Red Hat Enterprise Linux 6

仮想化のチューニングと最適化ガイド

仮想環境を最適化する

エディッション 0.3

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 仮想化のチューニングと最適化ガイド

仮想環境を最適化する
エディッション 0.3

Scott Radvan
Red Hat, Inc. Engineering Content Services
sradvan@redhat.com

法律上の通知

概要

Red Hat Enterprise Linux 仮想化のチューニングと最適化ガイドでは、 KVM および仮想化のパフォーマンスについて解説しています。 KVM パフォーマンス機能やオプションをホストとなるシステムやゲストとなる仮想マシンにフル活用させるためのヒントや推奨事項などが記載されています。

目次

第1章 はじめに

1.1. 本ガイドについて

Red Hat Enterprise Linux 仮想化のチューニングと最適化ガイドでは、 Red Hat Enterprise Linux ホストおよびゲスト仮想マシンの最適なパフォーマンスを得るため設定できるオプション、 セッティング、推奨事項などについて詳しく解説しています。

本ガイドは次のようなセクションで構成されています。

- Virt-manager
- tuned
- ネットワーク
- メモリー
- ブロック IO
- NUMA
- パフォーマンス監視ツール

1.2. 他の参考文献

今お読み頂いている本ガイドでは仮想環境のチューニングと最適化に焦点を置いていますが、 本ガイド以外にも Red Hat では仮想化関連のガイドを提供しています。

次のようなガイドもご参照ください、
https://access.redhat.com/knowledge/docs/Red_Hat_Enterprise_Linux/。

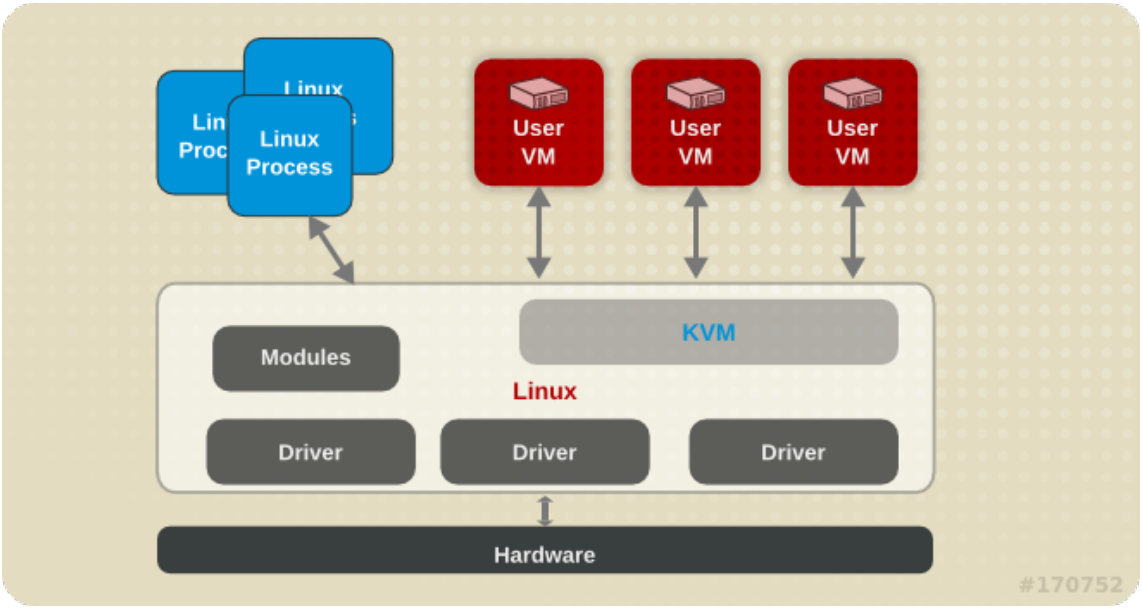
- **仮想化スタートガイド**
  - 仮想化の概念やその利点、 ツール関連などを解説している他、 Red Hat が提供している仮想化関連のドキュメントや製品などの概要についても触れています。
- **仮想化ホスト設定およびゲストインストールガイド**
  - 仮想化ソフトウェアのインストール方法、 および仮想化ホストでのゲストマシンの設定方法について解説しています。
- **仮想化管理ガイド**
  - ホスト、 ネットワーク、 ストレージ、 デバイス、 ゲストなどを管理する方法、 virt-manager または virsh の使い方などの他、 libvirt や qemu やトラブルシューティングについても触れています。
- **仮想化セキュリティガイド**
  - Red Hat で提供している仮想化の安全性に関する技術についての要約です。 また、 仮想化した環境下で、 ホストやゲスト、 共有インフラ、 共有リソースなどを保護するための推奨事項についても記載されています。
- **V2V ガイド**

- Red Hat Enterprise Virtualization 以外のハイパーバイザーから Red Hat Enterprise Virtualization に仮想マシンをインポートする方法について解説しています。

- **Hypervisor 導入ガイド**
  - Red Hat Enterprise Virtualization Hypervisor の取得、 導入、 設定について解説しています。

1.3. KVM の概要

KVM の構造を以下に図解します。



1.4. KVM のパフォーマンスについて

KVM はシステムのパフォーマンスやプロセス、 スレッドの管理に関連するため、 以下に概要を示します。

- KVM を使用する場合、 ゲストはホスト上の Linux プロセスとして実行されます。
- 仮想 CPU は通常のスレッドとして実装され、 Linux スケジューラで処理されます。
- ゲストは NUMA や Huge Pages などのようにカーネルから機能を継承します。
- ホスト内でのディスクおよびネットワーク I/O の設定はパフォーマンスに多大な影響を与えます。
- ネットワークトラフィックは一般的にはソフトウェアベースのブリッジを通過します。

1.5. パフォーマンス関連の特長および改善点

- CPU／カーネル
  - NUMA - Non-Uniform Memory Access、 詳細は *7章NUMA* を参照してください。
  - CFS - Completely Fair Scheduler、 クラスに焦点を置いた新しいスケジューラです。
  - RCU - Read Copy Update、 共有スレッドのデータの処理が向上しています。

- 仮想 CPU (vCPU) の最大数は 160 になります。

- メモリー
  - Huge Pages およびメモリー集約的環境での各種の最適化、 詳細は 5章*メモリー* を参照してください。

- ネットワーク
  - vhost-net - カーネルベースの高速 virtIO ソリューション
  - SR-IOV - ネイティブに近いネットワークパフォーマンスのレベル向け

- ブロック I/O
  - AIO - 他の I/O 動作にオーバーラップするスレッドのサポート
  - MSI - PCI バスデバイス割り込み生成
  - Scatter Gather - データのバッファ処理のため改善された I/O モード

注記

仮想化に関するサポート、 制約、 特長などの詳細については、 『Red Hat Enterprise Linux 6 仮想化スタートガイド』 および以下の URL を参照してください。

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

http://www.redhat.com/resourcelibrary/articles/virtualization-limits-rhel-hypervisors

第2章 VIRT-MANAGER

2.1. はじめに

本章では、 ゲスト仮想マシンの管理を目的としたデスクトップツールとなる virt-manager のパフォーマンス関連のオプションについて見ていきます。

2.2. オペレーティングシステムの詳細とデバイス

2.2.1. ゲスト仮想マシンの詳細を指定する

virt-manager ツールでは、 新規のゲスト仮想マシンに選択するオペレーティングシステムの種類やバージョンによって異なるプロファイルを提供します。 ゲストを作成する際は、 できるだけ詳細な情報を入力するようにしてください。 特定タイプのゲストに利用できる機能を有効にすることでパフォーマンスを向上させることができます。

