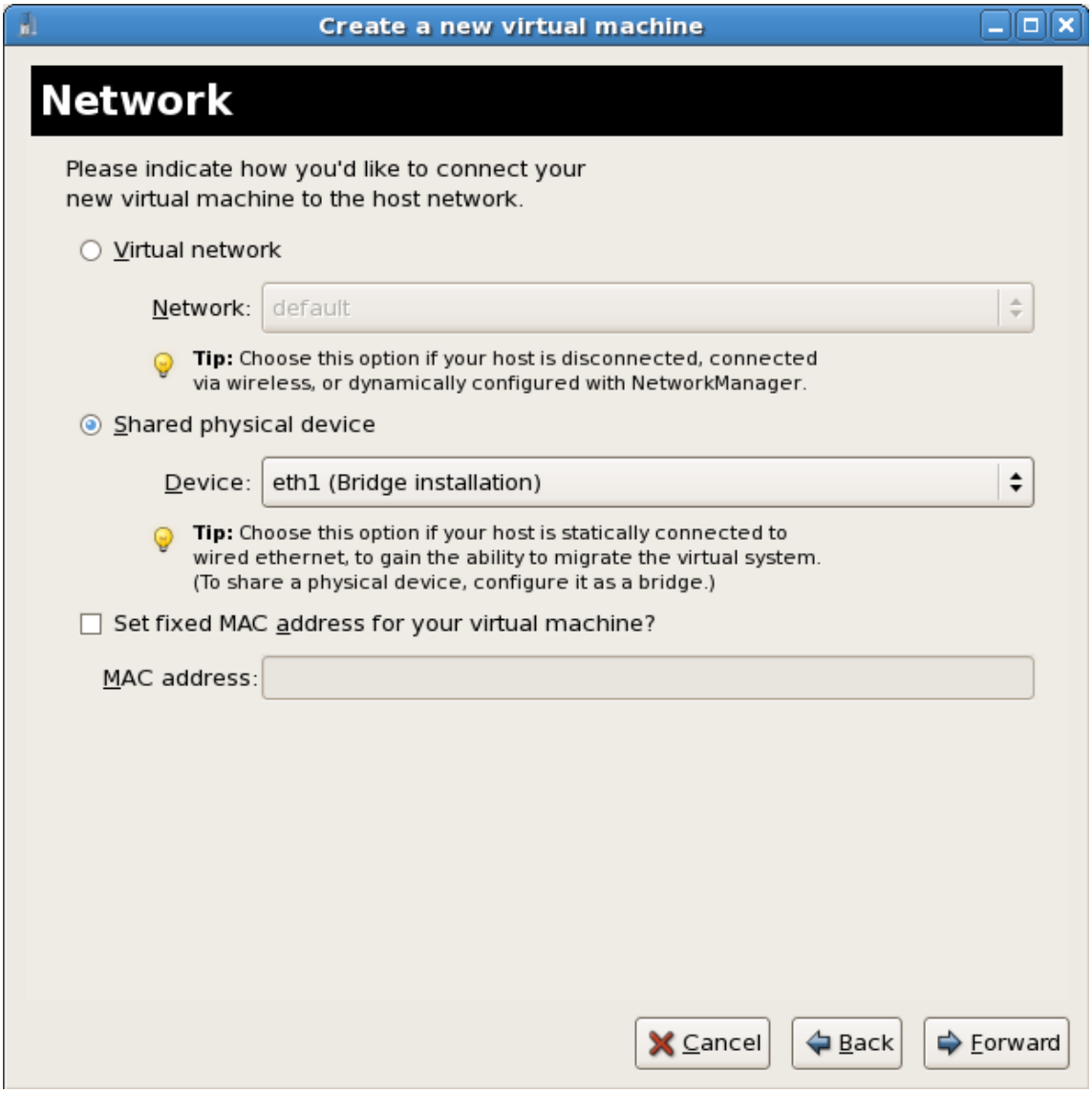
```
# virt-install --accelerate --hvm --connect qemu:///system \
    --network=bridge:installation --pxe\
    --name EL10 --ram=756 \
    --vcpus=4
    --os-type=linux --os-variant=rhel5
    --file=/var/lib/libvirt/images/EL10.img \
```

virt-manager を使用した PXE インストール

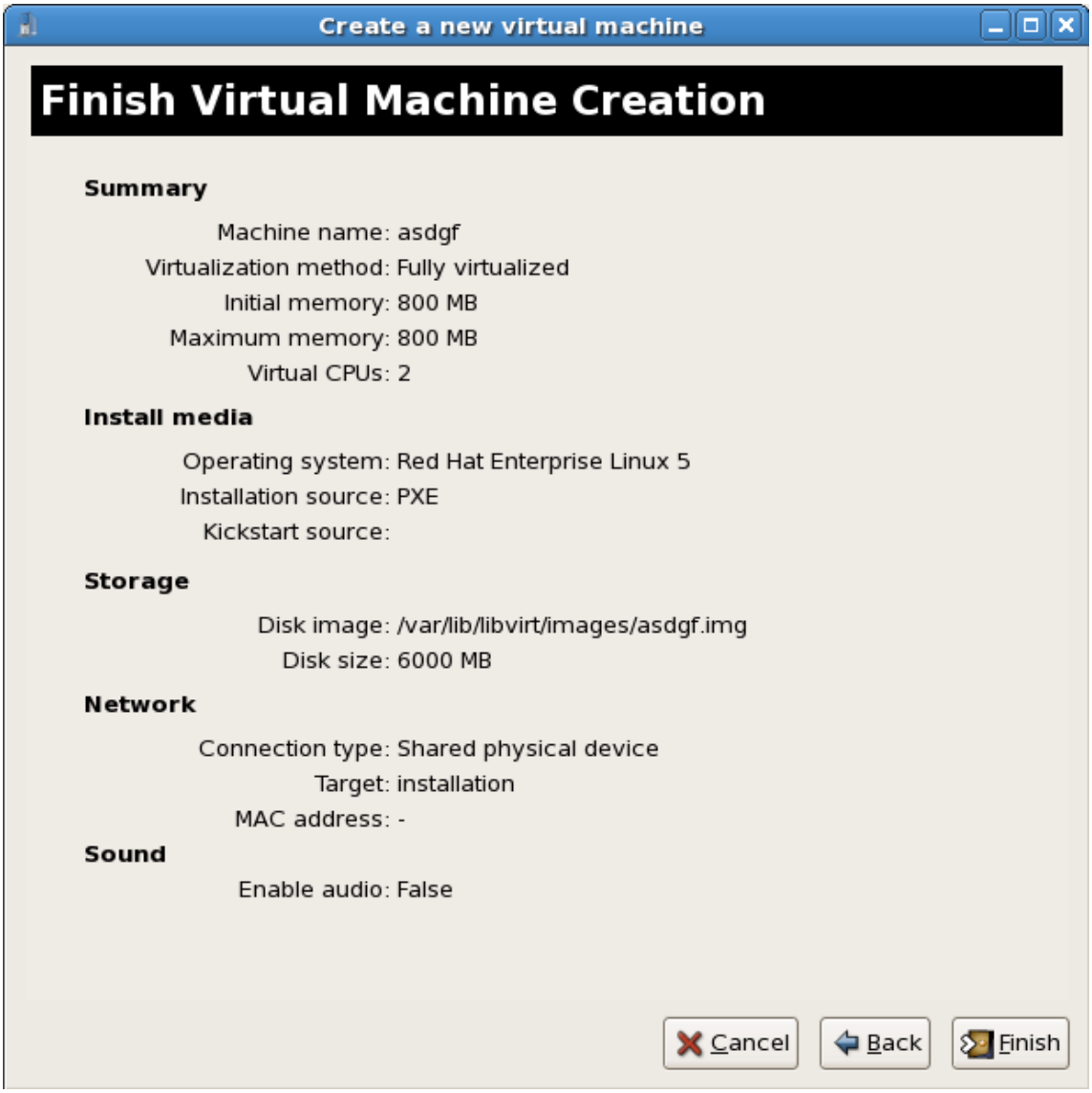
以下の手順は、標準の virt-manager インストール手続きとは異なる手順です。 標準の手続きには *7章ゲストオペレーティングシステムのインストール手順*を参照して下さい。

1. **PXE を選択**
   インストールメソッドとして PXE を選択。

2. **ブリッジの選択**
   物理デバイスを共有 を選択して、先の手順で作成してある ブリッジを選択します。



3. **インストールの開始**
   インストールの開始準備が出来ています。



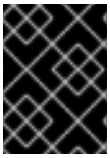
DHCP 要求が送信されて、有効な PXE サーバーが見付かるとゲストインストールの プロセスが始まります。

第7章 ゲストオペレーティングシステムのインストール手順

この章では、Red Hat Enterprise Linux 上の仮想化環境に於いて各種ゲストオペレーティング システムのインストール法を説明しています。基本的な手順を理解するためには、*6章 仮想化ゲストインストールの概要* を参照して下さい。

7.1. RED HAT ENTERPRISE LINUX 5.0 を PARA-VIRTUALIZED ゲストとしてインストール

このセクションでは、Red Hat Enterprise Linux 5 を para-virtualized ゲストとしてインストール する方法を説明します。Para-virtualization は完全仮想化よりも迅速であり、完全仮想化の 全ての利便性をサポートしています。Para-virtualization は特別なサポートのある **kernel-xen** カーネルを必要とします。



重要

Para-virtualization は Xen hypervisor とでのみ機能します。Para-virtualization は KVM hypervisor とでは機能しません。

インストールを開始する前に root アクセスがあることを確認して下さい。

この方法は Red Hat Enterprise Linux をリモートサーバーからインストールします。 このセクションに示されているインストール案内は、最低限インストールのライブ CD-ROM からのインストールに似ています。

virt-manager 又は、virt-install を使って para-virtualized Red Hat Enterprise Linux 5 ゲストを作成します。**virt-manager** についての案内には「virt-manager を使用してゲストを作成」にある手順を参照して下さい。

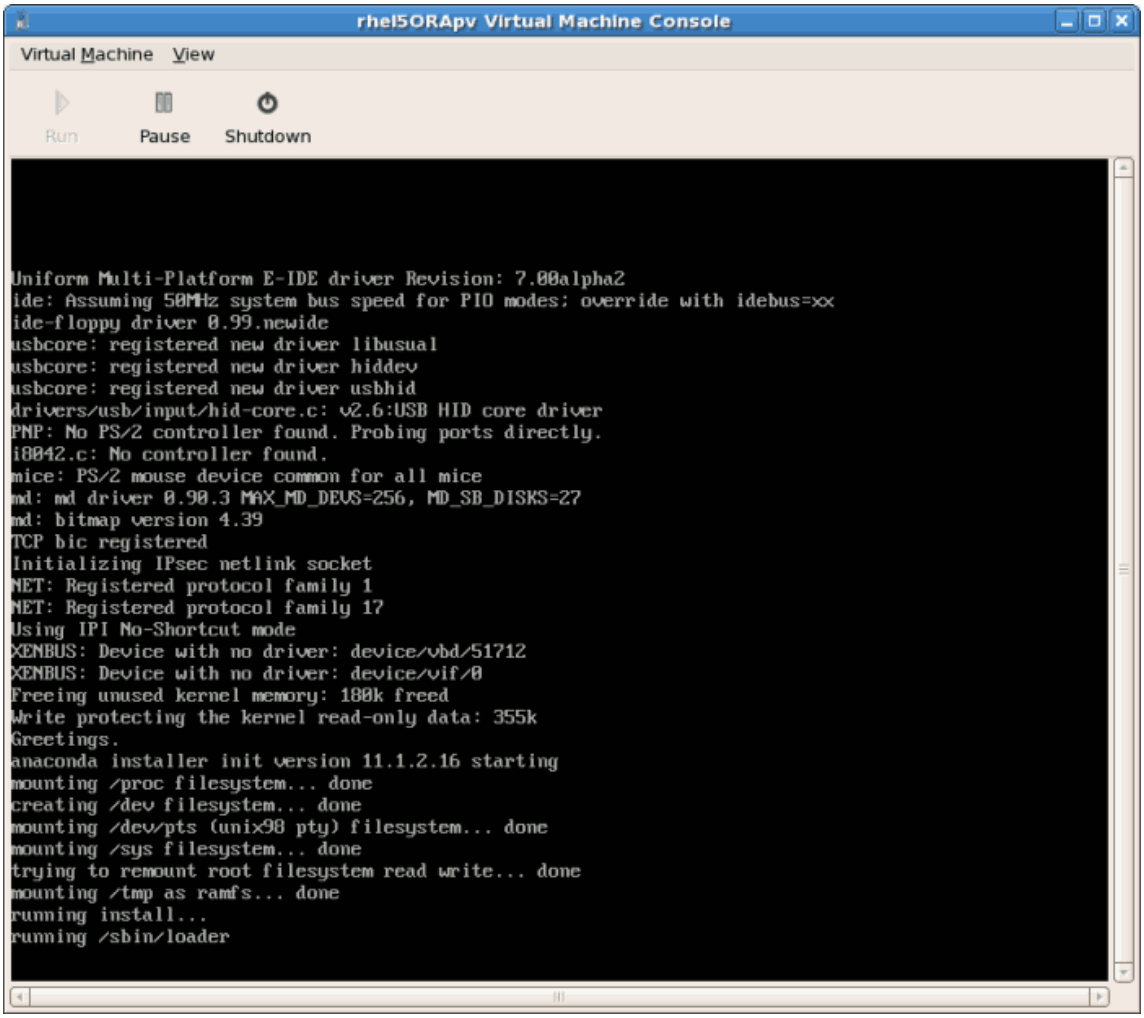
コマンドラインベースの **virt-install** ツールを使用して para-virtualized ゲストを作成します。**--vnc** オプションは、 グラフィックインストールを表示します。この例中のゲストの名前は*rhel5PV* であり、 ディスクイメージファイルは *rhel5PV.dsk* となり、Red Hat Enterprise Linux 5 のインストールツリーのローカルミラーは *ftp://10.1.1.1/trees/RHEL5-B2-Server-i386/* となります。これらの値をご使用のシステムとネットワーク用に 的確なものに変更して下さい。

```
# virt-install -n rhel5PV -r 500 \
-f /var/lib/libvirt/images/rhel5PV.dsk -s 3 --vnc -p \
-l ftp://10.1.1.1/trees/RHEL5-B2-Server-i386/
```

注記

Red Hat Enterprise Linux はグラフィカルインターフェイスや、手動入力を 使用せずにインストールできます。Kickstart ファイルの使用でインストール プロセスを自動化することができます。

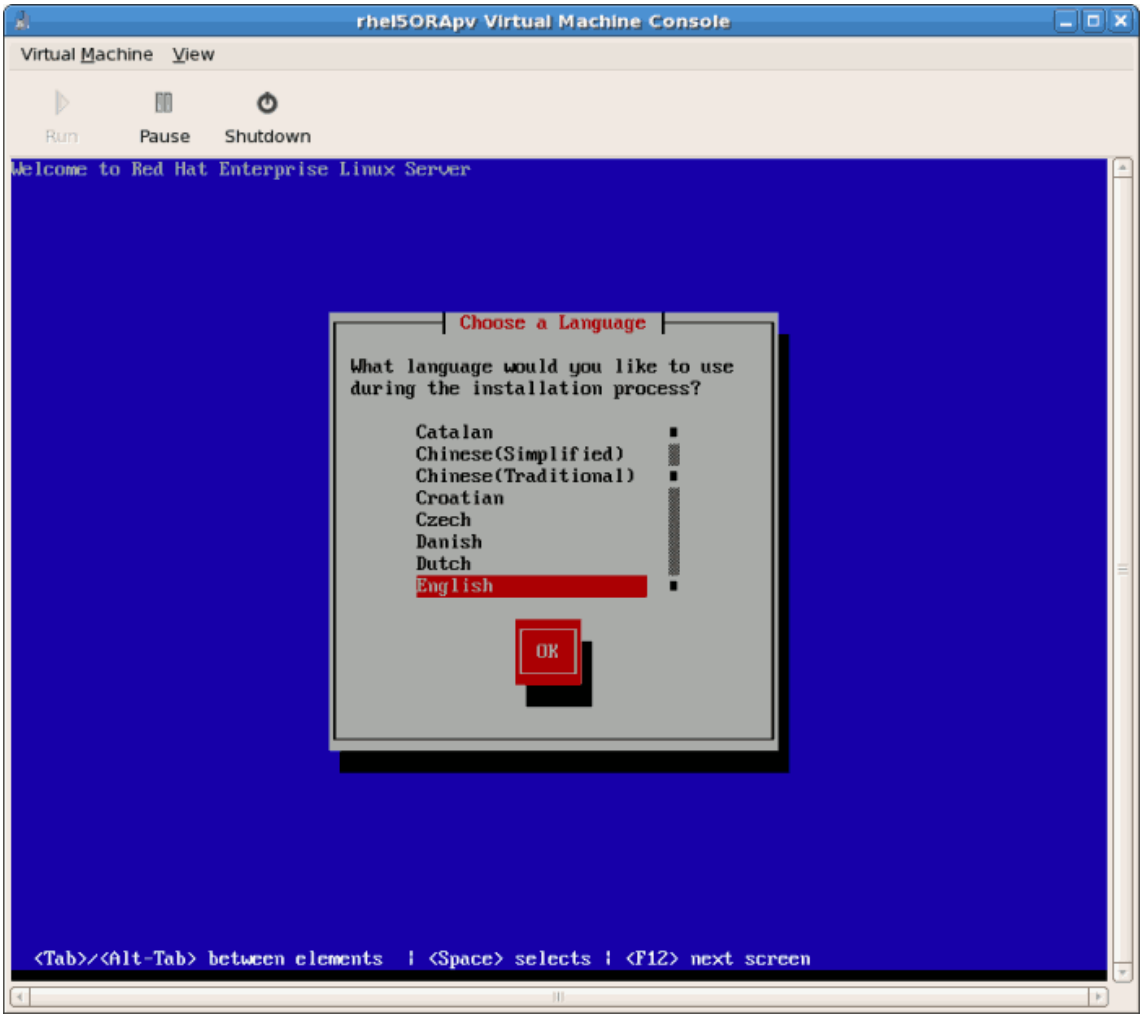
いずれの方法でもこのウィンドウを開き、使用するゲストの初期ブート段階を表示します:



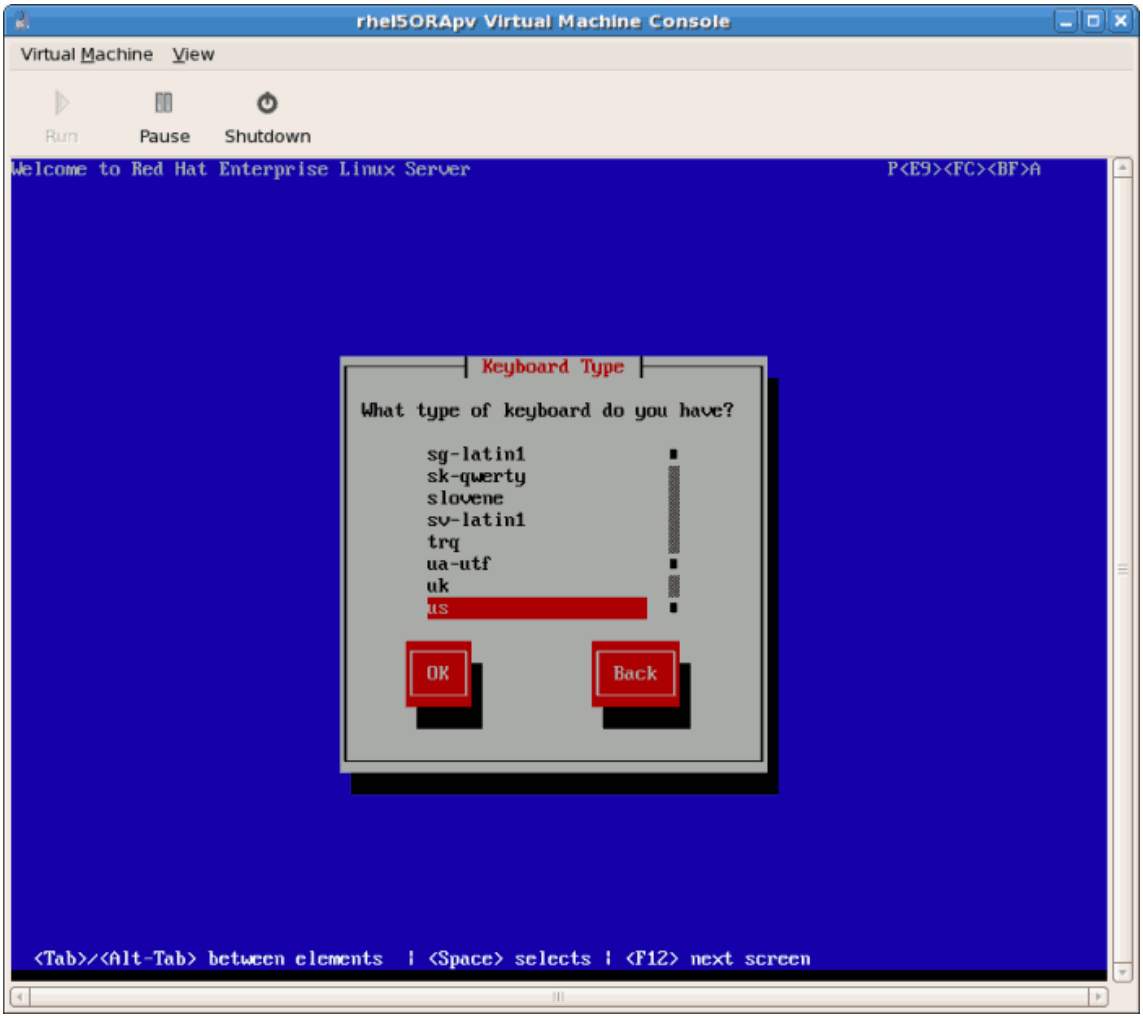
使用しているゲストが初期ブートを完了したら、Red Hat Enterprise Linux の 標準インストールプロセスが開始されます。ほとんどのシステムにはデフォルトの 回答が受け付けられます。

手順7.1 Para-virtualized Red Hat Enterprise Linux ゲストのインストール手順

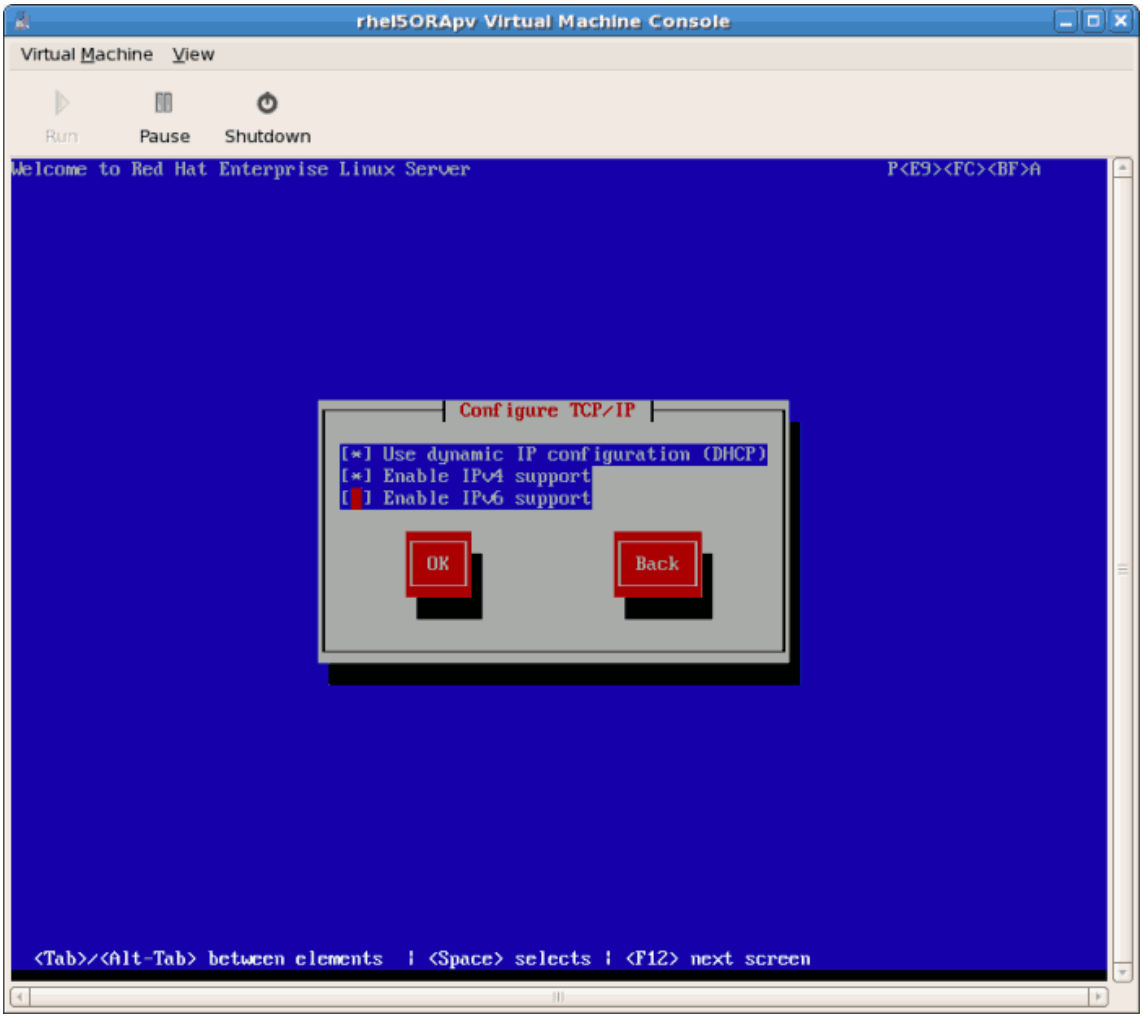
1. 言語を選択して、**OK** をクリックします。



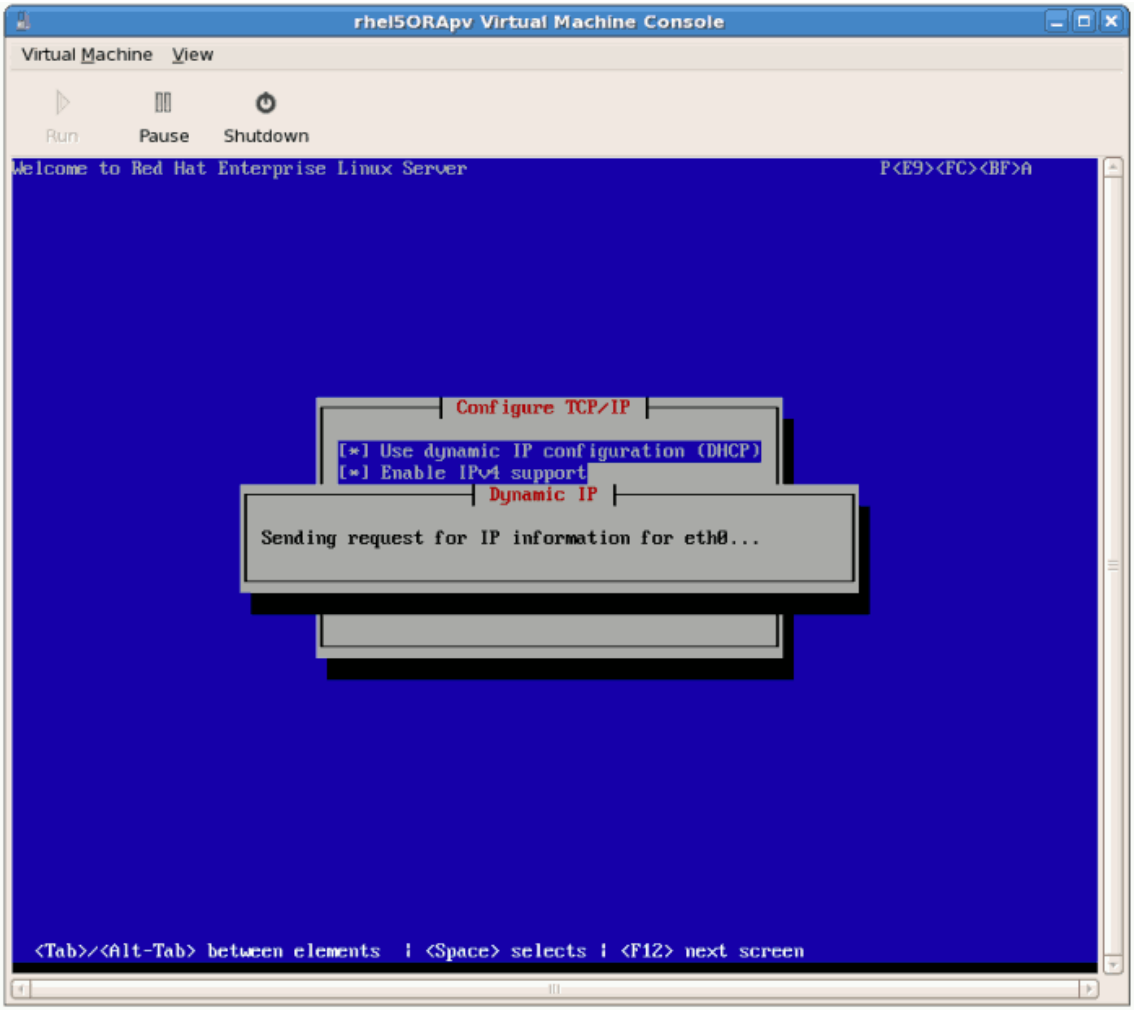
2. キーボードレイアウトを選択して、**OK** をクリックします。



3. ゲストのネットワークアドレスを割り当てます。（以下のように）**DHCP** の使用を選択するか、又は 静的 IP アドレスを 使います:



4. DHCP を選択すると、インストールプロセスは、ここで IP アドレスの取得を 試みます:

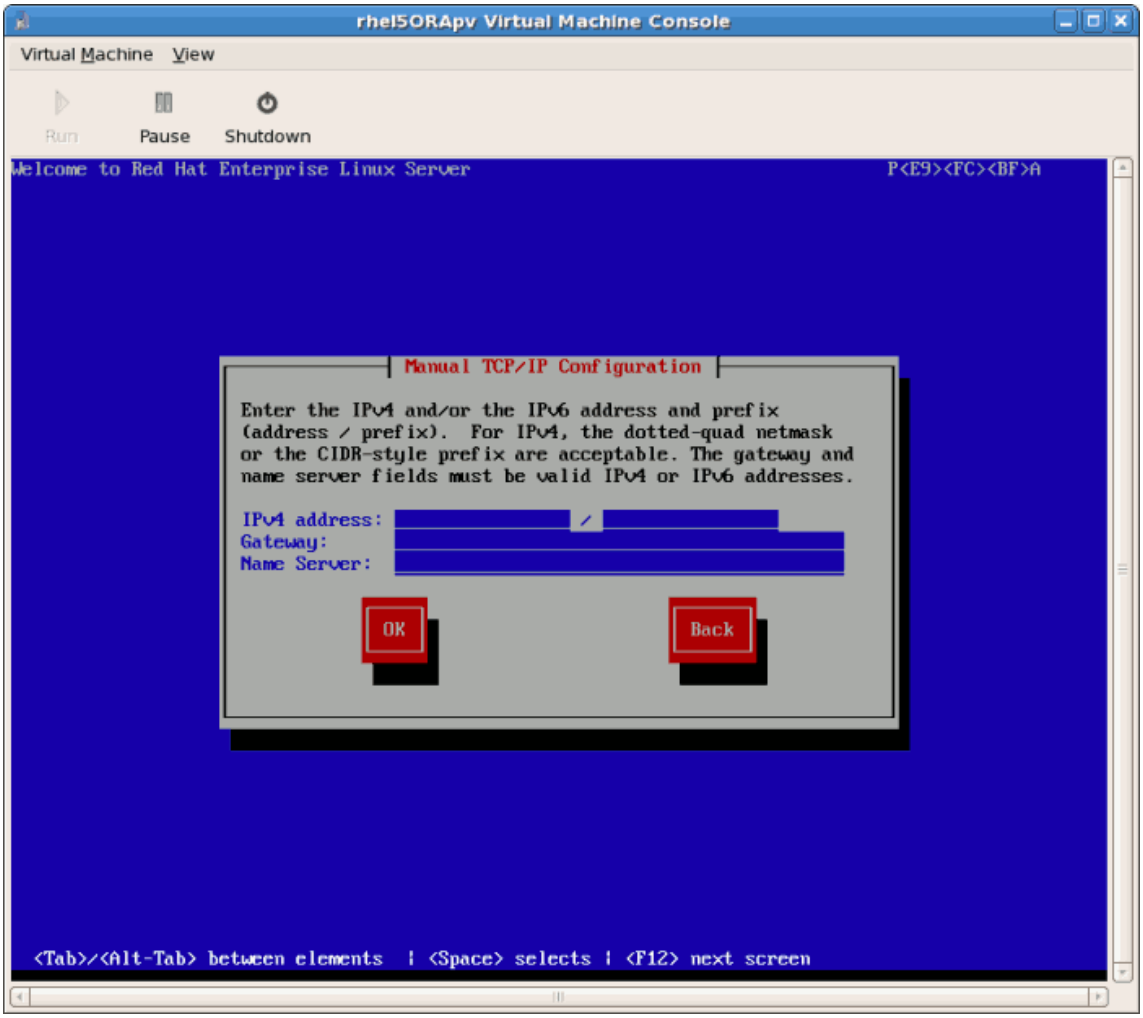


5. ゲスト用に 静的 IP アドレスを選択すると、このプロンプトが出現します。 ゲストのネットワーキング設定の詳細を入力します:

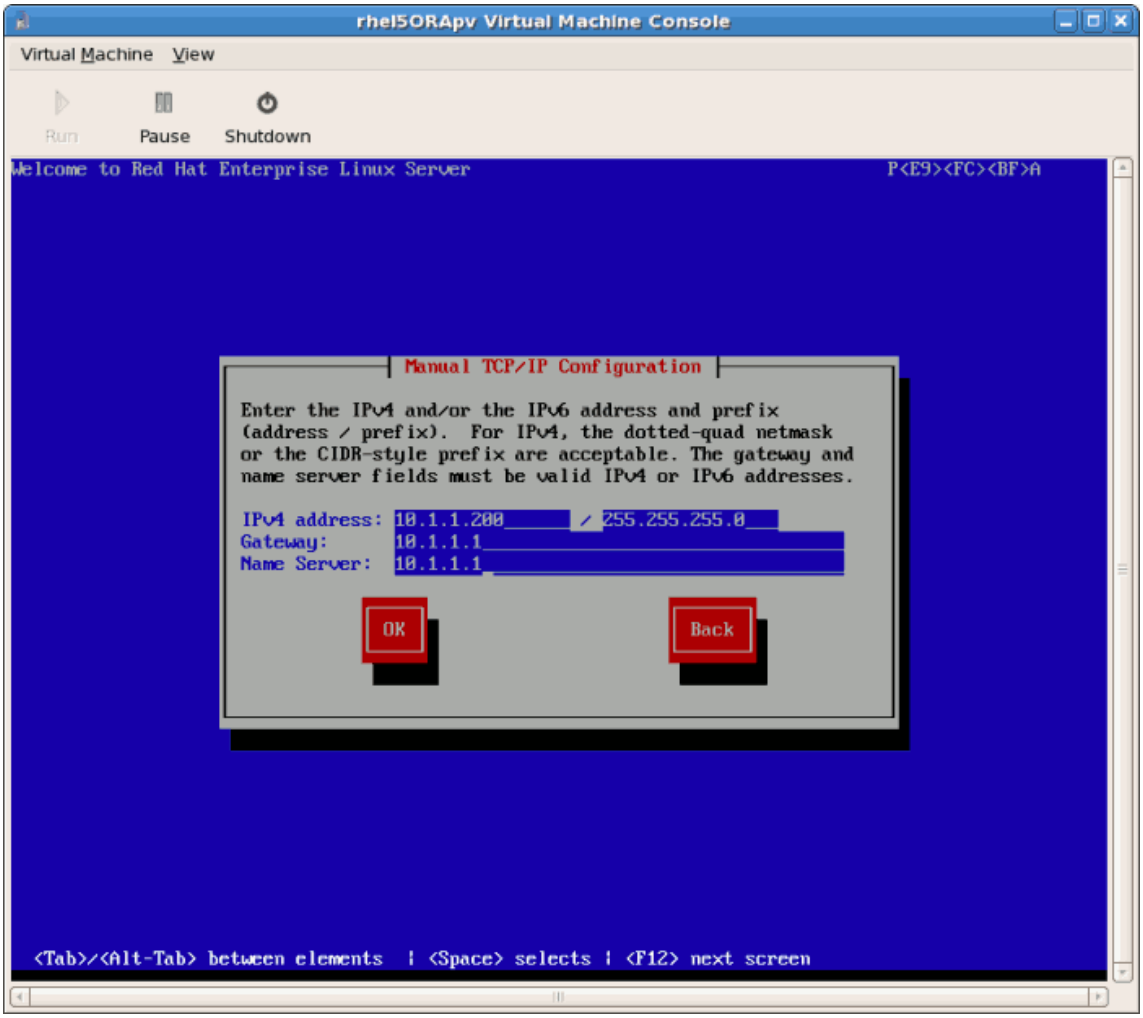
   a. 有効な IP アドレスを入力します。入力する IP アドレスがインストールツリーを持つ サーバーに到達するように確認して下さい。

   b. 有効なサブネットマスク、デフォルトのゲートウェイ、及びネームサーバーのアドレスを入力します。

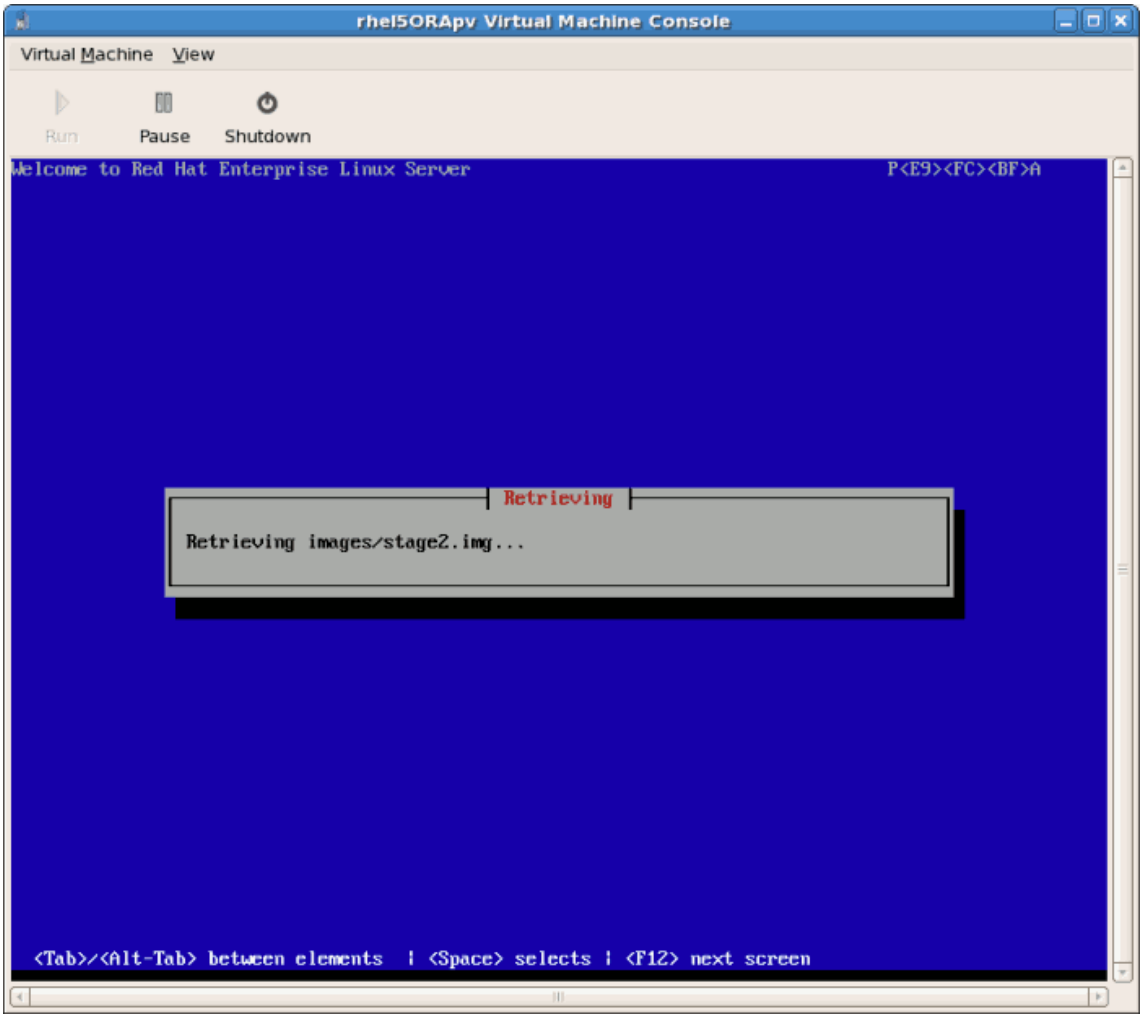
   言語を選択して、**OK** をクリックします。



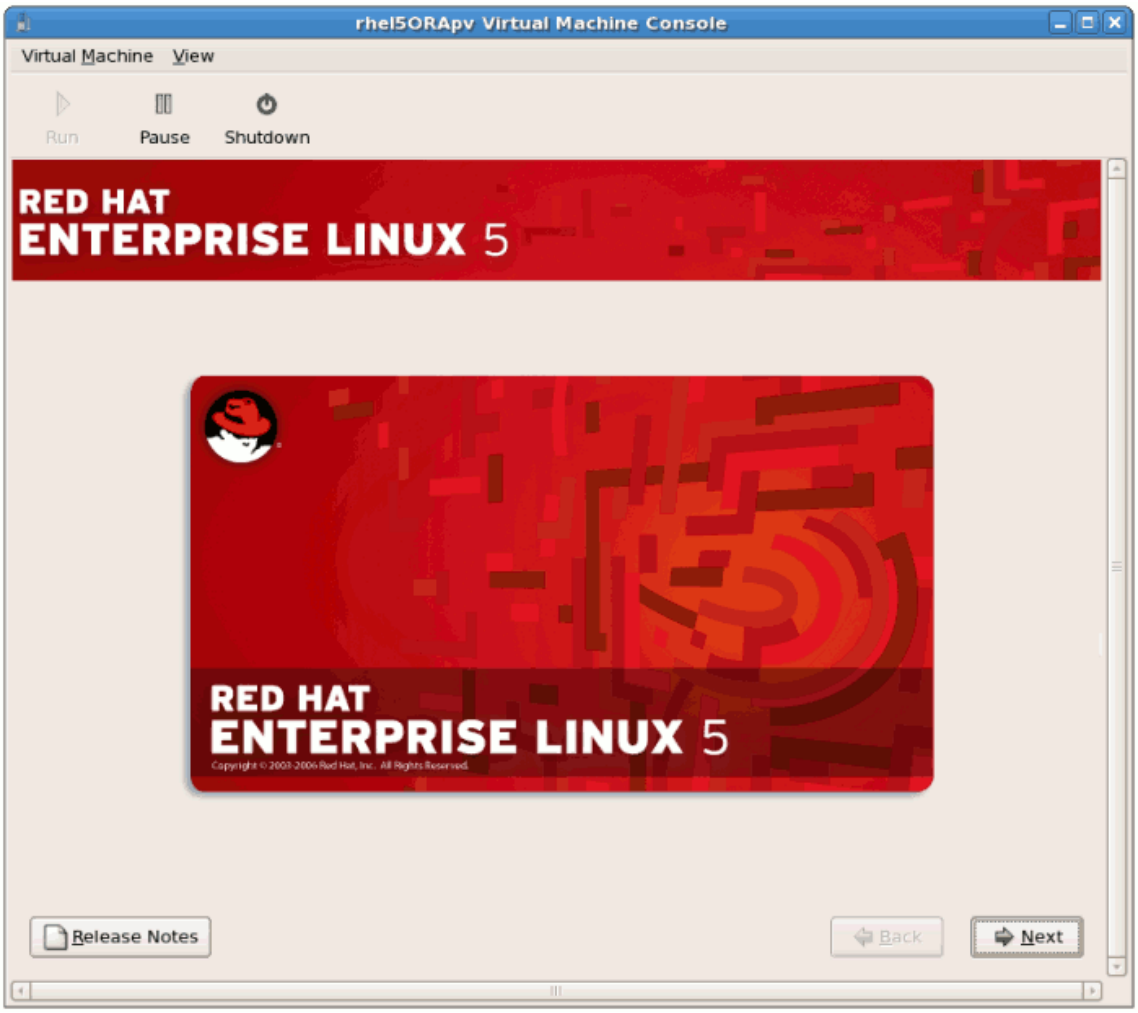
6. 以下に静的 IP アドレス設定の例を示します:



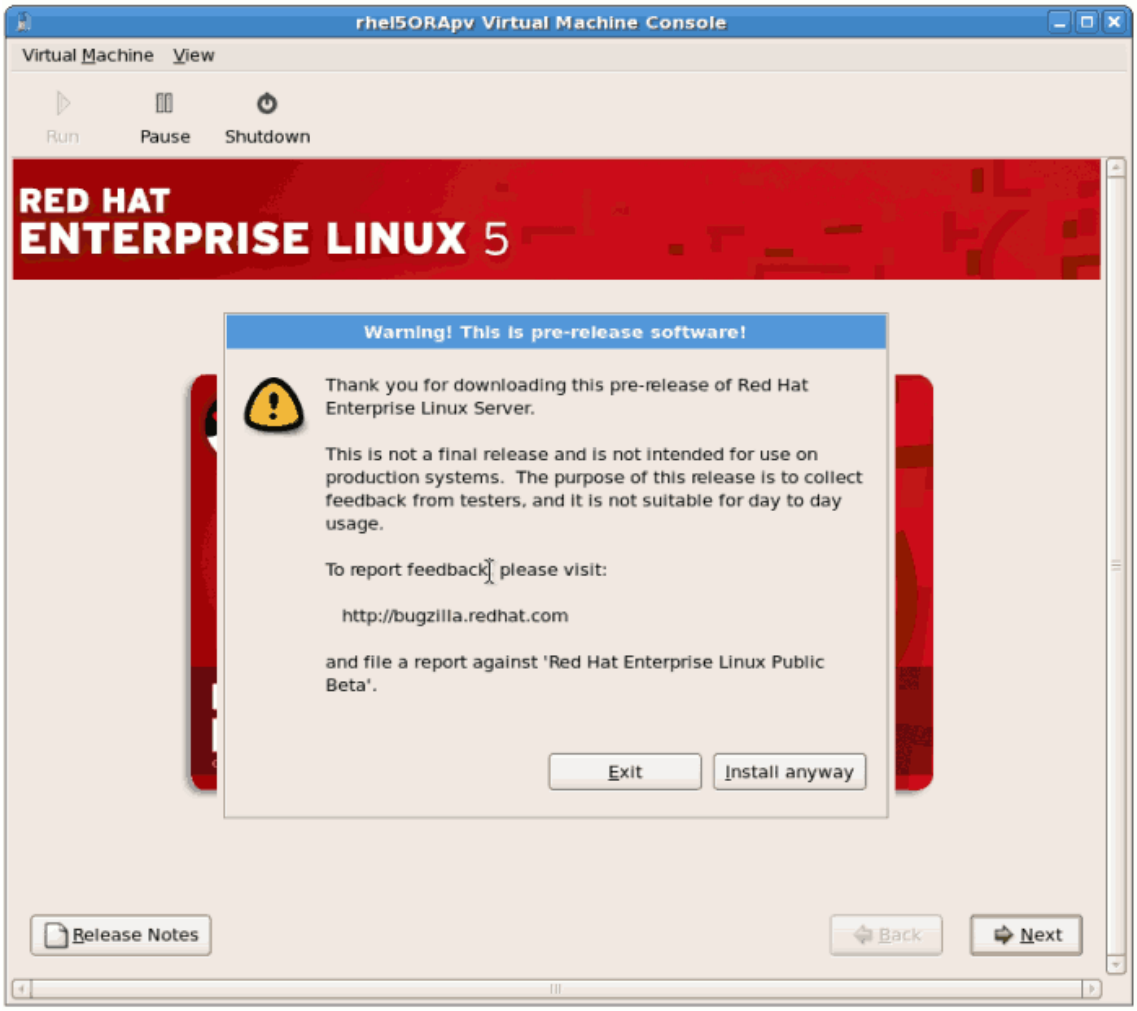
7. インストールプロセスはここで、必要とするファイルをサーバーから取り込みます:



初期のステップが終了すると、グラフィカルプロセスが開始されます。

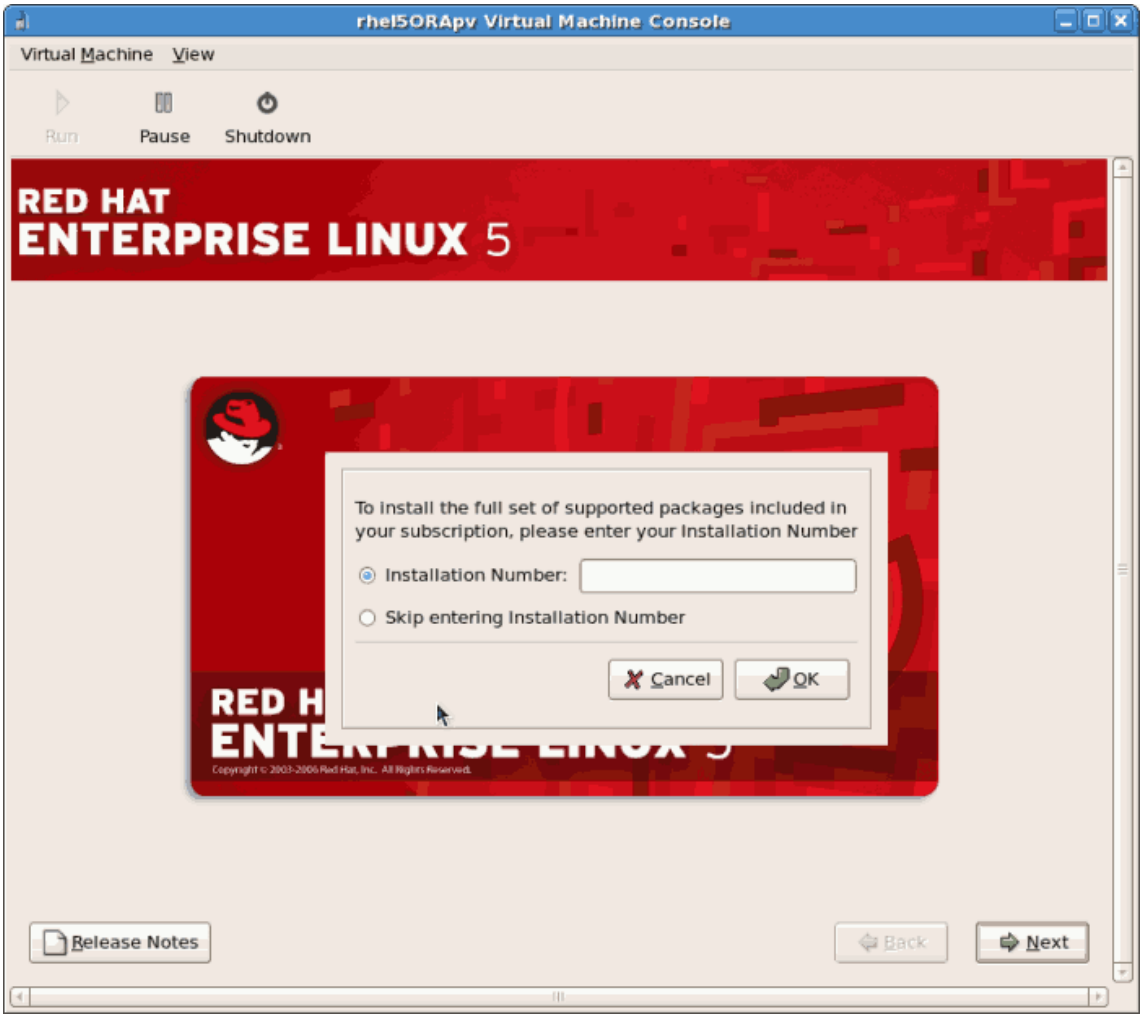


Beta バージョンや早期のリリースディストリビューションをインストールしている 場合は、オペレーティングシステムをインストールしたいことを確定して下さい。 **とにかくインストールする** をクリックして、**OK** を クリックします:



手順7.2 グラフィカルインストールプロセス

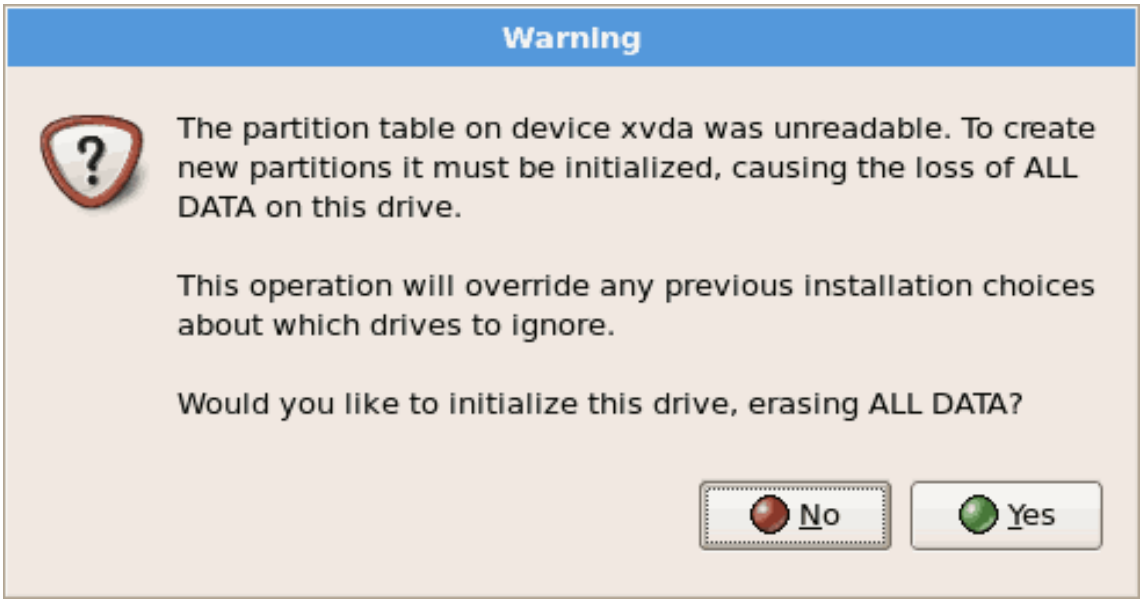
1. 有効な登録コードを入力します。有効な RHN サブスクリプションキーがある場合は、**Installation Number** フィールド内に それを入力します:



注記

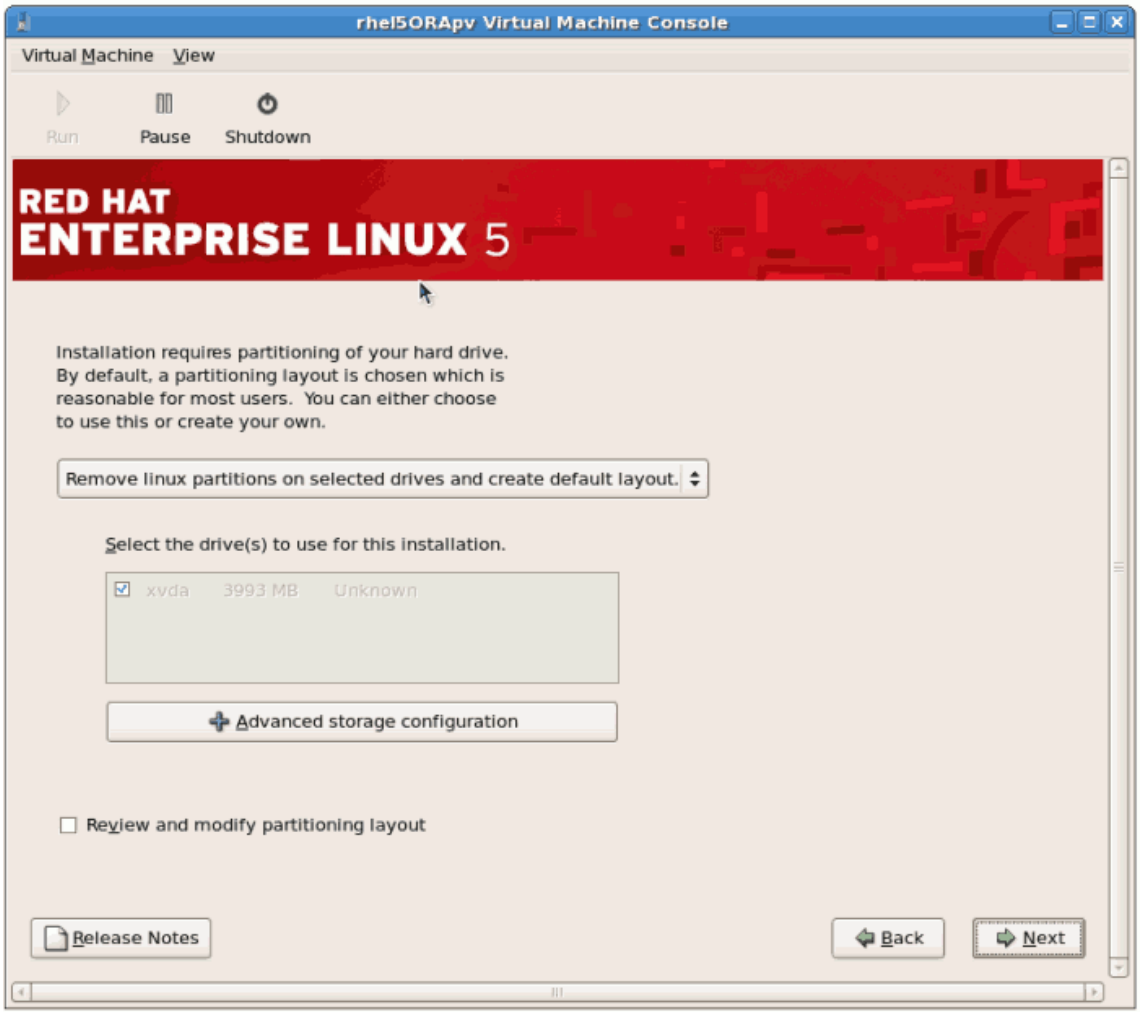
登録ステップを飛ばす場合、インストールの後で **rhn_register** コマンドを使用して、Red Hat Network アカウントの詳細を 確定することができます。**rhn_register** コマンドには root アクセスが必要です。

2. インストールプロセスがインストールに選択した場所に格納されている全てのデータの 抹消を是認するように催促します:



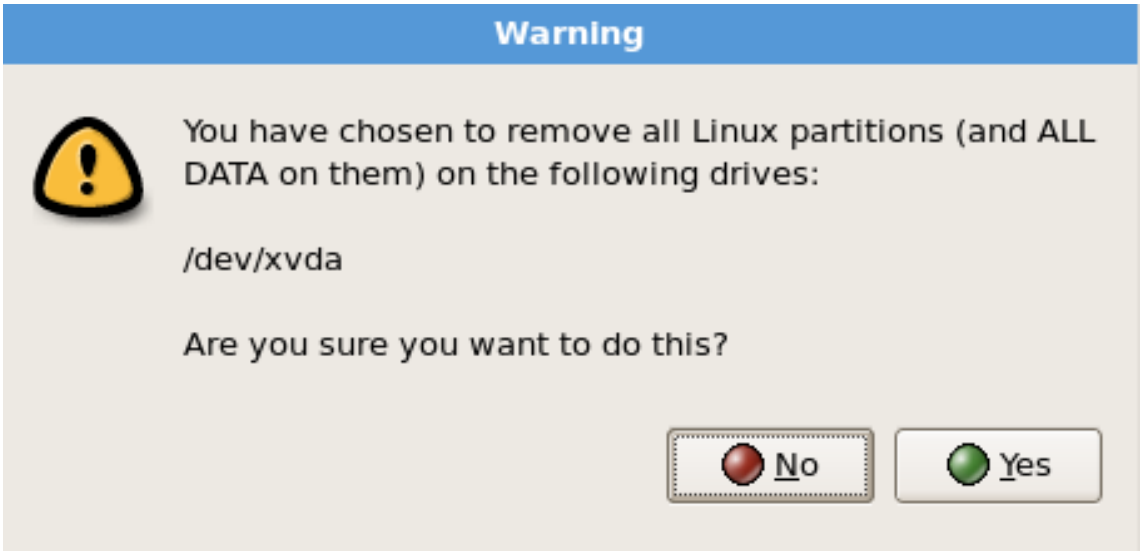
はい をクリックして継続します。

3. ストレージ設定とパーティションレイアウトを再確認します。ゲストのストレージ用に iSCSI を使用したい場合、高度なストレージ設定を選択することができます。



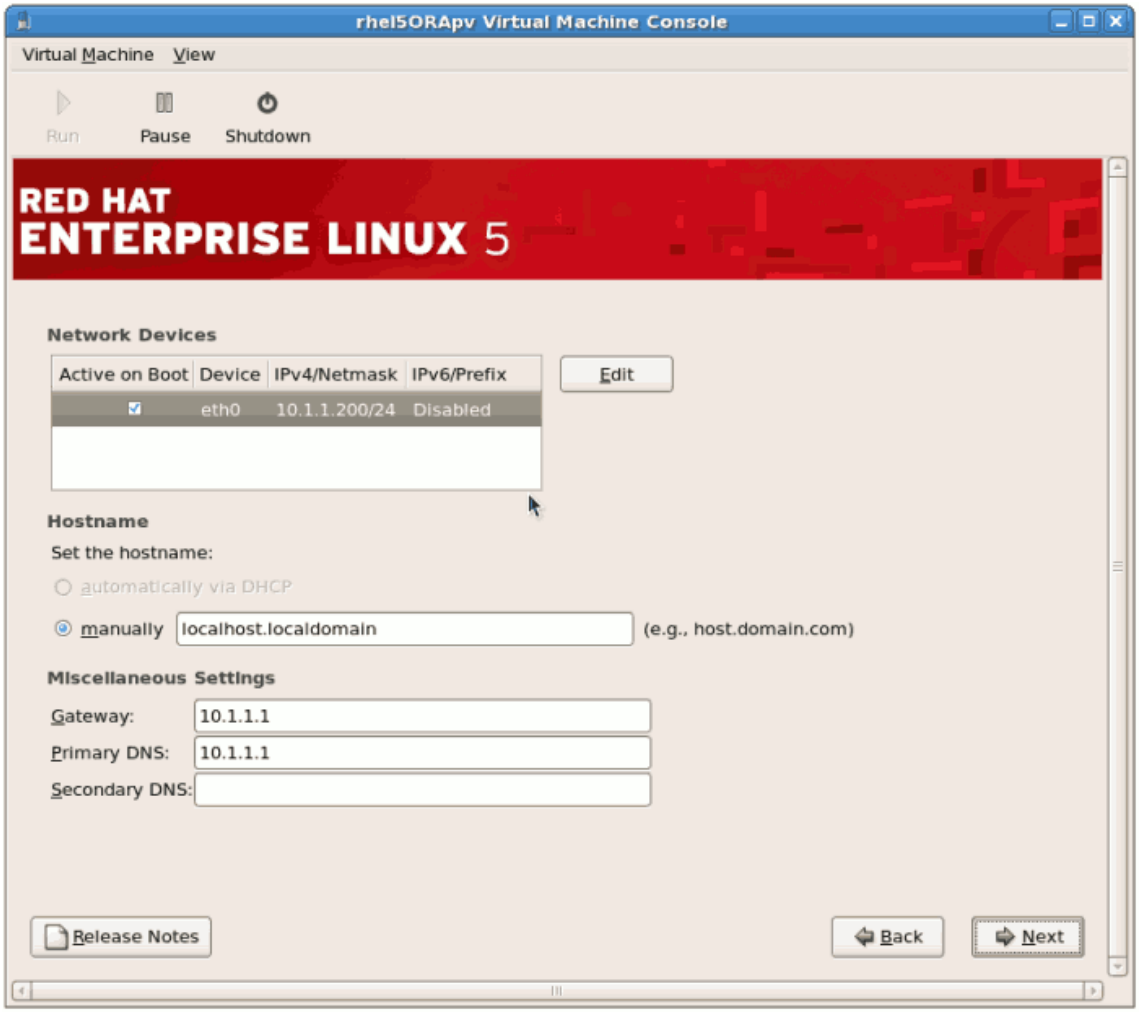
選択をして、それから **次** をクリックします。

4. インストール用に選択したストレージを確定します。



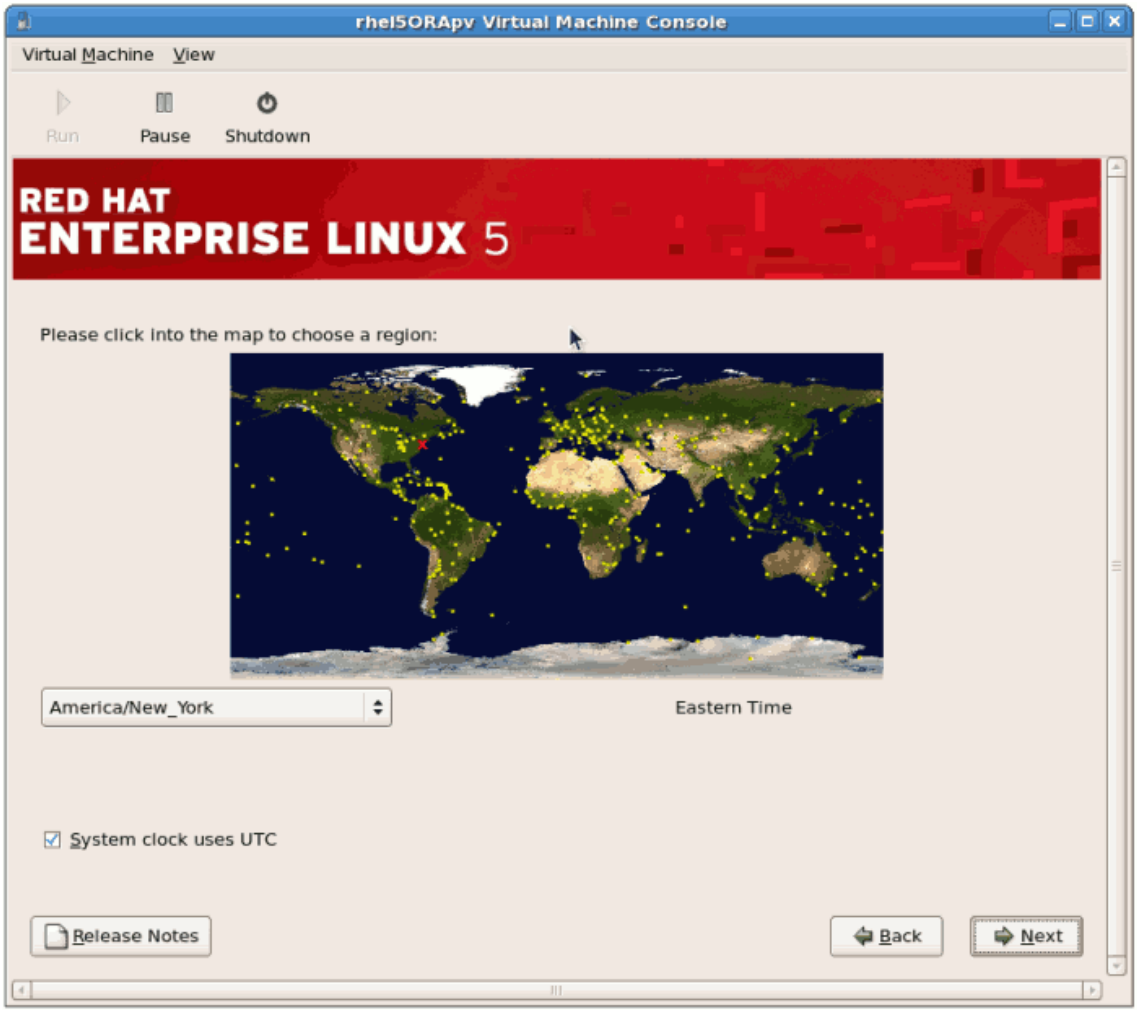
はい をクリックして継続します。

5. ネットワーキングとホスト名のセッティングを構成します。これらのセッティングは 先にインストールプロセスで入力したデータで充填されます。必要であればそれを 変更して下さい。

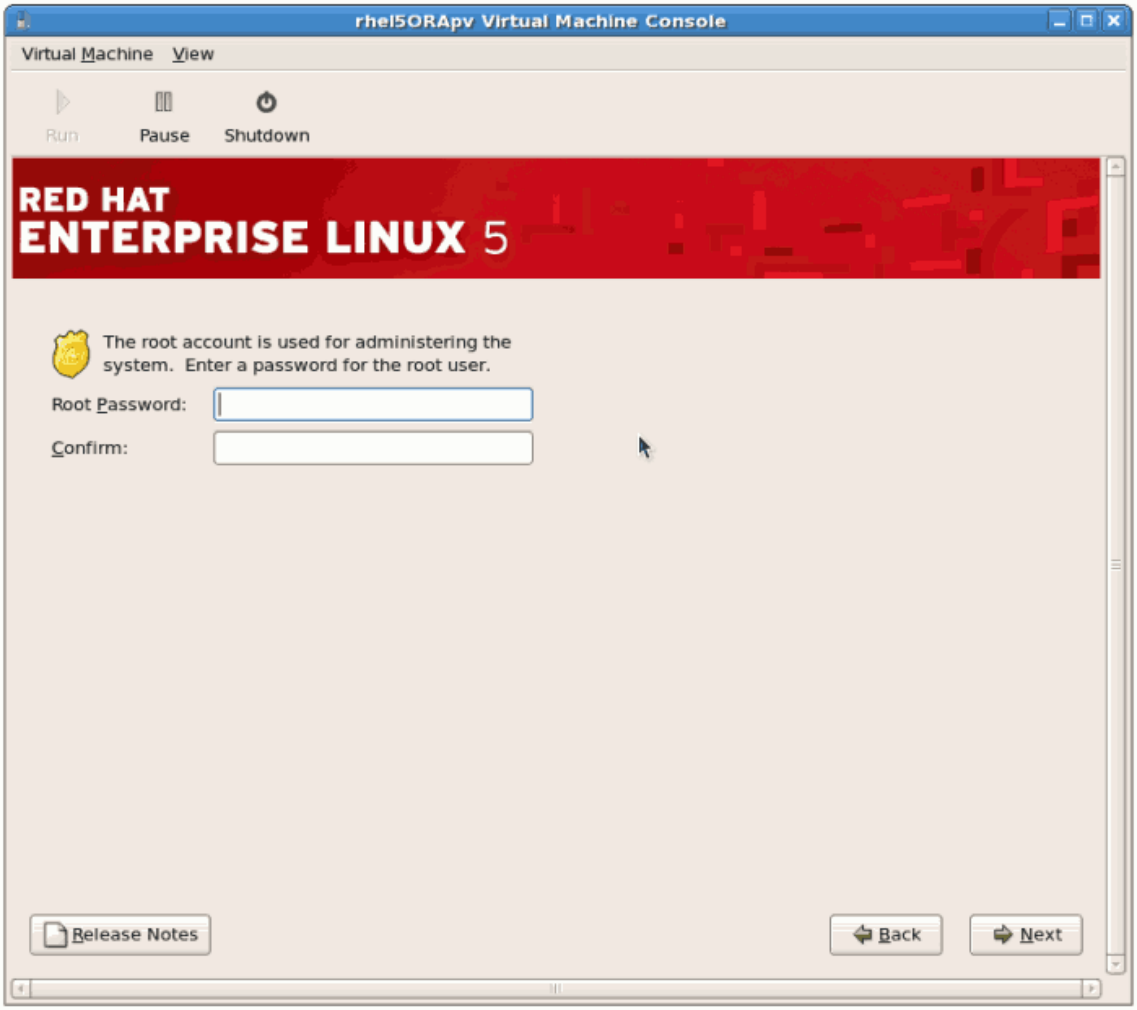


OK をクリックして継続します。

6. 現在地の適切なタイムゾーンを選択します。

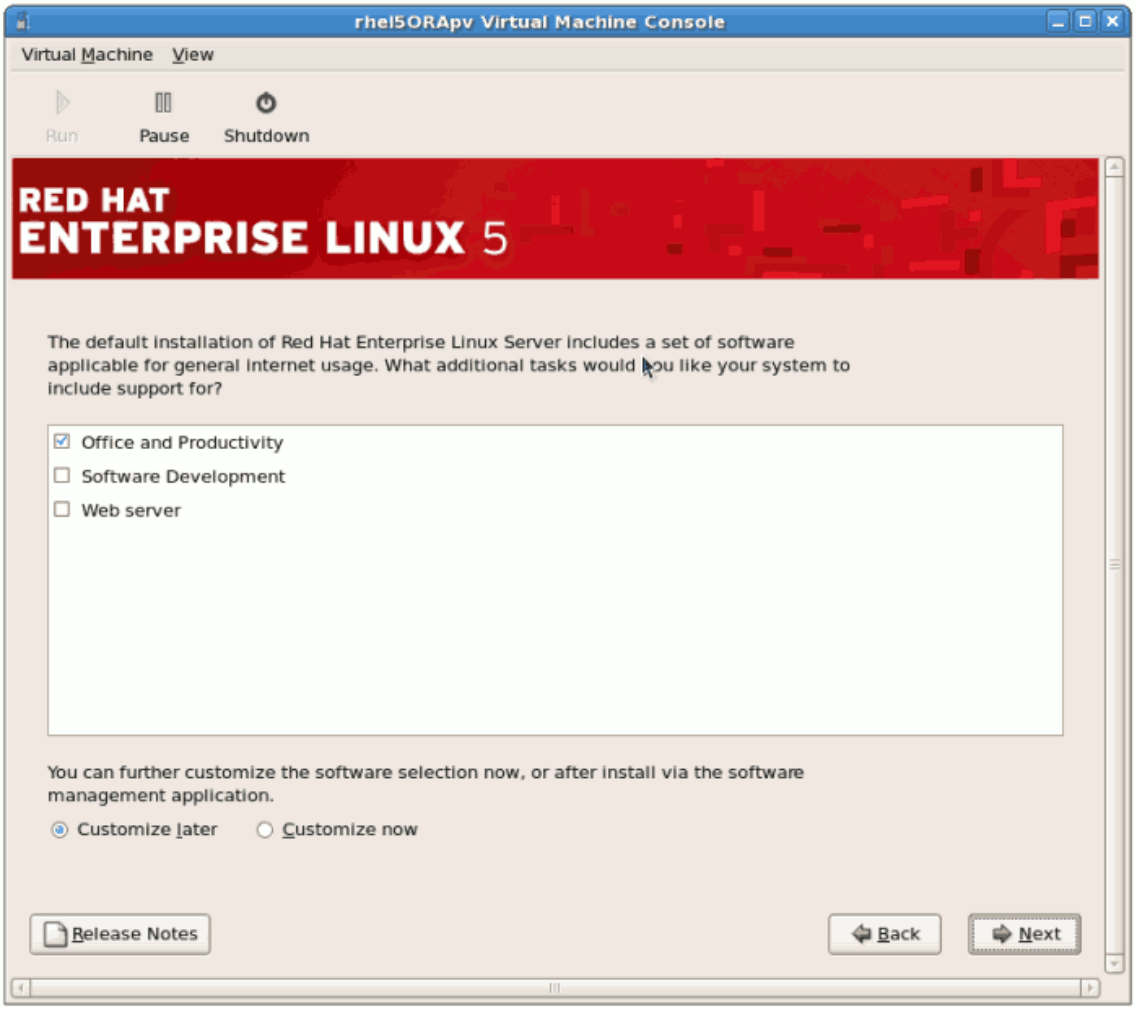


7. ゲスト用の root パスワードを入力します。



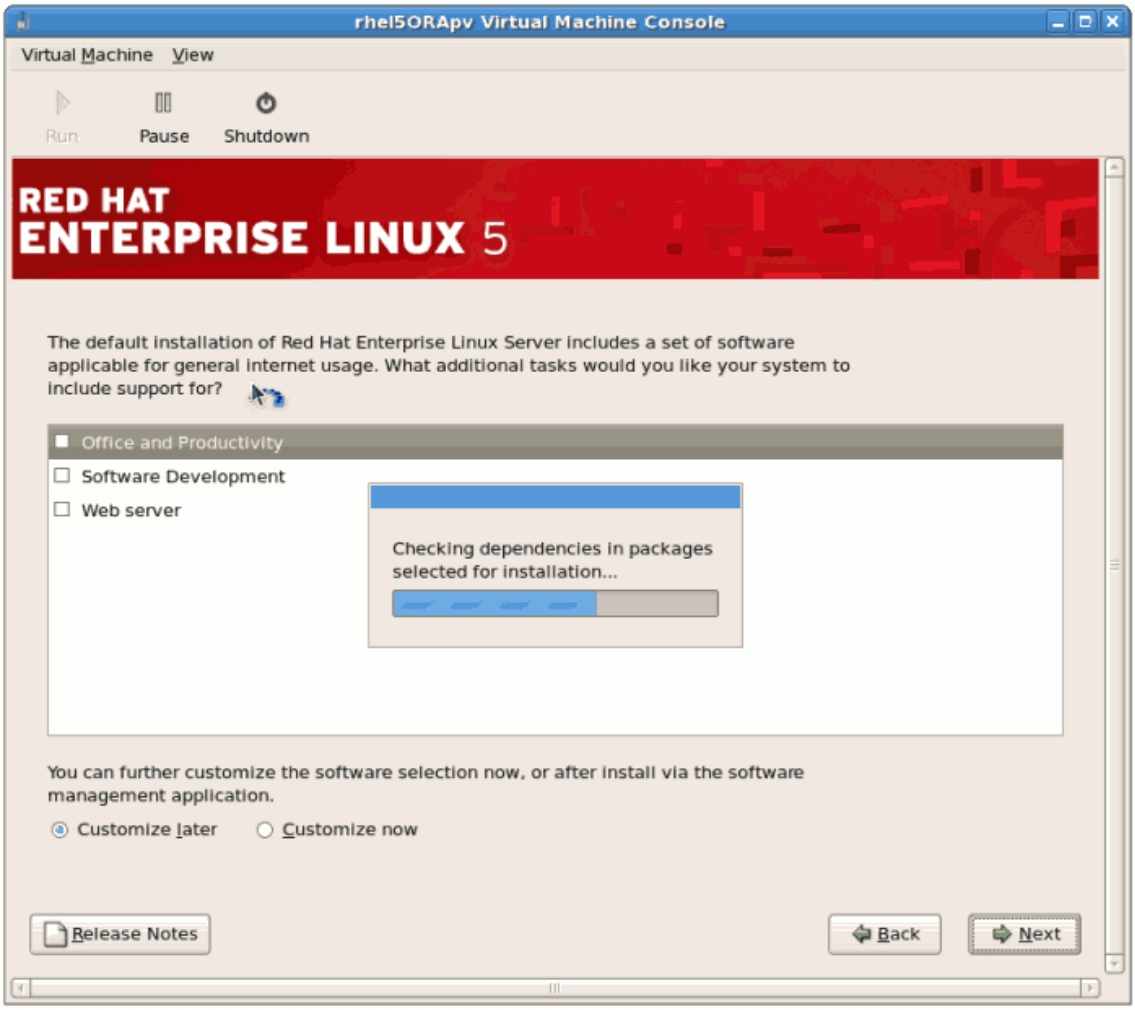
次 をクリックして継続します。

8. インストールするソフトウェアパッケージを選択します。**今すぐ カスタマイズ** ボタンを選択します。**kernel-xen** パッケージを **System** ディレクトリにインストールする必要があります。para-virtualization には **kernel-xen** パッケージが必要です。



次 をクリックします。

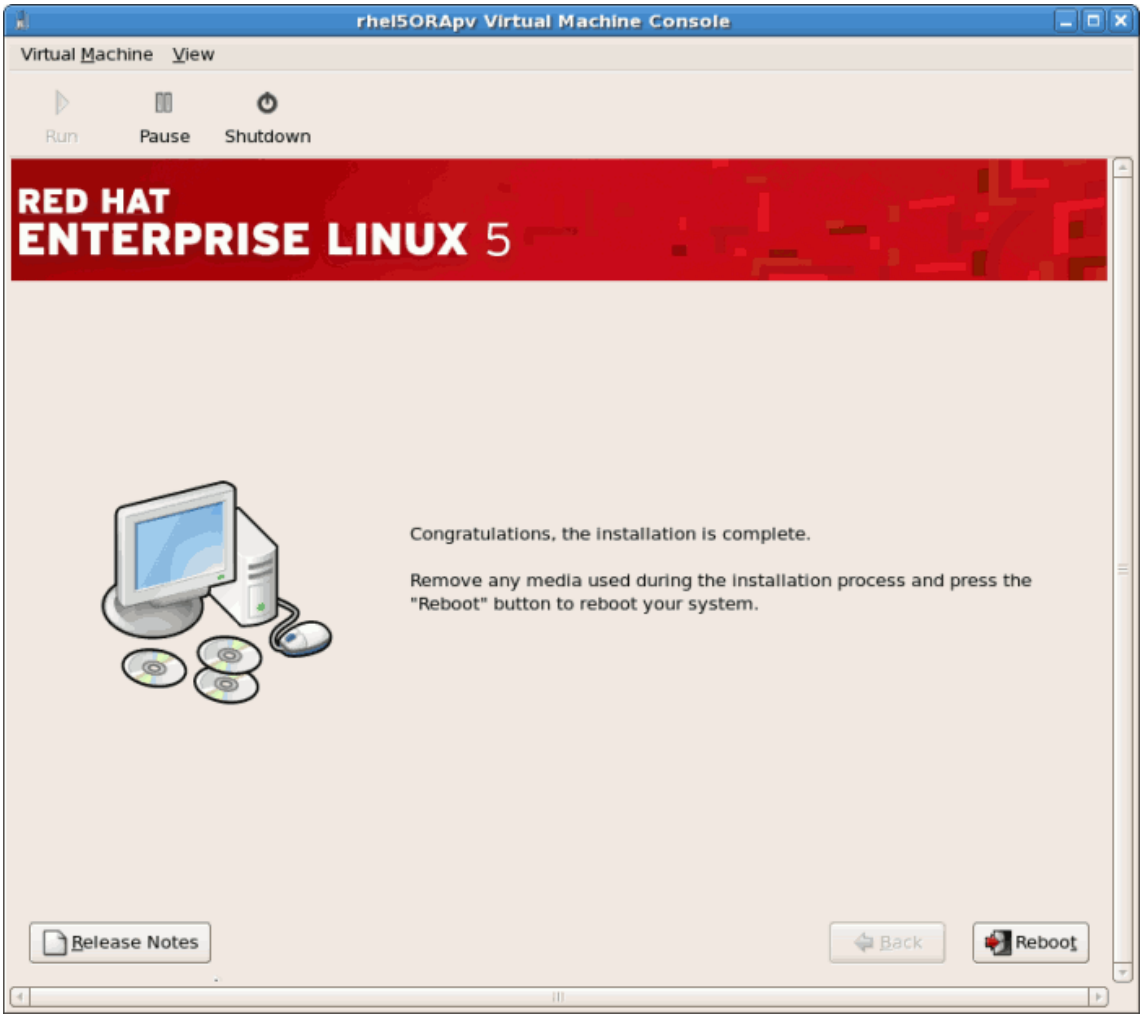
9. 依存関係と領域の要件が算出されます。



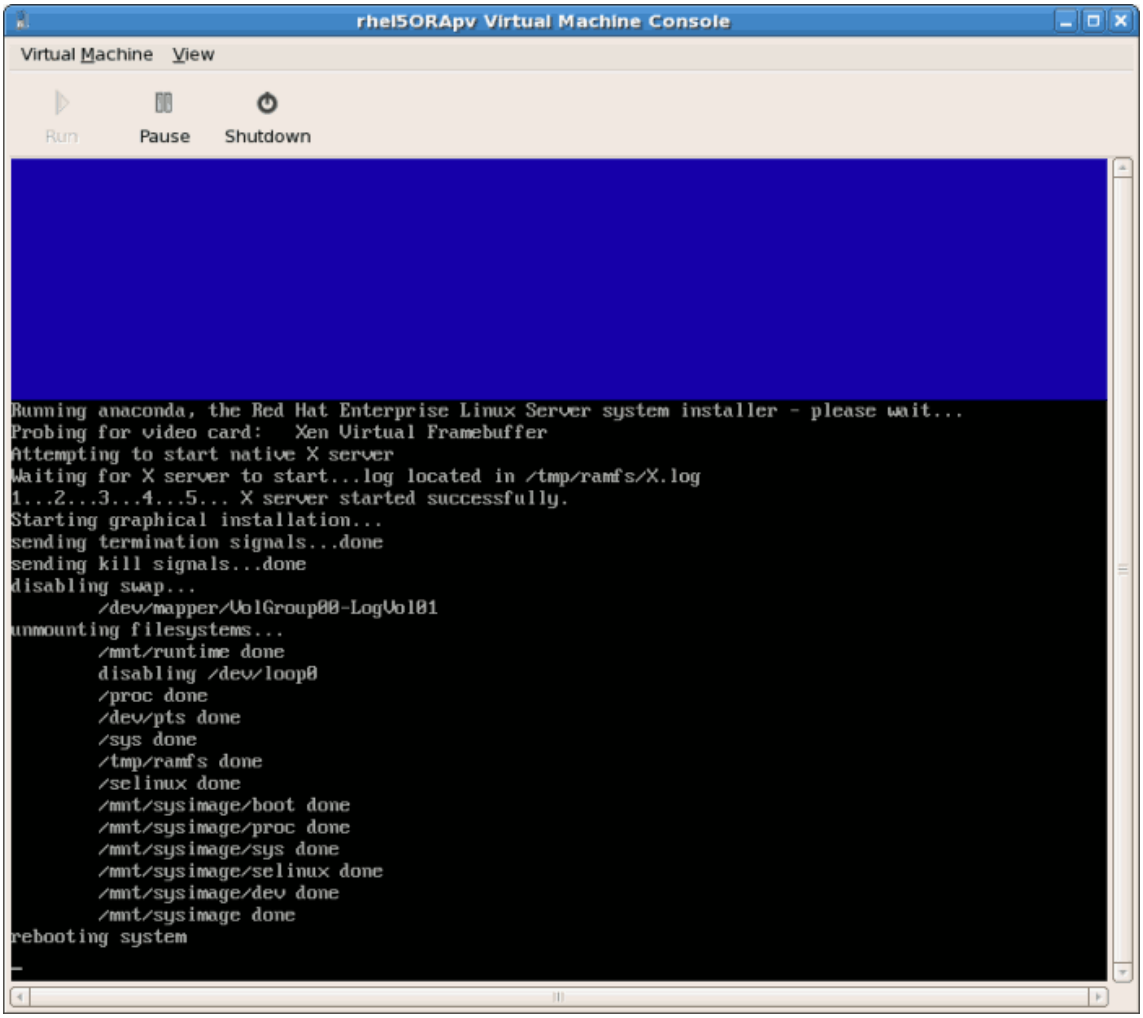
10. インストールの依存関係と領域要件が確証された後は、**次** を クリックして実際のインストールを開始します。

11. 選択したソフトウェアパッケージが全て自動的にインストールされます。

12. インストールが完了したら、ゲストを再起動します:



13. ゲストは再起動しません。代わりにシャットダウンします...