新しいゲスト仮想マシンを作成する際は、 必ずオペレーティングシステムのタイプとバージョンを指定してください。 以下に virt-manager ツールの画面を示します。

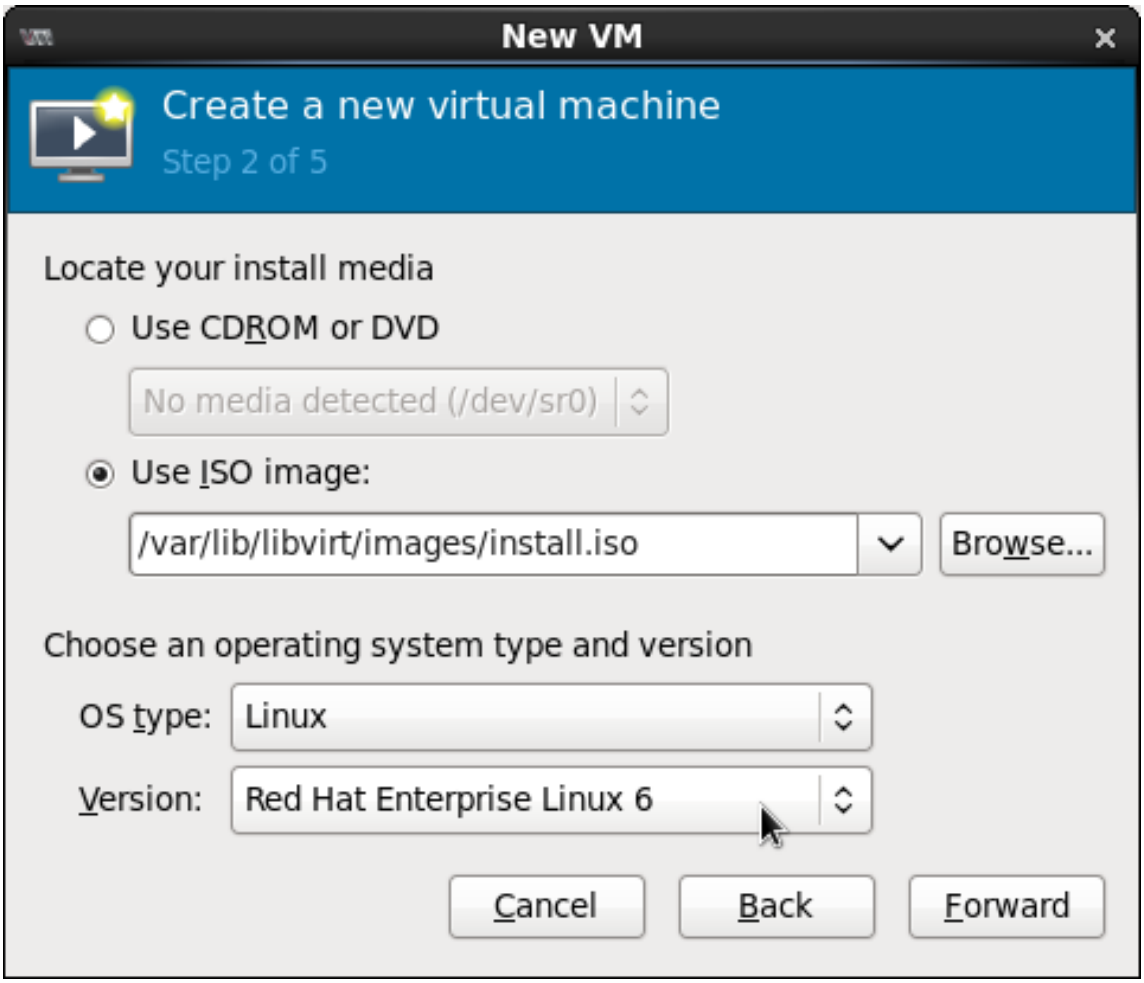


図2.1 OS の種類とバージョンを指定する

2.2.2. 未使用のデバイスを削除する

未使用や不要なデバイスを削除することでパフォーマンスが向上する場合があります。 たとえば、 Web サーバーとして稼働させることを目的としたゲストの場合、 オーディオ機能や接続タブレットなどは必要とされることはあまりありません。

削除ボタンをクリックして不要なデバイスを削除します。 以下に virt-manager ツールの画面を示します。

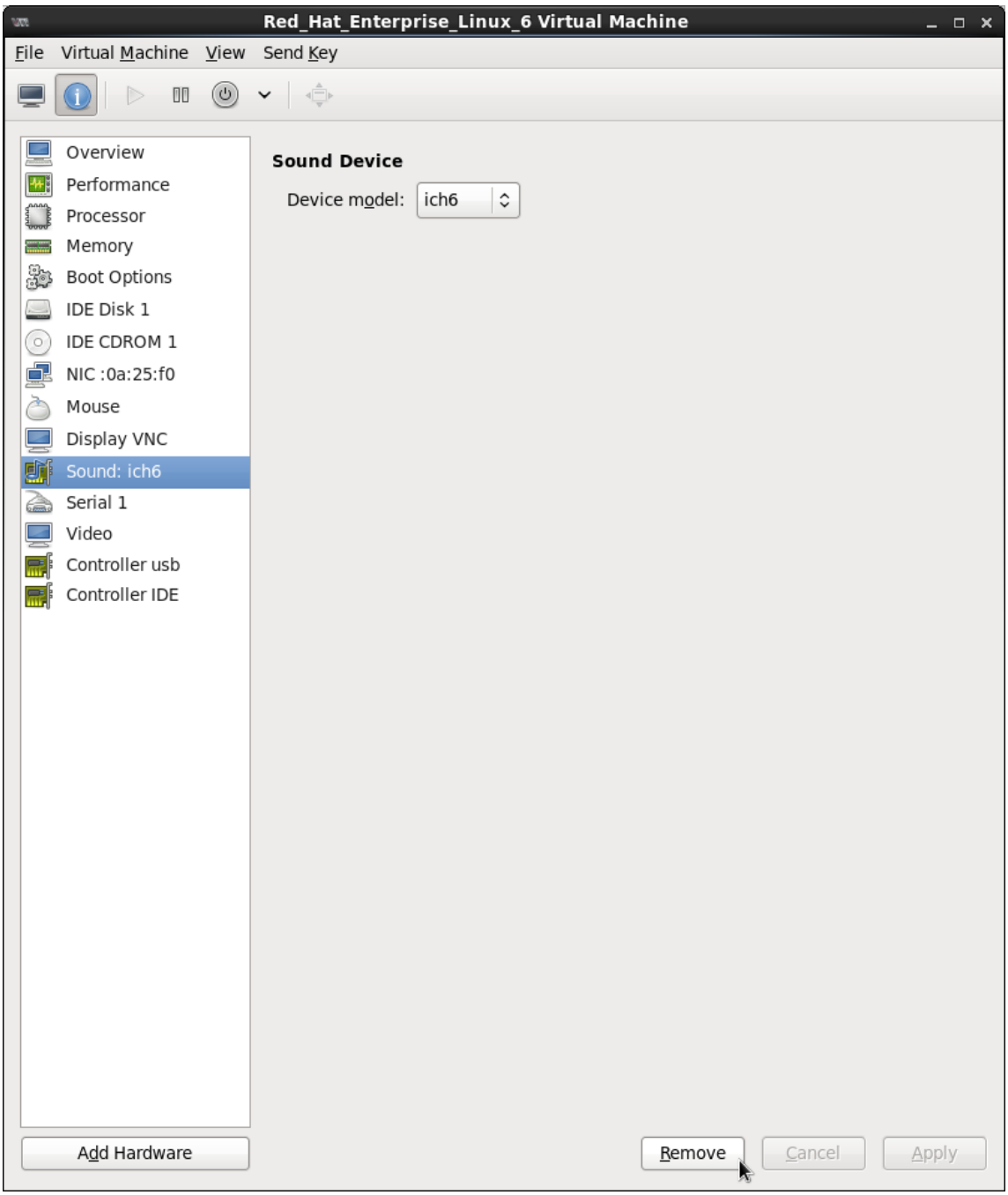


図2.2 未使用のデバイスを削除する

2.3. CPU パフォーマンスのオプション

ゲスト仮想マシンに設定できる CPU 関連のオプションがいくつかあります。 正しく設定することにより、 パフォーマンスに大きな影響を与えることができます。 ゲストに対して利用できる CPU オプションを以下に示します。 このセクションでは、 以下に示すオプションがもたらす影響について見ていきます。