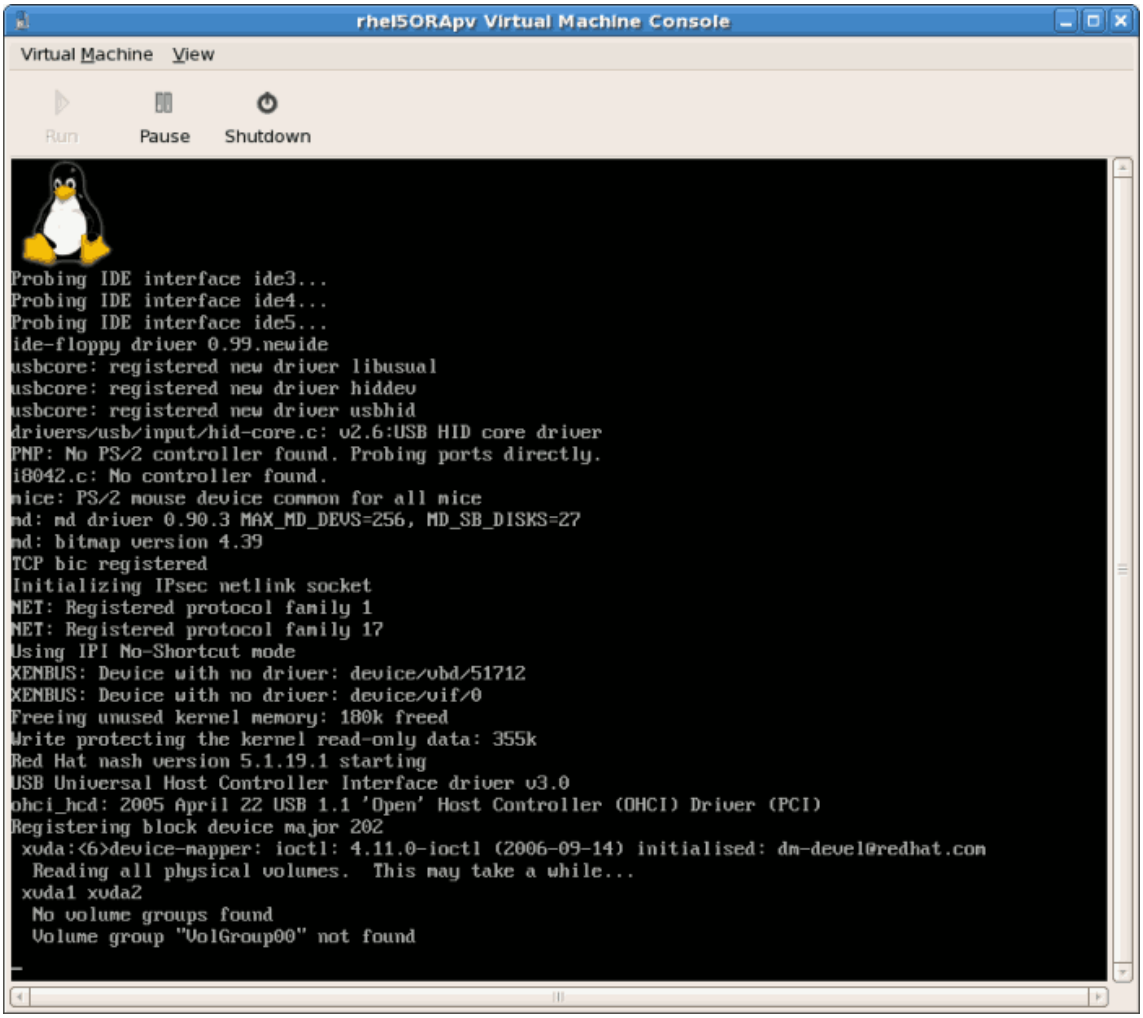
14. ゲストをブートします。ゲスト名は 「Red Hat Enterprise Linux 5.0 を para-virtualized ゲストとしてインストール」 内で **virt-install** を使用した時に選択されていました。デフォルトの 例を使用した場合は、名前は *rhel5PV* となります。

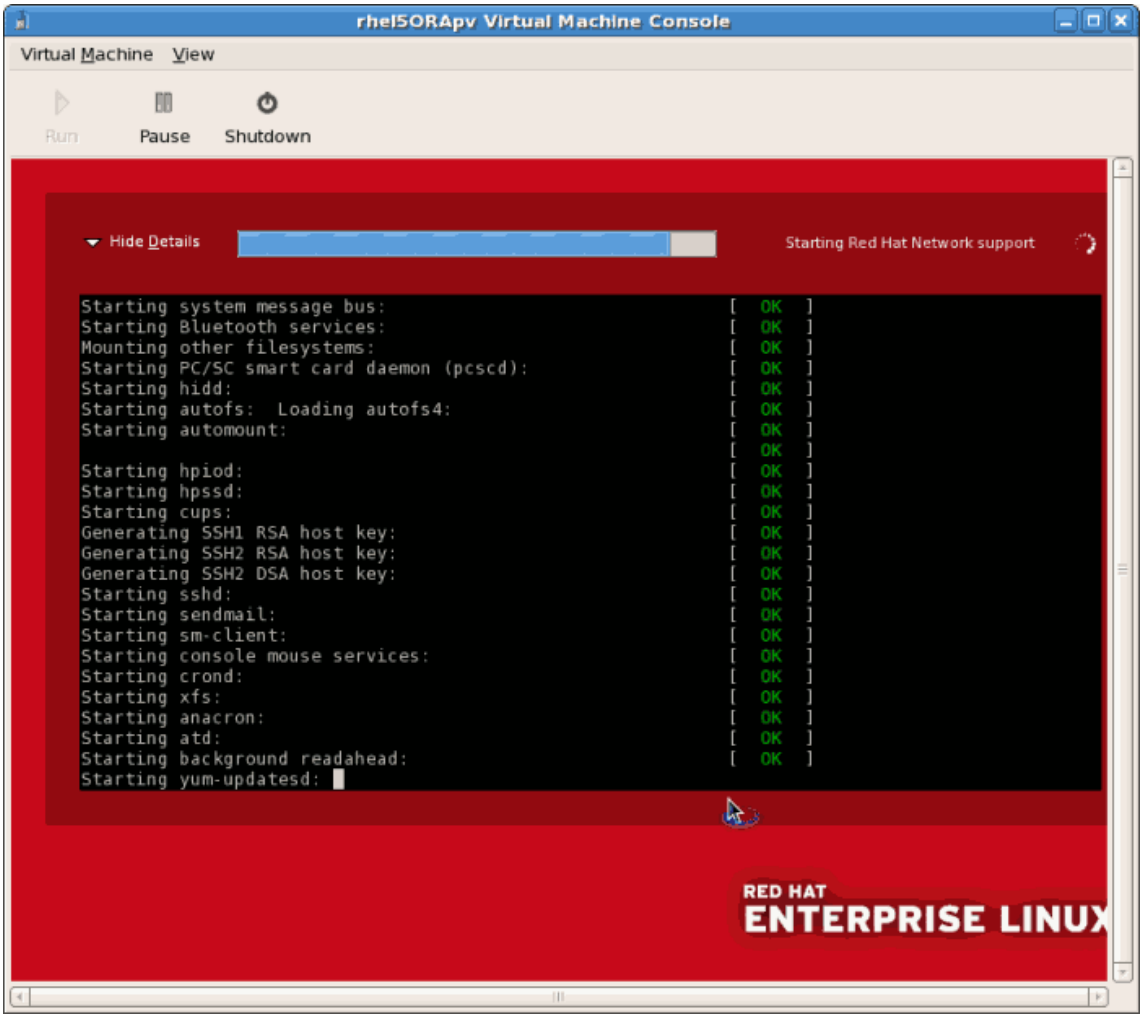
    virsh の使用でゲストの再起動ができます:

    ```
    # virsh reboot rhel5PV
    ```

    別の方法として、**virt-manager** を開いて、ゲスト用の 名前を選択し、**開く** をクリックします。それから **実行** をクリックします。

    ゲストのブートプロセスを表示する **VNC** ウィンドウが ここで開きます。


rhel5ORApv Virtual Machine Console
Virtual Machine View
Run
Pause
Shutdown
Probing IDE interface ide3...
Probing IDE interface ide4...
Probing IDE interface ide5...
ide-floppy driver 0.99.newide
usbcore: registered new driver libusual
usbcore: registered new driver hiddev
usbcore: registered new driver usbhid
drivers/usb/input/hid-core.c: v2.6:USB HID core driver
PNP: No PS/2 controller found. Probing ports directly.
i8042.c: No controller found.
mice: PS/2 mouse device common for all mice
md: md driver 0.90.3 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: bitmap version 4.39
TCP bic registered
Initializing IPsec netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI No-Shortcut mode
XENBUS: Device with no driver: device/vbd/51712
XENBUS: Device with no driver: device/vif/0
Freeing unused kernel memory: 180k freed
Write protecting the kernel read-only data: 355k
Red Hat nash version 5.1.19.1 starting
USB Universal Host Controller Interface driver v3.0
ohci_hcd: 2005 April 22 USB 1.1 'Open' Host Controller (OHCI) Driver (PCI)
Registering block device major 202
xvda:<6>device-mapper: ioctl: 4.11.0-ioctl (2006-09-14) initialised: dm-devel@redhat.com
Reading all physical volumes. This may take a while...
xvda1 xvda2
No volume groups found
Volume group "VolGroup00" not found



15. ゲストをブートすると **First Boot** 設定の画面がスタートします。 このウィザードは使用するゲスト用にいくつかの基本的な設定選択を催促してきます。

16. ライセンス同意書を読んで同意して下さい。

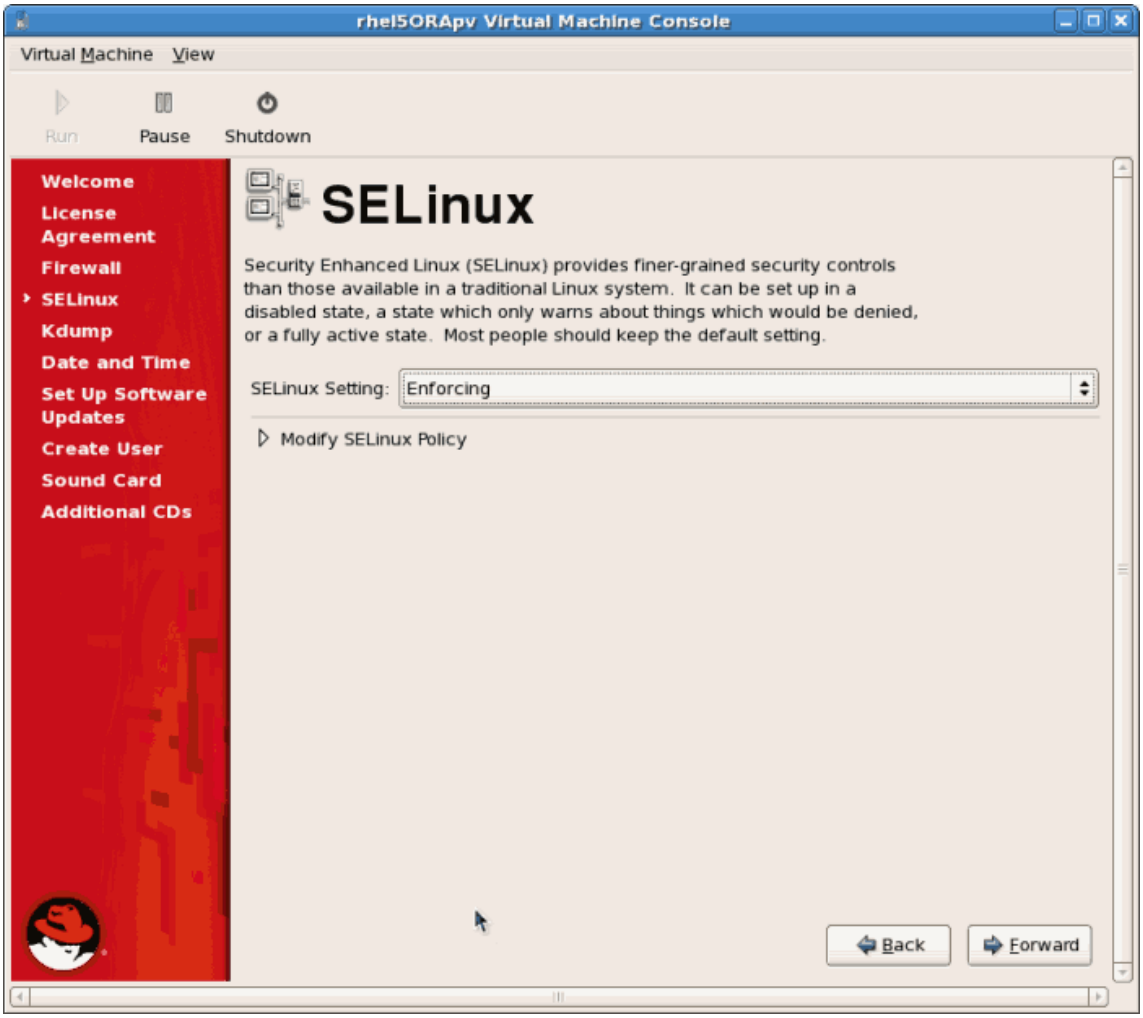
ライセンス同意のウィンドウで **進む** をクリックします。

17. ファイアウォールを設定します。

進む をクリックして継続します。

a. ファイアウォールを無効にすると、その選択を確定するように催促されます。 **はい** をクリックして確定し、継続します。但し、 ご自分のファイアウォールを無効にすることは推奨できません。

18. SELinux を設定します。SELinux を **強制モード** で実行する ことが強く推奨されます。SELinux は許容モードで実行するか、無効にすることも 選択できます。



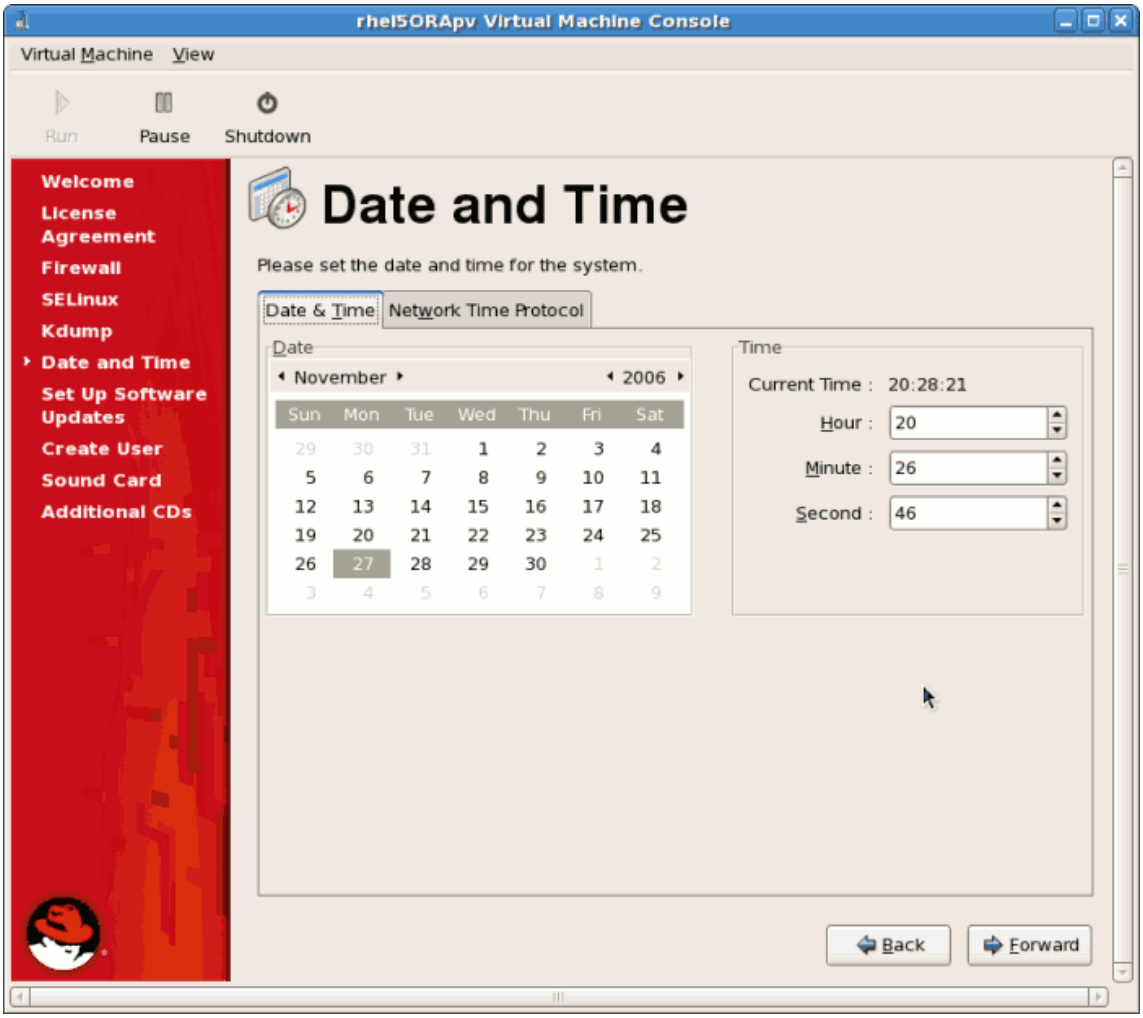
進む をクリックして継続します。

a. SELinux を無効にする選択をすると、この警告が表示されます。**はい** をクリックすると SELinux が 無効になります。

19. 必要であれば、**kdump** を有効にします。

進む をクリックして継続します。

20. ゲスト用の時刻と日付が正しいかどうか確認します。**para-virtualized** ゲストを インストールすると、時刻と日付は **hypervisor** と同期化するはずです。

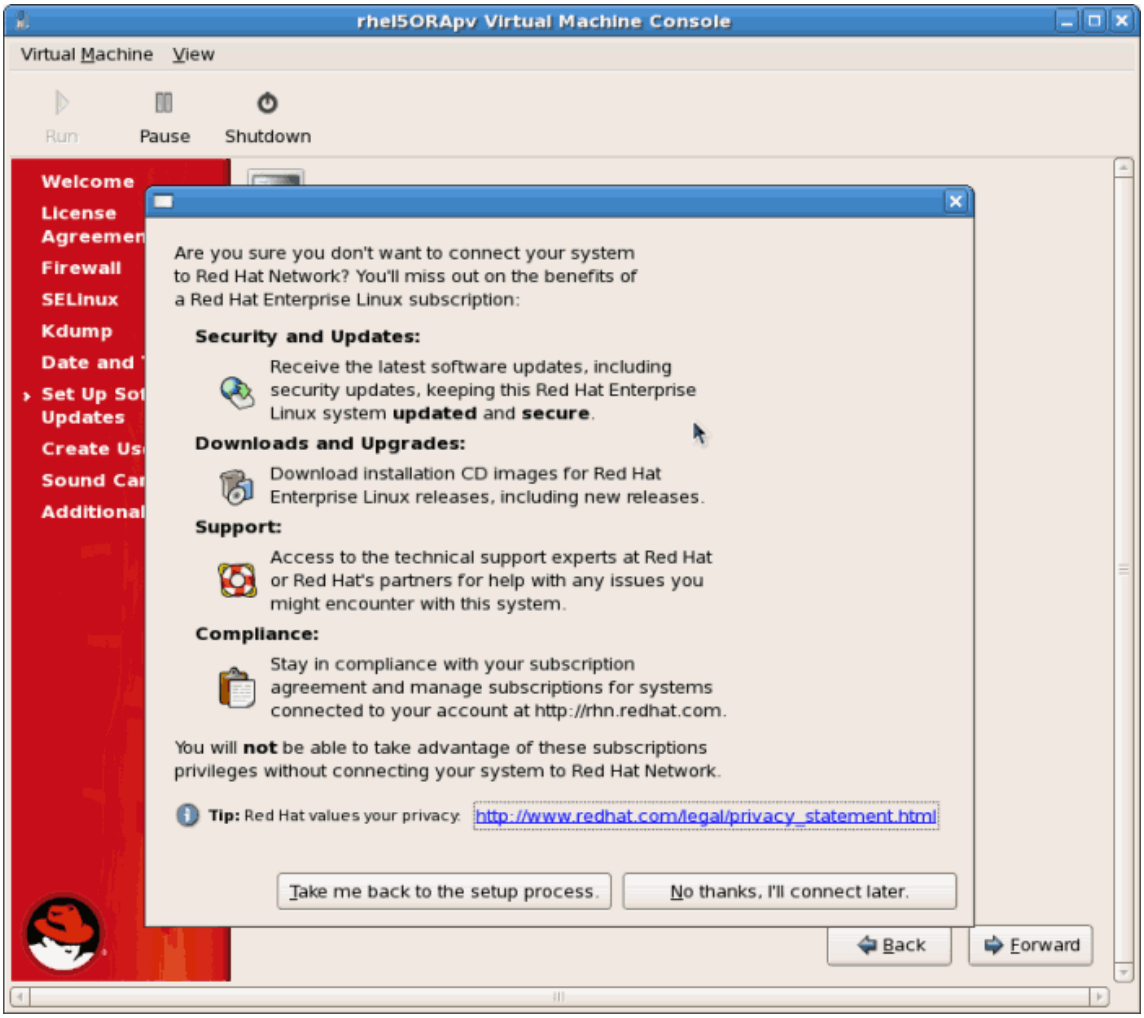


進む をクリックして継続します。

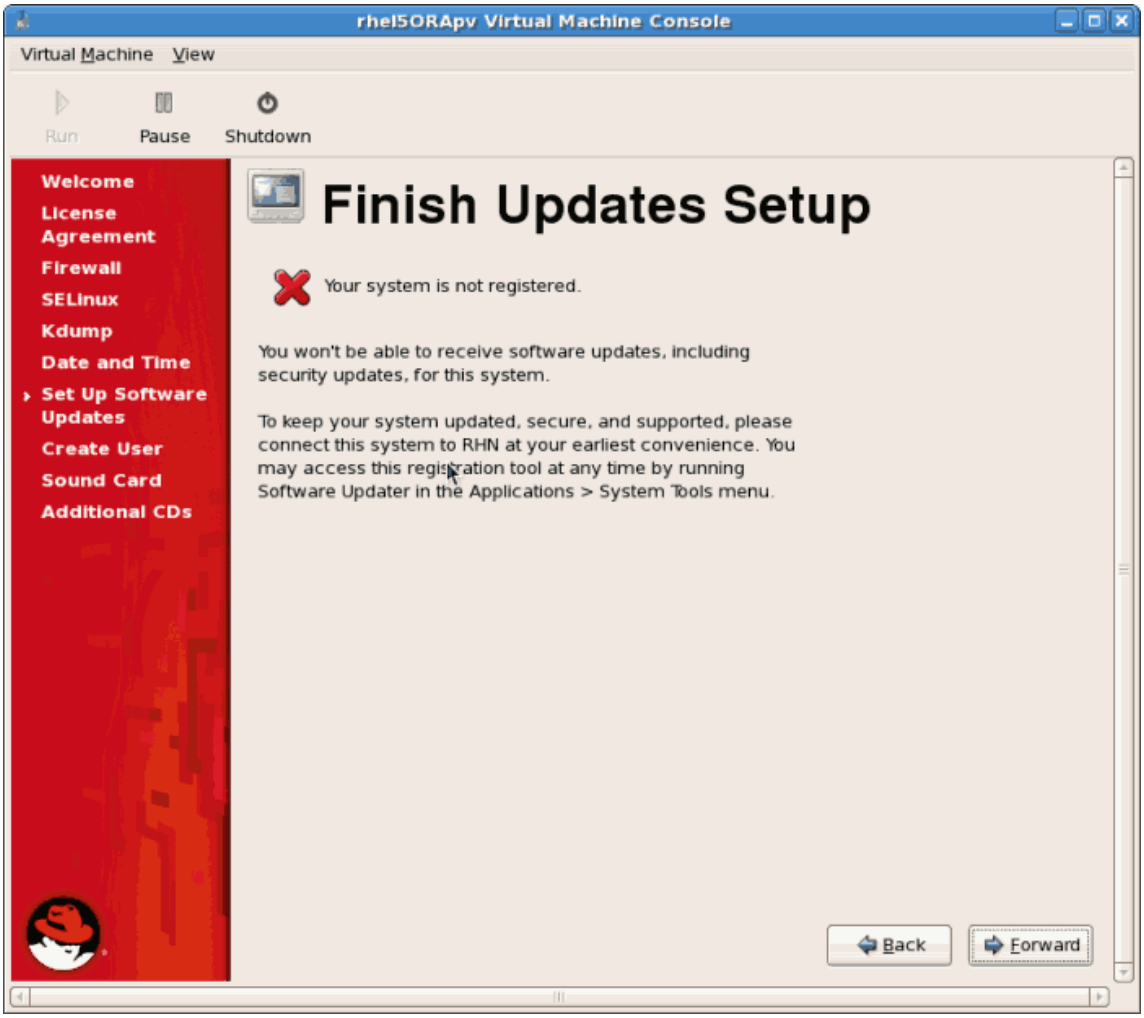
21. ソフトウェア更新をセットアップします。Red Hat Network サブスクリプションを持っている場合、 あるいは、そのお試し版を欲しい場合は、以下の画面を使用して新規にインストールしたゲストを RHN に登録します。

進む をクリックして継続します。

a. RHN 用の選択を確定します。

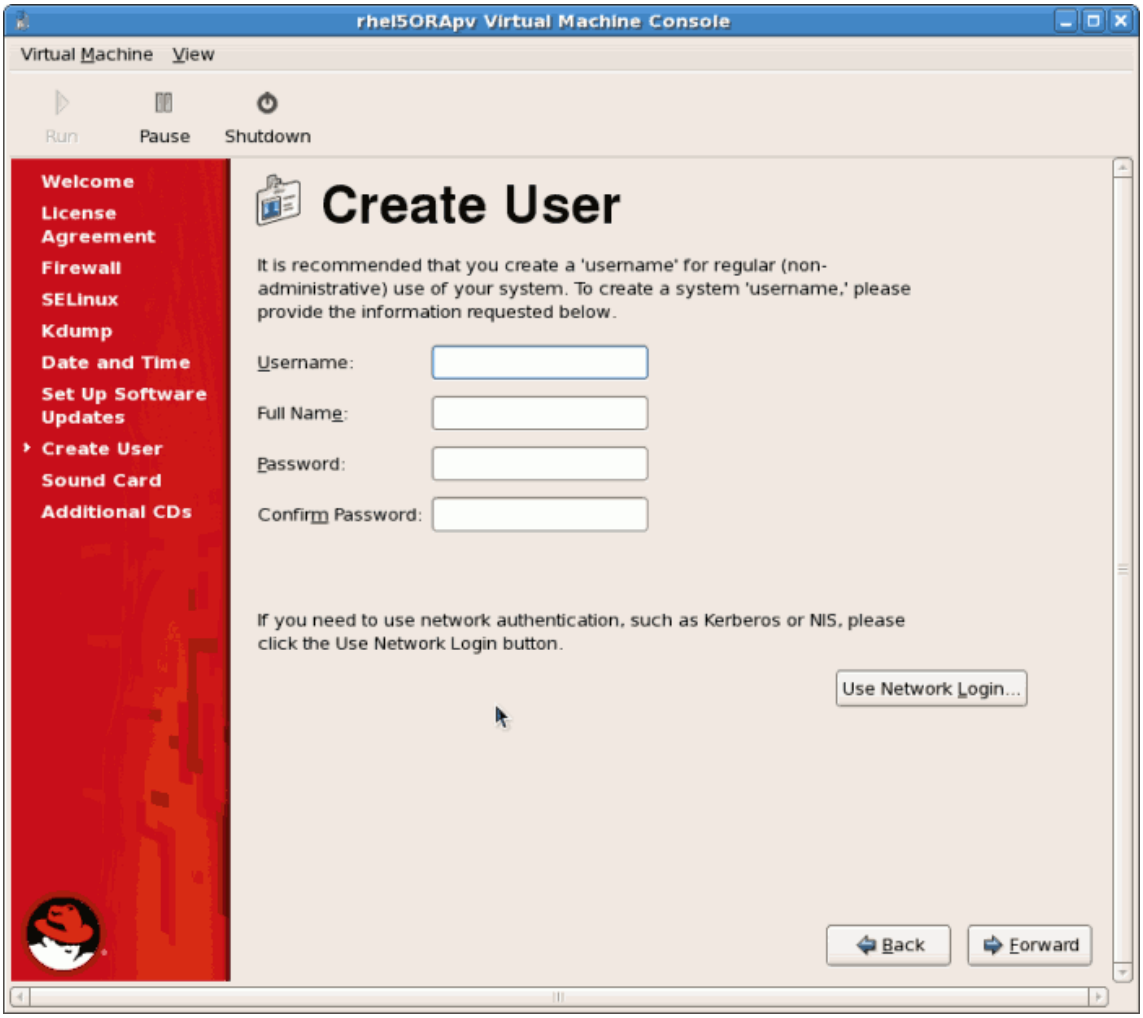


b. RHN アクセスを設定していない場合は、もう１つの画面が出てくるでしょう。 RHN アクセスが有効になっていなければ、ソフトウェア更新は受け取れません。



進む ボタンをクリックします。

22. root 以外のユーザーアカウントを作成します。通常の使用と強化セキュリティの為には root 以外のユーザーアカウントの作成が推奨されます。ユーザー名、氏名、及びパスワードを 入力します。

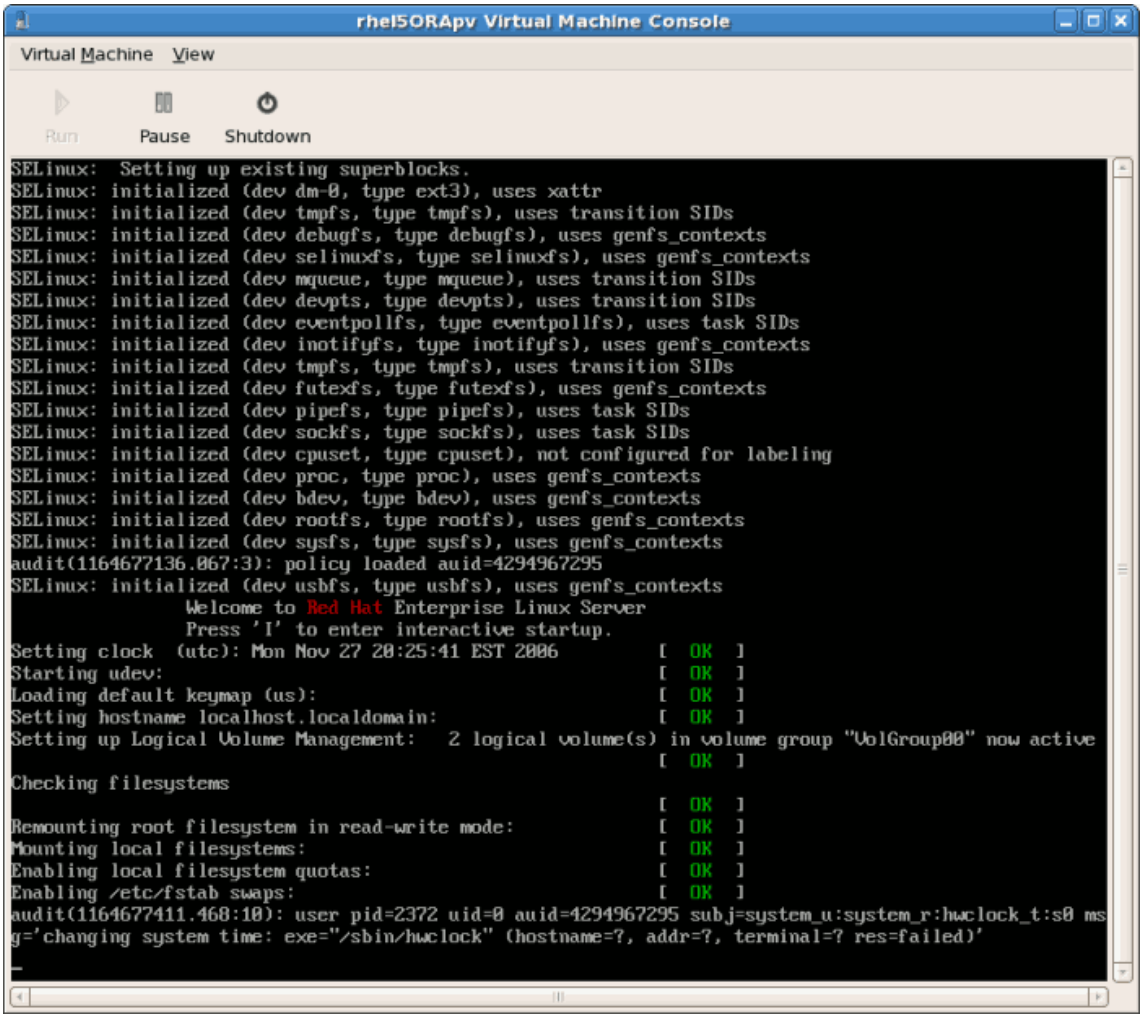


進む ボタンをクリックします。

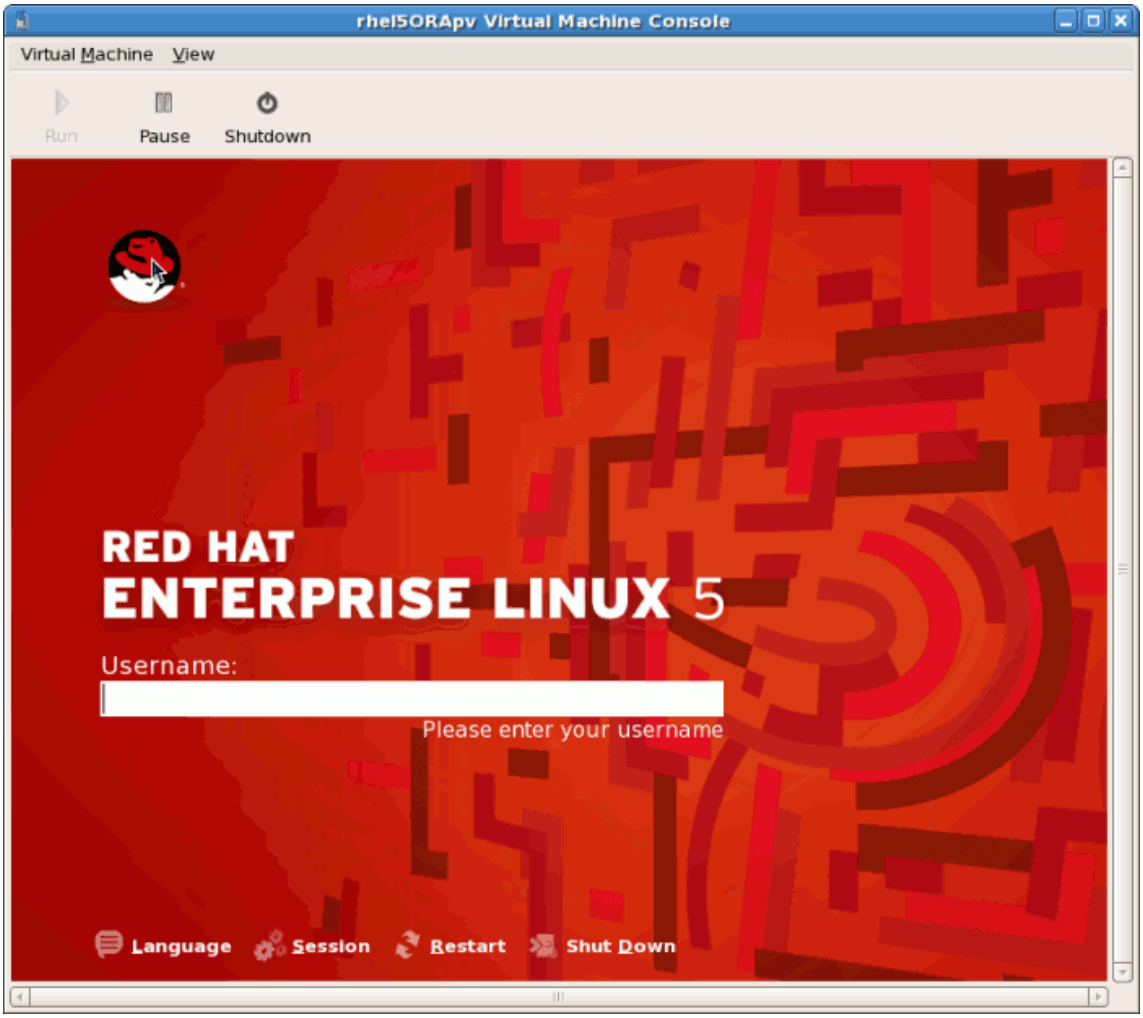
23. サウンドデバイスが検出されて、サウンドが必要であれば、それを調節します。 プロセスを完了して、**進む** をクリックします。

24. この画面を使用して、CD から又は別のレポジトリから追加のパッケージをインストールできます。 この時点では追加のソフトウェアをインストールしないで、後で **yum** コマンドを使用するか、RHN でパッケージを追加する方が効率的です。 **完了** をクリックします。

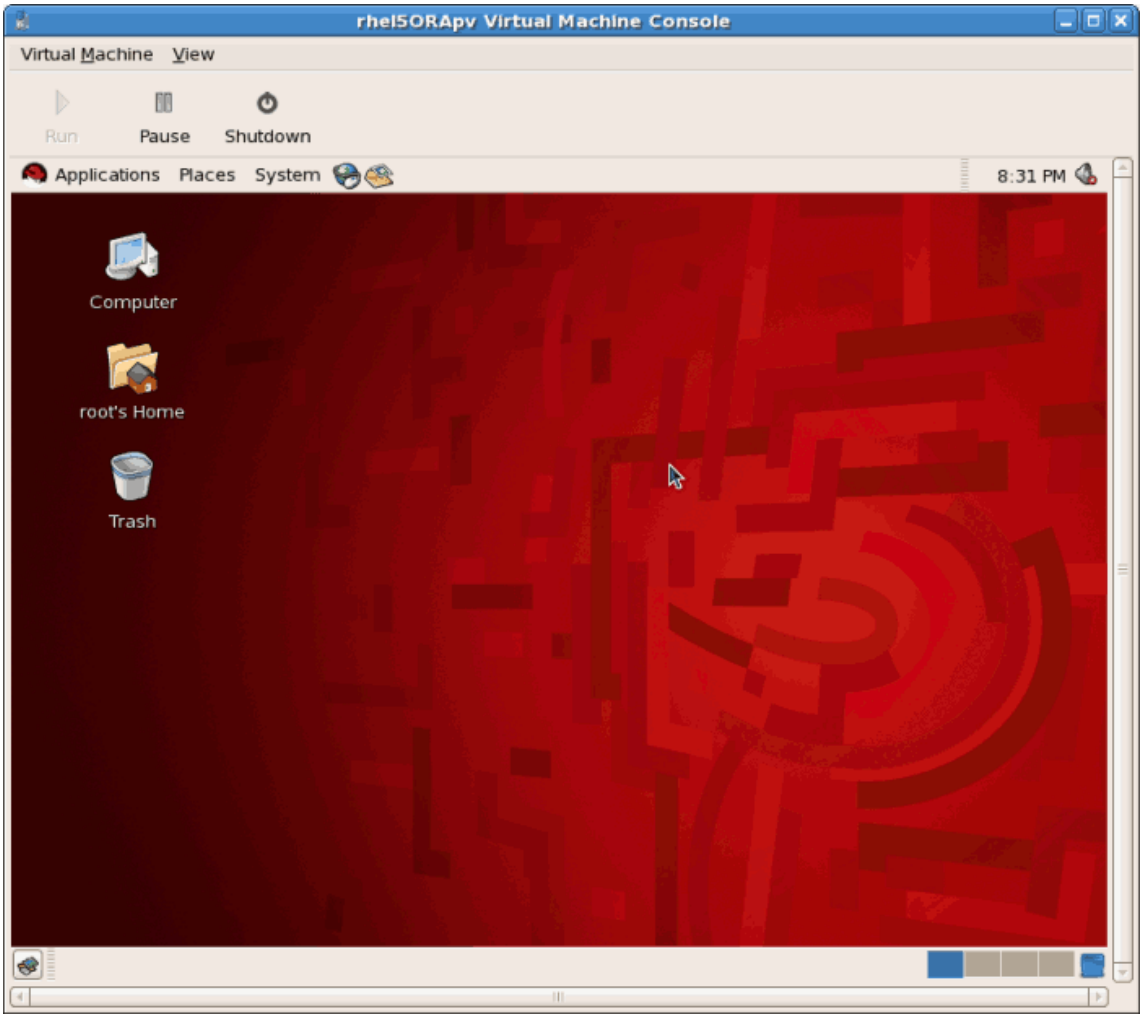
25. ゲストはここで、ユーザーが変更したセッティングを設定し、ブートプロセスを 継続します。



26. Red Hat Enterprise Linux 5 のログイン画面が表示されます。先のステップで 作成したユーザー名を使用してログインします。



27. これで、 para-virtualized Red Hat Enterprise Linux ゲストが正常にインストールされました。



7.2. RED HAT ENTERPRISE LINUX を完全仮想化ゲストとしてインストール

このセクションでは、完全仮想化の Red Hat Enterprise Linux 5 ゲストの インストールを説明しています。KVM hypervisor は、 Red Hat Enterprise Linux 5.4 又は それ以降を必要とします。

手順7.3 virt-manager を使用して完全仮想化 Red Hat Enterprise Linux 5 ゲストを作成

1. **virt-manager を開く**
   `virt-manager` を開始します。**アプリケーション** メニューと **システムツール**サブメニューから **仮想マシン マネージャ** のアプリケーションを起動します。別の方法として、**root** になって **`virt-manager`** コマンドを実行することもできます。

2. **Hypervisor を選択**
   hypervisor を選択します。Xen か KVM がインストールされている場合は、そのいずれかを 選択します。例えば、KVM を選択する場合、現在、KVM は **qemu** と命名されていることに注意して下さい。

   まだ hypervisor を接続していない場合は接続します。**ファイル** メニューを開いて **接続の追加...** オプションを選択します。 「開放接続のウィンドウ」 を参照して下さい。

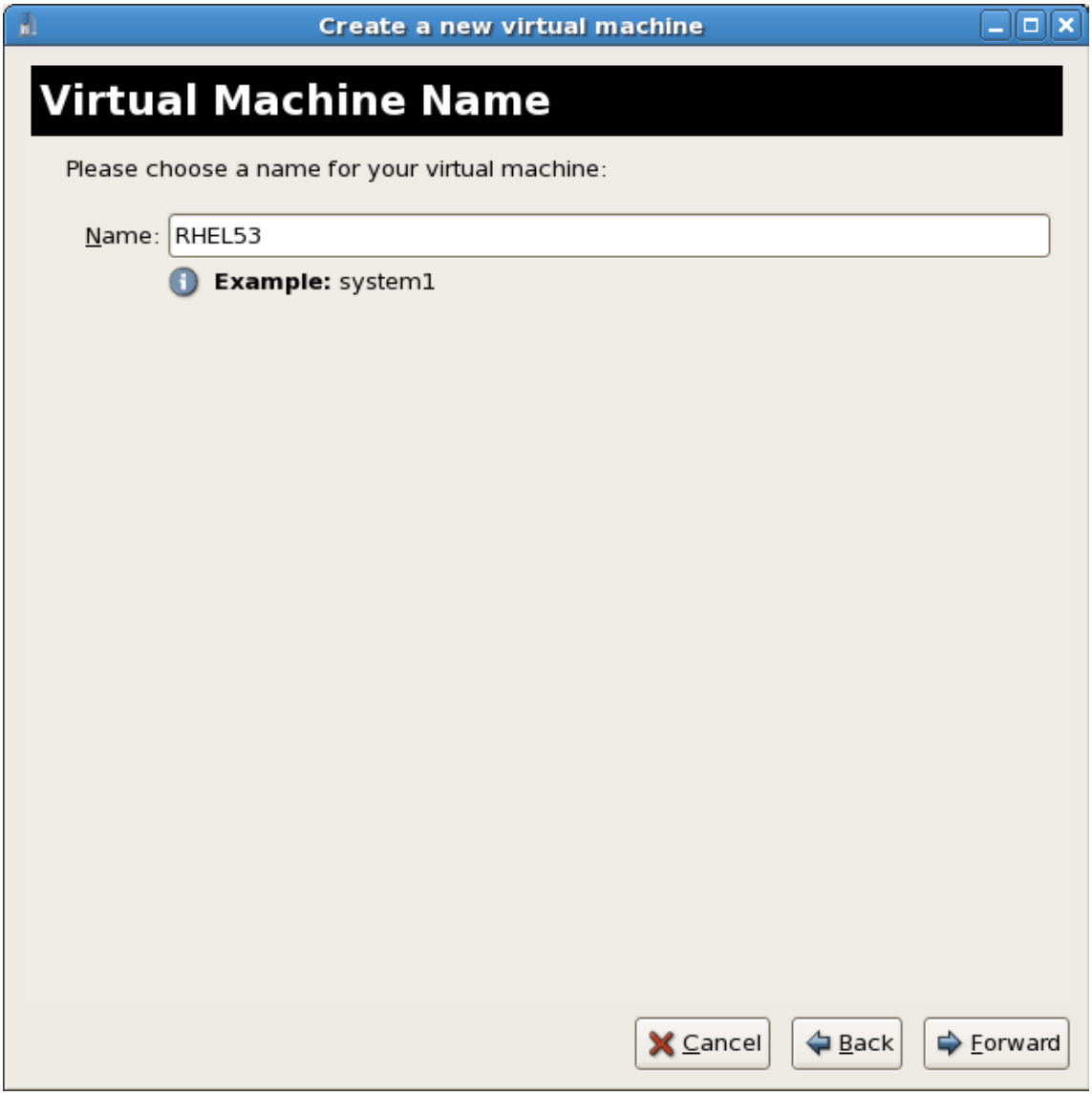
   hypervisor 接続が選択されると、**新規** ボタンが 利用可能になります。**新規** ボタンをクリックします。

3. **新規仮想マシンウィザードを開始**

新規 ボタンをクリックすると、仮想マシン作成ウィザードが スタートします。

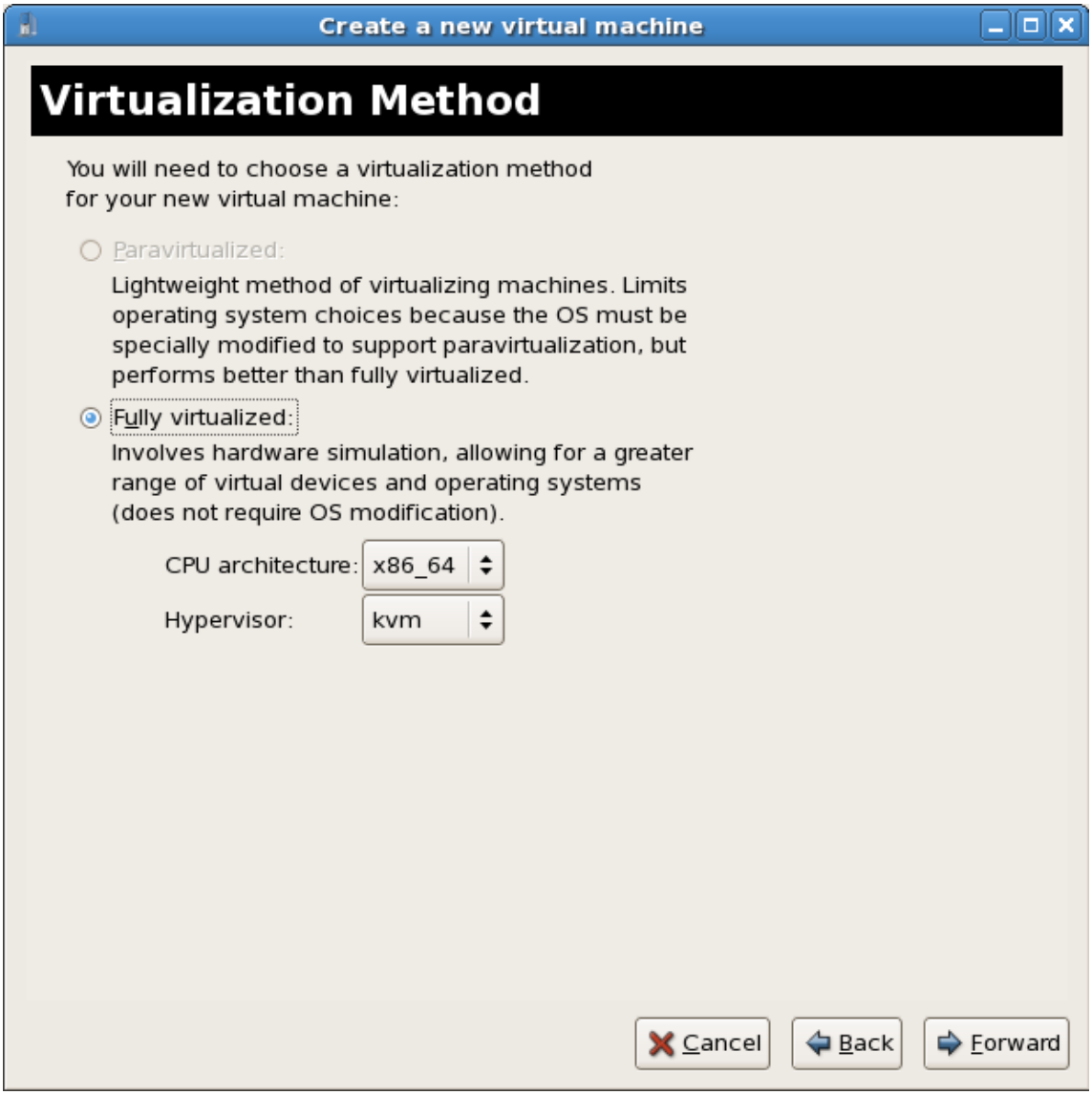
進む をクリックして継続します。

4. **仮想マシンの命名**
   使用する仮想ゲストの名前を付けます。句読点と空白文字は許可されません。



進む をクリックして継続します。

5. **仮想化のメソッドを選択**
   仮想化ゲスト用の仮想化メソッドを選択します。インストール済みの仮想化メソッドのみを 選択できることに注意して下さい。先の手順で KVM か Xen を選択している場合、 (ステップ 4) その選択している hypervisor を使用しなければなりません。 ここの例では、KVM hypervisor を使用します。



進む をクリックして継続します。

6. **インストールメソッドの選択**
   Red Hat Enterprise Linux は以下の方法の１つを使用してインストールできます:

   - **ローカルインストールメディア**：ISO イメージか、又は 物理光学メディア
   - Red Hat Enterprise Linux のインストールツリーが、HTTP、 FTP 、又は NFS を介して ホストされている場合は、**「ネットワークインストールツリー」** を 選択します。
   - Red Hat Enterprise Linux インストールメディアのブート用に PXE サーバーを設定している場合は、 PXE を使用することができます。Red Hat Enterprise Linux インストールを PXE ブートするように サーバー設定する方法はこのガイドでは言及しません。しかし、メディアブートの後はほとんどの インストール手順は同じです。

   スクリーンショットで示してあるように、**OS タイプ** を **Linux** に、 そして **OS 変種** を **Red Hat Enterprise Linux 5** に セットします。

進む をクリックして継続します。

7. **インストールメディアの場所を指定**
   ISO イメージの位置、又は CD-ROM か DVD のデバイスを選択します。ここの例では、 Red Hat Enterprise Linux installation DVD の ISO ファイルイメージを使用します。

   a. **閲覧** ボタンをクリック

   b. ISO ファイルの位置を検出して ISO イメージを選択します。**開く** を クリックして選択を確定します。

   c. ファイルが選択されてインストール準備ができました。

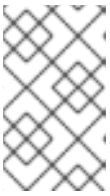
進む をクリックして継続します。

警告

ISO イメージファイルとゲストストレージイメージには、**/var/lib/libvirt/images/** ディレクトリの使用が推奨されます。他の場所では、追加の SELinux 設定が必要になります。詳細には、「SELinux と仮想化」 を参照して下さい。

8. **ストレージセットアップ**
   物理ストレージデバイス **(ブロックデバイス)** 又は ファイルベースイメージ **(ファイル)** を割り当てます。 ファイルベースイメージは **/var/lib/libvirt/images/** ディレクトリに 格納しなければなりません。使用する仮想化ゲストには、十分なストレージを割り当てます。 仮想化ゲストとそれが必要とするアプリケーションにも十分なスペースを割り当てます。

進む をクリックして継続します。



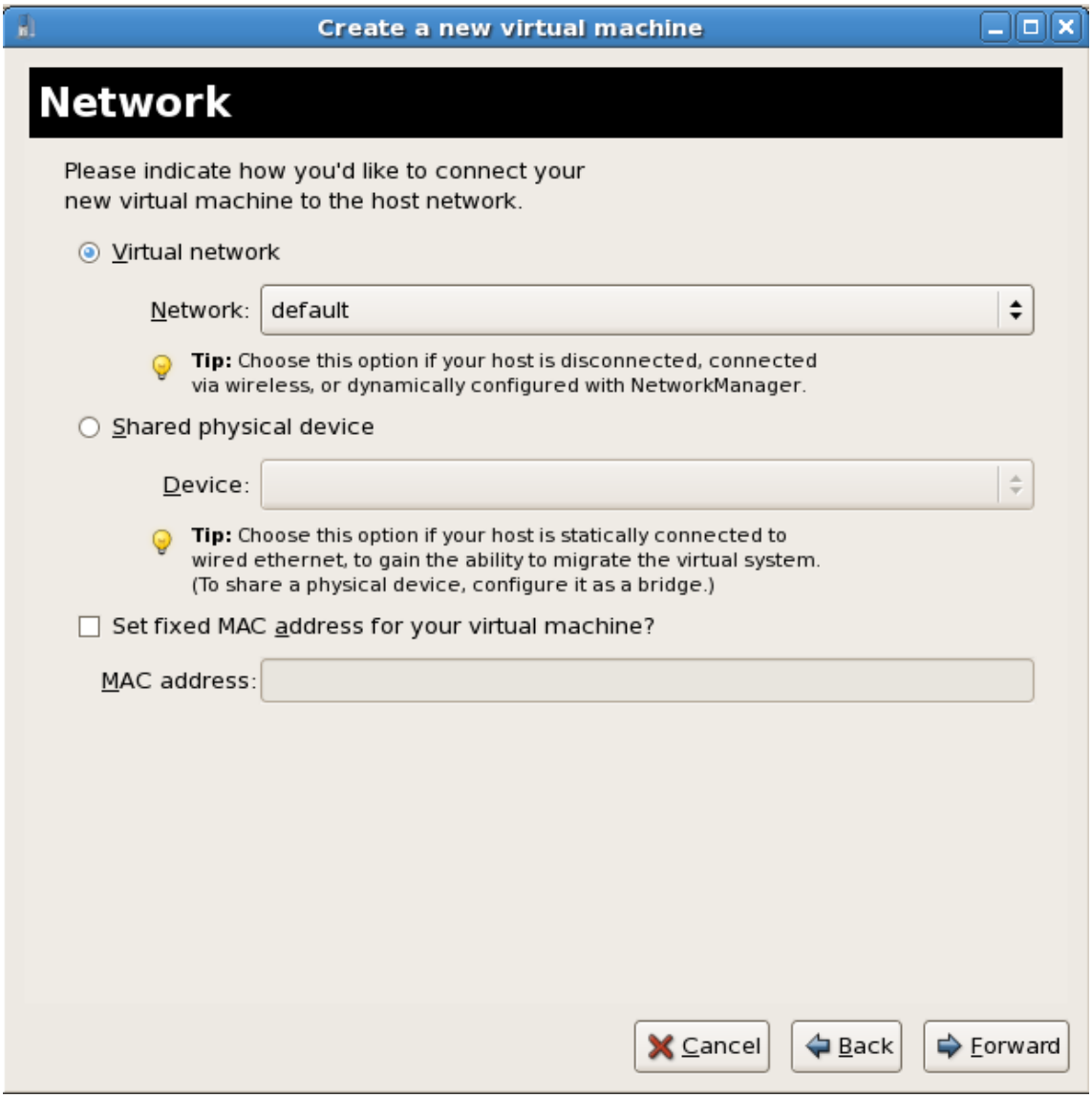
注記

ライブとオフラインの移行には、ゲストが共有のネットワークストレージにインストール されている必要があります。ゲスト用の共有ストレージの設定についての情報には、 9章*共有ストレージと仮想化*を 参照して下さい。

9. **ネットワーク設定**
   仮想ネットワーク 又は **共有の 物理デバイス** のどちらかを選択します。

   仮想ネットワークオプションは、NAT（Network Address Translation）を 使用してデフォルトのネットワークデバイスを仮想化ゲストと共有します。 ワイヤレスネットワークには仮想ネットワークオプションを使用します。

   共有の物理デバイスオプションはネットワークボンドを使用して、仮想化ゲストに ネットワークデバイスへの全面的アクセスを与えます。

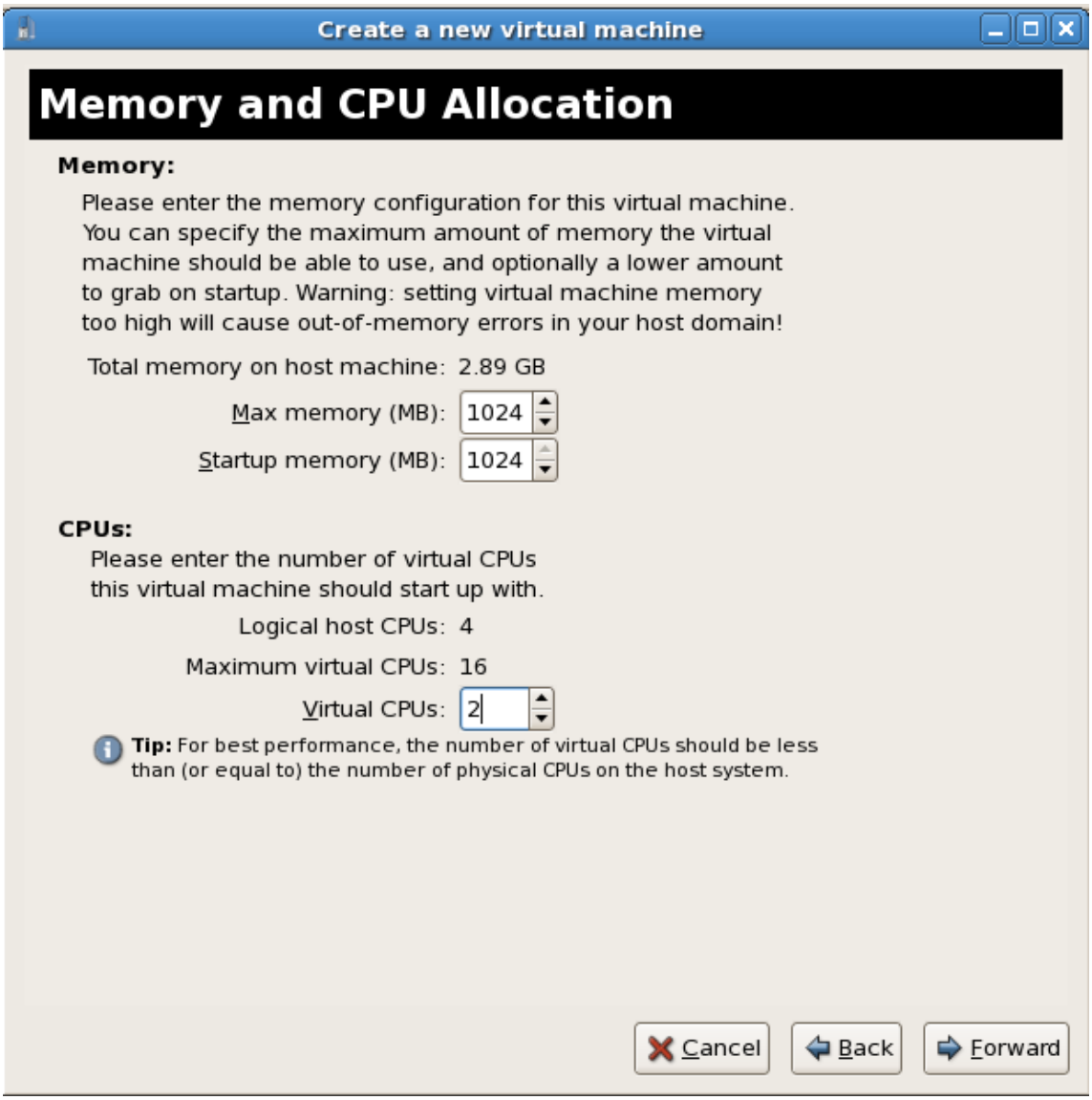


進む をクリックして継続します。

10. **メモリーと CPU の割り当て**
メモリーと CPU の割り当てウィンドウが表示されます。仮想化 CPU と RAM の割り当てに 適切な値を選択します。これらの値は、ホストとゲストのパフォーマンスに影響を与えます。

仮想化ゲストは、効率的にそして効果的に稼働するために十分な物理メモリー (RAM) を必要とします。使用するゲストオペレーティングシステムとアプリケーションの 必要性に適合するメモリーの値を選択します。ゲストは物理 RAM を使用することを忘れないで下さい。ホストシステムに対し、過度の数のゲストを稼働したり、不十分なメモリーを設定していると、仮想メモリーとスワップをかなり使用することになります。仮想メモリーは 確実に低速であり、システムパフォーマンスと反応性の低下の原因となります。 全てのゲストとホストが効率的に稼働できるように十分なメモリーを割り当てることを確認して下さい。

十分な仮想 CPU を仮想ゲストに割り当てます。ゲストがマルチスレッドのアプリケーションを実行する場合は、ゲストが効率良く実行するのに必要な仮想化 CPU の数を割り当てます。ホストシステム上で利用できる物理プロセッサ （又はハイパースレッド）の数量以上の仮想 CPU を割り当てないで下さい。仮想プロセッサの超過割り当ては可能ですが、超過割り当ては、プロセッサのコンテキストがオーバーヘッドを切り替えるため、ゲストとホストのパフォーマンスに重大な否定的影響を与えます。



進む をクリックして継続します。

11. **確認してゲストインストールを開始**
設定を確認します。

完了 をクリックしてゲストインストール工程を開始します。

12. **Red Hat Enterprise Linux のインストール**
    Red Hat Enterprise Linux 5 のインストールシーケンスを完了します。この インストールシーケンスは、『インストールガイド』 で説明してあります。Red Hat ドキュメント で Red Hat Enterprise Linux 『インストールガイド』 を参照して下さい。

これで、完全仮想化 Red Hat Enterprise Linux 5 ゲストはインストールできました。

7.3. WINDOWS XP を完全仮想化ゲストとしてインストール

Windows XP は完全仮想化ゲストとしてインストールできます。このセクションでは、 Windows XP を Red Hat Enterprise Linux 上に完全仮想化ゲストとしてインストール する方法を説明しています。

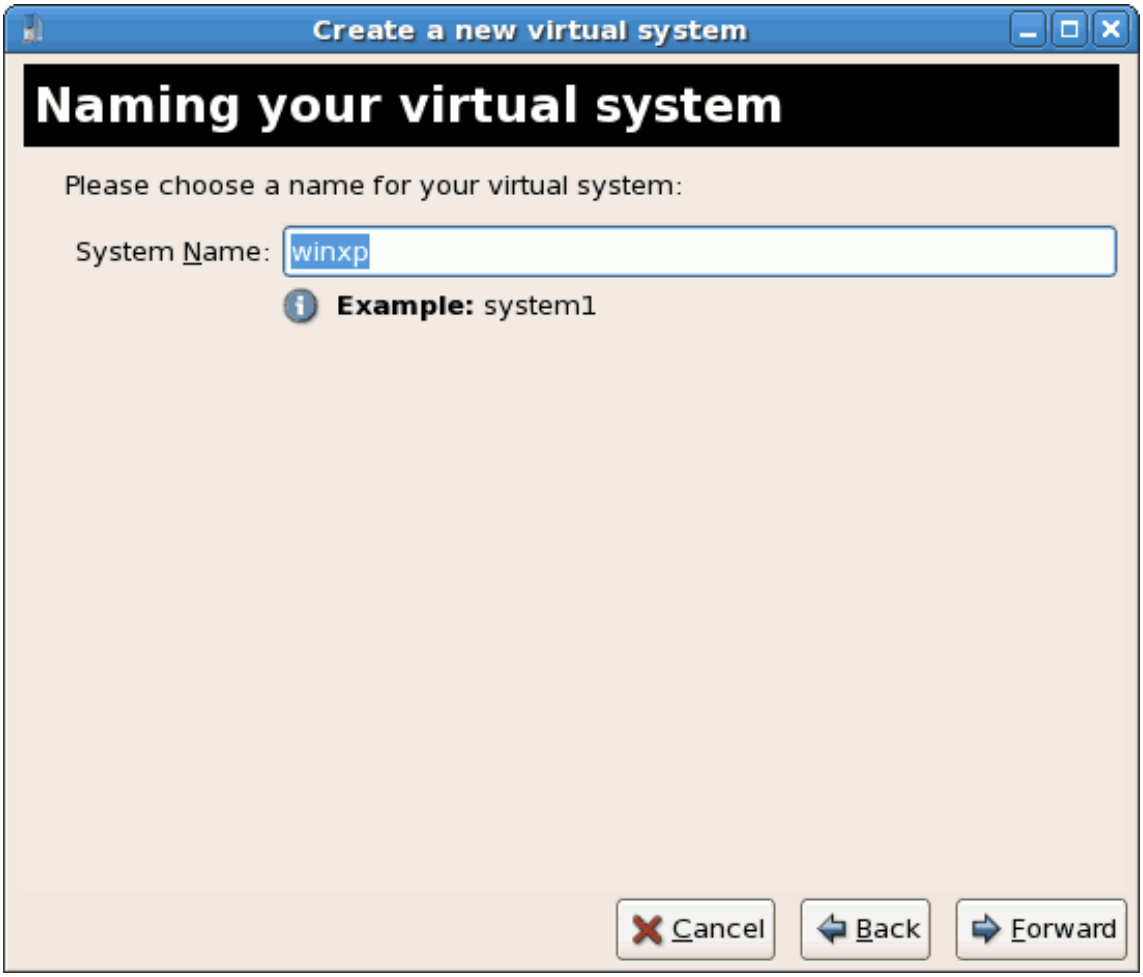
この工程を始める前に root アクセスを確定する必要があります。

重要

現時点では、Itanium® アーキテクチャ 上の Red Hat Enterprise Linux ホストは完全仮想化の Windows XP ゲストをサポートして いません。Itanium システムには、Itanium ベースシステム対応の Windows Server 2003 のみがサポートされています。

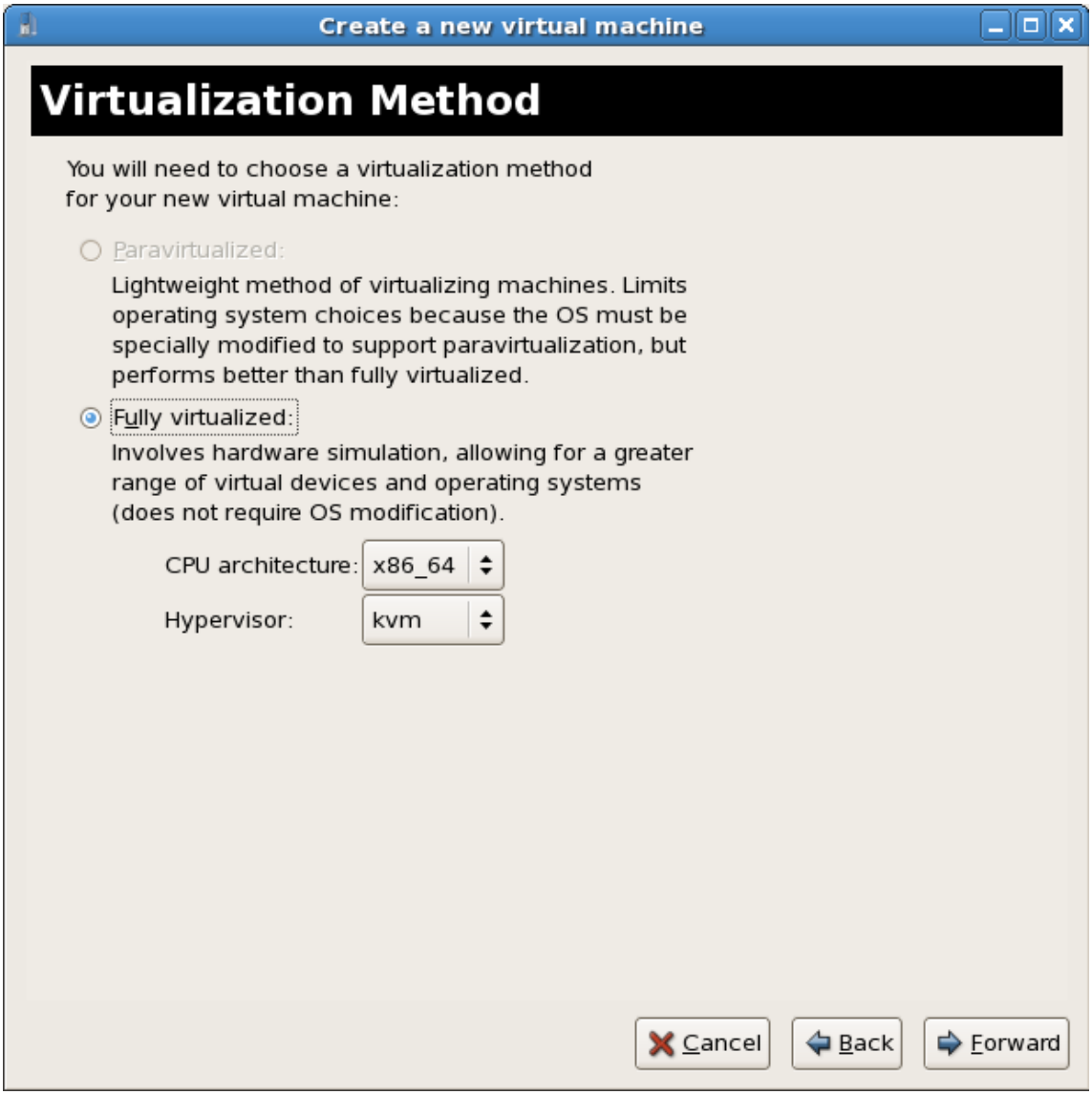
1. **virt-manager の開始**
   アプリケーション > システムツール > 仮想マシンマネージャ の順で開きます。ホストへの接続を開きます（**ファイル > 接続を開く** をクリック）。**新規** ボタンをクリックして 新規仮想マシンを作成します。

2. **仮想システムの命名**
   システムの名前 を入力して、それから **進む** ボタンを クリックします。



3. **仮想化メソッドの選択**
   先に KVM 又は Xen を選択（ステップ ステップ 1）している場合は、自分の選択した hypervisor を使用しなければなりません。 この例では、KVM hypervisor を使用します。

   Windows は完全仮想化を使用してのみインストールが可能です。



4. **インストールメソッドの選択**
この画面は、ユーザーがインストールのメソッドとオペレーティング システムのタイプを指定できるようにしてくれます。

CD-ROM 又は DVD でのインストールには、Windows インストールデスクが 入っているデバイスを選択します。**ISO イメージの場所** を選択する場合は、Windows インストールの .iso イメージへの パスを入力して下さい。

OS タイプ のリストから **Windows** を 選択して、**OS 変種** のリストから **Microsoft Windows XP** を 選択します。

PXE でのゲストのインストールは Red Hat Enterprise Linux 5.2 でサポートが あります。ただし PXE インストールはこの章では取り扱いません。

進む をクリックして継続します。

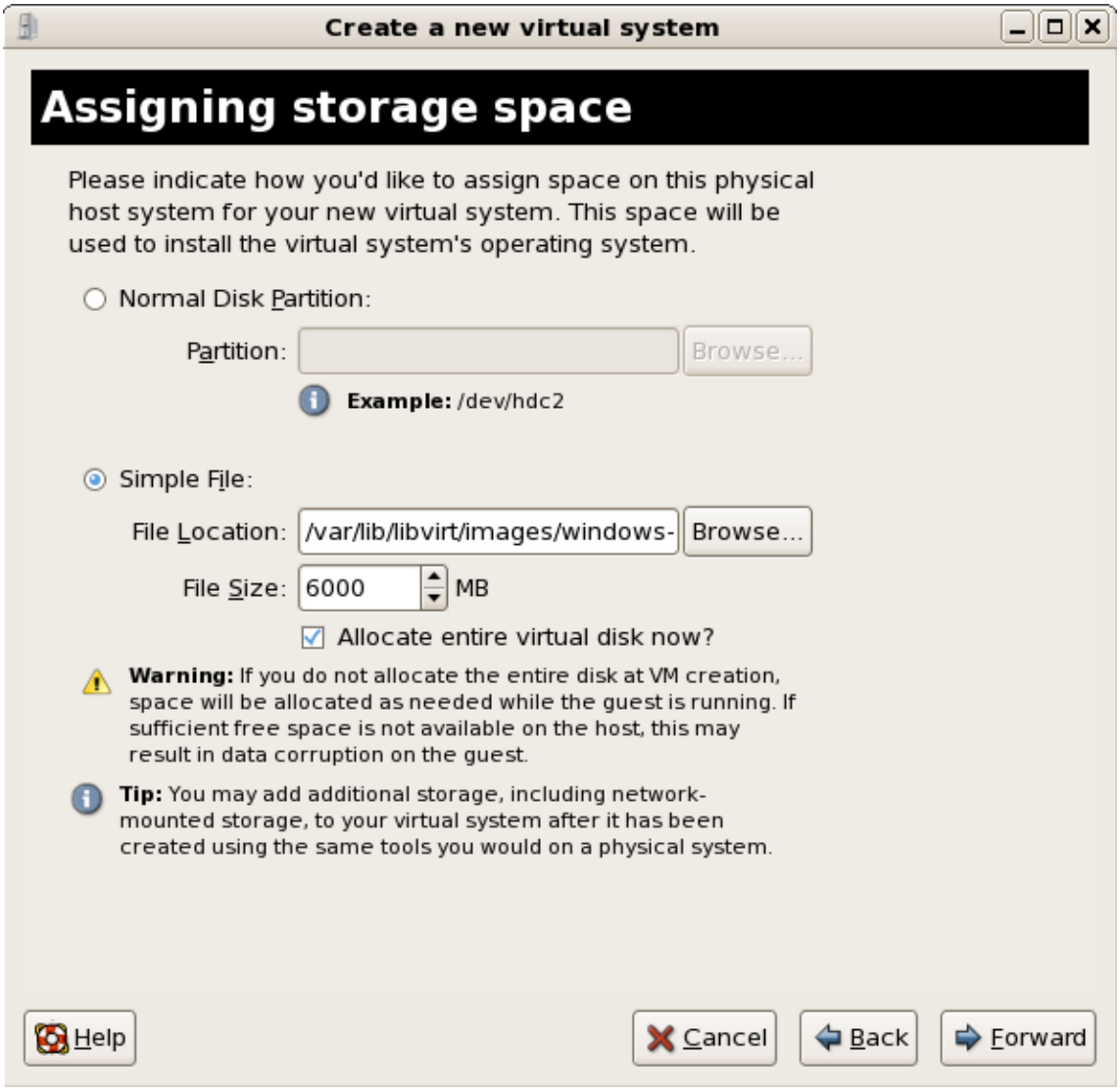
警告

ISO イメージファイルとストレージイメージには、**/var/lib/libvirt/images/** ディレクトリの使用が推奨されます。他の場所では、SELinux に追加の設定が必要になります。 詳細は、「SELinux と仮想化」 を参照して下さい。

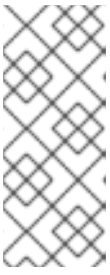
5. **ストレージスペースの割り当て** ウィンドウが表示されます。 ディスクパーティションか、LUN か、又は 「ゲストストレージ用のファイルベース イメージの作成」を選択します。

   全てのファイルベースゲストイメージは **/var/lib/libvirt/images/** ディレクトリに格納されるべきです。ファイルベースイメージ用には他のディレクトリは SELinux で禁止されています。SELinux を enforcing モードで実行する場合は、ゲストのインストール についての詳細を 「SELinux と仮想化」 で参照して下さい。

アプリケーションや他のデータの為にゲストが追加のスペースを必要とする場合、 余分のスペースを割り当てます。例えば、ウェブサーバーはログファイル用に 追加のスペースを必要とします。



選択したストレージタイプでゲスト用に適切なサイズを選びます。それから **進む** ボタンをクリックします。



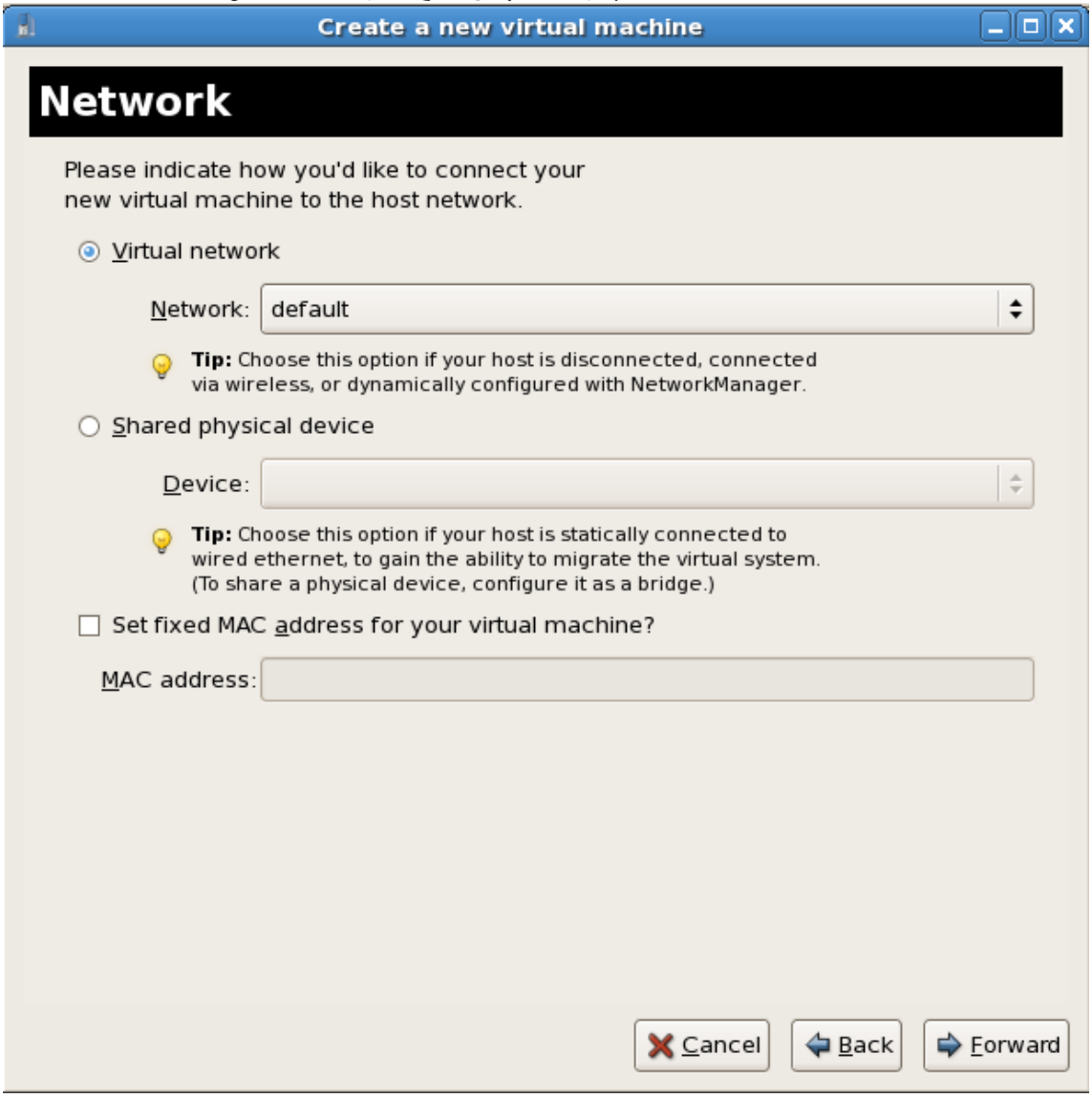
注記

仮想マシンにはデフォルトのディレクトリ、**/var/lib/libvirt/images/** の使用が推奨されます。別の場所（この例では、**/images/**）を使用している 場合は、インストールを継続する前に確実にそれが SELinux ポリシーに追加されるようにして下さい（このドキュメントの後で SELinux ポリシーの変更の仕方が案内してあります）。

6. **ネットワークのセットアップ**
   仮想ネットワーク 又は **共有物理デバイス**を選択します。

   仮想ネットワークオプションは NAT（Network Address Translation）を使用して 仮想化ゲストを持つデフォルトのネットワークデバイスを共有します。ワイヤレス ネットワークには仮想ネットワークオプションを使用して下さい。

共有物理デバイスオプションはネットワークボンドを使用して、仮想化ゲストが ネットワークデバイスに全面的なアクセスができるようにします。

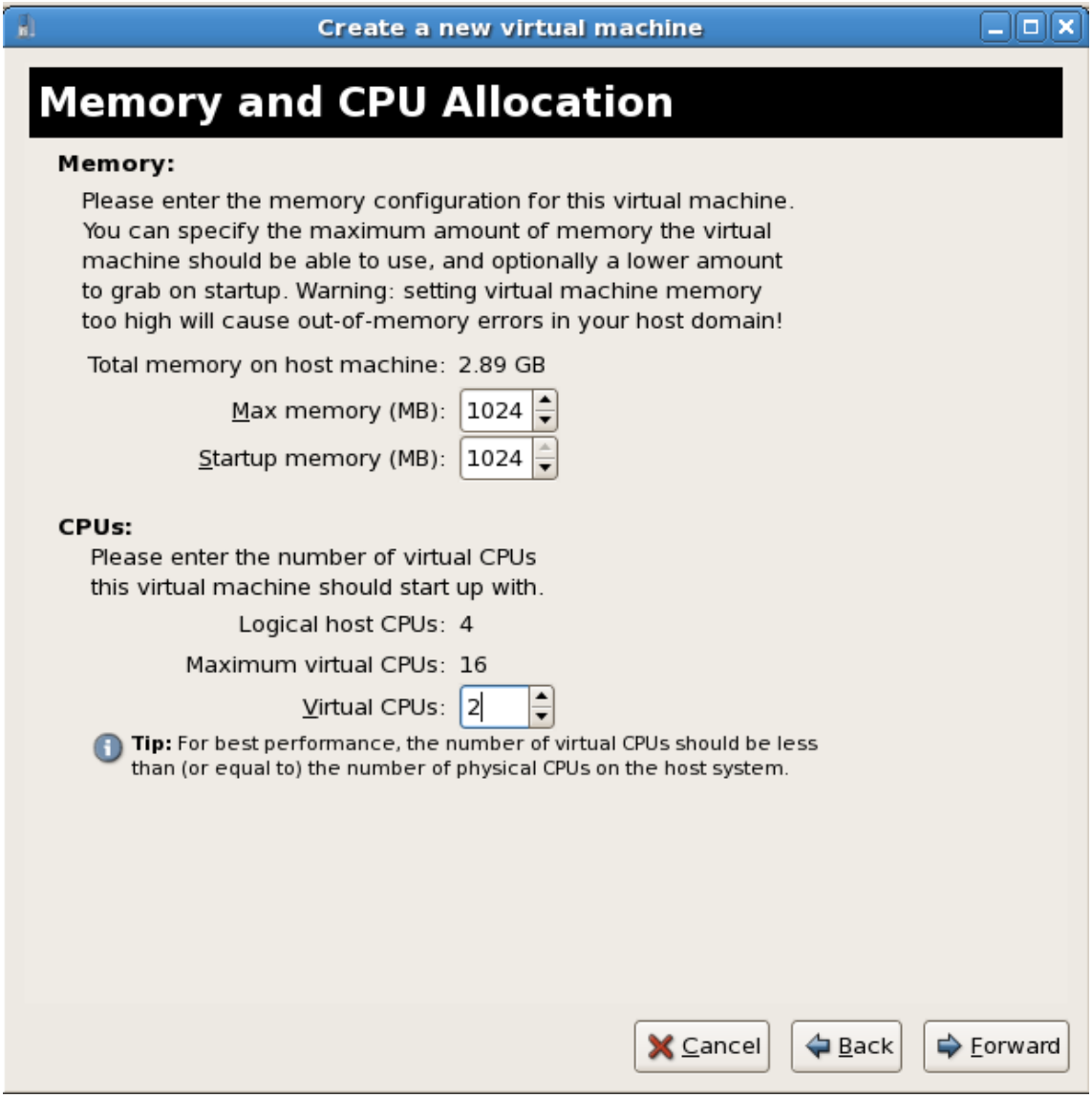


進む をクリックして継続します。

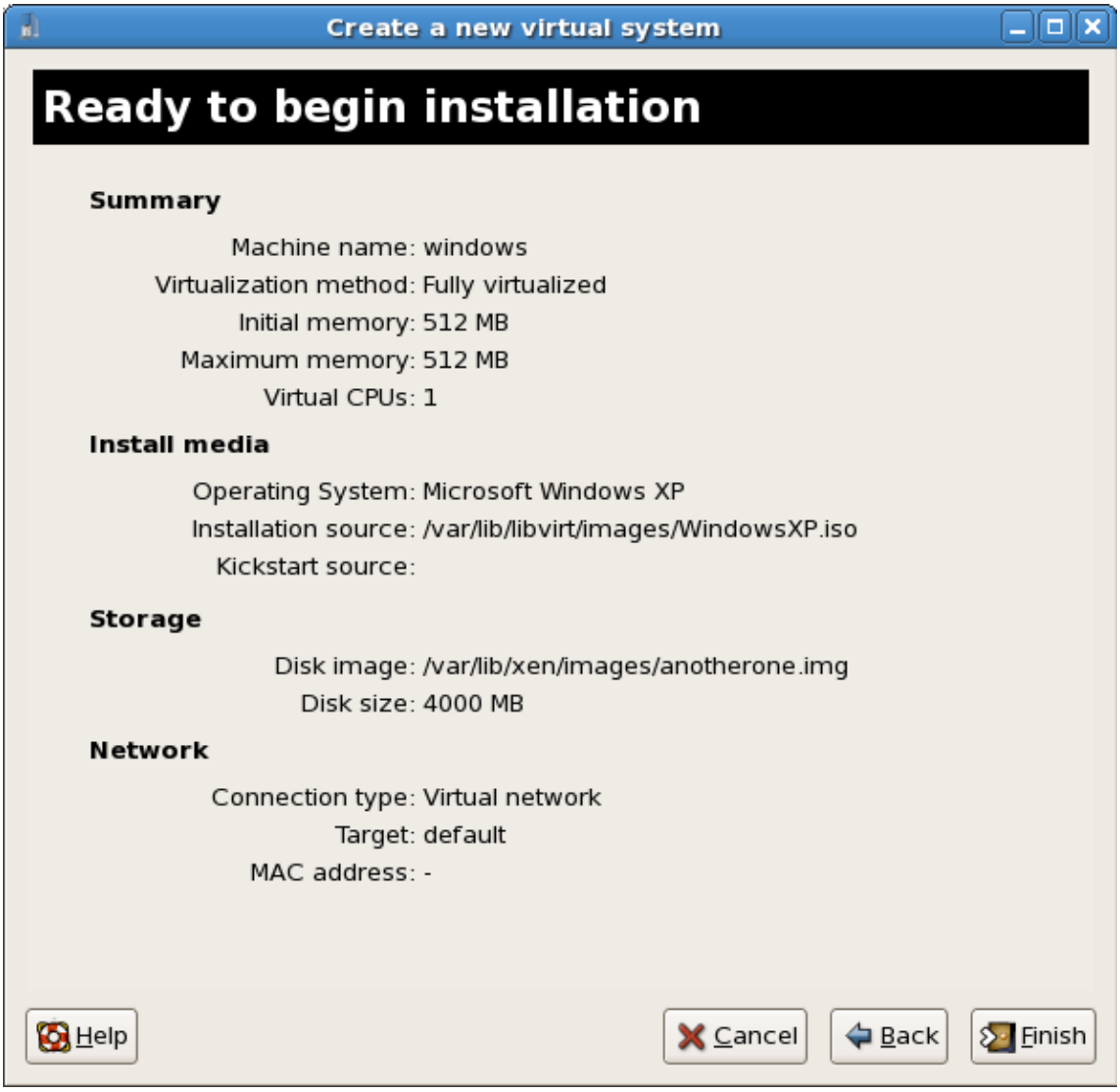
7. メモリーと CPU の割り当てウィンドウが表示されます。仮想化 CPU と RAM の 割り当てに適切な値を選択します。これらの値はホストとゲストのパフォーマンスに 影響します。

   仮想化ゲストは、効率的にそして効果的に稼働するために十分な物理メモリー (RAM) を必要とします。使用するゲストオペレーティングシステムとアプリケーションの 必要性に適合するメモリーの値を選択します。ほとんどのオペレーティングシステムは 正常に機能するために最低でも 512MB の RAM を必要とします。ゲストは物理 RAM を 使用することを忘れないで下さい。過度の数のゲストを稼働したり、ホストシステム用に不十分なメモリーを設定 していると、仮想メモリーとスワップをかなり使用することになります。仮想メモリーは 確実に低速であり、システムパフォーマンスと反応性の低下の原因となります。 全てのゲストとホストが効率的に稼働できるように十分なメモリーを割り当てる ことを確認して下さい。

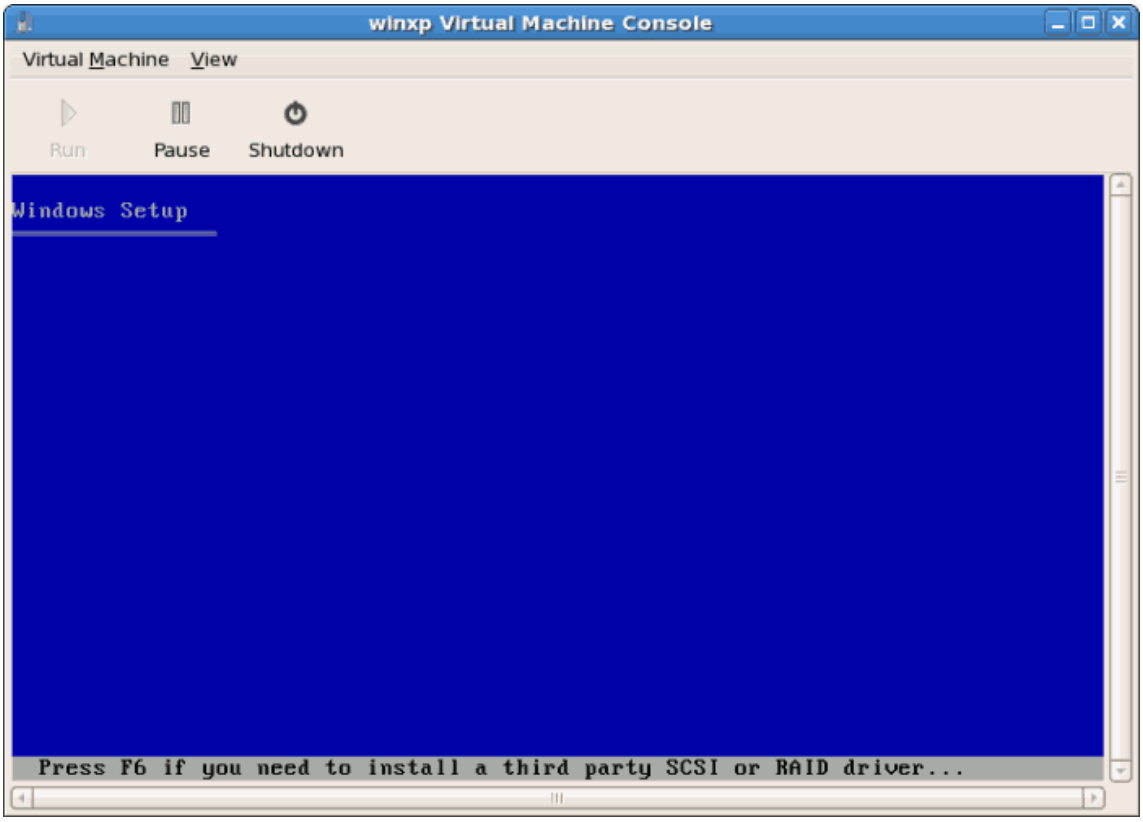
   十分な仮想 CPU を仮想ゲストに割り当てます。ゲストがマルチスレッドの アプリケーションを実行する場合は、最も効率良く実行するのにゲストが必要な仮想化 CPU の数を割り当てます。ホストシステム上で利用できる物理プロセッサ （又はハイパースレッド）の数量以上の仮想 CPU を割り当てないで下さい。 仮想プロセッサの超過割り当ては可能ですが、超過割り当ては、プロセッサの コンテキストがオーバーヘッドを切り替えるため、ゲストとホストのパフォーマンスに重大な悪影響を与えます。



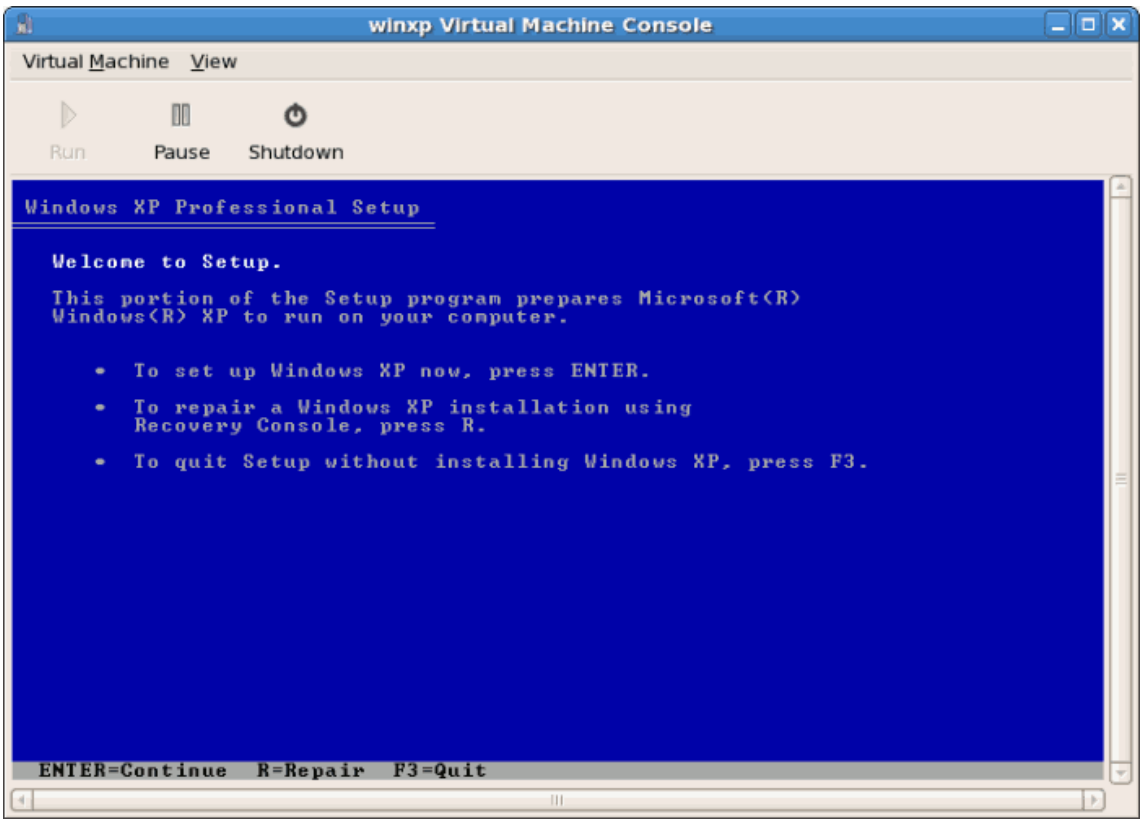
8. インストールが継続される前に、要約の画面が表示されます。**完了** をクリックしてゲストインストールへと進みます。

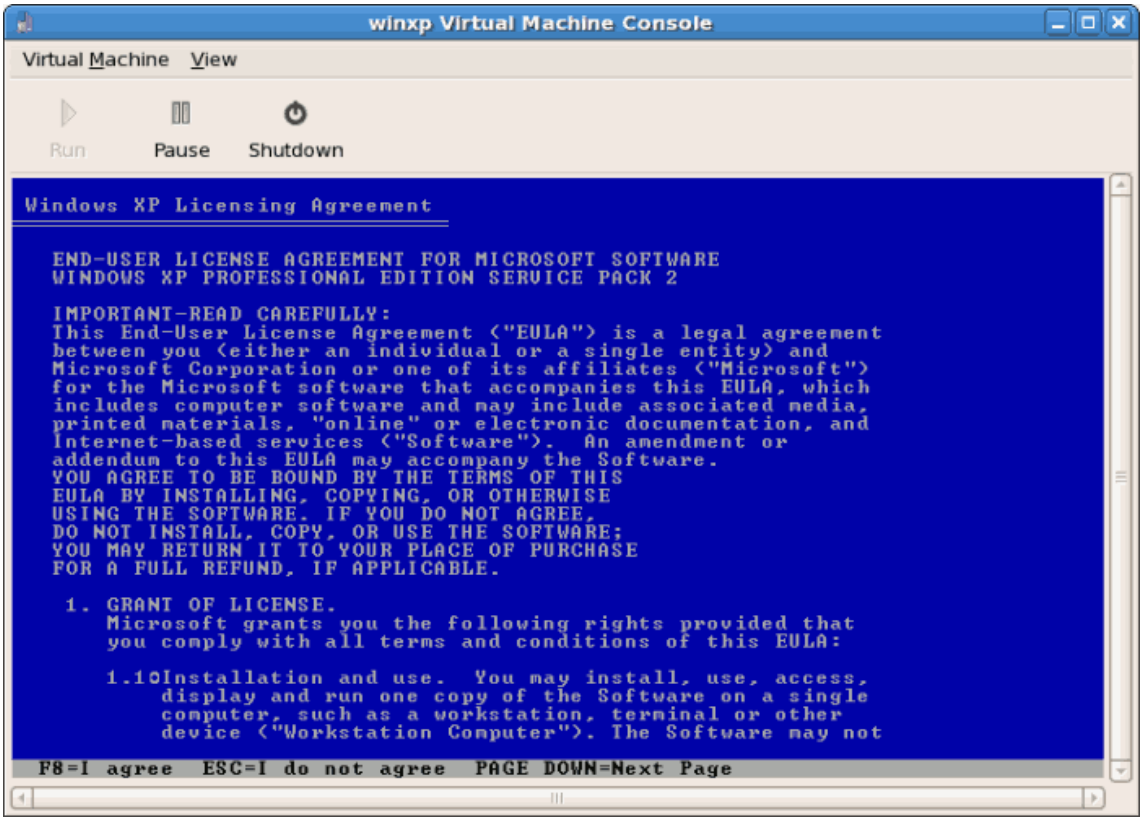


9. インストールがスタートした後に素早くコンソールウィンドウが開くように、ハードウェア選択をする必要があります。**完了** をクリックして、 **virt-manager** 要約ウィンドウに切り替えて、新しく開始した **Windows** ゲストを選択します。システム名をダブルクリックするとコンソールウィンドウが 開きます。素早く **F5** を数回押して、新規の **HAL** を 選択します。**Windows** インストールにダイアログボックスが出ると、**Generic i486 Platform** タブを選択します。**Up** と **Down** の矢印で選択肢を移動します。

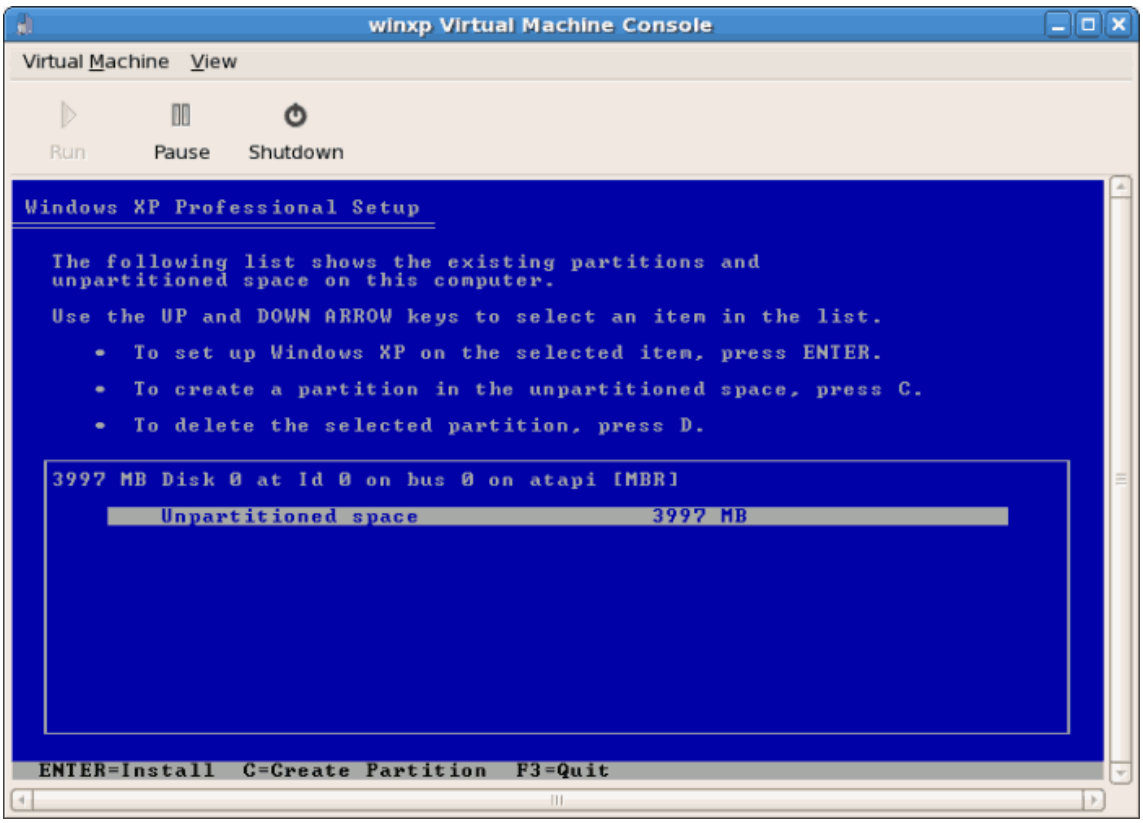


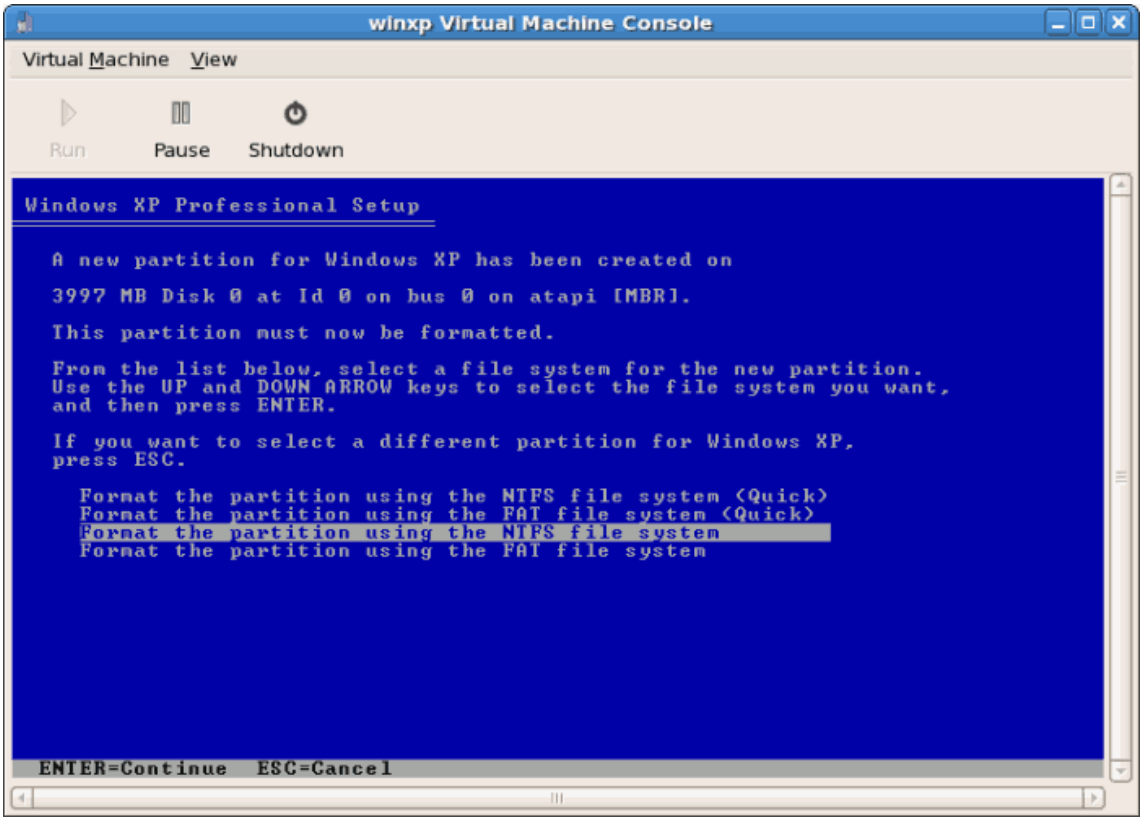
10. インストールは標準の Windows インストールと同様に進みます。



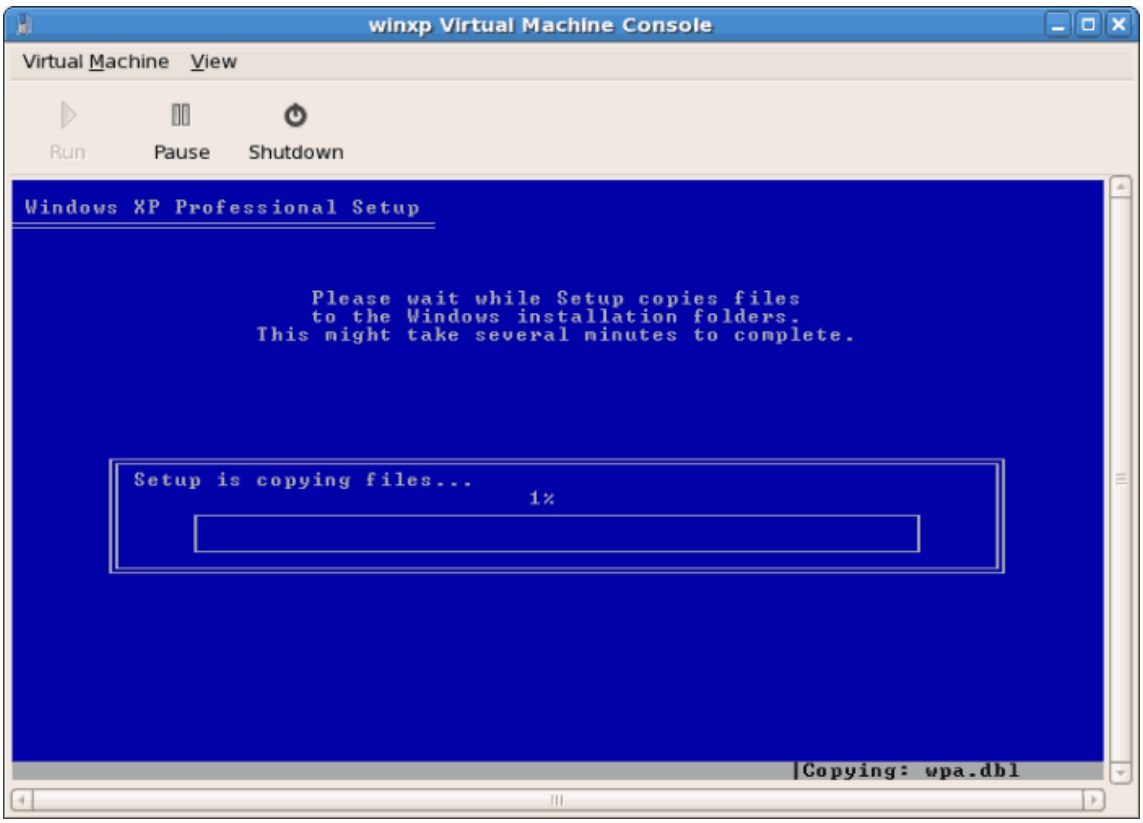


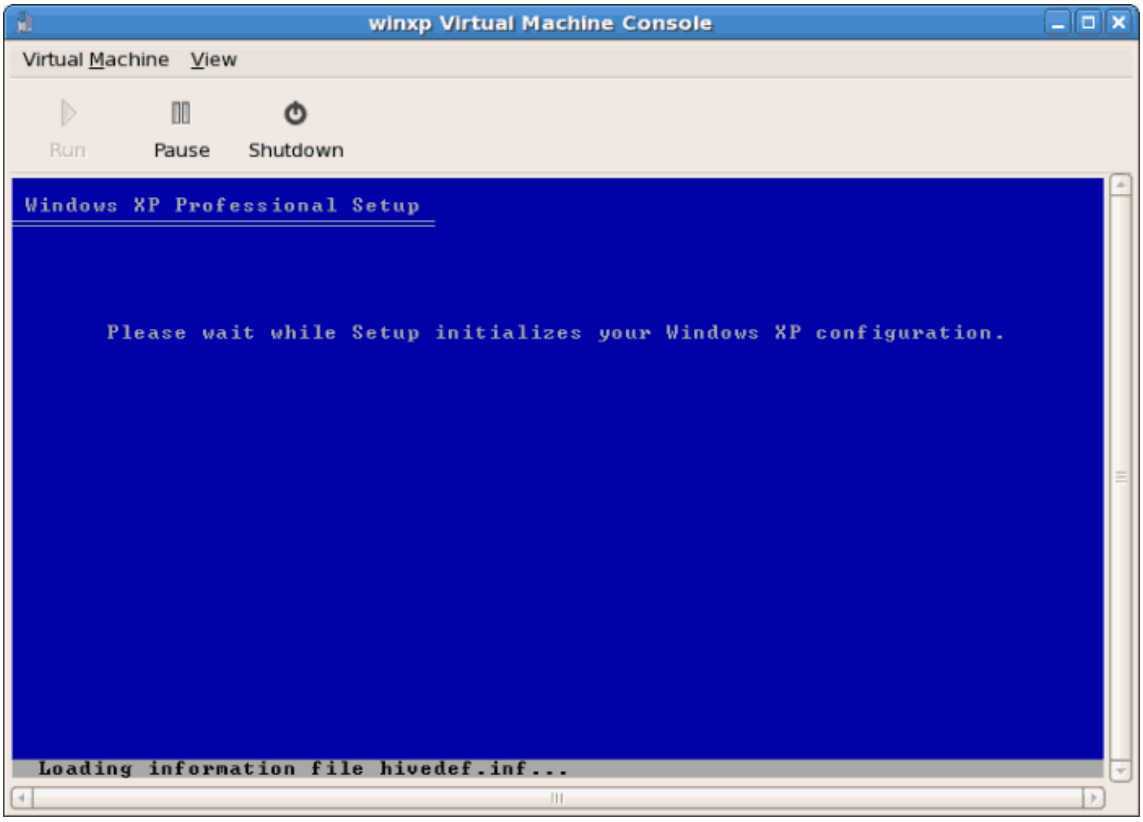
11. プロンプト時にハードドライブのパーティション設定



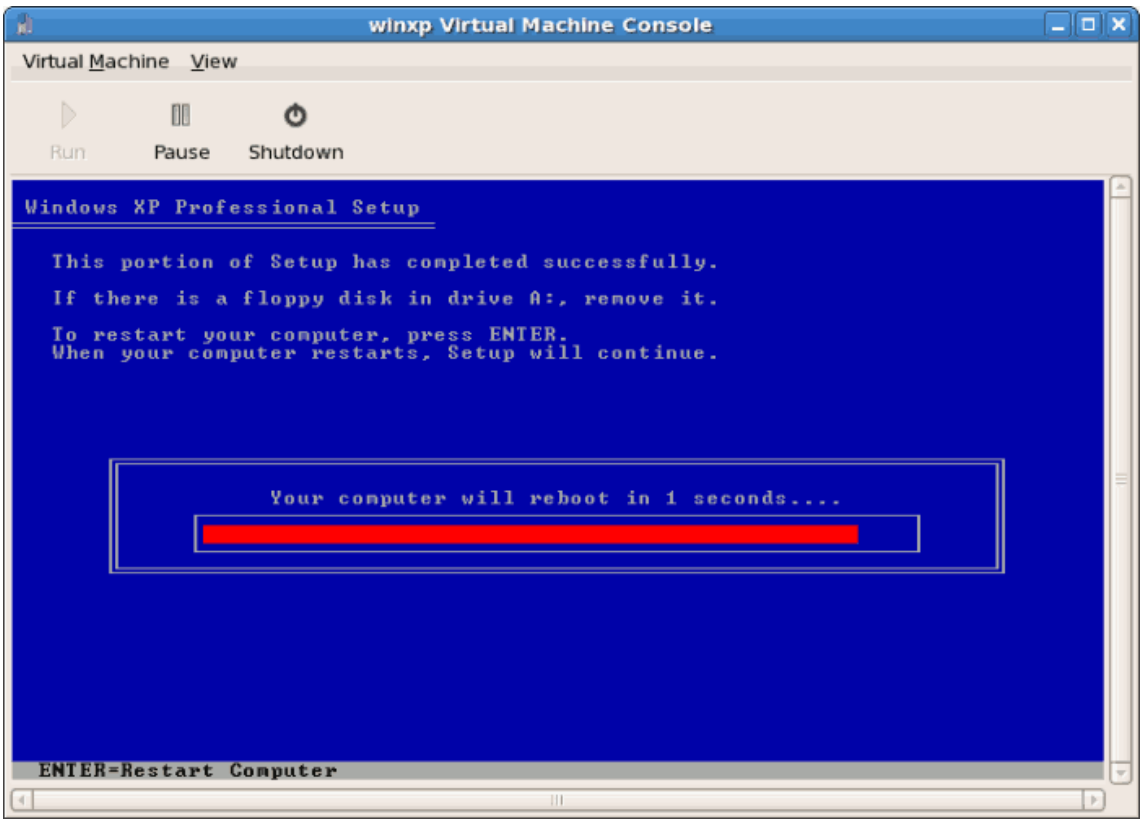


12. ドライブがフォーマットされた後に、Windows はハードドライブへファイルのコピーを開始します。





13. ファイルはストレージデバイスにコピーされて、Windows がここで再起動します。

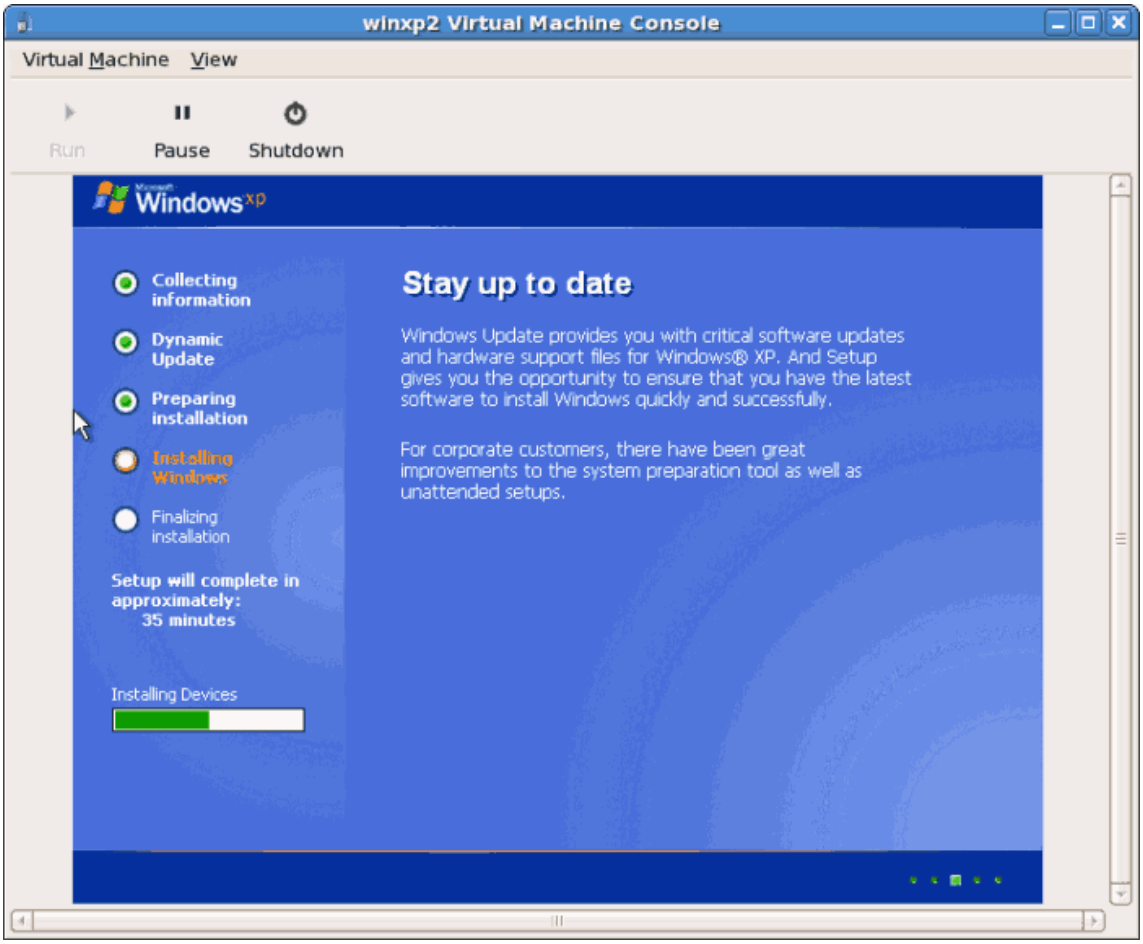


14. Windows ゲストを再起動:

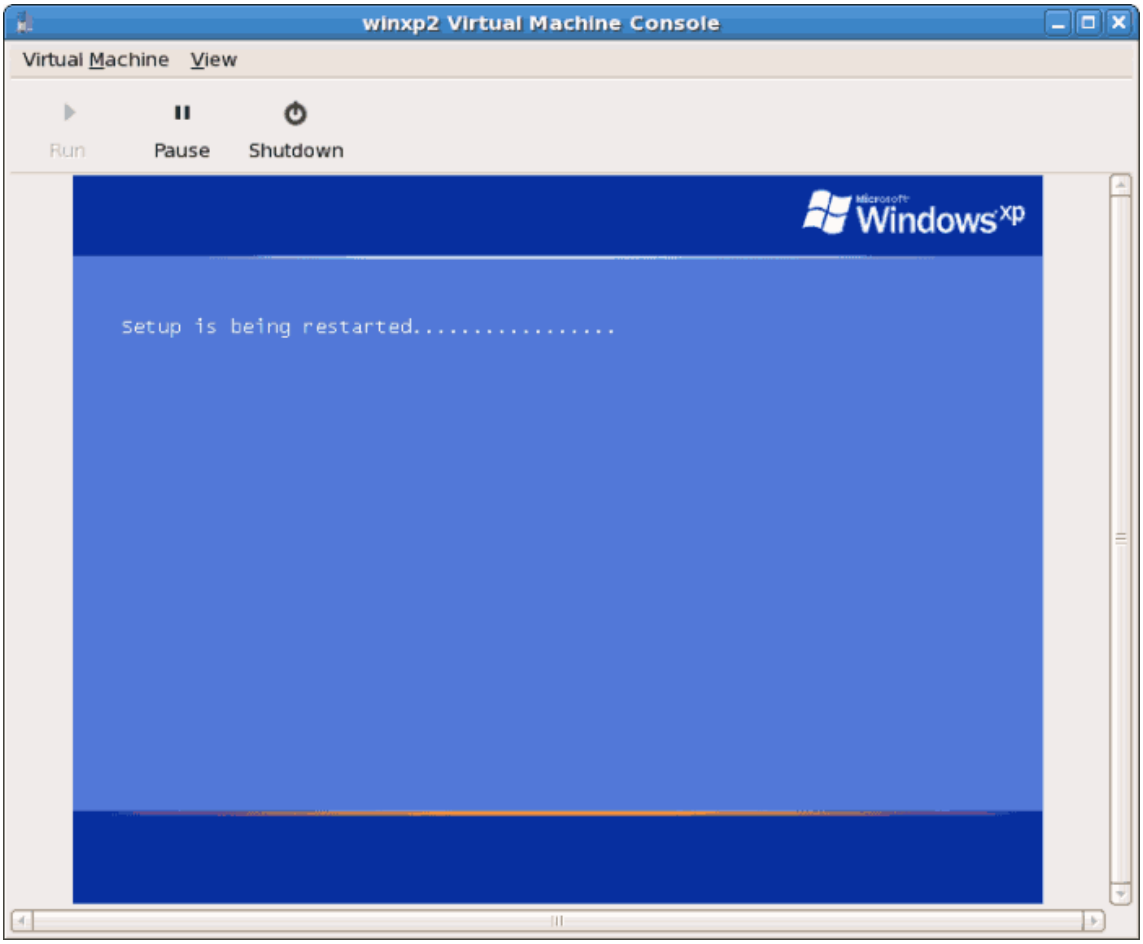
```
# virsh start WindowsGuest
```

ここで、*WindowsGuest* とは、ご使用の仮想マシンの 名前です。

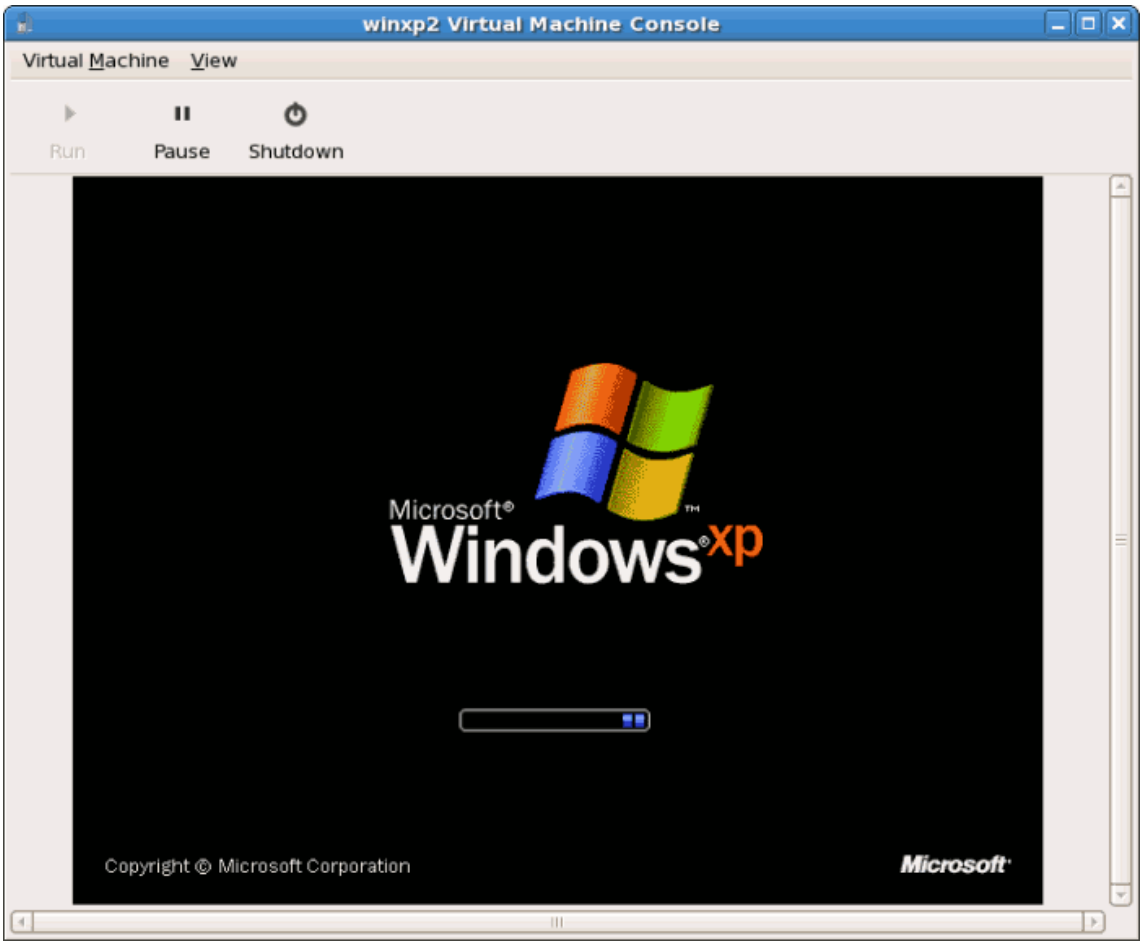
15. コンソールウィンドウが開くと、Windows インストールのセットアップ段階が出てきます。



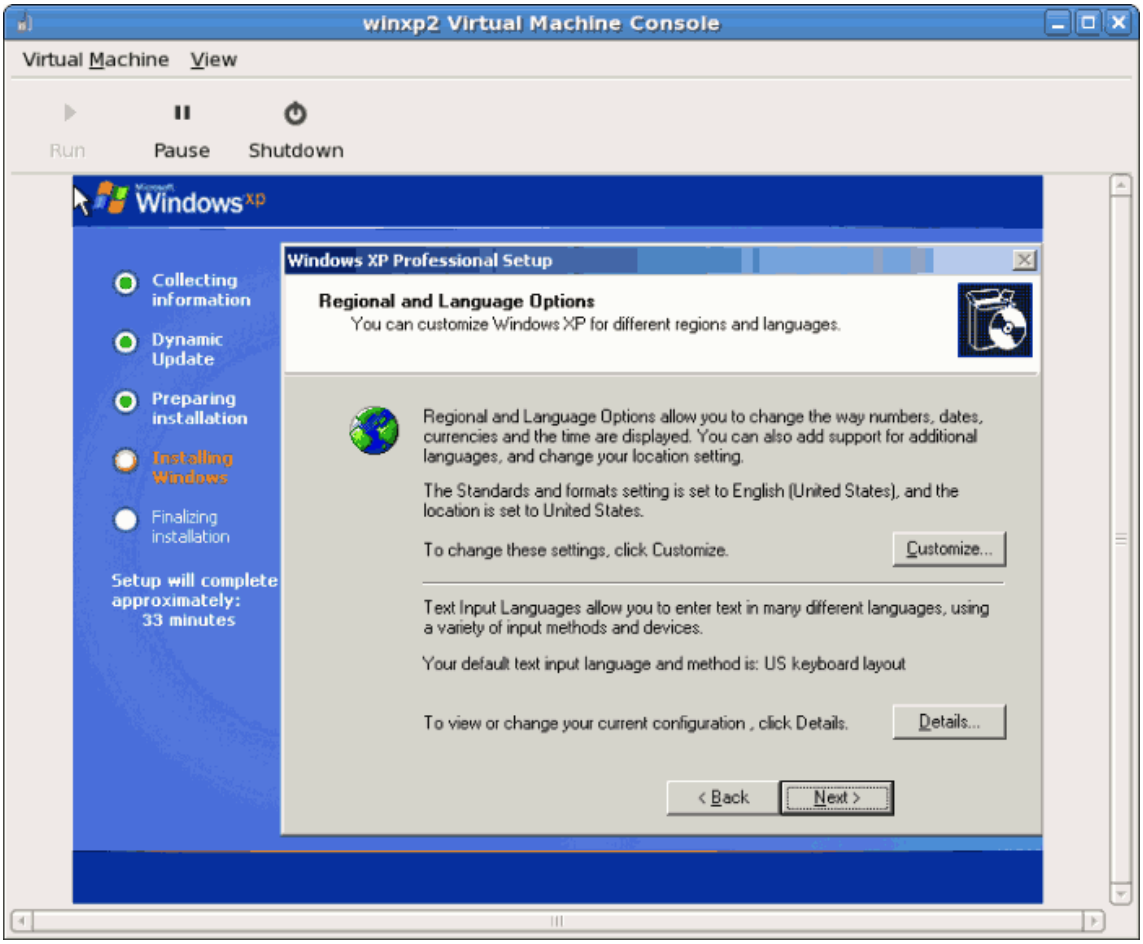
16. ユーザーのインストールがセットアップ段階で止まったように見える場合、 **`virsh reboot *WindowsGuestName*`** を使用してゲストを再起動して下さい。仮想マシンを再起動すると、 **`Setup is being restarted`** のメッセージが出てきます:



17. セットアップが終了したら、Windows のブート画面が出てきます:



18. この時点で Windows インストールの標準のセットアップを継続できます:



19. これでセットアッププロセスは完了です。

7.4. 完全仮想化ゲストとして WINDOWS SERVER 2003 をインストール

この章では、完全仮想化の Windows Server 2003 ゲストの **virt-install** コマンドを使用したインストールの説明をしています。**virt-install** は virt-manager の代用として使用できます。このプロセスは「Windows XP を完全仮想化ゲストとしてインストール」に案内してある Windows XP のインストールに似ています。

注記

現在、Itanium® アーキテクチャ上の Red Hat Enterprise Linux ホストは完全仮想化した Windows のゲストをサポートしていません。このセクションは x86 と x86-64 のホストにのみ適用されます。

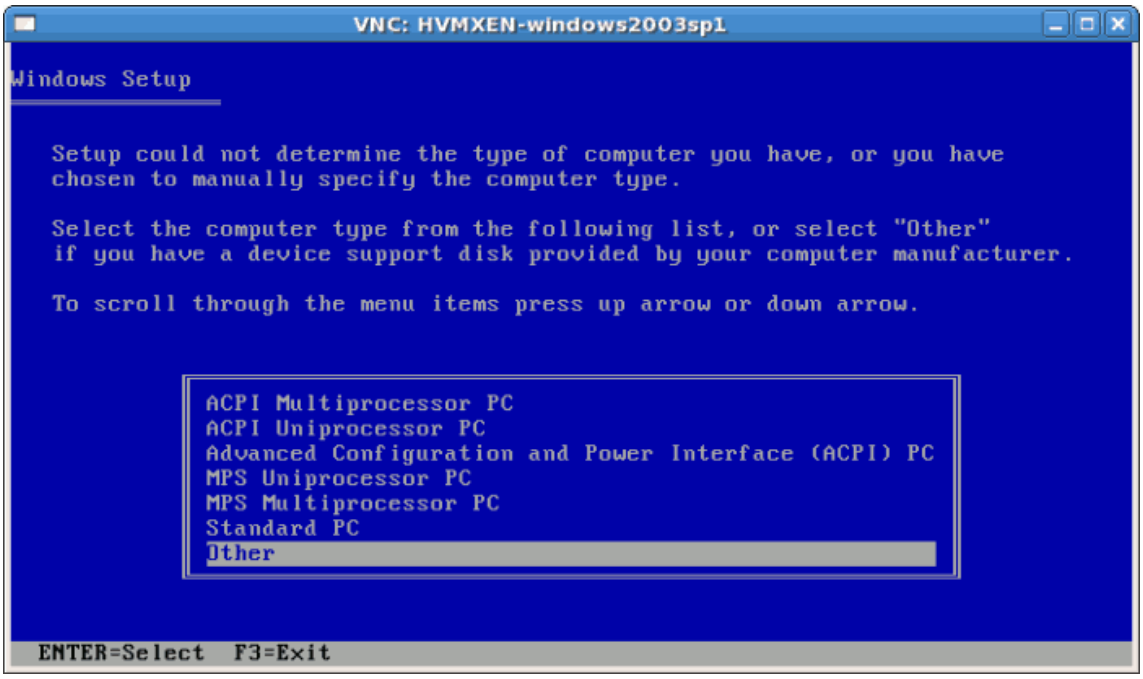
1. Windows ゲストのコンソールとして Windows Server 2003 のインストールに **virt-install** を使用すると、**virt-viewer** ウィンドウを開きます。以下に Windows Server 2003 ゲストのインストールの為の **virt-install** の使用サンプルを示します:

   virt-install コマンドでインストールを開始

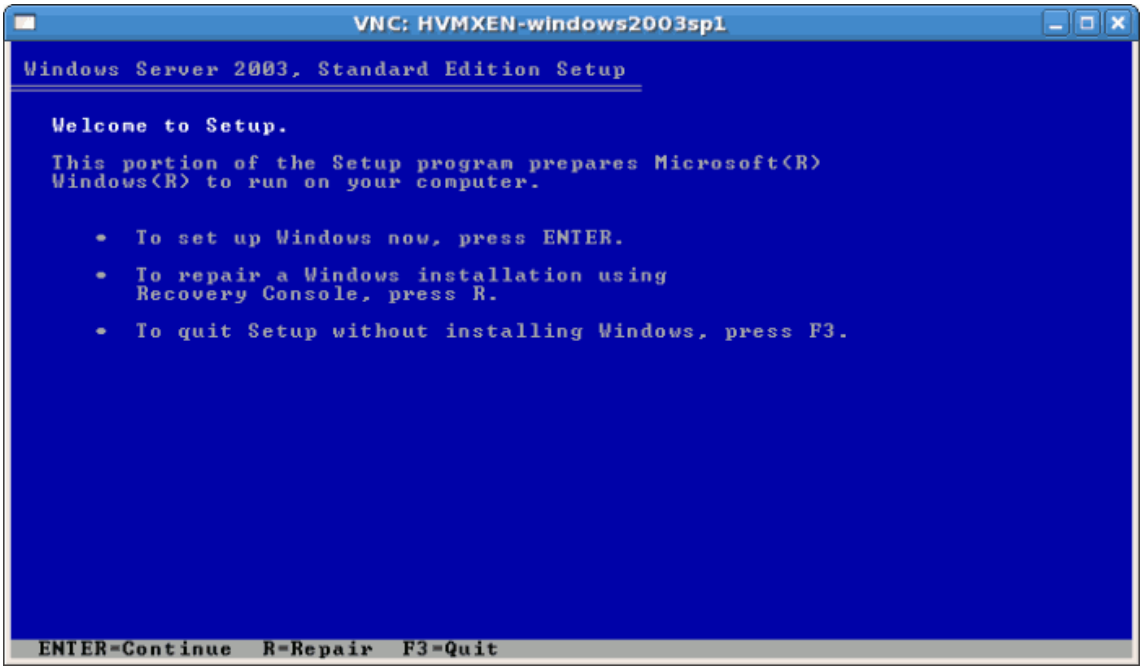
   ```
   # virt-install -hvm -s 5 -f
   /var/lib/libvirt/images/windows2003spi1.dsk \
   -n windows2003sp1 -cdrom=/ISOs/WIN/en_windows_server_2003_sp1.iso  \
   -vnc -r 1024
   ```

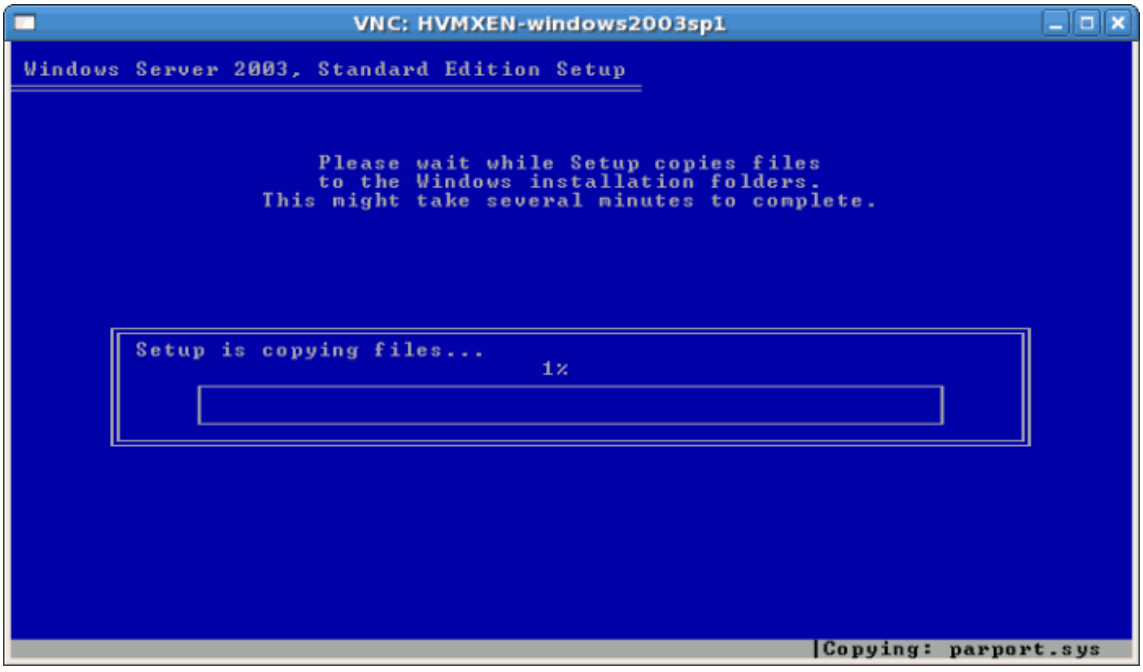
2. ゲストがインストール時点まで起動すると、すぐに **F5** を押す必要があります。適切なタイミングで **F5** を押さないと、インストールを再開始することになります。**F5** を押すことにより、

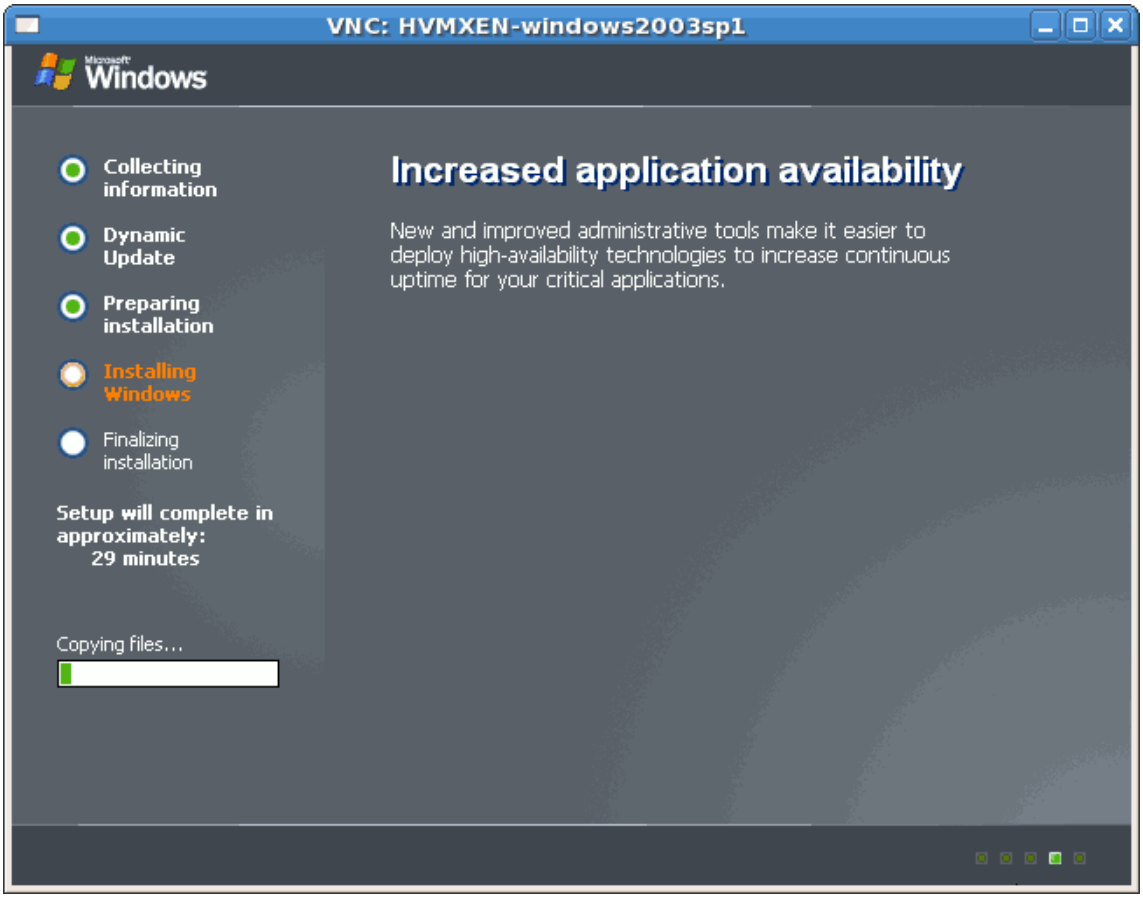
異なる **HAL** 又は、**Computer Type** を選択できるようになります。**Computer Type** として **Standard PC** を選択します。Windows Server 2003 の仮想化ゲストには **Computer Type** の変更が必要になります。



3. インストールの残りを完了する







4. これで Windows Server 2003 が完全仮想化ゲストとしてインストールされました。

7.5. WINDOWS SERVER 2008 を完全仮想化ゲストとしてインストール

このセクションでは、完全仮想化 Windows Server 2008 ゲストのインストールを説明します。KVM hypervisor の為には Red Hat Enterprise Linux 5.4 又は それ以降が必要となります。

手順7.4 virt-manager を使用した Windows Server 2008 のインストール

1. **virt-manager を開く**

virt-manager を開始します。**アプリケーション** メニュー内の **システムツール** サブメニューから **仮想マシン マネージャ** アプリケーションを起動します。別の方法としては、**root** として **virt-manager** コマンドを実行します。

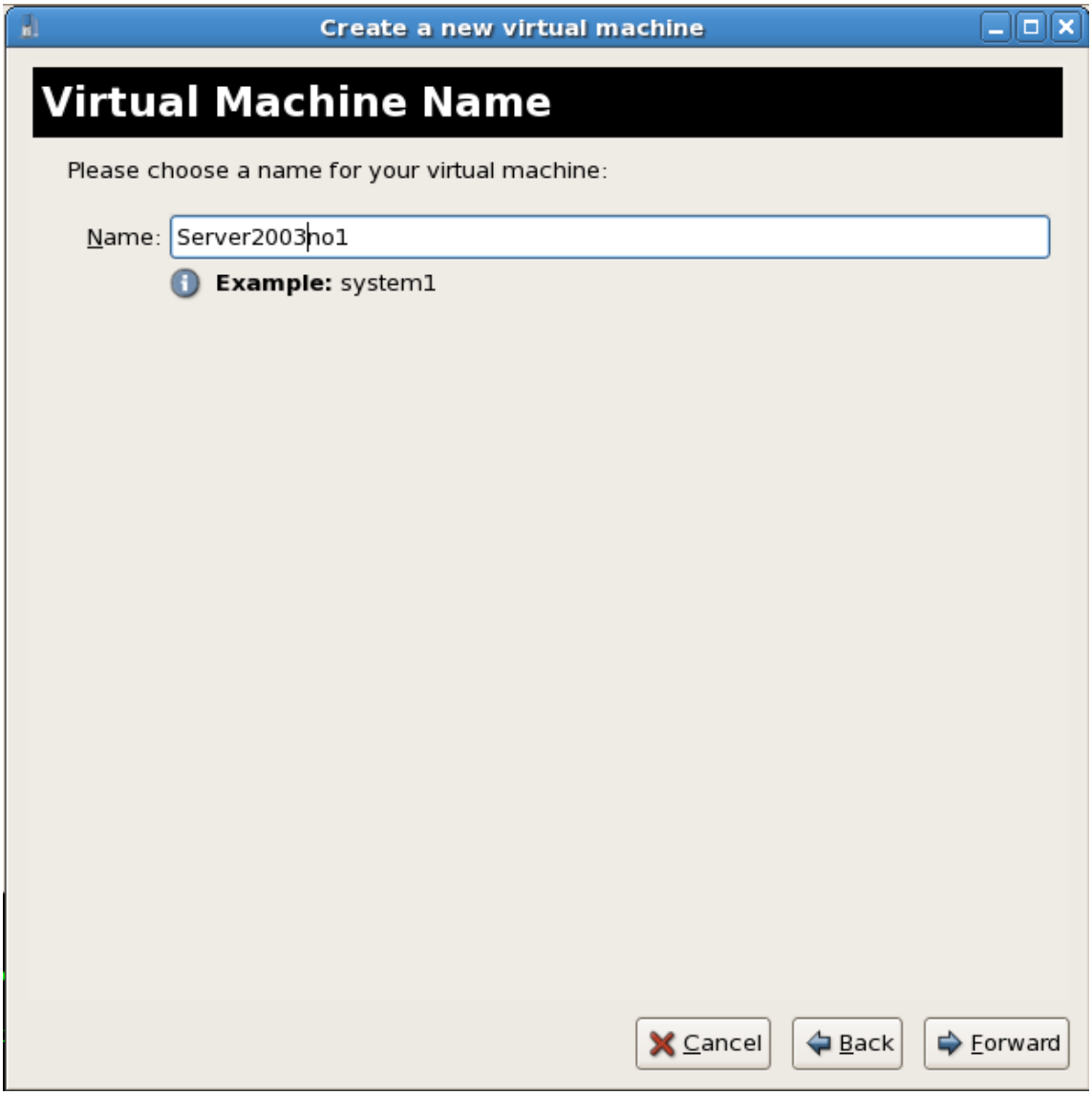
2. **Hypervisor を選択**
   Hypervisor を選択します。Xen か KVM がインストールされている場合は、そのいずれかを 選択します。ここでの例として KVM を選択しましょう。現在、KVM には **qemu** と言う呼称があることに注意して下さい。

   オプションが選択されると、**新規** ボタンが使用可能に なります。そこで**新規** ボタンを押します。

3. **新規仮想マシンウィザードを開始**
   新規 ボタンを押して、仮想マシン作成のウィザードを 開始します。

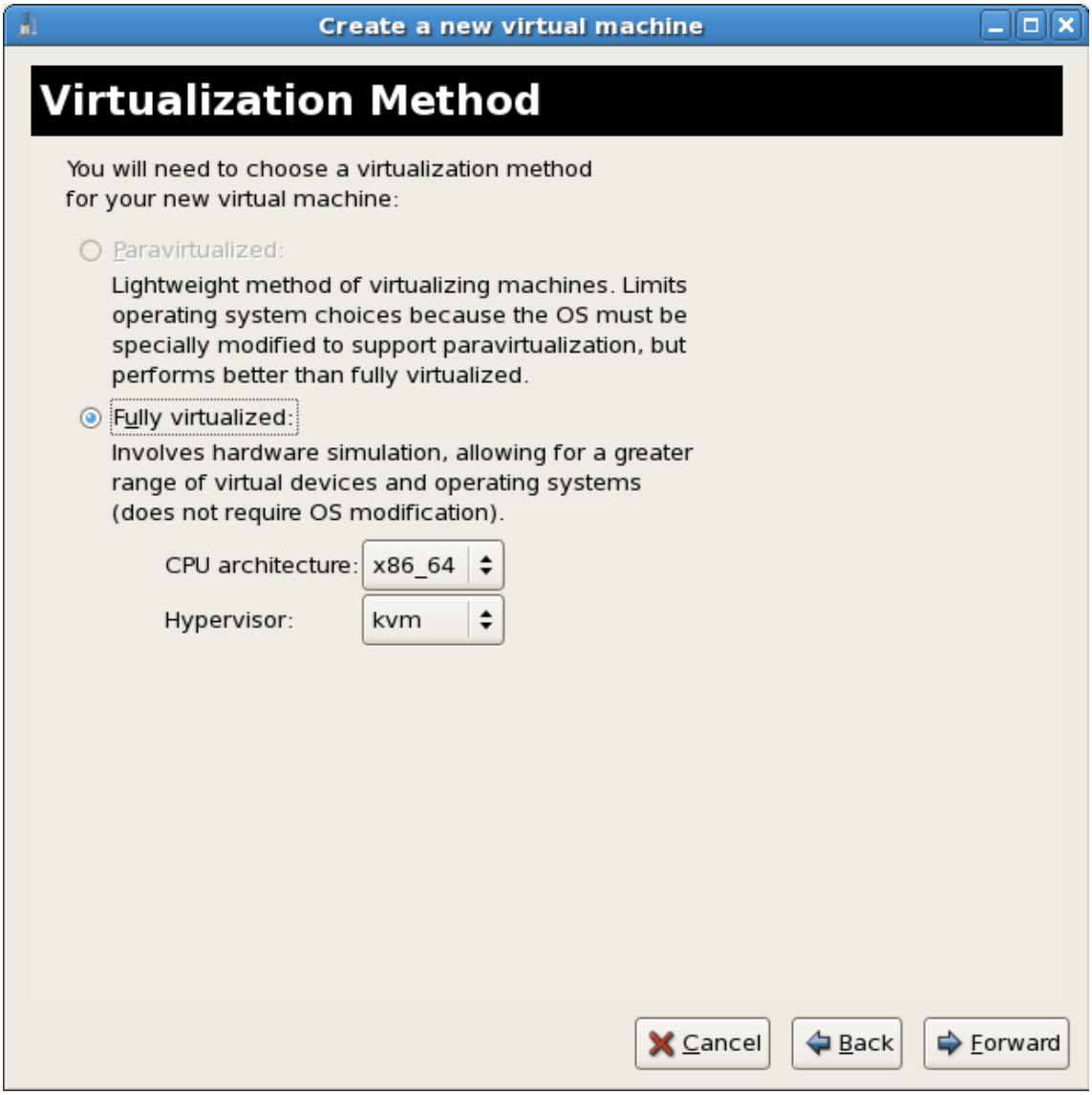
   進む を押して継続します。

4. **仮想マシンの命名**
   使用する仮想化ゲストに名前を付けます。句読点や空白は許可されません。



進む を押して、継続します。

5. **仮想化メソッドを選択**
   仮想化ゲスト用の仮想化メソッドを選択します。インストール済の仮想化メソッドのみが 選択できることに注意して下さい。先の手順（手順２）で KVM か Xen を選択した場合は、 自分が選択している hypervisor を使用しなければなりません。ここの例では KVM hypervisor を 使用しています。



進む を押して、継続します。

6. **インストールメソッドを選択**
   Windows の全てのバージョンには、ISO イメージか、又は物理光学メディアの いずれかの **ローカルインストールメディア** を使用する必要があります。

   Windows のネットワークインストールの為に PXE サーバーが設定されて いる場合は、PXE を使用できます。しかし、このガイドでは、PXE Windows インストールは説明していません。

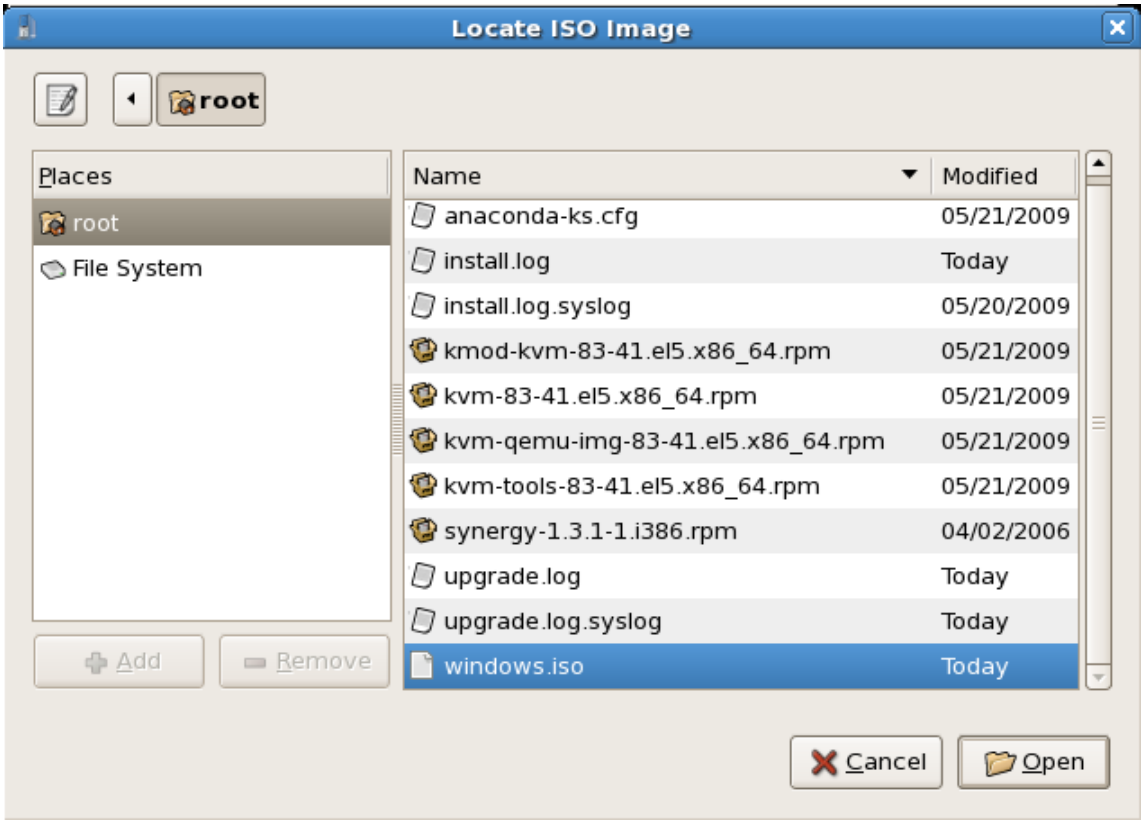
   スクリーンショットに示してあるように、**OS タイプ** を **Windows** にセットして、 **OS 変種** を **Microsoft Windows 2008** にセットします。

進む を押して継続します。

7. **インストールメディアの位置を指定**
   ISO イメージの位置、又は CD-ROM か DVD のデバイスを選択します。この例では、 Windows Server 2008 のインストール CD 用の ISO ファイルイメージを使用します。

   a. **閲覧** ボタンを押します。

   b. ISO ファイルの位置を見つけて、それを選択します。



開く を押して選択を確定します。

c. ファイルが選択されて、そこからのインストール準備ができました。

進む を押して継続します。

警告

ISO イメージファイルとゲストストレージイメージには、**/var/lib/libvirt/images/** ディレクトリの使用が推奨されます。他の場所では、SELinux への追加設定が必要となります。 詳細は「SELinux と仮想化」 を参照して下さい。

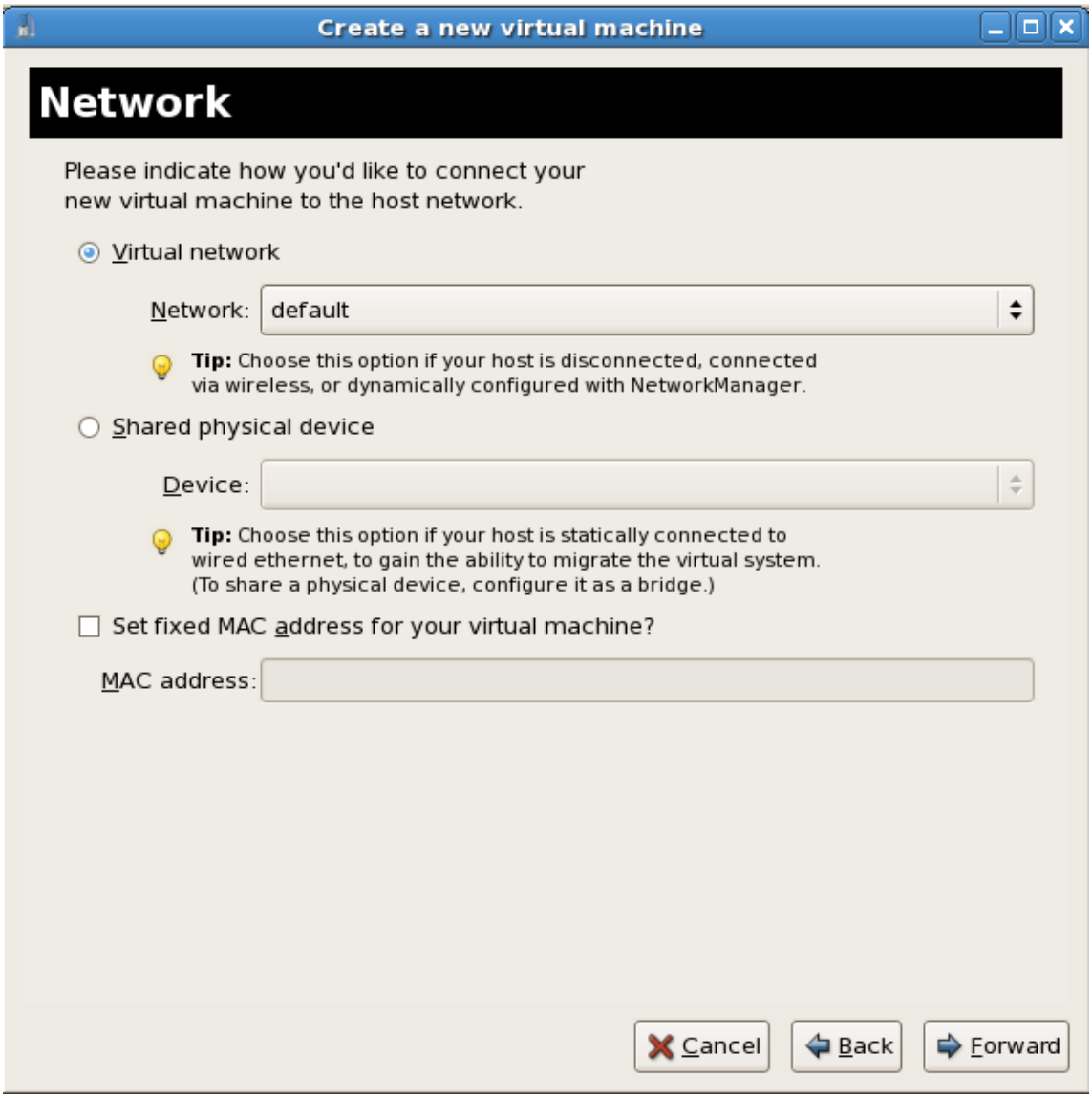
8. **ストレージセットアップ**
   物理ストレージデバイス **(ブロックデバイス)**、又は ファイルベースイメージ **(ファイル)** を割り当てます。 ファイルベースイメージは **/var/lib/libvirt/images/** ディレクトリに格納しなければなりません。 仮想化ゲストとそれが必要とするアプリケーションに十分な スペースを割り当てます。

進む を押して継続します。

9. **ネットワークセットアップ**
   仮想ネットワーク 又は **共有物理デバイス** のいずれかを選択します。

   仮想ネットワークオプションは **NAT (Network Address Translation)** を使用して、 デフォルトのネットワークデバイスを仮想ゲストと共有します。仮想ネットワーク オプションをワイヤレスネットワークに使用して下さい。

   共有の物理デバイスオプションはネットワークボンドを使用して、仮想ゲストに ネットワークデバイスへの全面的アクセスを与えます。

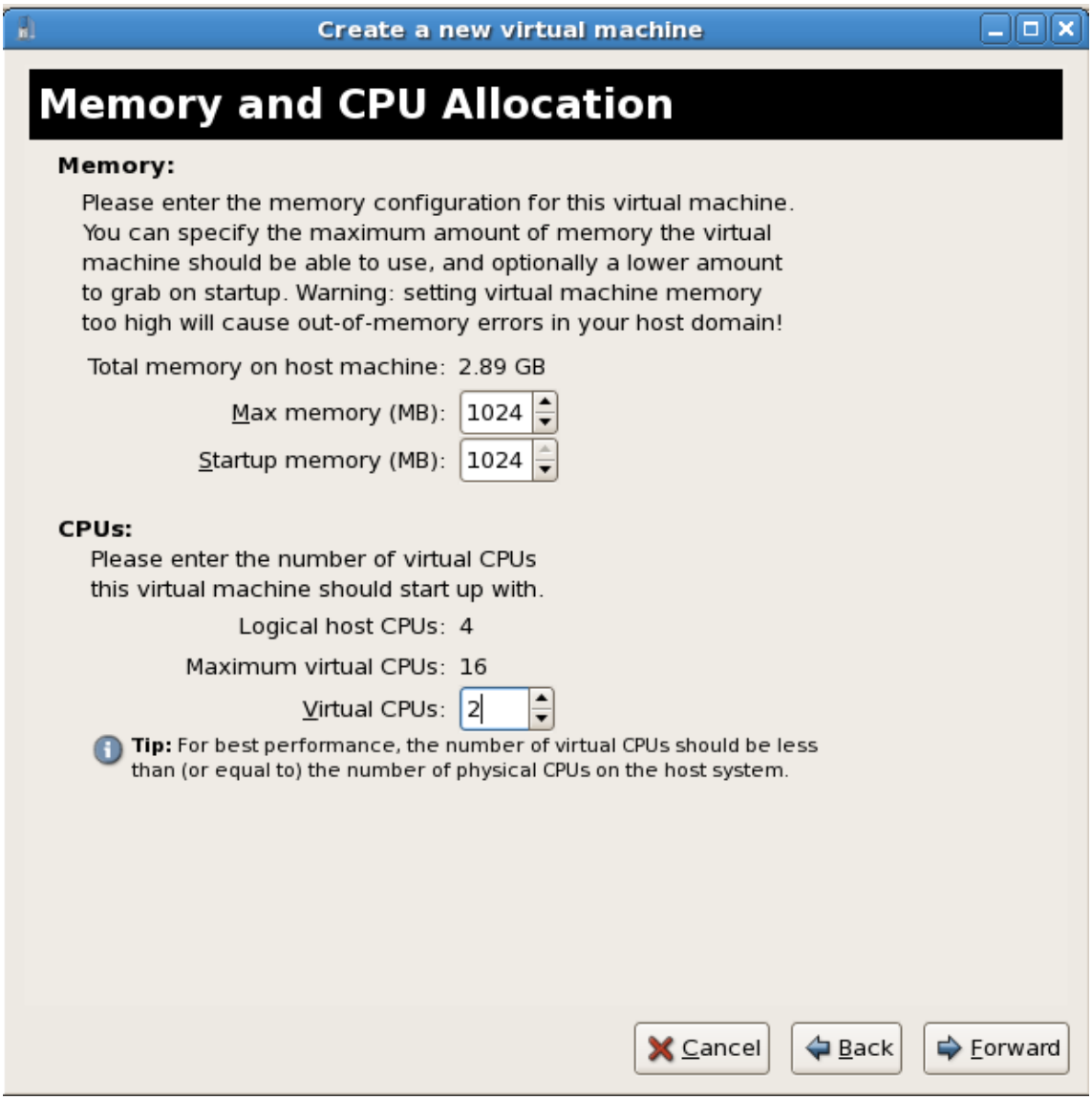


進む を押して継続します。

10. **メモリーと CPU の割り当て**
メモリーと CPU の割り当てウィンドウが表示されます。仮想化 CPU と 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とゲストのパフォーマンスに 影響します。

仮想化ゲストは、効率的にそして効果的に稼働するために十分な物理メモリー (RAM) を必要とします。使用するゲストオペレーティングシステムとアプリケーションの必要性に適合するメモリーの値を選択します。ゲストは物理 RAM を使用することを忘れないで下さい。過度の数のゲストを稼働したり、ホストシステム用に不十分なメモリーを設定していると、仮想メモリーとスワップをかなり消費することになります。仮想メモリーは 確実に低速であり、システムパフォーマンスと反応性の低下の原因となります。 全てのゲストとホストが効率的に稼働できるように十分なメモリーを割り当てることを確認して下さい。

十分な仮想 CPU を仮想ゲストに割り当てます。ゲストがマルチスレッドの アプリケーションを実行する場合は、ゲストが効率良く実行するのに必要な仮想化 CPU の数を割り当てます。ホストシステム上で利用できる物理プロセッサ （又はハイパースレッド）の数量以上の仮想 CPU を割り当てないで下さい。 仮想プロセッサの超過割り当ては可能ですが、超過割り当ては、プロセッサのコンテキストがオーバーヘッドを切り替えるため、ゲストとホストのパフォーマンスに重大な悪影響を与えます。

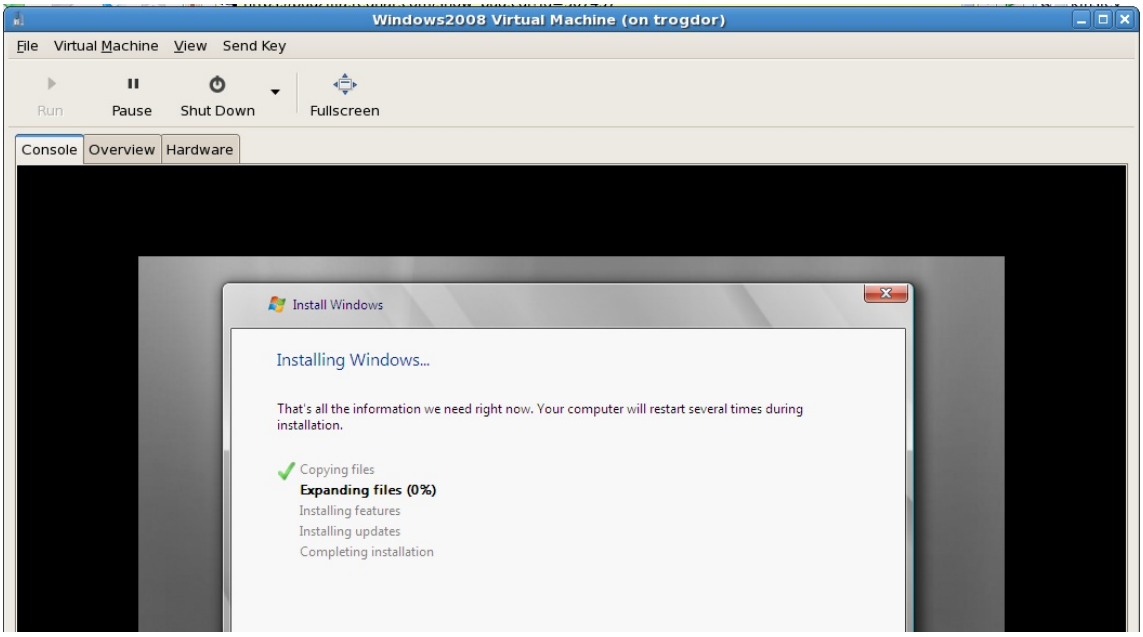


進む をクリックして継続します。

11. **ゲストのインストールを確認してからスタートします。**
設定を確認します。

完了 をクリックして ゲストインストールの工程を開始します。

12. **Windows のインストール**



Windows Server 2008 インストールのシーケンスを完了します。インストールの シーケンスはこのガイドでは説明しておりません。Windows のインストールに 関する情報には、Microsoft の ドキュメント を参照して下さい。

パート III. 設定

RED HAT ENTERPRISE LINUX 内で仮想化を設定

これらの章では、各種の高度な仮想化タスクの為の設定手順を言及しています。 これらのタスクにはネットワークとストレージの追加、セキュリティの強化、 パフォーマンスの向上、及び完全仮想化ゲスト上での para-virtualized ドライバーの 使用法が含まれます。

第8章 仮想ブロックデバイス

この章では、仮想化ゲストに於けるブロックデバイスのインストールと設定を扱います。ブロックデバイスと言う用語はストレージデバイスの各種形態を示します。

8.1. 仮想化フロッピィディスクコントローラの作成

フロッピィディスクコントローラは、多くの旧来のオペレーティングシステムで必要になります。特にドライバーのインストールで必要です。現時点では、物理的な フロッピィディスクデバイスは仮想ゲストからアクセスできません。しかし、フロッピィ ディスクイメージの作成と仮想化フロッピィドライブからのアクセスはサポートされて います。このセクションでは、仮想化フロッピィデバイスを取り扱います。

フロッピィディスクのイメージファイルが必要です。フロッピィディスクイメージファイルは **dd** コマンドを使用して作成します。 **/dev/fd0** の部分は使用する フロッピィデバイスの名前とディスクの名前で入れ替えます。

```
# dd if=/dev/fd0 of=~/legacydrivers.img
```

注記

para-virtualized ドライバーは物理フロッピィデバイスを完全仮想化ゲストにマップすることが できます。para-virtualized ドライバーの使用法に関する詳細は 14章 *Xen Para-virtualized ドライバー* でお読み下さい。

この例は、**/var/lib/libvirt/images/rhel5FV.img** にイメージを 配置した完全仮想化 Red Hat Enterprise Linux のインストールを実行している **virt-manager** で 作成されたゲストを使用します。Xen hypervisor がこの例で使用されています。

1. 実行中のゲストで **virsh** を使用して ゲストイメージ用に XML 設定ファイルを作成します。

   ```
   # virsh dumpxml rhel5FV > rhel5FV.xml
   ```

   これが、セッティング構成を XML ファイルとして保存して、このファイルの編集により ゲストで使用される操作とデバイスをカスタマイズ出来ます。virsh XML 設定ファイルの 使用法に関する詳細には、29章 *カスタムの libvirt スクリプト作成* を参照して下さい。

2. ゲスト用にフロッピィディスクイメージを作成

   ```
   # dd if=/dev/zero of=/var/lib/libvirt/images/rhel5FV-floppy.img
   bs=512 count=2880
   ```

3. 以下のコンテンツを追加します。必要な箇所では使用するゲストの設定 XML ファイルに変更します。この例は、ファイルベースイメージを使用したフロッピィデバイスの 模倣です。

   ```
   <disk type='file' device='floppy'>
           <source file='/var/lib/libvirt/images/rhel5FV-floppy.img'/>
           <target dev='fda'/>
   </disk>
   ```

4. ゲストを停止します。

```
# virsh stop rhel5FV
```

5. XML 設定ファイルを使用してゲストを再スタートします。

```
# virsh create rhel5FV.xml
```

フロッピィデバイスはこの時点で、ゲスト内で利用可能となりホスト上に イメージファイルとして保存されます。

8.2. ストレージデバイスをゲストに追加

このセクションでは、ストレージデバイスを仮想ゲストマシンに追加する方法を説明します。 追加のストレージはゲストが作成された後にのみ追加できます。サポートされているストレージデバイスとプロトコルを以下に示します:

- ローカルハードドライブのパーティション
- ローカルボリューム
- ファイバーチャンネル、又は ホストに直結の iSCSI
- ホストのファイルシステムに存在するファイルコンテナ
- 仮想マシンによって直接マウントされている **NFS** ファイルシステム
- ゲストにより直接アクセスされる iSCSI ストレージ
- クラスタファイルシステム **(GFS).**

ファイルベースストレージをゲストに追加

ファイルベースストレージ、又はファイルベースコンテナはホストファイルシステム上の ファイルであり、仮想化ゲストの為の仮想化ハードドライブとして動作します。ファイル ベースコンテナを追加するには、以下の手順を実行します:

1. 空のコンテナファイルを作成するか、又は、既存のファイルコンテナ（ISO ファイルなど）を使用。

   a. **dd** コマンドを使用して Sparse ファイルを作成します。 Sparse ファイルは、データの整合性とパフォーマンス問題のために推奨できませんが、 Sparse ファイルは より速く作成できてテストに使用できます。但し実稼働環境では使用できません。

   ```
   # dd if=/dev/zero of=/var/lib/libvirt/images/FileName.img bs=1M
   seek=4096 count=0
   ```

   b. 非 **sparse** の 事前割り当て済みのファイルがファイルベースストレージイメージ用に推奨されます。非 **sparse** ファイルを作成して、以下を実行します:

   ```
   # dd if=/dev/zero of=/var/lib/libvirt/images/FileName.img bs=1M
   count=4096
   ```

   これらの両方のコマンドは、仮想化ゲスト用に追加のストレージとして使用される **400MB** の ファイルを作成します。

2. ゲスト用の設定をダンプします。この例では、ゲストは *Guest1* と 言う名前であり、ファイル

はユーザーのホームディレクトリに保存されます。

```
# virsh dumpxml Guest1 > ~/Guest1.xml
```

3. テキストエディタで設定ファイル (この例では、*Guest1.xml*) を開きます。 **<disk>** エレメントを見つけて下さい。 このエレメントはストレージデバイスを記述するものです。以下にディスクエレメントの 例を示します:

```
<disk type='file' device='disk'>
    <driver name='tap' type='aio'/>
    <source file='/var/lib/libvirt/images/Guest1.img'/>
    <target dev='xvda'/>
</disk>
```

4. **<disk>** エレメントの複製により、又は 新規の書き込みにより追加のストレージを追加します。仮想ブロックデバイス属性用のデバイス名を 指定することを確認して下さい。この属性はそれぞれのゲスト設定ファイルのために独自のもので なければなりません。以下の例では、**FileName.img** と呼ばれる 追加のファイルベースストレージコンテナを含んでいる設定ファイルセクションを示しています。

```
<disk type='file' device='disk'>
    <driver name='tap' type='aio'/>
    <source file='/var/lib/libvirt/images/Guest1.img'/>
    <target dev='xvda'/>
</disk>
<disk type='file' device='disk'>
    <driver name='tap' type='aio'/>
    <source file='/var/lib/libvirt/images/FileName.img'/>
    <target dev='hda'/>
</disk>
```

5. 更新された設定ファイルからゲストを再起動します。

```
# virsh create Guest1.xml
```

6. 以下のステップは Linux ゲスト特有のものです。他のオペレーティングシステムは 別の方法で新規のストレージデバイスを処理します。Linux 以外のシステムではその オペレーティングシステムのドキュメントを参照して下さい。

   ゲストはこの時点で、ファイル **FileName.img** を **/dev/hdb** と言う名のデバイスとして使用します。 このデバイスはゲストからのフォーマットを必要とします。ゲスト上では、 パーティション設定で全デバイスを 1つのプライマリパーティションに して、それからそのデバイスをフォーマットします。

   a. 新規パーティション 用に ***n*** を押します。

   ```
   # fdisk /dev/hdb
   Command (m for help):
   ```

   b. プライマリパーティション用に ***p*** を押します。

```
Command action
   e   extended
   p   primary partition (1-4)
```

c. 利用可能なパーティションを１つ選択します。この例では、***1*** を 入力することにより、最初のパーティションが選択されます。

```
Partition number (1-4): 1
```

d. ***Enter*** を押すことでデフォルトの最初の シリンダを入力します。

```
First cylinder (1-400, default 1):
```

e. パーティションのサイズを選択します。この例では ***Enter*** を 押すことによりディスク全体が割り当てられます。

```
Last cylinder or +size or +sizeM or +sizeK (2-400, default 400):
```

f. ***t*** を押すことにより パーティションのタイプをセットします。

```
Command (m for help): t
```

g. 先のステップで作成したパーティションを選択します。この例では、 パーティション番号は ***1*** です。

```
Partition number (1-4): 1
```

h. linux パーティションとして ***83*** を入力します。

```
Hex code (type L to list codes): 83
```

i. 変更をディスクに書き込んで退出します。

```
Command (m for help): w
Command (m for help): q
```

j. **ext3** ファイルシステムを 使用して新規パーティションをフォーマットします。

```
# mke2fs -j /dev/hdb
```

7. ゲストにディスクをマウントします。

```
# mount /dev/hdb1 /myfiles
```

これでゲストは追加の仮想化ファイルベースストレージデバイスを持つことになります。

ゲストへハードドライブと他のブロックデバイスを追加

システム管理者は追加のハードドライブを使用して、より多くのストレージ領域を提供したり、 ユーザーデータからシステムデータを分離することができるようになります。この手順、手順8.1「物理ブロックデバイスを仮想化ゲストに追加」 はホスト上のハードドライブを 仮想化ゲストに追加する方法

を示しています。

この手順は、全ての物理ブロックデバイスに適用できます。それには、 CD-ROM、DVD 及び フロッピィディスクも含まれます。

手順8.1 物理ブロックデバイスを仮想化ゲストに追加

1. ハードディスクデバイスを物理的にホストに取り付けます。デフォルトでドライブが アクセス不可の場合は、ホストを設定します。
2. **multipath** の使用でデバイスを設定して、必要であれば ホスト上で永続化します。
3. **virsh attach** コマンドを使用します。*myguest* をご自分のゲスト名で、それから ***/dev/hdb1*** を追加するデバイスで、そして *hdc* をゲスト上のデバイスの場所でそれぞれ入れ替えます。*hdc* は未使用のデバイス名とする必要があります。そして、 Windows ゲストにも *hd** 表記を使用します。このゲストは デバイスを正しく認識するでしょう。

   --type hdd パラメータを CD-ROM 又は DVD デバイス用のコマンドに追記します。

   --type floppy パラメータを フロッピィデバイス用のコマンドに追記します。

   ```
   # virsh attach-disk myguest /dev/hdb1 hdc --driver tap --mode
   readonly
   ```

4. これで、ゲストは Linux 上に **/dev/hdb** と言う名の新規ハードディスクデバイスを持ち、また、Windows 上では **D: drive** かその類似名のハードディスクデバイスを持ちます。 このデバイスはフォーマットが必要かも知れません。

8.3. RED HAT ENTERPRISE LINUX 5 内に永続的ストレージを構成

このセクションは、ファイバーチャンネル、又は iSCSI ベースのストレージデバイスなどの 外部、あるいはネットワーク化したストレージを持つシステムについて説明しています。 これらのシステムでは永続的な名前が使用ホスト用に設定してあることが推奨されます。 これが移行を助けると共に、複数の仮想化システム用に不変のデバイス名とストレージを 提供することになります。

UUID (Universally Unique Identifiers) は分散型コンピューティング環境で コンピュータとデバイスの識別のための標準化した手法です。このセクションでは、 UUID を使用して iSCSI あるいはファイバーチャンネル LUN を識別します。UUID は 再始動、切断、そしてデバイススワップの後でも永続的に残ります。UUID はデバイス上の ラベルに似ています。

multipath を実行していないシステムは単独パスの設定 を使用しなければなりません。**multipath** を 実行しているシステムは マルチパスの設定を使用できます。

単独パスの設定

この手順は、**udev** を使用して *LUN* デバイスの永続化を実装します。この手順は、**multipath** を使用していないホストのためだけに使用して下さい。

1. **/etc/scsi_id.config** ファイルを編集します。
   a. **options=-b** の行がコメントアウトしてあることを確認します。

      ```
      # options=-b
      ```

   b. 以下の行を追加します:

```
options=-g
```

このオプションは、**udev** を設定して全ての付帯 SCSI デバイスが UUID を返すことを想定します。

2. 任意のデバイスの UUID を表示するには、 **scsi_id -g -s /block/sd*** コマンドを使用します。以下のようにします:

```
# scsi_id -g -s /block/sd*
3600a0b80001327510000015427b625e
```

実際の出力は上記の例と異なるでしょう。この出力は デバイス **/dev/sdc** の UUID を表示しています。

3. デバイスにアクセスするコンピュータから **scsi_id -g -s /block/sd*** コマンドで UUID 出力が 同一かどうか確認します。

4. デバイスを命名するルールを作成します。**/etc/udev/rules.d** ディレクトリ内に **20-names.rules** と言う名のファイルを作成します。このファイルに新規のルールを 追加します。全てのルールは同じ形式を使用して同じファイルに追加されます。ルールは以下の形式に従います:

```
KERNEL=="sd[a-z]", BUS=="scsi", PROGRAM="/sbin/scsi_id -g -s
/block/%k", RESULT="UUID", NAME="devicename"
```

*UUID*と *devicename* を上記で取り込んだ UUID とそのデバイス名で入れ替えます。 以下が上記の例のためのルールです:

```
KERNEL="sd*", BUS="scsi", PROGRAM="/sbin/scsi_id -g -s",
RESULT="3600a0b80001327510000015427b625e", NAME="rack4row16"
```

udev デーモンはここで、ルール内で UUID 用の **/dev/sd*** と言う名の全てのデバイスを検索します。マッチするデバイスが、 システムに接続されると、デバイスはルールから名前が割り当てられます。**3600a0b80001327510000015427b625e** の UUID を持つデバイス内では、**/dev/rack4row16** として表示されます。

5. **/etc/rc.local** に以下の行を追記します:

```
/sbin/start_udev
```

6. **/etc/scsi_id.config**、**/etc/udev/rules.d/20-names.rules**、及び **/etc/rc.local** のファイル内の変化をそれぞれ全ての関連ホストにコピーします。

```
/sbin/start_udev
```

設定済のルールを持つネットワークストレージデバイスはここで、 ファイルが更新された全てのホスト上で永続化した名前を取ります。このことは、 共有ストレージを使用してホスト間でゲストを移行できて、ゲストはそれらの 設定ファイル内のストレージデバイスにアクセスできるようになると言う意味です。

マルチパスの設定

multipath パッケージはコンピュータからストレージデバイスへ 複数のパスを持つシステム用に使用されます。**multipath** は障害 許容、フェイルオーバー、及びパフォーマンス強化を Red Hat Enterprise Linux システムに 付帯しているネットワークストレージデバイスに提供するものです。

multipath 環境内に LUN 永続化を実装するには マルチパスデバイス用に定義したエイリアス名が必要となります。各ストレージ デバイスは、エイリアス化した名前のキーとして機能する UUID を持っています。 **scsi_id** コマンドの使用で デバイスの UUID を識別 することが出来ます。

```
# scsi_id -g -s /block/sdc
```

multipath デバイスは **/dev/mpath** ディレクトリ内で 作成されます。以下の例では、４つのデバイスが **/etc/multipath.conf** 内で定義されています:

```
multipaths {
        multipath {
        wwid                3600805f30015987000000000768a0019
        alias                oramp1
        }
        multipath {
        wwid                3600805f30015987000000000d643001a
        alias                oramp2
        }
        mulitpath {
        wwid                3600805f300159870000000086fc001b
        alias                oramp3
        }
        mulitpath {
        wwid                3600805f3001598700000000984001c
        alias                oramp4
        }
}
```

この設定は、**/dev/mpath/oramp1**、**/dev/mpath/oramp2**、**/dev/mpath/oramp3**、及び **/dev/mpath/oramp4** と言う名前の ４つの LUN を作成します。入力されるとデバイスの WWID のそれらの新規名へのマッピングが この時点で再起動後に永続化されます。

8.4. 仮想化した CD-ROM 又は DVD デバイスをゲストに追加

ゲストがオンライン中の間にゲストへ ISO ファイルを添付するには、**virsh** コマンドを ***attach-disk*** パラメータ付きで使用します。

```
# virsh attach-disk  [domain-id] [source] [target] --driver file --type
cdrom --mode readonly
```

source と *target* のパラメータはそれぞれホストとゲスト上のファイルとデバイス用のパスです。*source* パラメータは ISO ファイルへのパス、又は **/dev** ディレクトリからのデバイスになります。

第9章 共有ストレージと仮想化

この章では、Red Hat Enterprise Linux 上での仮想化を使用したネットワーク化した 共有ストレージを説明しています。

以下のメソッドが仮想化でサポートされています:

- ファイバーチャンネル
- iSCSI
- NFS
- GFS2

ネットワーク化したストレージは、ライブ及びオフラインのゲスト移行の基盤と なります。共有ストレージ無しではゲストを移行できません。

9.1. ゲストのストレージの為に ISCSI を使用

このセクションでは、仮想化ゲストのストレージ用に iSCSI ベースのデバイスの使用を説明しています。

9.2. ゲストのストレージの為に NFS を使用

このセクションでは、仮想化ゲストのストレージ用に NFS の使用を説明しています。

9.3. ゲストのストレージの為に GFS2 を使用

このセクションでは、仮想化ゲストのストレージ用に Red Hat Global File System 2 (GFS2) の 使用を説明しています。

第10章 サーバーの最善使用法

以下のタスクとヒントは、ご使用の Red Hat Enterprise Linux 5 サーバーホスト (dom0) の信頼性の確立と保持を手助けします。

- SELinux を強制モードで実行します。これは以下のコマンドを実行して達成できます。

```
# setenforce 1
```

- **AutoFS, NFS, FTP, HTTP, NIS, telnetd, sendmail** などのような不要なサービスを削除、又は無効にします。
- サーバー上でプラットフォーム管理に必要となる最低限のユーザーアカウントだけを追加し、不要なユーザーアカウントは削除します。
- 使用中のホストでは必要でないアプリケーションの実行は避けて下さい。ホスト上で アプリケーションを実行すると、仮想マシンのパフォーマンスに影響を与えて、サーバーの 安定性にも影響します。サーバーをクラッシュする可能性のあるアプリケーションはいずれも サーバー上の全ての仮想マシンが落ちる原因にもなります。
- 仮想マシンのインストールとイメージには中心となる場所を使います。仮想マシンの イメージは **/var/lib/libvirt/images/** の下に格納すべきです。 仮想マシンのイメージ用に異なるディレクトリを使用している場合は、そのディレクトリを 確実に SELinux ポリシーに追加して、インストールを開始する前にそれを再ラベルします。
- インストールのソース、ツリー、及びイメージは中心的な場所に保存されるべき です。それは通常、ご使用の **vsftpd** サーバーの場所になります。

第11章 仮想化のセキュリティ

企業のインフラストラクチャに仮想化技術を導入する時には、ホストが侵害を受けないことを 確実にする必要があります。Xen hypervisor 内では、ホストはシステム管理を担当し、全ての仮想マシンを管理する権限のあるドメインです。このホストが安全でないと、システム内の他の全てのドメインが脆弱となります。仮想化を使用するシステム上でのセキュリティを強化するのに数種類の方法があります。担当者、又はその組織は、運営仕様を含む **導入プラン** を作成して、仮想化ゲストとホストサーバーにどのサービスが必要か、及びそれらのサービスに何が 必要かと言う項目を指定すべきです。導入プランを開発する段階で考慮すべきセキュリティ問題を いくつか以下に示します:

- ホスト上では必要なサービスのみを実行します。ホスト上で実行しているプロセスと サービスが少ない程、セキュリティとパフォーマンスのレベルが高くなります。
- hypervisor 上で *SELinux* を有効にします。SELinux と仮想化の使用に関する詳細には 「SELinux と仮想化」 を お読み下さい。
- ファイアウォールを使用して、dom0 へのトラフィックを制限します。デフォルトの拒否規則でファイアウォールを設定すると、dom0 への攻撃から保護をする助けになります。また、ネットワークが直面するサービスを制限することも大切です。
- 一般ユーザーには dom0 へのアクセスを禁止します。一般ユーザーに dom0 へのアクセスを許すと、dom0 を危険に曝す恐れがあります。dom0 は特権用であり、非権限者に許可をすることはセキュリティレベルを低下することになります。

11.1. SELINUX と仮想化

SELinux（Security Enhanced Linux）は Linux コミュニティの支援を受けて NSA によって開発されており、Linux の為により 強力なセキュリティを提供します。SELinux は攻撃者の能力を制限し、そしてバッファオーバーフロー攻撃や権限昇格などの多くの一般的な侵害を阻止する ように機能します。この利便性の理由で、全ての Red Hat Enterprise Linux システムが SELinux を有効にして強制モードになっていることを Red Hat は推奨しています。

SELinux は SELinux が有効になっていて、イメージが正しいディレクトリに無い場合には、 ゲストイメージのロードを阻止します。SELinux は全てのゲストイメージが **/var/lib/libvirt/images** 内に存在することを要求します。

SELinux を強制モードにして LVM ベースのストレージを追加

以下のセクションは、論理ボリュームを SELinux が有効になった仮想化ゲストに 追加するサンプルです。これらの案内はハードドライブのパーティション設定にも 役に立ちます。

手順11.1 SELinux が有効になった仮想化ゲスト上で論理ボリュームの作成とマウント

1. 論理ボリュームを作成します。このサンプルでは、***NewVolumeName*** と言う名前の 5 ギガバイトの論理ボリュームを ***volumegroup*** と言う名前のボリュームグループ上に 作成します。

   ```
   # lvcreate -n NewVolumeName -L 5G volumegroup
   ```

2. ***NewVolumeName*** の論理ボリュームを ext3 などの 拡張属性をサポートするファイルシステムでフォーマットします。

   ```
   # mke2fs -j /dev/volumegroup/NewVolumeName
   ```

3. 新規の論理ボリュームをマウントする為の新規ディレクトリを作成します。この ディレクトリはユーザーファイルシステムのどこにでも置けます。しかし、それは 重要なシステムディレク