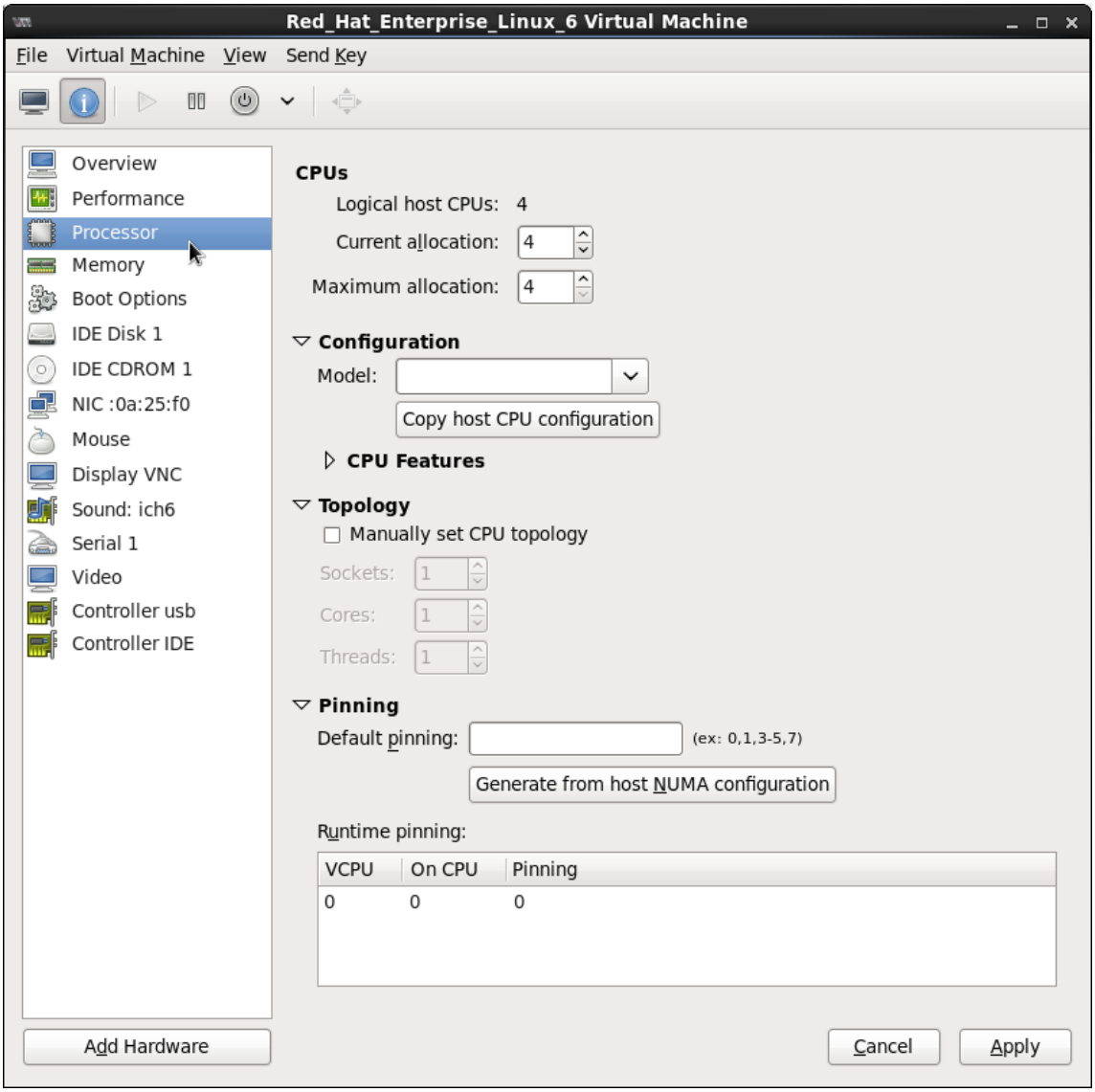


図2.3 CPU パフォーマンスのオプション

2.3.1. オプション: 使用できる CPU

このオプションでゲストが使用できる仮想 CPU 数を調整します。 ホストで使用可能な数以上の CPU 数を割り当てると (**オーバーコミット** と呼ばれる)、 以下のような警告が表示されます。

図2.4 CPU のオーバーコミット

警告

CPU のオーバーコミットはパフォーマンスに悪影響をもたらす場合があります。詳細については『Red Hat Enterprise Linux 6 仮想化管理ガイド』の「**KVM でオーバーコミットを行なう**」を参照してください。

2.3.2. オプション: CPU 構造

このオプションで目的の CPU モデルに応じた CPU 構造タイプを選択します。 一覧を表示して使用できるオプションから選択するか、 **ホストの CPU構造をコピー (Copy host CPU configuration)** ボタンをクリックしてホストの物理的な CPU モデルと構造を検出させ適用します。 CPU 構造を選択すると、それに応じた CPU 機能と指示が表示され、 **CPU 機能 (CPU Features)** の一覧で別々に有効にしたり無効にしたりすることができます。 例を以下に示します。

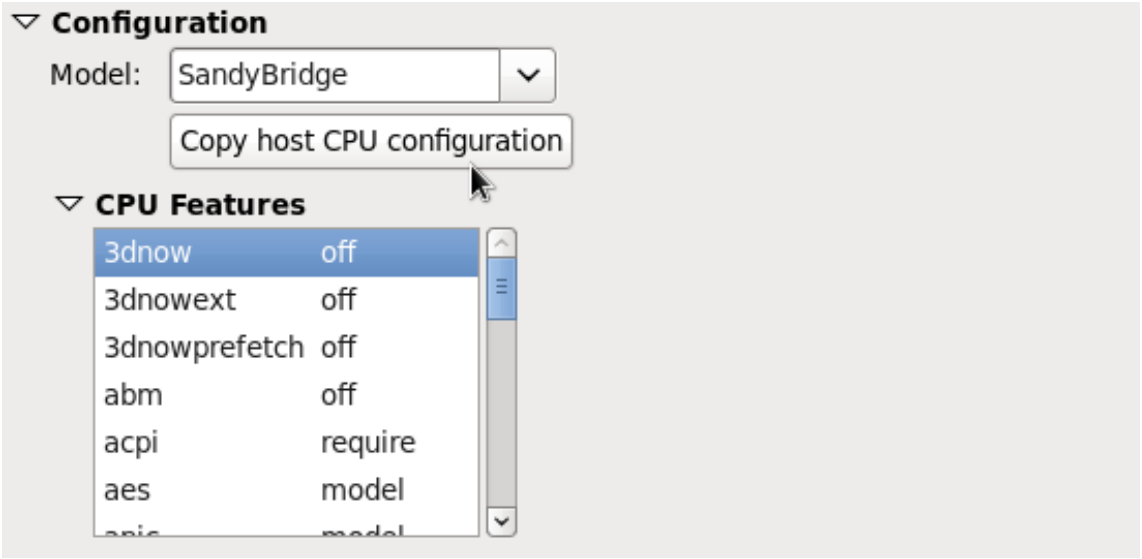


図2.5 CPU 構造のオプション

注記

手動で選択するより、 ホストの CPU 構造をコピーすることをお勧めします。

2.3.3. オプション: CPU トポロジー

このオプションでゲスト仮想マシンの仮想 CPU に特定の CPU トポロジーを適用します (ソケット、 コア、 スレッドなど)。 以下に例を示します。

図2.6 CPU トポロジーのオプション

注記

環境によっては要件が異なることもありますが、 ソケット数を選択する場合、 コア、スレッドいずれもひとつのみにした方が一般的には最善のパフォーマンスとなります。

2.3.4. オプション: CPU の Pinning (固定)