トリ（**/etc**、**/var**、**/sys** など）や、ホームディレクトリ（**/home** 又は **/root**）には配置しないように推奨します。 このサンプルでは、**/virtstorage** と言うディレクトリを使用します。

```
# mkdir /virtstorage
```

4. 論理ボリュームのマウント

```
# mount /dev/volumegroup/NewVolumeName /virtstorage
```

5. Xen フォルダー用に 正しい SELinux のタイプをセット

```
semanage fcontext -a -t xen_image_t "/virtualization(/.*)?"
```

別の方法として、KVM のフォルダーに正しい SELinux のタイプをセットします。

```
semanage fcontext -a -t virt_image_t "/virtualization(/.*)?"
```

targeted ポリシーが使用されている場合（targeted ポリシーがデフォルト）、 このコマンドは **/etc/selinux/targeted/contexts/files/file_contexts.local** ファイルに１行を追加して、それがこの変更を 恒久化します。追加される１行は以下に似ているものです:

```
/virtstorage(/.*)?    system_u:object_r:xen_image_t:s0
```

6. マウントポイント（**/virtstorage**）とその下の全てのファイルのタイプを**xen_image_t** に変更するようにコマンドを実行します。（**restorecon** と **setfiles** は **/etc/selinux/targeted/contexts/files/** 内のファイルを読み込みます）

```
# restorecon -R -v /virtualization
```

11.2. SELINUX の考慮

このセクションには、仮想化のデプロイに於いて SELinux を使用する時点に考慮すべき 事項が含まれています。システムの変更を導入したり、デバイスを追加する時は、 それに応じて SELinux ポリシーを更新する必要があります。ゲスト用に LVM ボリュームを 設定するには、それぞれの背後にあるブロックデバイスとボリューム グループ用に SELinux コンテキストを修正しなければなりません。

```
# semanage fcontext -a -t xen_image _t -f -b /dev/sda2
# restorecon /dev/sda2
```

ブーリアンパラメータ **xend_disable_t** はデーモンを再起動した後に **xend** を規制のないモードにします。システム全体よりも単独デーモンだけの保護を無効にする方が無難です。ディレクトリを他の場所で使用する **xen_image_t** として再ラベルすることは避けるように 推奨します。

第12章 ネットワークの設定

このページは、libvirt ベースのアプリケーションで使用される一般的なネットワーキング 設定への導入を提供しています。この情報は Xen、KVM、その他の全ての hypervisor に 適用されます。追加の情報には、libvirt ネットワークアーキテクチャドキュメントをご覧 下さい。

２つの一般的なセットアップとして「仮想ネットワーク」と「共有物理デバイス」が あります。前者は全てのディストリビューションに渡って同一であり配付状態のままで 利用できます。後者はディストリビューション特有の手動設定が必要です。

12.1. LIBVIRT を持つ NAT（NETWORK ADDRESS TRANSLATION）

ネットワーク接続共有の為の最も一般的な方法の１つは、NAT (Network address translation) の 転送（別名、仮想ネットワーク）です。

ホストの設定

全ての標準の **libvirt** インストールは仮想マシンへの NAT ベースの 接続機能を配付状態のままで提供します。これがいわゆる仮想ネットワークです。コマンド **virsh net-list --all** を使用すれば、その利用可能性を確認できます。

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

存在しない場合は、サンプルの XML 設定ファイルを再ロードしてアクティベートします:

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

このデフォルトのネットワークは **/usr/share/libvirt/networks/default.xml** で定義されています。

デフォルトネットワークを自動スタートとしてマークします:

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

デフォルトネットワークをスタートします:

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

libvirt デフォルトネットワークが 稼働始めると、孤立したブリッジデバイスを見ることができます。このデバイスは、NAT と IP 転送を使用して外部に接続するため、物理的なインターフェイスの追加は**ありません**。新規のインターフェイスを追加しないで下さい。

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

libvirt は、ゲストとのトラフィックを許可する **iptables** ルールを追加します。このゲストは **INPUT**、**FORWARD**、**OUTPUT**、及び **POSTROUTING** のチェーン内の **virbr0** デバイスに付帯していま

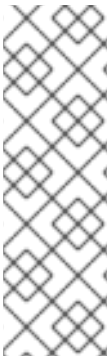
す。**libvirt** はそれから、 **ip_forward** パラメータの有効化を試みます。他の一部のアプリケーションが **ip_forward** を無効にする可能性があるため、最善の選択肢は**/etc/sysctl.conf** に以下を追加することです。

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

ゲストの設定

ホストの設定が完了すると、ゲストはその名前を基にした仮想ネットワークに 接続可能になります。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、 ゲスト内で以下の XML を使用します:

```
<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。無視すると、MAC アドレスは自動的に 生成されます。MAC アドレスの手動セッティングは一部の状況で役に立ちます。

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

12.2. LIBVIRT を使用したブリッジネットワーキング

ブリッジネットワーキング（別名、物理デバイス共有）は、物理デバイスを仮想マシンに 専従させるために使用します。ブリッジングは多くの場合、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上で使用されます。

Xen ネットワークスクリプトを無効にする

システムが Xen ブリッジを使用している場合は、**/etc/xen/xend-config.sxp** への編集で以下の行を変更することにより、デフォルトの Xen ネットワーキングブリッジを 無効にすることが推奨されます:

```
(network-script network-bridge)
```

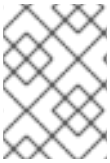
から:

```
(network-script /bin/true)
```

NetworkManager を無効にする

NetworkManager はブリッジングをサポートしません。NetworkManager を無効にして ネットワークスクリプト（**/etc/sysconfig/network-scripts/** ディレクトリ内に 存在）を介したネットワーキングを使用する必要があります。

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



注記

NetworkManager をオフにする代わりに、***"NM_CONTROLLED=no"*** を サンプルで使用されている **ifcfg-*** スクリプトに追加することができます。

network initscript を作成

次の２つのネットワーク設定ファイルを作成するか、 又は編集します。この手順を追加のネットワークブリッジで（別名で）繰り返します。

/etc/sysconfig/network-scripts ディレクトリへ移動します:

```
# cd /etc/sysconfig/network-scripts
```

ブリッジに追加するデバイス用のネットワークスクリプトを開きます。この例では、 **ifcfg-eth0** は、ブリッジの一部としてセットされている 物理ネットワークインターフェイスを定義しています:

```
DEVICE=eth0
# change the hardware address to match the hardware address your NIC uses
HWADDR=00:16:76:D6:C9:45
ONBOOT=yes
BRIDGE=br0
```

注記

デバイスの MTU (Maximum Transfer Unit) を設定するには、***MTU*** 変数を 設定ファイルの末尾に追記します。

```
MTU=9000
```

ifcfg-br0、又はそれに似た名前のネットワークスクリプトを**/etc/sysconfig/network-scripts** ディレクトリ内に作成します。 ***br0*** とは、ブリッジの名前です。これはファイル名が DEVICE パラメータと同じであれば、どんな名前でも結構です。

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=dhcp
ONBOOT=yes
DELAY=0
```

警告

TYPE=Bridge の行は、大文字/小文字の区別があります。これは 大文字の「B」と 小文字の 「ridge」 で構成れています。

設定が終了したら、ネットワークを再開始するか、マシンをリブートします。

■

```
# service network restart
```

iptables を設定して、全てのトラフィックがブリッジを渡って 転送されるようにします。

```
# iptables -I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
# service iptables save
# service iptables restart
```

注記

別の方法としては、**iptables** ルールを使って、ブリッジされた トラフィックがプロセスされることを阻止します。**/etc/sysctl.conf** 内で 以下の行を追記します:

```
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0
```

sysctl の使用で設定されたカーネルパラメータを再ロードします。

```
# sysctl -p /etc/sysctl.conf
```

libvirt デーモンを再起動

```
# service libvirtd reload
```

この時点で、「共有物理デバイス」があるはずです。これはゲストを 付帯できるもので、全面的な LAN アクセスを持ちます。以下のようにして 新規ブリッジを確認します:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
br0             8000.000e0cb30550       no              eth0
```

注記。このブリッジは完全に **virbr0** ブリッジから独立しています。 物理デバイスを **virbr0** に付帯する試行は **しないで下さい。 virbr0** ブリッジは NAT (Network Address Translation) 接続機能だけのために あります。

第13章 RED HAT ENTERPRISE LINUX 5.4 以前の XEN ネットワーキング

この章では、Xen hypervisor を使用したネットワーキングとネットワーク設定の為の 特殊なトピックを扱います。

ほとんどのゲストネットワーク設定はゲストの初期化中とインストールのプロセス中に 発生します。ゲストのインストールプロセス中のネットワーク設定に関する情報を 得るには、インストールプロセスの関連セクション 6章 *仮想化ゲストインストールの概要*をお読み下さい、

ネットワーク設定は、**virsh** (22章 *virsh でゲストを管理*) と **virt-manager** (23章*仮想マシンマネージャ(virt-manager) でゲストを管理する*) のための ツール特有の参照の章でも取り扱っています。これらの章では、これらのツールを使用した ネットワーキング設定の詳細説明が提供されています。

注記

para-virtualized ネットワークドライバーを使用すると、完全仮想化 Linux ゲストに パフォーマンスの向上を与えます。14章 *Xen Para-virtualized ドライバー*では、utilize para-virtualized ネットワークドライバーの 活用法を説明しています。

13.1. 複数のゲストネットワークブリッジを設定して複数のイーサネットカードを使用

ネットワークブリッジをセットアップするプロセス（Xen hypervisor 使用）:

1. **system-config-network** アプリケーションを使用して もう１つのネットワークインターフェイスを設定します。別の方法として、**/etc/sysconfig/network-scripts/** ディレクトリ内に **ifcfg-ethX** と言う名の新しい 設定ファイルを作成します。ここで***X*** とは、いずれかの未使用の番号です。 **eth1** と言う名の２つめのネットワークインターフェイス用の サンプルの設定ファイルを以下に示します。

```
$ cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
USERCTL=no
IPV6INIT=no
PEERDNS=yes
TYPE=Ethernet
NETMASK=255.255.255.0
IPADDR=10.1.1.1
GATEWAY=10.1.1.254
ARP=yes
```

2. ファイル **/etc/xen/scripts/network-bridge** を,**/etc/xen/scripts/network-bridge.xen** にコピーします。

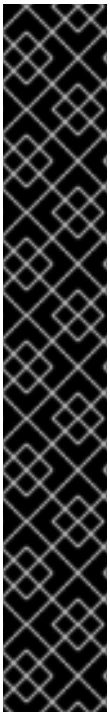
3. **/etc/xen/xend-config.sxp** 内の既存のネットワークスクリプトを コメントアウトして、***(network-xen-multi-bridge)*** と言う行を 追加します。

4. カスタムスクリプトを作成して複数のネットワークブリッジを作ります。以下に サンプルのスクリプトを示します。このサンプルスクリプトは２つの Xen ネットワーク ブリッジ（xenbr0 と xenbr1）作り、その１つは eth1 に付帯して、もう１つは eth0 に 付帯します。追加のブリッ

ジを作成したい場合は、スクリプト内の例に従って、該当する 行をコピー/ペーストして下さい:

```
#!/bin/sh
# network-xen-multi-bridge
# Exit if anything goes wrong.
set -e
# First arg is the operation.
OP=$1
shift
script=/etc/xen/scripts/network-bridge.xen
case ${OP} in
start)
        $script start vifnum=1 bridge=xenbr1 netdev=eth1
        $script start vifnum=0 bridge=xenbr0 netdev=eth0
        ;;
stop)
        $script stop vifnum=1 bridge=xenbr1 netdev=eth1
        $script stop vifnum=0 bridge=xenbr0 netdev=eth0
        ;;
status)
        $script status vifnum=1 bridge=xenbr1 netdev=eth1
        $script status vifnum=0 bridge=xenbr0 netdev=eth0
        ;;
*)
        echo 'Unknown command: ' ${OP}
        echo 'Valid commands are: start, stop, status'
        exit 1
esac
```

13.2. RED HAT ENTERPRISE LINUX 5.0 ラップトップネットワークの設定



重要

このセクションでは、ネットワークブリッジを手動で追加する方法を説明します。この手順は、 Red Hat Enterprise Linux のバージョン 5.0 以降の全てのバージョンでは必要でもなく、また 推奨もされません。5.0 以降の新しいバージョンは、**virt-manager** でゲストを作成するときに **"仮想ネットワーク"** アダプタを使用します。NetworkManager は Red Hat Enterprise Linux 5.1 及び それ以降でデフォルトで仮想ネットワークデバイスを使用して作動します。

仮想ネットワークデバイス用の virsh XML 設定ファイルのサンプル:

```
<interface type='network'>
        <mac address='AA:AA:AA:AA:AA:AA'/>
        <source network='default'/>
        <target dev='vnet0'/>
        <model type='virtio'/>
</interface>
```

xm 設定ファイル内で、仮想ネットワークデバイスは **"vif"** のラベルを持ちます。

ラップトップ上で Xen hypervisor を実行することに関しての挑戦課題は ほとんどのラップトップがワ

イヤレスネットワーク、又は有線の接続を介して ネットワークに接続することです。多くの場合、これらの接続は１日のうちに何回も オン/オフになります。そのような環境で、システムはいつでも同じインターフェイスに アクセスを持つと想定し、それが使用しているネットワークインターフェイスへ **ifup** コール又は、**ifdown** コールを 実行することもあります。更には、ワイヤレスネットワークカードは、 Xen の（デフォルトの）ブリッジ化したネットワーク使用の理由で、仮想化環境内では うまく機能しません。

このセットアップはユーザーがラップトップにアクティブなネットワーク接続を 持たない時にオフラインモードで Xen を実行できるようにします。ラップトップ上で Xen を実行する最も簡単な方法は、以下に概要のある手順に従うことです:

- ユーザーはダミーのネットワークインターフェイスを設定して、それが Xen で 使用できるようにします。この例では、インターフェイスは **dummy0** と名付けます。これによりユーザーはゲストの為に隠れ IP アドレスの スペースを使用することができます。
- DHCP は DHCP 要求のためのダミーインターフェイスをリッスンしないため、静的 IPアドレスを 使用する必要があります。ユーザー自身のバージョンの DHCP をコンパイルしてダミーインターフェイスで リッスンするようにはできます。しかし、Xen 環境内で DNS、DHCP、及び **tftpboot** サービスの為の **dnsmasq** の使用を考慮してみましょう。セットアップと設定はこの章/セクションの後半で説明してあります。
- NAT と IP マスカレーディングを設定することにより、ゲストからのネットワークへの アクセスを有効にできます。

ダミーネットワークインターフェイスの設定

使用するホスト上で以下の設定手順を実行します:

1. dummy0 のネットワークインターフェイスを作成して、それに静的 IP アドレスを割り当てます。 ここでの例では、現在の環境でルーティング問題を防止するために 10.1.1.1 を選択しています。 ダミーデバイスサポートを有効にするには、以下の行を **/etc/modprobe.conf** に追加します。

```
alias dummy0 dummy
options dummy numdummies=1
```

2. dummy0 用のネットワーキングを設定するには、**/etc/sysconfig/network-scripts/ifcfg-dummy0** を編集/作成します:

```
DEVICE=dummy0
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
IPV6INIT=no
PEERDNS=yes
TYPE=Ethernet
NETMASK=255.255.255.0
IPADDR=10.1.1.1
ARP=yes
```

3. **xenbr0** を **dummy0** に バインドして、物理ネットワークに接続していない時でもネットワークを使用できるようにします。 **/etc/xen/xend-config.sxp** を編集して、**netdev=dummy0** エントリを含むようにします:

```
(network-script 'network-bridge bridge=xenbr0 netdev=dummy0')
```

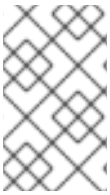
4. ゲスト内で **/etc/sysconfig/network** を開き、デフォルトのゲートウェイが **dummy0** を指すように修正します。静的 IP を使用している場合は、 ゲストの IP アドレスを **dummy0** と同じサブネット上で存在するように セットします。

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
GATEWAY=10.1.1.1
IPADDR=10.1.1.10
NETMASK=255.255.255.0
```

5. ホスト内に NAT をセットアップすると、ワイヤレスを含むインターネットアクセスがゲストで可能になります。 そして Xen とワイヤレスカードの問題を解決します。以下のスクリプトにより、現在ユーザーの ネットワーク接続で使用されているインターフェイスを基にした NAT を有効にします。

仮想化ゲスト用 NAT の設定

NAT (Network address translation) を使用すると、パケットの妨害をしてそれをプライベート IP アドレスに 渡すことにより複数のネットワークアドレスが 1 つの IP アドレスを介して接続できるようになります。 以下のスクリプトを **/etc/init.d/xenLaptopNAT** にコピーして、**/etc/rc3.d/S99xenLaptopNAT** へのソフトリンクを作成することができます。これが自動的にブート時に NAT を開始します。



注記

以下のスクリプトは、スタートアップ遅延のため、ワイヤレスネットワークや **NetworkManager** ではうまく機能しない可能性があります。 この場合、マシンがブートした後にスクリプトを手動で実行することになります。

```
#!/bin/bash
PATH=/usr/bin:/sbin:/bin:/usr/sbin
export PATH
GATEWAYDEV=`ip route | grep default | awk {'print $5'}`
iptables -F
case "$1" in
start)
        if test -z "$GATEWAYDEV"; then
        echo "No gateway device found"
    else
        echo  "Masquerading using $GATEWAYDEV"
        /sbin/iptables -t nat -A POSTROUTING -o $GATEWAYDEV -j MASQUERADE
fi
        echo "Enabling IP forwarding"
        echo 1 > /proc/sys/net/ipv4/ip_forward
        echo "IP forwarding set to `cat /proc/sys/net/ipv4/ip_forward`"
        echo "done."
        ;;
*)
echo "Usage: $0 {start|restart|status}"
;;
esac
```

DNS、DHCP、及び tftpboot サービスのために dnsmasq を設定します。

ラップトップ（又は、単独の安定したネットワーク接続に継っていないいずれかのコンピュータ）上

で 仮想化を実行することの挑戦課題の１つは、ネットワークインターフェイスの変化と可用性です。ダミーネットワークインターフェイスを使用すると、より安定した環境を設立できますが、それはまた、 DHCP、DNS、及び tftpboot サービスなどをユーザーの仮想化マシン/ゲストに提供することに於いて新しい挑戦を 持ち込みます。Red Hat Enterprise Linux と Fedora Core で配付されているデフォルトの DHCP デーモンは ダミーインターフェイスでリッスンしません。ユーザーの DNS 転送された情報は、異なるネットワークや VPN に 接続すると変化する可能性があります。

上記の挑戦課題への１つの解決案は、１つのパッケージのみで上記のサービスの全てを 提供できて、ユーザーが自分のダミーインターフェイスからの要求のみに利用可能なサービスを 制御できるようにしてくれる dnsmasq の使用です。以下に、仮想化を実行しているラップトップで **dnsmasq** を設定する方法についての短い説明を示します:

- ここ から最新バージョンの dnsmasq を取得します。
- dnsmasq のドキュメントは ここ で見ることができます。
- 以下で参照されている他のファイルを http://et.redhat.com/~jmh/tools/xen/ からコピーして、ファイル **dnsmasq.tgz** を取得します。この tar アーカイブには、以下のファイルが含まれています:
  - **nm-dnsmasq** は NetworkManager の為のディスパッチャスクリプトとして 使用できます。これは NetworkManager が接続に変化を検出する度に実行されて、dnsmasq の 再スタート/再ロードを強制します。これは **/etc/NetworkManager/dispatcher.d/nm-dnsmasq** に コピーする必要があります。
  - **xenDNSmasq** は、**/etc/init.d/xenDNSmasq** の 主要スタートアップ、又はシャットダウンのスクリプトとして使用できます。
  - **dnsmasq.conf** は **/etc/dnsmasq.conf** の サンプルの設定ファイルです。
  - **dnsmasq** は **/usr/local/sbin/dnsmasq** のバイナリイメージです。
- dnsmasq （デフォルトインストールは **/usr/local/sbin/dnsmasq** へのバイナリ）を展開してビルドした後は、dnsmasq 設定ファイルを編集する必要があります。このファイルは **/etc/dnsmaqs.conf** にあります。
- 設定を編集してユーザーのローカルニーズと要件に合うようにします。以下の パラメータはたぶんユーザーが修正すべきものでしょう:
  - **interface** パラメータは **dnsmasq** が 指定したインターフェイスのみで **DHCP** と **DNS** 要求をリッスンするようにします。それは ダミーインターフェイスでもありえますが、ユーザーの公共インターフェイスやローカルループバックインターフェイス ではありえません。複数インターフェイスの為にもう１つの **interface** 行を 追加します。**interface=dummy0** は、**dummy0** インターフェイスでリッスンする１つの例です。
  - 統合された **DHCP** サーバーを有効にする **dhcp-range** には、リース及びオプションのリース期間用に 利用できるアドレスの範囲を供給する必要があります。複数のネットワークを使用している場合は、 **DHCP** を適用したい各ネットワーク上でこれを繰り返す 必要があります。１つの例として、**dhcp-range=10.1.1.10,10.1.1.50,255.255.255.0,12h** が あります（ネットワーク 10.1.1.* 用であり、12時間のリース期間）。
  - dnsmasq により供給されたデフォルトのルートを上書きするための **dhcp-option** は ルーターが dnsmasq を実行しているマシンと同じだと想定します。例としては **dhcp-option=3,10.1.1.1** があります。

- dnsmasq を設定した後は、以下にあるスクリプトを **xenDNSmasq** として、 **/etc/init.d** にコピーできます。
- システムブート中に自動的に dnsmasq を開始したい場合は、chkconfig(8) を使用して それを登録する必要があります:

```
chkconfig --add xenDNSmasq
```

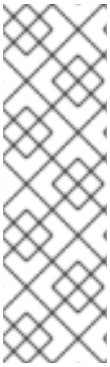
自動スタートアップ用にそれを有効にします:

```
chkconfig --levels 345 xenDNSmasq on
```

- **NetworkManager** が接続の変化を検出する度に 再スタートするように**dnsmasq** を設定するには、 供給されているスクリプト **nm-dnsmasq** を使用すれば 達成できます。
  - **nm-dnsmasq** スクリプトを **/etc/NetworkManager/dispatcher.d/** にコピーします。
  - **NetworkManager** ディスパッチャは、接続に 変化がある度にそのスクリプト（同じディレクトリ内に他のスクリプトがあるとアルファベット順）を 実行します。
- **dnsmasq** もまた ユーザーの **/etc/resolv.conf** 内の変化を検出して自動的にそれらを再ロードします （例えば VPN セッションをスタートした場合など）。
- ユーザーが隠れたネットワークに仮想化ゲストを持ち、それらに公共ネットワークへの アクセスを許可している場合、**nm-dnsmasq** スクリプトと **xenDNSmasq** スクリプトは両方共、NAT もセットアップします。

第14章 XEN PARA-VIRTUALIZED ドライバー

*Para-virtualized ドライバー*は完全仮想化した Red Hat Enterprise Linux ゲスト用に パフォーマンスの向上を提供します。完全仮想化した Red Hat Enterprise Linux ゲストを使用していて より良いパフォーマンスを必要とする場合には、これらのドライバーを使用して下さい。



注記

Windows 対応の Xen と KVM hypervisor 用には他の para-virtualized ドライバーが あります。

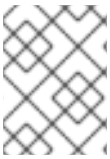
Xen ホスト上の Windows ゲスト用には、そのインストールと管理を案内している『Windows Para-virtualized ドライバーガイド』 を参照して 下さい。

KVM ホスト上の Windows ゲスト用には、15章*KVM Para-virtualized ドライバー*を参照して下さい。

para-virtualized ドライバー用の RPM パッケージには、サポートのある Red Hat Enterprise Linux ゲストオペレーティングシステムの為の ストレージとネットワーキング用の para-virtualized ドライバーモジュールが含まれています。これらのドライバーは Red Hat Enterprise Linux 5.1 （又はそれ以降）のホスト上で無修正の Red Hat Enterprise Linux ゲスト 内で I/O 処理能力のハイパフォーマンスを可能にします。

サポートされているゲストオペレーティングシステムを以下に示します:

- Red Hat Enterprise Linux 3
- Red Hat Enterprise Linux 4
- Red Hat Enterprise Linux 5



注記

最低限のゲストオペレーティングシステム要件はアーキテクチャ依存です。 x86 と x86-64 のゲストのみがサポートされています。

Red Hat Enterprise Linux 3 以前の Red Hat Enterprise Linux ゲストオペレーティング システムではこのドライバーはサポートされていません。

仮想化のプラットフォームとして Red Hat Enterprise Linux 5 を使用すると、 システム管理者が Linux と Windows の作業負荷を、増加したパワーと冷却効率を持つ 新しくてより強力なハードウェアに統合できるようになります。Red Hat Enterprise Linux 4 （update 6 の時点）と Red Hat Enterprise Linux 5 の ゲスト オペレーティングシステムは背後にある仮想化技術を認識し、特定のインターフェイスと 能力を使用して効率的にその技術と交流します。このアプローチはベアメタルシステム上で 実行しているのと比較して同等レベルの処理能力とパフォーマンス特性を達成できます。

para-virtualization は修正されたゲストオペレーティングシステムを必要と するため、全てのオペレーティングシステムが para-virtualization を使用できる わけではありません。修正できないオペレーティングシステムには、完全仮想化が 必要になります。完全仮想化はデフォルトで、模倣したディスク、ネットワーク、ビデオ、 及び他のハードウェアデバイスを使用します。模倣した I/O デバイスはかなり遅くなり、 高度のディスク、及びネットワークスループットを要求するアプリケーションには、 適合しません。仮想化に於けるパフォーマンス損失の大部分は模倣デバイスの使用を 通じて発生するものです。

para-virtualized デバイスドライバーは Red Hat Enterprise Linux パッケージに 収納されています。

para-virtualized デバイスドライバーのみが背後にある仮想プラット フォームを認識している（OS の他の部分は認識していない）ため、このドライバーは 無修正の OS に対して para-virtualized ゲスト OS の優越したパフォーマンスを提供 します。

para-virtualized デバイスドライバーをインストールした後は、ディスクデバイスや ネットワークカードは通常の物理ディスクやネットワークカードとしてオペレーティング システムに現れ続けます。しかし、その後デバイスドライバーは仮想化プラットフォーム （模倣無し）と直接交流して、効率的にディスクとネットワークアクセスを提供し、ディスクと ネットワークサブシステムが仮想化環境内でもネイティブに近いスピードで動作できるようにします。 そして既存のゲストオペレーティングシステムへの変更を必要としません。

para-virtualized ドライバーは特定のホスト要件を持っています。64 bit の ホストは以下を実行できます:

- 32 bit ゲスト
- 64 bit ゲスト
- 32 bit ゲストと 64 bit ゲストの混合

14.1. システム要件

このセクションでは、Red Hat Enterprise Linux で使用する para-virtualized ドライバーの要件を提供します。

インストール

para-virtualized ドライバーをインストールする前に、以下の要件（一覧表内）が 満足されることを確認して下さい。

重要

Red Hat Enterprise Linux の 4.7 と 5.3 以降の 全ての バージョンは、para-virtualized ドライバー、**pv-on-hvm** モジュール用のカーネル モジュールをデフォルトのカーネル パッケージ内に持っています。このことは、 para-virtualized ドライバーが Red Hat Enterprise Linux 4.7 とそれ以降、又は 5.3 とそれ以降のゲスト用に利用可能だと言う意味になります。

それぞれのゲストオペレーティングシステムインストール毎に para-virtualized ドライバーのための以下の RPM パッケージが必要です。

最低限のホストオペレーティングシステムバージョン:

- Red Hat Enterprise Linux 5.1 又はそれ以降。

最低限ゲストオペレーティングシステムバージョン:

- Red Hat Enterprise Linux 5.1 又はそれ以降。
- Red Hat Enterprise Linux 4 Update 6 又はそれ以降。
- Red Hat Enterprise Linux 3 Update 9 又はそれ以降。

Red Hat Enterprise Linux 5 は以下を要求:

- **kmod-xenpv**.

Red Hat Enterprise Linux 4 は以下を要求:

- **kmod-xenpv**,
- **modules-init-tools** (Red Hat Enterprise Linux 4.6z 以前のバージョンには、以下が必要: **modules-init-tools-3.1-0.pre5.3.4.el4_6.1** 又はそれ以降), 及び
- **modversions**.

Red Hat Enterprise Linux 3 は以下を要求:

- **kmod-xenpv**.

最低でも 50MB の空きディスク容量が **/lib** ファイルシステム内に必要。

14.2. PARA-VIRTUALIZATION の制約とサポート

このセクションでは、Red Hat Enterprise Linux 上での para-virtualized ドライバーの使用に於けるサポートの制約と必要事項に ついて概要を示します。サポートされる事項とサポートに課される 制約は以下のセクションでご覧になれます。

サポートされているゲストオペレーティングシステム

para-virtualized ドライバーへのサポートは、以下のオペレーティング システムとバージョンで利用可能です:

- Red Hat Enterprise Linux 5.1 及びそれ以降
- Red Hat Enterprise Linux 4 Update 6 及びそれ以降
- Red Hat Enterprise Linux 3 Update 9 及びそれ以降

64 bit Red Hat Enterprise Linux 5 Virtualization 上での、para-virtualized ドライバーを持つ 32 bit ゲストオペレーティングシステムの 実行はサポートされています。

以下の表では、para-virtualized ドライバーでサポートされているカーネルの変種が 示してあります。以下のコマンドを使用すると、自分のホスト上に現在インストール されている正確なカーネル改訂版を識別することができます。 出力を表と比較してサポートされているかどうか判定して下さい。

```
# rpm -q --queryformat '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}\n' kernel
```

Red Hat Enterprise Linux 5 i686 及び x86_64 のカーネル変種には、 SMP（Symmetric Multiprocessing）が含まれており、別途に SMP の カーネル RPM を必要としません。

以下の表で、Red Hat Enterprise Linux 3 ゲスト用のプロセッサ特有のカーネル 要件に注意を払って下さい。

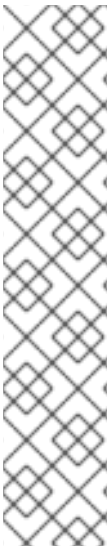
表14.1 para-virtualized ドライバー用のサポートされているゲストカーネルアーキテクチャ

カーネルアーキテクチャ	Red Hat Enterprise Linux 3	Red Hat Enterprise Linux 4	Red Hat Enterprise Linux 5
athlon	サポート有り (AMD)		
athlon-SMP	サポート有り (AMD)		

カーネルアーキテクチャ	Red Hat Enterprise Linux 3	Red Hat Enterprise Linux 4	Red Hat Enterprise Linux 5
i32e	サポート有り (Intel)		
i686	サポート有り (Intel)	サポート有り	サポート有り
i686-PAE	サポート有り		
i686-SMP	サポート有り (Intel)	サポート有り	
i686-HUGEMEM	サポート有り (Intel)	サポート有り	
x86_64	サポート有り (AMD)	サポート有り	サポート有り
x86_64-SMP	サポート有り (AMD)	サポート有り	
x86_64-LARGESMP	サポート有り		
Itanium (IA64)	サポート有り		

重要

ホストシステムは Red Hat Enterprise Linux 5.1 又はそれ以降を必要とします。



注記

以下のコマンドの出力を書き留めて記憶して置きます。この値が ダウンロードすべきパッケージとモジュールを決定する値です。

```
# rpm -q --queryformat '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}\n' kernel
```

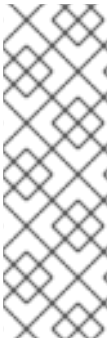
出力は以下に似ているはずです:

```
kernel-PAE-2.6.18-53.1.4.el5.i686
```

カーネルの名前は PAE （Physical Address Extensions の略）で、カーネル バージョンは 2.6.18 で、リリースが 53.1.4.el5 で、アーキテクチャが i686 となっています。カーネルの rpm は常に **kernel-name-version-release.arch.rpm** の 形式になっています。

重要な制約

Para-virtualized デバイスドライバーは、ゲストオペレーティングシステムの 正常なインストールの後で、インストールできます。これらのドライバーを インストールできるようになるには、稼働中のホストとゲストが必要です。



注記

GRUB は現時点では、para-virtualized の ブロックデバイスにアクセスできません。そのため、ゲストは、para-virtualized ブロックデバイスドライバーを使用するデバイスからはブート出来ません。 特に、MBR（Master Boot Record）を含むディスク、ブートローダー**(GRUB)** を 含むディスク、又はカーネルの **initrd** イメージを含むディスクでは出来ません。 **/boot** のディレクトリやパーティションを含んでいるディスクはいずれも、 para-virtualized ブロックデバイスドライバーを使用できないと言うことになります。

Red Hat Enterprise Linux 3 カーネル変種アーキテクチャの依存関係

Red Hat Enterprise Linux 3 ベースのゲストオペレーティングシステムには、 以下の表で見えるように、プロセッサ特有のカーネルと para-virtualized ドライバー RPM を 使用しなければなりません。適合する para-virtualized ドライバーパッケージをインストール できないと、**xen-pci-platform** モジュールのロードは失敗します。

以下の表は、ゲストが Intel プロセッサ用にコンパイルされている場合に Red Hat Enterprise Linux 3 ゲストを稼働するのに必要となるホストカーネルを 示しています。

表14.2 Intel プロセッサ用の Red Hat Enterprise Linux 3 上で para-virtualized ドライバーを使用するゲストの為の必須ホストカーネルアーキテクチャ

ゲストカーネルタイプ	必須のホストカーネルタイプ
ia32e (UP と SMP)	x86_64
i686	i686
i686-SMP	i686
i686-HUGEMEM	i686

以下の表は ゲストが AMD プロセッサ用にコンパイルされている場合に Red Hat Enterprise Linux 3 ゲストを稼働するのに必要となるホストカーネルを 示しています。

表14.3 AMD プロセッサ用の Red Hat Enterprise Linux 3 上で para-virtualized ドライバーを使用するゲストの為の必須ホストカーネルアーキテクチャ

ゲストカーネルタイプ	必須のホストカーネルタイプ
athlon	i686
athlon-SMP	i686
x86_64	x86_64
x86_64-SMP	x86_64

14.3. PARA-VIRTUALIZED ドライバーのインストール

以下の３つの章では、para-virtualized ドライバーを持つ Red Hat Enterprise Linux 5.1 か それ以降のバージョンで実行する完全仮想化ゲストのインストールと設定の方法を説明して います。

重要

Para-virtualized ドライバーは特定のハードウェアとバージョンの組み合わせでのみ サポートされています。para-virtualized ドライバーをインストールする前に、ご使用のハートウェアとオペレーティングシステムの要件が満足されているかどうかを確認して下さい。

注記

新しくゲストシステムをインストールしている場合は、para-virtualized の ブロックデバイスドライバーから最大限の利便性を得るために、少なくとも ２つのディスクでゲストを作成すべきです。

MBR と ブートローダ（**GRUB**）を収納 しているディスクのため、及び **/boot** パーティションのために para-virtualized ドライバーを使用します。このパーティションは非常に小さいもので十分です。 その理由は、それが **/boot** パーティションを保持するだけの十分な 容量があればいいからです。

２つめのディスクとその他のディスクはすべて他のパーティション用（例えば、**/**、**/usr**）、又は論理ボリューム用に使用します。

このインストール方法を使用すると、ゲストのインストールが完了した後に para-virtualized のブロックデバイスドライバーがインストールされた場合に、 ゲストの起動と **/boot** パーティションへのアクセス のみが仮想化ブロックデバイスドライバーを使用することになります。

14.3.1. 共通のインストール手順

以下の一覧は、全てのゲストオペレーティングシステムバージョンに渡って共通の 高水準の手順を説明しています。

1. 使用するハードウェアアーキテクチャ用の RPM をゲストオペレーティングシステム内の 適切な位置にコピーします。ユーザーのホームディレクトリで十分です。自分が 必要とする RPM が不明な場合は、「Para-virtualization の制約とサポート」 にある表で確認して 下さい。

2. **rpm** コマンド、又は **yum** コマンドを 使用して、パッケージをインストールします。**rpm** ユーティリティは 以下の ４つの新規カーネルモジュールを **/lib/modules/[%kversion] [%kvariant]/extra/xenpv/%release** にインストールします:

   - PCI インフラストラクチャモジュール、**xen-platform-pci.ko**
   - バルーニングモジュール、**xen-balloon.ko**
   - 仮想ブロックデバイスモジュール、**xen-vbd.ko**
   - 及び、仮想ネットワークデバイスモジュール、**xen.vnif.ko**.

3. ゲストオペレーティングシステムが para-virtualized ドライバーの自動ロードを サポートしていない場合（例えば、Red Hat Enterprise Linux 3）、必要な ポストインストールステップを実行して、そのドライバーをオペレーティングシステム 特有の場所にコピーします。

4. ゲストオペレーティングシステムをシャットダウン

5. ゲストオペレーティングシステムの設定ファイルをホスト上で再設定して、インストール済の para-virtualized ドライバーを使用します。

6. ネットワークデバイス用の “type=ioemu” エントリを削除します。

7. para-virtualized ブロックデバイスドライバー用に使用したい追加のストレージ エンティティを追加します。

8. ゲストを再起動します:

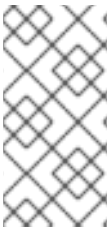
```
# virsh start YourGuestName
```

   ここで、*YourGuestName* はゲストオペレーティング システムの名前です。

9. ゲストネットワークを再設定します。

14.3.2. Red Hat Enterprise Linux 3 で Para-virtualized ドライバーのインストールと設定

このセクションには、Red Hat Enterprise 3 ゲストオペレーティングシステム内での para-virtualized ドライバーに関する詳細案内が含まれています。



注記

これらのパッケージは para-virtualized ディスクからのブートをサポートしていません。 ゲストオペレーティングシステムカーネルのブートには、まだ模倣 IDE ドライバーが 必要です。他の（システム関連ではない）ユーザースペースアプリケーションとデータは para-virtualized ブロックデバイスドライバーを使用することができます。

ドライバーのインストール

para-virtualized ドライバーを使用した Red Hat Enterprise Linux 3 ゲストのインストールに必要な手順を以下に示します。

1. 使用しているハードウェアアーキテクチャとカーネル変種用の **kmod-xenpv** rpm を ゲストオペレーティングシステムにコピーします。

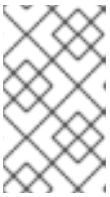
2. **rpm** ユーティリティを使用して RPM パッケージをインストール します。使用するゲストオペレーティングシステムの変種とアーキテクチャに必要なパッケージを 正しく識別していることを確認して下さい。

```
[root@rhel3]# rpm -ivh kmod-xenpv*
```

3. 以下のコマンドを実行して正しい自動化した para-virtualized ドライバーのロードを 有効にします。**%kvariant** は、それに対して para-virtualized ドライバーがビルドされたカーネル変種であり、**%release** は その para-virtualized ドライバーのリリースバージョンに相当します。

```
[root@rhel3]# mkdir -p /lib/modules/'uname -r'/extra/xenpv
[root@rhel3]# cp -R /lib/modules/2.4.21-
52.EL[%kvariant]/extra/xenpv/%release \
/lib/modules/'uname -r'/extra/xenpv
[root@rhel3]# cd /lib/modules/'uname -r'/extra/xenpv/%release
[root@rhel3]# insmod xen-platform-pci.o
```

```
[root@rhel3]# insmod xen-balloon.o`
[root@rhel3]# insmod xen-vbd.o
[root@rhel3]# insmod xen-vnif.o
```



注記

Red Hat Enterprise Linux 3 が **MODVERSIONS** を 有効にしているため、バイナリドライバーモジュールをインストールする時点で、 **insmod** が警告を生成します。これらの警告は無視 できるものです。

4. **/etc/modules.conf** を確証して、以下のような **eth0** 用のエイリアスがあることを 確認します。複数のインターフェイスを設定する予定の場合は、各インターフェイス毎に追加の 行を加えます。

```
alias eth0 xen-vnif
```

/etc/rc.local を編集して行を追加します:

```
insmod /lib/modules/'uname -r'/extra/xenpv/%release/xen-vbd.o
```

注記

“%release” を 実際の para-virtualized ドライバーのリリースバージョン（例えば、0.1-5.el）で 入れ替えます。para-virtualized ドライバーRPM パッケージを更新する場合は、そのリリースバージョンを 適切なバージョンに更新することを確認して下さい。

5. 仮想マシンをシャットダウンします（ゲスト内で **“#shutdown -h now”** を使用）。

6. 以下のようにして、**/etc/xen/YourGuestsName** 内のゲスト設定ファイルを編集します:

   - **“vif=”** エントリから **“type=ioemu”** エントリを削除します。
   - 追加のディスクパーティション、ボリューム、又は LUN をゲストに追加して para-virtualized **(xen-vbd)** ゲストドライバー経由で それらにアクセスできるようにします。
   - 各追加の物理デバイス、LUN、パーティション、又はボリューム用に、以下に示した ようなエントリを、ゲスト設定ファイル内の **“disk=”** セクションに追加します。オリジナルの **“disk=”** エントリも 以下のエントリに似ているかも知れません。

     ```
     disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w"]
     ```

   - 追加の物理デバイス、LUN、パーティション、又はボリュームを追加すると、 XML 設定ファイル内の para-virtualized ドライバーのエントリは以下にある エントリに似たものになります。

     ```
     disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w",
     "tap:aio:/var/lib/libvirt/images/UserStorage.dsk,xvda,w" ]
     ```

注記

ファイルベースのイメージが使用されている場合、para-virtualized デバイス用に **“tap:aio”** を使用します。

7. **virsh** コマンドを使用して仮想マシンをブートします:

```
# virsh start YourGuestName
```

警告

weak-modules と **modversions** のサポートは Red Hat Enterprise Linux 3 で提供されていないため、 para-virtualized ドライバーはシステムへの自動的追加とロードはされません。 モジュールを挿入するには、以下のコマンドを実行します。

```
insmod xen-vbd.ko
```

Red Hat Enterprise Linux 3 では **xen-vbd** を使用するブロック デバイス用に特殊ファイルの手動作成が必要になります。以下の手順が、para-virtualized ブロックデバイスの作成法と登録の案内となります。

para-virtualized ブロックデバイスドライバーがロードされた後に、 以下のスクリプトを使用して特殊なファイルを作成します。

```
#!/bin/sh
module="xvd"
mode="664"
major=`awk "\\$2==\"$module\" {print \\$1}" /proc/devices`
# < mknod for as many or few partitions on xvd disk attached to FV guest >
# change/add xvda to xvdb, xvbd, etc. for 2nd, 3rd, etc., disk added in
# in xen config file, respectively.
mknod /dev/xvdb b $major 0
mknod /dev/xvdb1 b $major 1
mknod /dev/xvdb2 b $major 2
chgrp disk /dev/xvd*
chmod $mode /dev/xvd*
```

各追加の仮想ディスクに、マイナー番号を 16 単位で増加します。以下の例では、追加デバイスマイナー番号 16 が作成されています。

```
# mknod /dev/xvdc b $major 16
# mknod /dev/xvdc1 b $major 17
```

これが、以下のコマンドで作成される次のデバイスを 32 にします:

```
# mknod /dev/xvdd b $major 32
# mknod /dev/xvdd1 b $major 33
```

ここで、作成したパーティションが使用できることを確認します。

```
[root@rhel3]# cat /proc/partitions
major   minor     #blocks   name

  3        0      10485760  hda
  3        1        104391  hda1
  3        2      10377990  hda2
202        0         64000  xvdb
202        1         32000  xvdb1
202        2         32000  xvdb2
253        0       8257536  dm-0
253        1       2031616  dm-1
```

上記の出力で、パーティション設定をしたデバイス **“xvdb”** がシステムに利用可能であることが判ります。

以下の手順はゲストに新しいデバイスを追加して、再起動後にそれを永続化します。 全てのこれらのコマンドはゲスト上で実行されるものです。

1. ブロックデバイスイメージをマウントするディレクトリを作成します。

   ```
   [root@rhel3]# mkdir /mnt/pvdisk_p1
   [root@rhel3]# mkdir /mnt/pvdisk_p2
   ```

2. デバイスを新しいフォルダにマウントします。

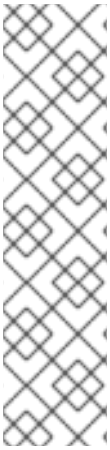
   ```
   [root@rhel3]# mount /dev/xvdb1 /mnt/pvdisk_p1
   [root@rhel3]# mount /dev/xvdb2 /mnt/pvdisk_p2
   ```

3. デバイスが正常にマウントされたことを確認します。

   ```
   [root@rhel3]# df /mnt/pvdisk_p1
   Filesystem           1K-blocks      Used   Available Use%  Mounted
   on
   /dev/xvdb1               32000        15       31985   1%
   /mnt/pvdisk_p1
   ```

4. ブートシーケンス中にデバイスがマウントされるようにゲスト内で **/etc/fstab** ファイルを更新します。以下の行を追加して下さい:

   ```
   /dev/xvdb1   /mnt/pvdisk_p1   ext3    defaults        1 2
   /dev/xvdb2   /mnt/pvdisk_p2   ext3    defaults        1 2
   ```



注記

Red Hat Enterprise Linux 3 ホスト (**dom0**) を使用して 以下に見えるように、**"noapic"** パラメータは仮想ゲストの **/boot/grub/grub.conf** エントリ内のカーネルブート行に追加 される必要があります。ご使用のアーキテクチャとカーネルバージョンは異なるかも 知れないことを忘れないで下さい。

```
kernel /vmlinuz-2.6.9-67.EL ro root=/dev/VolGroup00/rhel4_x86_64
rhgb noapic
```

Red Hat Enterprise Linux 5.2 dom0 はゲスト用にこのカーネルパラメータを 必要としません。

重要

Itanium (ia64) binary RPM パッケージとビルドは現在、使用できません。

14.3.3. Red Hat Enterprise Linux 4 上で Para-virtualized ドライバーのインストールと 設定

このセクションには、Red Hat Enterprise 4 ゲストオペレーティングシステム内での para-virtualized ドライバーに関する詳細案内が含まれています。

注記

これらのパッケージは para-virtualized ディスクからのブートをサポートしていません。 ゲストオペレーティングシステムカーネルのブートには、まだ模倣 IDE ドライバーが 必要です。他の（システム関連ではない）ユーザースペースアプリケーションとデータは para-virtualized ブロックデバイスドライバーを使用することができます。

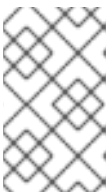
ドライバーのインストール

para-virtualized ドライバーを使用した Red Hat Enterprise Linux 4 ゲストのインストールに必要な手順を以下に示します。

1. 使用しているハードウェアアーキテクチャとカーネル変種用の **kmod-xenpv**、 **modules-init-tools**、及び **modversions** の RPM 群を ゲストオペレーティングシステムにコピーします。

2. **rpm** ユーティリティを使用して RPM パッケージ群を インストールします。使用するゲストオペレーティングシステムの変種とアーキテクチャに必要な パッケージを正しく識別していることを確認して下さい。このパッケージには、更新した module-init-tools が必要となり、これは Red Hat Enterprise Linux 4-6-z カーネル、又はそれ以降で利用できます。

```
[root@rhel4]# rpm -ivh modversions
[root@rhel4]# rpm -Uvh module-init-tools
[root@rhel4]# rpm -ivh kmod-xenpv*
```



注記

UP、SMP、Hugemem、及びアーキテクチャ用に異なるパッケージがあります。 そのため、ご使用のカーネルに適した RPM を持っていることを確認して下さい。

3. **cat /etc/modules.conf** を実行して、以下のような **eth0** 用のエイリアスがあることを 確認します。複数インターフェイス設定を計画している場合は、各インターフェイス毎に 追加の行を加えます。以下のエントリに似ていない場合は変更して下さい。

```
alias eth0 xen-vnif
```

4. 仮想マシンをシャットダウンします（ゲスト内で **“#shutdown -h now”** を使用）。

5. 以下のようにして **/etc/xen/YourGuestsName** 内の ゲスト設定ファイルを編集します:

   - **“vif=”** エントリから **“type=ioemu”** エントリを削除します。
   - 追加のディスクパーティション、ボリューム、又は LUN をゲストに追加して para-virtualized **(xen-vbd)** ゲストドライバー経由で それらにアクセスできるようにします。
   - 各追加の物理デバイス、LUN、パーティション、又はボリューム用に、以下に示した ようなエントリを、ゲスト設定ファイル内の **“disk=”** セクションに追加します。オリジナルの **“disk=”** エントリも 以下のエントリに似ているかも知れません。

     ```
     disk = [ "file:/var/lib/libvirt/images/rhel4_64_fv.dsk,hda,w"]
     ```

   - 追加の物理デバイス、LUN、パーティション、又はボリュームを追加すると、 XML 設定ファイル内の para-virtualized ドライバーのエントリは以下にある エントリに似たものになります。

     ```
     disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w",
     "tap:aio:/var/lib/libvirt/images/UserStorage.dsk,xvda,w" ]
     ```

     注記

     ファイルベースのイメージが使用されている場合、para-virtualized デバイスに **“tap:aio”** を使用します。

6. **virsh** コマンドを使用して仮想マシンをブートします:

```
# virsh start YourGuestName
```

仮想ゲストの最初の再起動で、**kudzu** がユーザーに **"Realtek ネットワークデバイスを維持するか、削除する"**、及び **"xen-bridge デバイスを設定する"** の選択を尋ねてきます。 **xen-bridge** は設定して、Realtek ネットワークデバイスは 削除すべきです。

注記

Red Hat Enterprise Linux 4 ホスト (**dom0**) を使用して 以下に見えるように、**"noapic"** パラメータは仮想ゲストの **/boot/grub/grub.conf** エントリ内のカーネルブート行に追加 される必要があります。ご使用のアーキテクチャとカーネルバージョンは異なるかも 知れないことを忘れないで下さい。

```
kernel /vmlinuz-2.6.9-67.EL ro root=/dev/VolGroup00/rhel4_x86_64
rhgb noapic
```

Red Hat Enterprise Linux 5.2 dom0 はゲストの為にこのカーネルパラメータを 必要としません。

ここで、作成したパーティションが使用できることを確認します。

```
[root@rhel4]# cat /proc/partitions
major    minor   #blocks   name

   3        0    10485760  hda
   3        1      104391  hda1
   3        2    10377990  hda2
 202        0       64000  xvdb
 202        1       32000  xvdb1
 202        2       32000  xvdb2
 253        0     8257536  dm-0
 253        1     2031616  dm-1
```

上記の出力で、パーティション設定をしたデバイス **“xvdb”** がシステムに利用可能であることが判ります。

以下の手順は新しいデバイスをゲストに追加して、再起動後にそれを永続化します。 全てのこれらのコマンドはゲスト上で実行されるものです。

1. ブロックデバイスイメージをマウントするディレクトリを作成します。

   ```
   [root@rhel4]# mkdir /mnt/pvdisk_p1
   [root@rhel4]# mkdir /mnt/pvdisk_p2
   ```

2. デバイスを新しいフォルダにマウントします。

   ```
   [root@rhel4]# mount /dev/xvdb1 /mnt/pvdisk_p1
   [root@rhel4]# mount /dev/xvdb2 /mnt/pvdisk_p2
   ```

3. デバイスが正常にマウントされたことを確認します。

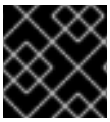
   ```
   [root@rhel4]# df /mnt/pvdisk_p1
   Filesystem           1K-blocks      Used   Available Use%  Mounted
   on
   /dev/xvdb1               32000        15       31985   1%
   /mnt/pvdisk_p1
   ```

4. ブートシーケンス中にデバイスがマウントされるようにゲスト内で **/etc/fstab** ファイルを更新します。以下の行を追加して下さい:

```
/dev/xvdb1   /mnt/pvdisk_p1   ext3    defaults        1 2
/dev/xvdb2   /mnt/pvdisk_p2   ext3    defaults        1 2
```

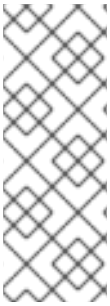
注記

このパッケージは Red Hat Enterprise Linux 4 update 2 のシステムとカーネルを 通じた Red Hat Enterprise Linux 4-GA 用にはサポートがありません。



重要

IA64 binary RPM パッケージとビルドは現在、使用できません。



注記

xen-vbd ドライバーは自動的にロードしない可能性が あります。以下のコマンドで **%release** を para-virtualized ドライバー の正しいリリースバージョンで入れ替えて、ゲスト上で実行して 下さい。

```
# insmod /lib/modules/'uname -r'/weak-updates/xenpv/%release/xen-vbd.ko
```

14.3.4. Red Hat Enterprise Linux 5 上で Para-virtualized ドライバーのインストールと 設定

このセクションには、Red Hat Enterprise 5 ゲストオペレーティングシステム内での para-virtualized ドライバーに関する詳細案内が含まれています。

注記

これらのパッケージは para-virtualized ディスクからのブートをサポートしていません。 ゲストオペレーティングシステムカーネルのブートには、まだ模倣 IDE ドライバーが 必要です。他の（システム関連ではない）ユーザースペースアプリケーションとデータは para-virtualized ブロックデバイスドライバーを使用することができます。

ドライバーのインストール

para-virtualized ドライバーを使用した Red Hat Enterprise Linux 5 ゲストのインストールに必要な手順を以下に示します。

1. 使用しているハードウェアアーキテクチャとカーネル変種用の **kmod-xenpv** パッケージを ゲストオペレーティングシステムにコピーします。
2. **yum** コマンドを使用してパッケージをインストールします。

```
[root@rhel5]# yum install kmod-xenpv*
```

3. 以下のコマンドを使用して、ゲストオペレーティングシステム内の自動ハードウェア 検出を無効にします。

```
[root@rhel5]# chkconfig kudzu off
```

4. **cat /etc/modules.conf** を実行することにより、以下のような**eth0** 用のエイリアスがあ

ることを 確認します。複数のインターフェイスを設定する予定の場合は、各インターフェイス毎に追加の 行を加えます。以下のエントリのように見えない場合は、変更して下さい。

```
alias eth0 xen-vnif
```

5. 仮想マシンをシャットダウンします（ゲスト内で **“#shutdown -h now”** を使用）。

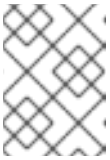
6. 以下のようにして、**/etc/xen/<Your GuestsName>** 内のゲスト設定ファイルを編集します。

   - **“vif=”** エントリから **“type=ioemu”** エントリを削除します。
   - 追加のディスクパーティション、ボリューム、あるいは LUN をゲストに追加して、 para-virtualized (**xen-vbd**) ディスクドライバー経由で それらにアクセスできるようにします。
   - 各追加の物理デバイス、LUN、パーティション、又はボリューム用に、以下に示した ようなエントリを、ゲスト設定ファイル内の **“disk=”** セクションに追加します。オリジナルの **“disk=”** エントリも 以下のエントリに似ているかも知れません。

     ```
     disk = [ "file:/var/lib/libvirt/images/rhel4_64_fv.dsk,hda,w"]
     ```

   - 追加の物理デバイス、LUN、パーティション、又はボリュームを追加すると、 XML 設定ファイル内の para-virtualized ドライバーのエントリは以下にある エントリに似たものになります。

     ```
     disk = [ "file:/var/lib/libvirt/images/rhel3_64_fv.dsk,hda,w",
     "tap:aio:/var/lib/libvirt/images/UserStorage.dsk,xvda,w" ]
     ```

     注記

     ファイルベースのイメージが使用されている場合は、para-virtualized デバイス用に **“tap:aio”** を使用します。

7. **virsh** コマンドを使用して仮想マシンをブートします:

   ```
   # virsh start YourGuestName
   ```

para-virtualized ドライバーをインストールした後に、ネットワークインターフェイスが 立ち上がっていることを確認するには、ゲスト上で以下のコマンドを発行します。 割り当て済みの IP アドレスを含んだインターフェイス情報が表示されるはずです。

```
[root@rhel5]# ifconfig eth0
```

ここで、作成したパーティションが利用可能であることを確認します。

```
[root@rhel5]# cat /proc/partitions
major minor  #blocks    name
   3     0   10485760   hda
   3     1     104391   hda1
   3     2   10377990   hda2
 202     0      64000   xvdb
 202     1      32000   xvdb1
```

```
202     2      32000   xvdb2
253     0    8257536   dm-0
253     1    2031616   dm-1
```

上記の出力では、パーティション設定したデバイス **“xvdb”** がシステムに利用可能であることが判ります。

以下の手順は、新しいデバイスをゲストに追加して、それを再起動後に永続化します。 全てのこれらのコマンドはゲスト上で実行します。

1. ディレクトリを作成してブロックデバイスイメージのマウント用とします。

```
[root@rhel5]# mkdir /mnt/pvdisk_p1
[root@rhel5]# mkdir /mnt/pvdisk_p2
```

2. デバイスを新しいフォルダにマウントします。

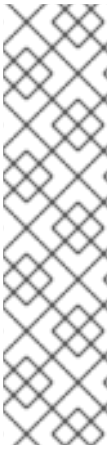
```
[root@rhel5]# mount /dev/xvdb1 /mnt/pvdisk_p1
[root@rhel5]# mount /dev/xvdb2 /mnt/pvdisk_p2
```

3. デバイスが正常にマウントされたことを確認します。

```
[root@rhel5]# df /mnt/pvdisk_p1
Filesystem           1K-blocks      Used   Available Use%  Mounted
on
/dev/xvdb1               32000        15       31985   1%
/mnt/pvdisk_p1
```

4. ゲスト内で **/etc/fstab** ファイルを更新して、ブートシーケンス中に デバイスをマウントするようにします。以下の行を追加して下さい:

```
/dev/xvdb1   /mnt/pvdisk_p1   ext3    defaults        1 2
/dev/xvdb2   /mnt/pvdisk_p2   ext3    defaults        1 2
```



注記

Red Hat Enterprise Linux 5.1 ホスト **(dom0)** を使用して 以下に見えるように、**"noapic"** パラメータは仮想ゲストの **/boot/grub/grub.conf** エントリ内のカーネルブート行に追加 される必要があります。ご使用のアーキテクチャとカーネルバージョンは異なるかも 知れないことを忘れないで下さい。

kernel /vmlinuz-2.6.9-67.EL ro root=/dev/VolGroup00/rhel4_x86_64 rhgb noapic

Red Hat Enterprise Linux 5.2 dom0 は、ゲスト用にこのカーネルパラメータを 必要としません。

14.4. PARA-VIRTUALIZED ネットワークドライバーの設定

para-virtualized ネットワークドライバーがロードされると、ゲストネットワークの インターフェイスを再設定して、ドライバーと仮想イーサネットカードの変更を反映させる 必要があるかも知れません。

ゲスト内でネットワークインターフェイスの再設定をするには以下の手順に従います。

1. **virt-manager** 内で、ゲスト用のコンソールウィンドウを開いて、**root** としてログインします。

2. Red Hat Enterprise Linux 4 では、ファイル **/etc/modprobe.conf** が **"alias eth0 xen-vnif"** を含んでいることを 確認します。

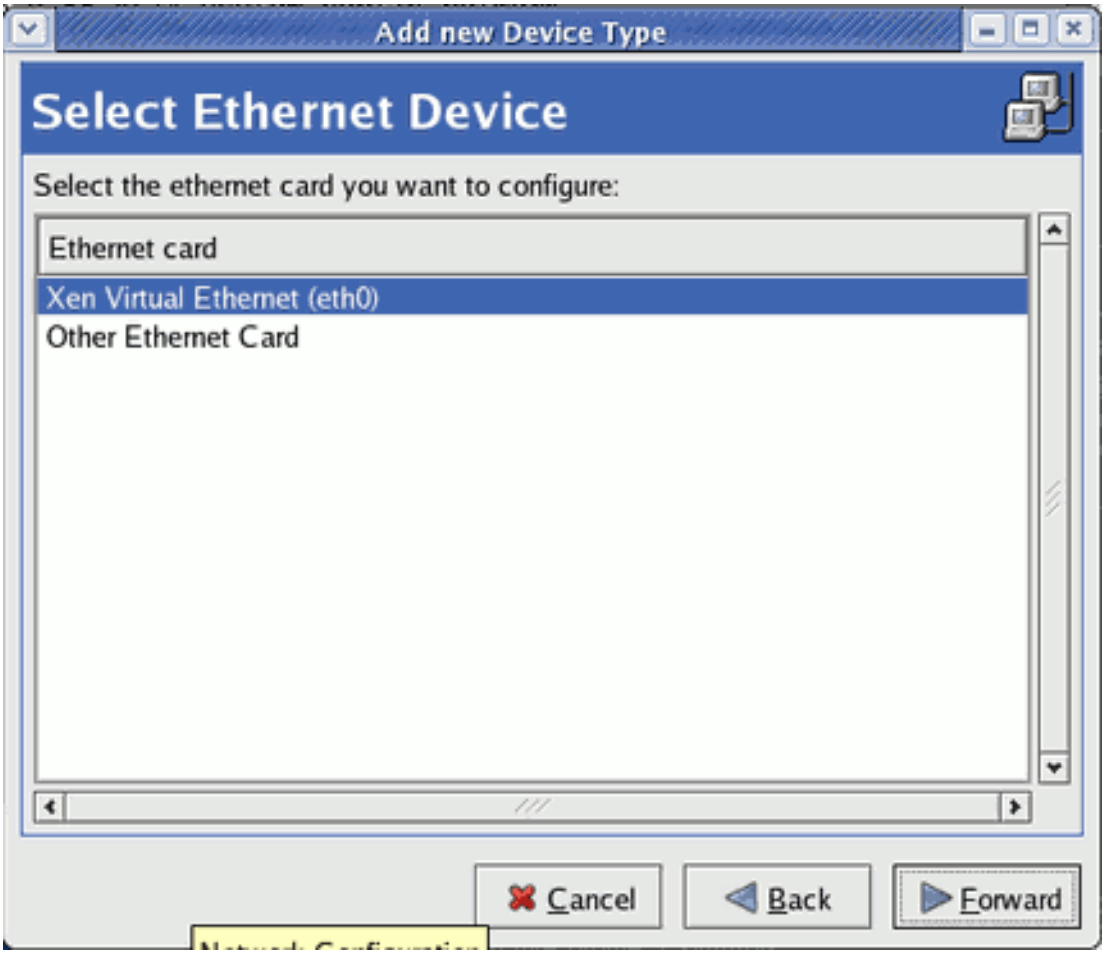
```
# cat /etc/modprobe.conf
alias eth0 xen-vnif
```

3. **eth0** の現在の設定を表示するには、**"# ifconfig eth0"** を 実行します。デバイスが存在しないことを示すエラーが出た場合は、「para-virtualized ドライバーを手動でロードする」に示してある手順で、手動でモジュールをロードする必要があります。

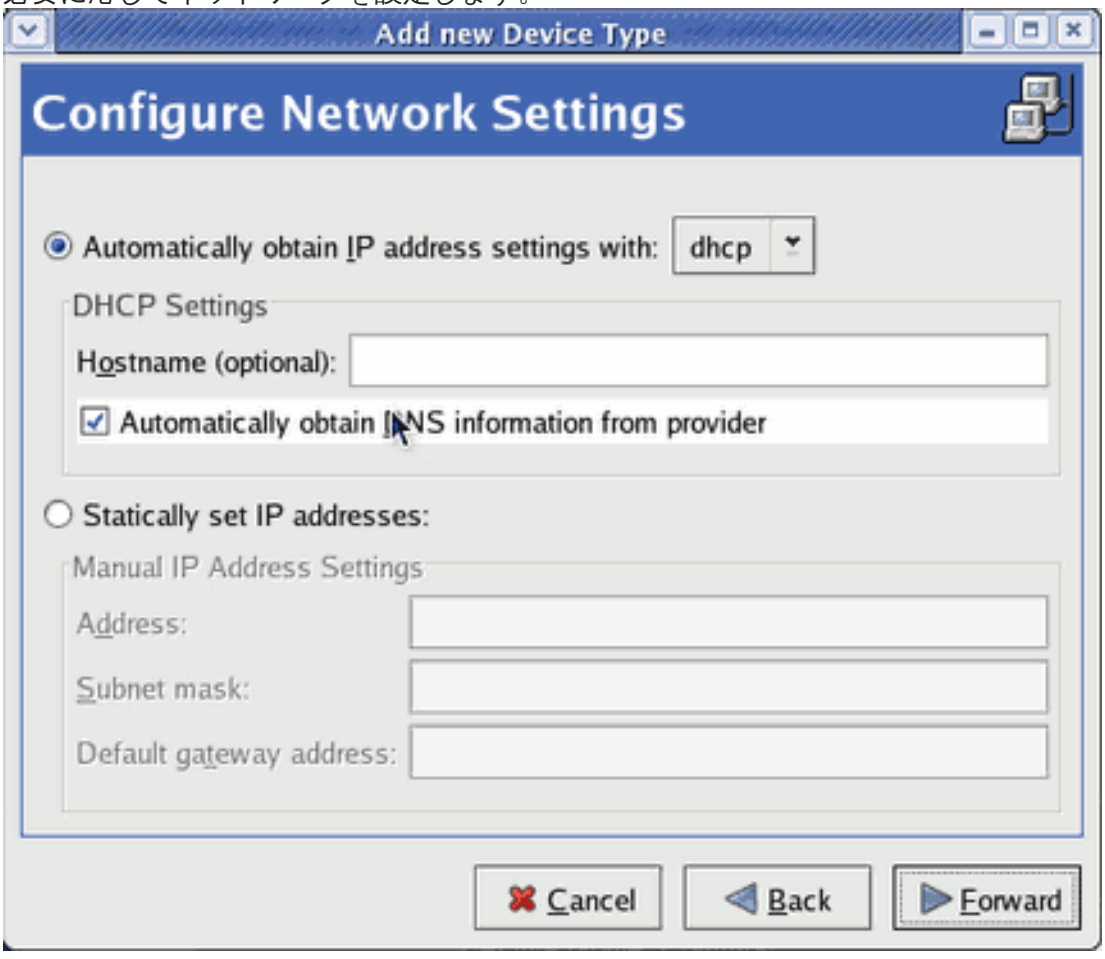
```
ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:00:00:6A:27:3A
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:630150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109336431 (104.2 MiB)  TX bytes:846 (846.0 b)
```

4. コマンド、**"# system-config-network"** を使用して ネットワーク設定ユーティリティ (NetworkManager) を開始します。**"「進む」（Forward）"** ボタンをクリックしてネットワーク 設定を開始します。

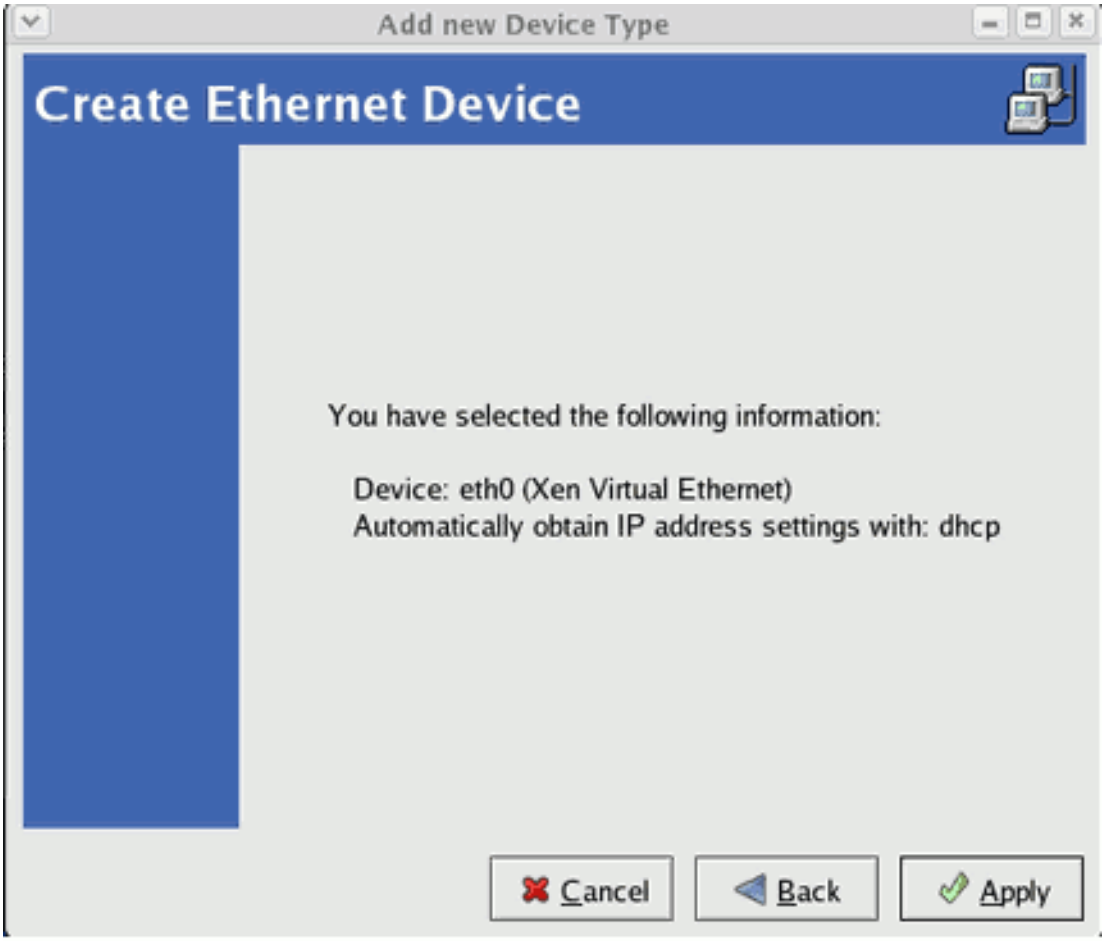
5. **「Xen 仮想イーサネットカード (eth0) の作成」（Xen Virtual Ethernet Card (eth0)）** エントリを選択します。そして、**「進む」（Forward）** を クリックします。



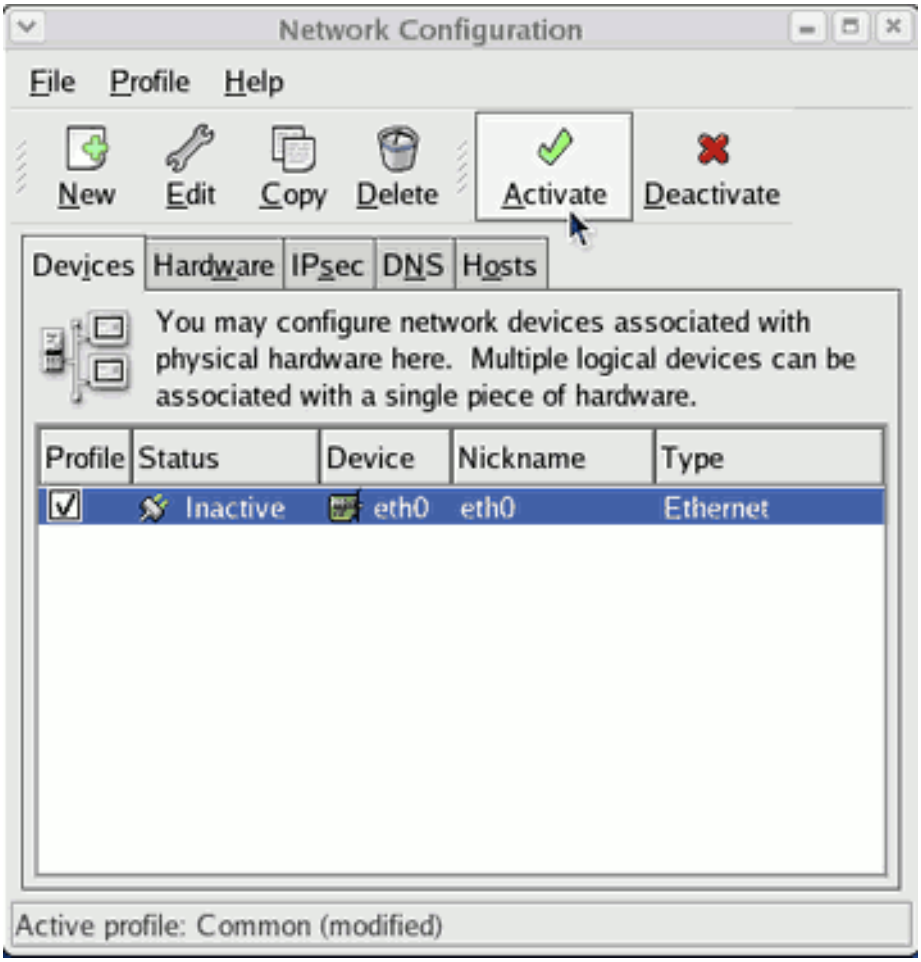
必要に応じてネットワークを設定します。



6. 「適用」（Apply）ボタンを押して設定を完了します。



7. 「アクティベート」（Activate）ボタンを押して、新しい設定を適用し、ネットワークを再開始します。



8. そうすると、IP アドレスが割り当てられた新規のネットワークインターフェイスが 見えるはずです。

```
ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:16:3E:49:E4:E0
          inet addr:192.168.78.180  Bcast:192.168.79.255
Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:630150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:501209 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109336431 (104.2 MiB)  TX bytes:46265452 (44.1
MiB)
```

14.5. 追加の PARA-VIRTUALIZED ハードウェア設定

このセクションでは、追加の仮想ネットワーク、又はストレージをゲストオペレーティング システムに追加する方法を説明しています。Red Hat Enterprise Linux 5 Virtualization 上での ネットワークとストレージリソースの設定に関する詳細には、Emerging Technologies, Red Hat.com から入手できる ドキュメントをお読み下さい。

14.5.1. 仮想化ネットワークインターフェイス

以下の手順を実行して、使用するゲスト用の追加のネットワークデバイスを設定します。

/etc/xen/YourGuestName 内の **YourGuestName** を 使用するゲスト名で入れ替えることで、ゲスト設定ファイルを編集します。

オリジナルエントリは以下のように見えるでしょう。

```
vif = [ "mac=00:16:3e:2e:c5:a9,bridge=xenbr0" ]
```

追加エントリを設定ファイルの **"vif="** セクションに追加します。 以下に似たものになります。

```
vif = [ "mac=00:16:3e:2e:c5:a9,bridge=xenbr0",
    "mac=00:16:3e:2f:d5:a9,bridge=xenbr0" ]
```

新規のインターフェイス用には独自の MAC アドレスを生成するようにします。以下のような コマンドを使用できます。

```
# echo 'import virtinst.util ; print virtinst.util.randomMAC()' | python
```

ゲストが再起動すると、ゲストオペレーティングシステムの中で以下の手順を実行します。 Red Hat Enterprise Linux 3 内の **/etc/modules.conf** か、あるいは Red Hat Enterprise Linux 4 及び Red Hat Enterprise Linux 5 では **/etc/modprobe.conf** に更新が追加されていることを確認します。追加した新規のインターフェイス用に新しいエイリアスを加えます。

```
alias eth1 xen-vnif
```

その後、追加した各新規インターフェイスをテストして、ゲスト内で利用可能な ことを確認します。

```
# ifconfig eth1
```

上記のコマンドは **eth1** のプロパティを表示します。 3つめのインターフェイスを追加した場合には、**eth2** 用に このコマンドを繰り返します。

ここで、Red Hat Enterprise Linux3 上では **redhat-config-network** を使用して、又は 、 Red Hat Enterprise Linux 4 と Red Hat Enterprise Linux 5 上では **system-config-network** を使用してネットワークインターフェイスを設定します。

14.5.2. 仮想ストレージデバイス

以下の手順を実行して、ゲスト用の追加の仮想ストレージデバイスを 設定します。

/etc/xen/YourGuestName で **YourGuestName** を 使用するゲスト名で入れ替えることで、ゲストの設定ファイルを編集します。オリジナルのエントリは 以下に似ているでしょう。

```
disk = [ "file:/var/lib/libvirt/images/rhel5_64_fv.dsk,hda,w"]
```

ここで、新規の物理デバイス、LUN、パーティション、又はボリューム用の追加のエントリを 設定ファイル内の **"disk="** パラメータに加えます。para-virtualized ドライバーを使用するストレージエントリは以下のエントリに似ています。**"tap:aio"** パラメータは hypervisor に対して para-virtualized ドライバーを使用するように指示します。

```
disk = [ "file:/var/lib/libvirt/images/rhel5_64_fv.dsk,hda,w",
    "tap:aio:/var/lib/libvirt/images/UserStorage1.dsk,xvda,w" ]
```

他のエントリを追加したい場合は、それらをコンマ分離の一覧として、**"disk="** セクションに追加します。

注記

xvd デバイス用の文字を進展する必要があります。これは、 2つめのストレージエンティティの為には、**xvda**ではなく、 **xvdb** となります。

```
disk = [ "file:/var/lib/libvirt/images/rhel5_64_fv.dsk,hda,w",
    "tap:aio:/var/lib/libvirt/images/UserStorage1.dsk,xvda,w",
    "tap:aio:/var/lib/libvirt/images/UserStorage2.dsk,xvdb,w" ]
```

パーティションが作成されて利用可能であることを確認します。

```
# cat /proc/partitions
major minor  #blocks    name
   3     0   10485760   hda
   3     1     104391   hda1
   3     2   10377990   hda2
 202     0      64000   xvda
 202     1      64000   xvdb
 253     0    8257536   dm-0
 253     1    2031616   dm-1
```

上記の出力では、パーティション、又はデバイス、**“xvdb”** はシステムに利用可能です。

新規のデバイスとディスクをローカルマウントポイントにマウントして、ゲストの内の **/etc/fstab** を更新することにより、 ブート時にデバイスとパーティションがマウントされます。

```
# mkdir /mnt/pvdisk_xvda
# mkdir /mnt/pvdisk_xvdb
# mount /dev/xvda /mnt/pvdisk_xvda
# mount /dev/xvdb /mnt/pvdisk_xvdb
# df /mnt
Filesystem           1K-blocks      Used   Available Use%  Mounted on
/dev/xvda                64000        15       63985   1%
/mnt/pvdisk_xvda
/dev/xvdb                64000        15       63985   1%
/mnt/pvdisk_xvdb
```

第15章 KVM PARA-VIRTUALIZED ドライバー

Para-virtualized ドライバーは、KVM ホストを稼働している 仮想化した Windows ゲストで使用できます。これらの Para-virtualized ドライバーは virtio パッケージに収納されており、virtio パッケージはブロック（ストレージ） デバイスとネットワークインターフェイスコントローラをサポートします。

Para-virtualized ドライバーは完全仮想化ゲストのパフォーマンスを強化します。 Para-virtualized ドライバーを使用すると、ゲスト I/O 遅延は減少し、スループットは ベアメタルレベル近くまで向上します。I/O が重いタスクやアプリケーションを実行している 完全仮想化ゲストには、Para-virtualized ドライバーの使用が推奨されます。

KVM para-virtualized ドライバーは Red Hat Enterprise Linux 4.8 とそれ以降、及び、 Red Hat Enterprise Linux 5.3 とそれ以降のバージョンで自動的にロードされてインストール されます。これらの Red Hat Enterprise Linux のバージョンはドライバーを検出して インストールしますので、追加のインストール手順は無用となります。

KVM モジュールの場合と同様に、virtio ドライバーは Red Hat Enterprise Linux 5.4 と それ以降を稼働しているホストでのみ使用できます。

重要

ゲスト毎の追加のデバイスとして 28 だけの PCI スロットが使用できます。 全ての para-virtualized ネットワーク、又はブロックデバイスはスロットを 1 つ使用します。各ゲストは para-virtualized ネットワーク、para-virtualized ディスクデバイス、あるいは、VT-d を使用した他の PCI デバイスのいずれかの組み合わせからなる 最大 28 までの追加のデバイスを使用できます。

以下の Microsoft Windows のバージョンは KVM para-virtualized ドライバーを サポートしています:

- Windows XP,
- Windows Server 2003,
- Windows Vista,
- Windows Server 2008.