システム固有の NUMA トポロジーに合わせると大幅なパフォーマンスの改善が見られます。 このオプションでホストに有効となる Pinning 構成を自動的に作成します。

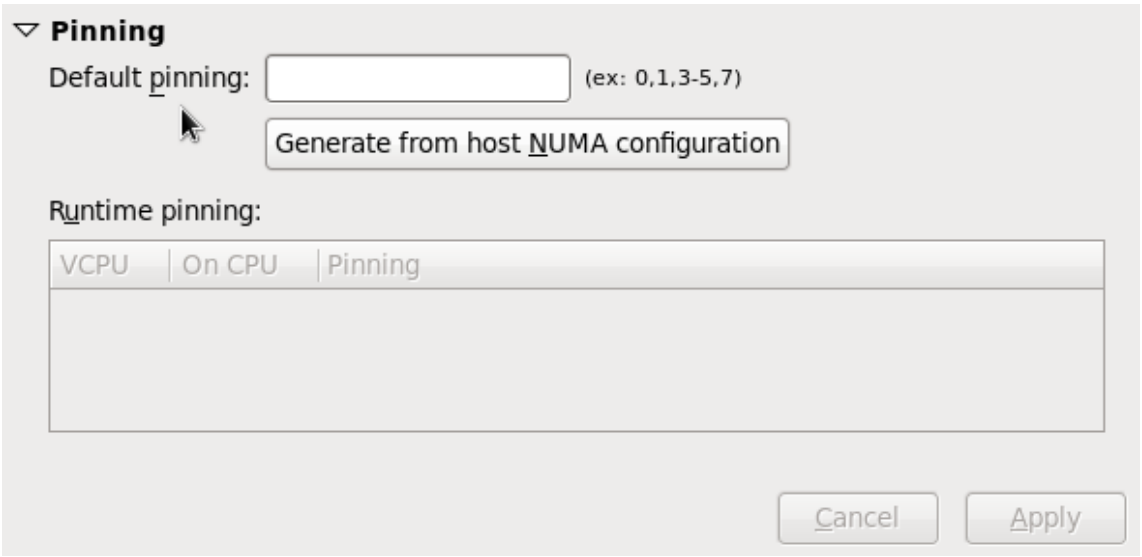


図2.7 CPU Pinning

警告

ゲストに単一 NUMA ノードを越える数の VCPU 数を持たせている場合は、 このオプションは使用しないでください。

Pinning オプションを使用すると、 ゲストの VCPU スレッドが単一の NUMA ノードに制約されます。 ただし、 その NUMAノード内でのスレッドの移動は可能です。 制約の機能を強化させる場合は、 **lscpu** コマンドからの出力を使用し、 **virsh cpupin** で 1:1 の物理 CPU と VCPU の結び付けを作ります。 NUMA と CPU Pinning については *7章NUMA* を参照してください。

第3章 TUNED

3.1. TUNED と TUNED-ADM

Tuned は、 システムの各種コンポーネントの使用量に関するデータを監視、 収集し、 その情報を使って必要に応じてシステム設定を動的に調整するデーモンです。 CPU やネットワークの使用量の変化に反応し、 使用中のデバイスのパフォーマンスを高めたり、 使用していないデバイスの電力消費を節約したりするため設定を調整することができます。

Tuned に付随する **ktune** は **tuned-adm** ツールとともに、 事前設定されたチューニングプロファイルを提供します。 特定のユースケースでパフォーマンスを強化し、電力消費量を抑制します。 プロファイルを編集したり、 新規のプロファイルを作成したりすることで、 環境に適したパフォーマンスソリューションを作り出すことができます。

tuned-adm の一部として提供される仮想化関連のプロファイルには、以下のものがあります。

virtual-guest

enterprise-storage プロファイルをベースとしています。 ***virtual-guest*** でも仮想メモリーの swap を低減します。 このプロファイルは Red Hat Enterprise Linux 6.3 以降よりご利用頂けます。 ゲストマシン向けに推奨されるプロファイルです。

virtual-host

enterprise-storage プロファイルをベースとしています。 ***virtual-host*** でも仮想メモリーの swap を低減し、 ダーティーページのより積極的なライトバックを可能にします。 このプロファイルは Red Hat Enterprise Linux 6.3 以降よりご利用頂けます。 KVM ホストや Red Hat Enterprise Virtualization ホストなど、 仮想化ホスト向けに推奨されるプロファイルです。

以下のコマンドを使用して tuned パッケージと関連の **systemtap** スクリプトをインストールします。

```
yum install tuned
```

tuned パッケージをインストールすると、 **/etc/tuned.conf** にサンプルの設定ファイルが設定され、デフォルトのプロファイルがアクティベートされます。

以下を実行して **tuned** を開始します。

```
service tuned start
```

マシンを起動する度に **tuned** が開始されるよう以下を実行します。

```
chkconfig tuned on
```

利用可能な全プロファイルを一覧表示して、 現在アクティブなプロファイルを特定するため以下を実行します。

```
tuned-adm list
```

現在アクティブなプロファイルだけを表示する場合は、 以下を実行します。

```
tuned-adm active
```

別のプロファイルに切り替える場合は、 以下を実行します。

```
tuned-adm profile profile_name
```

例えば以下のとおりです。

```
tuned-adm profile virtual-host
```

全てのチューニングを無効にする場合は、 以下を実行します。

```
tuned-adm off
```

注記

tuned、 **tuned-adm**、 **ktune** の詳細については、 Red Hat Enterprise Linux 6 『電力管理ガイド』 (http://access.redhat.com/knowledge/docs/Red_Hat_Enterprise_Linux/) を参照してください。

第4章 ネットワーク

4.1. はじめに

本章では、 仮想化した環境におけるネットワークの最適化について説明しています。

4.2. ネットワーク調整のヒント

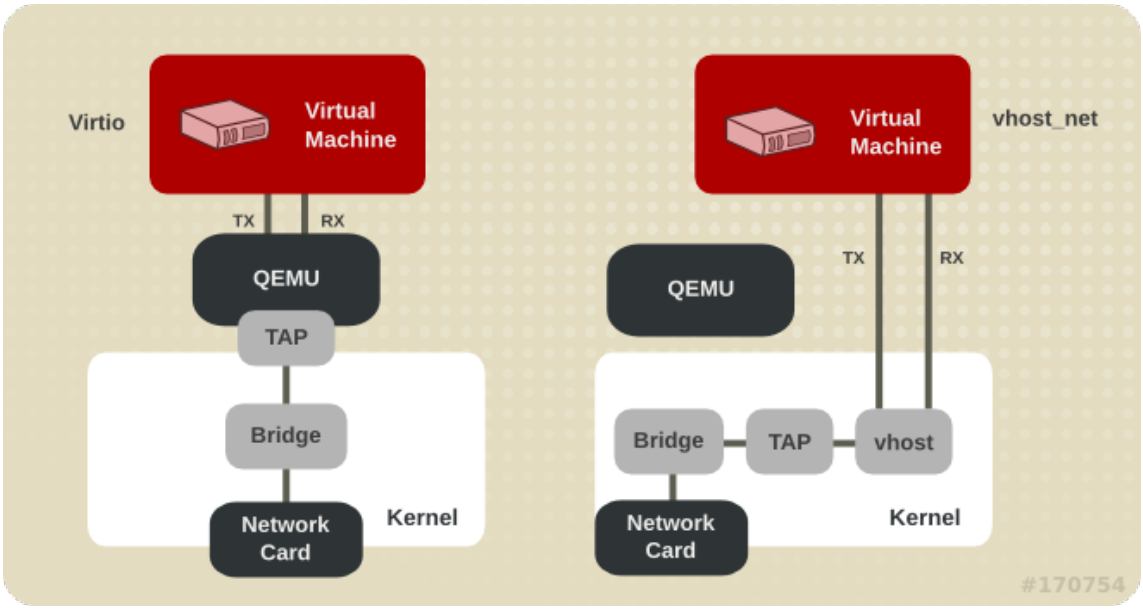
- 単一ネットワークでのトラフィック混雑を避けるため複数のネットワークを使用します。 たとえば、 管理専用のネットワーク、 バックアップ専用のネットワーク、 ライブ移行専用のネットワークなどを設けます。
- 通常は、 すべてのコンポーネントでデフォルトの MTU (1500 バイト) に合わせれば十分でしょう。 大量のメッセージを必要とする場合は、 MTU 値を上げると断片化を軽減することができます。 MTU を変更する場合は、 パス内の全デバイスをその MTU 値に合わせてください。
- arp_filter を使用して ARP 変動を防ぎます。 ARP 変動はホスト、 ゲストいずれでも発生する可能性のある望ましくない状況で、 マシンが ARP 要求に対して複数のネットワークインターフェースから応答してしまう原因となります。 この設定を永続的にするため、 **`echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter`** を行なうか、 **`/etc/sysctl.conf`** の編集をしてください。

注記

ARP 変動に関する詳細は、 次の URL http://linux-ip.net/html/ether-arp.html#ether-arp-flux を参照してください。

4.3. VIRTIO と VHOST_NET

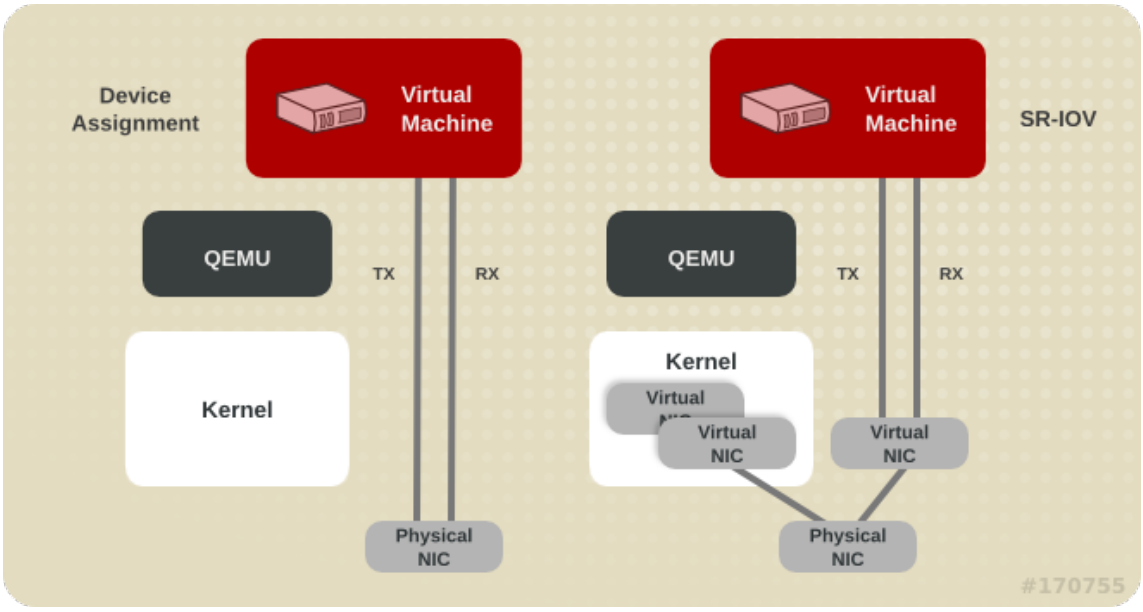
virtio と vhost_net の構造におけるカーネルの位置付け示します。



vhost_net では virtio ドライバーの一部をユーザー領域からカーネルに移動しています。 これによりコピー動作が減り、 待ち時間と CPU 使用量が低減されます。

4.4. デバイスの割り当てと SR-IOV

デバイス割り当てと SR-IOV の構造におけるカーネルの位置付けを示します。



デバイス割り当てでは、 ゲストからデバイス全体が見えます。 SR-IOV では、 NIC やシステムボードなどドライバーやハードウェアのサポートを必要とします。 複数の仮想デバイスを作成することが可能なためそれぞれ別々のゲストに渡すことができます。 ゲスト内に製造元固有のドライバーが必要になりますが、 SR-IOV はあらゆるネットワークオプションの中でも最小の待ち時間を実現します。

第5章 メモリー

5.1. はじめに

本章では、 仮想化した環境におけるメモリーの最適化オプションについて説明しています。

5.2. HUGE PAGES および TRANSPARENT HUGE PAGES

通常、 x86 CPU は 4kB ページ単位でメモリーに対応しますが、 *huge pages* とも言われる大容量ページを使用することも可能です。 TLB （Transaction Lookaside Buffer) に対して CPU キャッシュの使用を増加させてパフォーマンスを向上させる場合、 huge page メモリー対応で KVM のゲストを導入することができます。

Red Hat Enterprise Linux 6 ではデフォルトでカーネル機能が有効になっているため、 huge pages により特に大容量メモリーおよびメモリー集約型の負荷に対してパフォーマンスを大幅に向上させることができます。 Red Hat Enterprise Linux 6 では、 huge pages を使用することでページサイズを増加させ、より効率的な大容量メモリーの管理が可能になります。

ゲストの XML 設定を追加:

```
<memoryBacking>
 <hugepages/>
</memoryBacking>
```

現在の huge pages の値を確認:

```
cat /proc/sys/vm/nr_hugepages
```

```
cat /proc/meminfo | grep Huge
```

huge pages 数の設定:

```
echo xyz > /proc/sys/vm/nr_hugepages
```

注記

`/etc/sysctl.conf` 内の **vm.nr_hugepages** の値を変更すると設定が永続的になります。

huge pages はホストだけでなくゲストにとっても便利ですが、 合計ページ数は必ずホスト内で利用できるページ数より少なくしてください。

空きメモリーをすべてキャッシュとして使用できるようにするとパフォーマンスが向上します。 **`/sys/kernel/mm/redhat_transparent_hugepage/enabled`** を always に設定すると、 Transparent Hugepages がデフォルトで使用されます。

Transparent Hugepage のサポートで hugetlbfs の使用が妨げられることはありません。 ただし、 hugetlbfs を使用しない場合、 KVM では通常の 4kb ページサイズではなくTransparent hugepages が使用されます。