Windows XP は virtio para-virtualized ブロックドライバーの実行には サポートされていません。virtio ネットワークドライバーのみが Windows XP で サポートされています。

15.1. KVM WINDOWS PARA-VIRTUALIZED ドライバーのインストール

このセクションでは、KVM Windows para-virtualized ドライバーのインストール プロセスを取り扱います。KVM para-virtualized ドライバーは Windows のインストール中に、 ロードするか 又はゲストのインストール後にインストールすることができます。

以下の方法のいずれかにより、使用するゲストに para-virtualized ドライバーを インストールすることができます:

- ネットワーク上の インストールファイルをゲストにアクセス可能にする。
- ドライバーインストールディスクの .iso ファイルの 仮想化 CD-ROM デバイスを 使用する、あるいは、

- 仮想化したフロッピィデバイスを 使用して、ブート時にドライバーをインストールする (Windows ゲスト用)。

このガイドは仮想化した CD-ROM デバイスと 同じく、para-virtualized インストーラディスクからのインストールを説明します。

1. **ドライバーをダウンロード**
   yum コマンドを使用して、virtio-win パッケージをダウンロードします（Red Hat Enterprise Linux Supplementary チャンネル内）。

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

   ドライバーは Red Hat Enterprise Linux Supplementary ディスク、又は、Microsoft (windowsservercatalog.com) から入手することが出来ます。Red Hat Enterprise Virtualization Hypervisor 5.4 と Red Hat Enterprise Linux 5.4 は同じコードベースを基にしていることに注意して下さい。

   virtio-win パッケージは CD-ROM イメージ（**virtio-win.iso** ファイル）を **/usr/share/virtio-win/** ディレクトリ内にインストールします。

2. **para-virtualized ドライバーのインストール**
   デバイスを添付や修正して para-virtualized ドライバーを使用する前に、 ゲスト上にドライバーをインストールすることが推奨されます。

   root ファイルシステムを格納しているブロックデバイス、又はゲストのブートに必要となる 他のブロックデバイスには、ドライバーはデバイスが修正される前にインストールされなければなりません。ゲスト上にドライバーがインストールされていなくて、ドライバーが virtio ドライバーに セットされている場合は、ゲストはブートしません。

virt-manager でイメージをマウントする

手順15.1「Windows ゲスト用に **virt-manager** を使用して CD-ROM イメージを マウント」に従って、 **virt-manager** を使用して、CD-ROM イメージを追加します。

手順15.1 Windows ゲスト用に virt-manager を使用して CD-ROM イメージを マウント

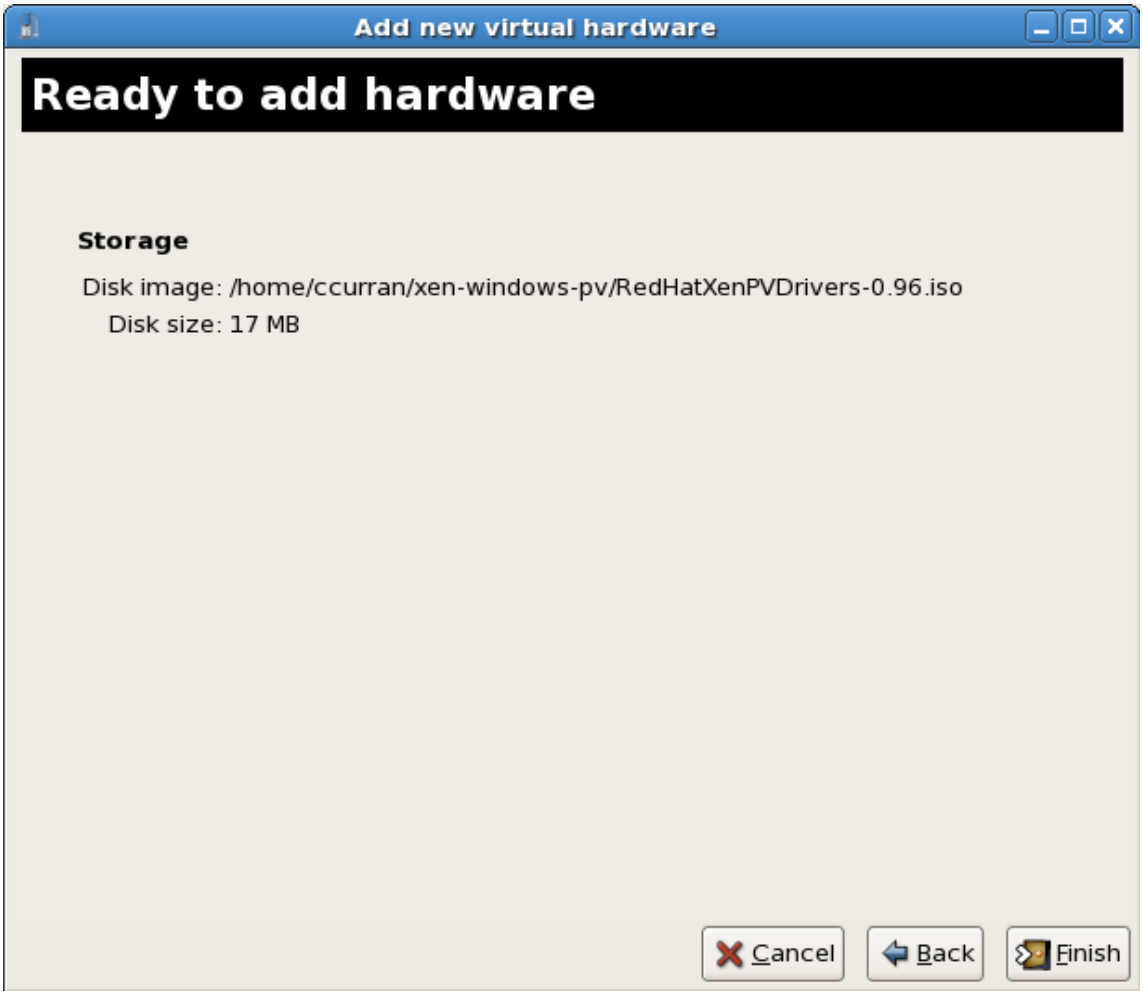
1. **virt-manager** を開いて、仮想マシンのリストから 使用する仮想化ゲストを選択して、**詳細** ボタンを クリックします。
2. **詳細** パネル内の **追加** ボタンをクリック します。
3. これが、新規デバイス追加のウィザードを開きます。ドロップダウンメニューから **ストレージデバイス** を選択して、それから **進む** をクリックします。

4. **ファイル（ディスクイメージ）** オプションを選択して、 para-virtualized ドライバーの .iso ファイルの場所をセットします。**yum** を使用して para-virtualized ドライバーパッケージをインストールした場合、 .iso ファイルは **/usr/share/xenpv-win** です。

   ドライバーが物理 CD 内に格納されている場合は、**通常のディスクパーティション** オプションを使用します。

   デバイスタイプ を **IDE cdrom** にセットして **進む** をクリックして継続します。

5. ゲストが開始されると、ディスクが割り当てられてゲスト用に使用可能になります。 **完了** をクリックして、ウィザードを閉じるか、又は 間違えた場合は戻ります。



仮想化フロッピィディスクによるインストール

この手続きは Windows のインストール時に para-virtualized ドライバーの インストールをします。

- 一度限りのメニューを使用して Windows VM を初めてインストールしたら、 **viostor.vfd** をフロッピィとして添付します。

  a. **Windows Server 2003**
  Windows がサードパーティドライバー用に F6 を押すように催促したら、 それを押して画面上の指示に従います。

  b. **Windows Server 2008**
  インストーラがドライバーを催促した時には、 **Load Driver** をクリックして、 インストーラをドライブ A: にポイントし、使用中のゲスト OS と アーキテクチャに適合する ドライバーを選択します。

既存のデバイスに KVM para-virtualized ドライバーを使用する

仮想化した IDE ドライバーの代わりに **virtio** ドライバーを 使用するためにゲストに付けられた既存のハードディスクドライバーを修正します。 この例は、libvirt 設定ファイルを編集します。別の方法として、**virt-manager**、**virsh attach-disk**、あるいは、**virsh attach-interface** が、para-virtualized ドライバーを使用する新規デバイスを追加 できます。新規デバイス用に KVM para-virtualized ドライバーを使用する

1. 仮想化した 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を以下に示します。これは、para-virtualized ドライバーを使用しない仮想化ゲストの標準的なエントリです。

```
<disk type='file' device='disk'>
```

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

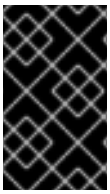
2. **bus=** エントリを **virtio** に 変更することにより、para-virtualized デバイスを使用するためにエントリを変更します。

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

新規デバイス用に KVM para-virtualized ドライバーを使用する

この手続きは、**virt-manager** で、KVM para-virtualized ドライバーを使用した新規デバイスの作成を示しています。

別の方法として、 para-virtualized ドライバーを使用するデバイスを添付するために **virsh attach-disk** か **virsh attach-interface** のコマンドが使用できます。

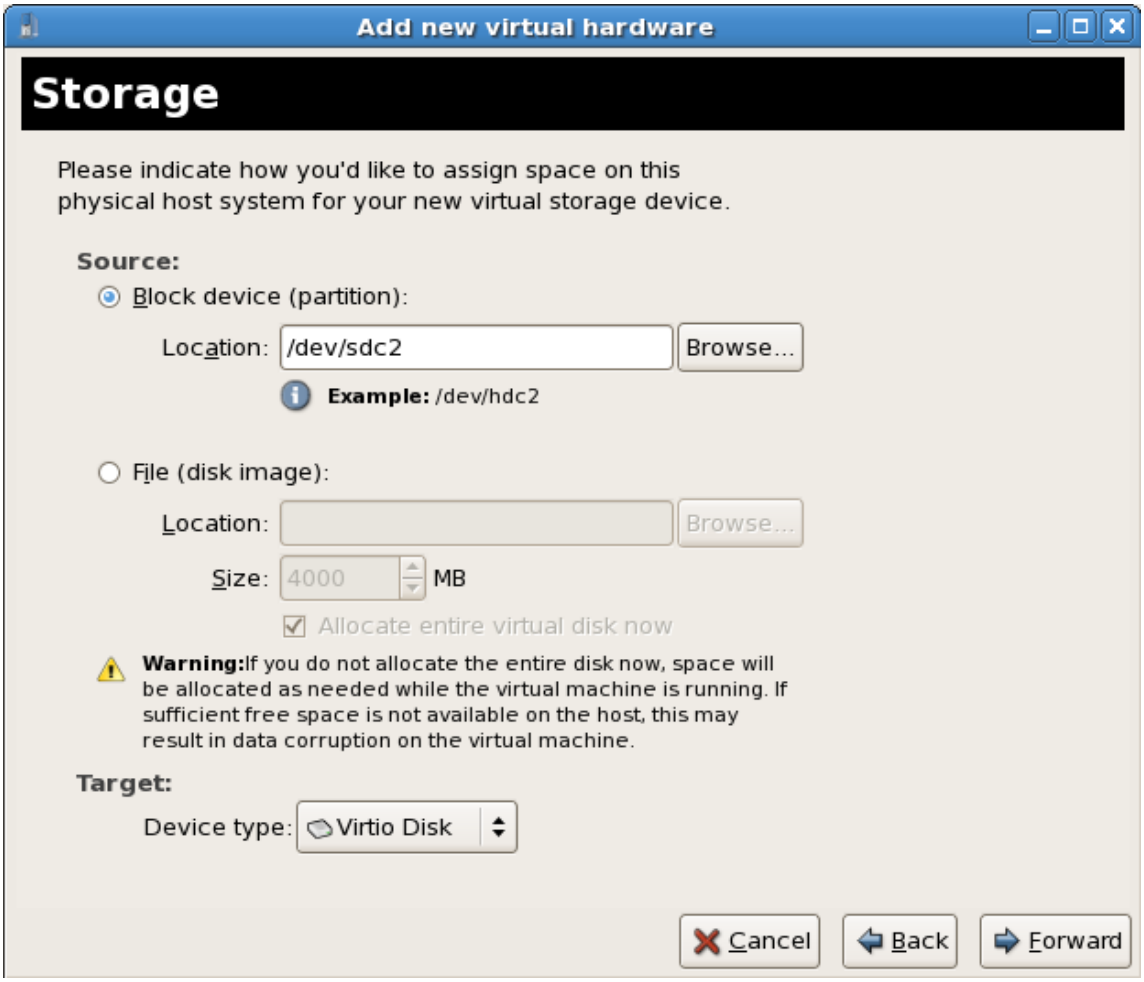


重要

新規のデバイスをインストールする前に、Windows ゲスト上にドライバーが インストールされていることを確認します。ドライバーが利用できない場合は、 デバイスは認識されず機能しません。

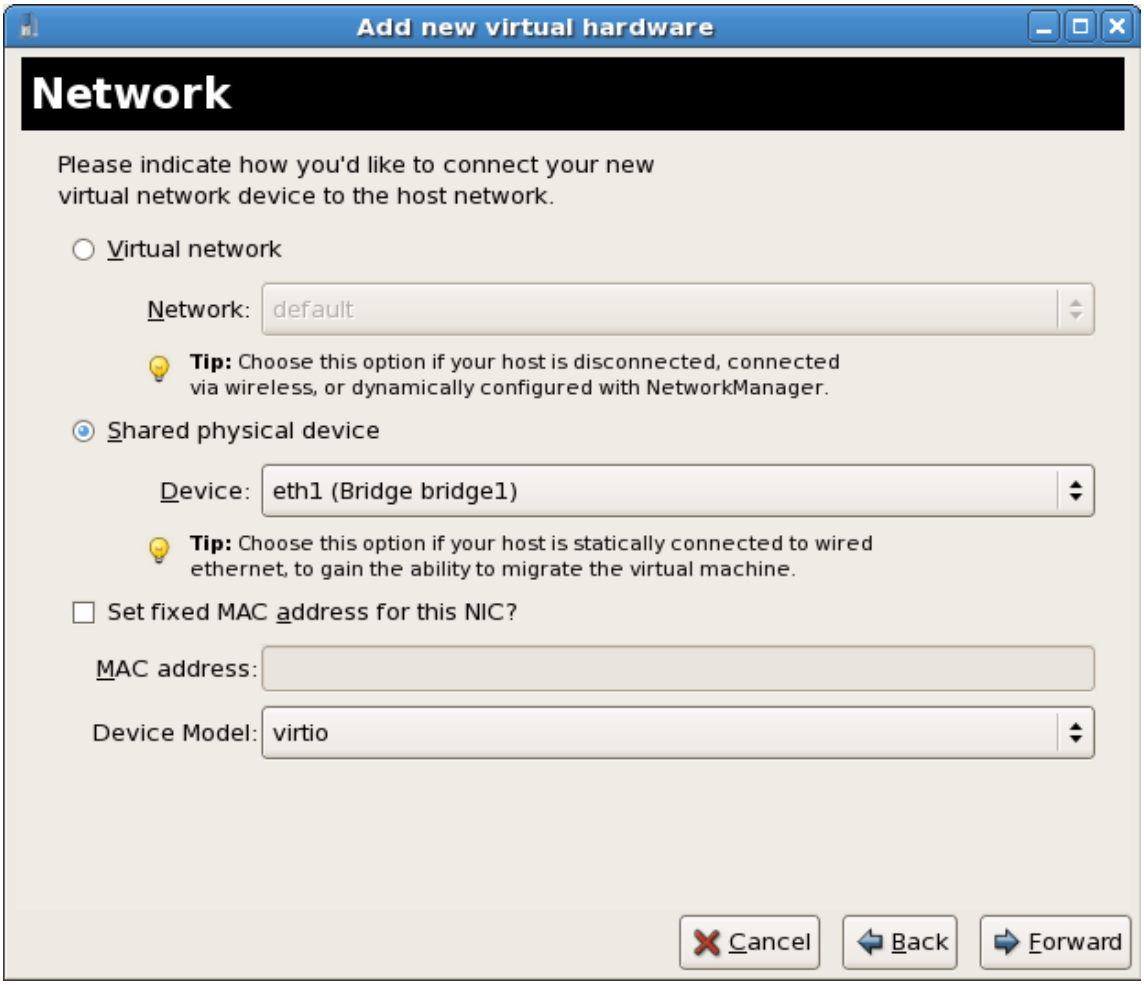
1. **virt-manager** 内のゲスト名をダブルクリックして 仮想化ゲストを開きます。
2. **ハードウェア** タブを開きます。
3. **ハードウェアを追加** をクリックします。
4. 仮想ハードウェアの追加タブで、デバイスのタイプに **ストレージ** 又は **ネットワーク** を選択します。
    1. **新規のディスクデバイス**

       ストレージデバイスか、又はファイルベースイメージを選択します。**Virtio Disk** を **デバイスタイプ** として選び、**進む** をクリックします。



2. **新規のネットワークデバイス**

   仮想ネットワーク 又は **共有物理デバイス** を 選択します。**デバイスタイプ** として **virtio** を 選択して、**進む** をクリックします。



5. **完了** をクリックしてデバイスを保存します。

6. ゲストをリブートします。デバイスは Windows ゲストが再スタートするまで 認識されないかも知れません。

パート IV. 管理

仮想化システムの管理

この章には、Red Hat Enterprise Linux に収納されているツールを使用した ホストと仮想化ゲストの管理に関する情報が含まれています。

第16章 XEND を使用したゲストの管理

xend ノード制御デーモンは、仮想マシンに関連した特定のシステム管理機能を果たします。このデーモンが仮想化リソースを制御するため、**xend** は 仮想マシンと相互交流するように稼動していなければなりません。**xend** を開始する前に、**xend** 設定ファイルである、**/etc/xen/xend-config.sxp** を編集することにより、操作パラメータを 指定する必要があります。**xend-config.sxp** 設定ファイルで有効、又は無効にできるパラメータを以下に示します:

表16.1 xend の設定パラメータ

項目	説明
(console-limit)	コンソールサーバーのメモリーバッファの限度を決定し、その限度をドメイン単位ベースで割り当てます。
(min-mem)	domain0 用に予約される最小限のメガバイト数を決定します(0 を入力すると値は変化しません)。
(dom0-cpus)	domain0 で使用する CPU の数量を決定します(デフォルトで、最低１つの CPU が割り当てられます)
(enable-dump)	これが有効になっている場合にクラッシュが発生すると、Xen はダンプファイルを 作成します（デフォルトは 0）。
(external-migration-tool)	外部デバイスの移行を処理するスクリプト、又はアプリケーションを決定します。スクリプトは **etc/xen/scripts/external-device-migrate** ディレクトリにある必要があります。
(logfile)	ログファイルの場所を決定します (デフォルトでは、**/var/log/xend.log**)。
(loglevel)	ログモードの値をフィルターにかけます: DEBUG, INFO, WARNING, ERROR, CRITICAL(デフォルトは DEBUG)。
(network-script)	ネットワーク環境を有効にするスクリプトを決定します。スクリプトは **etc/xen/scripts** ディレクトリに存在する必要があります。
(xend-http-server)	http ストリームパケット管理サーバーを有効にします (デフォルトは 「no」)。
(xend-unix-server)	UNIX ドメインソケットサーバーを有効にします。ソケットサーバーは、低レベルのネットワーク接続を処理し、来信の接続を受理したり拒否したりする通信エンドポイントです。デフォルト値は 「yes」にセットされています。
(xend-relocation-server)	相互マシン移行用に移転サーバーを有効にします (デフォルトは 「no」)。

項目	説明
(xend-unix-path)	**xend-unix-server** コマンドがデータを出力する場所を決定します (デフォルトでは、**var/lib/xend/xend-socket**)。
(xend-port)	http 管理サーバーが使用するポートを決定します (デフォルトは 8000)。
(xend-relocation-port)	移転サーバーが使用するポートを決定します (デフォルトは 8002)。
(xend-relocation-address)	移行が許可されるホストアドレスを決定します。デフォルト値は、 **xend-address** の値です。
(xend-address)	ドメインソケットサーバーのバインド先のアドレスを決定します。デフォルト値は 全ての接続を許可します。

これらの操作パラメータを設定した後には、**xend** が稼動していることを確認しなければなりません。稼動していない場合、デーモンを開始します。コマンドプロンプトで、次の入力をして **xend** デーモンを開始することができます:

```
service xend start
```

xend を使ってデーモンを停止することができます:

```
service xend stop
```

これがデーモンの動作を停止します。

xend を使ってデーモンを再開始することができます:で

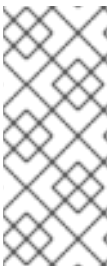
```
service xend restart
```

デーモンはもう一度開始します。

xend デーモンのステータスをチェックすることができます。

```
service xend status
```

出力はデーモンのステータスを表示します。



注記

chkconfig コマンドを使用して、**xend** を **initscript** に追加します。

chkconfig --level 345 xend

これで、**xend** は、ランレベル 3、4、及び 5 で開始します。

第17章 KVM ゲストのタイミング管理

仮想化はゲストの時刻維持に対して諸々の懸念を与えます。一部の CPU がこの 不変タイムスタンプ カウンタを持っていないため、TSC (Time Stamp Counter) をクロックのソースとして使用する ゲストは時刻維持問題に陥ります。正確な時刻維持を持たないゲストは実際の時刻よりも進んだり 遅れたりして同期から外れるため、一部のネットワークアプリケーションに深刻な影響を与えます。

KVM はゲストに para-virtualized クロックを装備することでこの問題を回避します。 別の方法として、一部のゲストはこれらのオペレーティングシステムの将来のバージョンで その時刻維持のために他の x86 クロックソースを使用する可能性があります。

今のところ、Red Hat Enterprise Linux 5.4 及びそれ以降のゲストのみが全面的に para-virtualized のクロックをサポートしています。

ゲストは、不正確なクロックとカウンタの原因で数種の問題を持つようになります:

- クロックは正確な時刻との同期から外れて、セッションを無効にしたり ネットワークに影響したりします。
- 遅くれるクロックを持つゲストは移行で問題を持つ可能性があります。

これらの問題は他の仮想化プラットフォーム上に存在しており、タイミングは 常にテストされるべきです。

重要

NTP (Network Time Protocol) デーモンはホストとゲスト上で稼働している必要があります。 **ntpd** サービスを有効にして下さい:

```
# service ntpd start
```

ntpd サービスをデフォルトのスタートアップシーケンスに追加します:

```
# chkconfig ntpd on
```

ntpd サービスを使用すると、全ての ケースでクロックのずれの効果を最低限に抑えることができるはずです。

使用している CPU が不変タイムスタンプカウンタを持つかどうかを判定

constant_tsc フラグが存在する場合は、 使用中の CPU が不変タイムスタンプカウンタを持っていることになります。 その CPU が **constant_tsc** フラグを持つか どうかを判定するには、以下のコマンドを実行します:

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

なんらかの出力が表示されると、その CPU が **constant_tsc** ビットを持つことになります。出力がない場合は以下の案内に 従ってください。

不変タイムスタンプカウンタ無しでホストを設定

不変タイムスタンプカウンタを持たないシステムは追加の設定を必要とします。 パワーマネジメント機能は正確な時刻維持を邪魔するため、ゲスト用にはそれを 無効にして KVM での時刻維持を保護します。

重要

これらの案内は AMD 改訂 F cpus 用のみのものです。

CPU に **constant_tsc** ビットが無い場合、 全てのパワーマネジメント機能 (BZ#513138) を無効にして下さい。 各システムはいくつかのタイマーで時刻を維持しています。TSC はホスト上で不安定 であり、時には **cpufreq** 変更、deep C 状態、又は より速い TSC を持つホストへの移行により影響を受けます。 カーネルの deep C 状態 を防止するには、ホスト上で **grub.conf** ファイル内のカーネルブートオプションに **"processor.max_cstate=1"** を追加します:

```
term Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro root=/dev/VolGroup00/LogVol00
rhgb quiet processor.max_cstate=1
```

（**constant_tsc** の無いホスト上でのみ）**cpufreq** を 無効にするには、**/etc/sysconfig/cpuspeed** 設定ファイルの編集により、**MIN_SPEED** 変数と **MAX_SPEED** 変数を利用できる最高の周波数に変更します。 その有効な限界は **/sys/devices/system/cpu/cpu*/cpufreq/scaling_available_frequencies** ファイル内で見ることができます。

Red Hat Enterprise Linux ゲストで para-virtualized クロックの使用

特定の Red Hat Enterprise Linux ゲストには、追加のカーネルパラメータが 必要になります。これらのパラメータは、ゲストの /boot/grub/grub.conf ファイル内の /kernel 行の末尾にそれら自身を追記することによりセットできます。

以下の表では、そのような Red Hat Enterprise Linux のバージョンと 不変タイムスタンプカウンタを持たないシステム上のゲストで必要となるパラメータを示しています。

Red Hat Enterprise Linux	その他のゲストカーネルパラメータ
para-virtualized クロックを持つ 5.4 AMD64/Intel 64	追加のパラメータは無用です
para-virtualized クロックを持たない 5.4 AMD64/Intel 64	divider=10 notsc lpj=n
para-virtualized クロックを持つ 5.4 x86	追加のパラメータは無用です
para-virtualized クロックを持たない 5.4 x86	divider=10 clocksource=acpi_pm lpj=n
5.3 AMD64/Intel 64	divider=10 notsc
5.3 x86	divider=10 clocksource=acpi_pm
4.8 AMD64/Intel 64	notsc divider=10
4.8 x86	clock=pmtmr divider=10
3.9 AMD64/Intel 64	追加のパラメータは無用です

Red Hat Enterprise Linux	その他のゲストカーネルパラメータ
3.9 x86	追加のパラメータは無用です

Windows ゲストで Real-Time Clock を使用

Windows は RTC (Real-Time Clock) と TSC (Time Stamp Counter) の両方を 使用します。Windows ゲストには、TSC の代わりに、全ての時刻ソースのために 時刻維持問題を解決する Real-Time Clock を使用します。

PMTIMER クロックソース（PMTIMER は通常 TSC を使用）用に Real-Time Clock を 有効にするには、Windows のブート設定に以下の行を追加します。Windows のブート設定は boot.ini ファイル内にありますので、**boot.ini** ファイルに以下の 行を追加して下さい:

```
/use pmtimer
```

Windows のブートセッティングと pmtimer オプションに関する詳細情報には Windows XP および Windows Server 2003 の Boot.ini ファイルで使用可能なスイッチ オプション を参照して下さい。

第18章 XEN ライブ移行

Xen hypervisor は para-virtualized ゲストと 完全仮想化ゲストのために *移行* をサポートします。移行は Red Hat Enterprise Linux 5.1 及びそれ以降のシステムで のみサポートされています。移行はオフライン、又はライブで実行できます。

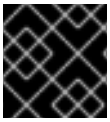
- オフライン移行はオリジナルホスト上の仮想化ゲストを休止して、それを目的地のホストへと移動し、それからそのゲストが完全に転送された時点でそれを復帰します。オフライン移行は **virsh migrate** コマンドを使用します。

  ```
  # virsh migrate GuestName libvirtURI
  ```

- ライブ移行では、送信元のホスト上でゲストの稼働を維持しながら、ゲストを停止 せずにメモリーの移動を始めます。そのイメージが送信された後で全ての修正済み メモリーページが追跡されて目的地まで送信されます。メモリーは変更のあったページで 更新を受けます。このプロセスはそれが発見的状況に到達するまで継続します。 それは全てのページが正常にコピーされるか、又は送信元の変化が速過ぎて、送信先の ホストが進展できない状態のどちらかです。発見的状況に到達した場合、ゲストは送信元ホストで一時的に休止して、レジスタとバッファが送信されます。レジスタは送信先のホストで ロードされてゲストはそれからその新しいホスト上で復帰します。ゲストがマージ出来ない 場合（ゲストの負荷が超過している場合に発生）、ゲストは一時停止して、それからは代わりに オフライン移行が開始されます。

  ライブ移行には **virsh migrate** コマンドで **--live** オプションを使用します。

  ```
  # virsh migrate--live GuestName libvirtURI
  ```



重要

移行は現時点では、Itanium® アーキテクチャ上でのサポートがありません。

Xen を使用した移行を有効にするには、 **/etc/xen/xend-config.sxp** の 設定ファイルにいくつかの変更が必要になります。 不正確な設定がセキュリティ問題になる可能性があるため、デフォルトでは移行は 無効になっています。移行ポートを開くと権限の無いホストが移行を開始したり、移行ポートに 接続したりできるようになります。移行要求用に認証と権限授与が設定されておらず、唯一の 制御メカニズムはホスト名と IP アドレスを基にしています。移行ポートが無許可のホストに アクセスされないように特別な注意が必要です。

重要

IP アドレスとホスト名のフィルタは最低限のセキュリティのみを提供します。これらの属性は両方共、攻撃者が移行クライアントのアドレスとホスト名を知っていると偽造できます。移行のための最善のセキュリティはネットワークを、外部と権限の無い内部接続から隔離することです。

移行を有効にする

/etc/xen/xend-config.sxp 内で以下のエントリを修正して、 移行を有効にします。必要であれば値を修正して、以下のパラメータの前にあるコメント （# マーク）を削除します。

(xend-relocation-server yes)

移行を無効にするデフォルト値は、**no** です。**xend-relocation-server** の値を **yes** に 変更して移行を有効にします。

(xend-relocation-port 8002)

xend-relocation-server が **yes** に セットされている場合、このパラメータ、**(xend-relocation-port)**、は ポート **xend** が移動インターフェイスの 為に使用されるべきことを指定します。

この変数のデフォルト値は ほとんどのインストールで機能するはずです。この値を 変更する場合は、移動サーバー上で未使用のポートを使用することを確認して下さい。

xend-relocation-port パラメータでセットされたポートは両方の システムで開かれる必要があります。

(xend-relocation-address '')

(xend-relocation-address) は、 **xend-relocation-server** がセットされている場合に、**relocation-socket** 接続の移行コマンドの為に、**xend** がリッスンするアドレスです。

デフォルト設定は全てのアクティブインターフェイスをリッスンします。**(xend-relocation-address)** パラメータは移行サーバーを制約して特定のインターフェイス だけをリッスンするようにできます。 **/etc/xen/xend-config.sxp** 内のデフォルト値は 空の文字列 **('')** です。この値は単独の有効な IP アドレスで入れ替える必要があります。例えば:

```
(xend-relocation-address '10.0.0.1')
```

(xend-relocation-hosts-allow '')

(xend-relocation-hosts-allow 'hosts') パラメータはどのホスト名が移動ポート上で通信できるかを制御します。

SSH 又は TLS を使用しているのでなければ、ゲストの仮想メモリーは暗号化の無い通信で 生の状態で転送されます。**xend-relocation-hosts-allow** オプションを修正して、移行サーバーへのアクセスを制限します。

単独引用句で囲まれた空の文字列で示した上のサンプルのように、その値が空である場合、全ての接続が許可されます。これは、移動サーバーがリッスンするポートとインターフェイスに接続が到着することを想定します。（上述の **xend-relocation-port** と **xend-relocation-address** も参照して下さい）

それ以外の場合は、**(xend-relocation-hosts-allow)** パラメータは空白で分離された正規表現の連続でなければなりません。 これらの正規表現の１つに一致する完全修飾ドメイン名、又は IP アドレスを持つ ホストはいずれも受理されます。

(xend-relocation-hosts-allow) 属性の サンプル:

```
(xend-relocation-hosts-allow '^localhost$ ^localhost\\.localdomain$')
```

使用する設定ファイル内でパラメータを設定した後は、Xen サービスを再スタートします。

```
# service xend restart
```

18.1. ライブ移行の例

以下にライブ移行の為の簡単な環境のセットアップの仕方の例を示します。 この設定では、共有ストレージの為に **NFS** を 使用します。**NFS** はデモ用の環境には適していますが、 実稼働用の環境には、ファイバーチャンネル又は、iSCSI と **GFS** を使用したより強健な共有ストレージが推奨されます。

以下の設定は２つのサーバーで構成されています（**et-virt07** と **et-virt08**）。この両方は **eth1** をそのデフォルトネットワークインターフェイスとして 使用しているため、**xenbr1** をその Xen ネットワーキング ブリッジとして使用しています。この例では、**et-virt07** 上でローカルに取り付けた SCSI disk (**/dev/sdb**) を **NFS** を 介して共有ストレージ用に使用しています。

ライブ移行のセットアップ

移行に使用されるディレクトリを作成してマウントします:

```
# mkdir /var/lib/libvirt/images
# mount /dev/sdb /var/lib/libvirt/images
```

警告

ディレクトリが正しいオプションでエキスポートされることを確認してください。デフォルトのディレクトリ、**/var/lib/libvirt/images/** を エキスポートしている場合は、**/var/lib/libvirt/images/** のみを エキスポートすることと、**/var/lib/xen/ ではないこと** を 確認して下さい。**/var/lib/xen/** ディレクトリは **xend** デーモンと他のツールにより使用されるものであり、このディレクトリを共有すると、 予知できない動作を起こす可能性があります。

```
# cat /etc/exports
/var/lib/libvirt/images  *(rw,async,no_root_squash)
```

NFS を介してエキスポートされていることを確認します:

```
# showmount -e et-virt07
Export list for et-virt07:
/var/lib/libvirt/images *
```

ゲストをインストール

ゲストのインストールの為に使用されるサンプルのインストールコマンド:

```
# virt-install -p -f /var/lib/libvirt/images/testvm1.dsk -s 5 -n\
testvm1 --vnc -r 1024 -l http://example.com/RHEL5-tree\
Server/x86-64/os/ -b xenbr1
```

ステップバイステップのインストール案内には、*7章ゲストオペレーティングシステムのインストール手順* を参照して下さい。

移行用の環境を確認

仮想化ネットワークブリッジが正しく設定されており、両方のホスト上で同じ名前を 持つことを確認します:

-

```
[et-virt08 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
xenbr1          8000.feffffffffff       no              peth1
vif0.1
```

```
[et-virt07 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
xenbr1          8000.feffffffffff       no              peth1
vif0.1
```

移動パラメータが両方のホスト上で設定されていることを確認します:

```
[et-virt07 ~]# grep xend-relocation /etc/xen/xend-config.sxp |grep -v '#'
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-address '')
(xend-relocation-hosts-allow '')
```

```
[et-virt08 ~]# grep xend-relocation /etc/xen/xend-config.sxp |grep -v '#'
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-address '')
(xend-relocation-hosts-allow '')
```

移動サーバーがスタートしたことと、Xen 移行 (8002) 用の専用ポートでリッスンしている ことを確認します:

```
[et-virt07 ~]# lsof -i :8002
COMMAND  PID  USER   FD   TYPE  DEVICE SIZE NODE NAME
python 3445 root 14u IPv4 10223 TCP *:teradataordbms (LISTEN)
```

```
[et-virt08 ~]# lsof -i :8002
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
python 3252 root 14u IPv4 10901 TCP *:teradataordbms (LISTEN)
```

NFS ディレクトリが他のホストにマウントされていることと、仮想マシンイメージとファイルシステムが 可視であり、アクセスできることを確認します:

```
[et-virt08 ~]# df /var/lib/libvirt/images
Filesystem           1K-blocks      Used Available Use% Mounted on
et-virt07:/var/lib/libvirt/images    70562400   2379712  64598336   4%
/var/lib/libvirt/images
```

```
[et-virt08 ~]# file /var/lib/libvirt/images/testvm1.dsk
/var/lib/libvirt/images/testvm1.dsk: x86 boot sector; partition 1:
ID=0x83,
active, starthead 1, startsector 63, 208782 sectors; partition 2: ID=0x8e,
starthead 0, startsector 208845, 10265535 sectors, code offset 0x48
```

```
[et-virt08 ~]# touch /var/lib/libvirt/images/foo
[et-virt08 ~]# rm -f /var/lib/libvirt/images/foo
```

ゲストの保存と復元を確認します

仮想マシンをスタートします（まだ仮想マシンがオンでない場合）:

```
[et-virt07 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

```
[et-virt07 ~]# virsh start testvm1
Domain testvm1 started
```

仮想マシンが稼働していることを確認します:

```
[et-virt07 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

ローカルホスト上の仮想マシンを保存します:

```
[et-virt07 images]# time virsh save testvm1 testvm1.sav
real    0m15.744s
user    0m0.188s
sys     0m0.044s
```

```
[et-virt07 images]# ls -lrt testvm1.sav
-rwxr-xr-x 1 root root 1075657716 Jan 12 06:46 testvm1.sav
```

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

ローカルホスト上で仮想マシンを復元します:

```
[et-virt07 images]# virsh restore testvm1.sav
```

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

domain-id のライブ移行を **et-virt07** から**et-virt08** へ開始します。移行先のホスト名と <domain-id> の部分は、有効な値で入れ替える必要があります。

```
[et-virt08 ~]# xm migrate --live domain-id et-virt07
```

仮想マシンが **et-virt07** 上でシャットダウンされたことを確認します。

```
[et-virt07 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

仮想マシンが **et-virt08** に移行されたことを確認します:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

進行具合をテストしてライブ移行を始動

仮想マシン内で以下のスクリプトを作成して、移行中に日付とホスト名をログ します。このスクリプトはゲストのファイルシステム上で I/O タスクを実行します。

```
#!/bin/bash

while true
do
touch /var/tmp/$$.log
echo `hostname` >>  /var/tmp/$$.log
        echo `date`     >>  /var/tmp/$$.log
        cat  /var/tmp/$$.log
        df /var/tmp
        ls -l  /var/tmp/$$.log
        sleep 3
        done
```

スクリプトは単にテスト目的だけのためであり、実稼働環境でのライブ移行には 不要であることを忘れないで下さい。

仮想マシンを **et-virt07** に移行する前に、 それが **et-virt08** 上で稼働していることを確認します:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

et-virt07 へのライブ移行を始動します。 **time** コマンドを追加すると、移行にかかる時間を見ることができます:

```
[et-virt08 ~]# xm migrate --live testvm1 et-virt07
```

ゲスト内でスクリプトを実行します:

```
# ./doit
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
```

```
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 62 Jan 12 02:26 /var/tmp/2279.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:30 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 124 Jan 12 02:26 /var/tmp/2279.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:30 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:33 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 186 Jan 12 02:26 /var/tmp/2279.log
Fri Jan 12 02:26:45 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:48 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:51 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:54:57 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:55:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:55:03 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 744 Jan 12 06:55 /var/tmp/2279.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:26:27 EST 2007
```

仮想マシンが **et-virt08** 上でシャットダウンされたことを確認します:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

仮想マシンが **et-virt07** 上でスタートしたことを確認します:

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

et-virt07 から **et-virt08** へと もう1度移行サイクルを実行します。**et-virt07** から **et-virt08** への移行を以下のようにして開始します:

```
[et-virt07 images]# xm migrate --live testvm1 et-virt08
```

仮想マシンがシャットダウンされたことを確認します:

```
[et-virt07 images]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
```

移行を初めて執行する前に、ゲスト内で簡単なスクリプトを開始して、ゲストが 移行するときの時間の変化に注意します:

```
# ./doit
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 62 Jan 12 06:57 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 124 Jan 12 06:57 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 186 Jan 12 06:57 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:00 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 248 Jan 12 02:30 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
```

```
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:03 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 310 Jan 12 02:30 /var/tmp/2418.log
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:53 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:57:56 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 06:58:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:00 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:03 EST 2007
dhcp78-218.lab.boston.redhat.com
Fri Jan 12 02:30:06 EST 2007
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
2983664   2043120    786536  73% /
-rw-r--r-- 1 root root 372 Jan 12 02:30 /var/tmp/2418.log
```

移行コマンドが **et-virt07** 上で完了すると、 **et-virt08** 上で仮想マシンがスタートしたことを 確認します:

```
[et-virt08 ~]# virsh list
 Id Name                 State
----------------------------------
Domain-0                 running
testvm1        blocked
```

そしてもう１つのサイクルを実行します:

```
[et-virt08 ~]# time virsh migrate --live testvm1 et-virt07
real    0m10.378s
user    0m0.068s
sys     0m0.052s
```

これで、オフラインとライブの移行テストを正常に実行できました。

18.2. ゲストライブ移行の設定

このセクションは、Red Hat Enterprise Linux を稼働している 他のサーバーへの Xen ゲストのオフライン移行を説明しています。 移行はオフラインメソッド内で 実行されます（**xm migrate** コマンドを使用）。ライブ移行は 同じコマンドで実行できますが、その場合、**xend-config** 設定ファイルにいくらかの追加修正が必要になります。この例では、正しい移行を達成する為に 修正すべきエントリを識別しています:

(xend-relocation-server yes)

このパラメータ用のデフォルトは「No」です。これは移動/移行サーバーを (信頼できるネットワーク上でない限り) 不活性にしておき、ドメイン仮想メモリーは暗号化無しに生のままで交換されます。

(xend-relocation-port 8002)

このパラメータは、**xend** が移行用に使用するポートを設定します。使用するネットワーク環境がカスタム値を要求しない限りは、この値を使用します。コメント用の シンボルを削除して有効にします。

(xend-relocation-address)

このパラメータは、ユーザーが **xend-relocation-server** を有効にした後に、移動ソケット接続の為にリッスンするアドレスです。Xen hypervisor は 指定されたインターフェイス上の移行ネットワークトラフィックだけをリッスンします。

(xend-relocation-hosts-allow)

このパラメータは移動ポートと通信するホストを制御します。その値が空であれば、全ての来信接続は許可されます。これを空白で区切られた正規表現の連続に変える必要があります。例えば:

```
(xend-relocation-hosts-allow- '^localhost\\.localdomain$' )>
```

受理される値には、完全修飾ドメイン名、IP アドレス、あるいは空白で隔離した 正規表現が含まれます。

設定した後に、ホストを再起動して新しいセッティングをロードします。

第19章 KVM ライブ移行

この章では、KVM hypervisor で稼働しているゲストを別の KVM ホストに移行する方法を説明しています。

移行とは、仮想化ゲストを１つのホストから別のホストへ移動することを指します。 ここではソフトウェアは完全にハードウェアから隔離されているため、移行は仮想化の 基幹機能と言えます。移行は以下のような状況で役に立ちます:

- ロードバランシング: あるホストが過大な負荷を持つとき、ゲストは使用度の低いホストへと移動できます。
- ハードウェアフェイルオーバー：ホスト上のハードウェアデバイスが障害を持つと、 ゲストが安全な場所に移動して、その間にホストは電源を落として修理ができます。
- エネルギー節約：ゲストが他のホストに再分配されると、ホストシステムは 電源を落とすことによりエネルギーを節約し、使用度が低い時にはコストを 低減します。
- 地域的な移行：ゲストは、低遅延度を求めて、又は重大な状況下で 他の地域に移動することができます。

移行はライブで、又はオフラインで実行できます。ゲストを移行するためには ストレージが共有されなければなりません。移行はゲストのメモリーを目的地の ホストに送信することにより達成されます。共有ストレージはゲストのデフォルト ファイルシステムを格納しています。このファイルシステムイメージは送信元のホストから直接 送信先のホストまでネットワーク上で送信されるのではありません。

オフライン移行はゲストを休止して、それからそのゲストのメモリーイメージを目的地の ホストに移動します。ゲストは目的地のホストで復帰して、送信元ホスト上でゲストが使用した メモリーは開放されます。

オフライン移行が必要とする時間はネットワークのバンド幅と遅延度によります。 2GB のメモリーを持つゲストでは 1 Gbit イーサネットリンクで平均的に10秒位の 時間がかかります。

ライブ移行は送信元のホスト上でゲストを稼働し続け、ゲストを停止することなく、 そのメモリーの移動を始めます。全ての修正されたメモリーページはその変更部を監視され、 そのイメージが送信されている間に目的地へ送信されます。メモリーは変更のあった ページで更新されます。このプロセスはゲスト用に許容された休止時間数が、転送される 最後の数ページ用の予想時間と同等になるまで継続します。KVM はその残りの時間を推測し、 仮想化ゲストの短い休止時間内に残りのページが転送できると予測できるまで 送信元から送信先まで最大量のページを転送する試みを続けます。レジスタが新規のホストにロードされて、ゲストはそれから送信先のホストで復帰します。ゲストが マージできない場合（ゲストが極端な負荷を持つ時に発生）、ゲストは休止して、 それから代わりに オフライン移行が始まります。

オフライン移行にかかる時間は、ネットワークのバンド幅とその遅延度によります。ネットワークが多大な負荷を持つ場合、又はバンド幅が低い場合は、移行には長い 時間がかかります。

19.1. ライブ移行の要件

ゲストの移行には以下が必要です:

移行の要件

- 以下のプロトコルの１つを使用して共有ネットワークストレージにインストール された仮想化ゲスト:

- ファイバーチャンネル
- iSCSI
- NFS
- GFS2

- 同じ更新を持つ同じバージョンの 2つ又はそれ以上の Red Hat Enterprise Linux システム
- 両方のシステムは適切なポートを開いている必要があります。
- 両方のシステムは同一のネットワーク設定を持つ必要があります。全ての ブリッジとネットワーク設定は両方のホスト上で全く同じでなければなりません。
- 共有ストレージは送信元と送信先のシステム上で同じ場所でマウントしなければ なりません。そしてマウントされたディレクトリ名も同一である必要があります。

ネットワークストレージの継続

共有ストレージを設定してその共有ストレージにゲストをインストールします。 共有ストレージに関する案内には、9章*共有ストレージと仮想化*を参照して下さい。

別の方法として、「共有ストレージサンプル: 簡単な移行のための NFS」内の NFS サンプルを使用します。

19.2. 共有ストレージサンプル: 簡単な移行のための NFS

この例では、NFS を使用して他の KVM ホストと共にゲストイメージを共有しています。 この例は、大規模のインストールには実用的ではありません。これ例は、単に移行の 技術を示すためのものであり、小規模の導入用です。この例を数個の仮想化 ゲスト以上の環境で移行、又は実行に使用しないで下さい。

高度な、そしてより強健な共有ストレージに関する案内には、以下を参照して 下さい。 9章*共有ストレージと仮想化*

1. **libvirt イメージディレクトリをエキスポート**
   デフォルトのイメージディレクトリを **/etc/exports** ファイルに 追加します:

   ```
   /var/lib/libvirt/images *.example.com(rw,no_root_squash,async)
   ```

   ご自分の環境に適するようにホストパラメータを変更します。

2. **NFS の開始**

   a. NFS パッケージがインストールされていない場合は、インストールします:

   ```
   # yum install nfs
   ```

   b. **iptables** で NFS 用のポートを開きます。そして NFS を **/etc/hosts.allow** ファイルに追加します。

   c. NFS サービスを開始:

   ```
   # service nfs start
   ```

3. **共有ストレージを目的地でマウントします**
   目的地のシステムで、**/var/lib/libvirt/images** ディレクトリをマウントします:

```
# mount sourceURL:/var/lib/libvirt/images /var/lib/libvirt/images
```

警告

ゲスト用にどのディレクトリが選択されても、それはホストとゲスト上で全く 同じでなければなりません。これは、全てのタイプの共有ストレージに該当 します。ディレクトリが同じでないと、移行は失敗します。

19.3. VIRSH を使用した ライブ KVM 移行

virsh コマンドを使用してゲストを別のホストに移行することが できます。**migrate** コマンドは 以下の形式のパラメータを 受け付けます:

```
# virsh migrate --live GuestName DestinationURL
```

GuestName パラメータは、移行したいゲストの 名前を表すものです。

DestinationURL パラメータは目的地システムの URL かホスト名です。目的地システムは Red Hat Enterprise Linux の同じバージョンを 実行しなければならず、同じ hypervisor を使用して、**libvirt** が 稼働している必要があります。

コマンドが入力されると、目的地システムの root パスワードを催促されます。

例: virsh を使用したライブ移行

この例では、**test1.example.com** から **test2.example.com** へ移行をします。ご自分の 環境に適したホスト名に変更して下さい。この例は **RHEL4test** と言う 仮想マシンを移行します。

この例では、ユーザーが共有ストレージを完全に設定していて、全ての 前提事項を満たしていると想定します（移行の要件 に一覧表示）。

1. **ゲストが稼働していることを確認します**
   送信元のシステム **test1.example.com** から **RHEL4test** が稼働していることを確認します:

```
[root@test1 ~]# virsh list
Id Name                 State
----------------------------------
 10 RHEL4                running
```

2. **ゲストを移行**
   以下のコマンドを実行して、ゲストを目的地、**test2.example.com** に ライブ移行します。**/system** を目的地の URL の末尾に追記することで全面的アクセスが 必要であることを libvirt に伝えます。

```
# virsh migrate --live RHEL4test qemu+ssh://test2.example.com/system
```

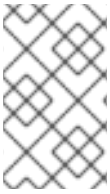
コマンドが入力されると、目的地システムの root パスワードを催促されます。

3. **待ち時間**
この移行はゲスト上の負荷とそのサイズによりいくらかの時間がかかります。**virsh** は エラーを報告するだけです。このゲストは完全に移行が終了するまで送信元のホストで稼働し続けます。

4. **ゲストが目的地のホストに到着したことを確認**
目的地のシステム、**test2.example.com** から、 **RHEL4test** が稼働していることを確認します:

```
[root@test2 ~]# virsh list
Id Name                 State
----------------------------------
 10 RHEL4                running
```

ライブ移行はこれで完了しました。



注記

libvirt は、TLS/SSL、unix sockets、SSH、及び暗号化の無い TCP を含む各種の ネットワーキングメソッドをサポートします。これらのメソッドの使用法に関する 詳細には、20章*仮想化ゲストのリモート管理*を参照して下さい。

19.4. VIRT-MANAGER での移行

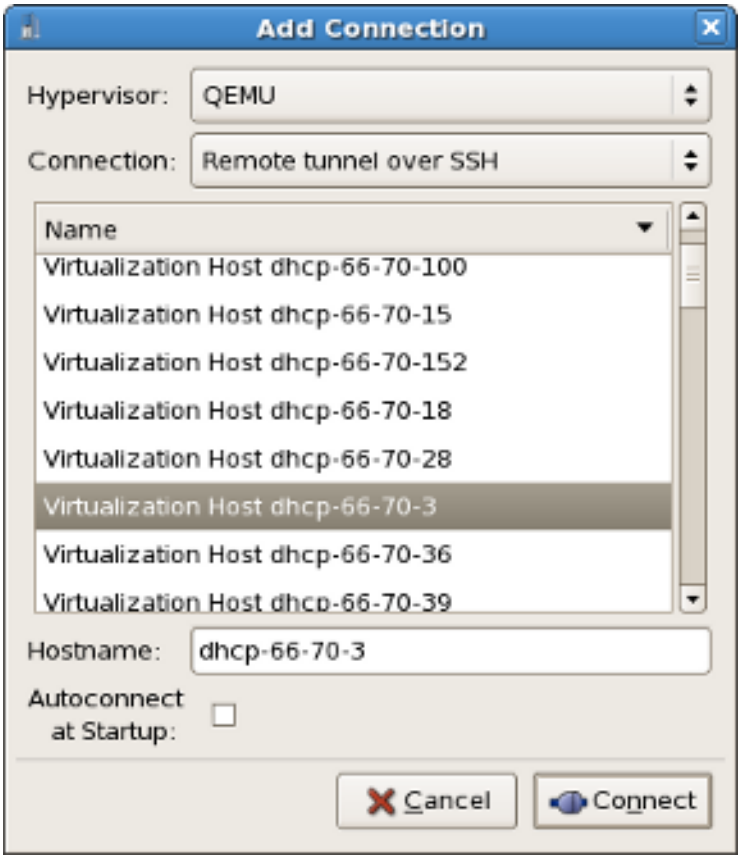
このセクションでは、**virt-manager** を使用した KVM ベース ゲストの移行を取り扱います。

1. 送信元を送信先のホストを接続します。**ファイル** メニューで、 **接続を追加** をクリックすると、**接続を 追加** のウィンドウが出現します。

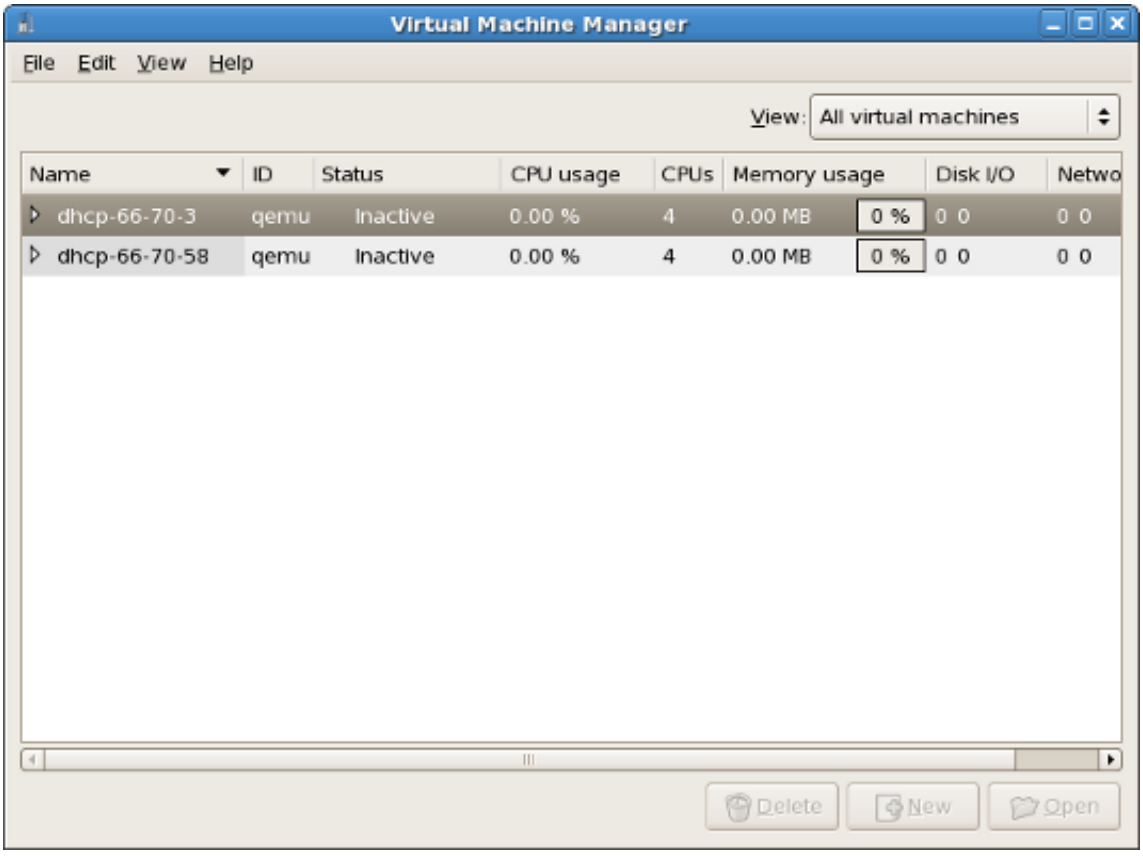
   以下の詳細を入力します:

   - **Hypervisor**: **QEMU** を選択。
   - **接続:** 接続タイプを選択します。
   - **ホスト名:** ホスト名を入力します。

   接続 をクリックします。



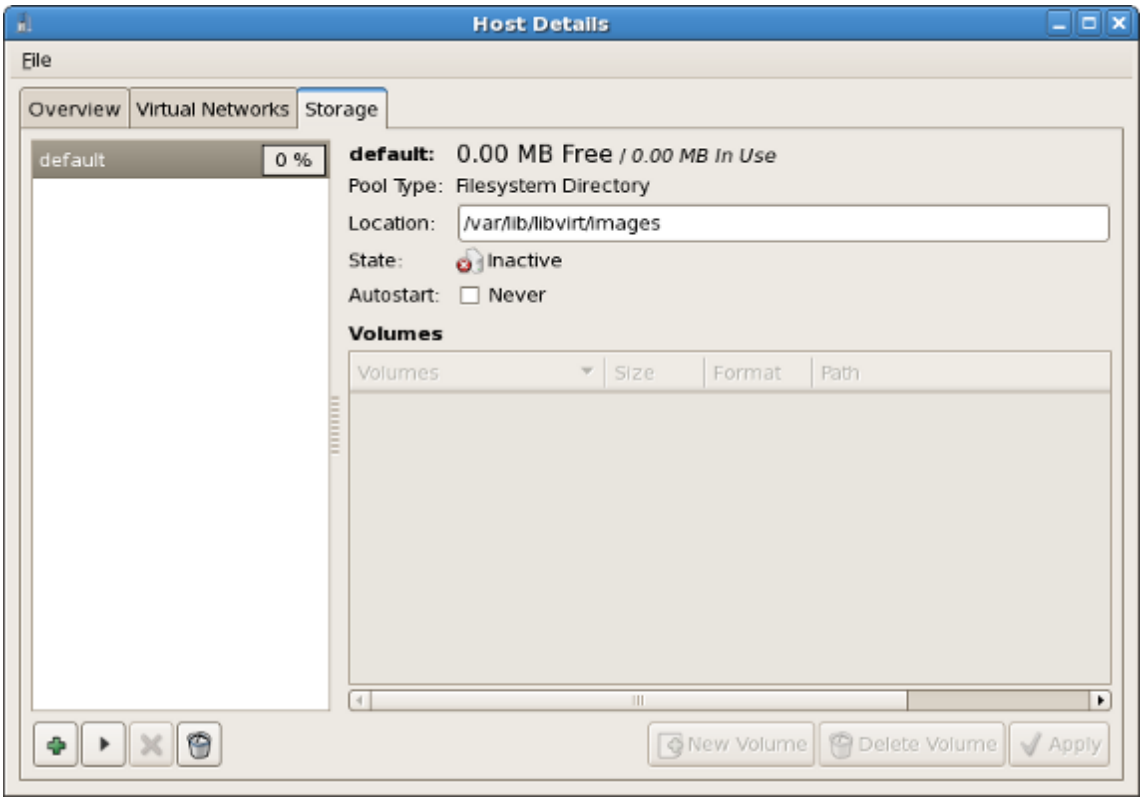
仮想マシンマネージャのウィンドウが接続してあるホストの一覧を表示します。



2. ソースとターゲットのホストに同じ NFS を使用するストレージプールを追加します。

   編集 メニューで、**ホストの詳細** をクリックすると、「ホストの詳細」ウィンドウが表示されます。

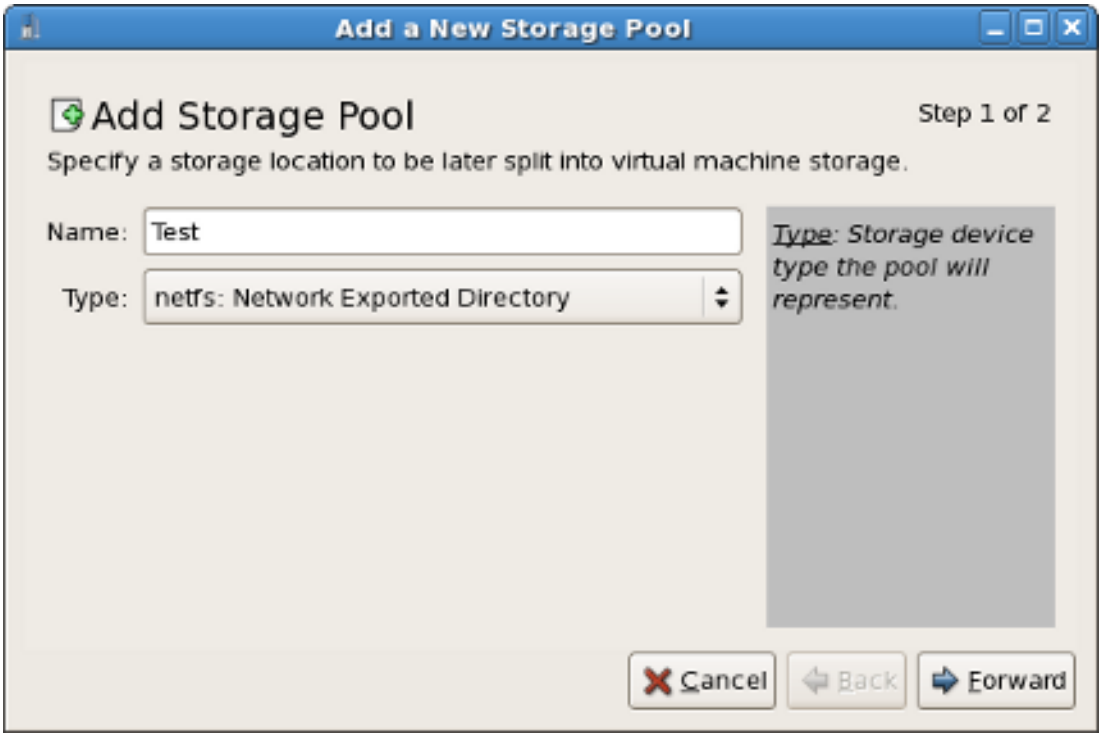
ストレージ タブをクリックします。



3. 新規のストレージプールを追加します。そのウィンドウの左下コーナーで **+** ボタンをクリックします。「新規ストレージプールの追加」ウィンドウが表示されます。

   以下の詳細を入力します:

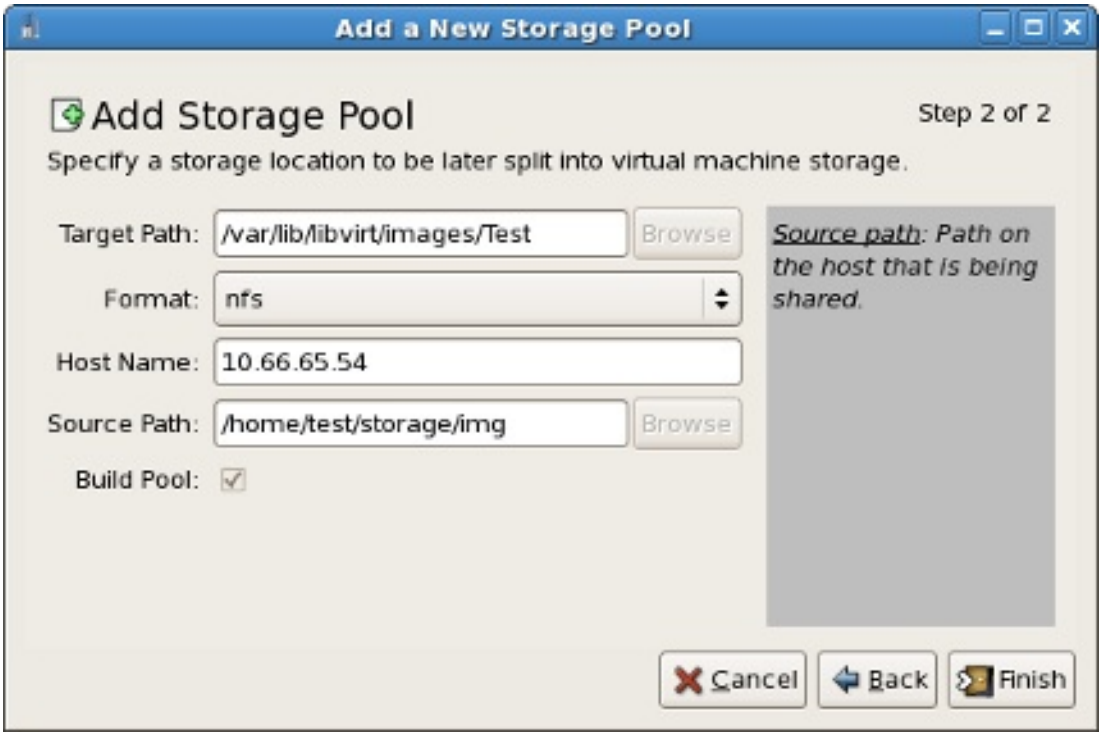
   - **名前:** ストレージプールの名前を入力します。
   - **タイプ: netfs: Network Exported ディレクトリ** を選択します。



進む をクリックします。

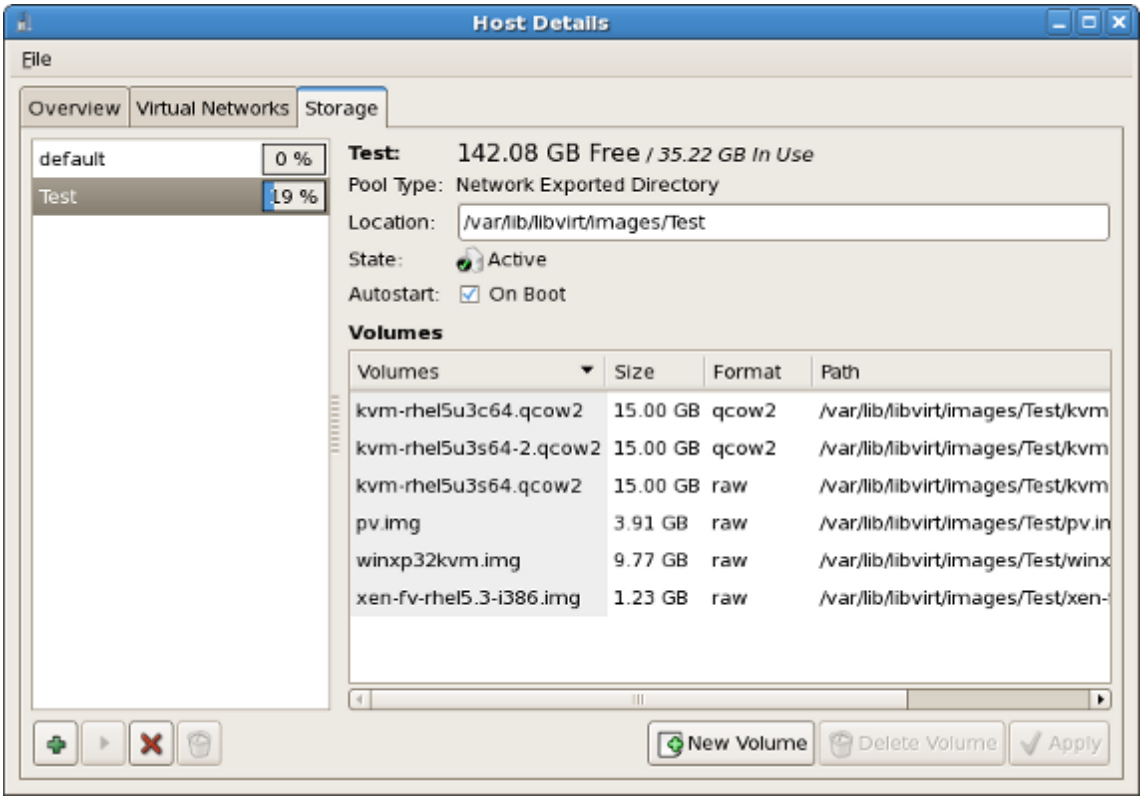
4. 以下の詳細を入力します:

   - **形式:** ストレージタイプを選択します。これはライブ移行用には NFS か、又は iSCSI のいずれかです。
   - **ホスト名:** ストレージサーバーの IP アドレスか、又は 完全修飾型ドメイン名を入力します。

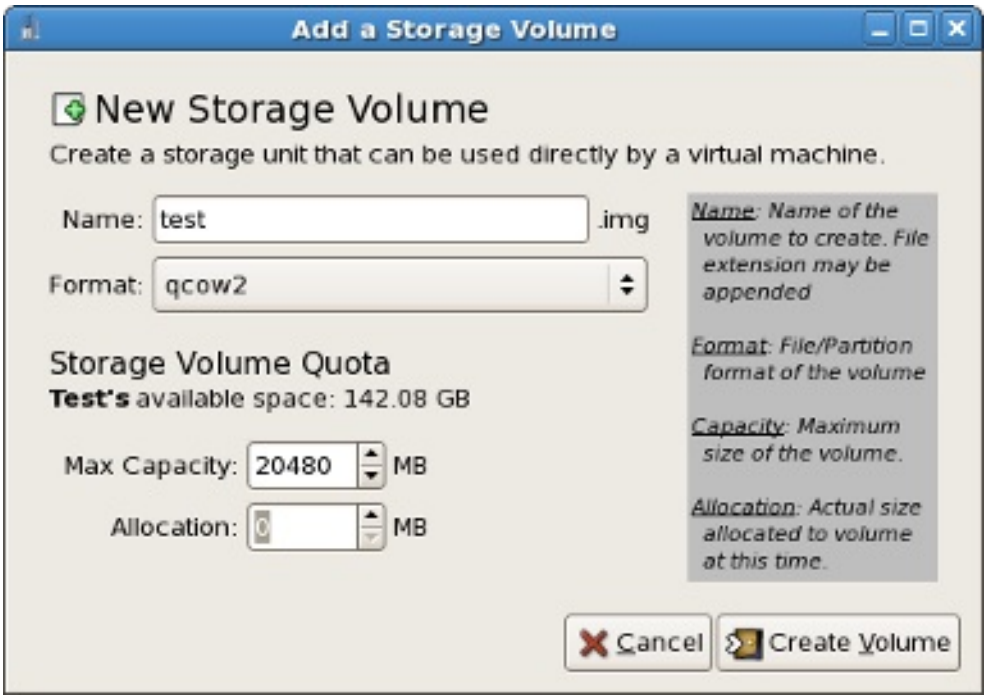


完了 をクリックします。

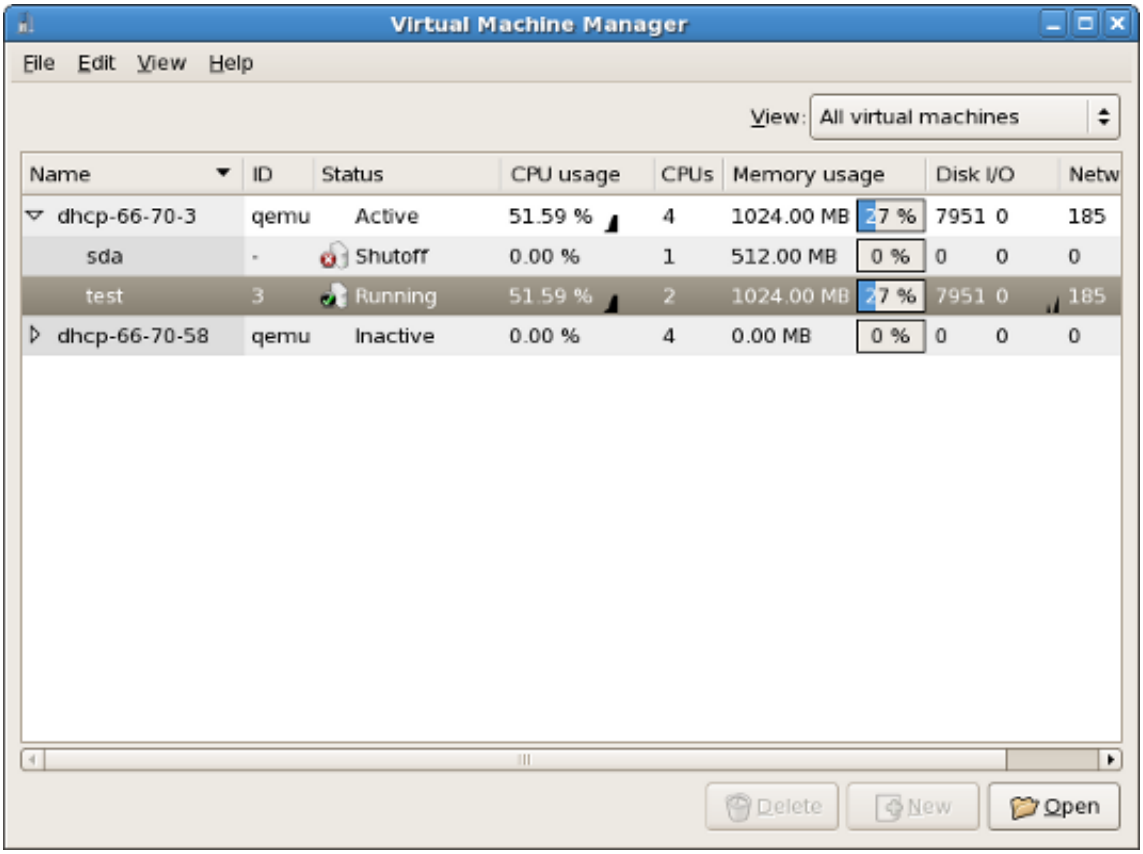
5. 共有ストレージプール内で新規のボリュームを作成して、**新規ボリューム** をクリックします。



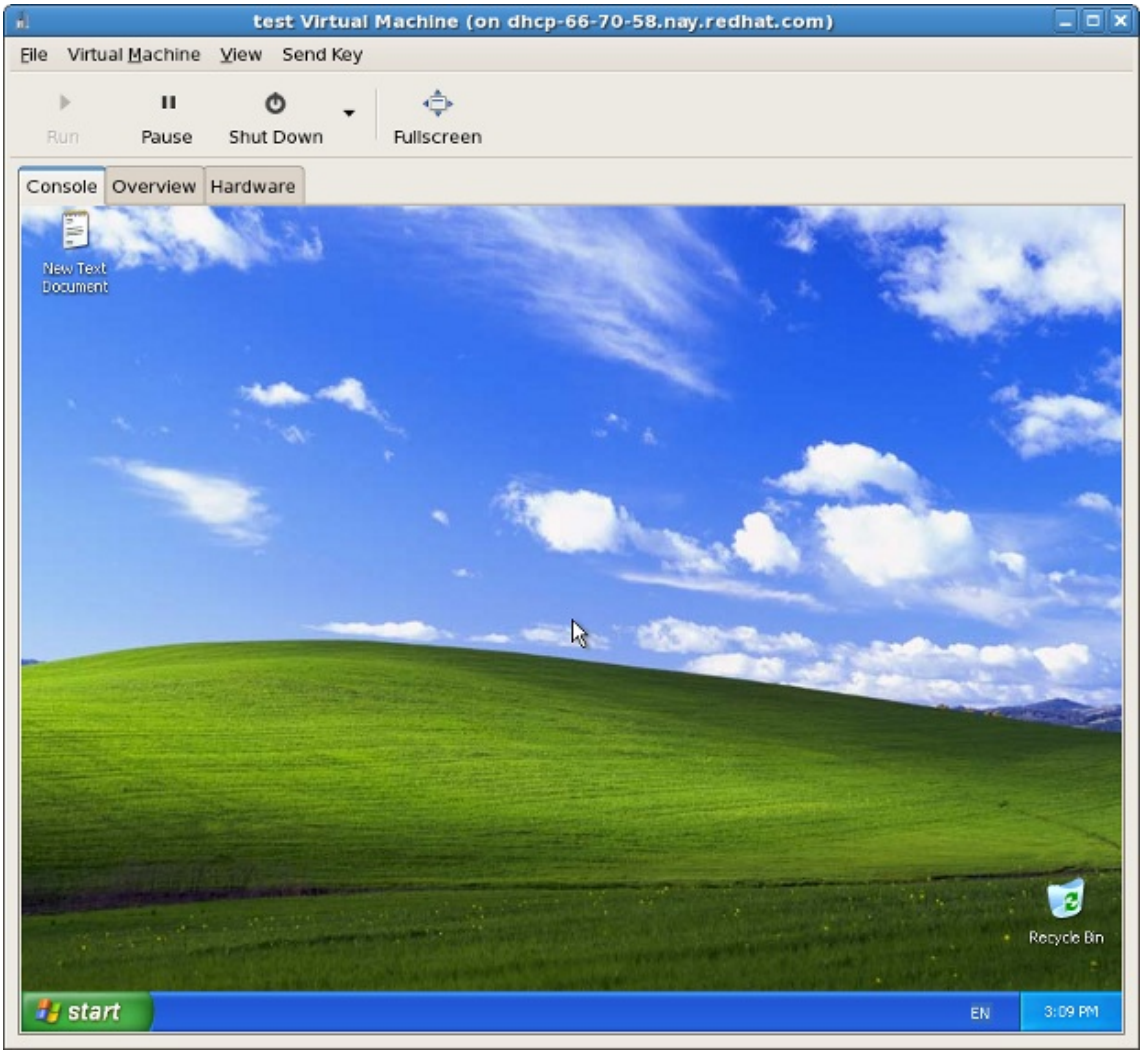
6. 詳細を入力して、それから **ボリュームの作成** をクリックします。



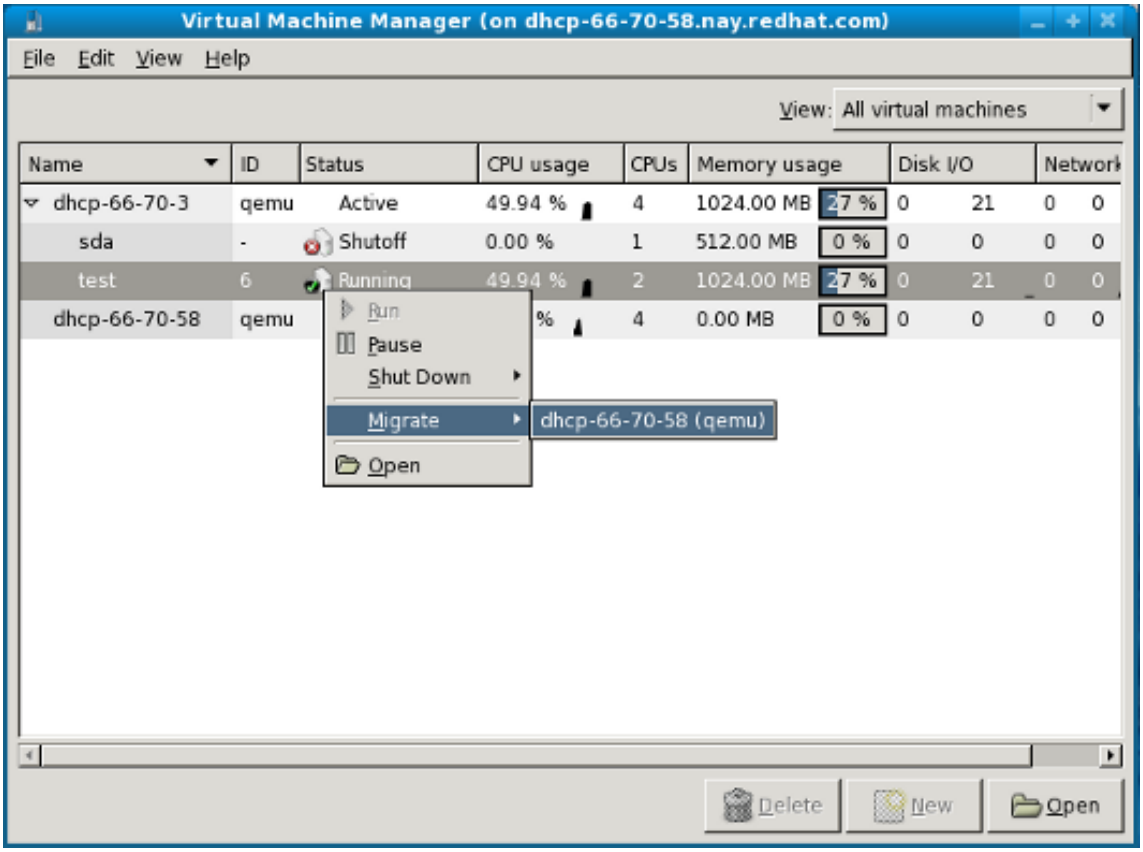
7. 新規のボリュームで仮想マシンを作成します。その後仮想マシンを実行します。



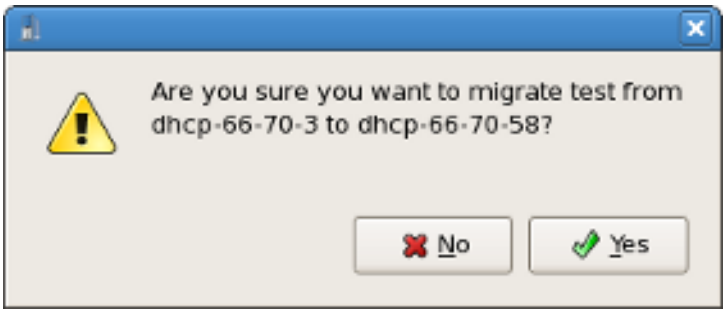
仮想マシンのウィンドウが開きます。



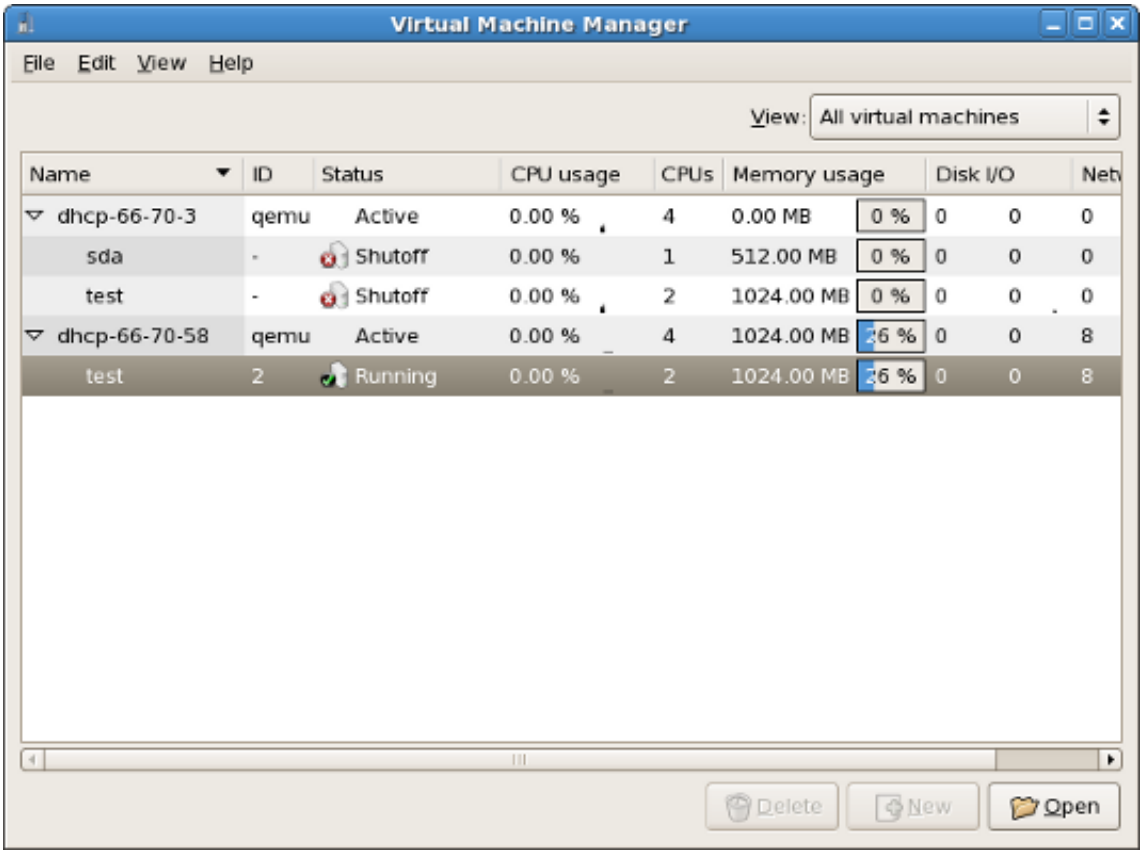
8. 仮想マシンマネージャのウィンドウで、仮想マシン上で右クリックします。 **移行** を選択してから、移行の場所をクリックします。



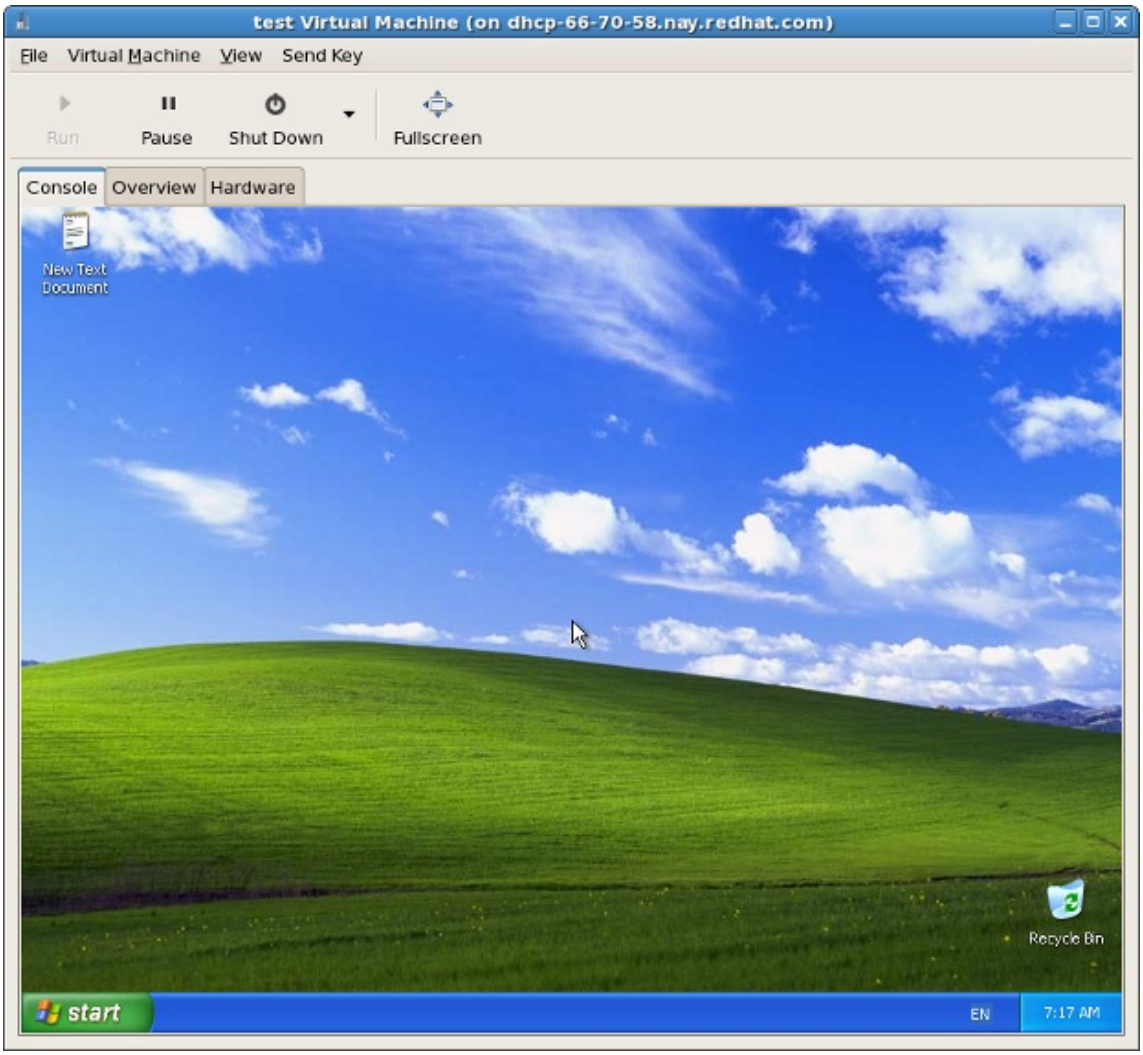
9. **はい** をクリックして移行を継続します。



仮想マシンマネージャは新しい場所での仮想マシンを表示します。



VNC 接続はそのタイトルバーにリモートホストのアドレスを表示します。



第20章 仮想化ゲストのリモート管理

このセクションでは、**ssh** か、TLS か、あるいは SSL を 使用して仮想化ゲストをリモートで管理する方法を説明しています。

20.1. SSH を使用したリモート管理

ssh パッケージは、リモート仮想化サーバーに対して安全に 管理機能を送信できる暗号化したネットワークプロトコルを提供します。ここに説明してある 方法は **SSH** 接続経由で安全にトンネル通過する **libvirt** 管理接続を使用して、リモートマシンを管理します。全ての認証は 使用中の**SSH** エージェントに よって収集された **SSH** パブリックキー暗号とパスワード、又はパスフレーズを 使用して行われます。更には、各ゲスト仮想マシンの **VNC** コンソールは **SSH** 経由でトンネル通過します。

SSH は通常、デフォルトで設定されており、 ユーザーには多分、既に SSH キーがセットされている可能性があり、管理サービスや **VNC** コンソールにアクセスするための余分のファイアウォール ルールは必要ないでしょう。

リモートで仮想マシンを管理するための **SSH** の 使用に於いて、以下を含む各種問題に注意して下さい:

- 仮想マシンの管理には、リモートマシンへの root ログインでアクセスする必要があります。
- 初期の接続セットアップは時間がかかるかも知れません。
- 全てのホスト、又はゲスト上でユーザーのキーを撤回するのに標準的な、あるいは平凡な方法はありません。
- **ssh** は多数のリモートマシン群に対してはうまく機能しません。

virt-manager の為の SSH アクセスの設定

以下の案内では、ユーザーが何もない状態からスタートしてまだ **SSH** キーを セットアップしていないと想定します。

1. **virt-manager** コマンドが使用されるマシン上で、パブリックキーのペアが 必要になります。**ssh** がすでに設定してある場合は、このコマンドは 無視することができます。

```
$ ssh-keygen -t rsa
```

2. リモートログインを許可するのに、**virt-manager** は、**libvirt** を 実行している各リモートマシン上のパブリックキーをコピーする必要があります。**scp** コマンドを 使用することにより、リモート管理に使用したいマシンから **$HOME/.ssh/id_rsa.pub** ファイルを コピーします:

```
$ scp $HOME/.ssh/id_rsa.pub root@somehost:/root/key-dan.pub
```

3. ファイルコピーが終了した後は、**root** として **ssh** を使用して リモートマシンに接続してコピーしたファイルを認可済みキーの一覧に追加します。 リモートホストの **root** ユーザーがまだ認可済みキーの一覧を所有していない場合は、 ファイル権限が正しくセットされているかどうか確認して下さい。

```
$ ssh root@somehost
# mkdir /root/.ssh
# chmod go-rwx /root/.ssh
# cat /root/key-dan.pub >> /root/.ssh/authorized_keys
# chmod go-rw /root/.ssh/authorized_keys
```

-

libvirt デーモン (libvirtd)

libvirt デーモンは仮想マシンの管理用の インターフェイスを提供します。**libvirtd** デーモンがインストール済みであり、それが管理したい全てのリモートホスト上で稼働して いなければなりません。Red Hat **kernel-xen** パッケージの使用には、 特別なカーネルと CPU ハードウェアのサポートが必要になります。詳細は 1章*システム要件*で確認して下さい。

```
$ ssh root@somehost
# chkconfig libvirtd on
# service libvirtd start
```

libvirtd と **SSH** の設定が終了した後は、リモートで仮想マシンへ アクセスして管理できるはずです。また、この時点で **VNC** を 使用してゲストへもアクセスできるはずです。

20.2. TLS と SSL を使用したリモート管理

TLS と SSL を使用して仮想マシンを管理することができます。TLS と SSL は より大きい拡張性を能えてくれますが、ssh よりも複雑です（「SSH を使用したリモート管理」 参照）。TLS と SSL は安全な接続用にウェブブラウザで 使用されている技術と同じです。**libvirt** 管理接続は 来信の接続用に TCP ポートを開きますが、それは **x509** 証明書を基にした安全な暗号化と認証を持っています。 更には、各ゲスト仮想マシン用の VNC コンソールは **x509** 証明書の認定を持つ TLS を 使うようにセットアップされます。

この方法は管理されているリモートマシン上にシェルアカウントを必要としません。 しかし、管理サービスや VNC コンソールにアクセスするために余分のファイアウォールルールが 必要になります。証明書剥奪一覧によりユーザーのアクセスを剥奪することが可能です。

virt-manager 用に TLS/SSL アクセスをセットアップする手順

以下の短いガイドはユーザーが初めての試みをしていること、及び TLS/SSL 証明書の 知識を持っていないことを想定しています。ユーザーが幸運にも証明書管理サーバーを 所有している場合は、最初のステップは多分飛ばすことができるでしょう。

libvirt サーバーセットアップ

証明書の作成についての詳細には、**libvirt website,** http://libvirt.org/remote.html を参照して下さい。

Xen VNC サーバー

Xen VNC サーバーは設定ファイル **/etc/xen/xend-config.sxp** を 編集することにより TLS を有効にすることができます。設定ファイル内の **(vnc-tls 1)** 設定パラメータのコメント化を外します。

/etc/xen/vnc ディレクトリは以下の３つのファイルが必要です:

- **ca-cert.pem** - CA 証明書
- **server-cert.pem** - CA によって署名されたサーバー証明書
- **server-key.pem** - サーバープライベートキー

これがデータチャンネルの暗号化を提供します。 クライアントが認証の形式としてそれ自身の **x509** 証明書を提示することを要求するのが妥当 でしょう。これを可能にするには、**(vnc-x509-verify 1)** パラメータの コメント化を外します。

virt-manager と virsh のクライアントセットアップ

クライアントのセットアップは少しここでは不均等です。TLS を介して **libvirt** 管理 API を有効にするには、CA と クライアントの証明書が **/etc/pki** 内に 配置されなければなりません。この詳細に関しては http://libvirt.org/remote.html をご覧下さい。

ホストに接続する時には、**virt-manager** ユーザーインターフェイス内で、 **SSL/TLS** トランスポートメカニズムオプションを使用します。

virsh 用に URI は以下のような形式を持ちます:

- KVM 対応の **qemu://hostname.guestname/system**
- Xen 対応の **xen://hostname.guestname/** .

VNC 用に SSL と TLS を有効にするには、証明書権限とクライアント証明書を、 **$HOME/.pki** に入れる必要があります。それは以下の ３つのファイルです:

- CA 又は、**ca-cert.pem** - CA 証明書
- **libvirt-vnc** 又は **clientcert.pem** - CA によって署名されたクライアント証明書。
- **libvirt-vnc** 又は **clientkey.pem** - クライアントのプライベートキーです。

20.3. トランスポートモード

リモートの管理には **libvirt** が以下のようなトランスポートモードをサポートします:

Transport Layer Security (TLS)

Transport Layer Security TLS 1.0 (SSL 3.1) で認証されて暗号化された TCP/IP ソケットは 通常、パブリックポート番号の内の１つでリッスンしています。これを使用するには、クライアントと サーバーの証明書を生成する必要があります。標準のポートは 16514 です。

UNIX ソケット

Unix ドメインソケットはローカルマシーン上でのみアクセス可能です。ソケットは暗号化 されておらず、認証のために UNIX 権限又は、SELinux を使用します。標準のソケット名は **/var/run/libvirt/libvirt-sock** と **/var/run/libvirt/libvirt-sock-ro** （読み込み専用接続）です。

SSH

Secure Shell protocol (SSH) 接続経由でのトランスポートです。Netcat （nc パッケージ）のインストールを必要とします。 libvirt デーモン (**libvirtd**) がリモートマシン上で 実行している必要があります。Port 22 が SSH アクセス用に開いていなければ なりません。なんらかの ssh キー管理（例えば、**ssh-agent** ユーティリティ）を 使用する必要があり、そうでないとパスワードを要求されます。

ext

ext パラメータはいずれかの外部プログラム用に使用 されるものです。これは libvirt の範疇にはない手法でリモートマシンに接続をします。 このパラメータはサポートされていません。

tcp

暗号化のない TCP/IP ソケットです。実稼働使用には推奨できません。これは 通常無効になっていますが、管理者はテスト目的や、信頼できるネットワーク上で 有効にすることができます。デフォルトのポートは 16509 です。

他に指定がない場合は、デフォルトのトランスポートは tls です。

リモート URI

Uniform Resource Identifier (URI) はリモートホストに接続するために virsh と libvirt により使用されます。 URI はまた、virsh コマンド用に --connect パラメータと 一緒にリモートホスト上で単独コマンドや移行を実行するのに使用されます。

libvirt URI は一般的な形式を取ります（角括弧 "[]" の中身はオプションの関数を示します）：

```
driver[+transport]://[username@][hostname][:port]/[path][?extraparameters]
```

外部の場所を目標とする為にトランスポートモード、又はホスト名が 供給される必要があります。

リモート管理パラメータのサンプル

- SSH トランスポート及び SSH ユーザー名 **ccurran** を使用して **towada** と言うホスト上のリモート Xen hypervisor に接続します。

  ```
  xen+ssh://ccurran@towada/
  ```

- TLS を使用して **towada** と言う名前のホスト上のリモート Xen hypervisor に接続します。

  ```
  xen://towada/
  ```

- TLS を使用してホスト **towada** 上のリモート Xen hypervisor に 接続します。***no_verify=1*** は libvirt にサーバーの証明書を 検証しないように伝えます。

  ```
  xen://towada/?no_verify=1
  ```

- SSH を使用して、ホスト **towada** 上のリモート KVM hypervisor に 接続します。

  ```
  qemu+ssh://towada/system
  ```

テスト用サンプル

- 標準でない UNIX ソケットを使用してローカルの KVM hypervisor に接続します。 この場合 Unix ソケットへの完全なパスは 明示的に供給されます。

  ```
  qemu+unix:///system?socket=/opt/libvirt/run/libvirt/libvirt-sock
  ```

- ポート 5000 で IP アドレス 10.1.1.10 を持つ サーバーへの 暗号化のない TCP/IP 接続を 使用して libvirt デーモンへ接続します。これは、デフォルト設定を持つテストドライバーを 使用します。

  ```
  test+tcp://10.1.1.10:5000/default
  ```

その他の URI パラメータ

他のパラメータはリモート URI へ追記できます。以下の表 表20.1「その他の URI パラメータ」 は 認識されているパラメータを説明しています。その他のパラメータはすべて無視されます。 パラメータの値は、URI エスケープ（パラメータの前に疑問符 (?) が付けてあり、他の特殊文字は URI 形式に変換）してなければなりません。

表20.1 その他の URI パラメータ

名前	トランスポートモード	説明	使用法のサンプル
name	全てのモード	リモート virConnectOpen 関数に渡される名前です。この名前は通常、リモート URI からトランスポート、ホスト名、 ポート番号、ユーザー名、及び余分のパラメータを削除したものですが、一部の複雑なケースでは、名前を明示的に 供給するのが適切な場合もあります。	name=qemu:///system
command	ssh と ext	外部コマンドです。外部のトランスポートにはこれが必須となります。 ssh 用には デフォルトは ssh です。コマンドの為に PATH が検索されます。	command=/opt/openssh/bin/ssh
socket	unix と ssh	UNIX ドメインソケットへのパスです。これはデフォルトを上書きします。 ssh トランスポートには、これがリモート netcat コマンドに渡されます（netcat 参照）。	socket=/opt/libvirt/run/libvirt/libvirt-sock

名前	トランスポートモード	説明	使用法のサンプル
netcat	ssh	**netcat** コマンドはリモートシステムに接続するのに 使用できます。デフォルトの netcat パラメータは **nc** コマンドを 使用します。SSH トランスポート用には、libvirt が以下の形式を使用して SSH コマンドを構築します: `command -p port [-l username] hostname netcat -U socket` ***port***、***username***、及び ***hostname*** パラメータはリモート URI の一部として 指定できます。***command***、***netcat***、及び ***socket*** は他の追加のパラメータ 由来のものです。	netcat=/opt/netcat/bin/nc
no_verify	tls	ゼロ以外の値にセットしてある場合、これはサーバーの証明書のクライアントチェックを無効にします。クライアントの証明書か、IP アドレスのサーバーチェックを無効にするには、libvirtd 設定を変更する必要があることに注意して下さい。	no_verify=1
no_tty	ssh	ゼロ以外の値にセットしてある場合、（ssh-agent 又は同類の使用で）自動的にリモートマシンに ログインできない場合に、ssh がパスワードを要求することを止めます。例えば、libvirt を使用するグラフィカルプログラム内のターミナルにアクセスを持たない時に これを使用します。	no_tty=1

パート V. 仮想化リファレンスガイド

仮想化のコマンド、システムツール、アプリケーション、及びシステム参照

これらの章では、Red Hat Enterprise Linux に収納されている仮想化コマンド、システムツール、 及びアプリケーションの詳細説明を提供しています。これらの章は高度な機能とその他の特徴に 関する情報を要求するユーザー向けにデザインされています。

第21章 仮想化のツール

Xen を実行するシステムに役に立つ仮想化の管理、デバッギング、及びネットワーキングの 為のツールリストを以下に示します。

システム管理ツール

- **vmstat**
- **iostat**
- **lsof**

```
# lsof -i :5900
xen-vncfb 10635  root  5u  IPv4 218738  TCP
grumble.boston.redhat.com:5900 (LISTEN)
```

- **qemu-img**

高度なデバッギングツール

- **systemTap**
- **crash**
- **xen-gdbserver**
- **sysrq**
- **sysrq t**
- **sysrq w**
- **sysrq c**

ネットワーキング

brtcl

- ```
  # brctl show
  bridge name  bridge id            STP enabled    interfaces
  xenbr0       8000.feffffffffff    no             vif13.0
                                                   pdummy0
                                                   vif0.0
  ```

- ```
  # brctl showmacs xenbr0
  port no  mac addr                is local?       aging timer
    1      fe:ff:ff:ff:ff:ff       yes             0.00
  ```

- ```
  # brctl showstp xenbr0
  xenbr0
  bridge id              8000.feffffffffff
  designated root        8000.feffffffffff
  root port              0                    path cost
  ```
  ```

```
0
max age                 20.00                   bridge max age
20.00
hello time              2.00                    bridge hello time
2.00
forward delay           0.00                    bridge forward delay
0.00
aging time             300.01
hello timer             1.43                    tcn timer
0.00
topology change timer   0.00                    gc timer
0.02
flags

vif13.0 (3)
port id                 8003                    state
forwarding
designated root         8000.feffffffffff       path cost
100
designated bridge       8000.feffffffffff       message age timer
0.00
designated port         8003                    forward delay timer
0.00
designated cost         0                       hold timer
0.43
flags

pdummy0 (2)
port id                 8002                    state
forwarding
designated root         8000.feffffffffff       path cost
100
designated bridge       8000.feffffffffff       message age timer
0.00
designated port         8002                    forward delay timer
0.00
designated cost         0                       hold timer
0.43
flags

vif0.0 (1)
port id                 8001                    state
forwarding
designated root         8000.feffffffffff       path cost
100
designated bridge       8000.feffffffffff       message age timer
0.00
designated port         8001                    forward delay timer
0.00
designated cost         0                       hold timer
0.43
flags
```

- **ifconfig**
- **tcpdump**

KVM ツール

- **ps**
- **pstree**
- **top**
- **kvmtrace**
- **kvm_stat**

Xen ツール

- **xentop**
- **xm dmesg**
- **xm log**

第22章 VIRSH でゲストを管理

virsh はゲストと hypervisor を管理するための コマンドラインインターフェイスです。

virsh ツールは **libvirt** 管理 API を 土台にして構築されており、**xm** コマンドとグラフィカルゲストマネージャ (**virt-manager**) への代替として機能します。 **virsh** は特別権限のないユーザーにより読み込み専用モードで 使用可能です。 **virsh** を使用してゲストマシン用のスクリプトを 実行することができます。

virsh コマンドのクイックリファレンス

以下の表では、全ての virsh コマンドラインオプションのクイックリファレンスを 提供します。

表22.1 ゲスト管理のコマンド

コマンド	説明
help	基本的なヘルプ情報を表示します。
list	全てのゲストを一覧表示します。
dumpxml	ゲスト用の XML 設定ファイルを出力します。
create	XML 設定ファイルからゲストを作成して新規のゲストを開始します。
start	停止中のゲストを開始します。
destroy	ゲストを強制的に停止します。
define	ゲスト用の XML 設定ファイルを出力します。
domid	ゲストの ID を表示します。
domuuid	ゲストの UUID を表示します。
dominfo	ゲスト情報を表示します。
domname	ゲスト名を表示します。
domstate	ゲストの状態を表示します。
quit	対話式のターミナルを終了します。
reboot	ゲストを再起動します。
restore	ファイル内に以前に保存されているゲストを復元します。
resume	休止中のゲストを復帰します。

コマンド	説明
save	ゲストの現在の状態をファイルに保存します。
シャットダウン中	ゲストを丁寧にシャットダウンします。
suspend	ゲストを休止します。
undefine	ゲストに関連のファイルをすべて削除します。
migrate	ゲストを別のホストに移行します。

以下の **virsh** コマンドはゲストと hypervisor リソースを 管理します:

表22.2 リソース管理のオプション

コマンド	説明
setmem	ゲストのために割り当てたメモリーを設定します。
setmaxmem	hypervisor 用の最大メモリー限度を設定します。
setvcpus	ゲストに割り当てた仮想 CPU の数を変更します。
vcpuinfo	ゲストに関して仮想 CPU 情報を表示します。
vcpupin	ゲストの仮想 CPU 同類を制御します。
domblkstat	実行中ゲストのブロックデバイス統計を表示します。
domifstat	実行中のゲストのネットワークインターフェイス統計を表示します。
attach-device	XML ファイル内のデバイス定義を使用してゲストへデバイスを添付します。
attach-disk	新規のディスクデバイスをゲストに添付します。
attach-interface	新規のネットワークインターフェイスをゲストに添付します。
detach-device	ゲストからデバイスを分離し、**attach-device** コマンドと 同じ種類の XML 記述を提示します。
detach-disk	ゲストからディスクデバイスを分離します。

コマンド	説明
detach-interface	ゲストからネットワークインターフェイスを分離します。

以下にその他の **virsh** オプションを示します:

表22.3 その他のオプション

コマンド	説明
version	**virsh**のバージョンを表示します。
nodeinfo	hypervisor に関する情報を出力します。

Hypervisor への接続

virsh で hypervisor セッションへ接続します:

```
# virsh connect {hostname OR URL}
```

ここで、**<name>** は hypervisor のマシン名です。読み込み専用の接続を開始したい場合、上記のコマンドに **-readonly** を追記します。

仮想マシン XML ダンプ（設定ファイル）を作成

virsh でゲストの XML 設定ファイルを出力します:

```
# virsh dumpxml {domain-id, domain-name or domain-uuid}
```

このコマンドはゲストの XML 設定ファイルを標準出力 (**stdout**) に 出力します。出力をファイルにパイプすることでデータを保存できます。*guest.xml* と言うファイルへ出力をパイプする例として：

```
# virsh dumpxml GuestID > guest.xml
```

と出来ます。このファイル **guest.xml** はゲストを再作成できるものです。 (ゲスト設定ファイルの編集 を参照) この XML 設定ファイルを編集して追加のデバイスを設定したり、 又は追加のゲストを導入したりすることも出来ます。**virsh dumpxml** を使用した ファイルの修正に関する情報には、「virsh を用いた XML 設定ファイルの使用」 を参照して下さい。

virsh dumpxml 出力の例:

```
# virsh dumpxml r5b2-mySQL01
<domain type='xen' id='13'>
    <name>r5b2-mySQL01</name>
    <uuid>4a4c59a7ee3fc78196e4288f2862f011</uuid>
    <bootloader>/usr/bin/pygrub</bootloader>
    <os>
        <type>linux</type>
        <kernel>/var/lib/libvirt/vmlinuz.2dgnU_</kernel>
        <initrd>/var/lib/libvirt/initrd.UQafMw</initrd>
        <cmdline>ro root=/dev/VolGroup00/LogVol00 rhgb quiet</cmdline>
```

```
    </os>
    <memory>512000</memory>
    <vcpu>1</vcpu>
    <on_poweroff>destroy</on_poweroff>
    <on_reboot>restart</on_reboot>
    <on_crash>restart</on_crash>
    <devices>
        <interface type='bridge'>
            <source bridge='xenbr0'/>
            <mac address='00:16:3e:49:1d:11'/>
            <script path='vif-bridge'/>
        </interface>
        <graphics type='vnc' port='5900'/>
        <console tty='/dev/pts/4'/>
    </devices>
</domain>
```

設定ファイルからゲストを作成

ゲストは XML 設定ファイルから作成することができます。以前に作成されているゲストから 既存の XML をコピーするか、又は **dumpxml** オプションを使用します。 (仮想マシン XML ダンプ（設定ファイル）を作成 参照) **virsh** を 使用して XML ファイルからゲストを作成するには:

```
# virsh create configuration_file.xml
```

ゲスト設定ファイルの編集

dumpxml オプション (仮想マシン XML ダンプ（設定ファイル）を作成 を参照) を使用する代わりに、ゲストは、それが稼働中でも、オフライン中でも編集することができます。**virsh edit** コマンドがこの機能を提供します。例えば、***softwaretesting*** と 言う名前のゲストを編集するには:

```
# virsh edit softwaretesting
```

これがテキストエディタを開きます。デフォルトのテキストエディタは **$EDITOR** シェルパラメータです（デフォルトで **vi** にセット）。

ゲストの休止

virsh を使用してゲストを休止します:

```
# virsh suspend {domain-id, domain-name or domain-uuid}
```

ゲストが休止状態にある時も、まだシステム RAM は消費していますが、プロセッサリソースは 消費しません。休止中にはディスク、又はネットワークの I/O はありません。この運用は直ちに 実行されますので、ゲストは **resume** (ゲストの復帰) オプションで再開始する必要があります。

ゲストの復帰

resume オプションと共に **virsh** を 使用して休止中のゲストを復帰します:

```
# virsh resume {domain-id, domain-name or domain-uuid}
```

この操作は直ちに反映されて、ゲストパラメータは **suspend** と **resume** 操作用に保持されます。

ゲストを保存

virsh コマンドを使用してゲストの現在の状態をファイルに 保存します:

```
# virsh save {domain-name, domain-id or domain-uuid} filename
```

これで、指定した仮想マシンが停止し、データはファイルに保存されます。この時、ゲストで使用しているメモリーの容量によっては少々時間がかかる可能性があります。このゲストの状態は **restore** (ゲストの復帰) オプションで復帰することができます。保存は休止と似ていますが、単なる休止の代わりに ゲストの現在の状態が保存されます。

ゲストの復帰

virsh save コマンド(ゲストを保存) で以前に保存されたゲストを **virsh** を使用して 復帰します:

```
# virsh restore filename
```

これが、保存していたゲストを再開始しますが、少々時間を取ります。ゲストの名前と UUID は保持されていますが、新しい id に割り当てされます。

ゲストのシャットダウン

virsh コマンドを使用してゲストをシャットダウンします:

```
# virsh shutdown {domain-id, domain-name or domain-uuid}
```

ゲストの再起動の仕方は ゲストの設定ファイルの **on_shutdown** パラメータを修正することで制御できます。

ゲストの再起動

virsh コマンドを使用してゲストを再起動します:

```
#virsh reboot {domain-id, domain-name or domain-uuid}
```

ゲストの再起動の仕方は、ゲストの設定ファイルの **on_reboot** エレメントを修正することで制御できます。

ゲストを強制的に停止

virsh コマンドを使用してゲストの停止を強制します:

```
# virsh destroy {domain-id, domain-name or domain-uuid}
```

このコマンドは素早くに強引にシャットダウンして、指定したゲストを停止します。 **virsh destroy** を使用すると、ゲストのファイルシステムを 破損するかも知れません。**destroy** オプションは、ゲストが 反応しない時にのみ使用して下さい。para-virtualized ゲスト用には代わりに **shutdown** オプション (ゲストのシャットダウン) を使用します。

ゲストのドメイン ID を取得

ゲストのドメイン ID を取得するには:

```
# virsh domid {domain-name or domain-uuid}
```

ゲストのドメイン名を取得

ゲストのドメイン名を取得するには:

```
# virsh domname {domain-id or domain-uuid}
```

ゲストの UUID を取得

ゲストの UUID（Universally Unique Identifier）を取得するには:

```
# virsh domuuid {domain-id or domain-name}
```

virsh domuuid の出力の例:

```
# virsh domuuid r5b2-mySQL01
4a4c59a7-ee3f-c781-96e4-288f2862f011
```

ゲスト情報の表示

ゲストのドメイン ID、ドメイン名、あるいは UUID と共に **virsh** を 使用すると、指定したゲストの情報を表示することができます:

```
# virsh dominfo {domain-id, domain-name or domain-uuid}
```

以下に **virsh dominfo** 出力の例を示します:

```
# virsh dominfo r5b2-mySQL01
id:             13
name:           r5b2-mysql01
uuid:           4a4c59a7-ee3f-c781-96e4-288f2862f011
os type:              linux
state:          blocked
cpu(s):         1
cpu time:             11.0s
max memory:     512000 kb
used memory:    512000 kb
```

ホスト情報の表示

ホストに関する情報を表示するには:

```
# virsh nodeinfo
```

virsh nodeinfo 出力の例:

```
# virsh nodeinfo
CPU model                    x86_64
CPU (s)                      8
CPU frequency                2895 Mhz
CPU socket(s)                2
Core(s) per socket           2
Threads per core:            2
Numa cell(s)                 1
Memory size:                 1046528 kb
```

これは、ノード情報と仮想化プロセスに対応するマシンを表示します。

ゲストの表示

virsh コマンドを使用してゲストの一覧とその現在状態を 表示するには:

```
# virsh list
```

使用できる他のオプションには以下があります:

活動していないゲスト（定義されていても現在活動していないゲスト）の一覧表示する **--inactive** オプション 。そして

全てのゲストを一覧表示する **--all** オプション。 例えば:

```
# virsh list --all
 Id Name                 State
----------------------------------
  0 Domain-0             running
  1 Domain202            paused
  2 Domain010            inactive
  3 Domain9600           crashed
```

virsh list からの出力は６つの状態の１つとして 分類されます（以下の一覧）。

- **running** 状態は CPU 上で現在活動中のゲストを 示します。
- **blocked** として表示してあるゲストは阻止されており、 実行していないか、又は実行不可能です。これは I/O 待ちのゲスト（旧来の wait 状態）か、 スリープモードのゲストがその要因です。
- **paused** 状態は休止中のドメインを一覧表示します。 これは、管理者が**virt-manager**、**xm pause**、 又は **virsh suspend** で、**pause** ボタンを使用する ことで発生します。ゲストが休止している時は、メモリーとその他のリソースを消費しますが、スケジュールと hypervisor からの CPU リソースには無視できる量です。
- **shutdown** 状態は シャットダウンプロセスにある ゲスト用のものです。ゲストはシャットダウン信号を受けてその運用を丁寧に終了するプロセスに 入るべき状態です。これは全てのゲストオペレーティングシステムでは機能しないかも知れません。 一部のオペレーティングシステムはこの信号に良く反応しません。
- **dying** 状態のドメインはご臨終のプロセスにある ものです。これはドメインがシャットダウンやクラッシュを完全に終了していない状態を指します。
- **crashed** の場合、ゲストは実行中に障害を受け、もう 実行していない状態です。この状態はクラッシュ時にゲストが再スタートしないように設定されている 場合にのみ発生します。

仮想 CPU 情報の表示

virsh を使用してゲストからの仮想 CPU の情報を 表示するには:

```
# virsh vcpuinfo {domain-id, domain-name or domain-uuid}
```

virsh vcpuinfo 出力の例:

```
# virsh vcpuinfo r5b2-mySQL01
VCPU:           0
CPU:            0
```

```
State:          blocked
CPU time:       0.0s
CPU Affinity:   yy
```

仮想 CPU 類似物の設定

物理 CPU を使用して、仮想 CPU の類似物を設定するには:

```
# virsh vcpupin domain-id vcpu cpulist
```

domain-id パラメータはゲストの ID 番号、又は名前です。

vcpu パラメータは、ゲストに割り当てられた仮想 CPU の数を示します。**vcpu** パラメータは必須項目です。

cpulist パラメータは、コンマで区切られた物理 CPU の 識別子番号の一覧です。**cpulist** パラメータはどの物理 CPU で VCPU が稼働するかを決定します。

仮想 CPU カウントの設定

virsh を使用してゲストに割り当てられた CPU の数を 修正するには:

```
# virsh setvcpus {domain-name, domain-id or domain-uuid} count
```

新しい ***count*** 値はゲストが作成された時に指定されたカウントを 超過することは出来ません。

メモリー割り当ての設定

virsh を使用してゲストのメモリー割り当てを修正するには:

```
# virsh setmem {domain-id or domain-name} count
```

count はキロバイトで指定する必要があります。新しいカウントは ゲストを作成した時に指定した数量を超えることができないことに注意して下さい。64 MB より低い値はほとんどのオペレーティングシステムでは多分機能しないでしょう。最大メモリーを上げても 活動中ゲストに影響することはありません。新しい値がより低い場合、利用可能なメモリーは 縮小し、ゲストはクラッシュする可能性があります。

ゲストブロックデバイス情報の表示

virsh domblkstat を使用すると稼働中のゲストの ブロックデバイス統計が表示できます。

```
# virsh domblkstat GuestName block-device
```

ゲストネットワークデバイス情報の表示

virsh domifstat を使用すると、稼働中のゲストの ネットワークインターフェイス統計が表示できます。

```
# virsh domifstat GuestName interface-device
```

virsh でゲスト移行を管理

ゲストは **virsh** を使用して別のホストへ移行することが できます。ドメインを別のホストへ移行します。ライブ移行用には **--live** を追加します。 **migrate** コマンドは以下の形式のパラメータを受け付けます:

```
# virsh migrate --live GuestName DestinationURL
```

--live パラメータはオプションです。ライブ移行用には***--live*** パラメータを追加します。

GuestName パラメータは移行したいゲストの名前を示します。

DestinationURL パラメータは移行先システムの URL 又はホスト名です。移行先システムは以下を必要とします:

- Red Hat Enterprise Linux の同じバージョン
- 同じ hypervisor (KVM か Xen)、それに
- **libvirt** サービスが開始する必要があります。

コマンドが入力された後は、目的地システムの root パスワードを 要求されます。

仮想ネットワークの管理

このセクションでは、**virsh** コマンドを使用した仮想化 ネットワークの管理を説明します。仮想化 ネットワークを一覧表示するには:

```
# virsh net-list
```

このコマンドは以下のような出力を出します:

```
# virsh net-list
Name                 State      Autostart
-----------------------------------------
default              active     yes
vnet1                     active     yes
vnet2                     active     yes
```

特定の仮想ネットワークの為のネットワーク情報を表示するには:

```
# virsh net-dumpxml NetworkName
```

この画面は指定された仮想ネットワークに関する情報を XML 形式で表示します:

```
# virsh net-dumpxml vnet1
<network>
  <name>vnet1</name>
  <uuid>98361b46-1581-acb7-1643-85a412626e70</uuid>
  <forward dev='eth0'/>
  <bridge name='vnet0' stp='on' forwardDelay='0' />
  <ip address='192.168.100.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.100.128' end='192.168.100.254' />
    </dhcp>
  </ip>
</network>
```

仮想ネットワークの管理に使用される他の **virsh** コマンドを以下に示します:

- **virsh net-autostart *[network name]*** — *network-name* で指定された ネットワークを自動開始します。
- **virsh net-create *XMLfile*** — 既存の XML ファイルを使用して新規のネットワークを生成して開始します。
- **virsh net-define *XMLfile*** — 既存の XML ファイルから新規のネットワークデバイスを生成しますが開始しません。
- **virsh net-destroy *network-name*** — *network-name* として指定された ネットワークを破棄します。
- **virsh net-name *networkUUID*** — 指定された *networkUUID* をネットワーク名に変換します。
- **virsh net-uuid *network-name*** — 指定された *network-name* をネットワーク UUID に変換します。
- **virsh net-start *nameOfInactiveNetwork*** — 休止中のネットワークを開始します。
- **virsh net-undefine *nameOfInactiveNetwork*** — 休止中のネットワークを定義解除します。

第23章 仮想マシンマネージャ(VIRT-MANAGER) でゲストを管理する

このセクションでは、仮想マシンマネージャ (**virt-manager**) ウィンドウ、ダイアログボックス、各種 GUI 制御などを説明しています。

virt-manager はユーザーのシステム上とリモートマシン上の hypervisor とゲストのグラフィカル表示を提供します。para-virtualized と完全仮想化の両方の ゲストを定義するのにも **virt-manager** を使用できます。そして **virt-manager** は以下を含む、仮想化管理のタスクも演じることができます:

- メモリーの割り当て、
- 仮想 CPU の割り当て、
- 動作パフォーマンスの監視、
- 仮想化ゲストの保存と復元、休止と復帰、及びシャットダウンと開始、
- テキストとグラフィカルのコンソールへのリンク、
- ライブとオフラインの移行。

23.1. 開放接続のウィンドウ

ウィンドウがまず表示されてユーザーに対して hypervisor セッションを選択するように催促します。特権の無いユーザーは読み込み専用のセッションを開始できます。Root ユーザーは完全な 読み込み/書き込みステータスでセッションを開始できます。通常の使用には、**ローカル Xen ホスト** オプションか、QEMU (KVM 用) を選択して下さい。

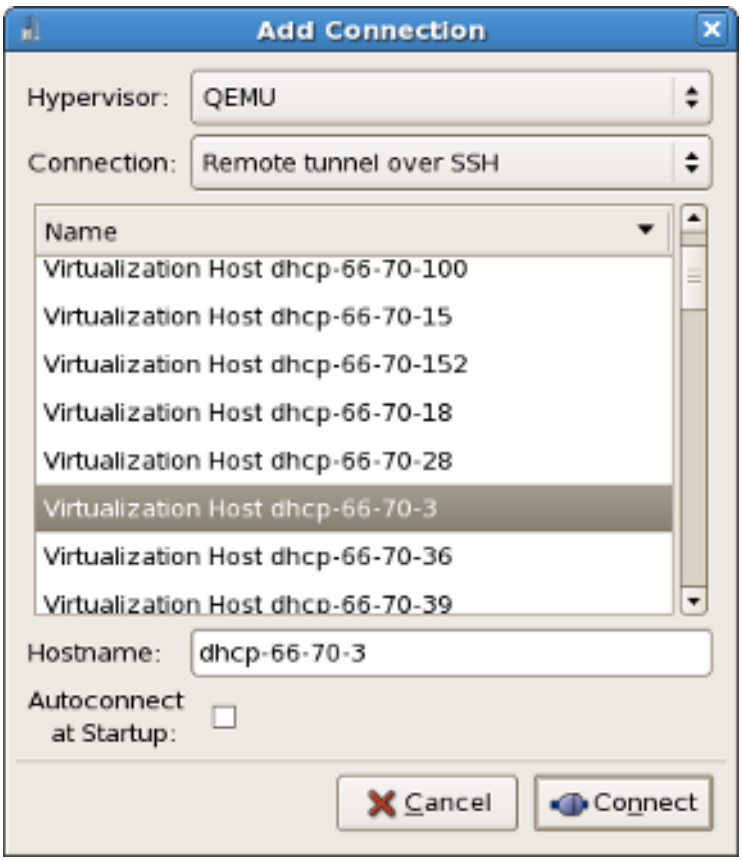


図23.1 仮想マシンマネージャの接続ウィンドウ

23.2. 仮想マシンマネージャの主要ウィンドウ

この主要ウィンドウは全ての稼動中仮想マシンと現在それらに割り当てられているリソースを表示します(domain0 を含む)。表示するフィールドはユーザーが決定できます。希望のバーチャルマシンをダブルクリックすると、そのマシン用の特定のコンソールが出て来ます。仮想マシンを選択して **詳細** ボタンをダブルクリックすると、そのマシンの詳細ウィンドウが表示されます。また、**ファイル** メニューにアクセスして新規の仮想マシンを作成することもできます。

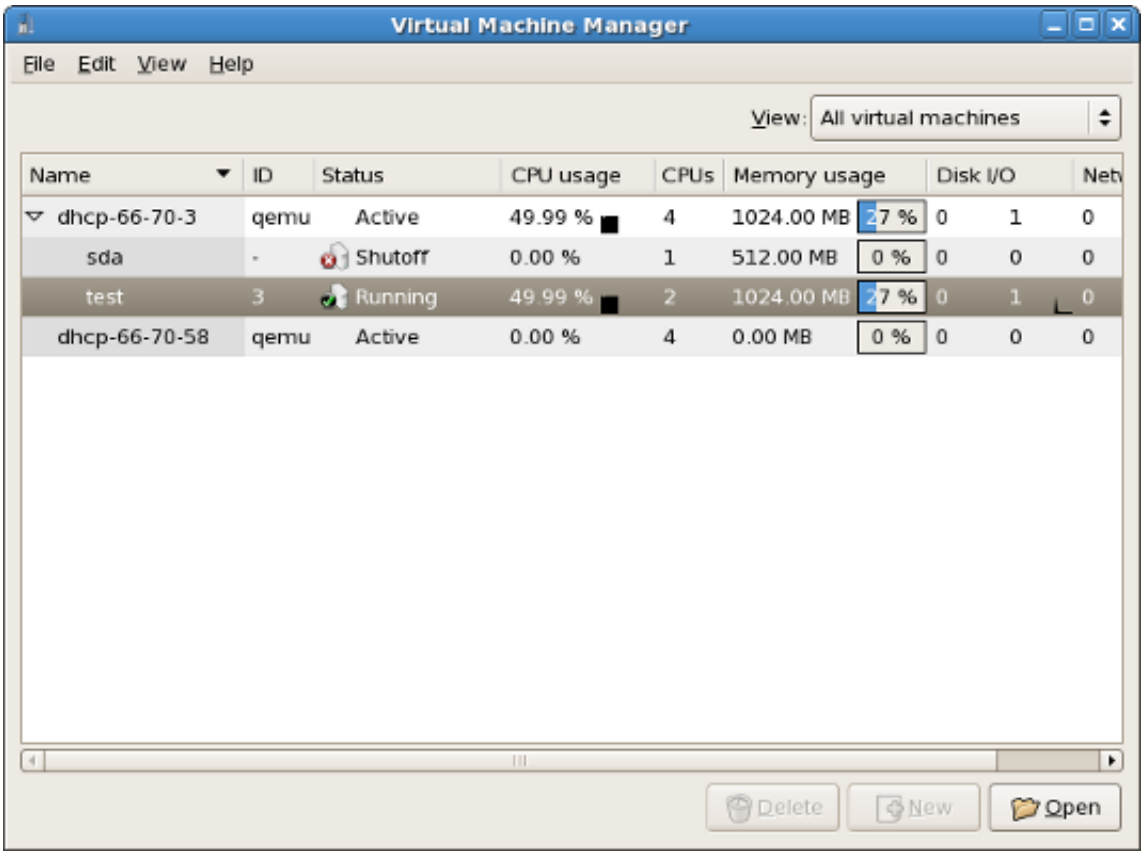


図23.2 仮想マシンマネージャの主要ウィンドウ

23.3. 仮想マシンマネージャの詳細ウィンドウ

このウィンドウでは、**virt-manager** で利用できる ゲストのライブリソース活用データのグラフと統計が表示されます。UUID フィールドは 仮想マシンのグローバルな特有識別子を表示します。

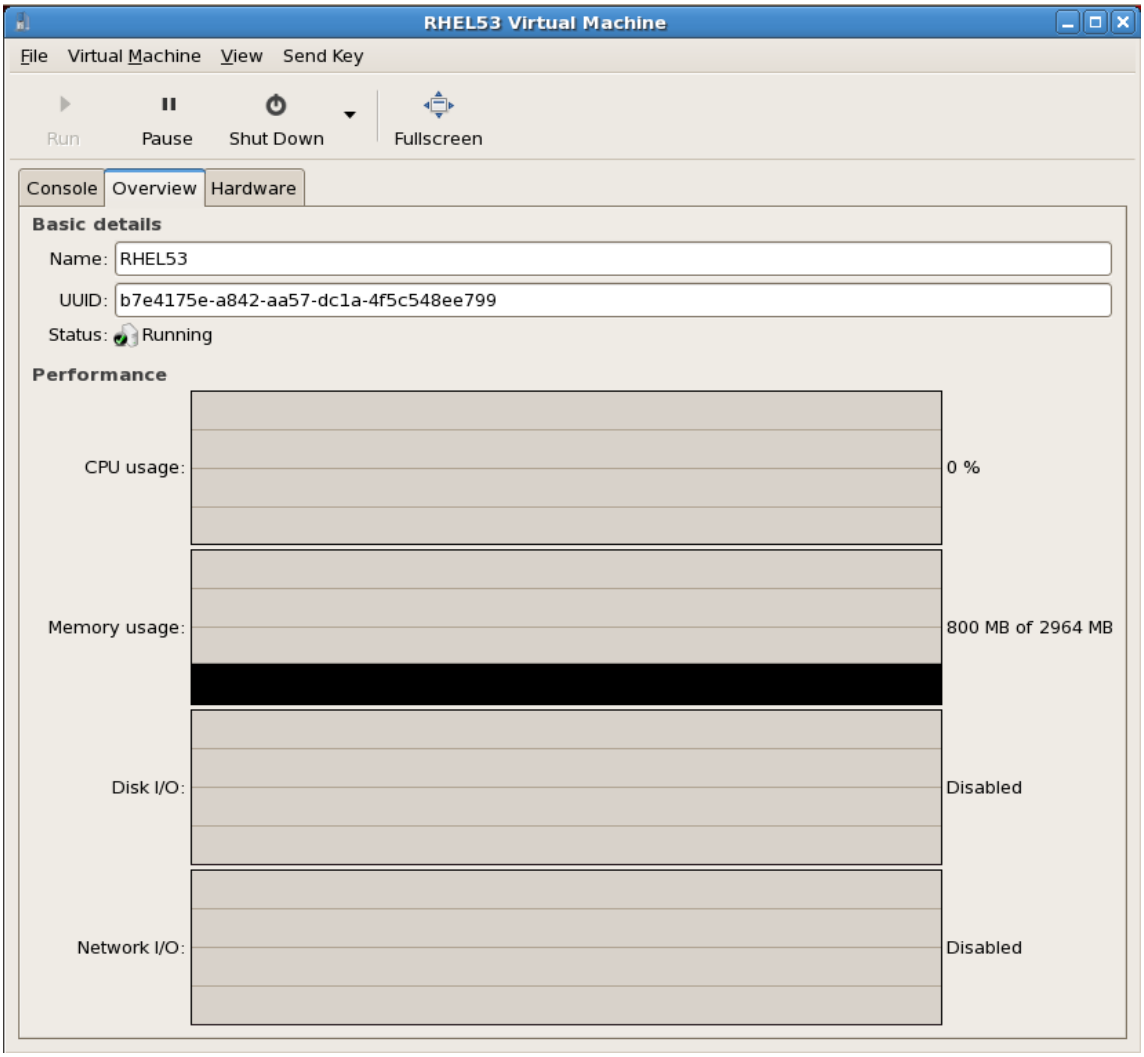


図23.3 `virt-manager` の詳細ウィンドウ

23.4. 仮想マシングラフィカルコンソール

このウィンドウには、仮想マシンのグラフィカルコンソールが表示されます。Paravirtual と完全仮想化のゲストは異なる技術を使用してそれらのローカル 仮想フレームバッファをエキスポートしますが、両方の技術共、**VNC** を 使用して仮想マシンマネージャのコンソールウィンドウがそれを利用できるようにします。仮想マシンが 認証を必要とする場合、仮想マシンのグラフィカルコンソールは表示をする前に ユーザーにパスワードを要求します。

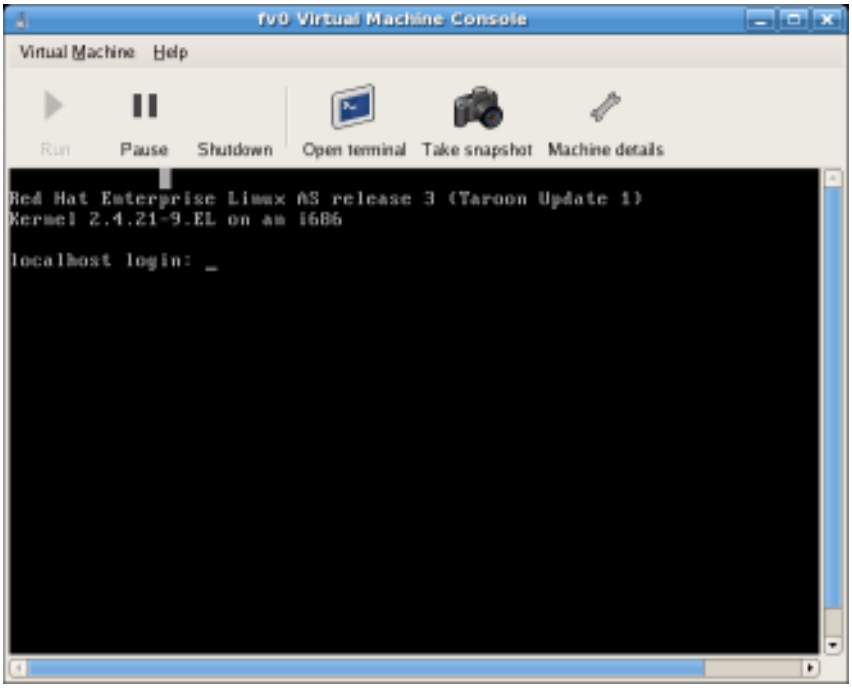
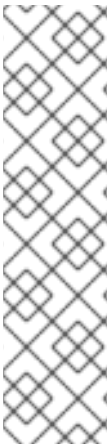


図23.4 グラフィカルコンソールウィンドウ



注記

VNC は多くのセキュリティ専門家から安全でないと考慮されていますが、数種の変更がなされており、Red Hat enterprise Linux 上での仮想化用の VNC に於いて安全な使用を可能にしています。ゲストマシンはローカルホスト (**dom0**) の ループバックアドレス (**127.0.0.1**) しか リッスンしません。このことが、ホスト上でシェル権限を持つ人達のみが VNC 経由で virt-manager と仮想マシンにアクセスできることを確実にしています。

20章 *仮想化ゲストのリモート管理*内の 案内に従えばリモート管理は実行できます。TLS はゲストとホストのシステム管理に於いてエンタープライズレベルの セキュリティを提供できます。

使用中のローカルデスクトップはキー組み合わせ操作 (例えば、Ctrl+Alt+F11) を阻止してそれがゲストマシンに送信されることを防止できます。ユーザーは **virt-manager** の スティッキーキー機能を使用してこれらの組み合わせを送信することができます。まず、いずれかの援助キー(Ctrl か Alt など) を三回押して、指定キーが次の非援助キー操作までアクティブと認識されるようにします。そうすると、'Ctrl Ctrl Ctrl Alt+F1' の組み合わせを押すことで、Ctrl-Alt-F11 をゲストに送ることができます。

23.5. VIRT-MANAGER の開始

virt-manager のセッションを開始するには、 **アプリケーション** メニューを開いて、それから **システムツール** メニュー、**仮想マシンマネージャ (virt-manager)** と進んで行きます。

virt-manager の主要ウィンドウが表示されます。

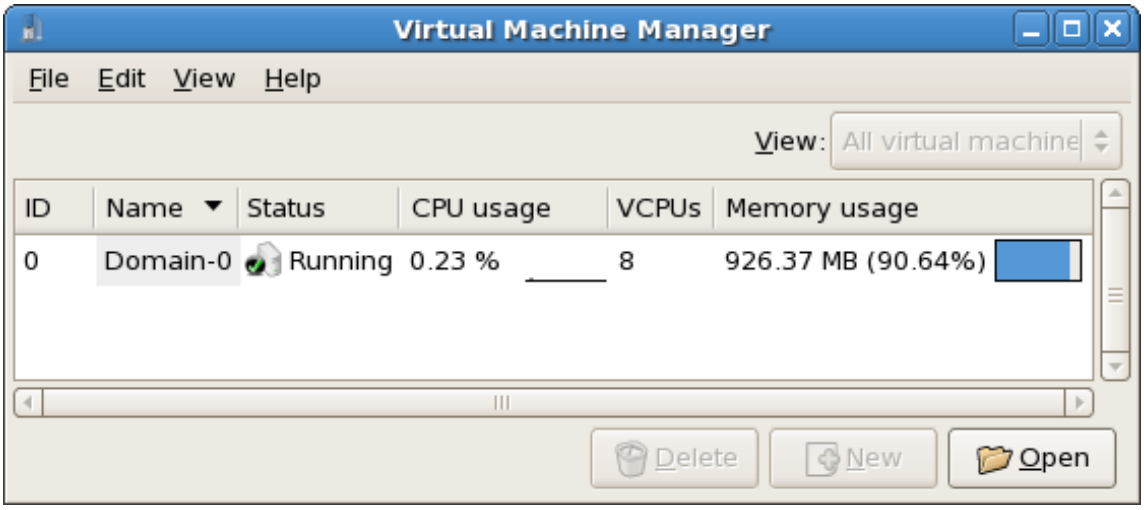


図23.5 `virt-manager` の開始

別の方法として、次のコマンドで示してあるように、**ssh** を使用することにより **`virt-manager`** を リモートで開始することができます:

```
ssh -X ホストのアドレス[remotehost]# virt-manager
```

ssh を使用した仮想マシンとホストの管理は「SSH を使用したリモート管理」でより詳細に 説明してあります。

23.6. 保存したマシンの復元

仮想マシンマネージャを開始した後は、システム上の全ての仮想マシンが主要ウィンドウに表示されます。Domain0 はユーザーのホストシステムです。マシンの表示がない場合は、現在、システム上で実行中のマシンがないと言う意味になります。

以前の保存セッションを復元するには:

1. **ファイル** メニューから **保存したマシンの復元** を選択します。

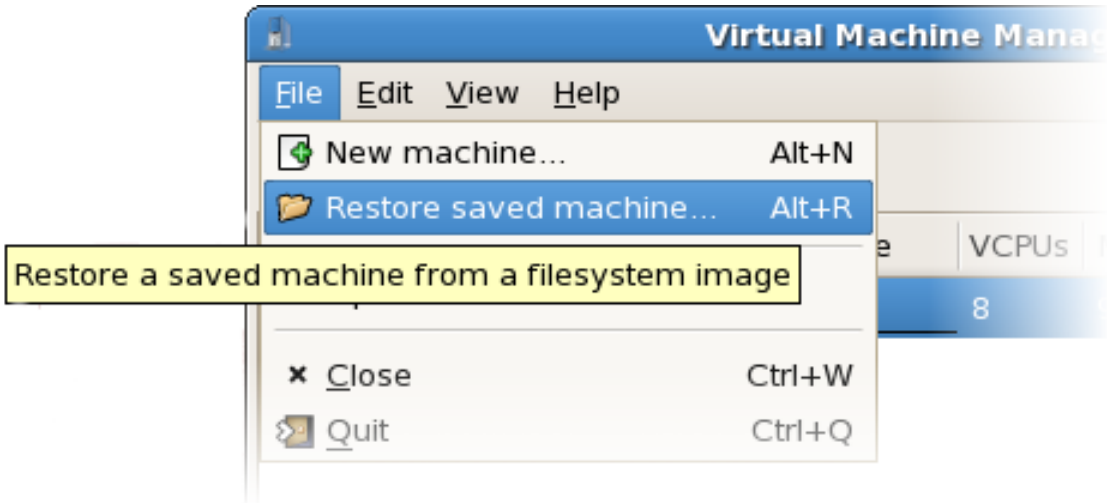


図23.6 仮想マシンの復元

2. **仮想マシンの復元** の主要ウィンドウが出ます。

3. 該当のディレクトリに移動して、保存したセッションファイルを選択します。

4. **開く** をクリックします。

保存してある仮想システムが仮想マシンマネージャの主要ウィンドウに出て来ます。

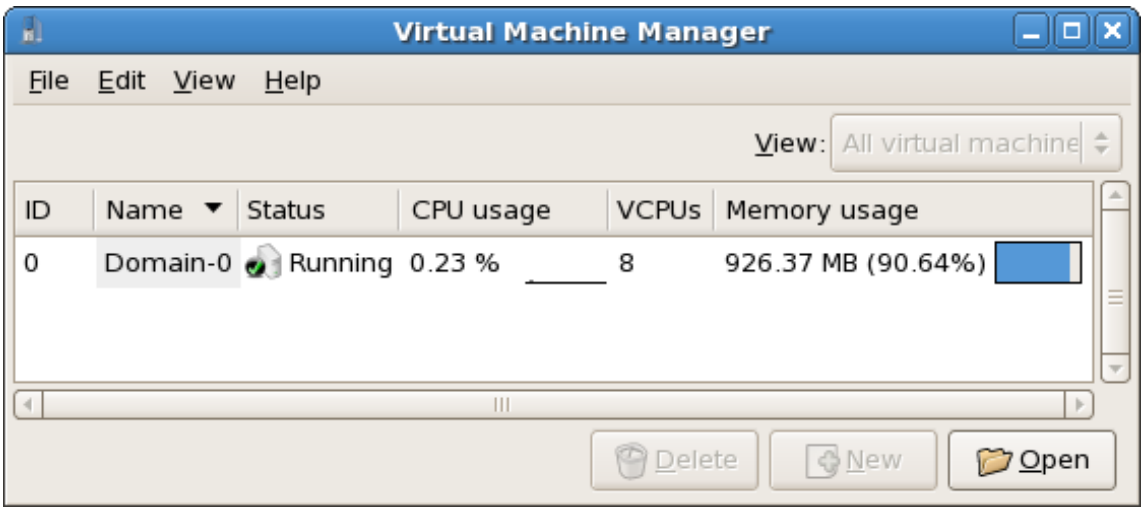


図23.7 復元された仮想マシンマネージャのセッション

23.7. ゲストの詳細表示

仮想マシンモニターを使用して、システム上の仮想マシン用の活動データ情報を表示することができます。

仮想システムの詳細を表示するには:

1. 仮想マシンマネージャの主要ウィンドウで、表示したい仮想マシンを強調表示します。

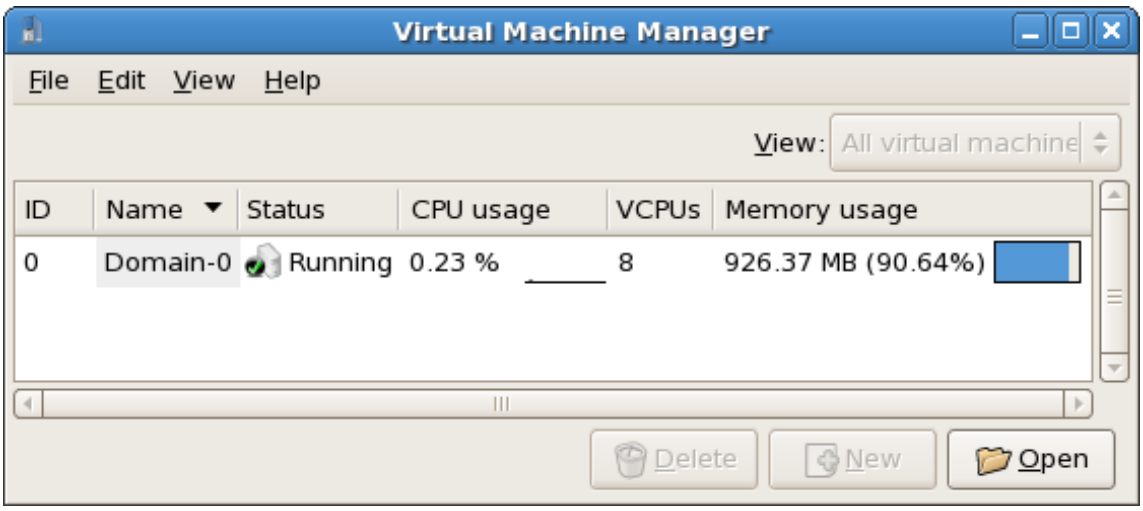


図23.8 表示する仮想マシンを選択

2. 仮想マシンマネージャの **編集** メニューから、**マシンの詳細** を選択します(又は、仮想マシンマネージャの主要ウィンドウの底辺にある **詳細** ボタンをクリックします)。

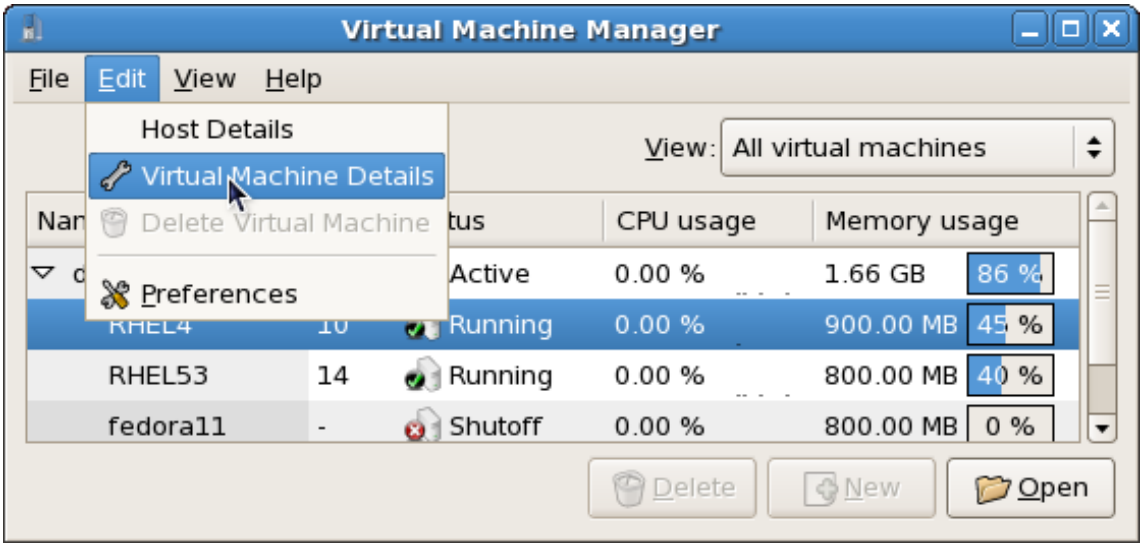


図23.9 仮想マシンの詳細メニューの表示

仮想マシンの詳細概要ウィンドウが現われます。このウィンドウは、指定したドメイン用の CPU とメモリーの使用量を要約表示します。

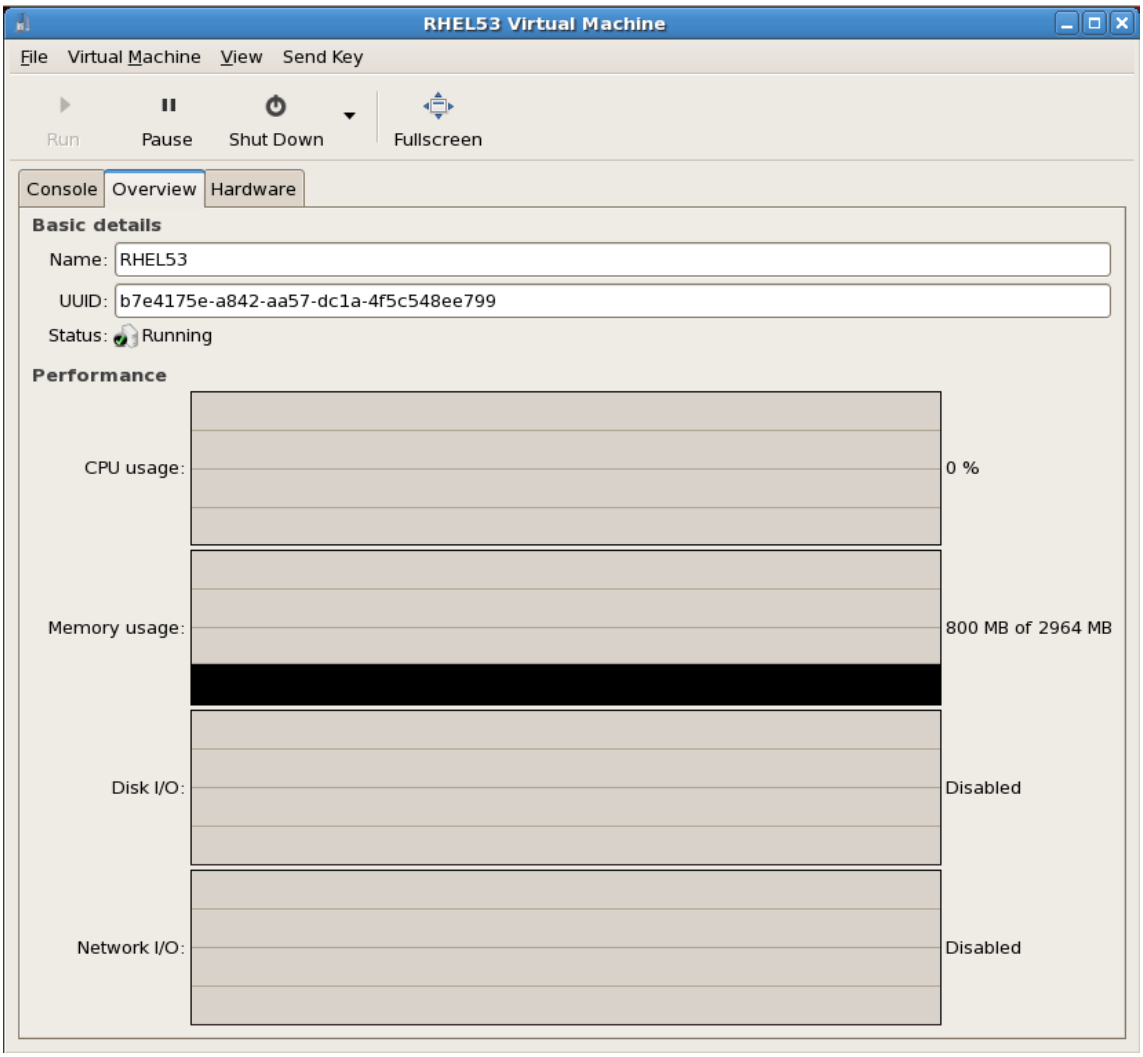


図23.10 ゲストの詳細概要を表示

3. **仮想マシンの詳細** ウィンドウで、**ハードウェア** タブをクリックします。

   仮想マシン詳細ハードウェア のウィンドウが現われます。

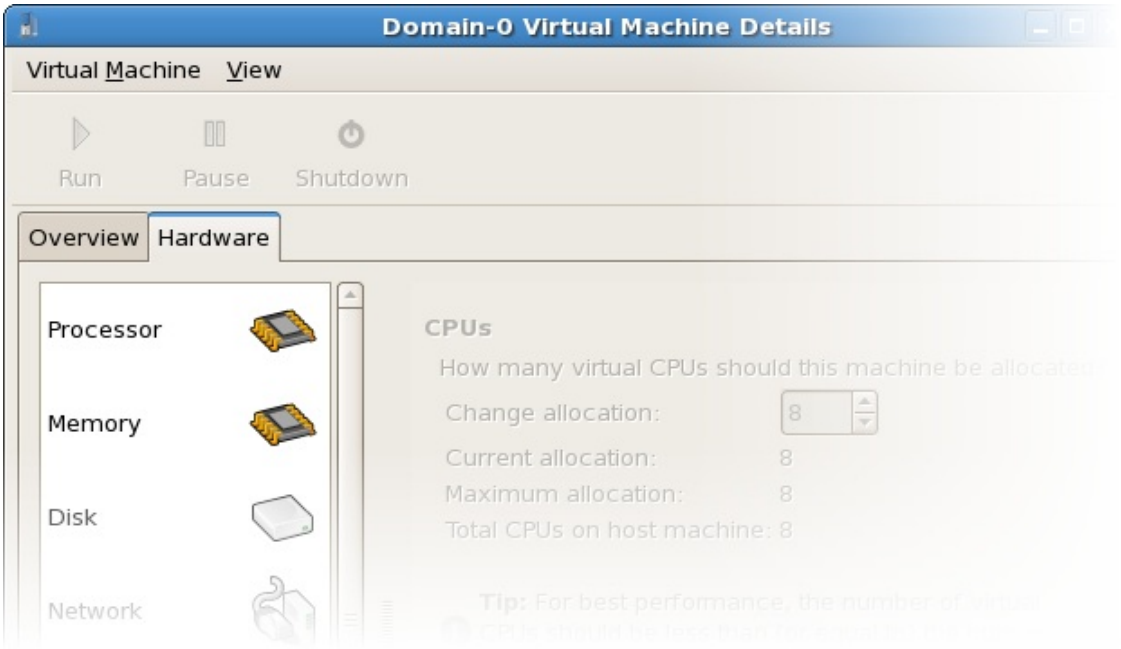


図23.11 ゲストハードウェアの詳細を表示

4. **ハードウェア** タブで **プロセッサ** をクリックして、現在のプロセッサ割り当ての表示や変更をします。

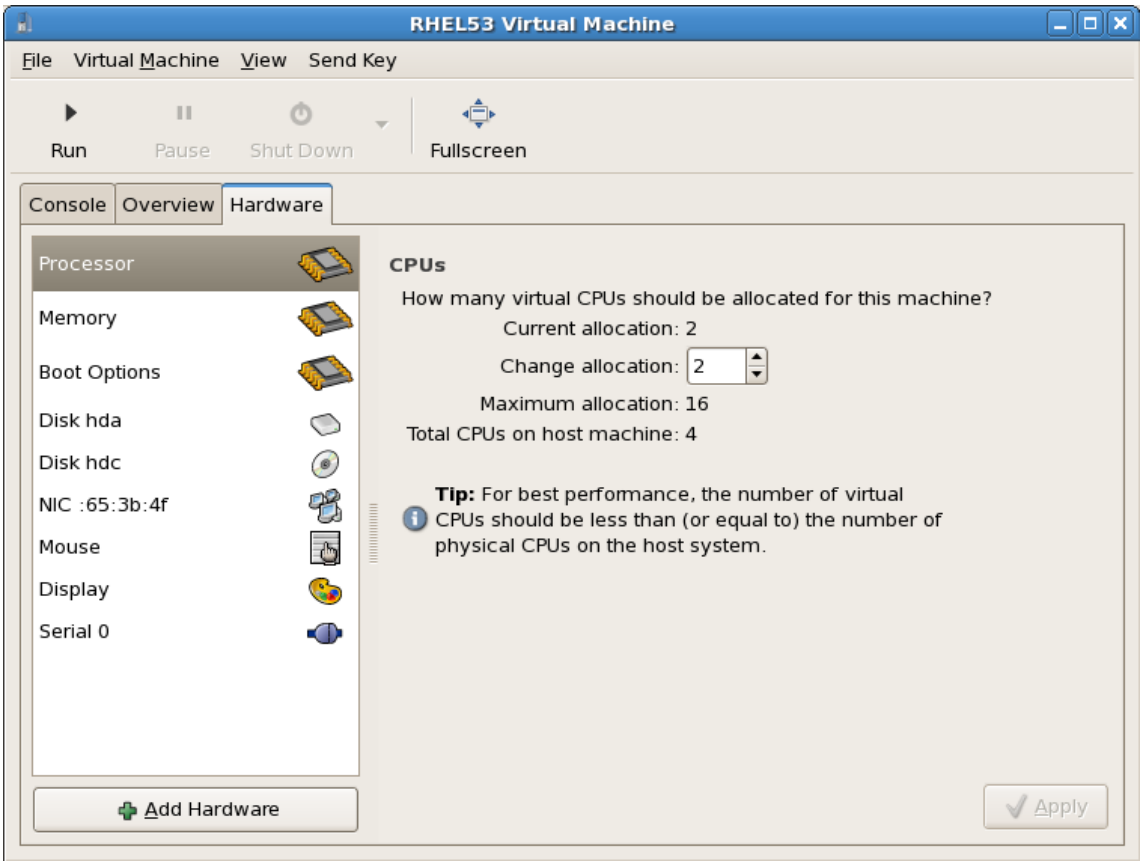


図23.12 プロセッサ割り当てのパネル

5. **ハードウェア** タブで、**メモリー** をクリックして、現在の RAM メモリー割り当ての表示や変更をします。

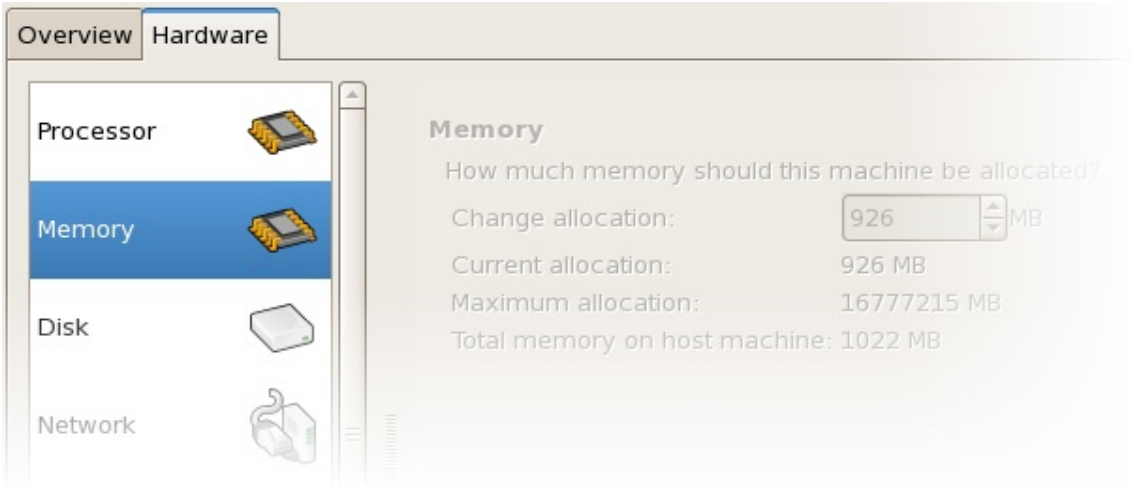


図23.13 メモリー割り当ての表示

6. **ハードウェア** タブで **ディスク** をクリックして現在のハードディスク設定の表示や変更をします。

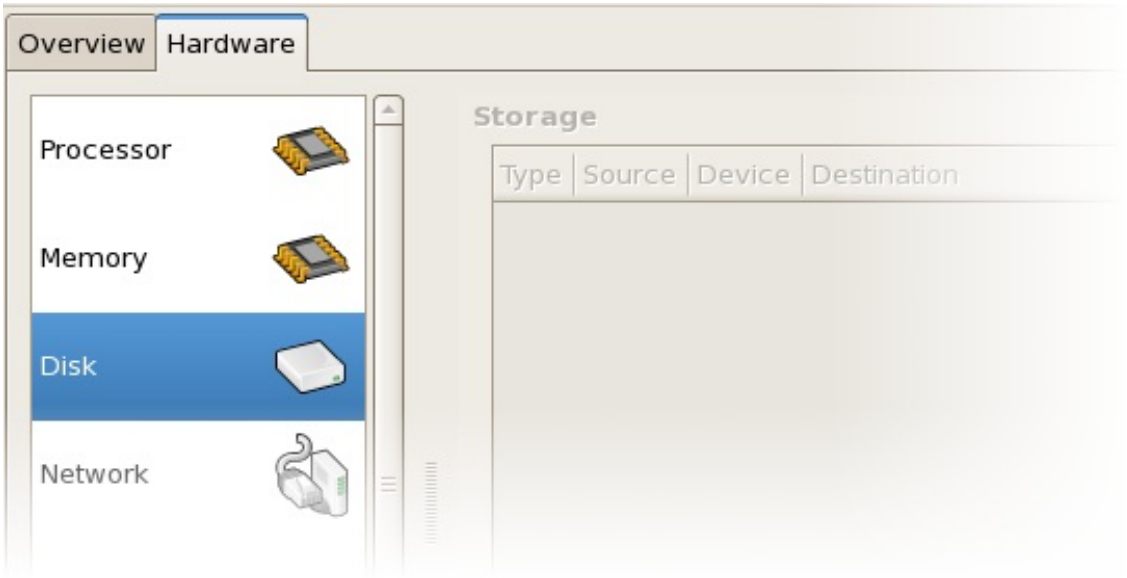


図23.14 ディスク設定の表示

7. **ハードウェア** タブで **ネットワーク** をクリックして現在のネットワーク設定の表示や変更をします。

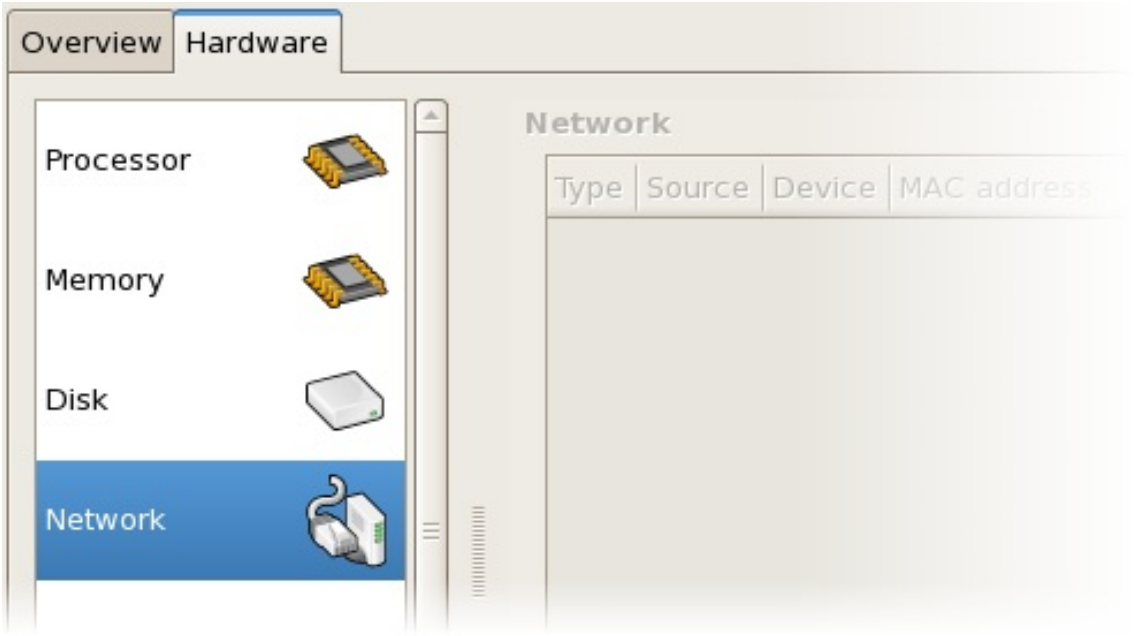


図23.15 ネットワーク設定の表示

23.8. ステータスの監視

仮想マシンマネージャを使用して、仮想システムのステータス監視を修正することができます。

ステータス監視を設定して、コンソールを有効にするには:

1. **編集** メニューから **個人設定** を選択します。

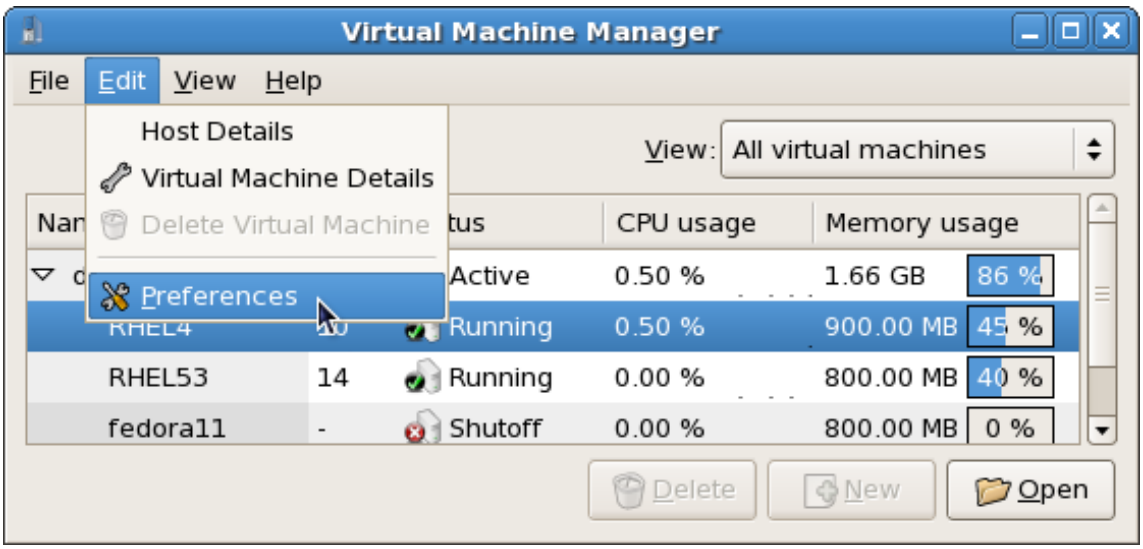


図23.16 ゲストの個人設定の修正

仮想マシンの個人設定ウィンドウが現われます。

2. ステータス監視エリア選択ボックスから、システムに更新させる時間(秒数)を指定します。

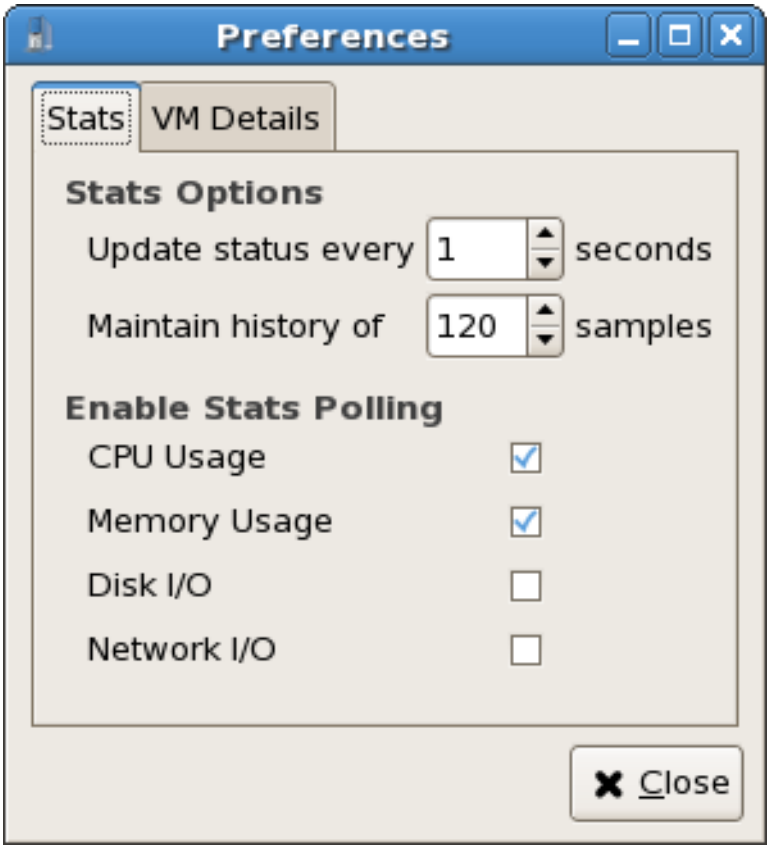


図23.17 ステータス監視の設定

3. コンソールエリアから、コンソールを開く方法と入力デバイスを指定します。

23.9. ゲスト識別子の表示

システム上の全ての仮想マシン用のドメイン ID を表示するには:

1. **表示** メニューから、**ドメイン ID** チェックボックスを選択します。

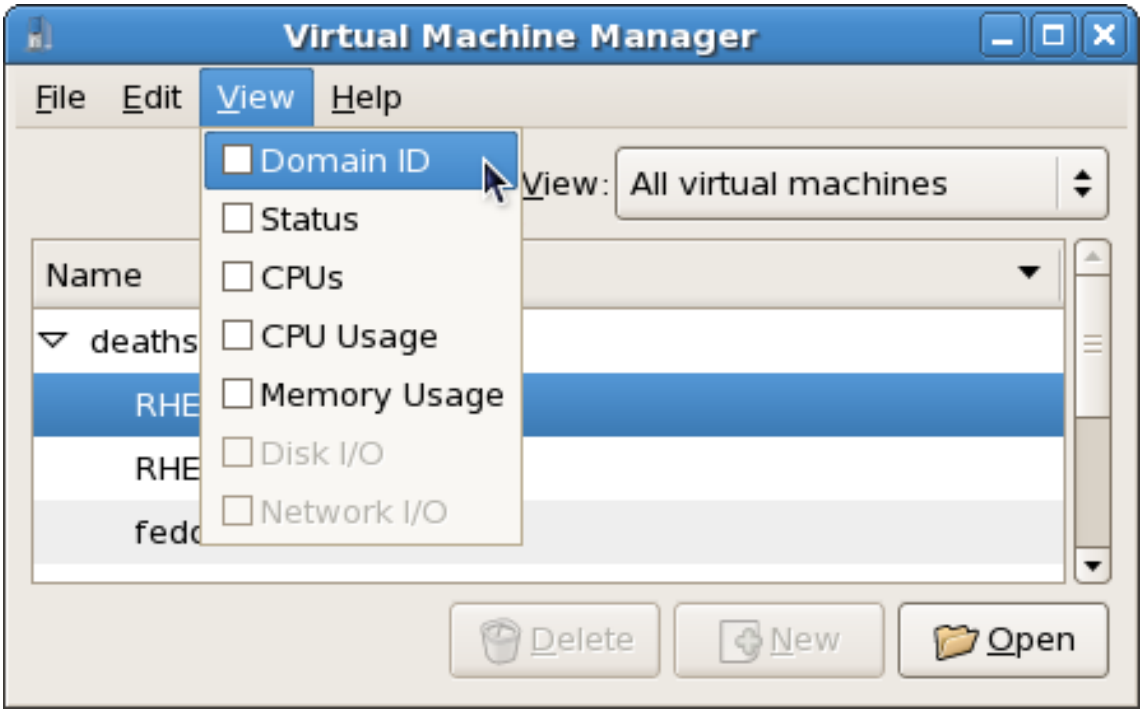


図23.18 ゲスト ID の表示

2. 仮想マシンマネージャは、システム上の全てのドメイン用のドメイン ID を一覧表示します。

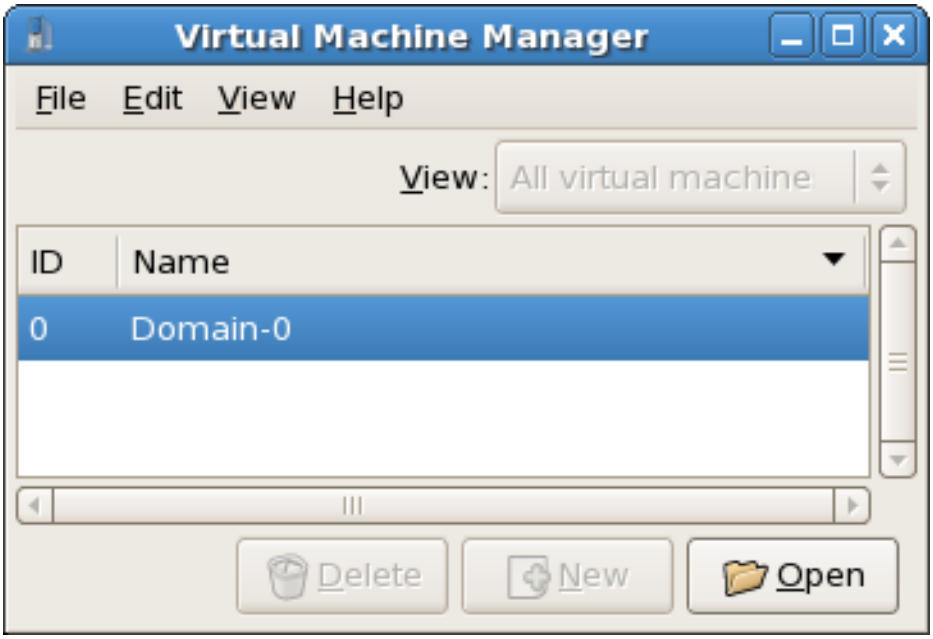


図23.19 ドメイン ID の表示

23.10. ゲストステータスの表示

システム上の全ての仮想マシンのステータスを表示するには:

1. **表示** メニューから、**ステータス** チェックボックスを選択します。

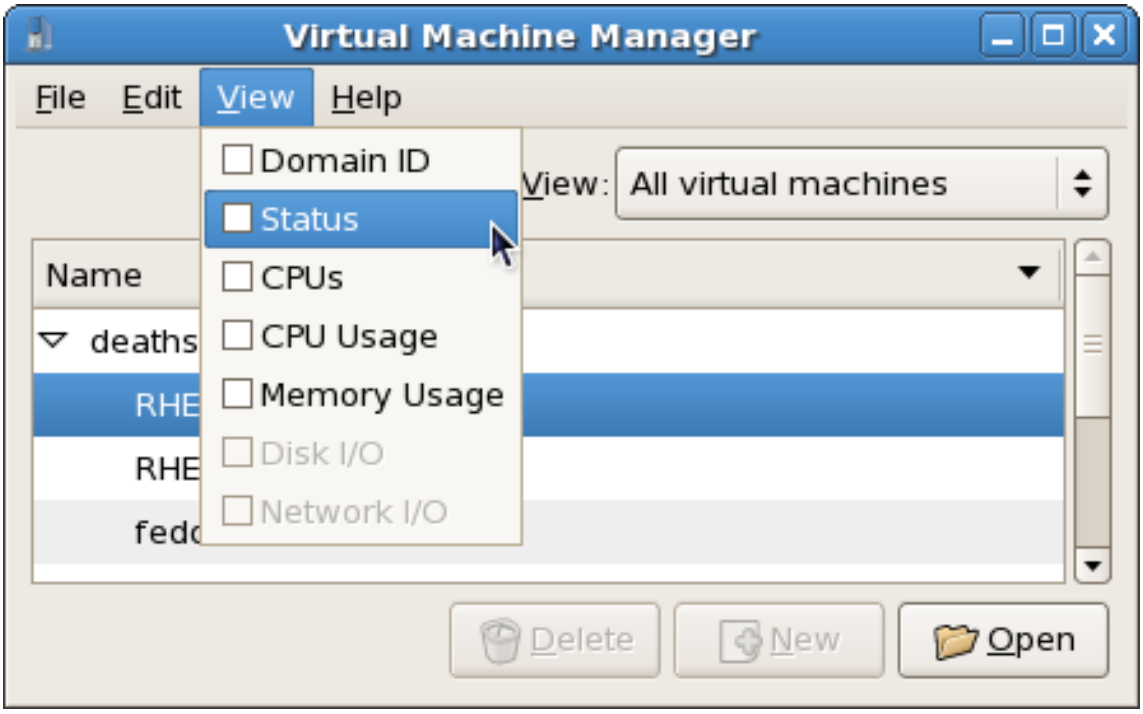


図23.20 仮想マシンステータスの選択

2. 仮想マシンマネージャはシステム上の全ての仮想マシンのステータスを一覧表示します。

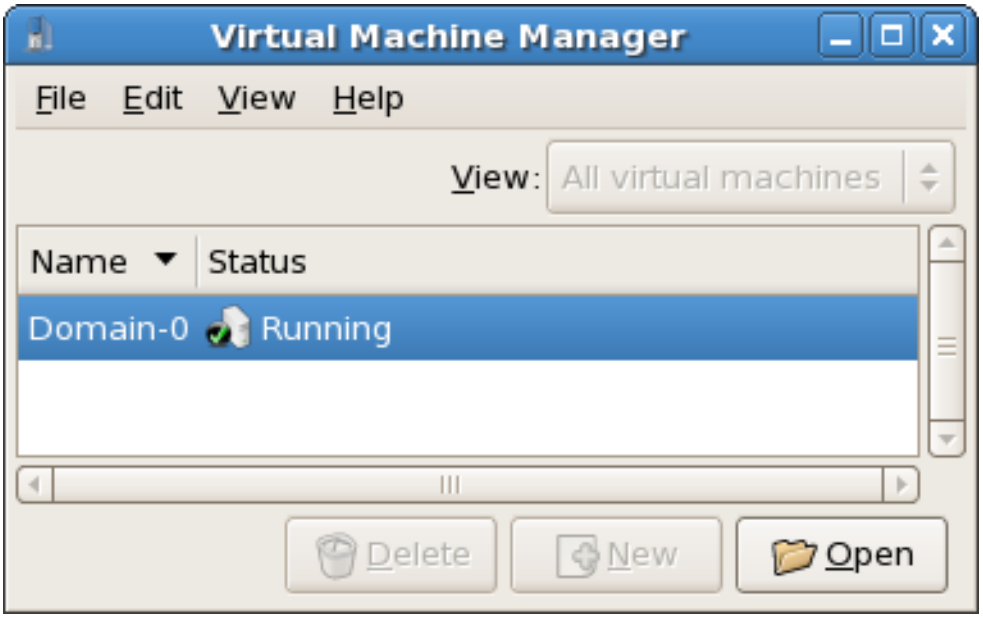


図23.21 仮想マシンステータスの表示

23.11. 仮想 CPU の表示

システム上の全ての仮想マシン用の仮想 CPU の数量を表示するには:

1. **表示** メニューから、**仮想 CPU** チェックボックスを選択します。

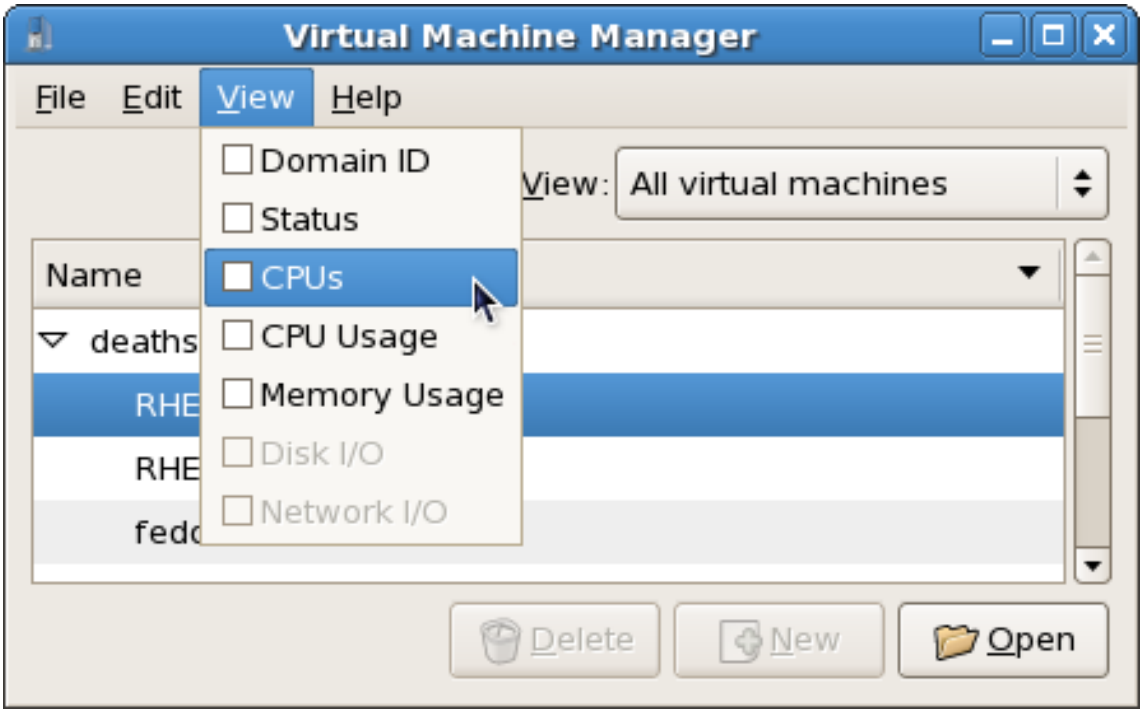


図23.22 仮想 CPU オプションの選択

2. 仮想マシンマネージャは、システム上の全ての仮想マシン用の仮想 CPU を一覧表示します。

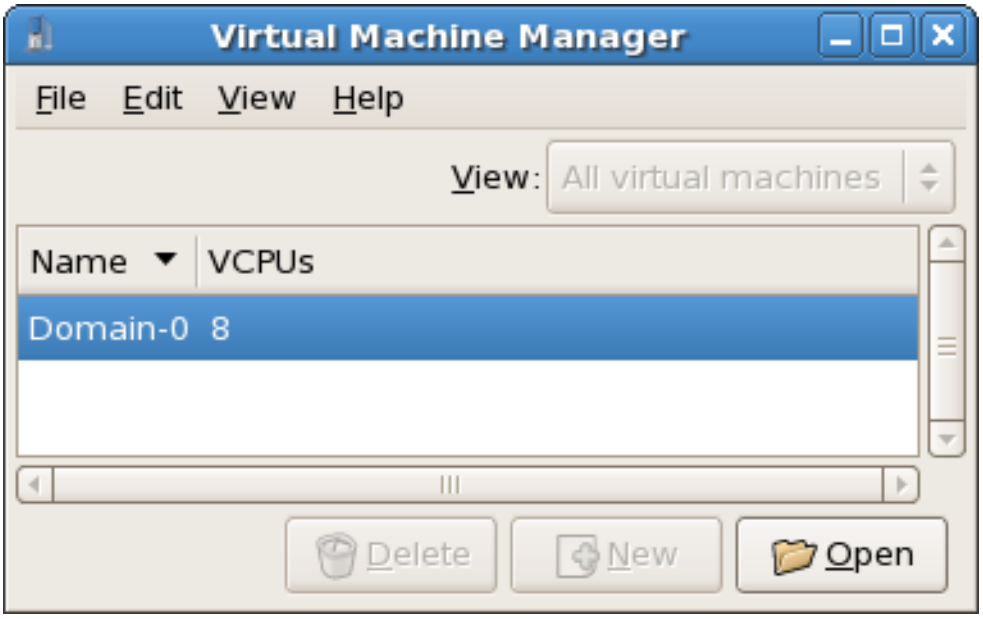


図23.23 仮想 CPU の表示

23.12. CPU 使用量の表示

システム上の全ての仮想マシン用の CPU 使用量を表示するには:

1. **表示** メニューから、**CPU 使用量** チェックボックスを選択します。

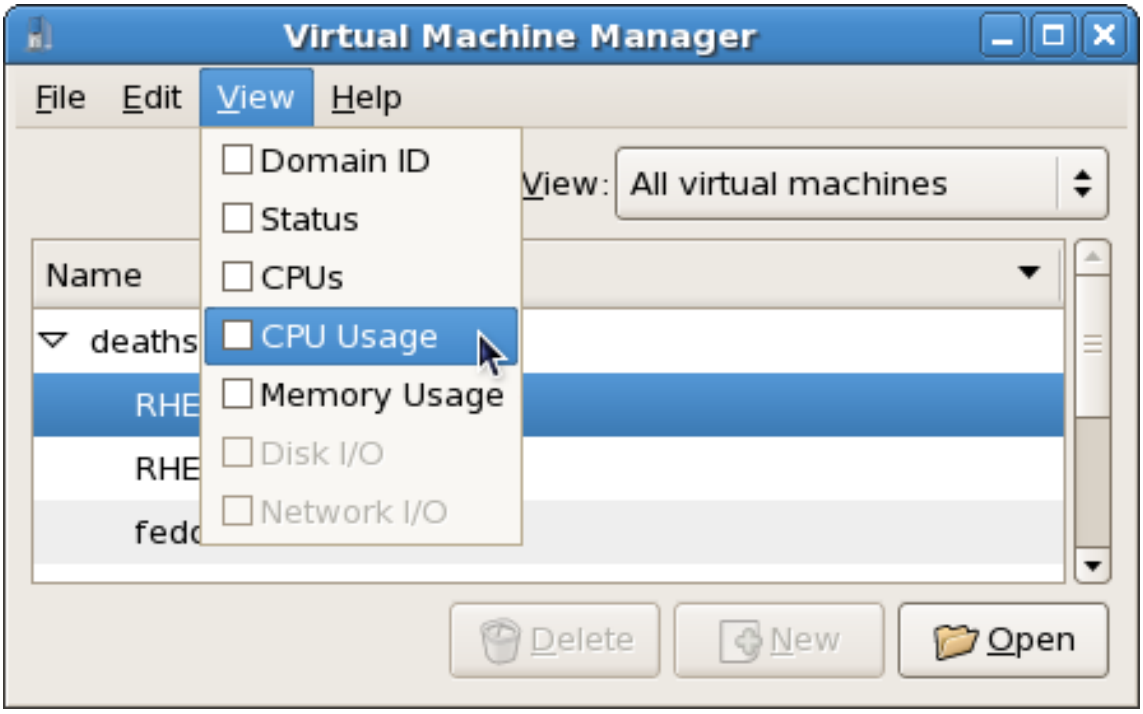


図23.24 CPU 使用量の選択

2. 仮想マシンマネージャは、システム上の全ての仮想マシン用の CPU 使用率を一覧表示します。

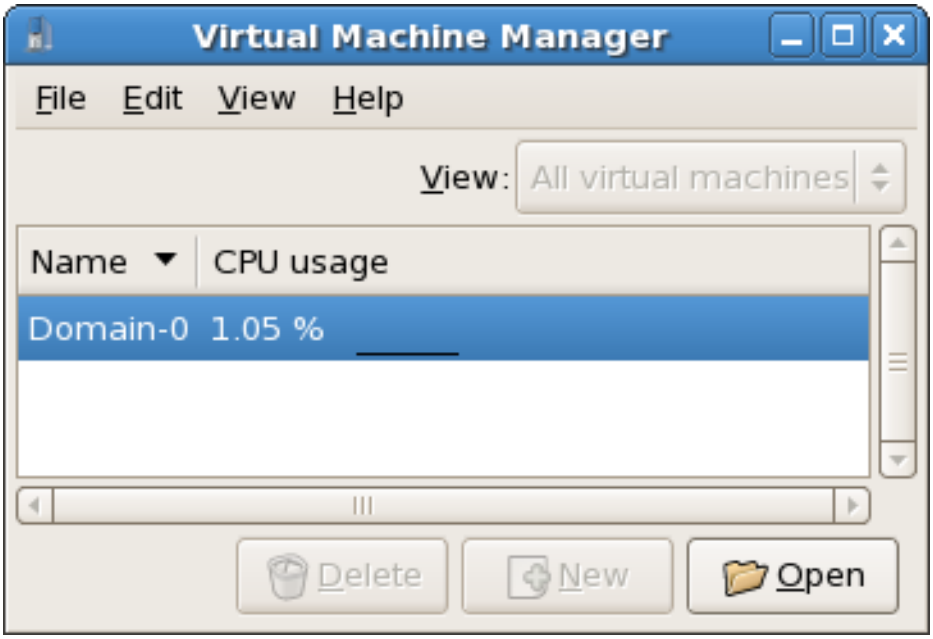


図23.25 CPU 使用量の表示

23.13. メモリー使用量の表示

システム上の全ての仮想マシン用のメモリー使用量を表示するには:

1. **表示** メニューから、**メモリー使用量** チェックボックスを選択します。

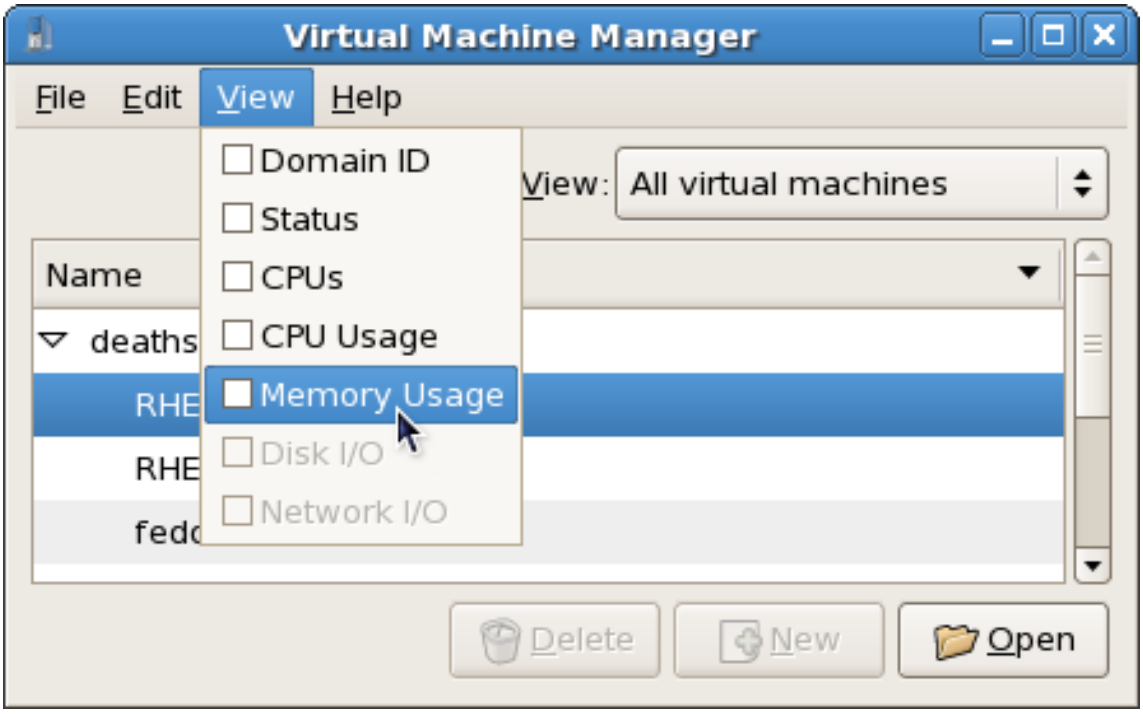


図23.26 メモリー使用量の選択

2. 仮想マシンマネージャは、システム上の全ての仮想マシン用のメモリー使用量をメガバイトで一覧表示します。

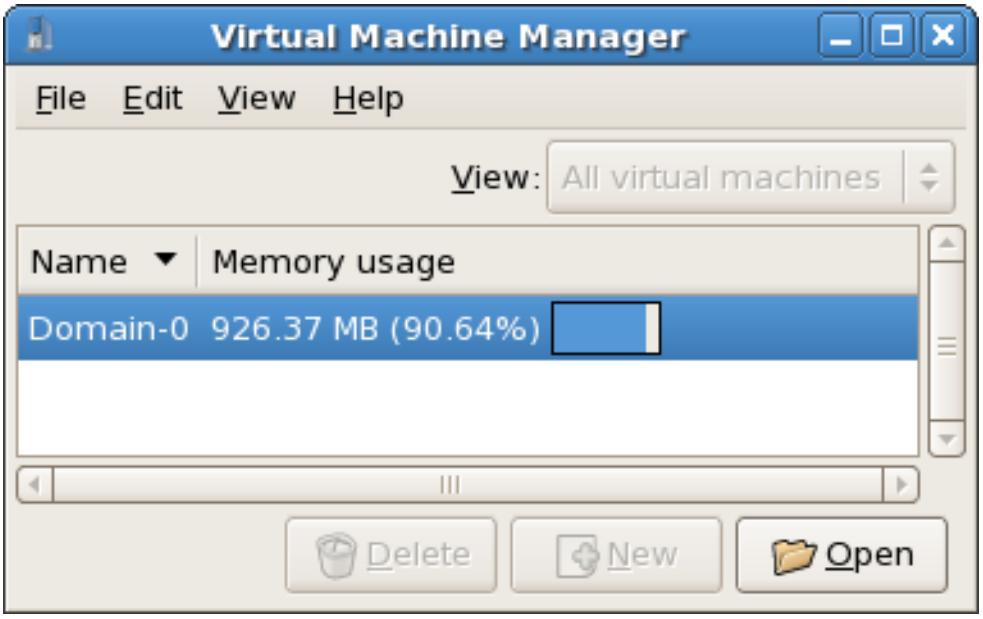


図23.27 メモリー使用量の表示

23.14. 仮想ネットワークの管理

システム上の仮想ネットワークを設定するには:

1. **編集** メニューから **ホストの詳細（Host Details）** を選択します。

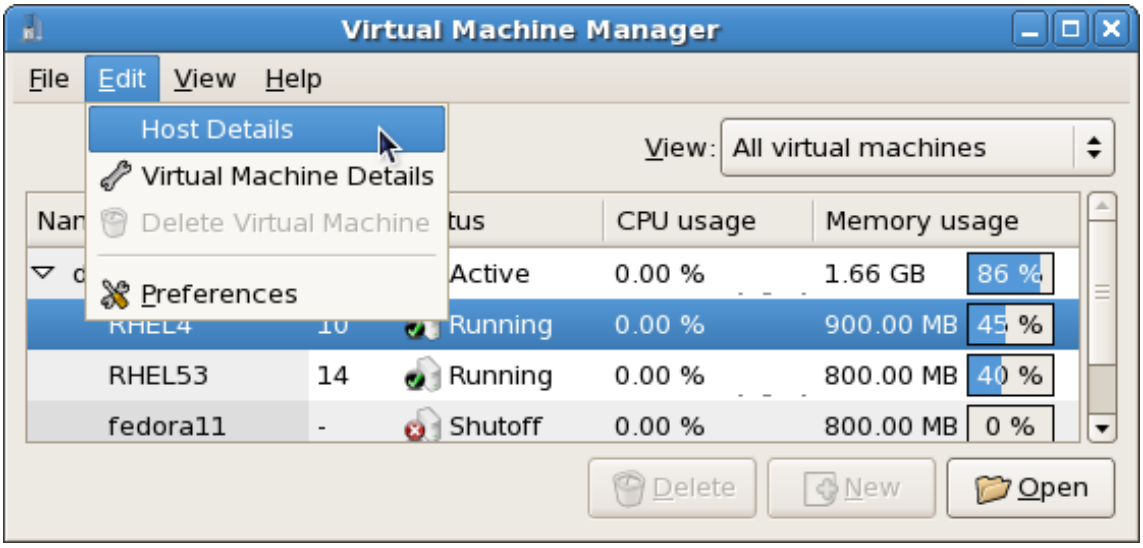


図23.28 ホスト詳細の選択

2. これが **ホストの詳細** メニューを開きます。 **仮想ネットワーク** タブをクリックします。

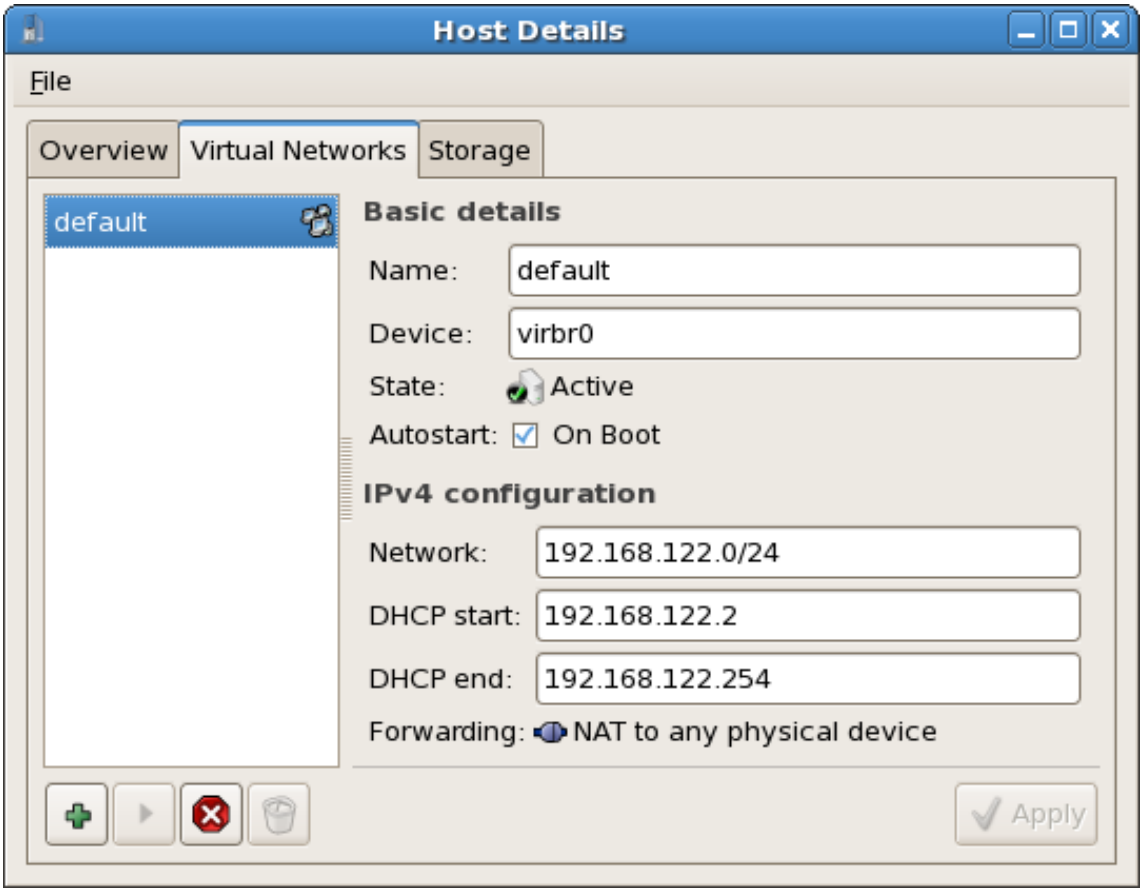


図23.29 仮想ネットワークの設定

3. 使用できる仮想ネットワークは全て、メニューの左側ボックスに表示されています。 このボックスから一つの仮想ネットワークを選択して、必要に応じた編集をすることで その設定を変更できます。