第6章 ブロック I/O

6.1. キャッシング

表6.1 キャッシングのオプション

キャッシングオプション	詳細
Cache=none	ゲストからのI/O はホストではキャッシュされませんが、ライトバックディスクキャッシュに保持することができます。 大量 I/O 要求が多いゲストにはこのオプションを使用します。 一般的にこのオプションが最適で移行に対応する唯一のオプションとなります。
Cache=writethrough	ゲストからの I/O はホストにはキャッシュされませんが、 物理的な媒体に書き込むことができます。 このモードを使用すると速度の低下が見られ、 またゲスト数の規模が一定以上になると問題が発生する傾向にあります。 I/O 要求が低い小規模のゲスト群への使用に適しています。 ゲスト移行の必要性がなく、 ライトバックキャッシュに対応していないゲストに推奨されるモードです(Red Hat Enterprise Linux 5.5 以前)。
Cache=writeback	ゲストからの I/O がホストにキャッシュされます。

キャッシングのモードは、 **virt-manager** の **仮想ディスク (Virtual Disk)** のセクションで選択することができます。 以下に示すように、 **パフォーマンス (Performance options)** でキャッシュモードを選択します。

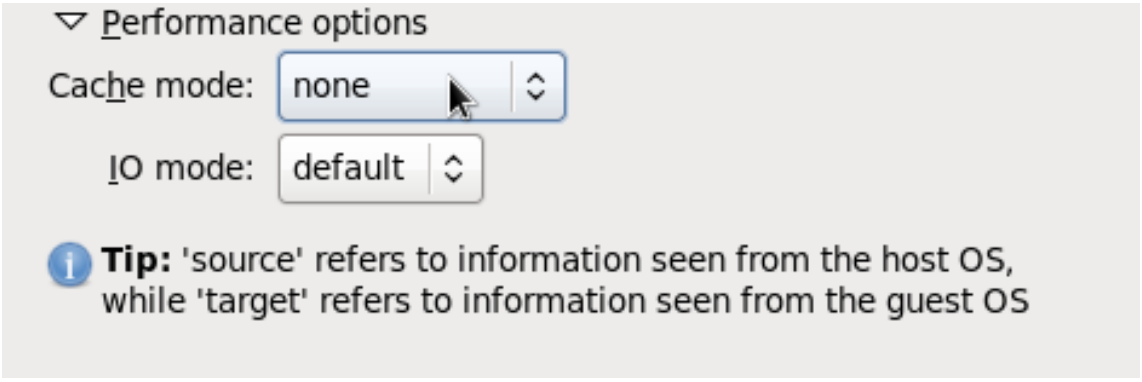


図6.1 キャッシングモード

6.2. ブロック I/O 関連のコマンド

ブロックディスクのパラメータの設定、 表示、 クエリーを行なう場合は、 **blkiotune** コマンドと **blkdeviotune** コマンドを使用します。 詳細については **virsh** の man ページを参照してください。

第7章 NUMA

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

最近の x86 プロセッサでは、 この動作が異なってきています。 NUMA (Non-Uniform Memory Access) では、 システムのメモリーが複数のゾーンに区分けされ (**ノード** と呼ばれる)、 特定の CPU やソケットに割り当てられます。 CPU に対してローカルとなるメモリーへのアクセスが、 そのシステムのリモート CPU に接続しているメモリーに比べて速くなります。

7.1. メモリー割り当てのポリシー

システム内のノードからどのようにメモリーを割り当てるのかは 3 つのポリシーで定義されます。

Strict

目的のノードにメモリーを割り当てられない場合、 他のノードにフォールバックするのが割り当てのデフォルト動作になります。 Strict ポリシーでは、 目的のノードにメモリーを割り当てられない場合は割り当てに失敗することになります。

Interleave

メモリーページはノードマスクで指定された複数のノード全体に割り当てられますが、 割り当てはラウンドロビン方式で行なわれます。

Preferred

単一の優先メモリーノードからのみメモリーの割り当てが行なわれます。 十分なメモリーが使用できない場合には、 他のノードからメモリーを割り当てることができます。

XML 設定で目的のポリシーを有効にします。

```
<numatune>
 <memory mode='preferred' nodeset='0'>
</numatune>
```

7.2. LIBVIRT の NUMA 調整

7.2.1. NUMA の VCPU pinning

以下の XML 設定例では、 ドメインプロセスが物理 CPU 0 から 7 に固定されています。 vCPU スレッドはそれ自体の cpusetに固定されています。 たとえば、 vCPU0 は物理 CPU0 に、 vCPU1 は物理 CPU 1 にといった具合に固定されています。

```
<vcpu cpuset='0-7'>8</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='0'/>
  <vcpupin vcpu='1' cpuset='1'/>
  <vcpupin vcpu='2' cpuset='2'/>
  <vcpupin vcpu='3' cpuset='3'/>
  <vcpupin vcpu='4' cpuset='4'/>
  <vcpupin vcpu='5' cpuset='5'/>
```

```
  <vcpupin vcpu='6' cpuset='6'/>
  <vcpupin vcpu='7' cpuset='7'/>
 </cputune>
```

vcpu と vcpupin のタグ間には直接的な関係があります。 vcpupin オプションを指定しないと、 その値が自動的に確定され親となる vcpu タグオプションから継承されます。 以下の設定には **vcpu 5** の <vcpupin> が足りません。 したがって、 **vCPU5** は親タグの <vcpu>: で指定しているように物理 CPU の 0 から 7 に固定されます。

```
<vcpu cpuset='0-7'>8</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='0'/>
  <vcpupin vcpu='1' cpuset='1'/>
  <vcpupin vcpu='2' cpuset='2'/>
  <vcpupin vcpu='3' cpuset='3'/>
  <vcpupin vcpu='4' cpuset='4'/>
  <vcpupin vcpu='6' cpuset='6'/>
  <vcpupin vcpu='7' cpuset='7'/>
 </cputune>
```

7.2.2. ドメインプロセス

Red Hat Enterprise Linux で規定されている通り、 libvirt ではドメインプロセスのメモリーバインディングのポリシー設定に libnuma を使用します。 このポリシーのノードセットは、 **static** (ドメイン XML に指定) か **auto** (numad のクエリーで設定) のいずれかに設定することができます。 <numatune> タグ内にこのポリシーを設定する方法を以下の XML 設定例で示します。

```
<numatune>
 <memory mode='strict' placement='auto'/>
</numatune>
```

```
<numatune>
 <memory mode='strict' nodeset='0,2-3'/>
</numatune>
```

libvirt では、 **sched_setaffinity(2)** を使ってドメインプロセスの CPU バインディングポリシーを設定します。 cpuset オプションは **static** (ドメイン XML で指定) か **auto** (numad のクエリーで設定) にすることができます。 <vcpu> タグ内にこのポリシーを設定する方法を以下の XML 設定例で示します。

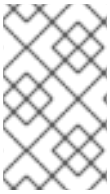
```
<vcpu placement='auto' current='8'>32</vcpu>
```

```
<vcpu placement='static' cpuset='0-10,^5'>8</vcpu>
```

<vcpu> および <numatune> に使用するモードの配置には暗黙的な継承ルールがあります。

- <numatune> の配置モードでは <vcpu> と同じ配置モードにデフォルト設定されます。 または、 <nodeset> を指定した場合は *static* に設定されます。
- 同様に、 <vcpu> の配置モードでは <numatune> と同じ配置モードにデフォルト設定されます。 または、 <cpuset> を指定した場合は *static* に設定されます。

つまり、 ドメインプロセスのメモリー調整と CPU 調整は別々に指定、 定義することができますが、互いの配置モードに依存するよう設定することもできるということです。



注記

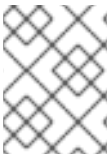
vcpu および numatune に関する詳細は、
http://libvirt.org/formatdomain.html#elementsCPUAllocation および
http://libvirt.org/formatdomain.html#elementsNUMATuningを参照してください。

7.2.3. ドメインの vcpu スレッド

ドメインプロセスの調整の他、 libvirt では XML 設定内の vcpu の各スレッドに pinning ポリシーの設定を行なうことができます。 設定は <cputune> タグ内で行ないます。

```
<cputune>
 <vcpupin vcpu="0" cpuset="1-4,^2"/>
 <vcpupin vcpu="1" cpuset="0,1"/>
 <vcpupin vcpu="2" cpuset="2,3"/>
 <vcpupin vcpu="3" cpuset="0,4"/>
</cputune>
```

このタグでは、 libvirt は cgroup か **sched_setaffinity(2)** のいずれかを使って vcpu スレッドを指定 cpuset に固定しています。



注記

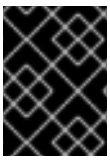
cputune の詳細については、 http://libvirt.org/formatdomain.html#elementsCPUTuning を参照してください。

7.2.4. emulatorpin の使い方

ドメインプロセスの pinning ポリシーを調整する別の方法として、 <cputune> 内で <emulatorpin> タグを使用する方法があります。 例を示します。

```
<cputune>
 <emulatorpin cpuset="1-3"/>
</cputune>
```

7.2.5. virsh を使って vcpu の CPU pinning を調整する



重要

以下に示すのは説明を目的とした例に過ぎません。 実際には、 使用する環境に適した値を入力する必要があります。

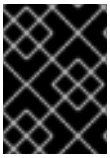
以下の **virsh** コマンドの例では、 ID が 1 となる vcpu スレッド (rhel6u4) を物理 CPU 2 に固定しています。

```
% virsh vcpupin rhel6u4 1 2
```

virsh コマンドでは、 現在の vcpu pinning 設定を取得することもできます。 例を示します。

```
% virsh vcpupin rhel6u4
```

7.2.6. virsh を使ってドメインプロセスの CPU pinning を調整する



重要

以下に示すのは説明を目的とした例に過ぎません。 実際には、 使用する環境に適した値を入力する必要があります。

emulatorpin オプションでは、 各ドメインプロセスに関連付けられたスレッドに CPU 親和性の設定を適用します。 pinning を完了するには、 **virsh vcpupin** (前述) と **virsh emulatorpin** の両方を各ゲストに使用する必要があります。 例を示します。

```
% virsh emulatorpin rhel6u4 3-4
```

7.2.7. virsh を使ってドメインプロセスのメモリーポリシーを調整する

ドメインプロセスのメモリーは動的な調整が可能です。 次にコマンドの使用例を示します。

```
% virsh numatune rhel6u4 --nodeset 0-10
```

コマンドの詳細な使用例については、 **virsh** の man ページをご覧ください。

第8章 パフォーマンス監視ツール

8.1. はじめに

本章では、 ゲストの仮想マシン環境の監視に使用するツールについて見ていきます。

8.2. PERF KVM

ホストからゲストのオペレーティングシステムの統計値を収集するには、 **perf** コマンドに **kvm** オプションを付けて使用します。

Red Hat Enterprise Linux では、 **perf** パッケージで **perf** コマンドを提供しています。 **rpm -q perf** を実行して **perf** パッケージがインストールされているか確認します。 インストールされていない場合は、ゲストのオペレーティングシステムの統計値を収集して分析するため、 root ユーザーになり次のコマンドを使用してこのパッケージをインストールします。

```
yum install perf
```

ホストで **perf kvm** を使用するには、 ゲストの **/proc/modules** ファイルと **/proc/kallsyms** ファイルにアクセスできなければなりません。 これを行なうための方法が 2 種類あります。 ホストにファイルを転送してからそのファイルで報告を実行する場合は、 手順8.1「ゲストの /proc ファイルをホストにコピーする」 を参照してください。 ゲストを直接マウントしてそのファイルにアクセスする場合は、 手順8.2「代替方法: sshfs を使ってファイルに直接アクセスする」 を参照してください。

手順8.1 ゲストの /proc ファイルをホストにコピーする

重要

必要なファイルを /proc ディレクトリから直接コピーしても (**scp** などを使用)、 コピーしたファイルは空になります。 本セクションでは、 まず、 ゲストのファイルを一時的な場所に保存し (**cat**)、 その保存先からファイルをホストにコピーして、 **perf kvm** でのイベントの記録や報告に使用する手順を説明しています。

1. **ゲストにログインしてファイルを保存する**
   ゲストにログインして、 **/proc/modules** と **/proc/kallsyms** を一時的な場所となる **/tmp** に保存します。

   ```
   # cat /proc/modules > /tmp/modules
   # cat /proc/kallsyms > /tmp/kallsyms
   ```

2. **一時ファイルをホストにコピーする**
   ゲストからログオフしたら、 今度は次の例に示す **scp** コマンドを実行して一時的な場所に保存したファイルをホストにコピーします。 必要に応じてホスト名と TCP ポートをご使用の値に置き換えてください。

   ```
   # scp root@GuestMachine:/tmp/kallsyms guest-kallsyms
   # scp root@GuestMachine:/tmp/modules guest-modules
   ```

   これでゲストからの 2 つのファイル (**guest-kallsyms** と **guest-modules**) がホスト上にコピーされ、 **perf kvm** で使用する準備が整いました。

3. **perf kvm でイベントの記録と報告を行なう**
前述の手順で入手したファイルを使って、 ゲスト内のイベント、 ホスト内のイベントのいずれかまたは両方を記録し報告を行なうことができるようになりました。

次のコマンドの例を実行します。

```
# perf kvm --host --guest --guestkallsyms=guest-kallsyms \
--guestmodules=guest-modules record -a -o perf.data
```

注記

--host と **--guest** の両方をコマンドに使用すると、 出力は **perf.data.kvm** というファイル名で保存されます。 **--host** だけを使用すると、 ファイル名は **perf.data.host** になります。 同じように、 **--guest** だけを使用すればそのファイル名は **perf.data.guest** になります。

記録の動作を停止させる場合は Ctrl-C を押します。

4. **イベントを報告する**
記録のプロセスで取得したファイルを使って出力を新しいファイル「**analyze**」にリダイレクトする例を示します。

```
perf kvm --host --guest --guestmodules=guest-modules report -i
perf.data.kvm \
--force > analyze
```

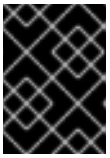
記録したイベントを調べるため、 **analyze** ファイルの内容を表示させます。

```
# cat analyze


# Events: 7K cycles
#
# Overhead       Command  Shared Object      Symbol
# ........  ............  ..................
.........................
#
    95.06%            vi  vi                 [.] 0x48287
     0.61%          init  [kernel.kallsyms]  [k] intel_idle
     0.36%            vi  libc-2.12.so       [.]
_wordcopy_fwd_aligned
     0.32%            vi  libc-2.12.so       [.] __strlen_sse42
     0.14%       swapper  [kernel.kallsyms]  [k] intel_idle
     0.13%          init  [kernel.kallsyms]  [k] uhci_irq
     0.11%          perf  [kernel.kallsyms]  [k]
generic_exec_single
     0.11%          init  [kernel.kallsyms]  [k] tg_shares_up
     0.10%      qemu-kvm  [kernel.kallsyms]  [k] tg_shares_up

[output truncated...]
```