23.15. 仮想ネットワークの作成

システム上に仮想ネットワークを作成するには:

1. **ホストの詳細** メニュー(「仮想ネットワークの管理」 を参照) を開いて、**追加** ボタンを クリックします。

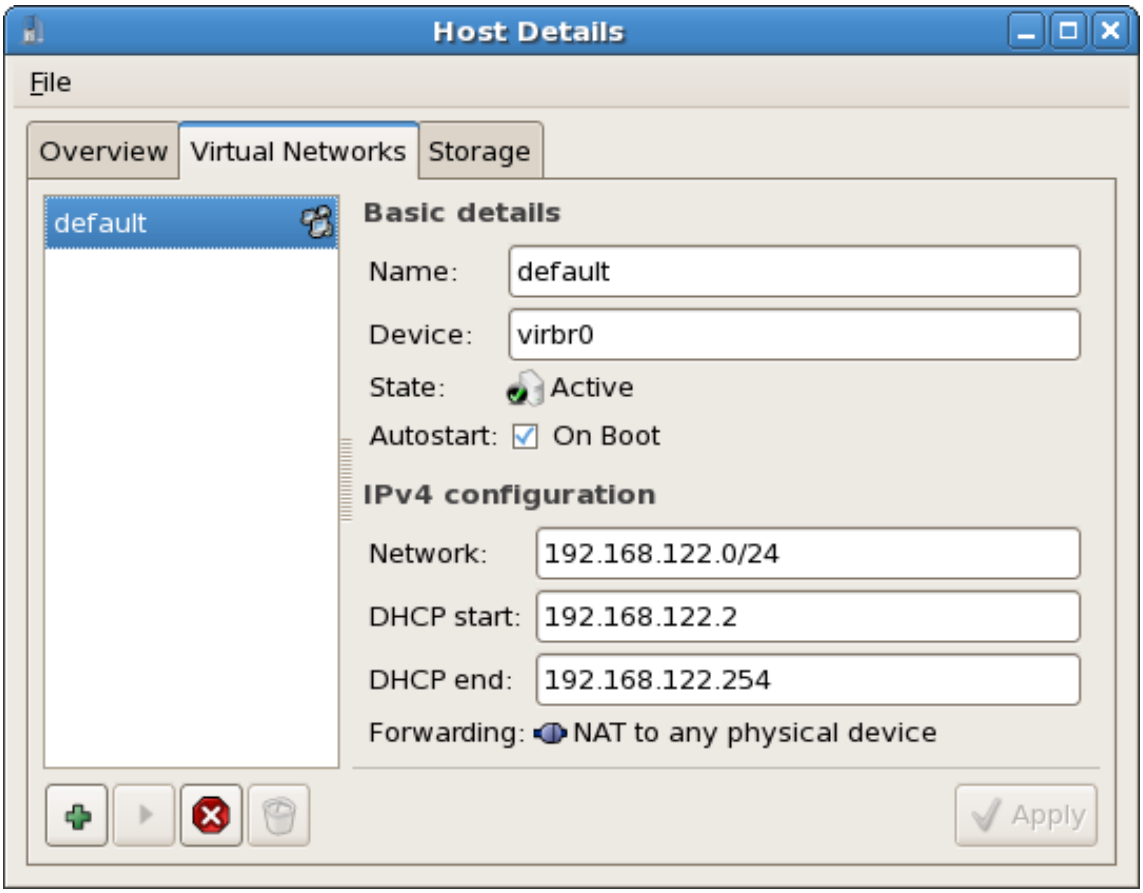


図23.30 仮想ネットワークの設定

そうすると、**新規仮想ネットワークの作成（Create a new virtual network）** メニューが開きます。**進む（Forward）** をクリックして 継続します。

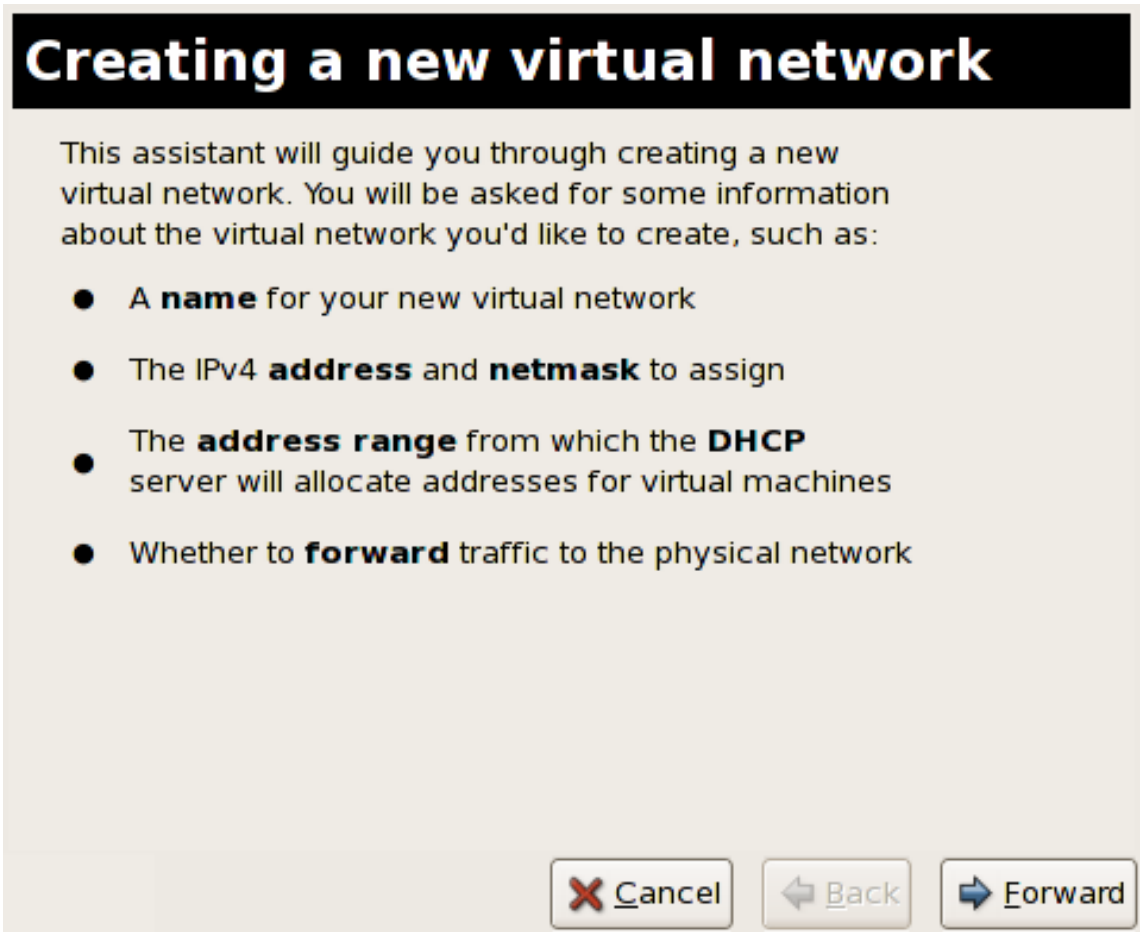


図23.31 新規仮想ネットワークの作成

2. 使用する仮想ネットワークの名前を入力して、それから **進む** をクリックします。

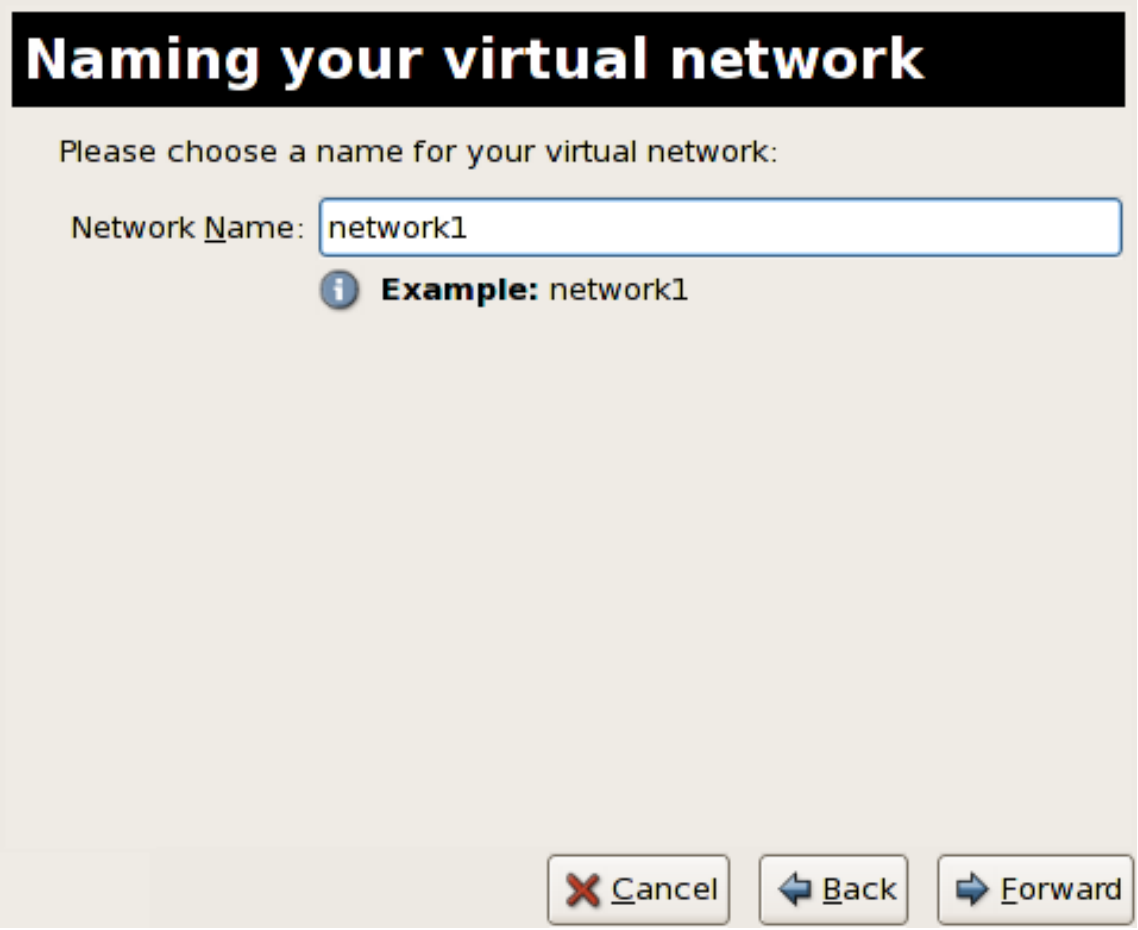


図23.32 仮想ネットワークの命名

3. 使用する仮想ネットワークの IPv4 アドレスを入力して、それから **進む** を クリックします。

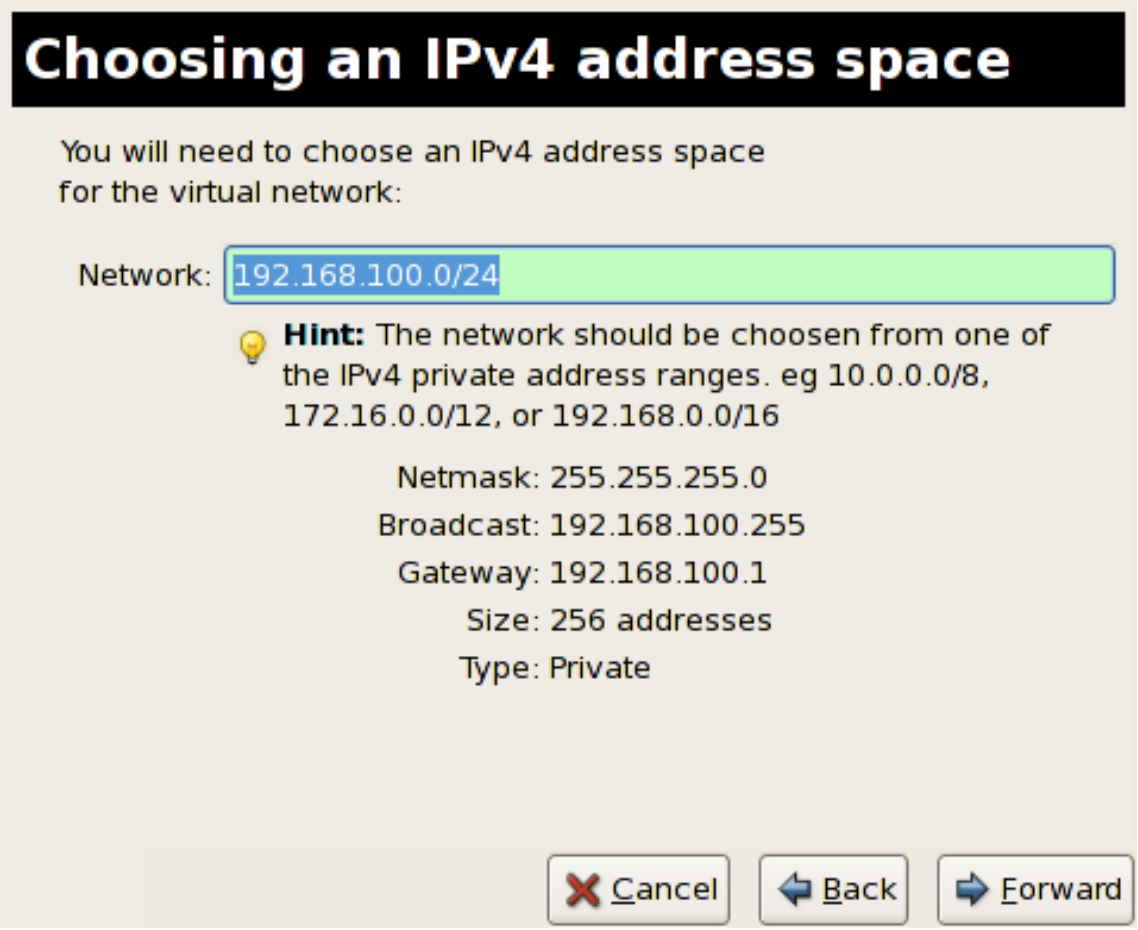


図23.33 IPv4 のアドレススペースの選択

4. IP アドレスの **開始** と **終了** の 範囲を指定して、使用したい仮想ネットワークの DHCP 幅を定義します。 **進む（Forward）** をクリックして継続します。

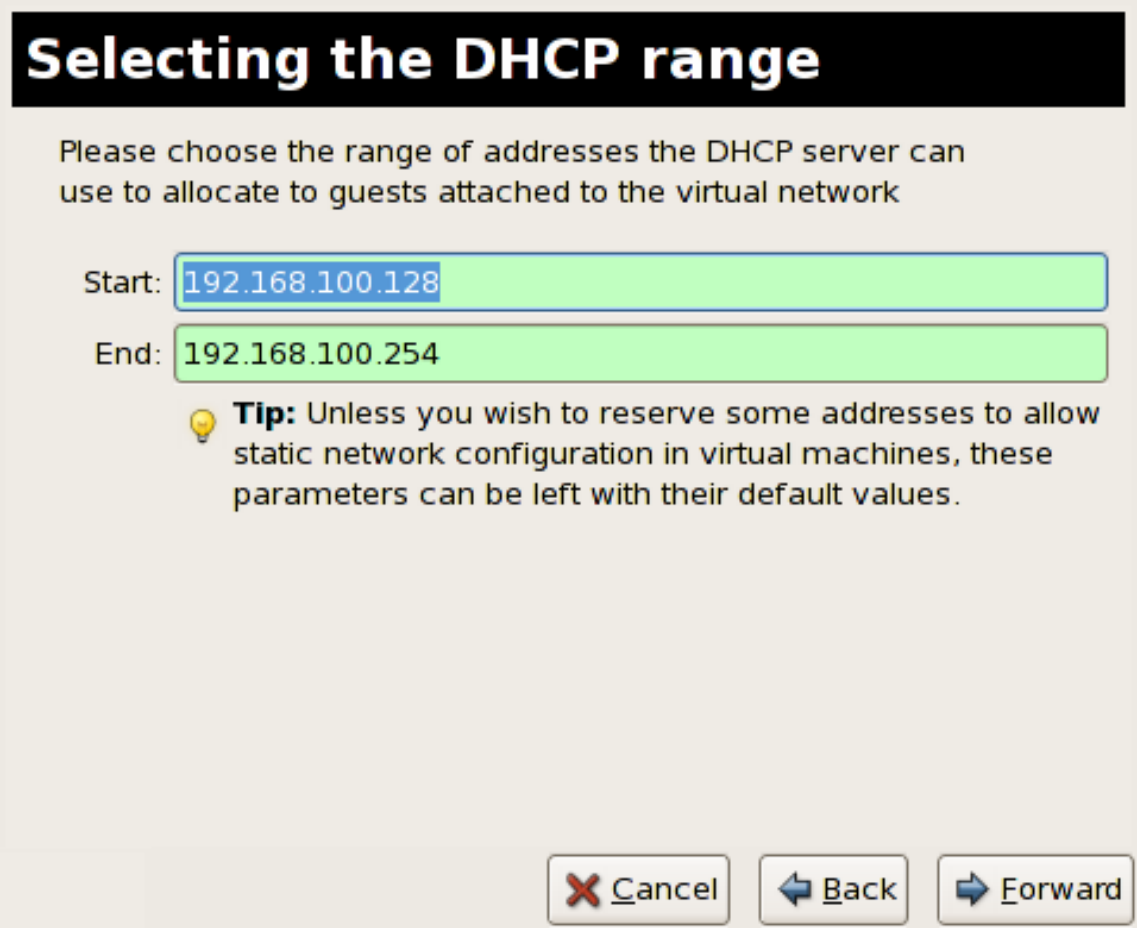


図23.34 DHCPの範囲を選択

5. 仮想ネットワークが物理ネットワークに接続する方法を選択します。

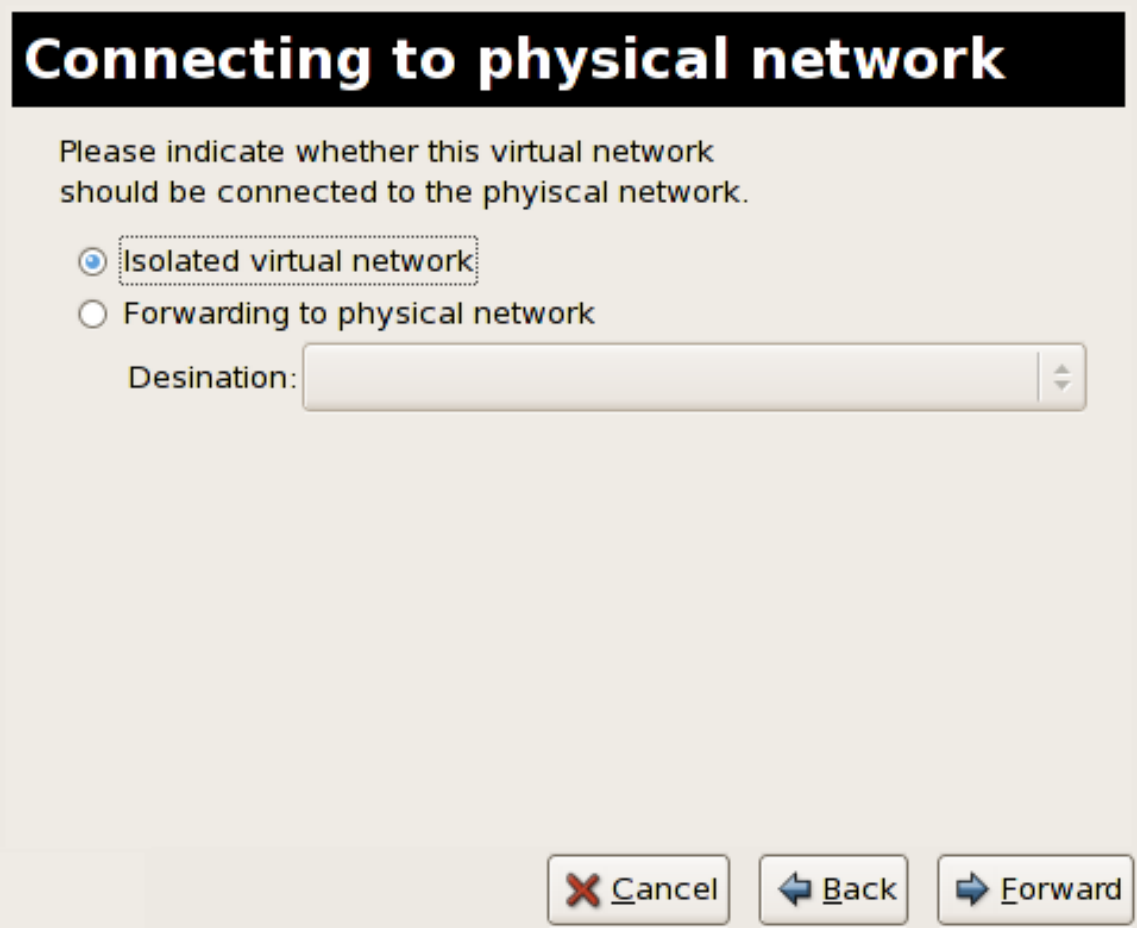


図23.35 物理ネットワークへの接続

物理ネットワークへ転送（Forwarding to physical network） を選択する場合、**転送先（Destination）** が **いずれかの物理デバイスに NAT** とすべきか、又は、**物理デバイス eth0 に NAT** とすべきを選びます。

進む をクリックして継続します。

6. これでネットワーク作成の準備ができました。ネットワークの設定を確認して、 **完了（Finish）** をクリックします。

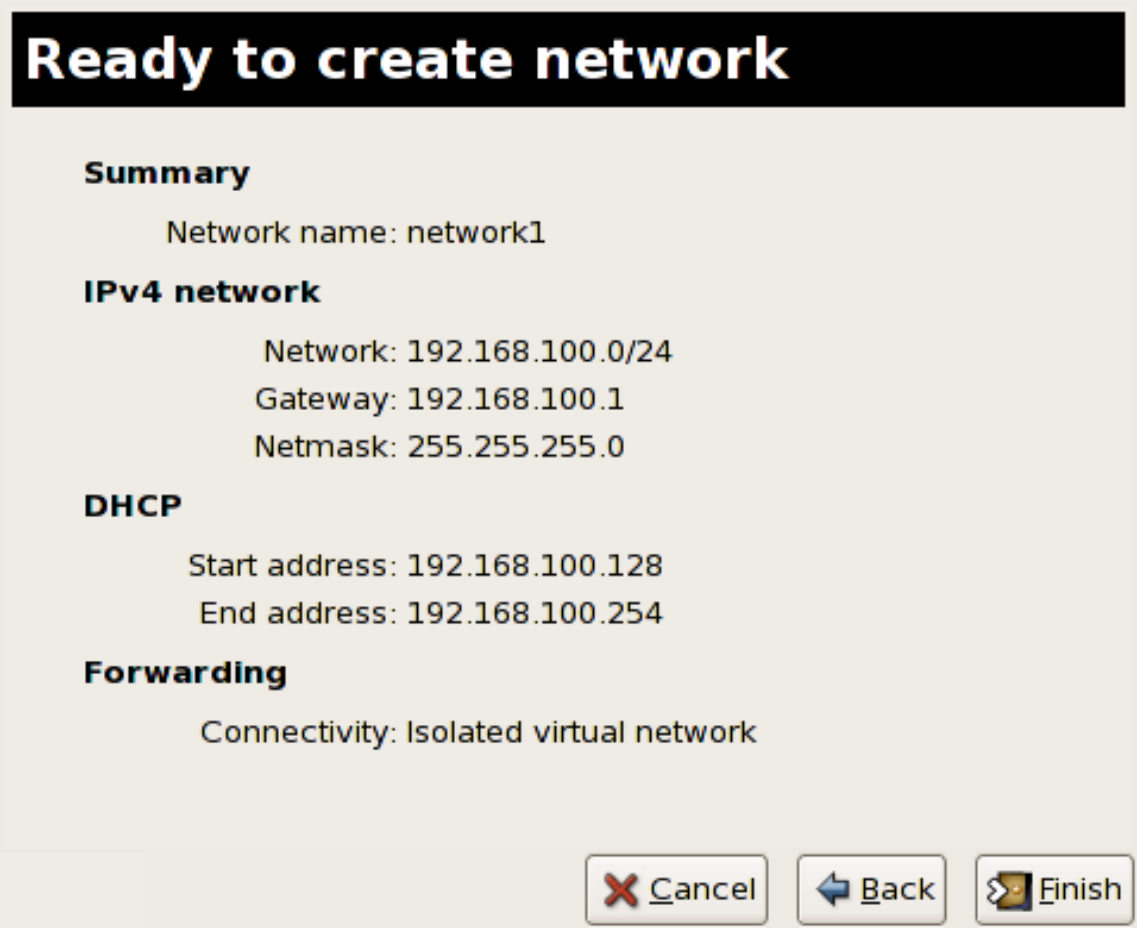


図23.36 ネットワーク作成への準備終了

7. 仮想ネットワークが今、**ホストの詳細（Host Details）** の **仮想ネットワーク** タブで使用できます。

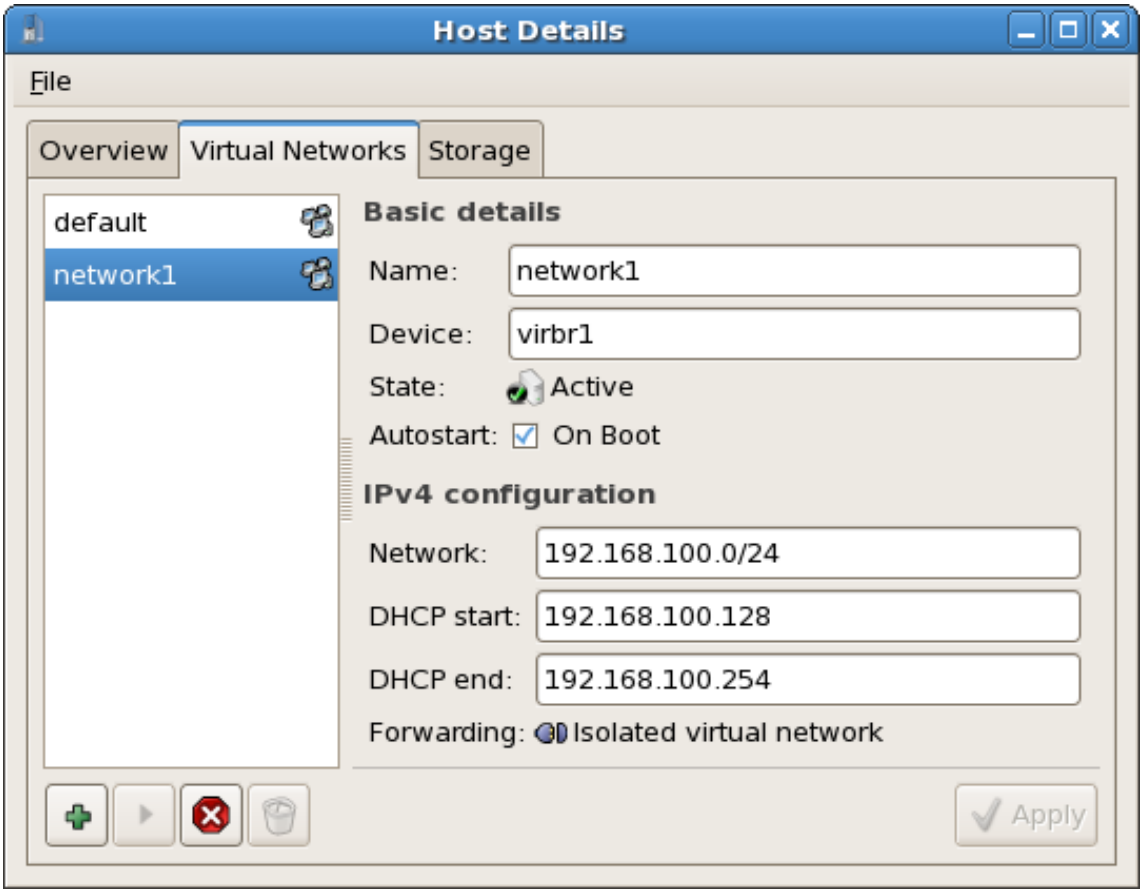


図23.37 新規の仮想ネットワークが今、使用できます。

第24章 XM コマンドのクイックリファレンス

xm コマンドには Xen hypervisor を管理する機能があります。 ほとんどの操作は libvirt ツール、**virt-manager** アプリケーション、 あるいは、**virsh** コマンドで実行されます。**xm** コマンドは、libvirt ツールのエラーチェック機能を持ちませんので libvirt ツールのタスクには 使用すべきではありません。

現時点では、**virt-manager** の使用では実行 できない操作がいくつかあります。**xm** コマンドの 他の Xen 実装用のオプションの一部は Red Hat Enterprise Linux 5 で動作 しません。以下のリストは利用可能と利用不可能なコマンドオプションの概要を 提供します。

警告

xm ではなく、**virsh**、又は **virt-manager** の使用が推奨されます。**xm** コマンドはエラーチャックや設定ファイルエラーをうまく処理できないため、誤動作が 仮想マシン内のシステム不安定性やエラーを招きます。**Xen** 設定ファイルを手動で編集することは 危険であり、回避すべきです。この章はご自分の責任で使用判断して下さい。

基本的管理オプション

一般に使用される基本的な **xm** コマンド群を以下に示します:

- **xm help [--long]**: 使用可能なオプションとヘルプテキストを表示します。
- **xm list** コマンドを使用して活動中のドメインをリストできます:

```
$ xm list
Name                                ID  Mem(MiB)   VCPUs
State      Time(s)
Domain-0                             0     520       2         r--
---     1275.5
r5b2-mySQL01                        13     500       1         -b--
--       16.1
```

- **xm create [-c]** ***DomainName/ID***: 仮想マシンを開始します。 **-c** オプションが使用されると、開始プロセスはゲストのコンソールを 添付します。
- **xm console** ***DomainName/ID***: 仮想マシンのコンソールに添付します。
- **xm destroy** ***DomainName/ID***: パワーオフと同様に、仮想マシンを終了します。
- **xm reboot** ***DomainName/ID***: 仮想マシンをリブートして、通常のシステムシャットダウンと開始のプロセスを実行します。
- **xm shutdown** ***DomainName/ID***: 仮想マシンをシャットダウンして、通常のシステムシャットダウン工程を実行できます。
- **xm pause**

- **xm unpause**
- **xm save**
- **xm restore**
- **xm migrate**

リソース管理のオプション

次の **xm** コマンド群を使用してリソースを管理することができます:

- **xm mem-set**
- **xm vcpu-list** を使用して仮想化 CPU のグループをリストできます:

```
$ xm vcpu-list
Name                                ID  VCPUs   CPU State  Time(s)  CPU
Affinity
Domain-0                             0     0     0    r--   708.9    any
cpu
Domain-0                             0     1     1   -b-    572.1    any
cpu
r5b2-mySQL01                        13     0     1   -b-     16.1    any
cpu
```

- **xm vcpu-pin**
- **xm vcpu-set**
- **xm sched-credit** コマンドを使用して任意のドメイン用の スケジューラパラメータを表示することができます:

```
$ xm sched-credit -d 0
{'cap': 0, 'weight': 256}
$ xm sched-credit -d 13
{'cap': 25, 'weight': 256}
```

監視とトラブルシューティングのオプション

監視とトラブルシューティングには、次の **xm** コマンド群を 使用します:

- **xm top**
- **xm dmesg**
- **xm info**
- **xm log**
- **xm uptime** を使用してゲストとホストのアップタイムを表示 できます:

```
$ xm uptime
Name                          ID  Uptime
Domain-0                       0  3:42:18
r5b2-mySQL01                  13  0:06:27
```

- **xm sysrq**
- **xm dump-core**
- **xm rename**
- **xm domid**
- **xm domname**

現在サポートのないオプション

xm vnet-list には、現在サポートがありません。

第25章 XEN カーネルブートパラメータの設定

GRUB (GNU Grand Unified Boot Loader) は各種のインストール済オペレーティングシステム、 またはカーネルをブートするプログラムです。またこれは、ユーザーがカーネルに引数を渡すことを 可能にします。GRUB 設定ファイル (**/boot/grub/grub.conf** に存在) は GRUB ブートメニューインターフェイス用にオペレーティングシステムのリストを作成します。**kernel-xen** RPM をインストールすると、スクリプトが **kernel-xen** エントリを GRUB 設定ファイルに追加し、これがデフォルトで **kernel-xen** を ブートします。 **grub.conf** ファイルの編集で、デフォルトの カーネルを修正するか、又は別のカーネルパラメータを追加します。

```
title Red Hat Enterprise Linux Server (2.6.18-3.el5xen)
root (hd0,0)
        kernel  /xen.gz.-2.6.18-3.el5
        module  /vmlinuz-2.6..18-3.el5xen ro root=/dev/VolGroup00/LogVol00
rhgb quiet
        module  /initrd-2.6.18-3. el5xenxen.img
```

Linux grub エントリをこの例のように設定すると、ブートローダは hypervisor と **initrd** イメージと Linux カーネルをロードします。カーネルエントリは他のエントリの上部にある為、カーネルが最初にメモリーにロードされます。ブートローダはコマンドライン引数を hypervisor と Linux カーネルに送り、またそれらから受理します。次の例では Dom0 linux カーネルメモリーを 800 MB に制限する方法を示しています:

```
title Red Hat Enterprise Linux Server (2.6.18-3.el5xen)
root (hd0,0)
        kernel  /xen.gz.-2.6.18-3.el5 dom0_mem=800M
        module  /vmlinuz-2.6..18-3.el5xen ro root=/dev/VolGroup00/LogVol00
rhgb quiet
        module  /initrd-2.6.18-3. el5xenxen.img
```

これらの GRUB パラメータを使用することで Virtualization hypervisor を設定できます:

```
mem
```

これが hypervisor カーネルで利用できるメモリーの容量を制限します。

```
com1=115200, 8n1
```

これによりシステム内の一番目のシリアルポートがシリアルコンソールとして動作できるようになります (com2 は その次のポートと言う順に割り当てられます)。

```
dom0_mem
```

これが、 hypervisor 用に使用できるメモリーを制限します。

```
dom0_max_vcpus
```

これが Xen domain0 から見える CPU の容量を制限します。

```
acpi
```

これが ACPI hypervisor を hypervisor と domain0 に切り替えます。ACPI パラメータのオプションには次のようなものがあります:

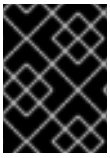
```
/*   ****  Linux config options: propagated to domain0  ****/
/*   "acpi=off":      Disables both ACPI table parsing and interpreter.
*/
/*   "acpi=force":    Overrides the disable blacklist.
*/
/*   "acpi=strict":   Disables out-of-spec workarounds.
*/
/*   "acpi=ht":       Limits ACPI from boot-time to enable HT.
*/
/*   "acpi=noirq":    Disables ACPI interrupt routing.
*/
```

```
noacpi
```

これは割り込み伝達用の ACPI を無効にします。

第26章 ELILO の設定

ELILO は 特に Itanium® などの EFI ベースの システムで使用されるブートローダです。x86 と x86-64 のシステム上のブートローダ、GRUB と同様に ELILO はユーザーがシステムブートシーケンスでインストール済みのカーネルを選択できるようにします。ELILO は また、ユーザーがカーネルに引数を渡すことができるようにします。ELILO 設定ファイルは ELILO ブート パーティションにあり、**/etc/elilo.conf** にシンボルリンクがあるもので、グローバル オプションとイメージスタンザの一覧を含んでいます。**kernel-xen** RPM をインストール すると、ポストインストールスクリプトが適切なイメージスタンザを **elilo.conf** に追加します。



重要

ELILO に関するこのセクションは、intel Itanium アーキテクチャ上で Xen カーネルを稼働しているシステムが対象です。

ELILO 設定ファイルは次の ２つのセクションを持ちます:

- ELILO の動作と全てのエントリに影響するグローバルオプション。 標準的には，これらをデフォルト値から変更する必要はありません。
- 関連のオプションと共にブート選択を定義するイメージスタンザ。

elilo.conf 内のイメージスタンザのサンプルをここに示します:

```
image=vmlinuz-2.6.18-92.el5xen
        vmm=xen.gz-2.6.18-92.el5
        label=linux
        initrd=initrd-2.6.18-92.el5xen.img
        read-only
        root=/dev/VolGroup00/rhel5_2
        append="-- rhgb quiet"
```

image パラメータはそれ以下の行が単独のブート選択に適用されることを示します。このスタンザは hypervisor (***vmm***) と、 **initrd** と、 更に、hypervisor とカーネルへのコマンドライン引数 (***read-only***、***root***、及び ***append***) を定義します。 ブートシーケンス中に ELILO がロードされると、そのイメージには ***linux*** のラベルが付きます。

ELILO は ***read-only*** をカーネルコマンドラインオプションの***ro*** と解釈して、ルートファイルシステムが読み込み専用としてマウントされる要因となります。この状態は **initscripts** がルート ドライブを読み込み/書き込みとしてマウントするまで継続します。ELILO は "***root***" 行を カーネルコマンドラインにコピーします。これらは "***append***" 行とマージされて、以下の ような 完全なコマンド行を構成します:

```
"-- root=/dev/VolGroup00/rhel5_2 ro rhgb quiet"
```

-- シンボルは、hypervisor とカーネルの引数に 区切りを付けます。hypervisor 引数が最初に来て、それから **--** デリミタで、 その後、カーネル引数が来ます。hypervisor は通常引数を取りません。

注記

ELILO は全コマンド行を hypervisor に渡します。hypervisor はそのコンテンツを 分割して、カーネルオプションをカーネルに渡します。

hypervisor をカスタマイズするには、**--** の前にパラメータを 挿入します。hypervisor メモリー (***mem***) パラメータと カーネル用の ***quiet*** パラメータのサンプルを以下に示します:

```
append="dom0_mem=2G -- quiet"
```

ELILO hypervisor のパラメータ

パラメータ	説明
mem=	***mem*** パラメータは hypervisor の最大 RAM 使用を定義 します。システム内の追加の RAM はいずれも無視されます。このパラメータは、B, K, M あるいは G の 接尾辞で指定でき、それぞれバイト、キロバイト、メガバイト、ギガバイトを 示します。接尾辞がない場合、デフォルトの単位はキロバイトです。
dom0_mem=	***dom0_mem=*** は dom0 に割り当てる RAM のサイズをセットします。 上記の mem パラメータと同じ接尾辞が通用します。Itanium® 上での Red Hat Enterprise Linux 5.2 のデフォルトは 4G です。
dom0_max_vcpus=	***dom0_max_vcpus=*** は、hypervisor に割り当てる CPU の 数量をセットします。Itanium® 上での Red Hat Enterprise Linux 5.2 のデフォルトは 4 です。
com1=<*baud*>,DPS,<*io_base*>,<*irq*>	***com1=*** は最初のシリアル行の為のパラメータをセットします。 例えば、**com1=9600,8n1,0x408,5** 。***io_base*** と ***irq*** のオプションは無視して標準のデフォルトで結構です。***baud*** パラメータは***auto*** にセットしてブートローダ設定がそのまま保持 されるようにします。ELILO 内か、EFI 設定内でグローバルシリアルオプションとしてシリアルパラメータがセットされている場合は、***com1*** パラメータは無視できます。
com2=<*baud*>,DPS,<*io_base*>,<*irq*>	２つめのシリアル行にパラメータをセットします。上述の ***com1*** パラメータの説明を参照して下さい。
console=<*specifier_list*>	***console*** はコンソールオプション用のコンマで区切られた 個人設定一覧です。オプションには、vga、com1 、及び com2 が含まれます。hypervisor が EFI コンソール設定の継承を試みるため、この設定は除外されるべきです。

注記

ELILO パラメータの完全リストは XenSource で取得できます。

上記の修正済み設定のサンプルは、メモリーと cpu 割り当てのパラメータを hypervisor に 追加する為の構文を示しています:

-

```
image=vmlinuz-2.6.18-92.el5xen
        vmm=xen.gz-2.6.18-92.el5
        label=linux
        initrd=initrd-2.6.18-92.el5xen.img
        read-only
        root=/dev/VolGroup00/rhel5_2
        append="dom0_mem=2G dom0_max_vcpus=2 --"
```

更には、このサンプルでは、カーネルパラメータ **"*rhgb quiet*"** を削除して、カーネルと **initscript** の出力がコンソールで 生成されるようにしています。append 行が hypervisor 引数として正しく解釈されるように 2つのダッシュは保持されることに注意して下さい。

第27章 XEN 設定ファイル

Red Hat Enterprise Linux はほとんどのタスクのために **libvirt** 設定ファイルを使用します。一部のユーザーは以下のような標準の変数を含んだ Xen 設定ファイルを 必要とするかも知れません。これらのファイル内の設定項目は単独引用句 (') で囲まなければなりません。 これらの設定ファイルは **/etc/xen** ディレクトリ内に存在します。

表27.1 libvirt 設定ファイル

項目	説明
pae	物理アドレス拡張子の設定データを指定します。
apic	高度なプログラム可能割り込みコントローラの設定データを指定します。
memory	メモリー容量をメガバイトで指定します。
vcpus	仮想 CPU の数量を指定します。
console	domain コンソールのエキスポート先のポート番号を指定します。
nic	仮想ネットワークインターフェイスの数量を指定します。
vif	ランダム割り当ての MAC アドレスと ドメインネットワークアドレスの使用に割り当てたブリッジを一覧表示します。
disk	ドメインにエキスポートするブロックデバイスを一覧表示して、物理デバイスを読み込み専用アクセスでドメインにエキスポートします。
dhcp	DHCP を使用してネットワークを有効にします。
netmask	設定した IP ネットマスクを指定します。
gateway	設定した IP ゲートウェイを指定します。
acpi	高度な電源インターフェイス設定の設定データを指定します。

以下のテーブル、表27.2「Xen 設定ファイルの参照」 は、**xm create --help_config** からの出力をフォーマットしたものです。

表27.2 Xen 設定ファイルの参照

パラメータ	説明

パラメータ	説明
vncpasswd=*NAME*	HVM ドメインにある VNC コンソール用のパスワード
vncviewer=**no| yes**	ドメイン内の vnc サーバーの為の vncviewer リスニングを引き起こします。 vncviewer のアドレスは、***VNC_SERVER=<host>:<port>*** を使用してカーネルコマンド行上のドメインへ渡されます。vnc に使用されるポートは 5500 + DISPLAY です。 フリーポートを持つ表示値は可能であれば選択されます。vnc=1 の時にのみ有効です。
vncconsole=**no|yes**	ドメインのグラフィカルコンソール用の vncviewer プロセスを引き起こします。 vnc=1 の時にのみ有効です。
name=*NAME*	ドメイン名。特有のものにします。
bootloader=*FILE*	ブートローダーへのパス
bootargs=*NAME*	ブートローダーへのパスの引数
bootentry=*NAME*	破棄されました。ブートローダーを経由したブートへのエントリ。***bootargs*** を使用
kernel=*FILE*	カーネルイメージへのパス
ramdisk=*FILE*	ramdisk へのパス
features=*FEATURES*	ゲストカーネル内で有効にする機能
builder=*FUNCTION*	ドメインをビルドするために使用する関数
memory=*MEMORY*	MB 表示のドメインメモリー
maxmem=*MEMORY*	MB 表示の最大ドメインメモリー
shadow_memory=*MEMORY*	MB 表示のドメインシャドーメモリー
cpu=*CPU*	VCPU0 を維持する CPU
cpus=*CPUS*	ドメインを稼働する CPUS
pae=*PAE*	HVM ドメインの PAE を有効化/無効化
acpi=*ACPI*	HVM ドメインの ACPI を有効化/無効化

パラメータ	説明
apic=*APIC*	HVM ドメインの APIC を有効化/無効化
vcpus=*VCPUs*	ドメイン内の仮想 CPUS の数
cpu_weight=*WEIGHT*	新しいドメインの cpu 量をセットします。*WEIGHT* は cpu に於けるドメインのシェアを制御する変動数です。
restart=*onreboot* | *always* | *never*	ドメインが終了時に再スタートすべきかどうかを決定するものです。 - ***onreboot*** = シャットダウンコード reboot を付けて終了時に再スタート - ***always*** = 常に終了時に再スタート、終了コードを無視。 - ***never*** = 終了時に再スタートしない、終了コードを無視。 これらはすべて破棄されました。代わりに***on_poweroff***、***on_reboot***、及び ***on_crash*** を使います。
on_poweroff=*destroy* | *restart* | *preserve* | *destroy*	ドメインが理由 'poweroff' で終了する時の動作 = - destroy : ドメインが通常にクリーンアップされる。 - restart: 古いドメインの代わりに新しいドメインがスタートする。 - preserve: ドメインが手動で破壊（例えば、**xm** destroy を使用）されるまではクリーンアップがされない。 - rename-restart: 古いドメインはクリーンアップされず改名されて、新しいドメインが代わりにスタートする。
on_reboot=*destroy* | *restart* | *preserve* | *destroy*	ドメインが理由 'reboot' で終了する時の動作 = - destroy: ドメインは普通にクリーンアップされる。 - restart: 新しいドメインが古いドメインの代わりにスタートする。 - preserve: ドメインが手動で破壊（例えば xm destroy の使用）されるまでクリーンアップはされない。 - rename-restart: 古いドメインはクリーンアップされず改名されて、新しいドメインが代わりにスタートする。
on_crash=*destroy* | *restart* | *preserve* | *destroy*	ドメインが理由 'crash' で終了する時の動作 = - destroy: ドメインは普通にクリーンアップされる。 - restart: 古いドメインの代わりに新しいドメインがスタートする。 - preserve: ドメインが手動で破壊（例えば xm destroy を使用）されるまでクリーンアップされない。 - rename-restart: 古いドメインはクリーンアップされず改名されて、新しいドメインが代わりにスタートする。
blkif=**no**|**yes**	ドメインをブロックデバイスバックエンドにする
netif=**no**|**yes**	ドメインをネットワークインターフェイスバックエンドにする

パラメータ	説明
***tpmif*=no|yes**	ドメインを TPM インターフェイスバックエンドにする
disk*=*phy:DEV,VDEV,MODE[,DOM]	ディスクデバイスをドメインに追加します。物理デバイスは *DEV* となり、ドメインに対して*VDEV* として現れます。*MODE* が **r** の場合は、 読み込み専用で、 *MODE* が **w** の場合は、読み込み/書き込みと なります。*DOM* が指定されると、それはディスクの為に使用するようにバックエンドドライバー ドメインを定義します。このオプションは複数のディスクを追加するために繰り返すことができます。
pci*=*BUS:DEV.FUNC	任意のパラメータ（16進法で）を使用して、ドメインに PCI デバイスを追加します。 例えば、***pci=c0:02.1a***。このオプションは、繰り返して 複数の PCI デバイスを追加することができます。
ioports*=*FROM[-TO]	パラメータ（16進法で）使用してドメインにレガシー I/O の幅を追加します。 ***ioports=02f8-02ff***。このオプションは、繰り返して 複数の I/O の幅を追加することができます。
irq*=*IRQ	ドメインに IRQ（割り込みの行）を追加します。例えば、***irq=7***。このオプションは繰り返して複数の IRQ を追加することができます。
usbport*=*PATH	そのポートへのパスで指定されている通りにドメインに物理 USB ポートを追加します。 このオプションは繰り返して複数のポートを追加することが出来ます。
vfb=type={vnc,sdl}, vncunused=1, vncdisplay=N, ***vnclisten=ADDR, display=DISPLAY,*** ***xauthority=XAUTHORITY, vncpasswd=PASSWORD,*** ***keymap*=*KEYMAP***	ドメインをフレームバッファのバックエンドにします。バックエンドタイプは、***sdl*** か ***vnc*** のいずれかにする必要があります。***type=vnc*** では、 外部の vncviewer を接続して下さい。サーバーはポート N+5900 の *ADDR* (デフォルト 127.0.0.1) で リッスンします。*N* はデフォルトでドメインの id となります。***vncunused=1*** となる場合は、サーバーは 5900 以上の任意の未使用ポートを探す試みをします。***type=sdl*** では、 該当する *DISPLAY* と *XAUTHORITY* を使用して viewer の１つが自動的にスタートし、デフォルトで現在のユーザーの設定になります。

パラメータ	説明
vif=type=TYPE, mac=MAC, bridge=BRIDGE, ip=IPADDR, ***script=SCRIPT, backend=DOM, vifname=****NAME*	該当する *MAC* アドレスとブリッジを使用してネットワーク インターフェイスを追加します。***vif*** は該当する設定スクリプトを コールすることにより、設定されます。タイプが指定されていない場合は、デフォルトは netfront となり、 ioemu ではありません。MAC が指定されていない場合は、ランダムに *MAC* アドレスが使用されます。指定されていないと、ネットワークバックエンドはそれ自身の *MAC* アドレスを選択します。ブリッジが指定されていないと、最初に見付かったブリッジが使用されます。スクリプトが 指定されていないと、デフォルトのスクリプトが使用されます。バックエンドが指定されていないと、デフォルトの バックエンドドライバードメインが使用されます。vif 名が指定されていない場合は、バックエンドの仮想インターフェイスが vifD.N と言う名前を取り、この D はドメイン id であり、N がインターフェイスの id となります。このオプションは 複数の vif を追加する時に繰り返されます。vif を指定することにより、必要に応じてインターフェイスの数を増加できます。
vtpm=instance=*INSTANCE*,backend=*DOM*	TPM インターフェイスを追加します。バックエンド側では、該当するインスタンスを 仮想 TPM インスタンスとして使用します。与えられた数値は単に好みのインスタンス番号です。 hotplug スクリプトはどのインスタンス番号が実際にドメインに割り当てられるかを決定します。 仮想マシンと TPM インスタンス番号は **/etc/xen/vtpm.db** 内で見る ことが出来ます。該当するドメインにはバックエンドを使用して下さい。
access_control=policy=*POLICY*,label=*LABEL*	セキュリティラベルとそれを定義するセキュリティポリシー参照を追加します。 ローカルの ssid 参照は、ドメインが開始、又は復帰する時点に算出されます。この 時点で、ポリシーがアクティブなポリシーに対してチェックされます。このようにして、「保存」又は「復元」の機能が処理されて、ローカルラベルはドメインが開始、又は 復帰するシステム上で自動的に正しく作成されます。
nics=*NUM*	破棄されています。代わりに空の vif エントリを使用します。ネットワークインターフェイスの 数を設定します。vif オプションを使用するとインターフェイスのパラメータを定義できます。 それ以外ではデフォルトが使用されます。vif を指定することにより必要に応じてインターフェイスの 数を増加できます。

パラメータ	説明
root=*DEVICE*	カーネルコマンド行に ***root***= パラメータをセットします。 NFS root には、例えば、**/dev/sda1** や **/dev/nfs** のようなデバイスを使用します。
extra=*ARGS*	カーネルコマンド行に追記するために余分の引数をセットします。
ip=*IPADDR*	カーネル IP インターフェイスアドレスをセットします。
gateway=*IPADDR*	カーネル IP ゲートウェイをセットします。
netmask=*MASK*	カーネル IP ネットマスクをセットします。
hostname=*NAME*	カーネル IP ホスト名をセットします。
interface=*INTF*	カーネル IP インターフェイス名をセットします。
dhcp=*off|dhcp*	カーネル dhcp オプションをセットします。
nfs_server=*IPADDR*	NFS root 用に NFS サーバーのアドレスをセットします。
nfs_root=*PATH*	root NFS ディレクトリのパスをセットします。
device_model=*FILE*	デバイスモデルプログラムへのパス
fda=*FILE*	fda へのパス
fdb=*FILE*	fdb へのパス
serial=*FILE*	シリアルか、pty か vc へのパス
localtime=**no|yes**	RTC がローカルタイムにセットされているかどうか
keymap=*FILE*	使用されるキーボードマップをセットします。
usb=**no|yes**	USB デバイスを模倣します。
usbdevice=*NAME*	追加する **USB** デバイスの名前
stdvga=**no|yes**	***std vga*** 又は Cirrus Logic グラフィックスを 使用します。
isa=**no|yes**	**ISA** のみのシステムをシミュレートします

パラメータ	説明
boot*=*a*|*b*|*c*|*d	デフォルトのブートデバイス
***nographic*=no|yes**	デバイスモデルがグラフィックスを使用すべきか？
soundhw*=*audiodev	デバイスモデルがオーディオデバイスを有効にすべきか？
vnc	デバイスモデルが VNC を使用すべきか？
vncdisplay	使用する VNC 表示
vnclisten	リッスンする VNC サーバー用のアドレス
vncunused	VNC サーバーには未使用のポートを見つけるようにします。vnc=1 の時にのみ有効。
sdl	デバイスモデルが SDL を使用すべきか？
display*=*DISPLAY	使用する X11 ディスプレイ
xauthority*=*XAUTHORITY	使用する X11 権限
uuid	使用する xenstore UUID (universally unique identifier) です。このオプションが セットされていないと、ランダムに１つ生成されます。仮想ネットワークインターフェイス用の MAC アドレスと同様です。これはクラスター全域に渡って特有である必要があります。

表27.4「設定パラメータのデフォルト値」値とデフォルトの値群をセットする Python パーサー関数で利用可能な設定パラメータの全てを一覧表示します。 このセッター関数はユーザーが指定する値でパーサーが何をするのか についての示唆を与えます。パーサーは値を Python 値として読み込み、それからそれを格納するために セッター関数に送り込みます。その値が有効な Python でなければ、曖昧なエラーメッセージが 表示されます。セッターがユーザーの値を拒否すると、ユーザーは妥当なエラーメッセージを 受けます。しかし、いい加減な設定ではそれが紛失したように見えます。セッターが値を受け付けても、 その値が正しくなければ、アプリケーションはまだ失敗する可能性があります。

表27.3 パラメータ値をセットする Python 関数

パーサー関数	有効な引数

パーサー関数	有効な引数
set_bool	受理される値: • **yes** • **y** • **no** • **yes**
set_float	Python の float() を使用する浮動小数点を受理します。例えば: • **3.14** • **10.** • **.001** • **1e100** • **3.14e-10**
set_int	Python の int() を使用する整数を受理します。
set_value	いずれの Python 値も受理します。
append_value	いずれの Python 値も受理して、アレイに格納されている以前の値に追記します。

表27.4 設定パラメータのデフォルト値

パラメータ	パーサー関数	デフォルト値
name	***setter***	*デフォルト値*
vncpasswd	***set_value***	*なし*
vncviewer	***set_bool***	*なし*
vncconsole	***set_bool***	*なし*
name	***set_value***	*なし*
bootloader	***set_value***	*なし*
bootargs	***set_value***	*なし*

パラメータ	パーサー関数	デフォルト値
bootentry	*set_value*	なし
kernel	*set_value*	なし
ramdisk	*set_value*	''
features	*set_value*	''
builder	*set_value*	'linux'
memory	*set_int*	128
maxmem	*set_int*	なし
shadow_memory	*set_int*	0
cpu	*set_int*	なし
cpus	*set_value*	なし
pae	*set_int*	0
acpi	*set_int*	0
apic	*set_int*	0
vcpus	*set_int*	1
cpu_weight	*set_float*	なし
restart	*set_value*	なし
on_poweroff	*set_value*	なし
on_reboot	*set_value*	なし
on_crash	*set_value*	なし
blkif	*set_bool*	0
netif	*set_bool*	0
tpmif	*append_value*	0
disk	*append_value*	[]

パラメータ	パーサー関数	デフォルト値
pci	*append_value*	*[]*
ioports	*append_value*	*[]*
irq	*append_value*	*[]*
usbport	*append_value*	*[]*
vfb	*append_value*	*[]*
vif	*append_value*	*[]*
vtpm	*append_value*	*[]*
access_control	*append_value*	*[]*
nics	*set_int*	*-1*
root	*set_value*	*"*
extra	*set_value*	*"*
ip	*set_value*	*"*
gateway	*set_value*	*"*
netmask	*set_value*	*"*
hostname	*set_value*	*"*
interface	*set_value*	*"eth0"*
dhcp	*set_value*	*'off'*
nfs_server	*set_value*	*なし*
nfs_root	*set_value*	*なし*
device_model	*set_value*	*"*
fda	*set_value*	*"*
fdb	*set_value*	*"*
serial	*set_value*	*"*

パラメータ	パーサー関数	デフォルト値
localtime	*set_bool*	*0*
keymap	*set_value*	*"*
usb	*set_bool*	*0*
usbdevice	*set_value*	*"*
stdvga	*set_bool*	*0*
isa	*set_bool*	*0*
boot	*set_value*	*'c'*
nographic	*set_bool*	*0*
soundhw	*set_value*	*"*
vnc	*set_value*	*なし*
vncdisplay	*set_value*	*なし*
vnclisten	*set_value*	*なし*
vncunused	*set_bool*	*1*
sdl	*set_value*	*なし*
display	*set_value*	*なし*
xauthority	*set_value*	*なし*
uuid	*set_value*	*なし*

パート VI. ヒントと裏技

生産性を向上する為のヒントと裏技

この章には、仮想化のパフォーマンス、拡張性、及び安定性を向上する為の 役立つヒントと裏技が含まれています。

第28章 ヒントと裏技

この章には仮想化のパフォーマンス、拡張性、及び安定性を改善する為の 役に立つヒントと裏技が含まれています。

28.1. 自動的にゲストを開始

このセクションはホストシステムのブート段階で、仮想化ゲストの自動スタートの 方法を説明しています。

ここにある例は、ゲストの開始に **virsh** を使用して、 ***TestServer*** を 介してホストがブートする時点に自動的にスタートします。

```
# virsh autostart TestServer
Domain TestServer marked as autostarted
```

これでゲストはホストと一緒に自動的にスタートします。

ゲストの自動ブートを停止するには、***--disable*** パラメータを使用します。

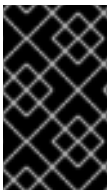
```
# virsh autostart --disable TestServer
Domain TestServer unmarked as autostarted
```

もうゲストは自動的にホストと一緒にスタートしません。

28.2. KVM と XEN HYPERVISOR との間での切り替え

このセクションは、KVM と Xen hypervisor 間での切り替えについて説明します。

Red Hat は一度に一つのみのアクティブ hypervisor をサポートします。



重要

現時点では、Xen ベースのゲストを KVM へ切り替えたり、又は KVM ベースの ゲストを Xen に切り替えたりするアプリケーションは有りません。ゲストはそれが 作成されたタイプの hypervisor でしか使用できません。

警告

この手順は Red Hat Enterprise Linux 5.4 又はそれ以降の Intel 64 又は AMD64 のバージョンにのみ利用可能です。他の構成や Red Hat Enterprise Linux の他のバージョンは サポートされていません。KVM は Red Hat Enterprise Linux 5.4 以前のバージョンでは 利用できません。

28.2.1. Xen から KVM へ

以下の手順は Xen hypervisor から KVM hypervisor への変更を説明しています。この 手順は kernel-xen パッケージがインストールしてあり、有効に なっていることを想定しています。

1. **KVM パッケージをインストール**
   まだ kvm パッケージをインストールしていない場合、インストールします。

```
# yum install kvm
```

2. **どのカーネルが使用中かを確認**
   kernel-xen パッケージがインストールされている可能性があります。**uname** コマンドを使用して、どのカーネルが稼働しているか判定します:

```
$ uname -r
2.6.18-159.el5xen
```

   現在、カーネル **"2.6.18-159.el5xen"** がシステムで稼働中です。 デフォルトカーネル、**"2.6.18-159.el5"** が稼働している場合は、 このサブステップは無視できます。

   a. **Xen カーネルをデフォルトカーネルに変更**
      grub.conf ファイルはどのカーネルがブートされるかを決定します。 デフォルトのカーネルを変更するには、**/boot/grub/grub.conf** ファイルを以下のように編集します。

```
default=1
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-159.el5.img
title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.18-159.el5
        module /vmlinuz-2.6.18-159.el5xen ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        module /initrd-2.6.18-159.el5xen.img
```

      default=1 パラメータを見てください。 これは GRUB ブートローダーに対して 2 つめのエントリである Xen カーネルをブートするように 指示しています。デフォルトを ***0*** （又は、デフォルトカーネルの番号）に 変更します。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-159.el5.img
title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.18-159.el5
        module /vmlinuz-2.6.18-159.el5xen ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        module /initrd-2.6.18-159.el5xen.img
```

3. **再起動して新しいカーネルをロードします。**
   システムを再起動します。コンピュータはデフォルトのカーネルで再スタートします。 KVM モジュールは、カーネルで自動的にロードされるはずです。KVM が稼働していることを 確認します:

```
$ lsmod | grep kvm
kvm_intel              85992  1
kvm                   222368  2 ksm,kvm_intel
```

   もし全てが正常に機能していれば、**kvm** モジュールと、 **kvm_intel** モジュール、あるいは **kvm_amd** モジュールが存在するはずです。

28.2.2. KVM から Xen へ

以下の手順は KVM hypervisor から Xen hypervisor への変更を説明しています。この 手順は kvm パッケージがインストールしてあり、有効になって いることを想定しています。

1. **Xen パッケージをインストール**
   まだ、kernel-xen と xen のパッケージをインストールしていない場合はインストールします。

```
# yum install kernel-xen xen
```

   kernel-xen パッケージはインストールされていて、無効に なっている可能性があります。

2. **どのカーネルが使用中かを確認**
   uname コマンドを使用すると、どのカーネルが実行中であるか 判定できます。

```
$ uname -r
2.6.18-159.el5
```

   現在のカーネル **"2.6.18-159.el5"** はシステム上で実行中です。 これがデフォルトのカーネルです。カーネルがその末尾に **xen** を 持つ場合（例えば、**2.6.18-159.el5xen**）、Xen カーネルが 実行中であるため、このサブステップは無視しても結構です。

   a. **デフォルトカーネルから Xen カーネルへ変更**
      grub.conf ファイルはどのカーネルがブートされるかを決定します。 デフォルトのカーネルを変更するには、**/boot/grub/grub.conf** ファイルを以下のように編集します。

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-159.el5.img
title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.18-159.el5
        module /vmlinuz-2.6.18-159.el5xen ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        module /initrd-2.6.18-159.el5xen.img
```

default=0 パラメータに注意して下さい。 これは、GRUB ブートローダに対して最初のエントリ（デフォルトカーネル）をブート するように指示しています。デフォルトを ***1*** （又は Xen カーネルの番号）に変更します:

```
default=1
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-159.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-159.el5 ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        initrd /initrd-2.6.18-159.el5.img
title Red Hat Enterprise Linux Server (2.6.18-159.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.18-159.el5
        module /vmlinuz-2.6.18-159.el5xen ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
        module /initrd-2.6.18-159.el5xen.img
```

3. **再起動して新しいカーネルをロードします。**
   システムを再起動します。コンピュータは Xen カーネルで再スタートするでしょう。 **uname** コマンドを使用して確認します:

   ```
   $ uname -r
   2.6.18-159.el5xen
   ```

   出力が末尾に **xen** を示すなら、Xen カーネルが稼働しています。

28.3. QEMU-IMG の使用

qemu-img コマンドラインツールは、Xen と KVM で使用している 各種ファイルシステムのフォーマットに使用されます。**qemu-img** は 仮想化ゲストイメージ、追加のストレージデバイス、及びネットワークストレージのフォーマットに 使用されるものです。**qemu-img** のオプションと使用法は以下の一覧に 示されています。

新規イメージ、又はデバイスのフォーマットと作成

サイズ ***size*** と形式 ***format*** の 新しいディスクイメージファイル名を作成します。

```
# qemu-img create [-6] [-e] [-b base_image] [-f format] filename [size]
```

base_image が指定されている場合、そのイメージは base_image との相違のみを 記録します。この場合、サイズは指定する必要はありません。base_image は、 **"commit"** と言うモニターコマンドを使用しない限りは、修正しません。

既存のイメージを別の形式に変換

認識されている形式を別のイメージ形式に変換するのには、***convert*** オプションが使用されます。

コマンドの形式:

```
# qemu-img convert [-c] [-e] [-f format] filename [-O output_format]
output_filename
```

フォーマット ***output_format*** を使用して、ディスクイメージ ***filename*** を ディスクイメージ ***output_filename*** に 変換します。ディスクイメージは ***-e*** オプションを使用して暗号化したり、 ***-c*** オプションで圧縮したりできます。

"qcow" フォーマットのみが暗号化、又は圧縮をサポートします。 圧縮は読み込み専用です。このことは圧縮したセクターが再書き込みされると、それは 圧縮無しのデータとして再書き込みされると言う意味です。

暗号化は非常に安全な 128-bit キーを持つ AES 形式を使用します。長いパスワード （16文字以上）の使用で最大の保護を得ることができます。

イメージ変換も、**qcow** 又は **cow** などの 増大する形式を使用する時には小さいイメージを取得するのに役立ちます。空のセクターは検知されて 目的地イメージから抑圧されます。

イメージ情報の取得

info パラメータはディスクイメージに関する情報を 表示します。**info** オプション用の形式は以下のようになります:

```
# qemu-img info [-f format] filename
```

これはディスクイメージのファイル名に関する情報を与えます。特に表示されたサイズとは 異なる可能性のあるディスク上で保存されたサイズを知る目的で使用します。vm スナップショットが ディスクイメージに格納されている場合は、それらも表示されます。

サポートされている形式

イメージのフォーマットは通常、自動的に行われます。以下のフォーマットが サポートされています:

raw

Raw ディスクイメージフォーマット（デフォルト）。このフォーマットは単純で、全ての 他のエミュレータに簡単にエキスポートできます。ユーザーのファイルシステムが holes（例えば、Linux での ext2 や ext3 で、Windows での NTFS で）をサポートしていれば、 書き込まれたセクターのみがスペースを保存します。**qemu-img info** を使用して、イメージ又は Unix/Linux 上で **ls -ls** で使用されている本当のサイズを知ることができます。

qcow2

QEMU イメージフォーマットは最も柔軟性のあるフォーマットです。より小さなイメージ （例えば Windows で使用フィルシステムが holes をサポートしない場合）、オプションの AES 暗号化、zlib ベースの圧縮、及び 複数 VM スナップショットのサポートなどを得るのに 使用します。

qcow

旧来の QEMU イメージフォーマットです。古いバージョンとの互換性の為にのみ含まれています。

cow

ユーザーモード Linux Copy On Write イメージフォーマットです。**cow** フォーマットは以前のバージョンとの互換性の為にのみ含まれています。Windows では機能しません。

vmdk

VMware 3 と 4 の互換イメージフォーマットです。

cloop

Linux 圧縮ループイメージは、例として Knoppix CD-ROM などに収納されている 直接圧縮 CD-ROM を再利用する時にのみ役に立ちます。

28.4. KVM でオーバーコミット

KVM hypervisor は CPU とメモリーの両方のオーバーコミットをサポートします。 オーバーコミットとは、システム上に存在する物理的リソース以上に仮想 CPU や 仮想メモリーを割り当てることです。CPU のオーバーコミットでは、低レベル設備の 仮想化サーバーやデスクトップがより少ない物理サーバー上で実行できるため、電力と 経費を節約します。

重要

CPU オーバーコミットは Xen hypervisor にはサポートされていません。Xen hypervisor での CPU のオーバーコミットはシステムの不安定性の原因及び、ホストと仮想化ゲストのクラッシュの 原因となります。

メモリーのオーバーコミット

ほとんどのオペレーティングシステムとアプリケーションは利用可能な RAM を常に 100% 使用することはありません。この状況を KVM で活用して使用物理メモリー 以上の仮想メモリーを仮想化ゲスト用に使用することが出来ます。

KVM の使用に於いては、仮想マシンは Linux プロセスです。KVM hypervisor 上のゲストは それに割り当てられた物理 RAM 集合を持っておらず、代わりにプロセスとして機能する わけです。各プロセスはより多くのメモリーを要求するとメモリーが割り当てられます。 KVM はこれを利用してゲストオペレーティングシステムのメモリー要求が上下する時に、 ゲスト用のメモリーを割り当てます。ゲストは、仮想化オペレーティングシステムが使用 すると見える量よりも少しだけ多く物理メモリーを使用します。

物理メモリーがほとんど全面的に使用されている時や、プロセスがしばらく休止している時は、 Linux はプロセスのメモリーをスワップに移動します。スワップは通常、ハードディスク、又は ソリッドステートドライブ上のパーティションであり、これを Linux が仮想メモリーを拡張する為に 使用します。スワップは RAM よりも確実に遅くなります。

KVM 仮想マシンは Linux プロセスであるため、仮想化ゲストで使用されるメモリーは、 ゲストが遊休時や低負荷の時にはスワップに移動されます。メモリーはスワップと 物理 RAM の合計サイズに対してコミット出来ます。これは、仮想化ゲストがそれらの全ての RAM を使用する時には問題の原因となります。**pdflush** プロセスに スワップされる仮想マシンのプロセス用に十分なスワップスペースが無いと、クリーンアッププロセスが スタートします。**pdflush** は、プロセスをキルしてメモリーを開放して、 システムがクラッシュしないようにします。**pdflush** はファイルシステムエラーの 原因になったり仮想化ゲストを起動不可にする可能性のある、仮想化ゲスト又は他のシステムのプロセスを 破棄します。

警告

十分なスワップが使用できない場合は、ゲストオペレーティングシステムは 強制的にシャットダウンされます。これによりゲストは運用不能になります。 これを回避するには、決して使用可能なスワップ以上のメモリーをオーバーコミット しないことです。

スワップパーティションは、メモリーのパフォーマンスを高速化するために低使用の メモリーをハードドライブにスワップするために使用します。スワップパーティションの デフォルトサイズは RAM のサイズとオーバーコミット率から算出されます。KVM を 使用してメモリーをオーバーコミットする計画があれば、スワップパーティションは 大きめにすることが推奨されます。推薦オーバーコミット率は 50% (0.5) です。 その数式は以下のようになります:

```
(0.5 * RAM) + (overcommit ratio * RAM) = Recommended swap size
```

Red Hat Knowledgebase はスワップパーティションのサイズを決定する 安全性と効率についての記事を収納しています。

システム内の物理 RAM サイズに対して仮想化ゲスト数の10倍のオーバーコミット率で 稼働することも可能です。これは特定のアプリケーション負荷状況でのみ機能できます （例えば、100% 以下の使用率のデスクトップ仮想化）。オーバーコミット率をセット することは厳格な数式ではありません。ご使用の環境に応じた率をテストしてカスタマイズ する必要があります。

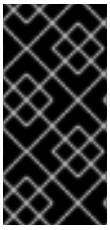
仮想化 CPU のオーバーコミット

KVM hypervisor は仮想化 CPU のオーバーコミットをサポートします。仮想化 CPU は 仮想化ゲストが許容する負荷限度までオーバーコミットが可能です。但し、100% 近くの負荷は 要求の未処理や、使用に適しない反応時間の原因になるため、VCPU のオーバーコミットには 注意が必要です。

仮想化 CPU は、仮想化ゲストが１つの VCPU のみを持つ時に最善のオーバーコミットと なります。Linux スケジューラはこのタイプの負荷で非常に効率良くなります。KVM は ５ VCPU の率で 100% 以下の負荷を持つゲストを安全にサポートするはずです。

対称型のマルチプロセッシングゲストは物理プロセッシングコアの数以上にオーバーコミット することは出来ません。例えば、４ VCPU を持つゲストは、デュアルコアプロセッサを持つ ホスト上では実行すべきではありません。物理プロセッシングコアの数に対して対称型 マルチプロセッシングゲストをオーバーコミットすると、深刻なパフォーマンス低下の 原因になります。

ゲスト VCPU を物理コアの数まで割り当てることが適切であり、それが期待どおりに 動作します。例えば、４ VCPU を持つ仮想化ゲストを４連コアのホストで実行することです。 100% 以下の負荷を持つゲストはこのセットアップで効率的に機能するはずです。



重要

徹底的なテスト無しでは、実稼働環境でのメモリーと CPU のオーバーコミットはしないで 下さい。100% のメモリーやプロセッシングリソースを使用するアプリケーションはオーバーコミット環境では使用不能になる可能性があります。導入する前にテストを 実行すべきです。

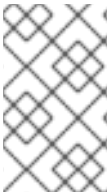
28.5. /ETC/GRUB.CONF の修正

このセクションでは、仮想化カーネルを使用するために安全にそして正確に **/etc/grub.conf** ファイルの変更をする方法を説明しています。Xen hypervisor を 使用するには、**xen** カーネルを使用しなければなりません。 既存の **xen** カーネルエントリのコピーにより、確実に全ての 重要な行をコピーして下さい。そうしないとシステムがブート時にパニックを起こします。 （**initrd** が **'0'** の連続を持つ）**xen** hypervisor 特有の値を必要とする場合は、それらを grub エントリの **xen** 行に追記する必要があります。

以下の出力は、kernel-xen パッケージを実行しているシステムからの **grub.conf** エントリの例です。個人のシステムの **grub.conf** は 異なるでしょう。以下の例の重要な部分は **title** 行から次の新しい行のセクションです。

```
#boot=/dev/sda
default=0
timeout=15
#splashimage=(hd0,0)/grub/splash.xpm.gz hiddenmenu
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console

title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1
        module /vmlinuz-2.6.17-1.2519.4.21.el5xen ro
root=/dev/VolGroup00/LogVol00
        module /initrd-2.6.17-1.2519.4.21.el5xen.img
```



注記

ユーザーのシステムが手動で編集されていたり、ある例からコピーされている場合は、その **grub.conf** は異なっている可能性があります。仮想化と **grub** の使用に関する詳細は *25章Xen カーネルブートパラメータの設定*でご覧下さい。

起動時にホストシステムに割り当てるメモリーの量を **256MB** にセットするには、 **grub.conf** の **xen** 行に **dom0_mem=256M** を追記する必要があります。 以前の例の中の **grub** 設定ファイルの変更バージョンは以下のようになります:

```
#boot=/dev/sda
default=0
timeout=15
#splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console

title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21.el5xen)
        root (hd0,0)
        kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1
dom0_mem=256MB
        module /vmlinuz-2.6.17-1.2519.4.21.el5xen ro
        root=/dev/VolGroup00/LogVol00
        module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

28.6. 仮想化拡張を確認

このセクションを活用してユーザーのシステムがハードウェア仮想化拡張を持つかどうか 判定して下さい。仮想化拡張 (Intel VT 又は AMD-V) は完全仮想化で必須項目です。

注記

ハードウェア仮想化拡張が存在しなければ、Red Hat kernel-xen パッケージを使用して Xen para-virtualization を利用することができます。

以下のコマンドを実行すると、CPU 仮想化拡張が利用可能であることを確認 できます:

```
$ grep -E 'svm|vmx' /proc/cpuinfo
```

以下の出力は **vmx** エントリを含んでおり、 Intel VT 拡張を持つ Intel プロセッサを示しています:

```
flags   : fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
        dts acpi mmx fxsr sse sse2 ss ht  tm syscall lm constant_tsc pni
monitor ds_cpl
        vmx est tm2 cx16 xtpr lahf_lm
```

以下の出力は **svm** エントリを含んでおり、 AMD-V 拡張を持つ AMD プロセッサを示しています:

```
flags   :  fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
        mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt lm 3dnowext 3dnow
pni cx16
        lahf_lm cmp_legacy svm cr8legacy ts fid vid ttp tm stc
```

"flags:" の内容は、システム上の各ハイパースレッド、コア、 又は CPU の複数倍として表ることがあります。

仮想化拡張は BIOS 内で無効にしてあるかも知れません。その拡張が出現しないと 完全仮想化は機能しません。手順31.1「BIOS 内で仮想化拡張を有効にする」 を参照して下さい。

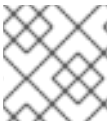
28.7. ゲストのタイプと実装を識別

以下のスクリプトは、システムが **para-virtualized** か、完全仮想化か、 又はベアメタルホストかを判定します。

```
#!/bin/bash
declare -i IS_HVM=0
declare -i IS_PARA=0
check_hvm()
{
        IS_X86HVM="$(strings /proc/acpi/dsdt | grep int-xen)"
          if [ x"${IS_X86HVM}" != x ]; then
           echo "Guest type is full-virt x86hvm"
           IS_HVM=1
        fi
}
check_para()
{
        if $(grep -q control_d /proc/xen/capabilities); then
          echo "Host is dom0"
          IS_PARA=1
        else
          echo "Guest is para-virt domU"
          IS_PARA=1
        fi
}
if [ -f /proc/acpi/dsdt ]; then
        check_hvm
fi

if [ ${IS_HVM} -eq 0 ]; then
        if [ -f /proc/xen/capabilities ] ; then
```

```
                check_para
        fi
    fi
if [ ${IS_HVM} -eq 0 -a ${IS_PARA} -eq 0 ]; then
        echo "Baremetal platform"
fi
```



注記

ホストの情報を得るには、**virsh capabilities** コマンドを 実行します。

28.8. 新規の特有 MAC アドレスを生成

一部のケースでは、新しい特有の *MAC アドレス*をゲスト用に 生成する必要があります。この資料作成時点では新規 MAC アドレス生成のためのコマンドライン ツールは有りません。以下に用意されているスクリプトはご使用のゲスト用の新規 MAC アドレスを 生成できます。そのゲスト用のスクリプトを **macgen.py** として保存します。保存ディレクトリから、**./macgen.py** を 使用してスクリプトを実行し、新規の MAC アドレスを生成します。サンプルの出力は以下のように なるでしょう:

```
$ ./macgen.py
00:16:3e:20:b0:11

#!/usr/bin/python
# macgen.py script to generate a MAC address for virtualized guests on Xen
#
import random
#
def randomMAC():
        mac = [ 0x00, 0x16, 0x3e,
                random.randint(0x00, 0x7f),
                random.randint(0x00, 0xff),
                random.randint(0x00, 0xff) ]
        return ':'.join(map(lambda x: "%02x" % x, mac))
#
print randomMAC()
```

使用ゲストの為の新規 MAC の別の生成方法

組込み済みのモジュール **python-virtinst** を使用しても 新規 MAC アドレスと ゲスト設定ファイル用の **UUID** を 生成できます。

```
# echo  'import virtinst.util ; print\
 virtinst.util.uuidToString(virtinst.util.randomUUID())' | python
# echo  'import virtinst.util ; print virtinst.util.randomMAC()' | python
```

上記のスクリプトは、下記に示してあるようにスクリプトファイルとしても 実装できます。

```
#!/usr/bin/env python
#  -*- mode: python; -*-
print ""
print "New UUID:"
import virtinst.util ; print
virtinst.util.uuidToString(virtinst.util.randomUUID())
```

```
print "New MAC:"
import virtinst.util ; print virtinst.util.randomMAC()
print ""
```

28.9. XEN ゲスト用のネットワークバンド幅を制限

一部の環境では、特定のゲストに利用可能なネットワークバンド幅を制限することが 必要かも知れません。これは、複数仮想マシンを実行しているホスト上で基本的な サービス品質を実装するのに使用できます。デフォルトで、ゲストは物理ネットワーク カードがサポートする利用可能なバンド幅セッティングを使用できます。物理ネットワーク カードは仮想マシンの仮想ネットワークインターフェイスの１つにマップされている 必要があります。Xen では、**VIF** エントリの **“rate”** パラメータ部分は仮想化ゲストを絞ることが出来ます。

この一覧は変数を取り扱います

rate

rate= オプションは仮想マシンの 設定ファイル内の **VIF=** エントリに追加して 仮想マシンのネットワークバンド幅を制限したり、又は時間ウィンドウ用に特定の 時間間隔を指定することができます。

時間ウィンドウ

時間ウィンドウは **rate=** オプションへの選択肢です:

デフォルトの時間ウィンドウは 50ms です。

小さい時間ウィンドウは少なめの突発伝送 (burst transmission) を提供しますが、 補充レートと遅延は増加します。

デフォルトの 50ms 時間ウィンドウは遅延とスループットの間の良いバランスです。 ほとんどのケースで変更は必要ないでしょう。

rate パラメータ値と使用法の例

rate=10Mb/s

ゲストからの発信ネットワークトラフィックを 10MB/s に制限します。

rate=250KB/s

ゲストからの発信ネットワークトラフィックを 250KB/s に制限します。

rate=10MB/s@50ms

バンド幅を 10MB/s に制限して 50ms 毎に 50KB のブロックをゲストに提供します。

仮想マシンの設定で、サンプルの **VIF** エントリは以下のように なります:

```
vif = [ 'rate=10MB/s , mac=00:16:3e:7a:55:1c, bridge=xenbr1']
```

この **rate** エントリは仮想マシンのインターフェイスを 発信トラフィック用に 10MB/s に制限します。

28.10. XEN プロセッサ類似物の設定

Xen は単数、又は複数の CPU と関連付けるために仮想 CPU を割り当てることができます。 これが本当のプロセッシングリソースを仮想化ゲストに割り当てます。このアプローチにより、 Red Hat Enterprise Linux は、デュアルコア、ハイパースレッド、あるいは、他の CPU 並行稼働 技術を使用する時にプロセッサリソースを最適化します。Xen クレジットスケジューラは自動的に 物理 CPU 間の仮想化 CPU のバランスを取り、システム使用を最大限にします。仮想化 CPU が物理 CPU に固定してある限りは、必要であれば Red Hat Enterprise Linux は、クレジット スケジューラが CPU を使い分けできるようにします。

I/O 集中のタスクを実行している場合、domain0 を稼働するためにハイパースレッドか、 あるいはプロセッサコア全体を専従させることが推奨されます。

KVM はデフォルトで Linux カーネルスケジューラを使用するため、KVM 用には これは必要ないことに注意して下さい。

CPU 類似物は **virsh** か **virt-manager** でセットできます:

virsh を使用して CPU 類似物をセットするには、 詳細を 仮想 CPU 類似物の設定 で参照して下さい。

virt-manager を使用して CPU 情報を設定し、表示するには、 その詳細を 「仮想 CPU の表示 」 で参照して下さい。

28.11. XEN HYPERVISOR の修正

ホストシステムの管理には、多くの場合ブート設定ファイル、**/boot/grub/grub.conf** への 変更がかかわってきます。複数ホストの設定ファイルの管理はその数の増加と共に難しくなってきます。 システム管理者は通常、複数の **grub.conf** ファイルの編集にはカットアンド ペーストを好むものです。これを実行する場合は、仮想化エントリ内の ５つの行すべてを含めることを 忘れないで下さい（さもないと、システムエラーの原因になります）。Hypervisor 特有の値はすべて、 **xen** の行にあります。この例では、正しい **grub.conf** 仮想化エントリを表示しています:

```
# boot=/dev/sda/
default=0
timeout=15
#splashimage=(hd0, 0)/grub/splash.xpm.gz

hiddenmenu
serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21. el5xen)
root (hd0, 0)
kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1
    module /vmlinuz-2.6.17-1.2519.4.21el5xen ro
root=/dev/VolGroup00/LogVol00
    module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

例えば、 hypervisor (dom0) のメモリーエントリを起動時に 256MB に変更するには、'xen' の行を編集して、エントリ、 **'dom0_mem=256M'** を追記します。以下の例は、hypervisor のメモリーエントリを修正した **grub.conf** です。

```
# boot=/dev/sda
default=0
timeout=15
#splashimage=(hd0,0)/grubs/splash.xpm.gz
```

```
hiddenmenu
serial --unit=0 --speed =115200 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
title Red Hat Enterprise Linux Server (2.6.17-1.2519.4.21. el5xen)
root (hd0,0)
kernel /xen.gz-2.6.17-1.2519.4.21.el5 com1=115200,8n1 dom0_mem=256MB
    module /vmlinuz-2.6.17-1.2519.4.21.el5xen ro
root=/dev/VolGroup00/LogVol00
    module /initrd-2.6.17-1.2519.4.21.el5xen.img
```

28.12. 非常にセキュアな FTPD

vsftpd は para-virtualized ゲストの インストールツリー（例えば、Red Hat Enterprise Linux 5 のレポジトリ）又は他のデータ へのアクセスを提供します。サーバーのインストール時に **vsftpd** を インストールしていない場合は、インストールメディアの **Server** ディレクトリから RPM パッケージを取得して、**rpm -ivh vsftpd*.rpm** を 使用してインストールできます（RPM パッケージは作業中のディレクトリになければならないことに 注意して下さい）。

1. **vsftpd** を設定するには、**vipw** を 使用して **/etc/passwd** を編集することにより、ftp ユーザーのホームディレクトリを para-virtualized ゲストのインストールツールの保存先となるディレクトリに変更します。ftp ユーザー用の エントリのサンプルは以下のようになります:

   ```
   ftp:x:14:50:FTP User:/xen/pub:/sbin/nologin
   ```

2. **chkconfig --list vsftpd** を使用して、**vsftpd** が 有効になっていないことを確認します:

   ```
   $ chkconfig --list vsftpd
   vsftpd          0:off   1:off   2:off   3:off   4:off   5:off
   6:off
   ```

3. **chkconfig --levels 345 vsftpd on** を実行して、ランレベル 3 と 4 と 5 で vsftpd が自動的にスタートするようにします。

4. **chkconfig --list vsftpd** コマンドを使用すると、**vsftpd** デーモン が システムブート時にスタートするように有効になっていることを確認できます:

   ```
   $ chkconfig --list vsftpd
   vsftpd          0:off   1:off   2:off   3:on    4:on    5:on
   6:off
   ```

5. **service vsftpd start vsftpd** を使用して、vsftpd サービスを スタートします:

   ```
   $service vsftpd start vsftpd
   Starting vsftpd for vsftpd:                  [  OK  ]
   ```

28.13. LUN 永続化の設定

このセクションは、ゲストとホストマシン上でマルチパスがある場合と無い場合に 於ける *LUN* 永続化の実装法を説明しています。

multipath を使用しないで LUN 永続化の実装

システムがマルチパスを使用していない場合、**udev** を使用して LUN 永続化を実装することができます。自分のシステムで LUN 永続化の実装をする前に、正しい UUID を取得することを確認して下さい。これらを取得した後は、**/etc** ディレクトリ内にある **scsi_id** ファイルを編集して LUN 永続化を設定できます。テキストエディタでこのファイルを開いたら、以下の行をコメントアウトする必要があります:

```
# options=-b
```

そして、これを次のパラメータで入れ換えます:

```
# options=-g
```

これが、返って来る UUID の為にシステム SCSI デバイス全てを監視するように udev に指示します。システム UUID を決定するには、以下のように **scsi_id** コマンドを使用します:

```
# scsi_id -g -s /block/sdc
*3600a0b80001327510000015427b625e*
```

この出力内の長い文字列は UUID です。UUID はユーザーが新規デバイスをシステムに 追加しても変化しません。デバイス群用のルールを作成するために各デバイス毎に UUID を取得して下さい。新規のデバイスルールを作成するには、**/etc/udev/rules.d** ディレクトリ内にある **20-names.rules** ファイルを 編集します。デバイス命名ルールは以下の形式に従います:

```
# KERNEL="sd*", BUS="scsi", PROGRAM="sbin/scsi_id", RESULT="UUID",
NAME="devicename"
```

既存の ***UUID*** と ***devicename*** を、上記の取り込んだ UUID エントリに入れ換えます。その規則は以下に似たものになります:

```
KERNEL="sd*", BUS="scsi", PROGRAM="sbin/scsi_id",
RESULT="3600a0b80001327510000015427b625e", NAME="mydevicename"
```

これが、**/dev/sd*** パターンに一致する全てのデバイスを有効にして、任意の UUID を検査するようになります。一致するデバイスを発見した場合、 **/dev/devicename** と呼ばれるデバイスノードを作成します。この例では、デバイスノードは **/dev/mydevice** にしています。最後に **/etc/rc.local** ファイルに以下の行を追記します:

```
/sbin/start_udev
```

multipath を使用した LUN 永続化の実装

マルチパス環境内で LUN 永続化を実装するには、マルチパスデバイス用のエイリアス名を定義する必要があります。例えば、**/etc/** ディレクトリ内にある **multipath.conf** ファイルを編集して、四つのデバイスエイリアスを定義する必要があります。

```
multipath  {
             wwid       3600a0b80001327510000015427b625e
             alias      oramp1
}
multipath  {
             wwid       3600a0b80001327510000015427b6
             alias      oramp2
}
```

```
multipath  {
              wwid       3600a0b80001327510000015427b625e
              alias      oramp3
}
multipath  {
              wwid       3600a0b80001327510000015427b625e
              alias      oramp4
}
```

これが、4 つ の LUN を定義します: **/dev/mpath/oramp1**、**/dev/mpath/oramp2**、**/dev/mpath/oramp3**、及び **dev/mpath/oramp4** です。デバイスは **/dev/mpath** ディレクトリ内に存在します。これらの LUN 名は、各 LUN 毎の **wwid** 上でエイリアス名を作成する為、再起動後にも残る永続性を持ちます。

28.14. ゲスト用の SMART ディスク監視を無効化

仮想ディスク上で実行しているため、及び物理ストレージがホストで管理されているため、 SMART ディスク監視は無効にすることができます。

```
/sbin/service smartd stop
/sbin/chkconfig --del smartd
```

28.15. 古い XEN 設定ファイルを整理

時間が経過すると、**/var/lib/xen** 内で、通常 **vmlinuz.******、及び **initrd.******** と名付けられたファイルが累積してきます。これらのファイルは ブートに失敗したか、又は他の理由で失敗した仮想マシンからの **initrd** 及び **vmlinuz** ファイルです。 これらのファイルはスタートアップシーケンス中に仮想マシンのブートディスクから抽出された一時 ファイルです。これらは仮想マシンが正常にシャットダウンした後は自動的に削除されるべきものです。 その後は、このディレクトリから古い無用なコピーを安全に削除できます。

28.16. VNC サーバーの設定

VNC サーバーを設定するには、**システム > 個人設定** と進んで、**リモートディスクトップ** を使用します。 別の方法として、**vino-preferences** コマンドを実行することもできます。

以下のステップは、専従の VNC サーバーセッションをセットアップします:

1. **~/.vnc/xstartup** ファイルを編集することで、 **vncserver** がスタートする時にはいつも GNOME セッションが スタートするようにします。**vncserver** スクリプトを初めて 実行する時は使用する VNC セッション用のパスワードを要求して来ます。

2. **xstartup** ファイルのサンプル:

```
#!/bin/sh
# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
#xsetroot -solid grey
#vncconfig -iconic &
#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
```

```
#twm &
if test -z "$DBUS_SESSION_BUS_ADDRESS" ; then
        eval `dbus-launch --sh-syntax –exit-with-session`
        echo "D-BUS per-session daemon address is: \
        $DBUS_SESSION_BUS_ADDRESS"
fi
exec  gnome-session
```

28.17. ゲスト設定ファイルのクローン

既存の設定ファイルをコピーして、全面的に新しいゲストを作成することができます。ゲスト設定ファイルの名前パラメータを修正しなければなりません。新しい、独自の名前が hypervisor に出現し、管理ユーティリティで見ることができます。完全に新しい UUID も生成する必要があります(**uuidgen** コマンド使用)。それから、**vif** エントリには、各ゲスト用に独自の MAC アドレスを定義する必要があります(既存のゲストからゲスト設定をコピーする場合は、スクリプトを作成して処理できます)。**xen** ブリッジの情報には、既存のゲスト設定ファイルを新規ホストに移動する場合、**xenbr** エントリを更新して、ユーザーのローカルネットワーキング設定に一致させます。デバイスエントリには、**'disk='** セクション内のエントリを修正して、それを正しいゲストイメージに向けます。

ゲスト上のこれらのシステム構成セッティングも修正する必要があります。**/etc/sysconfig/network** ファイルの HOSTNAME エントリを修正して、新しいゲストのホスト名に一致するようにしなければなりません。