手順8.2 代替方法: sshfs を使ってファイルに直接アクセスする

- 

  重要

  以下に示すのは一例です。 使用環境に合わせて適した値に置き換えてください。

```
# Get the PID of the qemu process for the guest:
PID=`ps -eo pid,cmd | grep "qemu.*-name GuestMachine" \
| grep -v grep | awk '{print $1}'`

# Create mount point and mount guest
mkdir -p /tmp/guestmount/$PID
sshfs -o allow_other,direct_io GuestMachine:/ /tmp/guestmount/$PID

# Begin recording
perf kvm --host --guest --guestmount=/tmp/guestmount \
record -a -o perf.data

# Ctrl-C interrupts recording. Run report:
perf kvm --host --guest --guestmount=/tmp/guestmount report \
-i perf.data

# Unmount sshfs to the guest once finished:
fusermount -u /tmp/guestmount
```

付録A 改訂履歴

改訂 0.3-48.2.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 0.3-48.2 **Thu Jun 27 2013** **Noriko Mizumoto**
翻訳および査読完了

改訂 0.3-48.1 **Wed Jun 5 2013** **Chester Cheng**
翻訳ファイルを XML ソースバージョン 0.3-48 と同期

改訂 0.3-48 **Mon Feb 18 2013** **Scott Radvan**
6.4 リリース向けバージョン

改訂 0.3-47 **Sun Feb 17 2013** **Scott Radvan**
表現を若干変更

改訂 0.3-46 **Sun Feb 17 2013** **Scott Radvan**
SME からのフィードバックを反映

改訂 0.3-45 **Wed Feb 13 2013** **Scott Radvan**
表現を若干変更

改訂 0.3-44 **Wed Feb 13 2013** **Scott Radvan**
キャッシングモードに関する記載について SME からのフィードバックを反映

改訂 0.3-43 **Tue Feb 12 2013** **Scott Radvan**
SME からのフィードバックを反映、 全体的な用語の使用方法について

改訂 0.3-42 **Mon Feb 11 2013** **Scott Radvan**
ドラフト版のステータスを削除

改訂 0.3-41 **Mon Feb 11 2013** **Scott Radvan**
SME からのフィードバックによる変更、 警告の追加、 virt-manager の CPU pinning オプションの表現を変更、 存在しない virsh コマンドへの参照を削除

改訂 0.3-40 **Fri Feb 8 2013** **Scott Radvan**
マイナーな表現の問題

改訂 0.3-39 **Fri Feb 8 2013** **Scott Radvan**
<screen> タグのインデントを修正

改訂 0.3-38 **Fri Feb 8 2013** **Scott Radvan**
ビルド関連のエラーを修正 (BZ#908666)

改訂 0.3-37 **Fri Feb 8 2013** **Scott Radvan**
libvirt の NUMA 調整のセクション

改訂 0.3-36 **Thu Feb 7 2013** **Scott Radvan**
パブリッシングツールチェーンの新バージョンで再ビルドを行ない警告の CSS エラーを修正

改訂 0.3-35 **Thu Feb 7 2013** **Scott Radvan**
各章の冒頭に概要を追記し、レイアウトを整備、 不足していたパラメータを含ませ vcpu の事例を改善

改訂 0.3-34 **Mon Feb 4 2013** **Scott Radvan**
全 PNG に対して縮尺比を合わせ幅パラメータを追加

改訂 0.3-33 **Mon Feb 4 2013** **Scott Radvan**
virt-manager のスクリーンショットとオプションの説明

改訂 0.3-32 **Thu Jan 31 2013** **Scott Radvan**
NUMA cpuset に関する SME からのフィードバックを追加、 開発者の意見を追加

改訂 0.3-31 **Thu Jan 31 2013** **Scott Radvan**
ガイド全体で SR/IOV を SR-IOV に修正

改訂 0.3-30 **Wed Jan 30 2013** **Scott Radvan**
「モード」の表現を「ポリシー」に変更 (NUMA の章)

改訂 0.3-29 **Wed Jan 30 2013** **Scott Radvan**
NUMA メモリーモードの修正: BZ#854099

改訂 0.3-28 **Tue Jan 29 2013** **Scott Radvan**
QE からのフィードバックによる修正、 #754935

改訂 0.3-27 **Tue Jan 22 2013** **Scott Radvan**
Memory.xml 内、 huge pages に関する記載の表現を修正

改訂 0.3-26 **Mon Jan 21 2013** **Scott Radvan**
CPU セクションを削除、 CPU pinning については NUMA のセクションで説明

改訂 0.3-25 **Mon Jan 14 2013** **Scott Radvan**
カーネルの章を削除、 著作権年度を 2013 に更新

改訂 0.3-24 **Mon Jan 14 2013** **Scott Radvan**
SME からのフィードバックをさらに追加、 ネットワークオプションおよび SR-IOV について

改訂 0.3-23 **Thu Jan 3 2013** **Scott Radvan**
パブリッシング関連の問題を修正

改訂 0.3-22 **Thu Jan 3 2013** **Scott Radvan**
SME からのフィードバックを追加、 numactl と numatune mpmp のノードセットについて

改訂 0.3-21 **Fri Dec 14 2012** **Scott Radvan**
SME からのフィードバックを追加、 hugetlbfs マウント、 numatune メモリーモードについて

改訂 0.3-20 **Thu Dec 06 2012** **Scott Radvan**
SME からのフィードバックを追加、 vcpu pinning について

改訂 0.3-19 **Mon Nov 12 2012** **Scott Radvan**
I/O 要件、 ゲスト数に関するキャッシングのオプションを記載

改訂 0.3-18 **Mon Oct 29 2012** **Scott Radvan**
検証エラーを修正

改訂 0.3-17 **Mon Oct 29 2012** **Scott Radvan**
ネスト化した一覧内のカーネルオプション

改訂 0.3-16 **Tue Oct 16 2012** **Scott Radvan**
誤字修正

改訂 0.3-15 **Mon Oct 15 2012** **Scott Radvan**
全体的に見出しを大文字に変更

改訂 0.3-14 **Sun Oct 14 2012** **Scott Radvan**

tuned セクションを追加し、 tuned-adm コマンドについて記載

改訂 0.3-13 **Tue Oct 2 2012** **Scott Radvan**
インフラストラクチャの変更、 誤字修正

改訂 0.3-12 **Tue Oct 2 2012** **Scott Radvan**
NUMA の要約とメモリーポリシーに関する記載を追加

改訂 0.3-11 **Tue Oct 2 2012** **Scott Radvan**
tuned-adm のプロファイルを拡張

改訂 0.3-10 **Tue Oct 2 2012** **Scott Radvan**
tuned-adm の表を追加 (後日、削除)

改訂 0.3-9 **Tue Oct 2 2012** **Scott Radvan**
キャッシングに関する表、 一般的なネットワーク関連のヒントを追加

改訂 0.3-8 **Thu Sep 27 2012** **Scott Radvan**
代わりに KVM の概要およびネットワーク構築のイメージを追加、 「他の参考文献」のセクションを追加

改訂 0.3-7 **Mon Sep 24 2012** **Scott Radvan**
誤字修正

改訂 0.3-6 **Wed Sep 19 2012** **Scott Radvan**
改行を入れて長いコマンドが正しく表示されるよう変更

改訂 0.3-5 **Tue Sep 18 2012** **Scott Radvan**
perf kvm の章を追加、 手順を記載

改訂 0.3-4 **Wed Sep 12 2012** **Scott Radvan**
各章を具体的に肉付け、「パフォーマンス監視ツール」の章を追加

改訂 0.3-3 **Wed Sep 12 2012** **Scott Radvan**
virt-manager の章を開始、 キャプチャした画面を追加

改訂 0.3-2 **Wed Sep 12 2012** **Scott Radvan**
ドラフト版

改訂 0.3-1 **Wed Sep 12 2012** **Scott Radvan**
ガイドをレイアウトし、 基本となるインフラストラクチャ設定と ID