/etc/sysconfig/network-scripts/ifcfg-eth0 ファイルの **HWADDR** アドレスを修正して、**ifconfig eth0** からの 出力に一致するようにしなければなりません。そして、静的 IP アドレスを使用している場合は、**IPADDR** エントリを修正する必要があります。

28.18. 既存ゲストとその設定ファイルを複製

このセクションは、新規のゲストを作成するために既存の設定ファイルをコピーする方法の 概要を示します。ゲストの設定ファイルには修正して、正常にゲストを複製するために ユーザーが認識しておくべき基幹パラメータがあります。

name

hypervisor によって知られており、管理ユーティリティに表示されているゲストの名前です。このエントリはシステム内で特有でなければなりません。

uuid

ゲスト用の特有の処理です。新規の UUID は **uuidgen** コマンドを 使用して再生成できます。UUID の出力のサンプルは以下のようになります:

```
$ uuidgen
a984a14f-4191-4d14-868e-329906b211e5
```

vif

- *MAC アドレス*は各ゲスト用に 特有の MAC アドレスを定義する必要があります。標準のツールが使用されると、これは自動的に行われます。既存のゲストからゲスト設定を コピーしている場合は、スクリプト「新規の特有 MAC アドレスを生成」の使用で達成できます。
- 新規のホストに向けて既存のゲスト設定ファイルを移動したり複製したりする場合は、**xenbr** エントリを調節して使用中のローカル ネットワーキング設定に対応するようにしな

ければなりません。（**brctl show** コマンドを使用すると、ブリッジ情報を取得できます）。

- デバイスエントリについては、そのエントリを **disk=** セクション内で調節して、正しいゲストイメージにポイントするように確認します。

ここで、使用ゲスト上のシステム設定セッティングを調節します:

/etc/sysconfig/network

HOSTNAME エントリをゲストの新しい**hostname** に 合わせて変更します。

/etc/sysconfig/network-scripts/ifcfg-eth0

- **HWADDR** アドレスを **ifconfig eth0** からの 出力に合わせて変更します。
- 静的 IP アドレスが使用されている場合は、**IPADDR** エントリを 変更します。

第29章 カスタムの LIBVIRT スクリプト作成

このセクションでは、**libvirt** の使用により、操作を簡単にするためのカスタムスクリプト作成を計画しているプログラマとシステム管理者に役立つ 一部の情報を提供します。

28章 *ヒントと裏技*は、**libvirt** を使用する新規のアプリケーションの作成を考慮しているプログラマに推奨される読み物です。

29.1. VIRSH を用いた XML 設定ファイルの使用

virsh は XML 設定ファイルを処理できます。これを 使用することで特殊なオプションを持つ大規模な導入のスクリプトの利便性を得ることが出来るでしょう。XML ファイルで定義されているデバイスを実行中の para-virtualized ゲストに追加することができます。例えば、ISO ファイルを **hdc** として実行中のゲストに追加するには、以下のような XML ファイルを作成します:

```
# cat satelliteiso.xml
<disk type="file" device="disk">
        <driver name="file"/>
        <source file="/var/lib/libvirt/images/rhn-satellite-5.0.1-11-
redhat-linux-as-i386-4-embedded-oracle.iso"/>
        <target dev="hdc"/>
        <readonly/>
</disk>
```

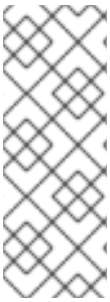
virsh attach-device を実行して、ISO を **hdc** として **"satellite"** と言うゲストに追加します:

```
# virsh attach-device satellite satelliteiso.xml
```

パート VII. トラブルシューティング

トラブルシューティングと問題解決への案内

以下の章では、仮想化の使用で遭遇するトラブルシューティング問題に於いて ユーザーを支援する情報を提供しています。



注記

バグの作成と修正をする継続的開発のため、このマニュアルには現在遭遇している問題は 記載されていないかも知れません。既知のバグとバグ修正に関する最新のリストには、 ご使用のバージョンとアーキテクチャ用の Red Hat Enterprise Linux 『Release Notes』 を お読み下さい。『Release Notes』 は Red Hat ウェブサイト www.redhat.com/docs/manuals/enterprise/ のドキュメントセクションで見つけることができます。

注記

Red Hat グローバルサポートサービス（https://www.redhat.com/apps/support/）までご連絡下さい。弊社のスタッフが喜んで問題解決のお手伝いをします。

第30章 XEN のトラブルシューティング

この章では、Xen に於けるトラブルシューティング問題でユーザーを支援するための 基本的な概念を扱います。この章で扱うトラブルシューティングのトピックには以下が 含まれます:

- Linux 及び仮想化のためのトラブルシューティングツール
- 問題判別の為のトラブルシューティング技術
- ログファイルの場所とログ内の情報の説明

この章では、読者に対して仮想化技術の問題場所を判別するための基礎を提供します。 トラブルシューティングは本で学習することが困難な実践と経験を要します。ユーザーは Red Hat Enterprise Linux 上で仮想化の実験とテストをして、トラブルシューティングの 能力を開発することが推奨されます。

このドキュメント内に解答を求めることが出来ない場合は、仮想化コミュニティからの オンライン解決案があるかも知れません。Linux 仮想化のウェブサイトの一覧は 「オンラインリソース」でご覧下さい。

30.1. XEN でのデバッギングとトラブルシューティング

このセクションはシステム管理アプリケーション、ネットワーキングユーティリティ、 及びデバグツールなどを要約しています。これらの標準のシステム管理者ツールとログを 活用してトラブルシューティングの手助けにできます:

トラブルシューティングの為の役に立つコマンドとアプリケーション

xentop

xentop はホストシステムとゲストドメインに関するリアルタイムの 情報を表示します。

xm

dmesg と **log** の使用

- **vmstat**
- **iostat**
- **lsof**

iostat コマンド、**mpstat** コマンド、及び **sar** コマンドは全て **sysstat** パッケージで用意されています。

これらの高度なデバグツールとログを活用してトラブルシュートの手助けにできます:

- **XenOprofile**
- **systemtap**
- **crash**
- **sysrq**
- **sysrq t**

- **sysrq w**

これらのネットワーキングツールを活用して仮想化ネットワーキング問題での トラブルシュートの手助けにできます:

- **ifconfig**
- **tcpdump**

  tcpdump は ネットワークのパケットを検査します。 **tcpdump** はネットワーク異常とネットワーク認証に関する 問題を見つけるのに役に立ちます。 **tcpdump** のグラフィカルバージョンが **wireshark** と言う名前で存在します。

- **brctl**

  brctl は Virtualization linux カーネル内のイーサネットブリッジ構成を検査して設定するネットワーキングツールです。これらのサンプルコマンドを実行するには root アクセスが必要になります:

```
# brctl show

bridge-name    bridge-id          STP  enabled  interfaces
-----------------------------------------------------------------
-------------
xenbr0              8000.fefffff       no        vif13.0
xenbr1              8000.ffffefff      yes       pddummy0
xenbr2              8000.fffffef       no        vif0.0

# brctl showmacs xenbr0

port-no           mac-addr                  local?       aging
timer

1                 fe:ff:ff:ff:ff:           yes            0.00
2                 fe:ff:ff:fe:ff:           yes            0.00


# brctl showstp xenbr0

xenbr0

bridge-id              8000.feffffffff

designated-root        8000.feffffffff

root-port              0                   path-cost             0

max-age                20.00               bridge-max-age
20.00

hello-time             2.00                bridge-hello-time
2.00

forward-delay          0.00                bridge-forward-delay
0.00
```

```
aging-time            300.01

hello-timer            1.43                   tcn-timer
0.00

topology-change-timer  0.00                   gc-timer
0.02
```

以下の一覧では、Red Hat Enterprise Linux 5 上の仮想化のトラブルシューティングの為の その他の役立つコマンドを表示しています。ここで言及してある全てのユーティリティは Red Hat Enterprise Linux 5 の **Server** レポジトリ内で見ることが できます。

- **strace** は他のプロセスで受信されて使用されたシステムコールと イベントを追跡するコマンドです。
- **vncviewer:** 使用中のサーバー、又は仮想マシン上で 稼働している VNC サーバーに接続します。 **yum install vnc** コマンドを使用すると **vncviwer** がインストールできます。
- **vncserver:** 使用するサーバー上でリモートデスクトップを 開始します。そしてユーザーがリモートセッションを通じて virt-manager などのグラフィカル ユーザーインターフェイスを実行できるようにします。**vncserver** の インストールには、**yum install vnc-server** コマンドを使用します。

30.2. ログファイルの概要

Virtualization が収納されている Red Hat Enterprise Linux 5.0 をネットワークインフラ ストラクチャに導入する場合、ホストの Virtualization ソフトウェアは重要な設定、ログファイル、及び他のユーティリティの為に多くの特定ディレクトリを使用します。全ての Xen ログファイルは標準の ASCII ファイル形式であり、テキストエディタで簡単に アクセスできます:

- Xen の設定ファイルディレクトリは **/etc/xen/** です。 このディレクトリには、**xend** デーモンと他の仮想マシン設定ファイルが含まれています。ネットワークスクリプトファイルは **scripts** ディレクトリ内に存在します。
- 全ての Xen ログファイルは **/var/log/xen** ディレクトリに 格納してあります。
- 全てのファイルベースイメージ用のデフォルトディレクトリは **/var/lib/libvirt/images** ディレクトリです。
- Xen カーネル情報は **/proc/xen/** ディレクトリに格納して あります。

30.3. ログファイルの説明

Xen は **xend** デーモンと **qemu-dm** プロセスを特長とします。これらの二つのユーティリティは複数ログファイルを **/var/log/xen/** ディレクトリに書き込みます:

- **xend.log** は **xend** デーモンにより収集された通常のシステムイベント、又はオペレータ先導のアクションのどちらでも、全てのデータを含むログファイルです。全ての仮想マシン運用 (作成、シャットダウン、破棄など) はこのログに現われます。 **xend.log** は通常、ユーザーがイベントやパフォーマンス問題を追跡する時に調査する最初の場所です。ここには、エラーメッセージの 詳細なエントリと状況が含まれています。
- **xend-debug.log** は **xend** と Virtualization サブシステム (フレームバッファ、Python スクリプトなど) からのイベントエラーの記録を含むログファイルです。

- **xen-hotplug-log** は、hotplug イベントからのデータを含むログファイルです。デバイス、又はネットワークのスクリプトがオンラインに出ない場合はイベントはここに表示されます。

- **qemu-dm.[PID].log** は各完全仮想化ゲスト用の **qemu-dm** プロセスで作成されたログファイルです。このログファイルを使用する時には、仮想マシン上の **qemu-dm** プロセスを隔離する為のプロセス引数を検査するのに **ps** コマンドを使用することで任意の**qemu-dm** プロセス PID を取り込む必要があります。この [PID] シンボルは実際の PID **qemu-dm** プロセスで入れ換えることに注意して下さい。

仮想マシンマネージャでなんらかの問題に遭遇した場合、**/.virt-manager** ディレクトリ内にある **virt-manager.log** ファイルの生成データで確認することができます。仮想マシンマネージャを開始する度に既存のログファイル内容は書き変えられることに注意して下さい。システムエラーの後では、仮想マシンマネージャを再起動する前に **virt-manager.log** をバックアップすることを忘れないで下さい。

30.4. 重要ディレクトリの場所

Xen でエラー追跡をしたり問題のトラブルシュートをしたりする時、認識しておくべき 他のユーティリティやログファイルがあります:

- 仮想ゲストイメージは **/var/lib/libvirt/images** ディレクトリ内にあります。

- **xend** デーモンを再開始する場合、それは **/var/lib/xen/xend-db** ディレクトリにある **xend-database** を更新します。

- 仮想マシンダンプ (**xm dump-core** コマンドで実行) は **/var/lib/xen/dumps** ディレクトリにあります。

- **/etc/xen** ディレクトリには、システムリソースの管理に使用する設定ファイルが含まれています。**xend** デーモン設定ファイルは **/etc/xen/xend-config.sxp** です。このファイルを編集して システム全体の変更を実装し、そしてネットワーキングを設定することができます。しかし、 **/etc/xen/** フォルダ内でファイルを手動で編集することは推奨 できません。

- **proc** フォルダは、ユーザーがシステム情報を取得できるようにするもう１つのリソースです。以下の proc エントリは **/proc/xen** ディレクトリにあります:

/proc/xen/capabilities

/proc/xen/balloon

/proc/xen/xenbus/

30.5. ログを使用したトラブルシューティング

Xen のインストール問題に遭遇した時には、ホストシステムの二つのログを参照して、トラブルシューティングに役立てることができます。**xend.log** ファイルには、**xm log** コマンドを実行する時と同じ基本情報があります。このログは **/var/log/** ディレクトリにあります。カーネルを実行してドメインを作成するためのログエントリの例を以下にしめします:

```
[2006-12-27 02:23:02 xend] ERROR (SrvBase: 163) op=create: Error creating
domain: (0, 'Error')
Traceback (most recent call list)
File "/usr/lib/python2.4/site-packages/xen/xend/server/SrvBase.py" line 107
in_perform val = op_method (op,req)
```

```
File
"/usr/lib/python2.4/site-packages/xen/xend/server/SrvDomainDir.py line 71
in op_create
raise XendError ("Error creating domain: " + str(ex))
XendError: Error creating domain: (0, 'Error')
```

もう１つのログファイル、 **xend-debug.log** は、システム管理者に非常に役に立ちます。これには、 **xend.log** 以上の詳細情報が含まれています。以下に同じカーネルドメイン作成問題についての同じエラーデータを示します:

```
ERROR: Will only load images built for Xen v3.0
ERROR: Actually saw: GUEST_OS=netbsd, GUEST_VER=2.0, XEN_VER=2.0;
LOADER=generic, BSD_SYMTAB'
ERROR: Error constructing guest OS
```

カスタマサポートに連絡する場合、技術サポートスタッフと通信する時に常にこれらの両方のログファイルコピーを含めて下さい。

30.6. シリアルコンソールを使用したトラブルシューティング

シリアルコンソールは難しい問題のトラブルシュートに役に立ちます。Virtualization カーネルがクラッシュしてhypervisor がエラーを生成する場合は、ローカルホスト上でエラーを追跡する方法はありません。しかし、シリアルコンソールでは、ユーザーはそれをリモートホストでキャプチャできます。その為には ホストを設定してデータをシリアルコンソールに出力できるようにしなければなりません。そして、そのデータをキャプチャするためにリモートホストを設定する必要があります。これを実行するには、**grub.conf** ファイル内のオプションを修正して、**com1 /dev/ttyS0** 上の 38400-bps シリアルコンソールを有効にする必要があります:

```
title Red Hat Enterprise Linux (2.6.18-8.2080_xen0)
                root (hd0,2)
                kernel /xen.gz-2.6.18-8.el5 com1=38400,8n1
                module /vmlinuz-2.618-8.el5xen ro root=LABEL=/rhgb quiet
console=xvc console=tty xencons=xvc
                module /initrd-2.6.18-8.el5xen.img
```

sync_console は非同期の hypervisor コンソール出力でハングの原因となる問題の判定の手助けになります。そして **"pnpacpi=off"** はシリアルコンソール上で入力を破損する問題を回避します。パラメータ **"console=ttyS0"** と **"console=tty"** はカーネルエラーが通常の VGA コンソールと シリアルコンソールの両方でログされることを意味します。それから、 **ttywatch** をインストール及び設定をして、標準の null-modem ケーブルで接続されたリモートホスト上でデータをキャプチャすることができます。例えば、リモートホスト上で以下のように入力します:

注記

Itanium® アーキテクチャ上の シリアルコンソールを介して hypervisor にアクセスするには、ELILO 内でコンソールを 有効にしなければなりません。ELILO の設定に関しては *26章ELILO の設定* を参照して下さい。

```
ttywatch --name myhost --port /dev/ttyS0
```

これが **/dev/ttyS0** からの出力をファイル **/var/log/ttywatch/myhost.log** にパイプします。

30.7. PARA-VIRTUALIZED ゲストコンソールのアクセス

Para-virtualized のゲストオペレーティングシステムは自動的にデータを ホストオペレーティングシステムに送り込むように仮想テキストコンソールを設定できます。 以下のコマンドを使用してゲストの仮想コンソールに接続します:

```
# virsh console [guest name, ID or UUID]
```

virt-manager を使用して仮想テキストコンソールを 表示することができます。仮想マシンの詳細ウィンドウ内で、**表示** メニューから **シリアルコンソール** を選択します。

30.8. 完全仮想化ゲストコンソールのアクセス

完全仮想化ゲストオペレーティングシステムは自動的にテキストコンソールの使用を設定します。しかし、相異点は、カーネルゲストが設定されていないことです。完全仮想化ゲストと共にゲスト仮想シリアルコンソールが稼動するようにするには、ゲストの **grub.conf** ファイルを修正して、**'console =ttyS0 console=tty0'** パラメータを含むようにする必要があります。これにより、カーネルメッセージが仮想シリアルコンソール (及び通常のグラフィカルコンソール) に確実に送信されることになります。完全仮想化ゲスト内で仮想シリアルコンソールを使用する予定であれば、**/etc/xen/** ディレクトリ内で設定ファイルを編集しなければなりません。ホストドメインで以下のコマンドを入力して、シリアルコンソールにアクセスできます:

```
# virsh console
```

virt-manager を使用して仮想テキストコンソールを 表示することができます。仮想マシンの詳細ウィンドウ内で、**表示** メニューから **シリアルコンソール** を選択します。

30.9. ゲストディスクイメージ上のデータにアクセス

二つの個別のアプリケーションを使用して、ゲストディスクイメージ内からデータへのアクセスの手助けにできます。これらのツールを使用する前に、先ずゲストをシャットダウンする必要があります。ゲストと dom0 からシステムファイルにアクセスすることはシステム破損の可能性を持ちます。

kpartx アプリケーションを使用して、パーティション化したディスクや LVM ボリュームグループを処理することができます:

```
yum install kpartx
```

```
kpartx -av /dev/xen/guest1
add map guest1p1 : 0 208782 linear /dev/xen/guest1 63
add map guest1p2: 0 16563015 linear /dev/xen/guest1 208845
```

二番目のパーティションの LVM ボリュームにアクセスするには、**vgscan** を使用して**LVM** を再スキャンし、**vgchange -ay** コマンドを使用して そのパーティションのボリュームグループ(デフォルトで **VolGroup00**) を アクティベートする必要があります:

```
# kpartx -a /dev/xen/guest1
#vgscan
Reading all physical volumes . This may take a while...
Found volume group "VolGroup00" using metadata type lvm2
# vgchange -ay VolGroup00
2 logical volumes in volume group VolGroup00 now active.
```

```
# lvs
LV VG Attr Lsize Origin Snap% Move Log Copy%
LogVol00 VolGroup00 -wi-a- 5.06G
LogVol01 VolGroup00 -wi-a- 800.00M
# mount /dev/VolGroup00/LogVol00 /mnt/
....
#umount /mnt/
#vgchange -an VolGroup00
#kpartx -d /dev/xen/guest1
```

終了したら、論理ボリュームを **vgchange -an** で活動停止すること、 **kpartx-d** でパーティションを削除すること、及び **losetup -d** でループデバイスを削除することを忘れないで下さい。

30.10. 一般的な XEN の問題

xend サービスを開始しようとして、何も起こらない場合、**virsh list** と入力して、以下の情報を受けます:

```
Error: Error connecting to xend: Connection refused. Is xend running?
```

手動で **xend start** の実行を試みて、他のエラーも受理します:

```
Error: Could not obtain handle on privileged command interfaces (2 = No
such file or directory)
Traceback (most recent call last:)

File "/usr/sbin/xend/", line 33 in ?

from xen.xend.server. import SrvDaemon

File "/usr/lib/python2.4/site-packages/xen/xend/server/SrvDaemon.py" ,
line 26 in ?

from xen.xend import XendDomain

File "/usr//lib/python2.4/site-packages/xen/xend/XendDomain.py" , line 33,
in ?


from xen.xend import XendDomainInfo

File "/usr/lib/python2.4/site-packages/xen/xend/image.py" , line37, in ?

import images

File "/usr/lib/python2.4/site-packages/xen/xend/image.py" , line30, in ?

xc = xen.lowlevel.xc.xc ()

RuntimeError: (2, 'No such file or directory' )
```

ここで、最も発生した可能性のあることは、ユーザーが **kernel-xen** 以外のカーネルでホストを再起動したことです。これを修正するには、起動時に **kernel-xen**カーネルを選択することです（又はそれを **grub.conf** ファイル内でデフォルトのカーネルにセットします）。

30.11. ゲスト作成のエラー

ゲストを作成しようとしている時に、**"Invalid argument"** のエラーメッセージが出ることがあります。これは通常、ブートしようとしているカーネルイメージが hypervisor に不適合と言う意味です。例として、PAE のみの FC6 hypervisor で非 PAE FC5 カーネルを実行しようとすると、この状況が出ます。

yum の更新をして、新規カーネルを取得したとします。すると **grub.conf** デフォルトのカーネルは Virtualization カーネルではなく、生のままのカーネルに戻ることになります。

この問題を是正するには、**/etc/sysconfig/kernel/** ディレクトリにあるデフォルトカーネル RPM を修正する必要があります。**grub.conf** ファイルの中で **kernel-xen** パラメータがデフォルトのオプションとして設定されていることを確認しなければなりません。

30.12. シリアルコンソールを使用したトラブルシューティング

Linux カーネルは情報をシリアルポートへ出力できます。これは、カーネルパニックや、 ビデオデバイス又はヘッドレスサーバーでのハードウェア問題のデバッグに役立ちます。 このセクションのサブセクションでは、Red Hat Enterprise Linux virtualization カーネルと それらの仮想化ゲストを実行しているマシン用のシリアルコンソール出力のセットアップに ついて説明していきます。

30.12.1. Xen 用のシリアルコンソール出力

デフォルトでは、Xen のシリアルコンソールは無効になっており、シリアルポートから データは出力されません。

シリアルポート上でカーネル情報を受信するには、適切なシリアルデバイスパラメータを 設定することにより **/boot/grub/grub.conf** ファイルを修正します。

使用するシリアルコンソールが **com1** にある場合は、 以下に示した位置に、***com1=115200,8n1*** の行、 ***console=tty0*** の行、及び ***console=ttyS0,115200*** の 行を挿入することにより **/boot/grub/grub.conf** を変更します。

```
title Red Hat Enterprise Linux 5 i386 Xen (2.6.18-92.el5xen)
        root (hd0, 8)
        kernel /boot/xen.gz-2.6.18-92.el5 com1=115200,8n1
        module /boot/vmlinuz-2.6.18-92.el5xen ro root=LABEL=VG_i386
console=tty0
        console=ttyS0,115200
        module /boot/initrd-2.6.18-92.el5xen.img
```

使用するシリアルコンソールが **com2** にある場合は、 以下に示してある位置に ***com2=115200,8n1 console=com2L*** 行、 ***console=tty0*** 行、及び ***console=ttyS0,115200*** 行を挿入することにより、**/boot/grub/grub.conf** を 変更します。

```
title Red Hat Enterprise Linux 5 i386 Xen (2.6.18-92.el5xen)
        root (hd0, 8)
        kernel /boot/xen.gz-2.6.18-92.el5 com2=115200,8n1 console=com2L
        module /boot/vmlinuz-2.6.18-92.el5xen ro root=LABEL=VG_i386
console=tty0
        console=ttyS0,115200
        module /boot/initrd-2.6.18-92.el5xen.img
```

その変更を保存してホストを再起動します。そうすると、先のステップで選択したシリアル上(**com1**, **com2** など）に hypervisor がシリアルデータを出力します。

com2 ポートを使用している例では、パラメータ ***console=ttyS0*** が ***vmlinuz*** 行上で使用されています。***console=ttyS0*** として 使用されている各ポートの動作は標準の Linux 動作ではありません。これは Xen 環境特有の 動作です。

30.12.2. para-virtualized ゲストからの Xen シリアルコンソール出力

このセクションでは、Red Hat Enterprise Linux para-virtualized ゲスト用に 仮想化シリアルコンソールの設定方法を説明しています。

para-virtualized ゲストからのシリアルコンソール出力は **"virsh console"** の 使用で取り込むか、あるいは **virt-manager** の **"シリアル"** ウィンドウ内に取り込めます。以下の手順を使用して仮想シリアルコンソールのセットアップを します:

1. para-virtualized ゲストへログイン
2. **/boot/grub/grub.conf** を以下のように編集します:

```
Red Hat Enterprise Linux 5 i386 Xen (2.6.18-92.el5xen)
        root (hd0, 0) kernel /boot/vmlinuz-2.6.18-92.el5xen ro
root=LABEL=VG_i386 console=xvc0
        initrd /boot/initrd-2.6.18-92.el5xen.img
```

3. para-virtualized ゲストを再起動

ここで virt-manager の "Serial Console" と "virsh console" にカーネルメッセージを 受け取るはずです。

para-virtualized ドメインのシリアルコンソール出力のログ

Xen デーモン (**xend**) を設定することで para-virtualized ゲストのシリアルコンソールからの出力をログすることができます。

xend を設定するには、**/etc/sysconfig/xend** を編集します。以下のようにエントリを変更します:

```
# Log all guest console output (cf xm console)
#XENCONSOLED_LOG_GUESTS=no
```

から

```
# Log all guest console output (cf xm console)
XENCONSOLED_LOG_GUESTS=yes
```

ゲストのシリアルコンソール出力のロギングをアクティブにするために再起動します。

ゲストシリアルコンソールからのログは **/var/log/xen/console** ファイルに格納されています。

30.12.3. 完全仮想化ゲストからのシリアルコンソール出力

このセクションでは、完全仮想化ゲスト用のシリアルコンソール出力を 有効にする方法を説明します。

完全仮想化ゲストのシリアルコンソール出力は **"virsh console"** コマンドを使用して見ることができます。

完全仮想化ゲストのシリアルコンソールはいくつかの制限を持つことに注意して 下さい。現在の制限には以下が含まれます:

- xend を使用した出力のログは利用できません。
- 出力データは欠如するか、混乱しているかも知れません。

シリアルポートは Linux で **ttyS0**、又は Windows で **COM1** と呼ばれます。

仮想シリアルポートに情報を出力するには、仮想化オペレーティングシステムを設定する 必要があります。

完全仮想化 Linux ゲストからのカーネル情報をドメインに出力するには、 **"*console=tty0 console=ttys0,115200*"** と言う行を 挿入することで **/boot/grub/grub.conf** ファイルを修正します。

```
title Red Hat Enterprise Linux Server (2.6.18-92.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-92.el5 ro root=/dev/volgroup00/logvol00
        console=tty0 console=ttys0,115200
        initrd /initrd-2.6.18-92.el5.img
```

ゲストを再起動します。

"virsh console" コマンドを使用するとシリアルコンソールの メッセージを見ることができます。

重要

完全仮想化ドメインからのシリアルコンソールメッセージは、para-virtualized ゲストとは 異なり、**/var/log/xen/console** にはログされていません。

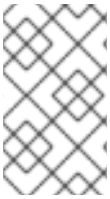
30.13. ゲスト設定ファイル

virt-manager 又は virt-install のツールを使用して、 Red Hat Enterprise Linux 5.0 で 新規ゲストを作成する場合、ゲストの設定ファイルは **/etc/xen** ディレクトリ内に 自動的に作成されます。以下の設定ファイルの例は、標準的な para-virtualized ゲスト用です:

```
name = "rhel5vm01"
memory = "2048"
disk = ['tap:aio:/var/lib/libvirt/images/rhel5vm01.dsk,xvda,w',]
vif = ["type=ieomu, mac=00:16:3e:09:f0:12 bridge=xenbr0',
"type=ieomu, mac=00:16:3e:09:f0:13 ]
vnc = 1
vncunused = 1
uuid = "302bd9ce-4f60-fc67-9e40-7a77d9b4e1ed"
bootloader = "/usr/bin/pygrub"
vcpus=2
on_reboot = "restart"
on_crash = "restart"
```

serial="pty" は設定ファイル用のデフォルトであることに注意して下さい。この設定ファイルの例は、完全仮想化ゲスト用です:

```
name = "rhel5u5-86_64"
builder = "hvm"
memory = 500
disk = ['/var/lib/libvirt/images/rhel5u5-x86_64.dsk.hda,w']
vif = [ 'type=ioemu, mac=00:16:3e:09:f0:12, bridge=xenbr0', 'type=ieomu,
mac=00:16:3e:09:f0:13, bridge=xenbr1']
uuid = "b10372f9-91d7-ao5f-12ff-372100c99af5'
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader/"
vnc = 1
vncunused = 1
apic = 1
acpi = 1
pae = 1
vcpus =1
serial ="pty" # enable serial console
on_boot = 'restart'
```



注記

Xen 設定ファイルの編集はサポートされていません。そのため、**virsh dumpxml** と **virsh create** (又は **virsh edit**) を使用して **libvirt** 設定ファイル (xml ベース) を編集して 下さい。これらはエラーチェックとセイフティチェック機能も持っています。

30.14. XEN エラーメッセージの解釈

次のようなエラー表示があるとします:

```
failed domain creation due to memory shortage, unable to balloon domain0
```

RAM が充分にないと、ドメインは失敗する可能性があります。Domain0 は、新規作成のゲスト用に領域を提供する為に縮小はしません。このエラーについては **xend.log**ファイルを確認することになります:

```
[2006-12-21] 20:33:31 xend 3198] DEBUG (balloon:133) Balloon: 558432 Kib
free; 0 to scrub; need 1048576; retries: 20
[2006-12-21] 20:33:31 xend. XendDomainInfo 3198] ERROR (XendDomainInfo:
202
Domain construction failed
```

xm list Domain0 コマンドを使うと domain0 が使用しているメモリーの量をチェックできます。domain0 が縮小しなければ、コマンド **virsh setmem dom0 NewMemSize** を使用してメモリーをチェックできます。

次のようなエラー表示があるとします:

```
wrong kernel image: non-PAE kernel on a PAE
```

このメッセージは、ユーザーが Hypervisor 上でサポートされていないゲストカーネルイメージを実行しようとしていることを示すものです。これは、Red Hat Enterprise Linux 5 ホスト上のので 非 PAE の

para-virtualized ゲストカーネルを起動しようとする時に発生します。Red Hat kernel-xen パッケージは PAE と 64bit アーキテクチャを 持つゲストカーネルのみをサポートします。

次のコマンドを入力します:

```
# xm create -c va-base

Using config file "va-base"
Error: (22, 'invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] ERRORs
(XendDomainInfo:202) Domain construction failed

Traceback (most recent call last)
File "/usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py", line
195 in create vm.initDomain()
File " /usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py", line
1363 in initDomain raise VmError(str(exn))
VmError: (22, 'Invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XenDomainInfo: 1449]
XendDlomainInfo.destroy: domain=1
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XenDomainInfo: 1457]
XendDlomainInfo.destroy:Domain(1)
```

32 bit 非-PAE カーネルを実行する必要がある場合は、使用中のゲストを完全仮想化の仮想マシンとして実行する必要があります。para-virtualized のゲストの為に 32bit PAE ゲストを実行する必要がある場合は、32bit PAE hypervisor が必要となります。para-virtualized のゲストの為に、64bit PAE ゲストを実行する必要がある場合は、64bit PAE hypervisor が必要となります。完全仮想化のゲスト用には、64bit hypervisor を持つ 64bit ゲストを実行しなければなりません。Red Hat Enterprise Linux 5 i686 に 同梱してある 32bit PAE hypervisor は 32bit PAE para-virtualized ゲスト OS と 32 bit の完全仮想化ゲスト OS の実行のみをサポートします。64bit hypervisor は 64bit para-virtualized のゲストのみをサポートします。

これは、完全仮想化の HVM ゲストを Red Hat Enterprise Linux 5 システムに移動する時に 発覚します。ユーザーのゲストは起動に失敗する可能性があり、コンソール画面にエラーが出るでしょう。設定ファイル内の PAE エントリをチェックして、pae=1 であることを確認します。32bit のディストリビューションを使用する必要があります。

次のようなエラー表示があるとします:

```
Unable to open a connection to the Xen hypervisor or daemon
```

これは、virt-manager アプリケーションが起動に失敗した時に発生します。このエラーは**/etc/hosts** 設定ファイル内にローカルホストのエントリがない場合に起こります。そのファイルをチェックして、ローカルホストが有効であるかどうか確認します。以下に間違えたローカルホストのエントリ例を示します:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
localhost.localdomain localhost
```

以下に正しいローカルホストエントリの例を示します:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1 localhost.localdomain localhost
```

```
localhost.localdomain. localhost
```

以下のエラーが表示されます (**xen-xend.logfile** 内で):

```
Bridge xenbr1 does not exist!
```

これは、ゲストのブリッジが間違えた設定を持っている時に発生し、これが **Xen hotplug** スクリプトを強制的にタイムアウトにします。設定ファイルをホスト間で移動する場合は、ゲストの設定ファイルを更新してネットワークのトポロジーと設定修正を反映するようにしなければなりません。間違えた、又は存在しない **Xen** ブリッジの設定を持つゲストを開始しようとすると、以下のようなエラーが出ます:

```
# xm create mySQL01

Using config file " mySQL01"
Going to boot Red Hat Enterprise Linux Server (2.6.18.-1.2747 .el5xen)
kernel: /vmlinuz-2.6.18-12747.el5xen
initrd: /initrd-2.6.18-1.2747.el5xen.img
Error: Device 0 (vif) could not be connected. Hotplug scripts not working.
```

更には、**xend.log** は次のエラーを表示します:

```
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:143) Waiting for
devices vif
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:149) Waiting for 0
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:464)
hotplugStatusCallback

/local/domain/0/backend/vif/2/0/hotplug-status

[2006-11-14 15:08:09 xend.XendDomainInfo 3875] DEBUG (XendDomainInfo:1449)
XendDomainInfo.destroy: domid=2
[2006-11-14 15:08:09 xend.XendDomainInfo 3875] DEBUG (XendDomainInfo:1457)
XendDomainInfo.destroyDomain(2)
[2006-11-14 15:07:08 xend 3875] DEBUG (DevController:464)
hotplugStatusCallback

/local/domain/0/backend/vif/2/0/hotplug-status
```

この問題を解決するには、**/etc/xen** ディレクトリにある ゲスト設定ファイルを開きます。例えば、ゲスト ***mySQL01*** を 編集する場合、

```
# vim /etc/xen/mySQL01
```

vif エントリを見つけます。**xenbr0** を デフォルトブリッジとして使用していると想定すると、適切なエントリは以下に似ているはずです:

```
# vif = ['mac=00:16:3e:49:1d:11, bridge=xenbr0',]
```

次のような **python depreciation** エラーを受け取ります:

```
# xm shutdown win2k3xen12
# xm create win2k3xen12

Using config file "win2k3xen12".
```

```
/usr/lib64/python2.4/site-packages/xenxm/opts.py:520: Deprecation Warning:
Non ASCII character '\xc0' in file win2k3xen12 on line 1, but no encoding
declared; see http://www.python.org/peps/pep-0263.html for details

execfile (defconfig, globs, locs,)
Error: invalid syntax 9win2k3xen12, line1)
```

Python は、無効な(又は間違えた) 設定ファイルがある場合にこれらのメッセージを生成します。この問題を解決するには、間違えた設定ファイルを修正するか、又は新しいものを生成します。

30.15. ログディレクトリのレイアウト

Red Hat Enterprise Linux 5 Virtualization 環境内の基本的なディレクトリ構成は 以下のようになります:

/etc/xen/ ディレクトリは以下を含みます

- **xend** デーモンで使用される 設定ファイル
- Virtualization ネットワーキング用のスクリプトを含んでいる **scripts** ディレクトリ

/var/log/xen/

- 全ての Xen 関連のログファイルを保有しているディレクトリ

/var/lib/libvirt/images/

- 仮想マシンイメージファイル用のデフォルトディレクトリ
- 仮想マシンイメージ用に異なるディレクトリを使用している場合は、そのディレクトリを確実に SELinux ポリシーに追加して、インストールを開始する前にそれを再ラベルします。

/proc/xen/

- /proc ファイルシステム内の xen 関連の情報

第31章 トラブルシューティング

この章では、Red Hat Enterprise Linux 仮想化での一般的な問題とその解決策を説明しています。

31.1. 利用可能なストレージとパーティションを判別する

ブロックドライバーがロードされたことと、デバイスとパーティションがゲストに 利用可能であることを確証します。これは、以下に示すように **"cat /proc/partitions"** を実行することで達成できます。

```
# cat /proc/partitions
major minor  #blocks   name
 202    16  104857600  xvdb
   3     0    8175688  hda
```

31.2. XEN ベースのゲストを再起動した後にコンソールはフリーズ

ゲストが起動する時に、時々 Xen ゲストのコンソールがフリーズすることがあります。 その場合、コンソールはメッセージを表示しますが、ログインは出来ません。

この問題を修復するには、以下の行を **/etc/inittab** ファイルに 追加します:

```
1:12345:respawn:/sbin/mingetty xvc0
```

ファイルを保存したら再起動します。コンソールセッションはこれで期待どおりに 対話型になるはずです。

31.3. 仮想化イーサネットデバイスはネットワーキングツールでは見付からない。

ネットワーキングツールは、ゲストオペレーティングシステム内の **Xen Virtual Ethernet** ネットワーキングを識別できません。これは以下を実行することにより 確証できます（Red Hat Enterprise Linux 4 及び Red Hat Enterprise Linux 5 用）:

```
cat /etc/modprobe.conf
```

又は （Red Hat Enterprise Linux 3 用）:

```
cat /etc/modules.conf
```

その出力は、各追加のインターフェイス用に次の行とそれに似た行を含んでいる はずです。

```
alias eth0 xen-vnif
```

この問題を修復するには、エイリアス化の行（例えば、**alias eth0 xen-vnif**）を ゲストの各 para-virtualized インターフェイスに追加する必要があります。

31.4. ループデバイスエラー

ファイルベースのゲストイメージを使用する場合、設定済みのループデバイスの数量を増加する必要があるかも知れません。デフォルト設定では、8つのループデバイス までがアクティブになる許可があり

ます。もし、8つ以上のファイルベースのゲスト、又はループデバイスを必要とするならば、設定済みのループデバイスの数量は **/etc/modprobe.conf** で調節できます。**/etc/modprobe.conf** を編集して、以下の行を追加します:

```
options loop max_loop=64
```

この例では 64 を使用していますが、他の数字を指定して最大ループ値を設定することができます。また、システム上でループデバイスで支持されているゲストも実装する必要があるかも知れません。para-virtualized ゲスト用にループデバイス支持のゲストを適用するには、**phy: block device** か **tap:aio** コマンドを使用します。完全仮想化システム用にループデバイス支持のゲストを使用するには、**phy: device** か **file: file** コマンドを使用します。

31.5. メモリー不足によるドメイン作成の失敗

これがドメインの開始失敗の原因になる可能性があります。その理由は、利用できる メモリーが十分でないか、又は直近に作成された、あるいは、開始したゲスト用に **dom0** がスペースを提供できるように縮小していないことが 考えられます。**/var/log/xen/xend.log** を見ると、サンプルの エラーメッセージがこの発生を示しています:

```
[2006-11-21 20:33:31 xend 3198] DEBUG (balloon:133) Balloon: 558432 KiB
free;
        0 to scrub; need 1048576; retries: 20.
[2006-11-21 20:33:52 xend.XendDomainInfo 3198] ERROR (XendDomainInfo:202)
Domain construction failed
```

"xm list Domain-0" を実行すると、現在 **dom0** で 使用されているメモリーの量が確認できます。**dom0** が縮小されていない場合は、**"xm mem-set Domain-0 NewMemSize"** を使用して小サイズをセットできます。 ここで、**NewMemSize** はより小さい値にします。

31.6. 間違えたカーネルイメージのエラー

Para-virtualized ゲストは kernel-xen カーネルを使用できません。para-virtualized ゲストには、標準のカーネルのみを使用してください。

Xen カーネル以外のカーネルを para-virtualized ゲストとしてブートを試みると、 以下のようなエラーメッセージが表示されます:

```
# xm create testVM
Using config file "./testVM".
Going to boot Red Hat Enterprise Linux Server (2.6.18-1.2839.el5)
kernel: /vmlinuz-2.6.18-1.2839.el5
initrd: /initrd-2.6.18-1.2839.el5.img
Error: (22, 'Invalid argument')
```

このエラー内で、非 kernel-xen カーネルをブートしようとしていることをカーネル行が示すのが判ります。この例での正しいエントリは、**"kernel: /vmlinuz-2.6.18-1.2839.el5xen"** です。

解決策として、使用するゲストに本当に kernel-xen をインストールしたことを 確認して、それがブートするために **/etc/grub.conf** 設定ファイル内で デフォルトカーネルになっていることを確認することです。

使用するゲストに **kernel-xen** をインストールしている場合は、 以下のようにしてゲストを開始できます:

```
xm create -c GuestName
```

ここで、**GuestName** は、ゲストの名前です。先の コマンドが**GRUB** ブートローダ画面を提示して、ブートするカーネルを選択できるようにします。ブート用に kernel-xen カーネルを 選択する必要があります。ゲストがブートプロセスを完了すると、ゲストにログインして、 **/etc/grub.conf** の編集により、デフォルトのブートカーネルを kernel-xen に変更することができます。単に **“default=X”**（**X** は **0** で始まる番号）の行を kernel-xen の行が持つエントリに 変更するだけです。番号付けは**0** から開始するため、kernel-xen エントリが ２つめのエントリの場合は、デフォルトとして、**1** を入力することになります。 例えば、**“default=1”** となります。

31.7. 間違えたカーネルイメージのエラー：PAE プラットフォーム上に非 PAE カーネル

非 PAE の para-virtualized ゲストをブートすると、以下のようなエラーメッセージが 表示されます。メッセージは 「Hypervisor 上でゲストカーネルを実行しようとしています。これは現在サポートされていません」と伝えています。Xen hypervisor は 現在、PAE と 64 bit の para-virtualized ゲストカーネルのみをサポートしています。

```
# xm create -c va-base
Using config file "va-base".
Error: (22, 'Invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] ERROR (XendDomainInfo:202)
Domain construction failed
Traceback (most recent call last):
File "/usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py",
        line 195, in  create vm.initDomain()
File "/usr/lib/python2.4/site-packages/xen/xend/XendDomainInfo.py",
        line 1363, in initDomain raise VmError(str(exn))
VmError: (22, 'Invalid argument')
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XendDomainInfo:1449)
XendDomainInfo.destroy: domid=1
[2006-12-14 14:55:46 xend.XendDomainInfo 3874] DEBUG (XendDomainInfo:1457)
XendDomainInfo.destroyDomain(1)
```

32 bit 又は、非 PAE のカーネルを実行する必要がある場合は、使用するゲストを 完全仮想化の仮想マシンとして実行する必要があります。hypervisor の互換性ルールは以下のように なります:

- para-virtualized のゲストは、使用する hypervisor のアーキテクチャタイプに一致 しなければなりません。32 bit PAE のゲストを実行するには、32 bit PAE hypervisor が 必要になります。
- 64 bit para-virtualized のゲストを実行するには、Hypervisor も 64 bit の バージョンでなければなりません。
- 完全仮想化のゲストでは、32 bit ゲスト用に hypervisor は 32 bit か 64 bit の どちらかになります。32 bit 又は 64 bit hypervisor 上で、32 bit (PAE と 非-PAE) の ゲストを実行できます。
- 64 bit の完全仮想化のゲストを実行するには、使用する hypervisor も 64 bit で ある必要があります。

31.8. 完全仮想化 64 BIT ゲストが起動失敗

設定ファイルを Red Hat Enterprise Linux 5 に移動した為に、完全仮想化ゲストが 起動に失敗して、**“Your CPU does not support long mode. Use a 32 bit distribution”** のエラー表示を出した場合、問題は **pae** 設定の欠如、又はその間違いです。使用ゲストの設定ファイル内に **‘pae=1”** の

エントリがあることを確認して下さい。

31.9. ローカルホストエントリの欠如が VIRT-MANAGER の失敗原因

virt-manager アプリケーションが起動を失敗して、**“Unable to open a connection to the Xen hypervisor/daemon”** と言うようなエラーメッセージを 表示する可能性があります。これは、通常 **/etc/hosts** ファイル内に **localhost** エントリが欠如していることが 原因です。実際に **localhost** エントリが あることを確認して、それが**/etc/hosts** 内に存在しない場合は、 新規のエントリを挿入します。不正な **/etc/hosts** の例として以下のような 状態があります:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
localhost.localdomain localhost
```

正しいエントリは以下に似たものになるはずです:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1               localhost.localdomain localhost
localhost.localdomain localhost
```

31.10. ゲスト起動時の MICROCODE エラー

仮想マシンの起動段階で、以下に似たエラーメッセージが表示される可能性があります:

```
Applying Intel CPU microcode update: FATAL: Module microcode not found.
ERROR: Module microcode does not exist in /proc/modules
```

仮想マシンは、仮想 CPU 上で実行しているため、microcode を 更新する意味はありません。仮想マシンの microcode 更新を無効にするとこの エラーを停止できます:

```
/sbin/service microcode_ctl stop
/sbin/chkconfig --del microcode_ctl
```

31.11. 仮想マシン開始時での PYTHON 低評価警告メッセージ

Python は、時に以下のようなメッセージを生成します。これらのメッセージは無効な，又は不正な 設定ファイルが原因で起こります。非 ascii 文字を含んだ設定ファイルもこれらのエラーになります。 解決策としては、間違えた設定ファイルを修正するか、又は新しいものを生成します。

もう１つの原因は、現在の作業ディレクトリ内の間違えた設定ファイルです。 **“xm create”** は 作業ディレクトリ内で設定ファイルを 見て、それから **/etc/xen** で見ます。

```
# xm shutdown win2k3xen12
# xm create win2k3xen12
Using config file "win2k3xen12".
/usr/lib64/python2.4/site-packages/xen/xm/opts.py:520: DeprecationWarning:
Non-ASCII character '\xc0' in file win2k3xen12 on line 1, but no encoding
declared; see http://www.python.org/peps/pep-0263.html for details
execfile(defconfig, globs, locs)
Error: invalid syntax (win2k3xen12, line 1)
```

31.12. BIOS 内で、INTEL VT と AMD-V の仮想化ハードウェア拡張を有効にする

このセクションでは、ハードウェア仮想化拡張を識別して、無効になっている場合は それらを BIOS 内で有効にする方法を説明しています。

Intel VT 拡張は、BIOS 内で無効にできます。特定のラップトップ製造元は その CPU 内で、デフォルトとして Intel VT 拡張を無効にしています。

仮想化拡張は、AMD-V 用には BIOS 内で無効に出来ません。

仮想化拡張は時々、BIOS 内で無効にされていることがあります。通常、ラップトップの 製造元でそうします。無効になっている仮想化拡張を有効にする案内には、「BIOS 内で、Intel VT と AMD-V の仮想化ハードウェア拡張を有効にする」 を参照して下さい。

仮想化拡張が BIOS 内で有効になっていることを確認します。Intel® VT 又は AMD-V 用の BIOS 設定は、通常 **チップセット** 又は **プロセッサ** メニュー内にあります。メニューの名前は、このガイドとは異なるかも知れませんが、 仮想化拡張の設定は **セキュリティ設定** か、他の非標準的なメニュー名内にあるでしょう。

手順31.1 BIOS 内で仮想化拡張を有効にする

1. コンピュータを再起動して、システムの BIOS メニューを開きます。これは通常 **delete** 又は **Alt** と **F4** を押すことで達成されます。
2. **Restore Defaults** を選択して、それから **Save & Exit** を選びます。
3. マシンの電源を切って電源供給を止めます。
4. マシンの電源をいれて、**BIOS Setup Utility** を開きます。 **Processor** セクションを開いて、**Intel®Virtualization Technology**、あるいは、**AMD-V** を有効にします。一部のマシンでは **Virtualization Extensions** と 言う名前になっているかもしれません。**Save & Exit** を選択します。
5. マシンの電源を切って電源供給を止めます。
6. **cat /proc/cpuinfo | grep vmx svm** を実行します。 このコマンドが出力を出せば、仮想化拡張はこれで有効になっています。出力が ない場合は、そのシステムに仮想化拡張が存在しない、又は正しい BIOS 設定が 有効でないことになります。

31.13. KVM ネットワーキングパフォーマンス

デフォルトで、KVM 仮想マシンは仮想 Realtek 8139 (rtl8139) NIC (network interface controller) に 割り当てられています。

rtl8139 仮想化 NIC は ほとんどの環境で正常に機能します。しかし、このデバイスは 一部のネットワーク上（例えば、10 Gigabit Ethernet ネットワーク）でパフォーマンス 劣化問題を被る可能性があります。

回避策としては、仮想化 NIC の異なるタイプに切り替えることです。例えば、 Intel PRO/1000 (e1000) や、あるいは、virtio（para-virtualized ネットワークドライバー）。

e1000 ドライバーに切り替えるには:

1. ゲストオペレーティングシステムをシャットダウンします。

2. ゲストの設定ファイルを **virsh** コマンドで編集します。（ここで、***GUEST*** とはゲスト名です）:

```
# virsh edit GUEST
```

virsh edit コマンドは **$EDITOR** シェル変数を使用して使用するエディタを決定します。

3. 設定内でネットワークインターフェイスセクションを見つけます。このセクションは、 以下の snippet に似ています:

```
<interface type='network'>
  [output truncated]
  <model type='rtl8139' />
</interface>
```

4. モデルエレメントのタイプ属性を ***'tl8139'*** から ***'e1000'*** へ変更します。これが rtl8139 ドライバーから e1000 ドライバーへとドライバーを変換します。

```
<interface type='network'>
  [output truncated]
  <model type='e1000' />
</interface>
```

5. 変更を保存してテキストエディタを終了します。

6. ゲストオペレーティングシステムを再スタートします。

別の方法では、異なるネットワークドライバーを使用して、新しい仮想化ゲストを インストールする方法があります。ネットワーク接続でゲストのインストールが 難しい場合に、この方法が必要になるかも知れません。この方法では、ユーザーが 既に作成済の仮想化マシンを少なくとも１つ持っていて（多分、CD か DVD でインストール済み） それをテンプレートとして使用することが要求されます。

1. 既存の仮想マシンから XML テンプレートを作成します:

```
# virsh dumpxml GUEST > /tmp/guest.xml
```

2. XML ファイルをコピーして編集により、特有のフィールドを更新します: 仮想マシン名、UUID、ディスクイメージ、MAC アドレス、他の特有なパラメータ。UUID と MAC アドレスの行を削除すると、virsh が UUID と MAC アドレスを生成します。

```
# cp /tmp/guest.xml /tmp/new-guest.xml
# vi /tmp/new-guest.xml
```

ネットワークインターフェイスセクションにモデルの行を追加します:

```
<interface type='network'>
  [output truncated]
  <model type='e1000' />
</interface>
```

3. 新しい仮想マシンを作成します:

```
# virsh define /tmp/new-guest.xml
# virsh start new-guest
```

e1000 or virtio ドライバーを使用した方がネットワークパフォーマンスは良くなります。 (BZ#517181)

第32章 XEN PARA-VIRTUALIZED ドライバーのトラブルシューティング

この章では、para-virtualized ドライバーを使用する Xen ホストと完全仮想化の Red Hat Enterprise Linux ゲストで遭遇する可能性のある問題を取り扱います。

32.1. RED HAT ENTERPRISE LINUX 5 仮想化のログファイルとディレクトリ

Red Hat Enterprise Linux 5 仮想化関連のログファイル

Red Hat Enterprise Linux 5 では、**xend** デーモンと **qemu-dm** プロセスで書き込まれる ログファイルは全て以下のディレクトリに保存されます:

/var/log/xen/

xend デーモンと qemu-dm プロセスで生成される 全てのログファイルを保持するディレクトリです。

xend.log

- このログファイルは、通常のシステムイベントか、オペレータ由来のイベントで 生成された全てのイベントをログするために xend で使用されます。
- create、shutdown、又は destroy などの仮想マシン操作は全てこの ログファイルにログされます。
- 通常、このログファイルは問題が発生した時に最初に見るべき場所です。多くの 場合、ログファイルを眺めていき、実際のエラーメッセージの直前にログされた エントリを再確認することにより根本にある原因を識別することができます。

xend-debug.log

- xend とそのサブシステム（フレームバッファと Python スクリプト）からの エラーイベントを記録します

xen-hotplug.log

- hotplug イベントからのイベントをログします。
- オンラインに来ないデバイスや、オンライン上にないネットワークブリッジからの イベント通知はこのファイルにログされます。

qemu-dm.*PID*.log

- このファイルは、各完全仮想化のゲストの為に開始した **qemu-dm** プロセスによって作成されます。
- **PID** は、関連した qemu-dm プロセスの プロセス **PID** で入れ替えるべきものです。
- 任意の **qemu-dm** プロセスの **PID** を 取り込むには、**ps** コマンドを使用して、**qemu-dm** が所属する仮想マシンを識別できるプロセスの引数を見ることで達成できます。

virt-manager アプリケーションでの問題をトラブルシューティング している場合、それにより生成さ

れたログファイルを確認することも良い方法です。**virt-manager** 用のログファイルは 、ユーザーの **home** ディレクトリ内の **.virt-manager** と 言うディレクトリにあります。このディレクトリは通常、**~/.virt-manager/virt-manager** となります。

注記

このログファイルは **virt-manager** を開始する度に 上書きされます。**virt-manager** での問題をトラブルシューティング している場合は、エラーが発生した後で、**virt-manager** を再起動 する前にログファイルを保存することを忘れないで下さい。

Red Hat Enterprise Linux 5 仮想化関連のディレクトリ

Red Hat Enterprise Linux 5 の仮想化環境でトラブルシューティングを実行している状況での 興味深いディレクトリとファイルが他にいくつかあります:

/var/lib/libvirt/images/

ファイルベースゲストイメージ用の標準ディレクトリ

/var/lib/xen/xend-db/

デーモンが再開始される度に生成される xend データベースを保持するディレクトリです。

/etc/xen/

Xen hypervisor 用に多くの設定ファイルを格納します。

- **/etc/xen/xend-config.sxp** は xend デーモン用の主要設定ファイルです。 **xend-config.sxp** ファイルは、**libvirt** で設定されていない他の機能と移行を有効/無効にすることができます。 それ以外の全ての機能には **libvirt** ツールを使用して下さい。

/var/xen/dump/

仮想マシンで生成されたり、**xm dump-core** コマンドを 使用した時に生成されるダンプを保管します。

/proc/xen/

以下のファイル内に **xen-kernel** 情報を格納します:

- **/proc/xen/capabilities**
- **/proc/xen/privcmd**
- **/proc/xen/balloon**
- **/proc/xen/xenbus**
- **/proc/xen/xsd_port**
- **/proc/xen/xsd_kva**

32.2. PARA-VIRTUALIZED ゲストは、RED HAT ENTERPRISE LINUX 3 のゲストオペレーティング システム上ではロードに失敗。

Red Hat Enterprise Linux 3 はプロセッサアーキテクチャ特有のカーネル RPM を 使用するため 、para-virtualized ドライバー RPM がインストールされているカーネル アーキテクチャに適合しない場合、para-virtualized ドライバーはロードに 失敗する可能性があります。

para-virtualized ドライバーが挿入されると、未解決モジュールの長い リストが表示されます。エラーの短縮した要約は以下のようになります。

```
# insmod xen-platform-pci.o
Warning: kernel-module version mismatch
xen-platform-pci.o was compiled for kernel version 2.4.21-52.EL
while this kernel is version 2.4.21-50.EL
xen-platform-pci.o: unresolved symbol __ioremap_R9eac042a
xen-platform-pci.o: unresolved symbol flush_signals_R50973be2
xen-platform-pci.o: unresolved symbol pci_read_config_byte_R0e425a9e
xen-platform-pci.o: unresolved symbol __get_free_pages_R9016dd82
[...]
```

解決策としては、 para-virtualized ドライバーを使用するハードウェアの為に 正しい RPM パッケージを使用することです。

32.3. RED HAT ENTERPRISE LINUX 3 に PARA-VIRTUALIZED ドライバーをインストールする時点に 警告メッセージ。

Red Hat Enterprise Linux 3 に para-virtualized ドライバーをインストールしている時点に、 2.4.21-52 より以前のカーネルでは、「実行中のカーネルよりも新しいバージョンで モジュールがコンパイルされました。」と言う警告メッセージが表示されるようになるでしょう。

以下に示してあるような、このメッセージは無視しても安全です。

```
Warning: kernel-module version mismatch
xen-platform-pci.o was compiled for kernel version 2.4.21-52.EL
while this kernel is version 2.4.21-50.EL
Warning: loading xen-platform-pci.o will taint the kernel: forced load
See http://www.tux.org/lkml/#export-tainted for information about tainted
modules
Module xen-platform-pci loaded, with warnings
```

上記メッセージの重要な部分は最後の行であり、これはモジュールが警告付きで ロードされたことを述べています。

32.4. PARA-VIRTUALIZED ドライバーを手動でロードする

なんらかの理由で、ブートプロセス中に para-virtualized ドライバーが自動的に ロードを失敗する場合は、手動でそれをロードする試みができます。

これにより、ユーザーはネットワークやストレージエンティティを再設定したり、 あるいは、最初にロード失敗した理由を識別することができるようになります。 以下の手順で、para-virtualized ドライバーモジュールをロードできるはずです。

先ず、システム上で para-virtualized ドライバーモジュールの位置を判定します。

```
# cd /lib/modules/`uname -r`/
# find . -name 'xen-*.ko' -print
```

その場所の記録を取り、モジュールを手動でロードします。**{LocationofPV-drivers}** の 部分を上記コマンドの出力から得た正しい場所で入れ替えます。

```
# insmod \
/lib/modules/'uname -r'/{LocationofPV-drivers}/xen-platform-pci.ko
# insmod /lib/modules/'uname -r'/{LocationofPV-drivers}/xen-balloon.ko
# insmod /lib/modules/'uname -r'/{LocationofPV-drivers}/xen-vnif.ko
# insmod /lib/modules/'uname -r'/{LocationofPV-drivers}/xen-vbd.ko
```

32.5. PARA-VIRTUALIZED ドライバーが正常にロードされたことを確認

実行すべき最初のタスクの１つは、ドライバーが実際にシステム内にロード されたことを確認することです。

para-virtualized ドライバーがインストールされて、ゲストが再起動した後に、 ドライバーがロードされたことを確認できます。先ず、ドライバーがそのローディングを **/var/log/messages** にログしていることを確認します。

```
# grep -E "vif|vbd|xen" /var/log/messages
                    xen_mem: Initialising balloon driver
                    vif vif-0: 2 parsing device/vif/0/mac
                    vbd vbd-768: 19 xlvbd_add at
/local/domain/0/backend/vbd/21/76
                    vbd vbd-768: 19 xlvbd_add at
/local/domain/0/backend/vbd/21/76
                    xen-vbd: registered block device major 202
```

又は、**lsmod** コマンドを使用してロードされた para-virtualized ドライバーを一覧表示することもできます。それが **xen_vnif**、 **xen_vbd**、**xen_platform_pci**、及び **xen_balloon** のモジュールを 含む一覧を出力するはずです。

```
# lsmod|grep xen
xen_vbd                19168  1
xen_vnif               28416  0
xen_balloon            15256  1 xen_vnif
xen_platform_pci       98520  3 xen_vbd,xen_vnif,xen_balloon,[permanent]
```

32.6. システムは、PARA-VIRTUALIZED ドライバーではスループットに制限があります

para-virtualized ドライバーのインストール後でもまだネットワークのスループットが 制限されていて、それらが正しくロードされていることを確認した場合（「para-virtualized ドライバーが正常にロードされたことを確認」 を参照）、この問題を 解決するには、ゲストの設定ファイル内の**vif=** 行の **type=ioemu** の 部分を削除します。

付録A XEN システムアーキテクチャ

仮想化を持つ稼働可能な Red Hat Enterprise Linux システムはマルチレイヤーであり、 特別権限を有する Xen カーネルコンポーネントにより駆動されます。Xen は複数のゲスト オペレーティングシステムを維持することができます。各ゲストオペレーティングシステムは それ自身のドメイン内で稼働します。Xen は仮想マシン内の仮想 CPU 群をスケジュールして 利用可能な物理 CPU 群の最善の使用法を選択します。各ゲストオペレーティングシステムは それぞれ自身のアプリケーションを処理します。これらのゲストオペレーティングシステムは 各自のアプリケーションを適切にスケジュールできます。

Xen は ２つの選択肢の内の１つを導入できます：*完全仮想化* かあるいは、 *para-virtualization* です。完全仮想化は、背後にある物理システムの 全面的な抽出を提供して、ゲストオペレーティングシステムが稼働できるための 新規の仮想システムを作り上げます。この場合、ゲスト OS や アプリケーションには 修正は不要です（ゲスト OS と アプリケーションは仮想化環境の認識がなく、普通に稼働）。Para-virtualization は、仮想マシン上で稼働するゲストオペレーティングシステムにユーザーの 修正を必要とし（ゲストオペレーティングシステムは仮想マシン上で稼働していることを認識）、 ネイティブに近いパフォーマンスを提供します。Para-virtualization と 完全仮想化のいずれも ご使用の仮想化インフラストラクチャ全域に渡って導入することができます。

domain0 (dom0) と呼ばれる一番目のドメインは、システムがブートする時点で自動的に作成されます。Domain0 は特権ゲストであり、新規のドメイン作成や仮想デバイス管理などができる管理能力を所有します。Domain0 はネットワークカードやハードディスクコントローラなどの物理ハードウェアを処理します。Domain0 はまた、他の仮想マシンに対してゲストの保留、復元、あるいは移行などの管理タスクも処理します。

hypervisor (Red Hat の仮想マシンモニター) は完全仮想化環境内に於て単独ホスト上で複数のオペレーティングシステムを同時に実行するようにできる仮想化プラットフォームです。ゲストとは、ホスト、すなわち主体 OS の他に、仮想マシン上で実行するオペレーティングシステムのことです。

Xen があると、各ゲストの **メモリー**はホストの 物理メモリーの一部から流用されます。para-virtualized ゲストには、仮想マシンの 初期メモリーと最大サイズの両方をセットできます。ユーザーが指定する最大サイズを 超過しないでランタイムに仮想マシンへの物理メモリーを追加（又は削除）することが 可能です。このプロセスは「バルーン化」と呼ばれます。

各ゲストを多数の仮想 **cpus** (VCPU と呼ばれる)で設定することができます。物理 CPU 上の作業負荷に応じて、VCPU がスケジュールされます。

ユーザーはゲストに対して、任意の数の **virtual disks (仮想ディスク)** を認可することができます。ゲストはこれらの仮想ディスクをハードディスク、又は CD-ROM デバイス (完全仮想化ゲスト用) と見なします。各仮想ディスクはブロックデバイスから、あるいはホスト上の通常ファイルからゲストへのサービス提供をします。ホスト上のデバイスにはゲスト用の完全ディスクイメージが含まれており、通常はパーティションテーブル、複数パーティション、及び殆んどの場合、LVM 物理ボリュームを含んでいます。

仮想ネットワーキングインターフェイス はゲスト上で 稼働します。他のインターフェイスにも、仮想イーサネットインターネットカード (VNIC) のようにゲスト上で稼働できるものがあります。これらのインターフェイスは永続化した仮想 メディアアクセスコントロール (MAC) アドレスで設定されます。新規ゲストのデフォルト インストールは、1600万以上のアドレスの保存プールからランダムに選択された MAC アドレスを 持つ VNIC をインストールします。そのため、２つのゲストが同じ MAC アドレスを受け取ることは ほとんどあり得ません。多数のゲストを持つ複雑なサイトは手動で MAC アドレスを割り当てて ネットワーク上で特有であることを確実にできます。

各ゲストは、ホストに接続する仮想 **テキストコンソール**を持っています。ゲストログインとコンソール出力をテキストコンソールに転送することができます。

設定すればどのゲストでも、物理ホスト上の通常のビデオコンソールに相当する仮想 **graphical console (グラフィカルコンソール)** を使用できるようになります。これは 完全仮想化ゲストでも paravirtualization ゲストででも実行できます。この機能はブートメッセージ、グラフィカルブーティ

ング、複数仮想ターミナルのような標準グラフィックアダプタの特長を活用し、X window システムを起動できます。ユーザーはまた、グラフィカルキーボードを使用して、仮想のキーボードとマウスを設定することもできます。

ゲストは、次の三つの **identities (識別子)** のいずれかで識別できます: ドメイン名 **(`domain-name`)**、識別子番号 **(`domain-id`)**、または UUID。**`domain-name`** はゲスト設定ファイルに適用するテキストの文字列です。このドメイン名はゲストの起動に使用され、ゲストの稼動時には同じ名前が、識別と制御のために使用されます。domain-id はゲストを識別し制御できるアクティブなドメインに割り当てられる特有であっても永続性のない番号です。UUID はゲストの設定ファイルから制御される 永続性の特有の識別子であり、これがシステム管理ツールによって常時ゲストの 識別を確実にするものです。これは実行時にゲストから認識できます。新規の UUID は、 ゲストが最初にインストールされる時にシステムツールによって各ゲストに自動的に割り当てられます。

付録B その他のリソース

仮想化と Red Hat Enterprise Linux についての詳細を見るには、以下のリソースを 参照して下さい。

B.1. オンラインリソース

- http://www.cl.cam.ac.uk/research/srg/netos/xen/ Red Hat kernel-xen パッケージの抽出元である Xen™ para-virtualization マシンマネージャの プロジェクトウェブサイトです。このサイトはアップストリームの xen プロジェクトバイナリとソース コードを維持し、また xen とその関連技術に関する情報、アーキテクチャ概要、ドキュメント、及び 関連リンクも含んでいます。
- Xen コミュニティウェブサイト

  http://www.xen.org/
- http://www.libvirt.org/ は **libvirt** 仮想化 API の為の正式ウェブサイトです。
- http://virt-manager.et.redhat.com/ 仮想マシンの管理用のグラフィカルアプリケーションである **仮想マシンマネージャ(Virtual Machine Manager)** (virt-manager)のプロジェクト WEB サイトです。
- オープン仮想化センター

  http://www.openvirtualization.com
- Red Hat ドキュメント

  http://www.redhat.com/docs/
- 仮想化技術概要

  http://virt.kernelnewbies.org
- Red Hat 発展技術グループ

  http://et.redhat.com

B.2. インストール済みドキュメント

- **/usr/share/doc/xen-<*version-number*>/** このディレクトリには、Xen para-virtualization hypervisor と設定の参考例、ハードウェア特有の情報、現在の Xen のアップストリームユーザードキュメントなどを含む、関連した管理ツールに関する豊富な情報が含まれています。
- **man virsh** と **/usr/share/doc/libvirt-<*version-number*>** – には、**virsh** 仮想マシン管理ユーティリティ用のサブコマンドと オプションと、更には **libvirt** 仮想化ライブラリ API に関する 総括的な情報が含まれています。
- **/usr/share/doc/gnome-applet-vm-<*version-number*>** – ローカルで実行している仮想マシンを監視して管理するGNOME グラフィカルパネルアプレット用のドキュメントがあります。
- **/usr/share/doc/libvirt-python-<*version-number*>** – **libvirt** ライブラリ用の Python バインディングについての詳細を提供します。**libvirt-python** パッケージによ

り、Python の開発者は **libvirt** virtualization 管理ライブラリを持つインターフェイスのプログラムを作成することが出来るようになります。

- **/usr/share/doc/python-virtinst-*<version-number>*** – **virt-install** コマンドに関するドキュメントを提供します。このコマンドは仮想マシン内の Fedora と Red Hat Enterprise Linux 関連のディストリビューションのインストール開始を援助するものです。
- **/usr/share/doc/virt-manager-*<version-number>*** – 仮想マシンマネージャに関するドキュメントを提供します。仮想マシンマネージャは仮想マシン管理用のグラフィカルツールです。

用語集

この語彙集はインストールガイドで使用されている用語を定義する為の ものです。

dom0

ホスト、又はホストオペレーティングシステムとしても 知られています。

dom0 とは、Hypervisor を 稼働している Red Hat Enterprise Linux のホストインスタンスを指します。Virtualization-Hypervisor は ゲストオペレーティングシステムの仮想化を促進します。Dom0 は物理ハードウェア上で稼働して、そのハードウェアを管理して、それ自身とゲストオペレーティングシステムの為にリソースを割り当てます。

domU

domU とは、ホストシステム上で稼働するゲストオペレーティング システムを指します (ドメイン)。

Hypervisor

hypervisor とは、ハードウェアをオペレーティングシステムから抽出して、複数の オペレーティングシステムが同じハードウェア上で稼働できるようにするソフトウェア です。hypervisor はホストオペレーティングシステム上で稼働し、他の仮想化 オペレーティングシステム群がそのホストのハードウェア上で稼働できるように します。

I/O

入力/出力（input/output）の短縮形です（発音：アイオー）。I/O と言う用語は データをコンピュータと周辺機器との間で転送する全てのプログラム、操作、あるいは デバイスを示します。全ての転送は 1 つのデバイスからの出力であり、別のデバイスへの 入力となります。キーボードやマウスなどのデバイスは入力のみのデバイスであり、プリンター などのデバイスは出力のみとなります。書き込み可能な CD ROM は入力と出力両方のデバイス です。

Itanium®

Intel Itanium® プロセッサアーキテクチャ

Kernel SamePage Merging

Kernel SamePage Merging (KSM) モジュールは、KVM ゲストが同一のメモリーページ群を共有 できるようにする KVM hypervisor によって使用されます。共有されるページは通常、共通の ライブラリか、又は他の同一の使用頻度の高いデータです。KSM は各種のゲストと 増加続けるゲスト密集のためにキャッシュ内にこれらのライブラリを維持することにより、 特定のゲストのパフォーマンスを増強できます。

LUN

LUN（Logical Unit Number）は論理ユニット（SCSI プロトコルエンティティ）に 割り当てられた番号です。

MAC アドレス

MAC（Media Access Control）アドレスはネットワークインターフェイスコントローラの ハードウェアアドレスです。仮想化のコンテキストでは、MAC アドレスは、ローカルドメイン上の 各 MAC が特有である状態で仮想ネットワークインターフェイス用に生成されなければなりません。

Para-virtualization

Para-virtualization とは、時として xen カーネル、又は kernel-xen パッケージと呼ばれる特殊なカーネルを使用します。Para-virtualized ゲストカーネルは、ホストの ライブラリとデバイスを使用しながら、ホスト上で並行して実行されます。Para-virtualization の インストールは、セキュリティ設定（SELinux とファイル制御）で制限可能なシステム上で全ての デバイスへの完全なアクセスを持っています。完全仮想化よりも Para-virtualization の方が高速です。 Para-virtualization はロードバランシング、プロビジョニング、セキュリティ、及び統合性利便に 於いて効果的に使用できます。

Fedora 9 の時点で、特別なカーネルはもう必要なくなりました。このパッチが 基幹の Linux ツリーに受け入れられてから、このバージョン以降の全ての Linux カーネルは para-virtualization 認識ができて利用可能になっています。

Para-virtualized

Para-virtualization を参照

Para-virtualized のドライバー

Para-virtualized のドライバーとは、完全仮想化の Linux ゲスト上で 動作するデバイスドライバーです。これらのドライバーは完全仮想化の ゲスト用にネットワークとブロックデバイスのパフォーマンスを大幅に 向上します。

Security Enhanced Linux

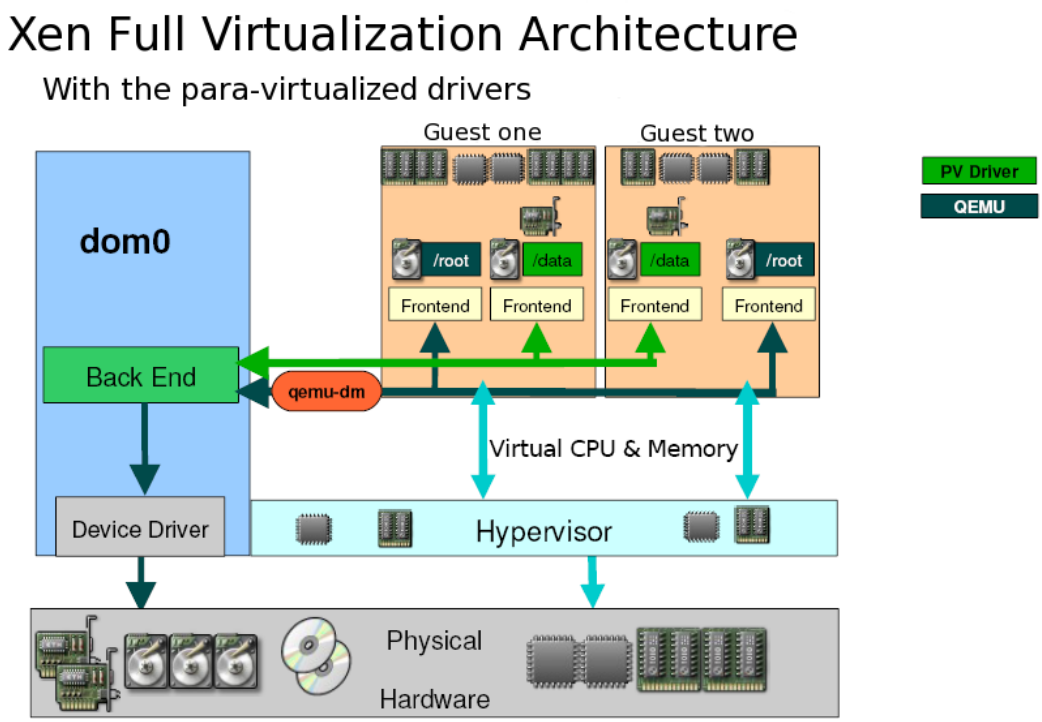
Security Enhanced Linux の省略名である、SELinux は LSM (Linux Security Modules) を Linux カーネル内で使用して、セキュリティポリシーに必要な各種の最低限特権を提供します。

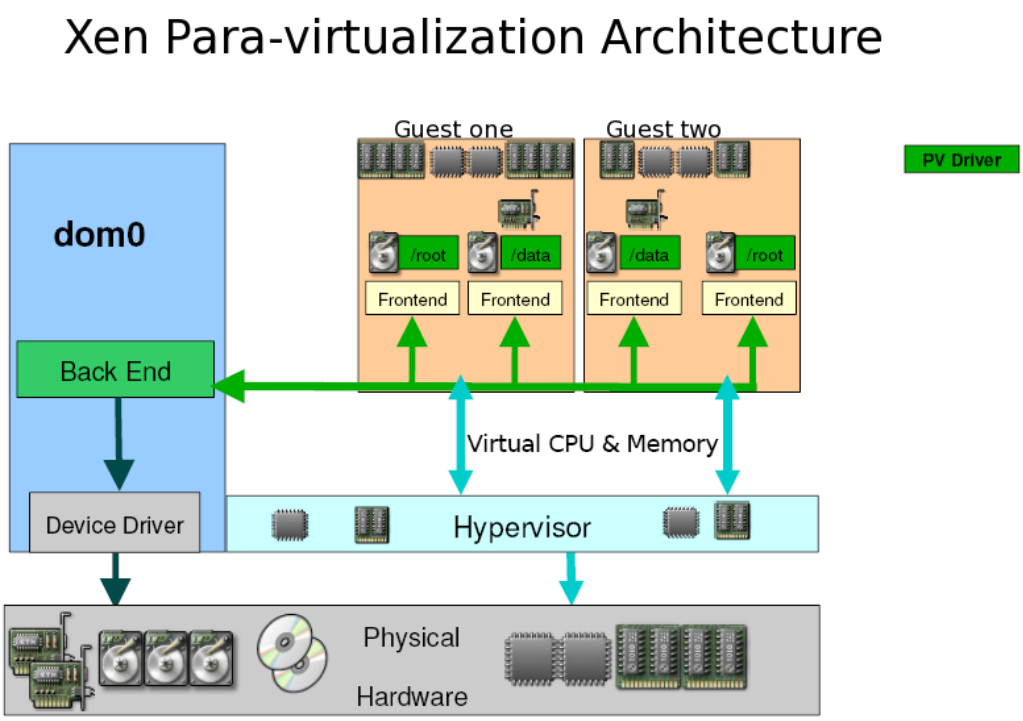
Universally Unique Identifier

UUID (Universally Unique Identifier) とは、分散型コンピューティング環境の中で デバイス、システム、及び特定のソフトウェアオブジェクトの為の標準化した番号付け 手段です。仮想化の中での UUID のタイプとしては、**ext2** と **ext3** のファイルシステムの識別子、 RAID デバイス識別子、iSCSI と LUN の識別子、MAC アドレスと仮想マシンの識別子が あります。

Xen

Red Hat Enterprise Linux は Xen hypervisor 及び KVM hypervisor をサポートします。 （カーネルベース仮想マシン を参照） この 2 つのハイパバイザーは異なるアーキテクチャと開発アプローチを持ちます。 Xen hypervisor は 主要ホスト管理システムリソース及び 仮想化 API として 機能する Red Hat Enterprise Linux オペレーティングシステム下で動作します。 ホストは時には、dom0 とか、Domain0 と 呼ばれます。





カーネルベース仮想マシン

KVM (Kernel-based Virtual Machine カーネルベースの仮想マシン) とは、AMD64 と Intel 64 のハードウェア上の Linux 用 完全仮想化 ソリューションです。VM は 標準の Red Hat Enterprise Linux カーネル用に構築された Linux カーネルモジュールです。KVM は 仮想化ゲストとしての Windows と Linux のオペレーティングシステムを無修正で複数実行できます。KVM は hypervisor であり、これは libvirt 仮想化ツール (virt-manager 及び virsh) を使用します。

KVM は Linux カーネルモジュールの集合であり、デバイス、メモリー、及び Hypervisor モジュール自身の為の API 管理を管理します。仮想化ゲストは これらのモジュールに制御される Linux のプロセスとスレッドとして実行 されます。

ゲストシステム

ゲスト、仮想マシン、あるいは domU としても知られています。

ドメイン

domU と ドメイン は両方共ドメインです。 ドメインは Hypervisor 上で稼働します。 ドメインと言う用語は 仮想マシン に 似た意味を持ち、その双方は技術的に交換可能です。ドメインは仮想マシンです。

ハードウェア仮想マシン

参照 完全仮想化

ベアメタル

この用語、ベアメタルとはコンピュータの背後にある物理的アーキテクチャを指す言葉です。 ベアメタル上でオペレーティングシステムを実行すると言うことは、言い方を変えると、物理的 ハードウェア上で無修正のオペレーティングシステムを実行すると言うことです。ベアメタル上で 実行しているオペレーティングシステムの例としては *dom0* 、又は通常にインストールされたオペレーティングシステムがあります。

ホスト

ホストオペレーティングシステム。別名、dom0

ホストオペレーティングシステム環境は 完全仮想化した と Para-virtualized の両ゲストシステムの為に仮想化ソフトウェアを 実行します。

仮想マシン

仮想マシンとは、物理マシンのソフトウェア実装、あるいはプログラミング 言語です（例えば、Java ランタイム環境や LISP）。仮想化のコンテキストでは、仮想マシンは仮想化したハードウェア上で稼働しているオペレーティングシステムと 言う意味になります。

仮想化

仮想化とは、ソフトウェア実行の為の幅広い意味のコンピューティング用語です。通常は、１つのシステム上で他のプログラムと並行していて、且つ分離しているオペレーティングシステムのことです。仮想化のほとんどの既存実装は、hypervisor を 使用します。hypervisor はオペレーティングシステム上にあるソフトウェアレイヤーで あり、ハードウェアを抽出するためのものです。hypervisor は ゲストオペレーティング システムに仮想化ハードウェアを与えることによって複数のオペレーティング システムが同じ物理システム上で稼働できるようにします。オペレーティングシステムを 仮想化する方法は以下のように各種存在します:

- ハードウェアで支援された仮想化として、Xen と KVM を使用した 完全仮想化があります。（定義：完全仮想化）
- Para-virtualization は Linux ゲストを実行するために Xen で使用される技術です。 （定義：Para-virtualization ）
- ソフトウェア仮想化、又は模倣。ソフトウェア仮想化はバイナリトランスレーションと 他の模倣技術を使用して無修正のオペレーティングシステムを実行します。ソフトウェア 仮想化はハードウェア支援の仮想化や para-virtualization よりもかなり遅くなります。 ソフ

トウェア仮想化は、QEMU の形式であり、Red Hat Enterprise Linux ではサポートが ありません。

仮想化 CPU

システムは、物理プロセッサコアの数に比較して数多くの仮想 CPU (VCPU) を持ちます。 仮想 CPU の数量は有限であり、ゲスト仮想マシンに割り当て可能な仮想 CPU の 全数を示します。

完全仮想化

Xen と KVM は完全仮想化を使用できます。完全仮想化はプロセッサのハードウェア機能を 使用して、背後にある物理システム (ベアメタル) の 全面的抽出を提供して、ゲストオペレーティングシステムが稼働できる新規の仮想システムを 作成します。このゲストオペレーティングシステムには修正の必要がありません。ゲスト オペレーティングシステムとゲスト上のアプリケーションはいずれも仮想環境の認識がなく、 通常通りに稼働します。Para-virtualization は Linux オペレーティングシステムの修正版を 必要とします。

完全仮想化した

完全仮想化 を参照

移行

移行（Migration）とは、仮想化ゲストを１つのホストから別のホストに移動する プロセスの呼び名です。移行はオフラインでも実施できます（ゲストは休止してそれから 移動）。又はライブでも可能です（ゲストは休止せずに移動）。Xen 完全仮想化ゲスト、 Xen para-virtualized ゲスト、及び KVM 完全仮想ゲストは全て移行できます。

移行は仮想化の基幹機能です。ここではソフトウェアが完全にハードウェアから 分離されています。移行は以下の部分で役に立ちます:

- ロードバランシング：ホストがロード超過になった時点で、ゲストは使用度の低い 別のホストへと移動されます。
- ハードウェアフェイルオーバー：ホスト上のハードウェアデバイスがスタートに 失敗すると、ゲストは移動してホストが安全に停止と修理をできるようになります。
- 節電：低使用の期間にはゲスト群は他のホスト群に分配できるため、主ホストシステムは電力を落として、節電とコスト低減ができます。
- 地域的な移行：より少ない遅延の為や、重大な事態の場合にはゲストは他の 場所に移動できます。

共有のネットワークストレージがゲストイメージの保存に使用されます。共有 ストレージ無しでは移行は不可能です。

オフライン移行はゲストを休止して、それからゲストのメモリーのイメージを 移動先のホストに移動します。ゲストは移動先ホスト上で再開始されて、移動前の ホスト上でゲストが使用したメモリーは開放されます。

オフライン移行にかかる時間はネットワークのバンド幅と遅延により変化します。 2GB のメモリーを持つゲストは、1 Gbit のイーサネットリンク上で数秒かかる でしょう。

ライブ移行は移動元のホストでゲストを稼働状態に保ち、ゲストを停止せずに メモリーの移動を開始します。修正されたメモリーページは全て追跡されて イメージが送信された後に、目的地に送信されます。そのメモリーは変更のあったページで更新されます。このプロセスは、なんらかの発見的手法に到達するまで継続されます：それは全てのページが正常にコピーされるとか、又は 移動元の変更が速過ぎて、目的地のホストが進展できない状態などです。発見的手法に到達すると、ゲス

トは移動元のホスト上で一時停止して、レジスターとバッファが 送信されます。レジスターは目的地のホスト上でロードされて、ゲストはそれから その新しいホスト上で復帰します。ゲストがマージできない場合（ゲストがロード超過の時など 発生）、ゲストは一時停止して、それからオフライン移行が代理として開始されます。

ライブ移行にかかる時間はネットワークのバンド幅と遅延度と、更にはゲスト上の 活動にも影響されます。ゲストがかなりの量の I/O 又は CPU を使用している場合、 移行にはより時間がかかります。

付録C 改訂履歴

改訂 4-74.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 4-74 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

改訂 5.4-72 **Fri Nov 06 2009** **Christopher Curran**
Fixes errors reported by translators.

改訂 5.4-64 **Wed Oct 07 2009** **Christopher Curran**
Fixes VCPU allocation bug.

改訂 5.4-61 **Wed Sep 23 2009** **Christopher Curran**
Fixes BZ#524350, BZ#508606 and BZ#517221
Remote 管理、タイミング管理、及び para-virtualized ドライバーを拡張して、是正します。
ローカリゼーションチームのクレジットを追加

改訂 5.4-59 **Thu Sep 17 2009** **Christopher Curran**
Remote Management の章のバグを修復します。

改訂 5.4-58 **Tue Sep 15 2009** **Christopher Curran**
各種エラーを修復します。

改訂 5.4-55 **Fri Sep 11 2009** **Christopher Curran**
Fixes BZ#510058 と BZ#513651

改訂 5.4-53 **Tue Sep 08 2009** **Christopher Curran**
制約の更新、KVM virtio ドライバーの拡張、及び各種編集。

改訂 5.4-51 **Tue Aug 25 2009** **Christopher Curran**
制約の修復と各種編集

改訂 5.4-48 **Thu Aug 20 2009** **Christopher Curran**
新規の PXE インストールセクション

改訂 5.4-47 **Wed Aug 19 2009** **Christopher Curran**
制約 (BZ#517439) 及び デフォルトイメージストレージ場所 (BZ#514117) の更新

改訂 5.4-46 **Thu Aug 13 2009** **Christopher Curran**
制限 と para-virtualized ドライバーコンテンツの更新

改訂 5.4-45 **Mon Aug 10 2009** **Christopher Curran**
KVM para-virtualized ドライバーセクションの更新

改訂 5.4-44 **Thu Aug 06 2009** **Christopher Curran**
32 バグの修正

改訂 5.4-42 **Tue Aug 04 2009** **Christopher Curran**
新規の Red Hat Enterprise Linux インストールの章

改訂 5.4-41 **Mon Aug 03 2009** **Christopher Curran**
ライブ移行の章が完成。各種他のバグ修正とコンテンツ更新。

改訂 5.4-38 **Thu Jul 30 2009** **Christopher Curran**

新規のストレージセクション。ライブ移行の更新。11バグの修正。

改訂 5.4-36 **Thu Jul 30 2009** **Christopher Curran**
各種編集

改訂 5.4-35 **Wed Jul 29 2009** **Christopher Curran**
KVM 用のライブ移行の拡張。各種スクリーンショットエラーの修正。 Xen ライブ移行セクションの更新。

改訂 5.4-33 **Tue Jul 28 2009** **Christopher Curran**
Fixes BZ#513887.

改訂 5.4-32 **Mon Jul 27 2009** **Christopher Curran**
hypervisor スイッチングセクションの追加

改訂 5.4-31 **Fri Jul 24 2009** **Christopher Curran**
KVM 用にインストール章の更新

改訂 5.4-29 **Thu Jul 23 2009** **Christopher Curran**
KVM サポートと制約の更新

改訂 5.4-26 **Fri Jul 17 2009** **Christopher Curran**
新規の Windows Server 2008 インストールセクション

改訂 5.4-24 **Wed Jun 24 2009** **Christopher Curran**
各種コピーの編集と是正。

改訂 5.4-23 **Mon May 11 2009** **Christopher Curran**
制約エラーを修復します。

改訂 5.4-20 **Mon Mar 09 2009** **Christopher Curran**
487407 の解決、及び細かいコピー編集

改訂 5.3-19 **Mon Feb 23 2009** **Christopher Curran**
486294 の解決、及び細かいコピー編集

改訂 5.2-18 **Tue Jan 20 2009** **Christopher Curran**
以下を含む各種バグの解決、及び他のドキュメント修正:
Resolves: BZ #461440
Resolves: BZ #463355
各種スペル間違いと印字エラーの修正

改訂 5.2-16 **Mon Jan 19 2009** **Christopher Curran**
以下を含む各種バグの解決、及び他のドキュメント修正:
Resolves: BZ #469300
Resolves: BZ #469316
Resolves: BZ #469319
Resolves: BZ #469326
Resolves: BZ #444918
Resolves: BZ #449688
Resolves: BZ #479497

改訂 5.2-14 **Tue Nov 18 2008** **Christopher Curran**

以下を含む各種バグの解決、及び他のドキュメント修正:
Resolves: BZ #469300
Resolves: BZ #469314
Resolves: BZ #469319
Resolves: BZ #469322
Resolves: BZ #469326
Resolves: BZ #469334
Resolves: BZ #469341
Resolves: BZ #371981
Resolves: BZ #432235
Resolves: BZ #432394
Resolves: BZ #441149
Resolves: BZ #449687
Resolves: BZ #449688
Resolves: BZ #449694
Resolves: BZ #449704
Resolves: BZ #449710
Resolves: BZ #454706

改訂 5.2-11 **Fri Aug 01 2008** **Christopher Curran**

Resolves: BZ #449681
Resolves: BZ #449682
Resolves: BZ #449683
Resolves: BZ #449684
Resolves: BZ #449685
Resolves: BZ #449689
Resolves: BZ #449691
Resolves: BZ #449692
Resolves: BZ #449693
Resolves: BZ #449695
Resolves: BZ #449697
Resolves: BZ #449699
Resolves: BZ #449700
Resolves: BZ #449702
Resolves: BZ #449703
Resolves: BZ #449709
Resolves: BZ #449711
解決: BZ #449712 各種印字エラー及びスペル間違い
Resolves: BZ #250272
Resolves: BZ #251778
Resolves: BZ #285821
Resolves: BZ #322761
Resolves: BZ #402161
Resolves: BZ #422541
Resolves: BZ #426954
Resolves: BZ #427633
Resolves: BZ #428371
Resolves: BZ #428917
Resolves: BZ #428958
Resolves: BZ #430852
Resolves: BZ #431605
Resolves: BZ #448334
Resolves: BZ #449673
Resolves: BZ #449679
Resolves: BZ #449680

改訂 5.2-10 **Wed May 14 2008** **Christopher Curran**

トラブルシューティング、ネットワーク、及びインストール用の 新規、又は書き換えのセクション
スペリング、文法、及び言語用の各種更新
解決した形式とレイアウト問題
使い易さと読みやすさを強化する為の専門用語と単語使用の更新

改訂 5.2-9 **Mon Apr 7 2008** **Christopher Curran**

余分な章とヘッディングを削除するための文書更新
仮想マシンマネージャは 5.1 に更新

改訂 5.2-7 **Mon Mar 31 2008** **Christopher Curran**

Resolves: #322761
多くのスペリングと文法エラーの修正
Remote Management の章を追加

改訂 5.2-5 **Wed Mar 19 2008** **Christopher Curran**

Resolves: #428915
新規の仮想化ガイドの作成

付録D COLOPHON

このマニュアルは DocBook XML v4.3 形式で書かれています。

このマニュアルは Jan Mark Holzer と Chris Curran の作業を基にしています。

その他の著作クレジット:

- Don Dutile は para-virtualized ドライバーセクションの技術編集に貢献。
- Barry Donahue は para-virtualized ドライバーセクションの技術編集に貢献。
- Rick Ring は仮想マシンマネージャセクションの技術編集に貢献。
- Michael Kearey は virsh 及び 仮想化フロッピードライブでの XML 設定ファイル使用に関するセクションの 技術編集に貢献。
- Marco Grigull は ソフトウエア互換性とパフォーマンスセクションの技術編集に貢献。
- Eugene Teo は virsh セクションでのゲスト管理の技術編集に貢献。

このマニュアルを作成した発行ツール、Publican は Jeffrey Fearn によって 書かれました。

Red Hat ローカリゼーションチームは以下の人々で構成されています:

- 簡体中国語
  - Leah Wei Liu
- 繁体中国語
  - Chester Cheng
  - Terry Chuang
- 日本語
  - Kiyoto Hashida
- 韓国語
  - Eun-ju Kim
- フランス語
  - Sam Friedmann
- ドイツ語
  - Hedda Peters
- イタリア語
  - Francesco Valente
- ブラジル系ポルトガル語
  - Glaucia de Freitas

    - Leticia de Lima
- スペイン語
    - Angela Garcia
    - Gladys Guerrero
- ロシア語
    - Yuliya Poyarkova